THESIS

LAPLACIAN EIGENMAPS FOR TIME SERIES ANALYSIS

Submitted by

Patrick J. Rosse

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2020

Master's Committee:

Advisor: Michael Kirby

Chris Peterson
Henry Adams
Chuck Anderson

ABSTRACT

LAPLACIAN EIGENMAPS FOR TIME SERIES ANALYSIS

With "Big Data" becoming more available in our day-to-day lives, it becomes necessary to make meaning of it. We seek to understand the structure of high-dimensional data that we are unable to easily plot. What shape is it? What points are "related" to each other? The primary goal is to simplify our understanding of the data both numerically and visually. First introduced by M. Belkin, and P. Niyogi in 2002, Laplacian Eigenmaps (LE) is a non-linear dimensional reduction tool that relies on the basic assumption that the raw data lies in a low-dimensional manifold in a high-dimensional space. Once constructed, the graph Laplacian is used to compute a low-dimensional representation of the data set that optimally preserves local neighborhood information. In this thesis, we present a detailed analysis of the method, the optimization problem it solves, and we put it to work on various time series data sets. We show that we are able to extract neighborhood features from a collection of time series, which allows us to cluster specific time series based on noticeable signatures within the raw data.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Michael Kirby for supporting me when other people would not. I would also like to thank Dr. Manuchehr Aminian for being my "IT guy", mentor, and friend throughout my projects in graduate school. I would like to thank my parents and brother for their lifelong support. Last but not least, I would like to thank my fiancé, Danielle Fennell for keeping me sane throughout this process.

This work is based on research partially supported by the National Science Foundation under grant No. DMS-1513633 and DARPA contracts N66001-17-2-4020 and D17AP00004. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or DARPA.

DEDICATION

I would like to dedicate this thesis to my dog, Luna.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Motivation

Data Visualization is the graphical representation of information. More often than not, the data that we wish to visualize has too many dimensions to plot on a three, or two-dimensional plot. The most naive approach to data reduction and visualization is to simply choose 2 or 3 features from a set of data and use those to plot the data points on a 2 or 3-dimensional Cartesian plane. Unless those few features chosen are highly distinguishing features of the data set, we lose much of the information, structure, and shape when we visualize this low-dimensional plot. Fortunately for us, there exists a handful of algorithms with accompanying modifications that allow us to visualize this high-dimensional data with more purpose.

The Curse of Dimensionality [1] also plays a role in the necessity for dimensional reduction. One aspect of The Curse is that when dimensionality increases, the volume of the space increases so fast that the available data becomes sparse. Because of this, given a high-dimensional data set, there will likely exist many dissimilarities between data points which would normally be considered similar in their respective domain. This vast growth of the space also prevents common data clustering strategies from being efficient. We would like to be able to reduce the dimension of the data points without losing too much information about the data set as a whole.

More specifically, given a data matrix

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{m \times n}, \tag{1.1}$$

we wish to find a low-dimensional analogue

$$Y = \begin{bmatrix} | & | & & | \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{d \times n} \tag{1.2}$$

such that $d \ll m$, and Y is a faithful representation of X in some sense. In general, not all information will be preserved when reducing dimension, but as stated before, we have at our disposal a handful of reduction methods that retain specific properties of the data set. Examples of properties to be preserved may include the global geometry, neighborhood information such as local neighborhoods and local tangent spaces, distances between data points, or angles formed by adjacent line segments [2].

1.1.1 Data Reduction

Data reduction can be separated into two groups: Linear and nonlinear. Linear methods are *projective* in nature and are implemented by finding an explicit linear map $V \in \mathbb{R}^{m \times d}$ such that $Y = V^T X$ is a reduced-dimension representation of X. Principal Component Analysis (PCA) is one of the most commonly implemented *linear* dimensional reduction techniques, and has been around for quite some time [3]. Although, over recent years, we've been introduced to highly nonlinear dimensional reduction techniques such as ISOMAP [4], Locally Linear Embedding (LLE) [5], Hessian Eigenmaps [6], and Laplacian Eigenmaps [7]. The primary goal of *nonlinear* dimensional reduction, specifically, is to take a high-dimensional data set potentially living on a low-dimensional manifold and provide a representation of that data in a low-dimensional space [7].

LLE is the closest related method to LE, as their corresponding objective functions are very similar. They both also rely on the assumption that there is sufficient data (such that the manifold is well-sampled) in order to provide an accurate embedding. In LLE, we expect each data point and its neighbors to lie on or close to a locally linear patch of the manifold [8]. On the other hand, instead of reproducing small linear patches around each data point, LE relies on graph-theoretic concepts like the Laplacian operator on a graph [9].

In this paper, we explain the algorithm of Laplacian Eigenmaps along with a few modifications. The algorithm itself is fairly rudimentary, and can be implemented with very few lines of code. However, the theory behind it is rich and meaningful, with subtle nuances that we discuss along the way. We break down and give a thorough analysis of the optimization problem that LE aims to solve. We apply LE to synthetic/toy data to acquire an understanding of how time series data embeds to lower dimensions. Then we use this understanding to analyze the embeddings of mouse temperature time series for the duration of dealing with a salmonella inoculation.

1.2 Mathematical Framework

In an attempt to build some mathematical framework, we introduce some of the terminology and notation used in the following chapters. Capital letters (i.e., X, D, L) are used to denote matrices, while bold-faced lower case letters (i.e., $\mathbf{f}, \mathbf{x}$) are used to denote column vectors. In terms of indexing, X_{ij} denotes the $(i,j)^{th}$ component of matrix X, while $\mathbf{x}_i$ denotes the i^{th} component of vector $\mathbf{x}$. Additionally, $\mathbf{x}^{(i)}$ denotes the i^{th} $\mathbf{x}$ vector in an ordered collection of vectors.

The method of Laplacian Eigenmaps operates under the assumption that our data sets can be modeled by a graph $G = (N, E)$ with N nodes and E edges. This allows us to leverage many properties of graph theory and generate the embeddings described in this paper. Below is a list of definitions and ideas typically used in graph theory that we will be using in our derivation of LE.

- A graph $G = (N, E)$ is a collection of nodes $n \in N$ and edges $e \in E$. In our case, each node n corresponds to a point within a data set. And two nodes may or may not be connected with an edge e.
- The degree of a node corresponds to how many edges are attached to it.
- Each edge can be assigned a *weight* to it corresponding to the level of connection between the two nodes.
- A directed graph has directions attached to each edge. In our case, we will only be dealing with *undirected* graphs.

- A graph $G = (N, E)$ is *connected* if it is possible to get from every node in the graph to every other node through a series of edges, called a path. Otherwise, G is disconnected and has more than 1 *connected components*.

- A graph G has m *connected components* if there are m subgraphs (disconnected from one another) within G that are *connected*.

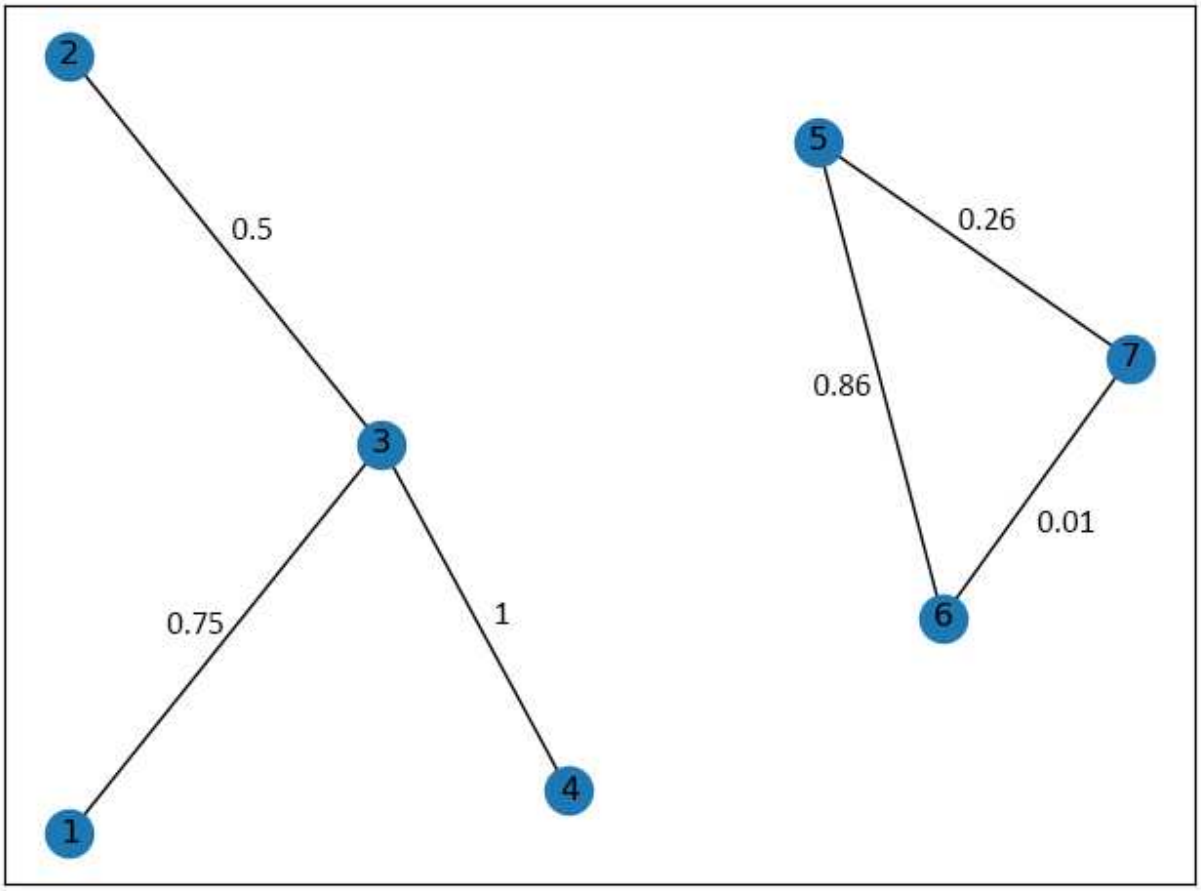


Figure 1.1: An example of an undirected graph $G = (7, 6)$. G is disconnected and has 2 connected components. The degree of node 6 is 2, and the edge connecting node 1 to node 3 has weight 0.75.

Chapter 2

Laplacian Eigenmaps

2.1 The Algorithm

Much of the Laplacian Eigenmaps algorithm for dimensional reduction falls out of the derivation from Section 2.2, albeit with a few optional modifications. Under the assumption that the data set has been pre-processed/curated, an in-depth explanation of the algorithm can be found below, with justification in the following section. (This section comes from the exposition in [7].)

1. Construct distance matrix from raw data values or from a "pairwise similarities" matrix. For two points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$, the distance between the two is commonly calculated by the Euclidean metric $d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = ||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||^2$. This value populates the $(i, j)^{th}$ component of the distance matrix. Although the use of Euclidean distances is common, various metrics can be used depending on their relevance to the data set.

2. Use the pairwise distances to construct an adjacency graph $G = (N, E)$. We put an edge between nodes i and j if $\mathbf{x}^{(i)}$ is "close" to $\mathbf{x}^{(j)}$. There are two possible variations to capture the "closeness" between points:

   (a) ϵ-ball: Choose parameter $\epsilon \in \mathbb{R}^+$. Connect nodes i and j with an edge if $d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) < \epsilon$.

   (b) k-nearest neighbors: Choose parameter $k \in \mathbb{N}$. Connect nodes i and j with an edge if $\mathbf{x}^{(i)}$ is one of the k-nearest neighbors of $\mathbf{x}^{(j)}$ **or** $\mathbf{x}^{(j)}$ is one of the k-nearest neighbors of $\mathbf{x}^{(j)}$.

   Note that both of the above options for generating edges on our graph induce the notion of symmetry. One advantage to the ϵ-ball method is that it is geometrically motivated [7], although choosing the best ϵ can be very difficult. If ϵ is too small, our graph ends up with more than one connected component, which dilutes the accuracy and inhibits the efficiency

of the low-dimensional embeddings. If ϵ is too large, our graph ends up with too many "shortcut edges", which can also corrupt the low-dimensional embeddings. Outliers (i.e., points with no neighbors) will likely never be connected using this method.

Taking the k-nearest neighbors approach is inherently less geometric, but more robust to outliers. (e.g., Let $\mathbf{x}^{(j)}$ be an outlier. So for all $i \neq j$, $x^{(i)}$ does not have $\mathbf{x}^{(j)}$ as a nearest neighbor. However, $\mathbf{x}^{(j)}$ will still have k neighbors, regardless of how far away they are. Therefore, node j *must* be connected to k other nodes.) The large distance of the outlier is an issue that can be handled in the following step of the algorithm.

3. Assign weights to each of the edges generated in the above step. Again, there are two possible variations:

   (a) Heat Kernel: Choose parameter t. If there is an edge connecting nodes i, and j, then the weight of that edge is set to $W_{ij} = \exp\left(-\frac{d(x^{(i)}, x^{(j)})^2}{t}\right)$. Else, $W_{ij} = 0$. If $x^{(i)}$ and $x^{(j)}$ are close *and* node i is connected to node j, then W_{ij} will be close to 1. If $x^{(i)}$ and $x^{(j)}$ are far *and* node i is connected to node j, then W_{ij} will be closer to zero.

   (b) Simple-Minded: Set $t = \infty$. If there is an edge connecting nodes i, and j, then $W_{ij} = 1$. Else, $W_{ij} = 0$. This eliminates the need to tune parameter t. However, the distant neighbors in high-dimensional space may not maintain their relative distance in the embedding due to the loss of encoding of that information.

4. If the graph constructed above is not connected, repeat this step for each connected component (i.e., for each block of the Laplacian matrix L). Compute the eigenvalues and eigenvectors for the generalized eigenvalue problem,

$$L\mathbf{f} = \lambda D\mathbf{f}, \tag{2.1}$$

where D is a diagonal weight matrix such that $D_{ii} = \sum_{j=1}^{n} W_{ij}$, and $L = D - W$ is the Graph Laplacian. For the embedding, we order the eigenvalues from least to greatest, and throw out

the first vector $\mathbf{f}^{(0)}$ corresponding to the eigenvalue $\lambda = 0$. To embed into d dimensions, we pick the next d eigenvectors corresponding to the next d nonzero eigenvalues in ascending order, $\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \ldots, \mathbf{f}^{(d)}$. (This step is justified in Section 2.2.3.) In practice, d is typically chosen to be 2 or 3.

2.1.1 Analogous Algorithm

Note that an equivalent approach [10] to obtain an m-dimensional embedding (up to a componentwise scaling) consists of normalizing the Laplacian matrix

$$L' = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \qquad \text{where} \qquad L'_{ij} = \frac{L_{ij}}{\sqrt{D_{ii} D_{jj}}}, \tag{2.2}$$

and finding directly its eigenvectors $L' = U \Lambda U^T$. The eigenvectors associated with the first d smallest nonzero eigenvalues form an d-dimensional embedding of the data set. The eigenvalues are the same as for the generalized eigenvalue problem, and the eigenvectors are scaled by the values on the diagonal of $D^{\frac{1}{2}}$. As vectors, we have $\mathbf{u}^{(i)} = D^{\frac{1}{2}} \mathbf{f}^{(i)}$, and in matrix notation, $U = D^{\frac{1}{2}} Y^T$. Using the fact that the matrix of eigenvalues U of a symmetric matrix L is unitary (i.e., $U^{-1} = U^T$), justification is below:

$$L' = U \Lambda U^T \tag{2.3}$$

$$L' U = U \Lambda \tag{2.4}$$

$$D^{-\frac{1}{2}} L D^{-\frac{1}{2}} U = \Lambda U \tag{2.5}$$

$$L D^{-\frac{1}{2}} U = D^{\frac{1}{2}} \Lambda U \tag{2.6}$$

$$L D^{-\frac{1}{2}} D^{\frac{1}{2}} Y^T = \Lambda D^{\frac{1}{2}} D^{\frac{1}{2}} Y^T \tag{2.7}$$

$$L Y^T = \Lambda D Y^T \tag{2.8}$$

$$\implies L \mathbf{f}^{(i)} = \lambda D \mathbf{f}^{(i)} \tag{2.9}$$

2.1.2 Drawbacks

It should be noted that as we have currently described it, Laplacian Eigenmaps does not possess the capability to embed out-of-sample points. If an embedding has been learned, and we are given new, high-dimensional points for the data set, we are required to construct a new graph Laplacian and recalculate the eigenvectors. With enough data points, this can be a very expensive operation. Embeddings are unique up to rotation and scaling, so an embedding with just one new point could be potentially very difficult to compare to the original embedding. However, techniques based on reproducing kernel Hilbert space regularization exist and have been shown to work quite well given a set of out-of-sample points to a learned embedding [10]. These techniques also work on related dimensional reduction methods such as MDS, Isomap, and LLE.

2.2 The Optimization Problem

We seek to minimize the objective function

$$E_{\mathcal{LE}} = \frac{1}{2} \sum_{i,j=1}^{N} ||\mathbf{y}^{(i)} - \mathbf{y}^{(j)}||_2^2 W_{ij} \tag{2.10}$$

Recall that each $\mathbf{y}^{(i)}$ is an embedded data point from its respective $\mathbf{x}^{(i)}$. (i.e., $\mathcal{LE}(\mathbf{x}^{(i)}) = \mathbf{y}^{(i)}$). Prior to optimizing, each $\mathbf{y}^{(i)}$ is unknown, but the weights W_{ij} are given. These weights are only greater than zero when $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are "close" or nodes i and j are "connected". Else, they are set to zero. Thus, (2.10) incurs a heavy penalty if $\mathbf{x}^{(i)}$ is "close" to $\mathbf{x}^{(j)}$, but $\mathbf{y}^{(i)}$ is *not* "close" to $\mathbf{y}^{(j)}$.

Let $\mathbf{f}^{(p)}$ be an N-dimensional vector giving the p'th coordinate of each of the embedded points. We use $\mathbf{f}^{(p)}$ to build a matrix Y such that $\mathbf{f}^{(p)}$ is the transpose of the pth row of Y. So Y can be thought of as either of the two following forms:

$$Y = \begin{bmatrix} \text{—}\mathbf{f}^{(1)^T}\text{—} \\ \text{—}\mathbf{f}^{(2)^T}\text{—} \\ \vdots \\ \text{—}\mathbf{f}^{(n)^T}\text{—} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(n)} \\ | & | & & | \end{bmatrix}.$$

We can now manipulate (2.10) as follows:

$$E_{\mathcal{LE}} = \frac{1}{2}\sum_{i,j=1}^{N} ||\mathbf{y}^{(i)} - \mathbf{y}^{(j)}||_2^2 W_{ij} \tag{2.11}$$

$$= \frac{1}{2}\sum_{p=1}^{P}\sum_{i,j=1}^{N} (\mathbf{y}_p^{(i)} - \mathbf{y}_p^{(j)})^2 W_{ij} \tag{2.12}$$

$$= \frac{1}{2}\sum_{p=1}^{P}\sum_{i,j=1}^{N} ((\mathbf{y}_p^{(i)})^2 - (\mathbf{y}_p^{(j)})^2 - 2\mathbf{y}_p^{(i)}\mathbf{y}_p^{(j)}) W_{ij} \tag{2.13}$$

$$= \frac{1}{2}\sum_{p=1}^{P}\left(\sum_{i=1}^{N}(\mathbf{y}_p^{(i)})^2 D_{ii} + \sum_{j=1}^{N}(\mathbf{y}_p^{(i)})^2 D_{jj} - 2\sum_{i,j=1}^{N}\mathbf{y}_p^{(i)}\mathbf{y}_p^{(j)} W_{ij}\right) \tag{2.14}$$

$$= \frac{1}{2}\sum_{p=1}^{P} 2\mathbf{f}^{(p)^T} D\mathbf{f}^{(p)} - 2\mathbf{f}^{(p)^T} W\mathbf{f}^{(p)} \tag{2.15}$$

$$= \sum_{p=1}^{P} \mathbf{f}^{(p)^T} D\mathbf{f}^{(p)} - \mathbf{f}^{(p)^T} W\mathbf{f}^{(p)} \tag{2.16}$$

$$= \sum_{p=1}^{P} \mathbf{f}^{(p)^T} (D - W)\mathbf{f}^{(p)} \tag{2.17}$$

$$= \sum_{p=1}^{P} \mathbf{f}^{(p)^T} L\mathbf{f}^{(p)} \tag{2.18}$$

$$= \mathrm{Tr}(YLY^T) \tag{2.19}$$

Note that the above derivation shows that L is positive semi-definite. Equation (2.12) shows that we are summing over non-negative values. Thus, $\mathbf{f}^{(p)^T} L\mathbf{f}^{(p)} \geq 0$ for all $\mathbf{f}^{(p)}$.

Now, (2.10) reduces to solving

$$\min_{\substack{Y \\ YDY^T=I}} \operatorname{Tr}(YLY^T). \tag{2.20}$$

2.2.1 The Constraint

For the problem of embedding into d-dimensional space, this optimization is constrained by $YDY^T = I$ in order to prevent collapsing onto a subspace of dimension less than $d-1$. This helps us avoid trivial mappings such as

$$\mathcal{LE}(\mathbf{x}^{(i)}) = \mathbf{y} \text{ for } i = 1, \ldots, n. \tag{2.21}$$

Sending each high-dimensional data point to the same low-dimensional data point would certainly minimize (2.10) with any choice of weights, but that solution is uninteresting and impractical. More intuition and justification for this constraint can be acquired in the following toy problem.

Toy Problem

$$YDY^T = I \implies {\mathbf{f}^{(i)}}^T D\mathbf{f}^{(j)} = \delta_{ij} \tag{2.22}$$

Where δ_{ij} is the *Kronecker delta* and is defined as

$$\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}. \tag{2.23}$$

For our toy problem, consider the coordinate vector corresponding to the first coordinates of an embedding $\mathbf{f}^{(1)}$. And let D be the degree matrix such that

$$D = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.24}$$

Notice that $D_{11} = 3$. This means that node 1 has 3 edges connected to it, which is greater than the degree of each of the other nodes. Hence, the data point that is represented by said node has the most "relation" to each of the other data points in this small, toy data set. Now, we look at the constraint $\mathbf{f}^{(1)^T} D\mathbf{f}^{(1)} = 1$. In matrix form, we have

$$\begin{bmatrix} \text{—} \mathbf{f}^{(1)^T} \text{—} \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} | \\ \mathbf{f}^{(1)} \\ | \end{bmatrix} = 1, \tag{2.25}$$

and element-wise, we have

$$3(\mathbf{f}_1^{(1)})^2 + 2(\mathbf{f}_2^{(1)})^2 + 1(\mathbf{f}_3^{(1)})^2 + 1(\mathbf{f}_4^{(1)})^2 = 1. \tag{2.26}$$

In order to satisfy the equality in (2.26), it can be seen that $\mathbf{f}_1^{(1)}$ *should* be less than $\mathbf{f}_i^{(1)}$ for $i \in [2, 3, 4]$. This value corresponds to the first coordinate of the first data point in the embedding. Thus, the first embedded point should be more central (closer to the origin) than each of the other points, which correctly captures the neighborhood property of that popular data point. Certainly, $\mathbf{f}_1^{(1)}$ doesn't necessarily have to be less than the other components of $\mathbf{f}^{(1)}$. But in regards to a statement about intuition, we claim that since each of the coordinate vectors are constrained by this weighted inner product, they seem to work together to create a more centralized embedded point corresponding to the popular high-dimensional data point.

With the same toy problem here, we analyze the other part of the constraint $\mathbf{f}_i^T D\mathbf{f}_j = 0$. This is an orthogonality constraint on the coordinate vectors with respect to the weighted inner product $\langle \cdot, \cdot \rangle_D$. This is the part of the constraint that prevents a collapse of the embedding onto a subspace of dimension less than $d - 1$, which was stated above. It is slightly less geometrically intuitive, but if we just considered the traditional inner product without matrix D (i.e., $\mathbf{f}^{(1)^T}\mathbf{f}^{(2)} = 0$), we would have that each embedding coordinate must be orthogonal to the others. This gives our embedding

the structure we desire, and allows us to plot on the Cartesian plane with orthogonal axes. Now, considering the weighted inner product $\langle \cdot, \cdot \rangle_D$, the diagonal matrix D is again designed to mitigate the effects of an imbalanced graph. D can be thought of as the identity matrix I in a changed basis, so the coordinate vectors are, in fact, orthogonal within that basis, and therefore not collapsed onto one another.

2.2.2 Lagrange Multipliers

Equation (2.20) is a constrained optimization problem with equality constraints. Thus, we use the method of Lagrange Multipliers to minimize it. Because our objective function has matrix constraints, the Lagrangian for (2.20) is

$$\mathcal{L}(Y, \Lambda) = \mathrm{Tr}(YLY^T) - \langle \Lambda, (YDY^T - I) \rangle_F \tag{2.27}$$

$$= \mathrm{Tr}(YLY^T) - \sum_{i,j=1}^{N} ((\Lambda^T)_{ij}(YDY^T - I)_{ij}) \tag{2.28}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius Inner Product, and Λ is a matrix whose entries are the Lagrange Multipliers. In general, an optimization problem with matrix constraints of size $n \times n$ yields n^2 individual constraints, each accompanied by a value Λ_{ij}. Further, with *symmetric* matrix constraints, we can reduce to just $\frac{n(n+1)}{2}$ values of Λ_{ij}, since $\Lambda_{ij} = \Lambda_{ji}$. However, we claim that the Λ here is a diagonal matrix, which reduces the number of Lagrange Multipliers to just n. With the help of the Courant-Fischer characterization, the claim is justified below.

Theorem 2.2.1 (Courant-Fischer Formula)**.** Let A be an $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. Then

$$\lambda_1 = \min_{||\mathbf{x}||=1} \mathbf{x}^T A \mathbf{x} = \min_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

$$\lambda_2 = \min_{\substack{||\mathbf{x}||=1 \\ \mathbf{x} \perp \mathbf{v}_1}} \mathbf{x}^T A \mathbf{x} = \min_{\substack{\mathbf{x} \neq 0 \\ \mathbf{x} \perp \mathbf{v}_1}} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

$$\lambda_n = \lambda_{\max} = \max_{||\mathbf{x}||=1} \mathbf{x}^T A \mathbf{x} = \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

In general, for $1 \leq k \leq n$, let S_k denote the span of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ (with $S_0 = \{0\}$), and let $S_k^{\perp}$ denote the orthogonal complement of S_k. Then

$$\lambda_k = \min_{\substack{||\mathbf{x}||=1 \\ \mathbf{x} \in S_{k-1}^{\perp}}} \mathbf{x}^T A \mathbf{x} = \min_{\substack{\mathbf{x} \neq 0 \\ \mathbf{x} \in S_{k-1}^{\perp}}} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

In our case take $A = L'$, the normalized laplacian from Section 2.1.1. Then as a consequence of Theorem 2.2.1, the trace of YLY^T is minimized when Y^T is an orthogonal basis of the eigenspace associated with the (algebraically) smallest eigenvalues of L'. In particular, it is achieved with the eigenbasis itself [2]: If the eigenvalues are labeled in increasing value, $\lambda_1, \lambda_2, \ldots, \lambda_d$, with corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d$, we can build matrix U such that $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d]$ and $UU^T = I$. Then we have

$$\min_{\substack{Y^T \in \mathbb{R}^{n \times d} \\ YY^T = I}} \mathrm{Tr}(YLY^T) = \mathrm{Tr}(ULU^T) = \lambda_1 + \lambda_2 + \cdots + \lambda_d \tag{2.29}$$

At most, we have $d = n$ eigenvalues with n corresponding eigenvectors. So this satisfies the claim that Λ is diagonal, consisting of the n eigenvalues of L (or the n Lagrange Multipliers of $\mathcal{L}(Y, \Lambda)$.)

Using this information, we can further manipulate the Lagrangian into a form that has a nicer looking derivative.

$$\mathcal{L}(Y, \Lambda) = \mathrm{Tr}(YLY^T) - \langle \Lambda, (YDY^T - I) \rangle_F \tag{2.30}$$

$$= \mathrm{Tr}(YLY^T) - \sum_{i,j=1}^{N} ((\Lambda^T)_{ij} (YDY^T - I)_{ij}) \tag{2.31}$$

$$= \mathrm{Tr}(YLY^T) - \mathrm{Tr}(\Lambda^T (YDY^T - I)) \tag{2.32}$$

$$= \mathrm{Tr}(YLY^T) - \mathrm{Tr}(\Lambda (YDY^T - I)) \tag{2.33}$$

We take the derivative of $\mathcal{L}$ with respect to Y and set it equal to zero to get:

$$\nabla_Y \mathcal{L} = \frac{\partial}{\partial Y}\left(\mathrm{Tr}(YLY^T) - \mathrm{Tr}(\Lambda(YDY^T - I))\right) \tag{2.34}$$

$$= (YL + YL^T) - (\Lambda(YD + YD^T)) \tag{2.35}$$

$$= 2YL - 2YD\Lambda \stackrel{\text{set}}{=} 0 \tag{2.36}$$

$$\implies YL = YD\Lambda \tag{2.37}$$

$$\implies LY^T = DY^T\Lambda \tag{2.38}$$

Here, we are using the fact that L and D are symmetric, and that $\nabla_A \mathrm{Tr}(ABA^TC) = CAB + C^TAB^T$. In our case, $C = I$.

This derivation leads to the generalized eigenvalue problem,

$$LY^T = DY^T\Lambda \tag{2.39}$$

$$L\mathbf{f}^{(p)} = \lambda_p D\mathbf{f}^{(p)} \qquad \text{for } p = 0, \ldots, n-1 \tag{2.40}$$

Recall that L is real, symmetric, and positive semi-definite. So respectively, L has real eigenvalues and they are non-negative. As described by Ghojogh, B., et al., in [11], if Equation (2.20) was a *maximization* problem, the eigenvectors and eigenvalues are sorted from largest to smallest eigenvalue. However, since Equation (2.20) is a *minimization* problem, the eigenvectors and eigenvalues are sorted from smallest to largest eigenvalue [11].

2.2.3 A Trivial Solution

At this point, one might notice that (2.40) possesses a trivial solution. Consider applying the linear transformation L to the $n \times 1$ vector filled with ones. Then we have,

$$L \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = (D - W) \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = 0 \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = 0 \cdot D \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{2.41}$$

So $\lambda = 0$ is an eigenvalue with corresponding eigenvector of ones. In fact, the graph G created to model our data set is *connected* if the algebraic multiplicity of $\lambda = 0$ is exactly 1 (See [12] for justification.) If the algebraic multiplicity of $\lambda = 0$ is greater than 1, then we can conclude that G is *not* connected, and thus, has more than one connected component. In fact, G has k connected components if and only if the algebraic multiplicity of $\lambda = 0$ is k. Below is a proof of the statement.

Proof. Assume a graph G has k connected components. Then it is possible to write the adjacency matrix of G as block diagonal with k blocks. In turn, the Laplacian matrix is block diagonal with k blocks, denoted $L_1, L_2, \ldots, L_k$. Each L_i for $i = 1, \ldots, k$ is a Laplacian matrix for connected component i. As we have shown above, the Laplacian matrix has eigenvalue $\lambda = 0$. Thus, for each i, $\det(\lambda I - L_i)$ has exactly one $(\lambda - 0)$ term. Using the fact that $\det(\lambda I - L) = \det(\lambda I - L_1)\det(\lambda I - L_2)\ldots\det(\lambda I - L_k)$, we have that the number of $(\lambda - 0)$ terms of $\det(\lambda I - L)$ is exactly k. In other words, the algebraic multiplicity of $\lambda = 0$ is exactly k. For the other direction, if the algebraic multiplicity of the the Laplacian L is k, then there must be k blocks in the Laplacian. Thus, G must have exactly k connected components. □

Chapter 3

Applications to Data

3.1 Synthetic Data

In order to better understand a Laplacian Eigenmaps embedding of time series data, we first attempt to interpret low-dimensional embeddings of high-dimensional, synthetic data. Let X be the high-dimensional data set consisting of phase-shifted sine curves. $X \in \mathbb{R}^{m \times n}$ where m represents the length of the sine curve and n represents the number of sine curves. For this experiment, and future experiments, we consider data living in the same dimension. This provides us with a concept of distance or similarity between data points.

3.1.1 Toy Data Set and Embeddings

Take X to be of dimension 1000×100 (i.e., 100 sine curves of length 1000.) For the sake of simplicity, consider sine curves that are evenly spaced via phase shifts between 0 and 2π. So $\sin(x - t) \in X$ such that $x = \frac{2\pi i}{1000}$, $i = 1, \ldots, 1000$ and $t = \frac{2\pi j}{100}$, $j = 1, \ldots, 100$. A subset of the data is plotted below.

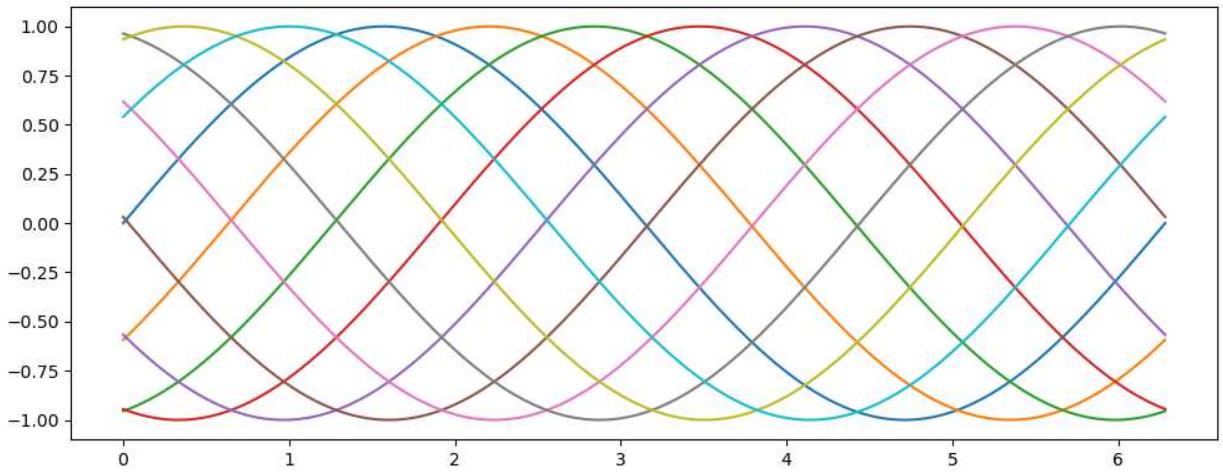


Figure 3.1: A subset of the vanilla sine curve data set.

Again for simplicity, assume that we use $k = 2$ nearest neighbors for clustering. (Similarly, we could choose ϵ such that each node corresponding to a sine curve has exactly 2 edges.) Notice that the Laplacian $L = D - W$ takes on the form of a *circulant matrix*.

$$L = \begin{bmatrix} l_0 & l_{n-1} & \dots & l_2 & l_1 \\ l_1 & l_0 & l_{n-1} & & l_2 \\ \vdots & l_1 & l_0 & \ddots & \vdots \\ l_{n-2} & & \ddots & \ddots & l_{n-1} \\ l_{n-1} & l_{n-2} & \dots & l_1 & l_0 \end{bmatrix} = \begin{bmatrix} 2 & -1 & \dots & 0 & -1 \\ -1 & 2 & -1 & & 0 \\ \vdots & -1 & 2 & \ddots & \vdots \\ 0 & & \ddots & \ddots & -1 \\ -1 & 0 & \dots & -1 & 2 \end{bmatrix} \tag{3.1}$$

The values on the diagonal are all 2, the values on the super-diagonal and sub-diagonal are -1, and the top-right corner and bottom-left corner are also occupied by -1. The rest of the matrix is filled with zeros. Given this vanilla data set, L will always be circulant for all values of k (or any choice of ϵ.) We now solve the generalized eigenvalue problem, $L\mathbf{f} = \lambda D\mathbf{f}$, pull the first d eigenvectors corresponding to the smallest nonzero eigenvalues, and use them to plot an d-dimensional embedding.

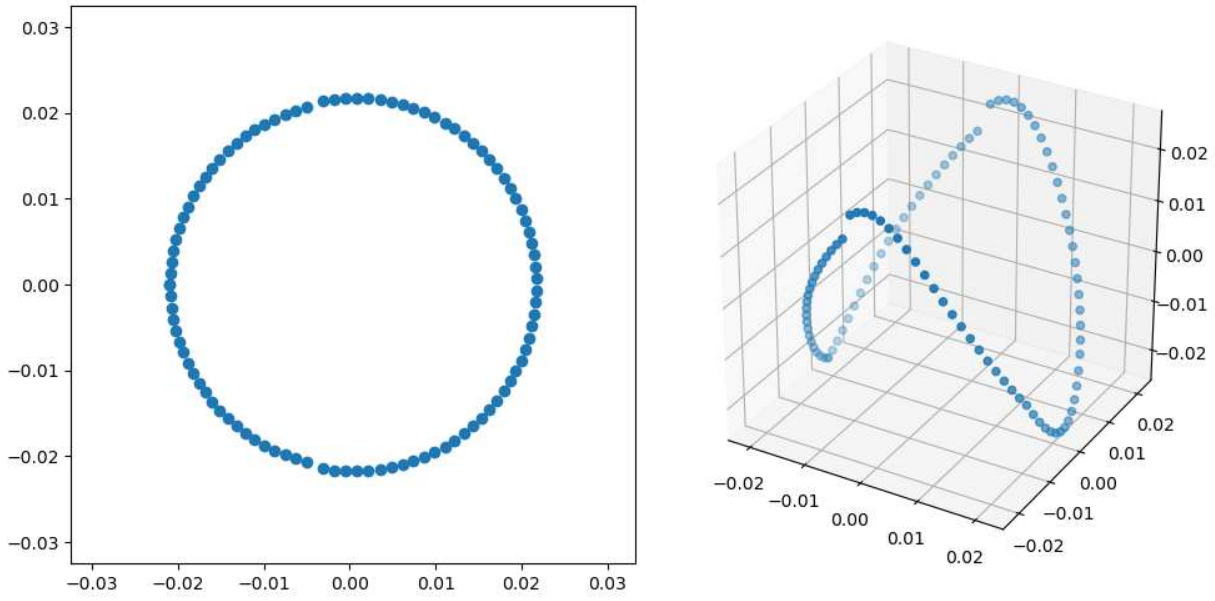


Figure 3.2: A 2 and 3-dimensional embedding of the vanilla Sine Curve data set

In the left part of Figure 3.2, each embedded data point corresponds to a point on a circle. In the right part, we see that the third coordinate also possesses oscillatory behavior, and by including it in a 3-dimensional plot, we get a "saddle" shape. The structure of these embeddings come as no surprise, as it is known that the eigenvectors of circulant matrices form a Fourier Basis.

We continue down this path of understanding Laplacian Eigenmap embeddings by applying noise and random phase shifts to our synthetic data set. A subset of the data can be found below.

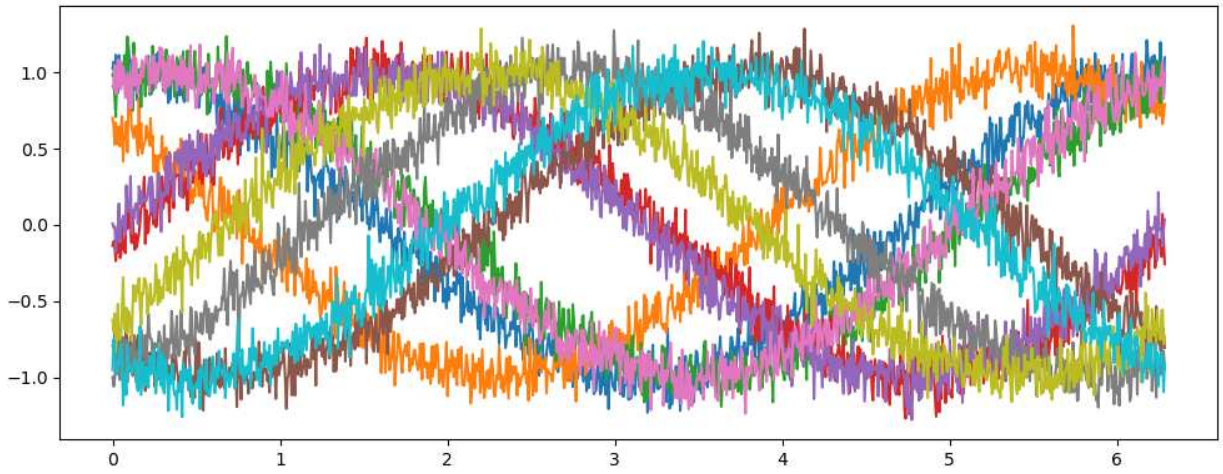


Figure 3.3: A subset of the noisy sine curve data set

Gaussian noise was added with a variance of 0.1 for illustrative purposes. This experiment was run with a Gaussian noise variance as high as 1. The results are unchanged, and all choices of Gaussian noise variance between 0.1 and 1 allow for an embedding into a circle with some structure, as seen in Figure 3.4. Similar results are achieved with minor adjustments to amplitude and/or period.

Note that with the noisy Sine Curve data set, the graph Laplacian generated loses the circulant property. The 2-nearest neighbors we chose earlier no longer allows for each data point to be a neighbors with its prior and subsequent data points. So one question that remains to be asked is, why do we still get an embedding that appears to be a circle?

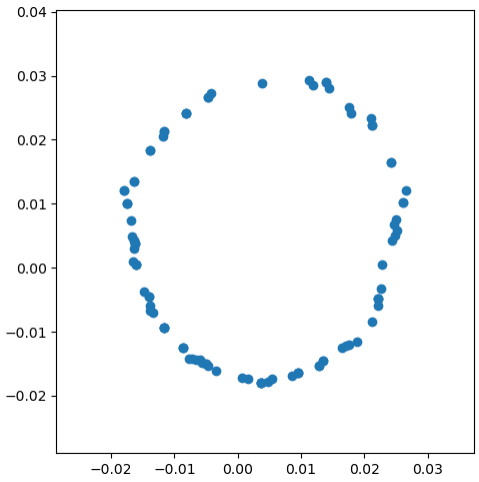


Figure 3.4: 2-Dimensional embedding of the noisy Sine Curve data set

To conclude our findings more precisely: Given a data set in which each data point is a high-dimensional time series with at least one period sampled, reducing the dimension to 2 via LE will produce an embedded topological circle. The rigidity/structure of the circle tells us how close the overall data set coincides with "nice" sine curves. Nice sine curves are those that have consistent oscillatory behavior. Given a new data point (time series) that is not oscillatory, it will not lie on the circle when embedded with the rest of the data set.

3.2 Comparing Signatures in Time Series Data

3.2.1 The Data Set

The data set we consider is a set of time series collected from 306 Collaborative Cross mice [13] challenged with *Salmonella typhimurium* as part of a broader study conducted by the Andrews-Polymenis and Threadgill laboratories at Texas A&M University, College Station, TX, USA. The experiment at Texas A&M was implemented to better understand the broad range of host-pathogen dynamics in mice with *Salmonella* [14].

A month or so prior to inoculation, each mouse is surgically implanted with a telemetry device. This device serves to measure body temperature and "activity" (net movement) of the mouse. Each mouse has time to recover from the implant surgery so as to not compromise the experiment. Ex-

periments run between approximately 14 and 28 days. Approximately 7 days of baseline telemetry data are recorded prior to infection for all mice. Each mouse is then innoculated with a dosage of *S. typhimurium* and observation continues for the remaining 7-21 days of experiment [14]. Body temperature and net movement are sampled once per minute for the duration of the experiment. For our purposes, we restrict our focus to the temperature time series data only.

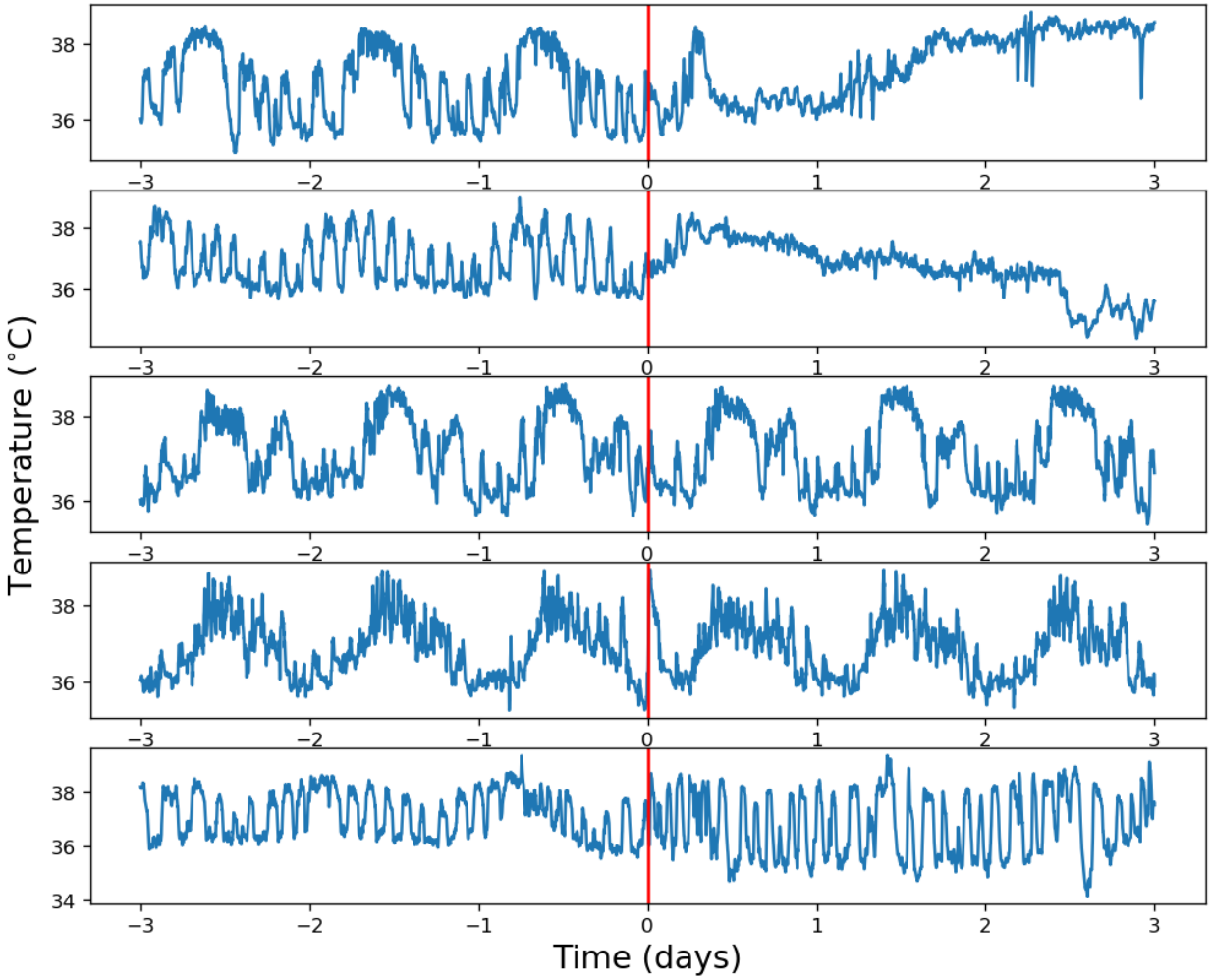


Figure 3.5: A collection of prototypical time series signatures of internal mice temperature aligned to the time of infection (red). (Figure taken from [14])

Figure 3.5 shows a very small subset of the original data set. Prior to infection, most mice temperatures generally remain between $34°$C and $39°$C. They also maintain very consistent circadian rhythms due to their consistent lifestyles. The temperature rises when they are awake and active.

The temperature then falls when they are asleep or inactive. Post infection, we notice that in most cases, the circadian rhythm is thrown off. Each mouse reacts to the inoculation differently, and we can see that in their temperature over the course of the following three days. Significant qualitative differences in patterns before and after infection are observed. After sifting through the raw time series data manually, we notice that there are about 5 defining characteristics, or signatures, that each post-infection time series possesses.

The top two time series in Figure 3.5 display a significant decrease in variance, and a complete loss of circadian rhythm. The only major difference between the two is that the top one ends up with hyperthermia, and the low one with hypothermia. Another signature that a good number of the time series exhibit is little to no reaction from the inoculation. The third and fourth time series in Figure 3.5 show this. Even three days post infection, both of them maintain a consistent circadian rhythm. The difference between these two is that the third one is uninterrupted at infection, while the fourth one has a quick spike at infection due to a very short-lived reaction. We would consider these mice to be tolerant, or even resistant, to the disease. The last major signature in the collection of time series is displayed by the final plot in Figure 3.5. Many time series are just noisy and erratic. Some maintain a circadian rhythm prior to infection, and become erratic post infection, and some are just inconsistent the entire time. This could be due to inaccurate measurements, or simply because those particular mice (or strains of mice) have inherently erratic temperatures.

While we have the ability to cluster this data set the old-fashioned way using the "eyeball technique," it would be convenient and beneficial to plot these high-dimensional time series in a low dimensional space in such a way that displays similarities and dissimilarities between their signatures explained above. In the following sections, we will attempt to embed this data set into 2 dimensions with a goal of similar signatures embedding close together and dissimilar signatures embedding far apart.

Pre-processing

A handful of the mice succumb to the infection prior to "completing" the experiment. A handful of mice have missing entries for the temperature sample ranging from 1 to a few thousand

consecutive minutes. Due to these reasons and more, it is imperative to pre-process the data in order to maintain consistency in our analysis.

As mentioned before, we need each of our data points to live in the same vector space. This provides us with a sense of distance/similarity between points by using various metrics within the vector space. The raw data contains 306 time series with lengths that range from 15833 to 40313. We begin by generating our data set X such that $X \in \mathbb{R}^{8640\times 306}$. The number 8640 corresponds to the number of minutes in 6 days. More specifically, we truncate each data point to just 3 days prior and 3 days after their respective time of infection (observable in Figure 3.5). During this phase, we also throw out 36 time series due to the amount of missing data. If a time series has more than 60 consecutive minutes of missing data, it is deemed insufficient, and not considered for the experiment. The rest of the missing data samples (less than 60 consecutive minutes) are handled via linear interpolation. The remaining time series are then each run through a median filtering which is provided by the `scipy` package in Python with `kernel_size=31`. Now we have 271 smooth, uniform-length data points with no missing entries for our data set X. Mathematically,

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(271)} \\ | & | & & | \end{bmatrix} \tag{3.2}$$

Different signatures appear at different time intervals during the experiment. So it wouldn't make sense to attempt to embed the entire data set X into 2 dimensions. Therefore, we partition X into windowed intervals and observe how the embedding changes as we slide this window across the data set day by day. Mathematically, we create a collection of $X_{[a,b]}$ from X such that each data point $x^{(i)}_{[a,b]} \in X_{[a,b]}$ and $[a, b]$ is an interval in days. So X has the following form:

$$X = \begin{bmatrix} X_{[-3,-2]} \\ X_{[-2,-1]} \\ \vdots \\ X_{[2,3]} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{x}^{(1)}_{[-3,-2]} & \mathbf{x}^{(2)}_{[-3,-2]} & \cdots & \mathbf{x}^{(271)}_{[-3,-2]} \\ | & | & & | \\ \mathbf{x}^{(1)}_{[-2,-1]} & \mathbf{x}^{(2)}_{[-2,-1]} & \cdots & \mathbf{x}^{(271)}_{[-2,-1]} \\ | & | & & | \\ \vdots & \vdots & & \vdots \\ | & | & & | \\ \mathbf{x}^{(1)}_{[2,3]} & \mathbf{x}^{(2)}_{[2,3]} & \cdots & \mathbf{x}^{(271)}_{[2,3]} \\ | & | & & | \end{bmatrix} \tag{3.3}$$

3.2.2 Results

With the data set sufficiently preprocessed, we now apply Laplacian Eigenmaps to acquire 2-dimensional embeddings. For neighborhoods, we use k-nearest neighbors with $k = 5$. To populate our weight matrix W, we use *simple minded* weights (i.e., for the heat kernel parameter in $W_{ij} = \exp\left(-\frac{d(\mathbf{x}^{(i)},\mathbf{x}^{(j)})^2}{t}\right)$, set $t = \infty$.) Figure 3.6 shows the time series of 5 specific mice over the course of the experiment. This figure has colored (green, yellow, red) windows corresponding to the embeddings found in Figure 3.7a, Figure 3.7b, and Figure 3.7c, respectively.

The first window from Figure 3.6 starts 3 days prior to inoculation and ends 2 days prior to inoculation. These are considered *healthy* time series which exhibit a consistent circadian rhythm to some extent. This is also seen in the corresponding embedding in Figure 3.7a. Similar to the noisy synthetic data set in Section 3.1.1, we have what appears to be a topological circle.

This "circle" retains its structure as the window continues forward towards the time of inoculation. However, when the sampling window crosses through the time of inoculation (yellow shading in Figure 3.6), we see that the circle collapses with no visible structure, as the immediate responses to infection are varied and inconsistent.

As we continue to slide this window, we begin to regain structure in the embeddings. Once the sliding window reaches the interval $[2, 3]$, we notice in Figure 3.7c that although there is structure in the embedding, it is no longer a circle, but more of a necklace or "V"-shape. When comparing

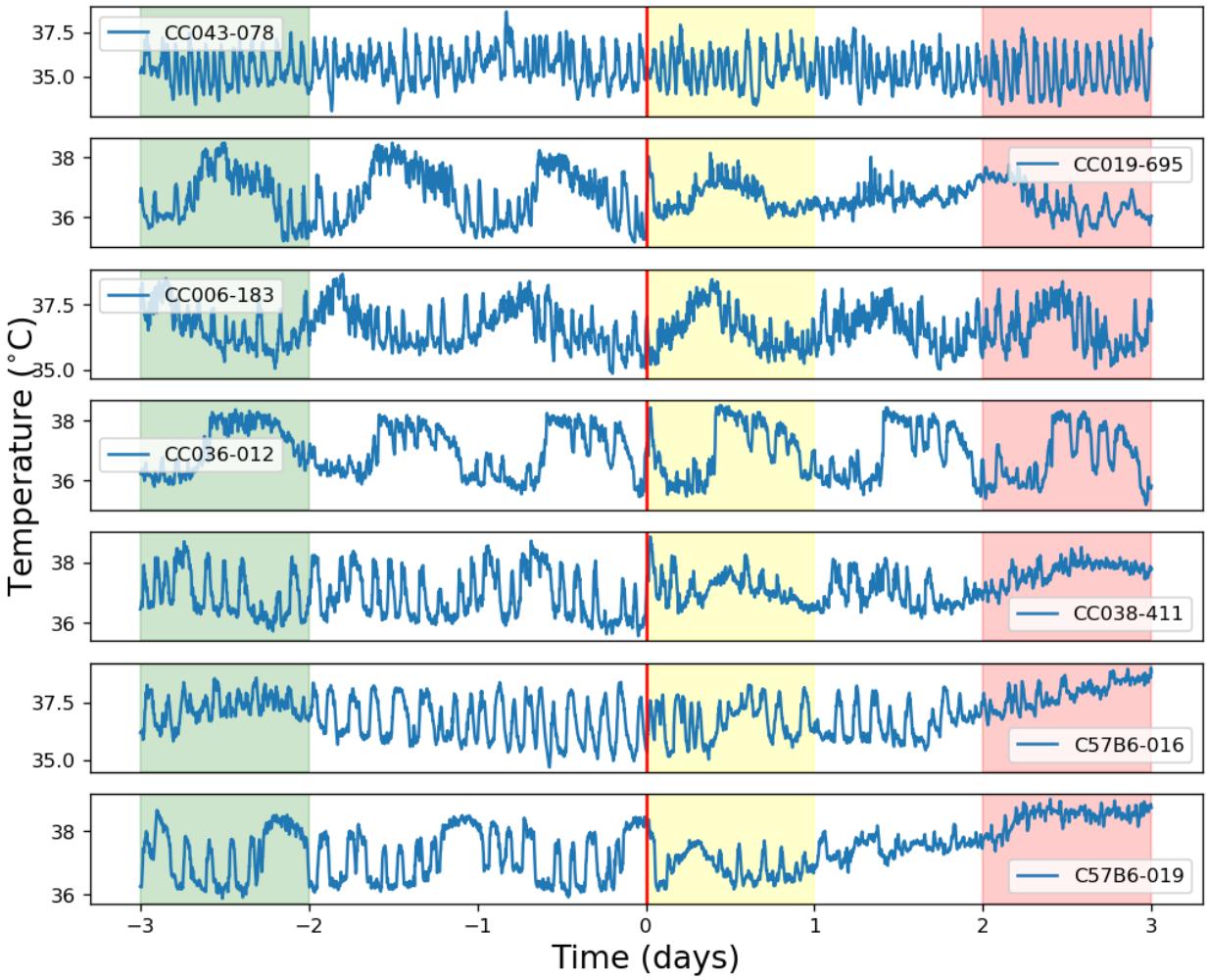
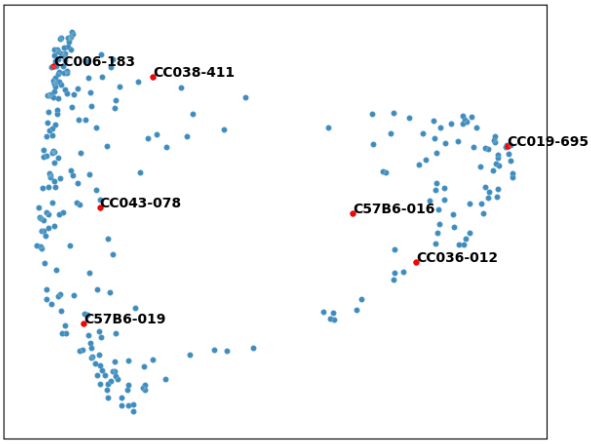
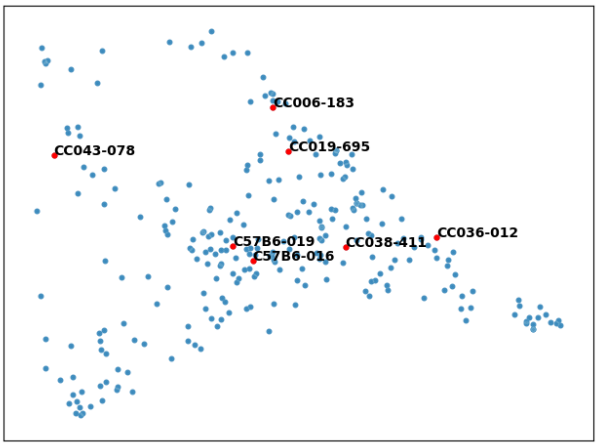


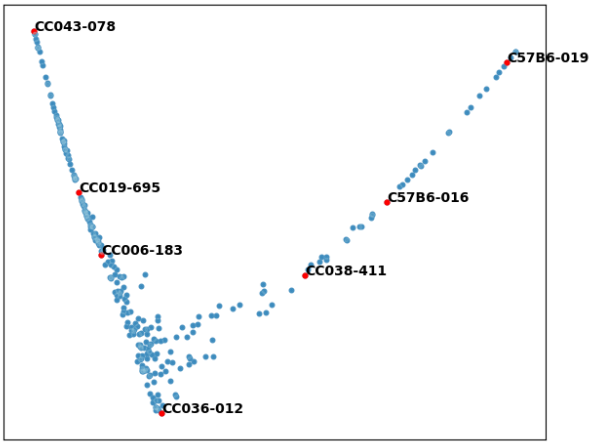
Figure 3.6: Seven selected time series; visualizing region of data used in Laplacian Eigenmaps embeddings in the corresponding panels in Figure 3.7. Figure 3.7a corresponds to the green window, Figure 3.7b to the yellow window, and Figure 3.7c to the red window.



(a) 2-D embedding of 1-day window beginning 3 days prior to infection



(b) 2-D embedding of 1-day window beginning at infection



(c) 2-D embedding of 1-day window beginning 2 days after infection

Figure 3.7: 2-dimensional embeddings of mice temperature time series using a 1-day sliding window.

the representative time series of mice along this "V"-shaped line in the red highlighted regions, we see explicit differences in the behaviors. We see a continuum of signature changes as we follow the necklace. Although these are not quite the original signatures we set out cluster, these are still distinguishable signatures that were captured by the dimensional reduction of Laplacian Eigenmaps.

The left side of the necklace corresponds to noisy and erratic temperatures over the course of that day. As we follow the necklace to the bottom, we see much more clustering of points. These coincide with the resistant/tolerant mice which are hardly affected by the innoculation and are able to maintain circadian rhythm. If we were to solely embed these time series, they would embed back to a circle, as they are considered to be "healthy" time series. We continue along the necklace to the upper-right side and notice that these time series are the ones with a major decrease in variance and end up with hyper/hypothermia.

In summary, although we were unable to automatically classify/cluster the time series based on pre-defined signatures, we were able to embed the data set in such a way that displayed a continuum of signature changes. This technique could be beneficial in determining whether or not a time series is "healthy" or not. Given a new sample, we embed it with the data set via Laplacian Eigenmaps. If the embedded point corresponding to the new sample is within some threshold of the cluster of healthy time series, we could be able to make further inferences about that point in terms of the signatures within the time series.

3.3 Conclusion

We began this paper with an introduction to data reduction methods, which can be categorized as linear (projections) or nonlinear. We provided motivation for the necessity of data reduction for visualization as well as motivation for an understanding of how nonlinear data reduction might be leveraged. After laying some foundational mathematical framework to pave the way, we set off to understand the inner-workings of the nonlinear dimensional reduction tool known as Laplacian Eigenmaps (LE). After explaining the algorithm, along with a few modifications, we analyzed, in

detail, the optimization problem that LE aims to solve, along with a comprehensive analysis of the constraints used in the optimization problem. After deducing that the solution can be found by solving a generalized eigenvalue problem, we attempted to understand 2-D and 3-D embeddings of time series data with a synthetic/toy data set. With this newfound understanding that a collection of "nice" sinusoidal data points embed to a topological circle, we applied LE to a much noisier, complex, real-world data set.

While we were unable to categorize the mice temperature time series into the specific bins we initially created, we did have success in finding and visualizing a continuum of signature changes within the data set via their 2-dimensional embeddings. This approach and its subsequent results could be used to potentially categorize new time series as healthy or unhealthy based on where they lie in the embedded space.

3.3.1 Future Work

As for future work in the theory of non-linear dimensional reduction, it would be interesting to explore the idea of "online embedding". Malik, et al. has fairly recent results in [15] discussing Generalized Incremental Laplacian Eigenmaps (GENILE), which is a novel online version of the Laplacian Eigenmaps. The online embedding of out-of-sample points would not only be less computationally expensive, but might also provide more insight to a data set.

Another question to explore is: What is a "sufficient" amount of data needed to embed? On the surface, the answer to this question seems like it could be an ad-hoc heuristic, but further exploration may reveal some lower bounds on how many data points from specific data sets are actually needed to produce a meaningful embedding.

Finally, one parameter that was not tuned during this experiment was the metric chosen to generate the distance matrix. We used Euclidean distances to determine how close one time series was to another. But is the 2-norm really the best way to compare similarities between time series?

Bibliography

[1] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[2] Effrosini Kokiopoulou, Jie Chen, and Yousef Saad. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, 18(3):565–602, 2011.

[3] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[4] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[5] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[6] David L Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[7] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[8] Lawrence K Saul and Sam T Roweis. An introduction to locally linear embedding. *unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html*, 2000.

[9] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[10] Yoshua Bengio, Jean-françcois Paiement, Pascal Vincent, Olivier Delalleau, Nicolas L Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in neural information processing systems*, pages 177–184, 2004.

[11] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint arXiv:1903.11240*, 2019.

[12] Anne Marsden. Eigenvalues of the laplacian and their relationship to the connectedness of a graph. *University of Chicago, REU*, 2013.

[13] David W Threadgill and Gary A Churchill. Ten years of the collaborative cross. *Genetics*, 190(2):291–294, 2012.

[14] Manuchehr Aminian, Helene Andrews-Polymenis, Jyotsana Gupta, Michael Kirby, Henry Kvinge, Xiaofeng Ma, Patrick Rosse, Kristin Scoggin, and David Threadgill. Mathematical methods for visualization and anomaly detection in telemetry datasets. *Interface Focus*, 10(1):20190086, 2019.

[15] Zeeshan Khawar Malik, Amir Hussain, and Jonathan Wu. An online generalized eigenvalue version of laplacian eigenmaps for visual big data. *Neurocomputing*, 173:127–136, 2016.