Non-linear dimension reduction: Laplacian Eigenmaps and Locally Linear Embeddings

MATH4069 Statistical Machine Learning

Group project report

2021/22

School of Mathematical Sciences

University of Nottingham

Group H:

James Dupère

Gurdeep Lidder

Albert Nyarko-Agyei

Mustafa Omar

Joseph Walsh

Abstract

With the increasing size and dimensionality of data, the interpretation and insight into said data becomes hard to extract. In this investigation, we look at two non-linear dimensionality reduction methods; Laplacian Eigenmaps and Locally Linear Embeddings (LLE). We look at the intuition, mathematical formulation and effectiveness of these methods. We then apply those methods to data sampled from non-linear manifolds, and observe their resultant embedding to gain insight into how the methods behave and their effectiveness at reducing the dimensionality of the data.

We find that for some datasets LLE performs better while for others not so well, and similarly for Laplacian Eigenmaps. The datasets that we used showed that LLE was better at mapping the points to distinct values whereas, for many different choices of t, Laplacian Eigenmaps gave plots where many points overlapped. This potentially speaks to the sensitivity of the first method to the choice and proximity of neighbouring points and how it is better at preserving variation in the data. However, Laplacian Eigenmaps was able to capture an added feature of the data in that some completely distinct sections of the data were close in the 3D space.

Contents

1 Introduction

1.1 Background

Data with a high number of dimensions is often difficult to model and interpret. To understand this concept on a theoretical level, consider the Euclidean distance between two distinct points, $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ in a unit hypersphere. For every extra non-negative feature we add to these points, we add a non-negative amount to the Euclidean distance between them. Notice that despite the two points both being within the unit hypersphere, they are increasing far apart by this metric. High dimensional datasets that are not limited to the unit hypersphere fare worse and are often sparse despite existing underlying structure in the data. This illustrates the problem of working with a sparse high-dimensional data set or the Curse of dimensionality. This report ultimately questions if considering smaller sections of our data set prove useful to understanding the aforementioned structures.

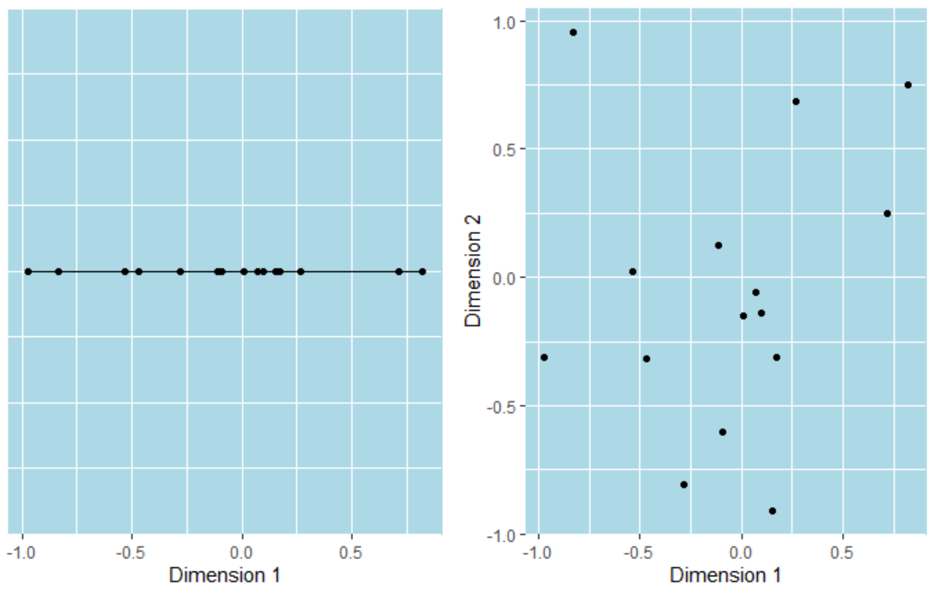


Figure 1: Effect of increasing dimensions on sparsity

Figure 1 shows a slightly different example and how the expansion in features causes points that are seemly close in one dimension to be different in others. This shows the problem for two dimensions but one can imagine that the issue is compounded when more dimensions are added and the data set is no longer restricted to vectors with points drawn from a uniform random distribution on $(-1, 1)$.

Linear dimensionality reduction techniques map the data set down to a lower dimension and circumvent the issue of high dimensionality. An example of such a method is Principal Component Analysis. This specific technique uses the Singular Value decomposition of the data to project it to a lower-dimensional space. The method uses the within feature variance of the data and constructs new variables that maximize the variance of the data set. A problem with this class of methods, however, is that they typically do not adequately respect the proximity of 'close' data points and this information is not used in the mapping to new variables/features. In this report, we use this notion of closeness between points in a data set to encompass images of the people driving with varying angles of their face or two numerical high dimensional data points that vary in just a few features. Linear dimension reduction techniques do not try and learn the underlying structure in the data so the top two principal components of the images of the same person disregard this information and focus on combinations of the features that show maximum variation.

In contrast, Non-linear dimensionality techniques place emphasis on points that are close in the high dimensional space being close in the low dimensional space. The non-linear techniques that we will focus on in this paper fall under a branch of unsupervised learning, otherwise known as Manifold Learning.

This section of the report explores the topic of Manifold Learning, why it is needed in contrast to the problems with PCA and how we assess the quality of a method. Section 2 disuses the methodology of the report and Section 3 begins the applications of these methods.

1.2 Manifold Learning

Manifold Learning has a central assumption which is that high dimensional data drawn from real world scenarios lie on low dimensional manifolds embedded within the high dimensional space. Loosely speaking the techniques associated with manifold learning make the assumption that the original data set actually comes from the specified lower-dimensional space and has been embedded in the high-dimensional space, to begin with. With this assumption of an underlying structure within the data, we then use methods such as LLE or SE to find the mapping that best describes this structure.

A manifold can be thought of as any surface in high dimensional space that can be 'charted/mapped out' or is locally Euclidean. To formally define a manifold, we must first define a topological space as a set X that is equipped with a set of subsets U, which are known as the open sets and they have the properties of being close under finite intersections and arbitrary unions. Formally then, manifolds are topological spaces that locally resemble Cartesian space. This means regions on the manifold appear to be flat when viewed closely and this property is important because it forms the basis for mapping the data set onto a flat 2-dimensional axis. For the purposes of this report, the best definition of a manifold is a topological space that is equipped with an atlas. This is a collection of mappings to $\mathbb{R}^2$ that describe the manifold and this is exactly our goal when attempting to visualize high dimensional datasets. From this, it is clear that treating datasets as manifolds has some useful properties and lead to new interpretations of data [C. Fefferman and Narayanan, 2016].

1.3 Potential Problems with PCA

Before using PCA it is reasonable to check for linear correlations between the features of the data set. This is because standard PCA creates linear combinations of the existing features that try and maximize variance however if the features are already linearly independent then there is no need. This particular existence of correlations corresponds to the covariance matrix of the data set having non-zero entries on either side of the leading diagonal. Manifold Learning techniques take a different approach by computing the similarity between points in the data set rather than the similarity between features. Naturally, this means that pairwise relationships are largely destroyed by PCA and preserved by the techniques we will review in this paper.

Following on from the orthogonality of the new features computed by PCA, this property means that the new projections created by PCA are constrained to this property when there could be basis vectors that are much better at representing the information in the data set. Also, the features have to be scaled before PCA reduces the dimensions. This is another way in which the method destroys features in the data set especially if the information of the weight of the variables has real-world implications. This leads to the question of the interpretability of the new features. Originally if the features were drawn from a real-world scenario, then their meanings are likely to be clear however linear combinations of these meaningful features are typically less meaningful and can only be referred to as principal components [Jolliffe and Cadima, 2016].

On the whole, the most important reason for investigating non-linear methods of dimension reduction is their ability to preserve pairwise relationships, and this is one of the criteria we will use to assess the chosen methods.

1.4 Assessing the quality of a method

A useful way of comparing how well a particular method has mapped the data into a lower dimension is to look at the co-ranking matrix Q of the points in the two dimensions [Lee and Verleysen, 2009]. It does this by ranking the points by distance in both the higher and lower dimensions (for each point) and then forms a matrix based on the change in this rank once the relevant dimension reduction method has been applied. If we define the distance between two points $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ as d_{ij} and then r_{ij} as the rank of $\boldsymbol{x_i}$ with respect to $\boldsymbol{x_j}$ (or saying that $\boldsymbol{x_j}$ is the r_{ij}th nearest neighbour of $\boldsymbol{x_i}$) can be defined by Equation 1.1[1] [G. Krämer and Mahecha, 2018].

$$r_{ij} = |\{k : d_{ik} \leq d_{ij}\}| \tag{1.1}$$

For the sake of notation we will define the rank in higher dimensional space as r_{ij} and the rank in lower dimensional space as $\hat{r}_{ij}$. Comparing these ranks allows us to compute Q whose elements are defined by Equation 1.2.

$$q_{kl} = |\{(i, j) : \hat{r}_{ij} = k, r_{ij} = l\}| \tag{1.2}$$

A simple way of interpreting the elements of this matrix q_{ij} is to say that this is the number of points which had the rank j in the higher dimension that became rank i in the lower dimension. From this definition it follows that in a perfect lower dimensional representation the only non-zero entries in Q will appear in the diagonal and that any non-zero entries in the upper triangle will be points that have been pushed further apart than they were in the higher dimension and points in the lower triangle corresponding to points which were initially further apart becoming closer in the lower dimension.

From the entries in Q we can now assess how well a method performs. For example, a good measure of quality would be to test how similar the nearest neighbours in the higher dimension is to the nearest neighbours in the lower dimension. One way of doing this is to calculate the Local Continuity Meta Criterion (LCMC) [Chen and Buja, 2009] which for a given k counts how many of the k nearest neighbours in the higher dimension are still in the k nearest neighbours in the lower dimension (as a proportion), with an adjustment to account for random embeddings (you would expect that this measure would be high for a large number of k nearest neighbours whether you apply an NDLR method or just simply apply the dimension reduction at random hence the adjustment). This is summarised by Equation 1.3 for a sample of size n.

$$LCMC(k) = \frac{1}{kn} \sum_{i=1}^{k} \sum_{j=1}^{k} q_{ij} - \frac{k}{n-1} \tag{1.3}$$

[1]The notation $|S|$ refers to the number of elements belonging to a set S

Adjusting LCMC to have a maximum of 1 allows for an easy measure for comparison regardless of sample size and so the quality measure R_{NX} can be defined by Equation 1.5 [Lee et al., 2013] where a perfect method would have an R_{NX} value of 1 and a random embedding would have an R_{NX} value of 0.

$$R_{NK}(k) = \frac{\frac{n-1}{kn} \sum_{i=1}^{k} \sum_{j=1}^{k} q_{ij} - k}{n-1-k} \tag{1.4}$$

From this definition it follows that a large R_{NX} value for low k represents a method that preserves local distances between points well and a large R_{NX} for high k represents a method that does well in preserving global distances. As R_{NX} is a function of k, this measure also allows us to produce a plot (typically on a logarithmic scale) over all possible choices of k to see how the performance of a particular method changes as k increases. Calculating the area underneath the corresponding curve $(AUC_{ln(k)}(k))$also allows for an overall assessment in quality.

$$AUC_{ln(k)}(k) = \frac{\sum_{k=1}^{n-1} R_{NK}(k)}{\sum_{k=1}^{n-1} \frac{1}{k}} \tag{1.5}$$

This measure is also normalised to have a maximum of 1 and due to it being on a logarithmic scale, AUC rewards methods which are superior at preserving local distances over global distances.

2 Methodology

2.1 Spectral Embedding

The first method of non-linear dimension reduction that we considered is Spectral Embedding. This method reduces the dimensions of data on manifolds by defining the data using Graph Theory and translating this into vector data by solving an eigenvalue problem produced from Spectral theory. Producing an embedding involves building a neighbourhood graph with the high dimensional data, and preserving the maximum possible features of the data in a lower dimension. An understanding of Graph Theory is required for this, and is provided in Section 2.1.1.

2.1.1 Graph Theory

Consider a data set with points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{R}^D$ present on a manifold. Graph Theory is such that each data point is considered a 'node' or 'vertex' making up set V. These vertices are connected by 'edges' forming set E. Edges form only between nodes that are considered 'close' and the presence of an edge implies the vertices are connected. The definition of close is dependant on context and relevant definitions shall be explored in Section 2.1.2.2. A graph G can be defined as an ordered pair of sets V and E as $G = (V, E)$ and is considered connected if there exists a path along edges from any node u to any other node v. A disconnected graph is divided into k connected sub-graphs $G_1, G_2, ..., G_k$ where $G = (G_1 \cup ... \cup G_k)$ representing all of the data [Chartrand and Zhang, 2012].

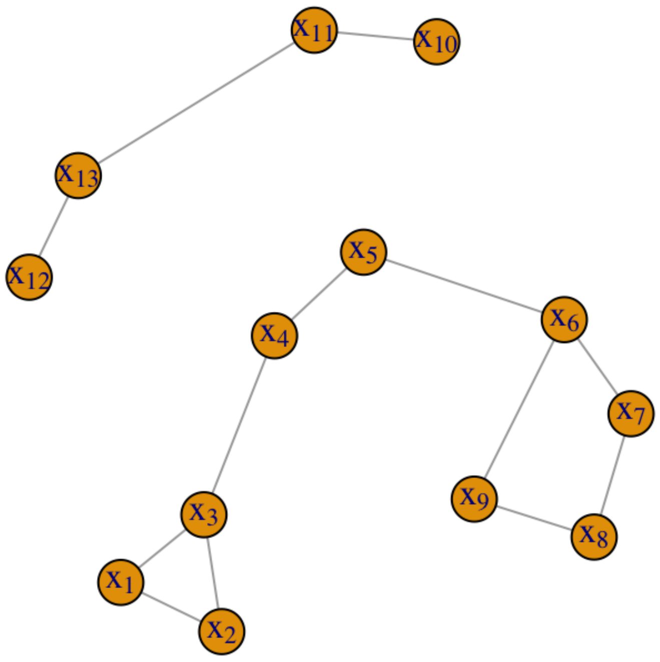


Figure 2: Demonstration of a dataset represented as a graph

2.1.1.1 Spectral Theorem

Applying Graph Theory to data sets allows for the creation of adjacency matrices which capture the relationship between vertices. By nature, these matrices are symmetric and their properties will be shown below:

Consider a symmetric matrix $A \in \mathbb{R}^{n \times n}$. Every eigenvalue λ is real and has a corresponding real eigenvector $\boldsymbol{u} \in \mathbb{R}^n$, such that:

$$A\boldsymbol{u} = \lambda \boldsymbol{u}$$

where these eigenvectors are also orthogonal, i.e. $A\boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1$, $A\boldsymbol{u}_2 = \lambda_2 \boldsymbol{u}_2$, $\boldsymbol{u}_1 \cdot \boldsymbol{u}_2 = 0$, where $\lambda_1 \neq \lambda_2$. Furthermore,

$$\lambda_i = \min_{\boldsymbol{x} \perp \boldsymbol{u}_1, \ldots, \boldsymbol{u}_i - 1} \frac{\boldsymbol{x}^T M \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} \tag{2.1}$$

and

$$\boldsymbol{u}_i = \operatorname*{arg\,min}_{\boldsymbol{x} \perp \boldsymbol{u}_1, \ldots, \boldsymbol{u}_i - 1} \frac{\boldsymbol{x}^T M \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}$$

define the eigenvalues and eigenvectors using the Rayleigh Quotient. [Spielman, 2015]

2.1.2 Laplacian Eigenmaps

One application of Spectral Embedding is using Laplacian Eigenmaps. This method approximates the Laplace-Beltrami operator by building a neighbourhood graph G of the data and computing the Laplacian of this matrix. This allows the embedding of the manifold to be found with the goal of the method being to preserve locality such that nodes that are considered close $\in \mathbb{R}^D$ (where D is the initial dimensions of the data), remain close once dimensions have been reduced.

2.1.2.1 Laplace-Beltrami Operator

Applying the Laplace-Beltrami operator to a manifold finds the optimal embedding. This is proven by considering an m-dimensional Riemannian manifold $(\mathcal{M}, g)$ present within a Riemannian Structure $\in \mathbb{R}^l$ where $m \ll l$. The operator is represented by:

$$\Delta_g f = -div\, grad\, f = \nabla \cdot \nabla f$$

where f represents a function which maps points on the manifold to a real line, preserving the closeness of data points such that f: $\mathcal{M} \to \mathbb{R}$. The operator also requires f to be twice differentiable as it computes the divergence of the gradient [Urakawa,]. By considering two neighbouring points, finding the following most preserves locality:

$$\operatorname*{arg\,min}_{\|f\|_{L^2(\mathcal{M})}=1} \int_{\mathcal{M}} \|\nabla f(x)\|^2,$$

and minimising this is equivalent to finding the eigenfunctions of the Laplace-Beltrami operator[M.Belkin and P.Niyogi, 2003].

The graph discretisation of the Laplace-Beltrami operator proves that the graph Laplacian of the data is a sufficient approximation to the operator in finding the embedding. This is because the eigenvalues and eigenfunctions of the operator are approximated by the eigenvalues and eigenvectors of a suitably weighted Laplacian of a proximity graph [Burago et al., 2014].

The solutions of the eigenvalue problem of the Laplacian matrix allow dimensions to be reduced to a data set $\boldsymbol{y}_1, ..., \boldsymbol{y}_n \in \mathbb{R}^d$ where $d \ll D$.

2.1.2.2 Closeness Definitions

The definition of close can now be revisited and how this is used to represent graph data in a matrix reconsidered. In the context of Laplacian Eigenmaps, two common definitions exist:

1. **$\boldsymbol{\varepsilon}$-neighbourhoods approach.** Nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are considered close if the distance between them (Euclidean distance $\in \mathbb{R}^D$) $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq \varepsilon$, where ε represents a boundary around each vertex of radius ε.

2. **$\boldsymbol{k}$-nearest neighbours approach.** Nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are considered close if one of the nodes is within the k nearest neighbours of the other.

For either approach, the variable k or ε is chosen by the user. Both definitions have their justifications. The ε-neighbourhood approach represents the geometric properties of the data better than k-nearest neighbours, although choosing a suitable ε is often more difficult than choosing k (since prior knowledge of the data is needed to determine what might be considered close or otherwise). Applying a k-nearest neighbours approach is less geometrically intuitive than ε-neighbourhood as it does not consider distance between nodes, however this means there is less of a chance of producing disconnected graphs and it is easier to implement in practice.

2.1.2.3 Weighting edges

The definitions in Section 2.1.2.2 determine whether edges between vertices exist, however in order to construct a Laplacian matrix that can be approximated to the Laplace-Beltrami operator, the edges need to be suitably weighted. A weight matrix $W \in \mathbb{R}^{n \times n}$ can be produced in one of two ways:

1. **Combinatorial.** If an edge exists between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ then $W_{ij} = 1$

2. **Heat Kernel.** Where a Gaussian relation is applied depending on the Euclidean distance between the nodes such that

$$W_{ij} = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{t}} \tag{2.2}$$

   where t is a parameter that the user decides (choosing $t = \infty$ is equivalent to the Combinatorial method).

For both methods, all weights are non-negative if no edge exists, i.e. $W_{ij} = 0$. Although the Combinatorial method is more convenient as it removes the problem of choosing a suitable t, it provides a Laplacian matrix which is a less close approximation to the Laplace-Beltrami operator, potentially leading to poorer results when reducing the data. This means it is usually advantageous to apply the Heat Kernel method, though this depends on the size of the data set and how long it takes computationally.

The choice for the use of the Heat Kernel equation in the weights for the Laplacian matrix stems from its relationship to the Laplace-Beltrami operator. Returning to the manifold $\mathcal{M}$ as discussed in Section 2.1.2.1, f is now defined as an initial heat distribution, where f: $\mathcal{M} \to \mathbb{R}$. The heat distribution at time t is $u(x,t)$ and the heat equation is:

$$(\frac{\partial}{\partial t} + \Delta_{\mathcal{M}})\, u = 0. \tag{2.3}$$

A solution to the heat equation is the following: $u(x,t) = \int_{\mathcal{M}} H(t,x,y)f(y)$, where $H(t,x,y)$ is called the Heat Kernel [Cruz, 2003]. The Heat Kernel $H(t,x,y)$ can be approximated by the following:

$$H(t,x,y) \approx \frac{1}{(4\pi t)^{\frac{m}{2}}} e^{-\frac{\|x-y\|^2}{4t}}, \tag{2.4}$$

assuming that x and y are sufficiently close, t is small and an exponential co-ordinate system is used [M.Belkin and P.Niyogi, 2003]. If x and y represent points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ within $\mathcal{M}$, which are packed densely, the integrals can be replaced with summations. Then as $t \to 0$, substituting Equation 2.4 into Equation 2.3, gives:

$$\Delta_{\mathcal{M}} f(\boldsymbol{x}_i) \approx \frac{1}{t}\big[f(\boldsymbol{x}_i) - \frac{1}{k}\frac{1}{(4\pi t)^{\frac{m}{2}}}\sum_{\boldsymbol{x}_j} e^{\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{4t}} f(\boldsymbol{x}_j)\big], \tag{2.5}$$

since $\int_{\mathcal{M}} H(t,x,y)f(y) \to f(x)$ as t decreases. This equation is a constant equation and so applying the Laplace-Beltrami operator gives a result of 0 (as the function is not twice differentiable). Therefore for the right hand side of Equation 2.5 to equal 0, the following must be true:

$$\Big(\frac{1}{k}\frac{1}{(4\pi t)^{\frac{m}{2}}}\Big)^{-1} = \sum_{\boldsymbol{x}_j} e^{\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{4t}}.$$

This is validates the use of the Heat Kernel in choosing weights for the Laplacian matrix, as it is analagous to weighting in this context.

2.1.2.4 Laplacian Matrix

The graph structure of the data has been captured in matrix W, and so now $G = (V, E, W)$. The Laplacian matrix L can now be computed and is defined as $L = D - W$. Degree matrix $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix with entries $Dii = \sum_{j=1}^{n} W_{ji}$. Solving eigenvalue problem

$$L\boldsymbol{f} = \lambda D\boldsymbol{f} \tag{2.6}$$

gives non-negative real eigenvalues, due to L being a positive semi-definite matrix.[Mittal,] Ordering the eigenvalues as follows:

$$0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{n-1}$$

gives eigenvectors $\boldsymbol{f_0}, ..., \boldsymbol{f_{n-1}}$. This allows the creation of points $\boldsymbol{y}_1, ..., \boldsymbol{y}_n \in \mathbb{R}^d$, such that the embedding of the high dimensional data in d-dimensions is:

$$\boldsymbol{y}_i \to (\boldsymbol{f}_1, ..., \boldsymbol{f}_d)$$

Note the embedding starts from eigenvector $\boldsymbol{f}_1$. This is because eigenvalue $\lambda_0 = 0$ with corresponding eigenvector $\boldsymbol{f}_0 = \mathbf{1} \in \mathbb{R}^n$. This result is due to matrix L being symmetric and its rows and columns of summing to zero. [G.Chen,].

To demonstrate the goal of the method introduced in 2.1.2, it can be shown that the Laplacian Eigenmap algorithm optimally preserves locality in the embedding. For this to be the case, consider a fully connected graph of data $(\hat{\boldsymbol{x}}_i, ..., \hat{\boldsymbol{x}}_n) \in \mathbb{R}^D$. The algorithm would find reduced dimension points $(\hat{y}_i, ..., \hat{y}_n) \in \mathbb{R}^1$ by minimising the following function:

$$\sum_{ij} W_{ij}(\hat{y}_i - \hat{y}_j)^2$$

under some appropriate constraints which removes trivial solutions. Minimising the function ensures data points remain close due to the choice of W_{ij}. If points $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ are close, W_{ij} will be close to 1, applying a large penalty if $\hat{y}_i$ and $\hat{y}_j$ are not close. Function 2.1.2.4 can be expanded to:

$$\sum_{ij} W_{ij}\hat{y}_i^2 + \sum_{ij} W_{ij}\hat{y}_j^2 - 2\sum_{ij} W_{ij}\hat{y}_i\hat{y}_j$$

and since $D_{ii} = \sum_{j=1}^{n} W_{ji}$ this becomes:

$$\sum_{i} D_{ii}\hat{y}_i^2 + \sum_{j} D_{jj}\hat{y}_j^2 - 2\sum_{ij} W_{ij}\hat{y}_i\hat{y}_j$$

Let $\hat{\boldsymbol{Y}} = (\hat{y}_1, ..., \hat{y}_n)^T$, and so the above reduces to:

$$2\hat{\boldsymbol{Y}}^T D\hat{\boldsymbol{Y}} - 2\hat{\boldsymbol{Y}}^T W\hat{\boldsymbol{Y}}$$

$$= 2\hat{\boldsymbol{Y}}^T(D - W)\hat{\boldsymbol{Y}} = 2\hat{\boldsymbol{Y}}^T L\hat{\boldsymbol{Y}}$$

The above also proves matrix L is positive semi-definite, as was stated with Equation 2.6. This then means that:

$$\hat{\boldsymbol{Y}}^T L\hat{\boldsymbol{Y}} = \frac{1}{2}\sum_{ij} W_{ij}(\hat{y}_i - \hat{y}_j)^2 \tag{2.7}$$

and so the minimisation becomes:

$$\underset{\hat{\boldsymbol{Y}}^T D\hat{\boldsymbol{Y}}=1, \hat{\boldsymbol{Y}}^T D=0}{\arg\min} \quad \hat{\boldsymbol{Y}}^T L\hat{\boldsymbol{Y}}$$

where the constraint $\hat{\boldsymbol{Y}}^T D\hat{\boldsymbol{Y}} = 1$ removes an arbitrary scaling factor, and constraint $\hat{\boldsymbol{Y}}^T D = 0$ eliminates the trivial solution of 0 for the smallest eigenvalue. Applying Equation 2.1 gives Equation 2.6 as the eigenvalue problem for the optimal embedding. This is the case for a 1-dimensional embedding. For higher dimensions, the same result is found where the minimisation function is now:

$$\sum_{ij} W_{ij}(y_i - y_j)^2 = trace(\hat{\boldsymbol{Y}}^T L \hat{\boldsymbol{Y}})$$

where $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_n)^T \in \mathbb{R}^d$. Similar to the 1-dimensional case, the function becomes:

$$\underset{\hat{\boldsymbol{Y}}^T D \hat{\boldsymbol{Y}} = \boldsymbol{I}, \hat{\boldsymbol{Y}}^T D = 0}{\arg\min} \quad trace(\hat{\boldsymbol{Y}}^T L \hat{\boldsymbol{Y}})$$

once again giving Equation 2.6 as the eigenvalue problem for the optimal embedding, from applications of the Spectral Theorem. In this case, the constraint constraint $\hat{\boldsymbol{Y}}^T D \hat{\boldsymbol{Y}} = 1$ prevents a collapse onto a smaller than d-dimensional subspace[M.Belkin and P.Niyogi, 2003].

2.2 Locally Linear Embedding

Locally Linear Embedding (LLE), is an unsupervised dimension reduction algorithm. In this section, LLE is explained, by looking at the goal, the assumption it makes and the algorithm behind the method.

2.2.1 Mapping a D-dimension vector to d-dimensions

The primary goal of LLE is to reduce the dimensions of our dataset going from a D-dimensions manifold to a different d-dimension manifold, where $d << D$. For an example manifold (Swiss roll), we generally would want to discard the manifold, and preserve the local relationship between an instance ($\boldsymbol{x}_i$) and its k-nearest-neighbours (local neighbourhood).

Generally an instance of data could be represented as a vector. We can think of high dimensional datasets as high dimensional vectors (D-dimensions). Figure 3 represents the unitary components of a D-dimensional vector (each arrow is orthogonal to the rest). It's impossible for us to visualise such vectors completely, and therefore we cannot plot the dataset and gain insight.

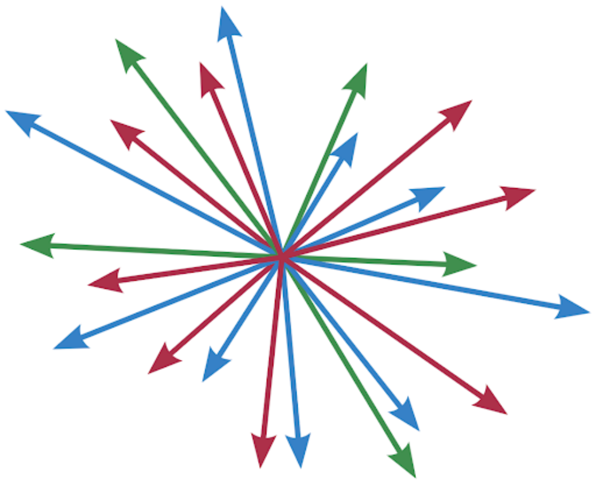


Figure 3: Showing a non accurate representation of a D-dimension vector's unit components, where each arrow is orthogonal to every other arrow.

However, if we can accurately reduce the dimensions of D-dimensional vectors to d-dimensions (4), then we can have a representation of the dataset that we could comprehend and therefore gain insight from. Furthermore, some other machine learning models/algorithms (Neural networks, SVM, Random forests) struggle when training with D-dimensional data, therefore we could also use our reduced d-dimensional representation of the dataset to train such models.

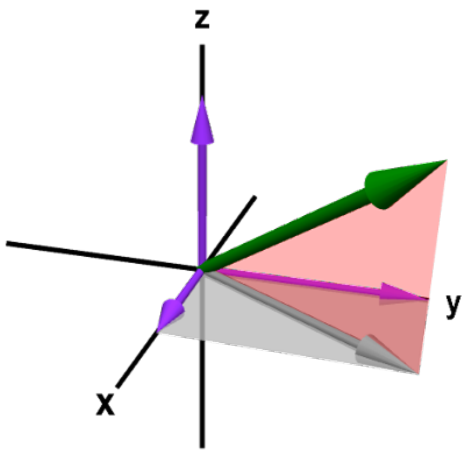


Figure 4: Showing a 3 dimensional vector with its orthogonal components in 3-dimensions.

LLE focuses on reducing the dimensions of the dataset while maintaining the relationship between local neighbourhoods. To illustrate this Figure 5 shows the result when LLE is applied to the Swiss roll manifold. (A) represents the D-dimensional manifold, (B) represents the D-dimensional dataset and (C) represents the d-dimensional output of LLE. As you can intuitively observe, the relationship between local neighbourhoods is preserved, and a significant amount of information from the manifold is extracted while omitting the manifold it self. In other words we have correctly obtained a mapping (this could have been a non-linear manifold; this will be clearer later) of the Swiss roll.

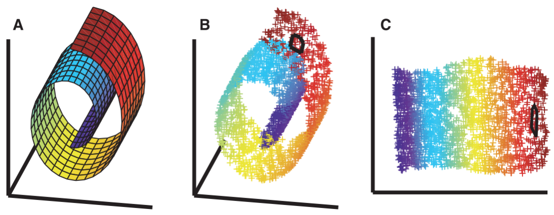


Figure 5: [Roweis and Saul, 2000] Showing the result of LLE (C) when applied to a sufficient dataset (B) of the example manifold (A).

By analogy, LLE can be compared to a scanner, by looking at Figure 5, we can observe that the purple section is scanned first, followed by the blue, yellow and finally the red sections. This scan or unrolling of the manifold is what LLE aims to achieve. In Figure 6 [Saul and Roweis, 2001], we see a comparison between PCA and LLE when applied to a dataset of noise behind faces. We can clearly see that LLE is better at preserving the neighbourhood structure of the data.
A further discussion of the results obtained by applying LLE on toy and real datasets is in Section 3.

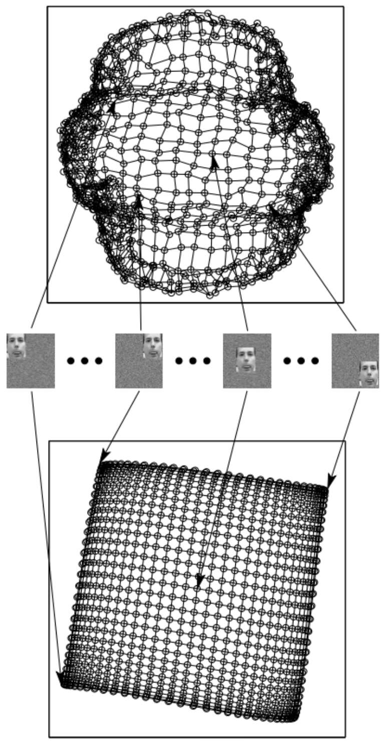


Figure 6: [Saul and Roweis, 2001] Showing a comparison between LLE (bottom) and PCA (top), when applied to a dataset of noise behind faces.

2.2.2 Assumptions made by LLE

A linear embedding is simply a set of vertices connected to each other, and they represent a dimension. This is illustrated in Figure 7, where each dimension has a linear embedding associated with it.

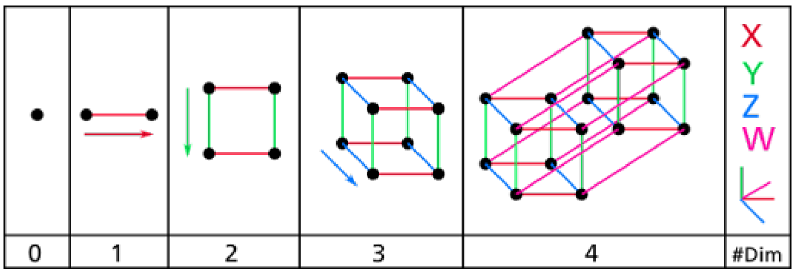
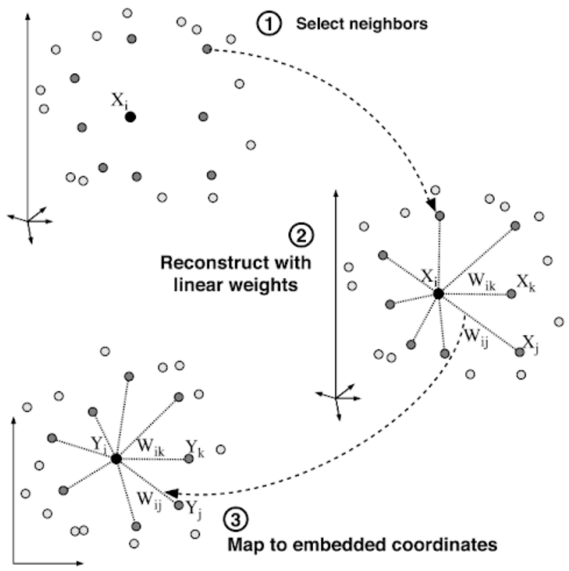


Figure 7: Showing a representation of linear embeddings. 0) A dot is a linear embedding in 0-dimensions. 1) A Line is a linear embedding in 1-dimension. 2) A square/plane is a linear embedding in 2-dimensions. 3) A cube is a linear embedding in 3-dimensions.4) A tesseract/hyper-cube is a linear embedding in 4-dimensions. And so on.

The algorithm of LLE as discussed in Section 2.2.3 , starts by obtaining a sample of k-points from the local neighbourhood of a point $\boldsymbol{x}_i$. The mathematics/method behind LLE then makes the assumption that those k-points/local-neighbourhoods lay on or near a linear embedding (hence the name locally linear embedding), most of the time this linear embedding on which local neighbourhood lay are of dimensions (D - 1). Therefore, for a 3-dimensional dataset (5) we expect that any local neighbourhood lay on a (3 - 1) dimensional linear embedding in this case it is a plane.

2.2.3 Algorithm

The algorithm behind LLE has three main parts, with goal being to reduce a D-dimensional vector to d-dimensions while also maintaining the relationship between local neighbourhoods of the manifold. However, it is important to understand that LLE maps from D to d-dimensions a single point/vector at a time, and this is done for every single point/vector in the dataset. The key behind maintaining neighbourhood relations comes from the use of the k-nearest neighbours to point $\boldsymbol{x}_i$. Figure 8 shows the general outline of the algorithm when a single vector is mapped to a lower dimension.



① Select neighbors
X_i
② Reconstruct with linear weights
X_i
W_ik
X_k
W_ij
X_j
Y_i
W_ik
Y_k
W_ij
Y_j
③ Map to embedded coordinates

Figure 8: Showing the outline of the LLE algorithm. 1) Selecting the k-nearest neighbours of point $\boldsymbol{x}_i$. 2) Using linear weights, we create a reconstruction of point $\boldsymbol{x}_i$. 3) Mapping the point $\boldsymbol{x}_i$ into a lower dimensional vector of d-dimensions.

The algorithm begins by calculating the k-nearest neighbours of any point i from the dataset.

2.2.3.1 *k*-nearest neighbours

The first part of the algorithm is to simply find the k-nearest neighbours. The number of nearest neighbours for the point $\boldsymbol{x}_i$ is given by k, which is a hyperparameter of LLE. Once k is defined, we find the k-nearest neighbours by calculating the Euclidean distances (Equation 2.8) between $\boldsymbol{x}_i$ and every other point in the dataset, then selecting the k-minimum distances observed. Each of those points is then assigned a weight, the weights of all k-nearest neighbours are then used to calculate a reconstruction of point $\boldsymbol{x}_i$.

$$d(p,q) = \sqrt{\sum_{i}^{D}(q_i - p_i)^2} \tag{2.8}$$

2.2.3.2 The weights

Following the calculation of the k-nearest neighbours to point $\boldsymbol{x}_i$, we proceed by assigning a random weight to each of the neighbouring points. A unique property of the weights is that they must equate to one, reasons for this will be discussed later in Section 2.2.3.6. The reason behind giving each neighbouring point a weight is to create a reconstruction of point $\boldsymbol{x}_i$.

2.2.3.3 Calculating a reconstruction

The reconstruction of the point $\boldsymbol{x}_i$ is obtained using the following expression:

$$reconstruction(\boldsymbol{x}_i) = \sum_{i}^{K} W_i \vec{X}_i \tag{2.9}$$

Where $\boldsymbol{x}_i$ is our target point, W_i being the weights and X_i representing a neighbouring point to $\boldsymbol{x}_i$.

2.2.3.4 The Reconstruction error with D-dimensional vectors

Initially the weights are arbitrarily selected, while respecting the condition that they sum to one. However, LLE aims at obtaining the optimum weights that best reconstructs point $\boldsymbol{x}_i$. Those optimum weights are obtained by minimising the following reconstruction error cost function:

$$\varepsilon(W) = \sum_i |\vec{X}_i - \sum_j W_{ij}\vec{X}_j|^2 \tag{2.10}$$

where $\varepsilon(W)$ is the reconstruction error as a function of the weights, and the inner sum is the reconstruction of the unit vectors of the target point/vector, here both $\vec{X}_i$ (Target) and $\vec{X}_j$(Part of the k-neighbours) are vectors from the dataset and are known. Equation 2.10 reads as follows:

"For every unit vector of X_i do and sum the following: calculate the difference between the reconstruction (of the unit vector) and the real value of said unit vector."

2.2.3.5 Optimising the weights

Refer to appendix **??**, where a constrained least squares problem is solved in order to minimise the cost function and obtain the optimum weights.

2.2.3.6 Properties and significance of the weights

In previous Sections (2.2.3.2 & 2.2.3.4), it was briefly mentioned that the weights of the reconstruction defined in Equation 2.9 equate to one, and that this rule is significant for LLE to work correctly. The goal of LLE as mentioned at the start of this current Section 2.2 is to remap from D-dimensions to d-dimensions while maintaining local relationships between the neighbours, therefore, by design, the reconstruction weights reflect intrinsic geometric properties of the data [Saul and Roweis, 2001], those properties are invariant to the Scale, Rotation and Translation of the data. This means that the weights that reconstruct a point, are constant despite a change in those properties.

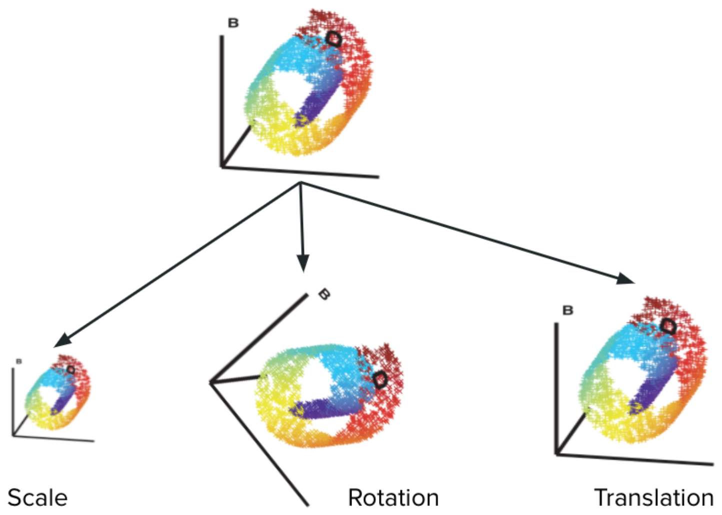


Figure 9: Showing the invariance properties of a linear manifold as a result of the reconstruction weights equating to one.

Furthermore, suppose the whole dataset lay on or near a non-linear embedding of d-dimensions, where $d << D$, we can therefore say that local neighbourhoods can be mapped onto this non-linear embedding. This remapping is done by means of re-scaling, rotating and translating those neighbourhoods. Those transformations as previously mentioned are made invariance by design through the weights, therefore, we would expect that the local geometry of the remapping in d-dimensions to be similar and valid when compared to the local geometry of the original D-dimensions manifold [Saul and Roweis, 2001].

To illustrate this lets consider Figure 10, here as captioned, the fox can represent a D-dimensional non-linear manifold, we can (non-formally) picture how we can reconstruct the fox in d-dimensions by cutting the it into smaller linear pieces, and placing them onto a non-linear manifold of d-dimensions in a logical order that respects the original higher dimension manifold i.e respecting the geometric relationships between local neighbourhood.

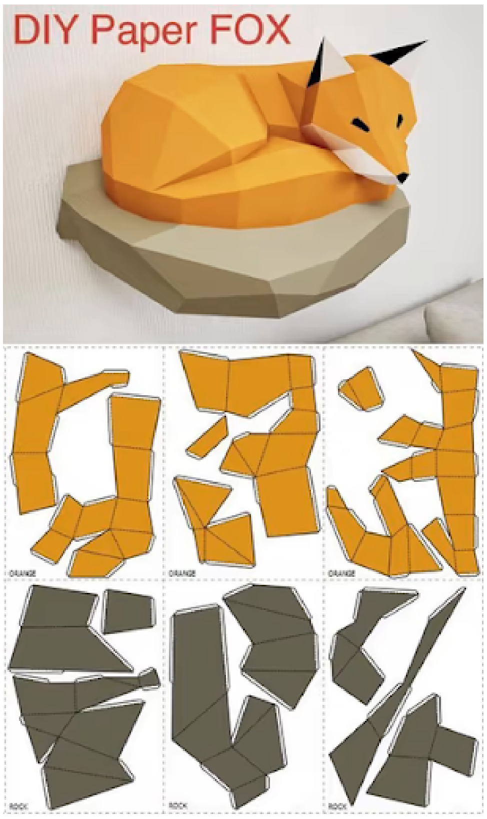


Figure 10: Showing an intuitive example of how LLE operates, the fox represents a non-linear manifold, which could be cut into pieces and placed into a d-dimensional non-linear manifold that preserves local relationships.

However, formally, this is done by recalculating the vectors that best reconstruct the point $\boldsymbol{x}_i$ in d-dimensions, while keeping the weights constant.

2.2.3.7 Calculating d-dimensional vectors

To calculate the d-dimensional vectors, the embedding cost function in equation 2.11, which takes in the weights as a constant for each point; and computes the reconstruction error for the remapped vector [Roweis and Saul, 2000], has to be minimised in order to obtain the optimum reconstructions. To minimise this embedding cost function we can solve a sparse $N \times N$ eigenvalue problem. This is detailed in Section **??**.

$$\Phi(Y) = \sum_i |\vec{Y_i} - \sum_j W_{ij}\vec{Y_j}|^2 \tag{2.11}$$

Finally, given the minimised reconstruction vectors obtained by solving the sparse $N \times N$, LLE manages to reduce the dimensions of the original D-dimension manifold into a non-linear d-dimension manifold, while also preserving geometric relationships between locally linear neighbourhoods. A further discussion about the effectiveness of LLE when applied to different data sets; and when compared to other methods, is in Section 3.

2.2.4 Optimising of k

Before applying our LLE to a data set, we consider the two parameters involved with this method: the number of nearest neighbours for each data point, k, and the dimensionality of the embedded space, d. The parameter, d, can be seen as the minimal number of degrees of freedom needed for us to generate our original toy data set from the data on the embedded space we produce from using LLE.

With any data we are visualising a $\mathbb{R}^n$ data set from an n-dimensional space to a lower-dimensional space, so we are able to see our parameter d as fixed, which means that the only parameter to be estimated before LLE on our toy data set is k.

We want to find our optimum value for k as when the number of nearest neighbours is too large, smoothing occurs where the small-scale structures across the manifold disappear. Contrary to this, if we choose a value of k that is too small, this can lead to the divide of the continuous manifold which would be the data, into disjoint sub-manifolds, which makes it hard for us to visualise our original manifold on the new lower-dimensional embedded space [Kayo, 2006].

We are therefore finding an estimate that shows our high-dimensional structure in the best possible way on our embedded space, hence we will measure the quality of our estimates on the residual variance, 2.11, of each value, defined as $1 - \rho^2_{D_X D_Y}$, where ρ is our standard linear correlation coefficient taken over all our entries from our matrices of Euclidean distance D_X D_Y of X and Y, where X and Y construct the data that has been produced on our embedded space during LLE when the reduced dimensional space is now two-dimensional, as an example. Furthermore, as we are looking for the minimal residual variance, we can determine k_{opt} as

$$k_{opt} = \underset{K}{argmin}(1 - \rho^2_{D_X D_Y \ldots}). \tag{2.12}$$

Our R function `calc_K` uses the hierarchical method for the automatic selection of the optimal number of nearest neighbours. We choose a set that could contain our k_{opt} values (without proceeding with the LLE steps) and calculates the optimal measure for each value in the set via computing residual variance. Then it finds the value with the best optimal measure, meaning residual variance is minimal [Kouropteva et al., 2002]. The function therefore can be described as follows:

- Select an interval for k that we assume contains our k_{opt} value.
- Caclulate ϵ for each k, $k \in [k_{min}, k_{max}]$, according to 2.10.
- Find all minima of $\epsilon(k)$ and corresponding k's.
- Run LLE on each value of k collected, computing the residual variance .
- Selects k_{opt} according to 2.12.

3 Application, Results and Discussion

3.1 Applying dimension reduction methods to a toy data set

This theory can now be applied to a toy data set to see how the methods discussed behave in practice and how changes in the various parameters (if applicable) affect the lower dimensional representation. A data set of a "Swiss Roll" shape with 2000 randomly selected points was generated to test out the affects of changing the different values as mentioned above [G. Krämer and Mahecha, 2018]. Figure 11 shows a 3-dimensional illustration of the data set. The goal of non-linear dimension reduction is to "unroll" the Swiss roll with local distances between points being preserved but with the general shape being removed (in this example the general shape being the curves of the Swiss roll). In practice this means that the ideal shape for the 2-dimensional representation of the Swiss roll would be a rectangular plane with it ranging from red and blue at one end to yellow and green at the other.

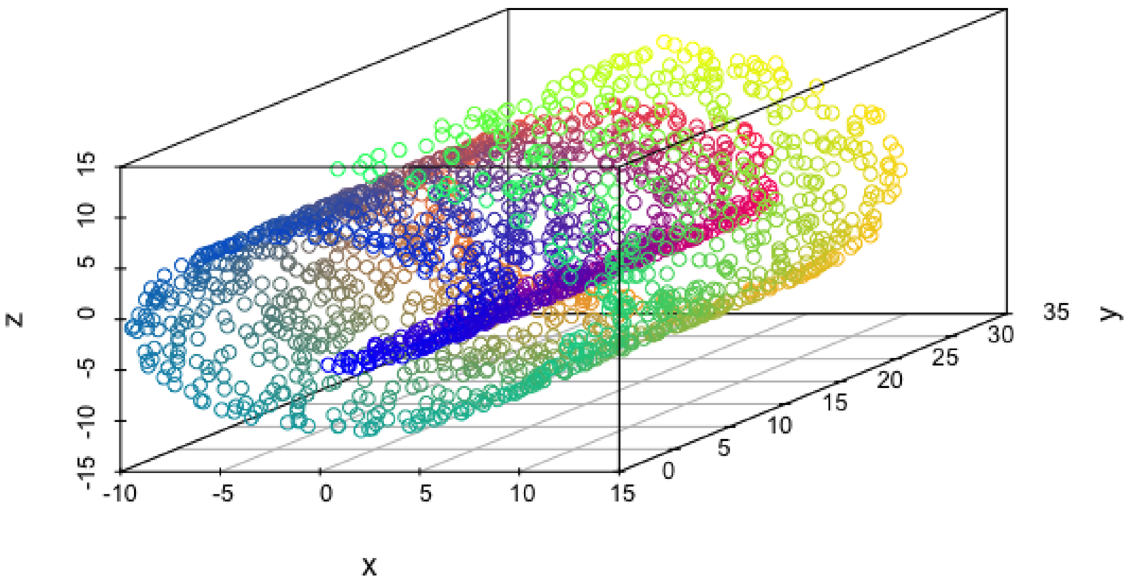


Figure 11: A toy swill roll data set with 2000 data points

3.2 PCA applied to the Swiss roll

The first thing to test is whether PCA really does produce a poor representation of the data in 2-dimensions. Figure 12 shows the representation that PCA produces for the dataset shown in Figure 11. The Figure shows that a plane which contains 2 distinct layers merged into each other (orange to red, teal to blue being the first and orange to yellow, teal to green being the second). When looking at Figure 11, this plane seems to be equivalent to producing a plane that only looks at the points in the z and y axes (an equivalent way of saying this would be to say that PCA has taken a picture while standing on the y axis), while making no attempt to unroll it. In this regard PCA has done well at preserving the global picture of the data but has performed poorly at a local level, where the non-linear change as the Swiss roll curves has been ignored. To capture this, we can apply the non-linear methods as discussed in Section 2.

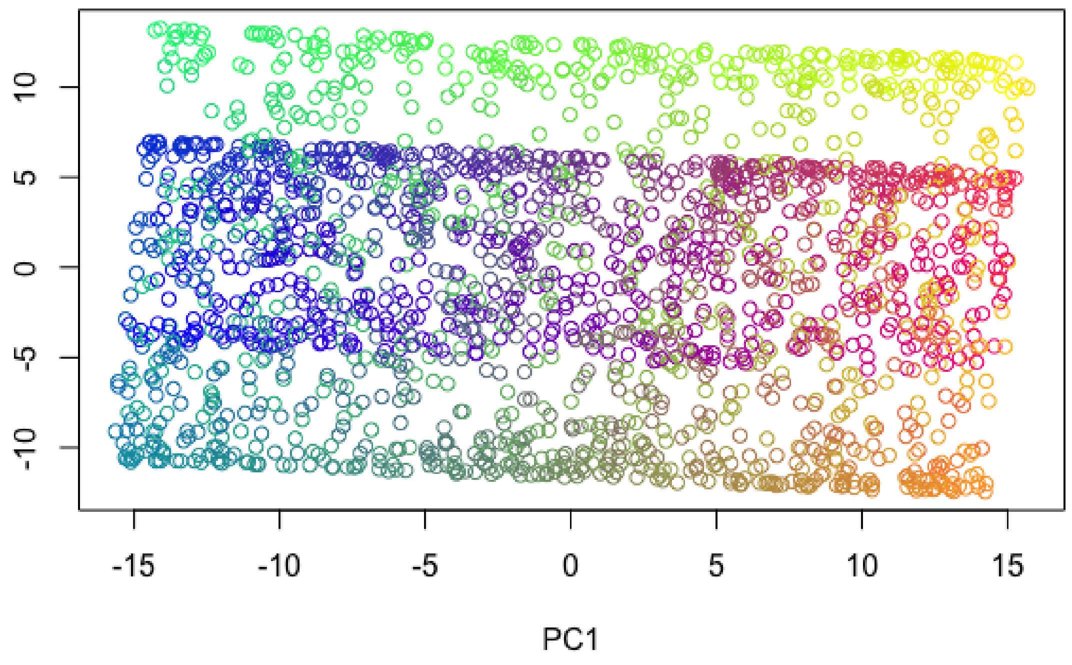


Figure 12: PCA has produced a plot which poorly represents the local distances in the Swiss roll as it bends

3.3 Laplacian Eigenmaps applied to the Swiss roll

The challenge for Laplacian Eigenmaps is to see which values of the parameters leads to the best results for this data set. Using the k-nearest neighbours approach for a fixed $t = \infty$ (this is easier to implement without prior knowledge of the distances between the points in the data), we can plot the relevant changes to k to see how the 2-dimensional representation changes. Figure 13 shows how the representation changes for varying values of k. A choice of $k = 2$ leads to an extremely poor representation of the data due to the number of links between points being so small that it leads to a disjoint graph. Increasing k slightly so that it is still small but large enough such that the graph is connected leads to a parabolic-like shape which captures the general shape of the data but ignores the width of the data. A geometric interpretation is that in Figure 11 the representation of $k = 5$ has unrolled the Swiss Roll but has taken a picture looking at the x-z plane whereas a more useful approach would be to take the picture looking at the x-y plane as this would encapsulate the length of the Swiss Roll.

As k increases, the picture seems to twist to start to show the length of the Swiss roll (though it seems to show the length of the red-blue end of the Swiss roll far quicker than the green-yellow end). This continues until a certain point where the closest neighbours of a given point include points from the "lower level" of the Swiss roll. This causes the two ends to merge (you can start to see this merger at $k = 40$) and as the $k = 80$ plot shows, the two ends have completely merged and the 2-dimensional representation is meaningless as there are too many connected points and you can no longer see the difference between the two ends of the Swiss roll.

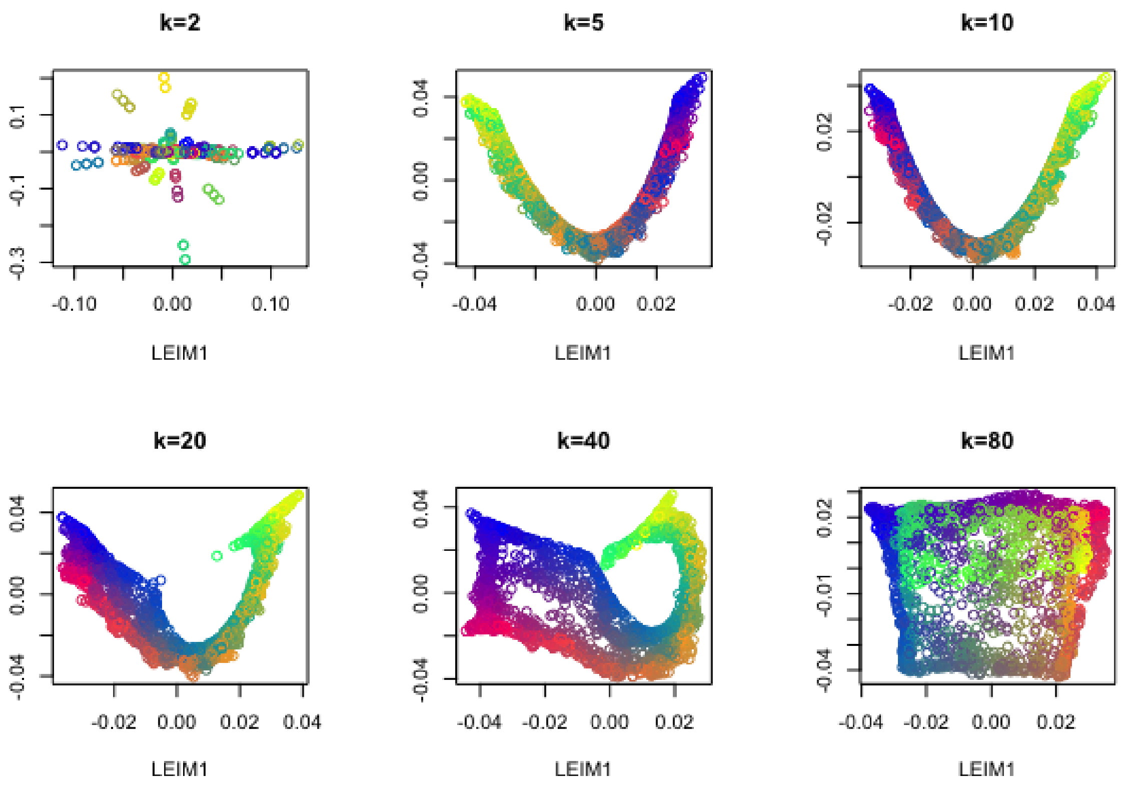


Figure 13: For a constant $t = \infty$, the effects of changing k on the 2-dimensional representation of the Swiss roll

We now have an idea of how changing k affects the 2-dimensional representation of the Swiss roll but the other variable to change is t. Figure 14 shows how various values of t affect the plot for $k = 5$ neighbours.

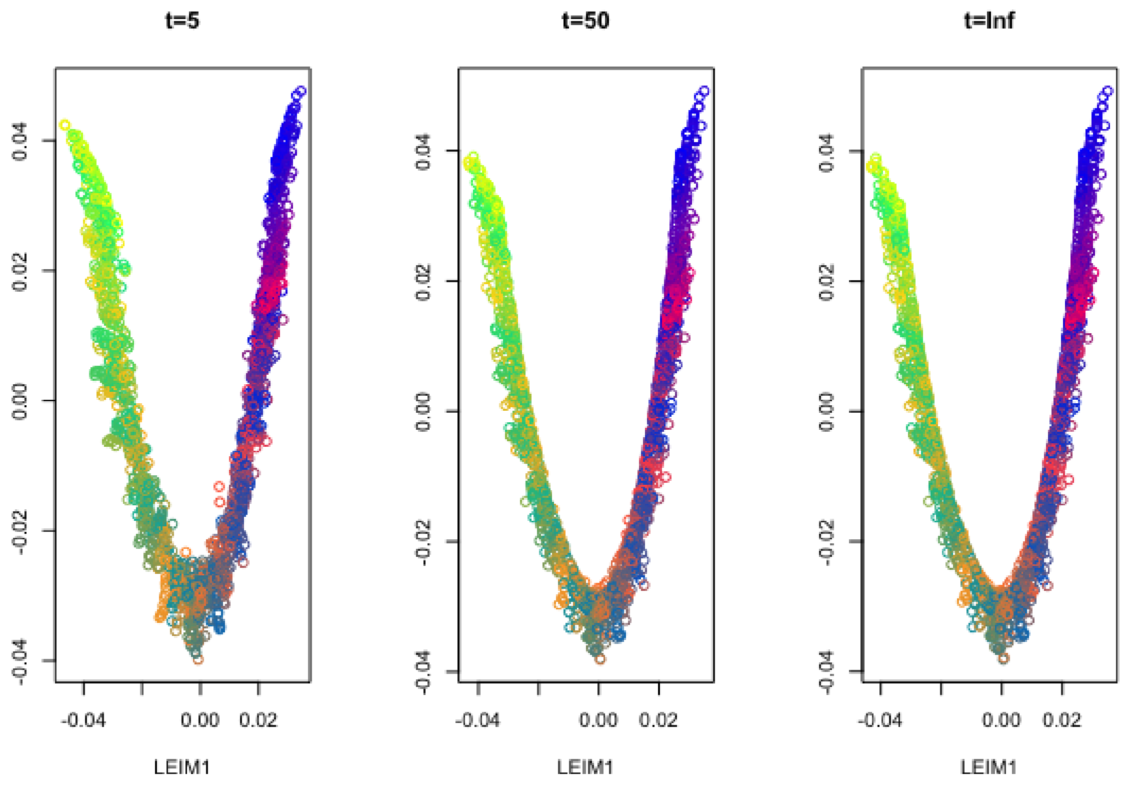


Figure 14: For a constant $k = 5$, the effects of changing t on the 2-dimensional representation of the Swiss roll

Figure 14 shows that the change in t has caused very little change in the approximation for a small value of k. This is because there are only 5 non-zero entries in the Laplacian anyway and since these values are all similarly close to one another, the change in weighting for each point makes little difference to the eigenvalues and subsequent 2-dimensional representation. We can contrast this to a larger k (where points that are connected in the graph may differ far more in their distance) where t seems to have a far greater impact on the 2-dimensional representation. Figure 15 shows this impact for $k = 40$ (the best choice presented on Figure 13). For a small t the plot is almost identical to that when $k = 5$ as the weightings are significantly higher for closer points and although there are more connected points when $k = 40$, the points whose distance is furthest away is so far away that the values in the Laplacian are not that much greater than zero, causing a representation which is similar to if the weight was zero. As t increases the points that are connected but further away have more weight applied to them and so the differences between the different k values can be more easily seen.

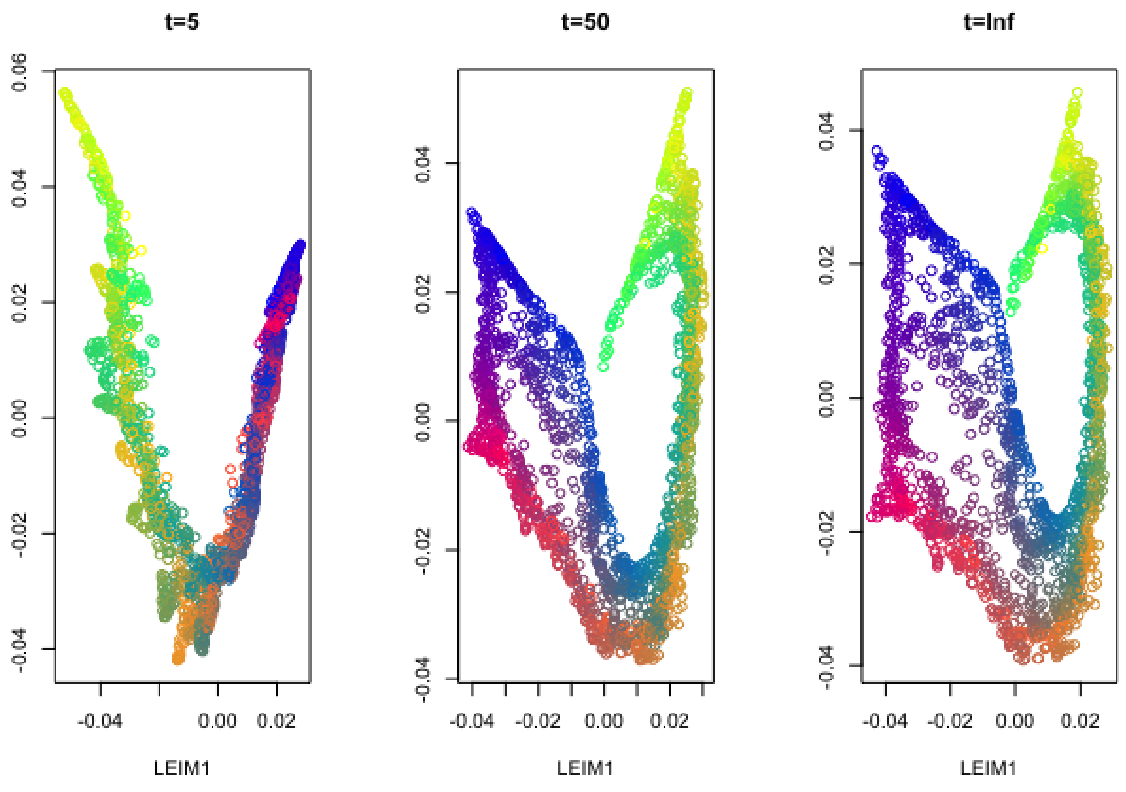


Figure 15: For a constant $k = 40$, the effects of changing t on the 2-dimensional representation of the Swiss roll

For this toy data set, due to increased weighting of further out points, it appears that a greater choice of t and k is preferred though only up to a point, seemingly selecting t to be around 2% of the total number of points (this number seems to be the cutoff when briefly applied to different number of points in the Swiss roll) as otherwise the Swiss roll ceases to be unrolled and the 2 ends merge with each other. This theory can be tested by applying plotting Equation 1.5 for different values of k and t mentioned in this section. Figure 16 [2] shows that the theory described above is matched by the data with the overall quality being highest for $k = 40$ and $t = \infty$ as well as providing a better quality measure for all but the extreme global distances.

[2]A reference of 20,5 refers to $k = 20$ and $t = 5$

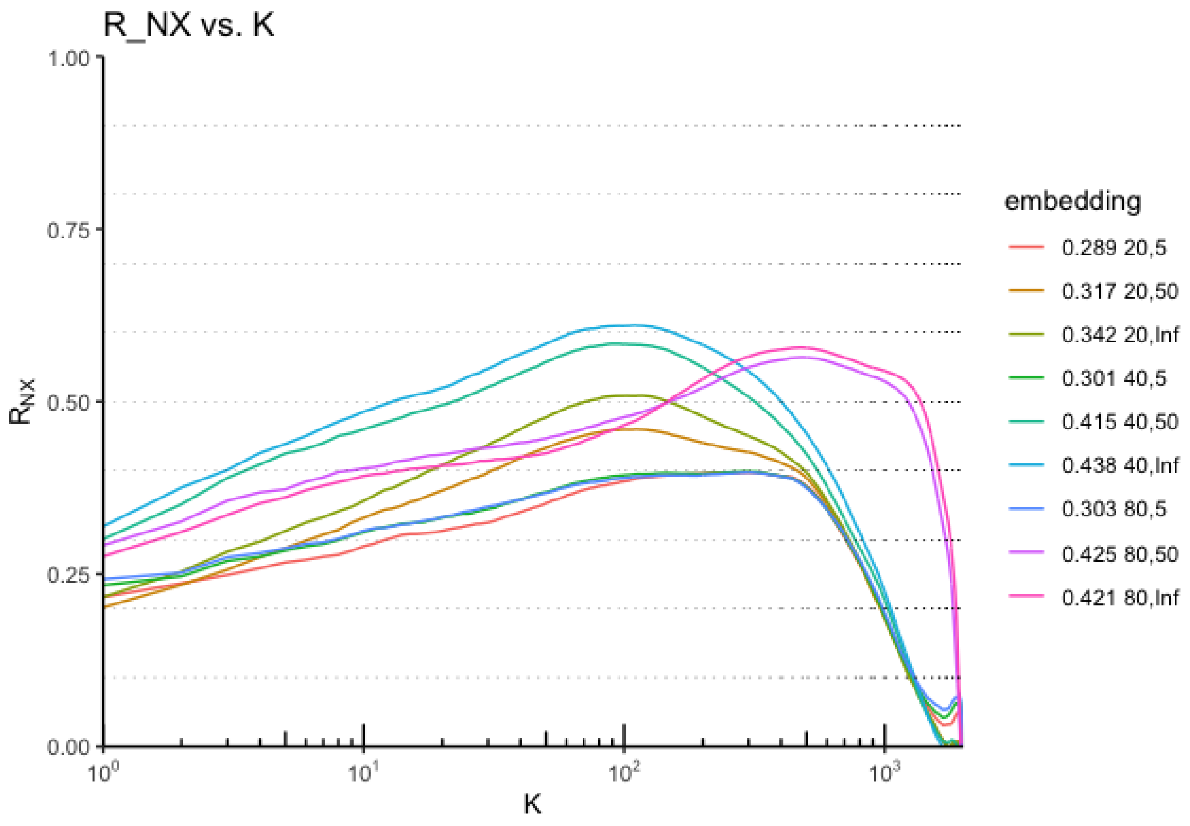


Figure 16: A choice of $k = 40$ and $t = \infty$ is the best choice here

3.4 LLE applied to the Swiss roll

3.4.1 Determining the optimal value of k-nearest neighbours

Figure 17 shows the output from using `calc_K` and makes it transparent how the optimal value of k, in our case $k_{opt} = 14$, is chosen. All that is needed to do then is to extract the corresponding k value to the minimum residual variance and then run the `lle` function with that number of nearest neighbours for our data.

However, in R, the `lle` function does not use the sparsity matrix of the function, so by finding an optimal value of k and simulating a well sampled manifold, like our Swiss roll data set, we have reduced the error of using this `lle` package.

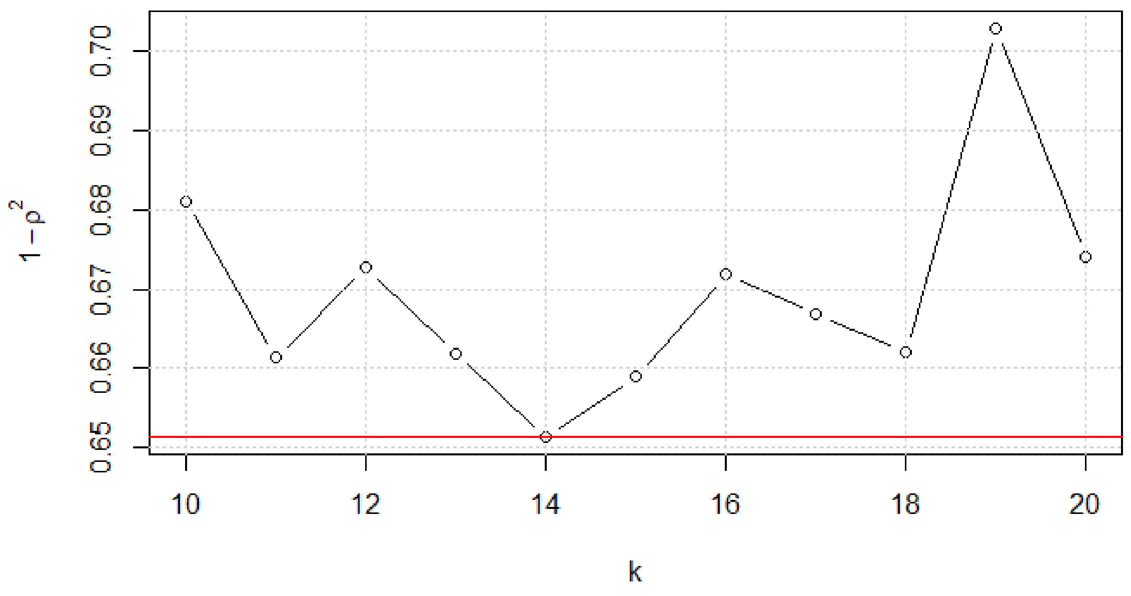


Figure 17: By providing an interval we believed to contain k_{opt}, we produced the minima of residual variance for all values of k to see which was the optimal choice of k, for the swiss roll data $k_{opt} = 14$.

3.4.2 Applying LLE to the Swiss roll

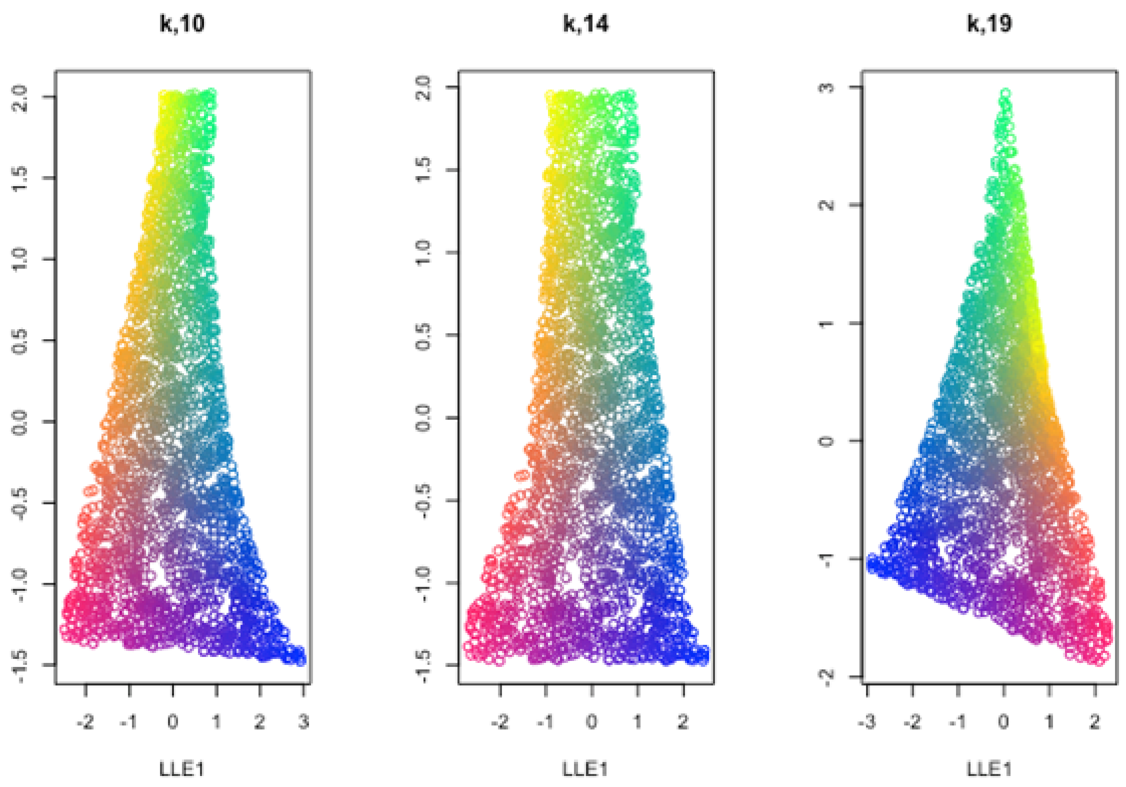


Figure 18: Using LLE with different values for number of nearest neighbours, $k = 10, 14, 19$. $k_{opt} = 14$ so as expected our data on the embedded space when number of nearest neighbours for LLE is 14 was our best method.

Figure 18 shows how LLE with optimal number of nearest neighbours produces our toy data onto a lower dimension embedded space. Using Figure 17 we also chose to produce plots when using non optimal number of nearest neighbours to compare. It is apparent that it was effective to calculate the optimal number of nearest neighbours as our manifold for $k = 14$ produced a plot of data that retains the shape of our unraveled Swiss curl while still incorporating the characteristics of our data, clearly preserving the local distances between neighbouring data points. On the contrary, these number of neighbours were expected to all produce somewhat well preserved plots as their residual variance from figure 17 only varied by roughly 0.05.

3.5 Plotting R_{NX} for Swiss roll dimension reduction

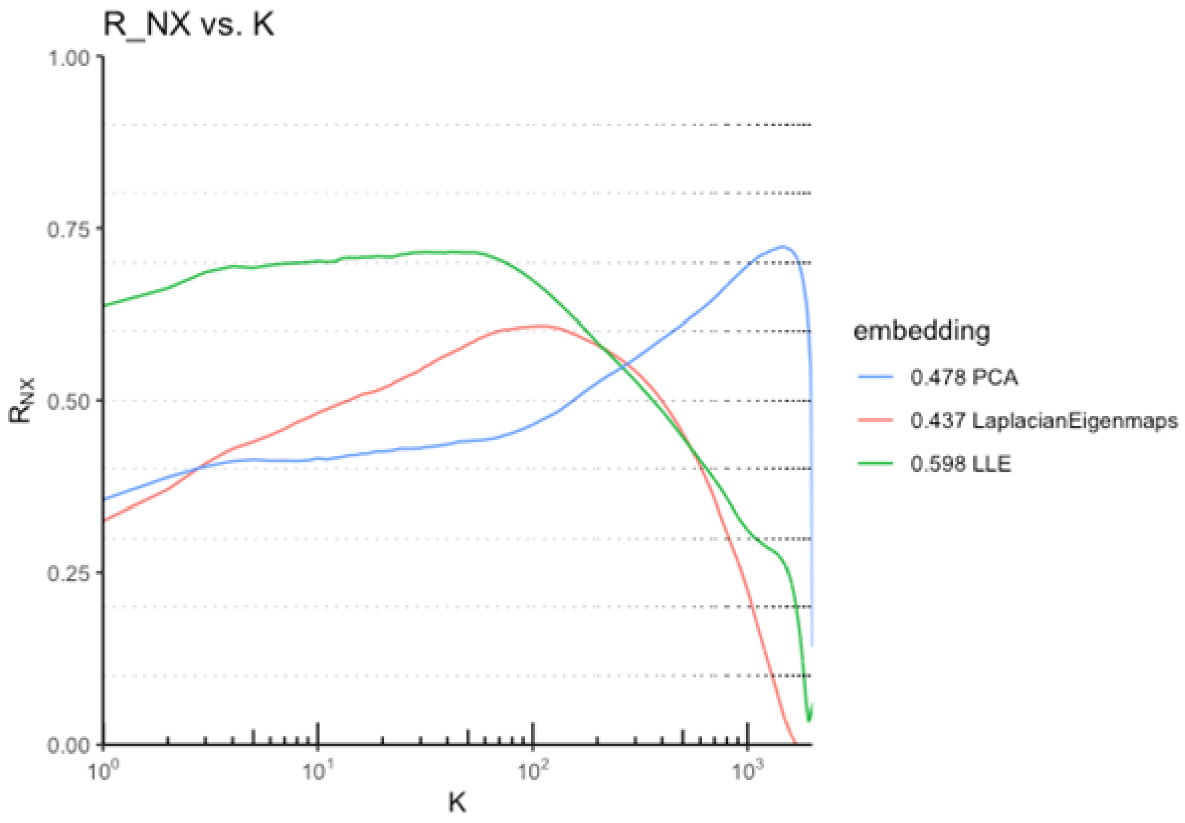


Figure 19: The area under the graph on the left hand side suggests that LLE is the best at preserving distances locally while PCA is the best at preserving global distance across the data set, suggested from the area under the right hand side of the graph.

Figure 19 shows us how well distances are preserved within our Swiss roll data when we follow through with dimension reduction techniques via the area under the graph. The left hand side portrays efficiency at more local distances while the right hand side shows how well distances are preserved more globally across the data set, meaning how well the overall shape of the data set is retained.

We can therefore suggest that LLE was the best technique for dimension reduction as we are more focused on preserving local distances so that we can keep the characteristics of the small-scale structures within our data better.

3.6 Testing NLDR on real data sets

3.6.1 DrivFace dataset

We can now apply these methods to real data sets to see how they perform in practice. The first data set we considered consisted of 606 80x80 pixel pictures of four different people driving in real scenarios, with variations in the direction that the head is tilted [Diaz-Chito et al., 2016]. For clearer pictures the drivers also had distinctive facial features such as glasses. Figure 20 shows the 2-dimensional representation of the data from Laplacian Eigenmaps using the optimal choice of the parameters (these were found using similar methods to that in Section 3.1). Example images are also overlaid to show an illustrative example of what the different points in the plot correspond to.

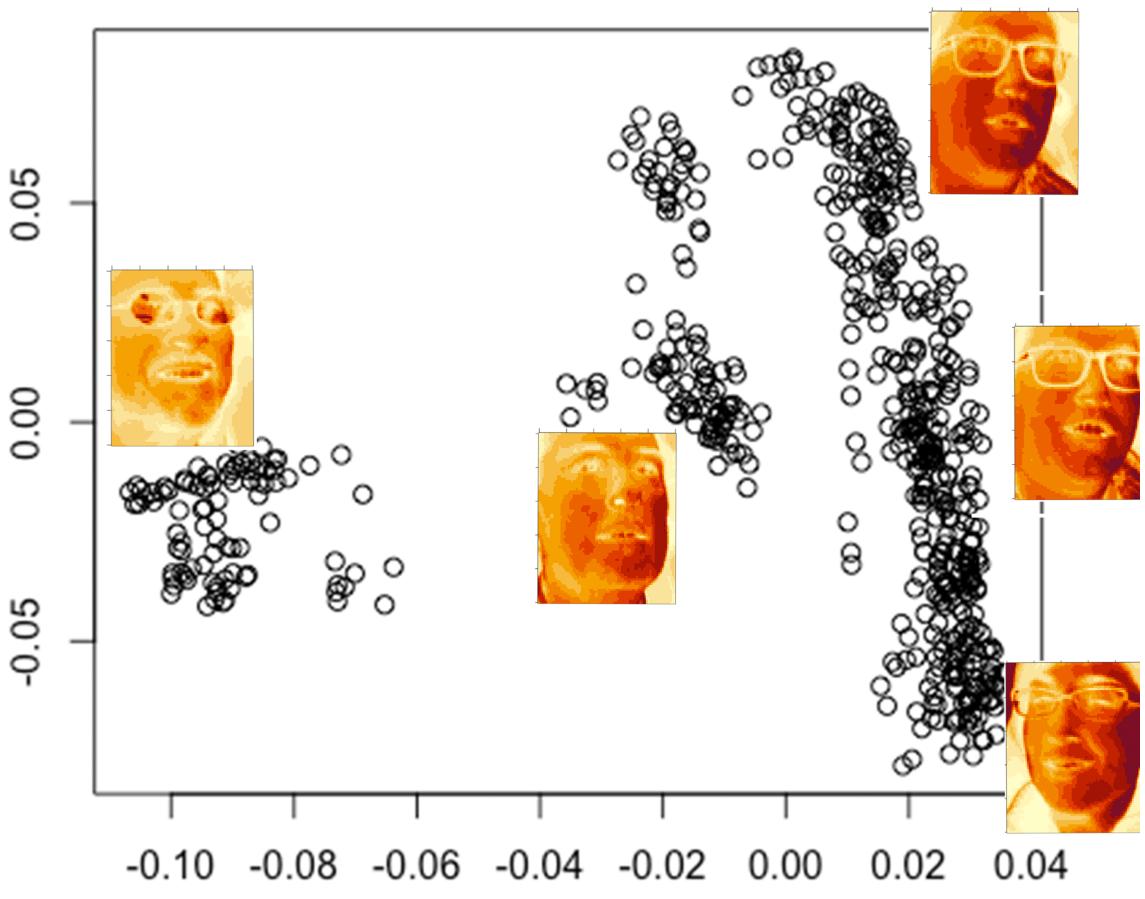


Figure 20: Laplacian Eigenmaps applied to face data with parameters. ε =1000, t=500.

Figure 20 shows that Laplacian Eigenmaps has separated the data into three distinct clusters (these will be referred to as the left, middle and right clusters) with the x axis appearing to represent the brightness of the image (with this seemingly also related to which driver the picture is of) and the y axis representing the direction the face is looking. The clusters are separated however with the left and the middle clusters being noticeably different people to that of the right cluster. The right cluster also appeared to have merged two different people into the same cluster with the shape of the glasses changing from the top image on the right to the bottom. This suggests that while the people are different, the images are similar enough (at least compared to the other two people) that Laplacian Eigenmaps has treated them as the same person. Figure 21 shows how Laplacian Eigenmaps and LLE compares to PCA for this dataset by plotting the R_{NX} function as discussed in Section 1.4.

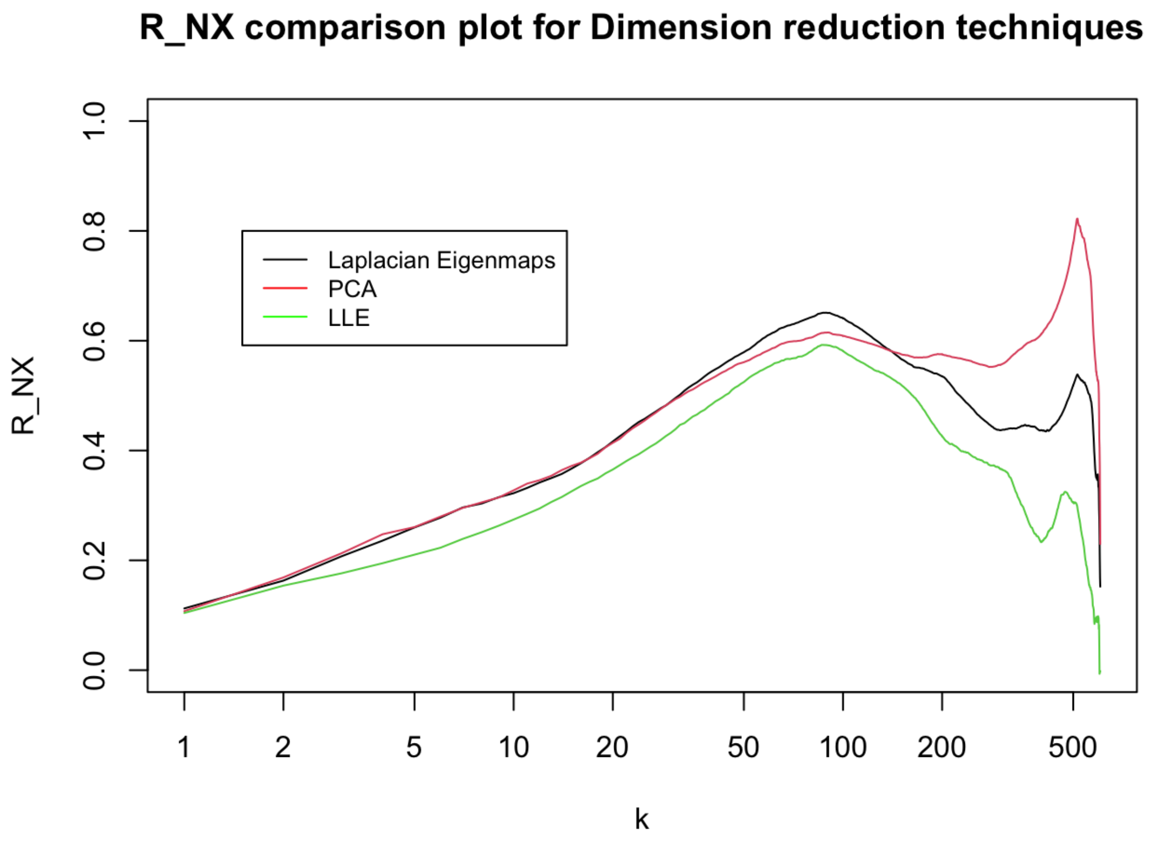


Figure 21: R_{NX} comaprison for the three dimension reduction techniques for the face data.

Figure 21 shows that the three methods are almost identical for local distances with all three methods providing remarkably poor representations of the data. This is likely due to the fact that the manifold isn't continuous (as shown by the disconnected points in Figure 20) and so Laplacian Eigenmaps and LLE both struggle to separate the points out due to them already being separated anyway. As expected, PCA performs better for global distances and so if the data is disconnected in any way (such as pictures of multiple people) then the data can't be projected onto a manifold and so PCA is a better method to use.

3.6.2 MNIST dataset

The second dataset we considered was the MNIST dataset used in the labs. This dataset consists of 200 16x16 pixel hand drawn images of integers from 0 to 9. This dataset is often used as a way of producing a model to predict the value of future drawings though this consists of a dimension reduction step and thus allows us to compare how different methods perform when reducing the dimension. Figures 22, 23 and 24 show how the different methods represent this data in 2 dimensions, with the labels representing what digit each drawing was supposed to represent.

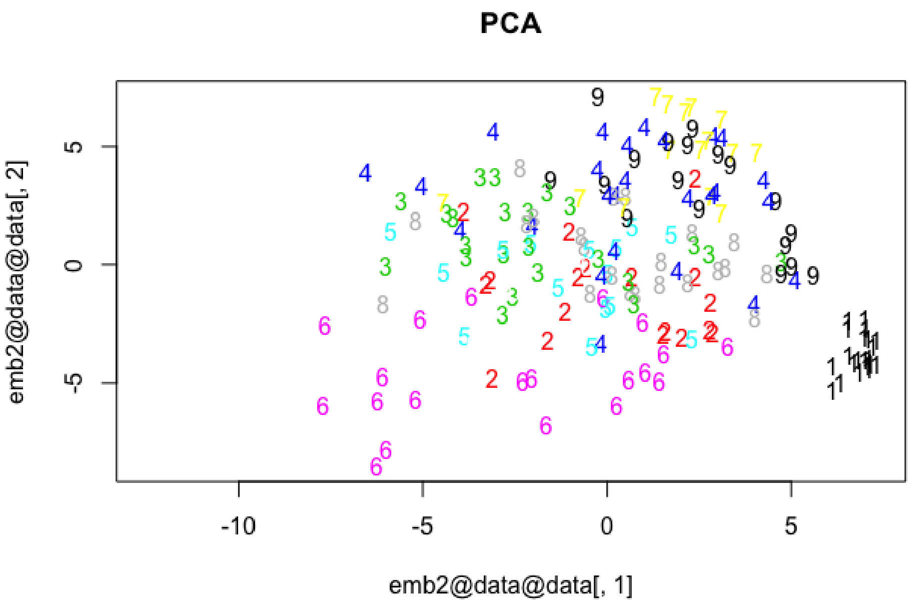


Figure 22: PCA applied to the MNIST data

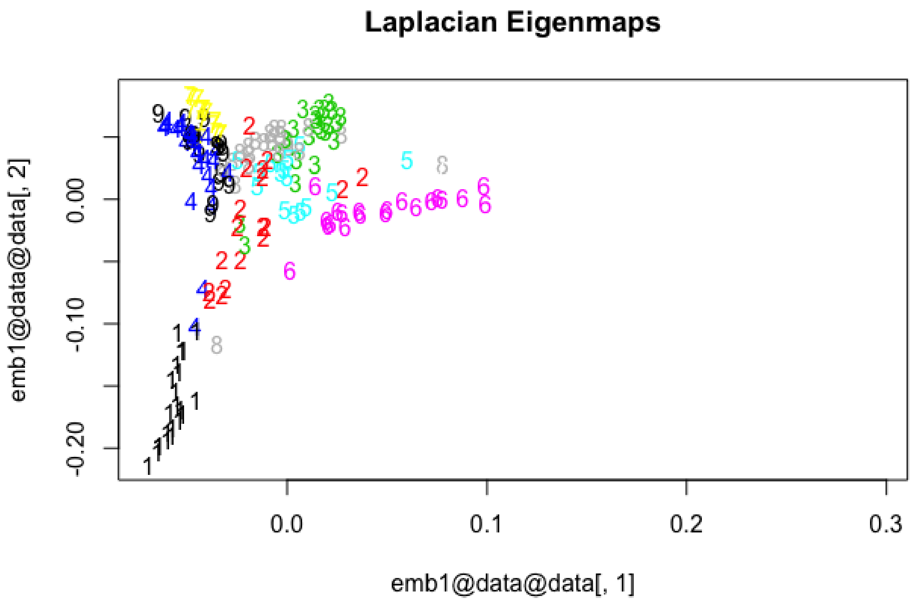


Figure 23: Laplacian Eigenmaps applied to the MNIST data

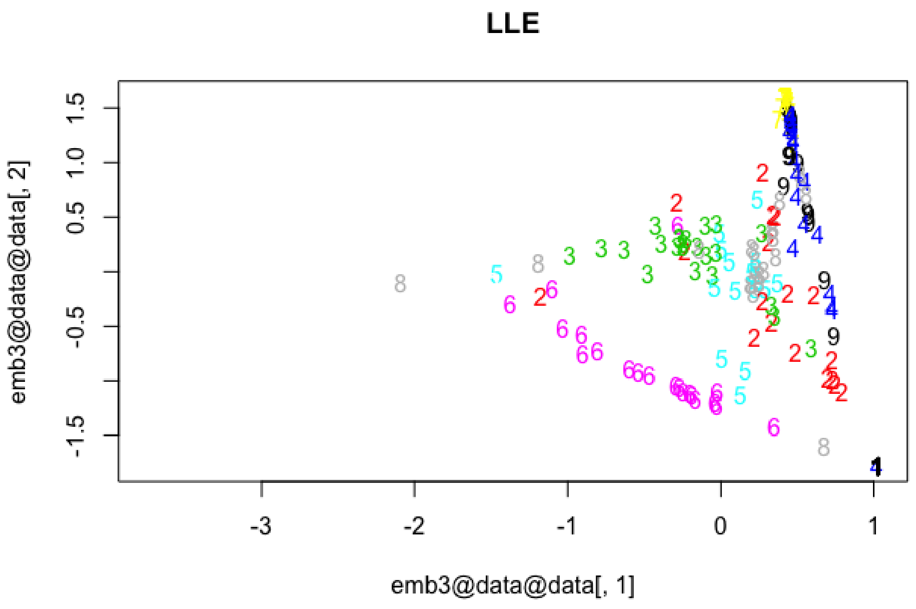


Figure 24: LLE applied to the MNIST data

While all the methods seem to separate the 1s well, Figures 22, 23 and 24 show that the other digits are far less defined into clusters for PCA compared with Laplacian Eigenmaps and LLE, with the latter two methods showing far greater distinction between the numbers (for example numbers 3 and 6). As before, the $R_N X$ plot was also computed to see whether the visual interpretation matched the reality of how good the methods performed.

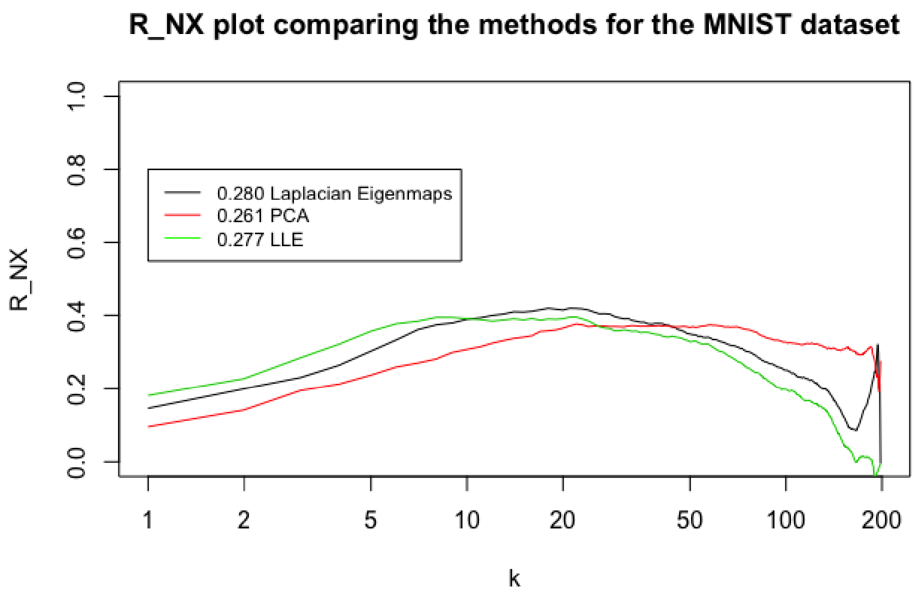


Figure 25: R_{NX} plot for the various methods when applied to the MNIST data

While all the graphs have a similar area under the curve, Figure 25 shows that the non-linear methods perform better than PCA for the closest neighbours as expected and like the Swiss roll toy data, this shows that if the data can be easily projected onto a manifold (it seems that this data can be) then NLDR methods are superior to PCA.

4 Conclusion

This investigation found informative embeddings for the Swiss roll toy dataset using Laplacian Embedding and Locally Linear Embedding. The dataset was chosen to highlight the power of these methods against the standard PCA technique. We found the methods to be very sensitive to hyperparameters so future work should involve using grid search methods to robustly find the best values otherwise insightful mappings could be missed.

The R_{NX} metric that was introduced behaves in the manner that was expected across the different methods and values of k. For large k, PCA outperforms the non-linear methods when the resulting displacement of neighbours after the dimension reduction is considered. However the opposite is true when we score the embeddings by the displacement of a low number of neighbours. Initially, we expected that PCA would fail massively because of what the method does to the manifolds visually but by this metric it's performance is comparable in terms of maintaining the pairwise relationships.

These embedding offered an ideal template to produce further work on image data. There is potential to work on a model that uses embeddings to cluster different types of images based on how well the different colourings in the Swiss roll dataset were clustered by LLE. The results of the Laplacian Eigenmapping were also promising because the method was able to adequately capture the neighborhood region of points when k was changed. This suggests that similar results will be found when working on data that is less accepting to visualisations of the whole dataset.

References

[Burago et al., 2014] Burago, D., Ivanov, S., and Kurylev, Y. (2014). A graph discretisation of the Laplace-Beltrami operator .

[C. Fefferman and Narayanan, 2016] C. Fefferman, S. M. and Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049.

[Chartrand and Zhang, 2012] Chartrand, G. and Zhang, P. (2012). A First Course in Graph Theory . pages 25–33.

[Chen and Buja, 2009] Chen, L. and Buja, A. (2009). Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 10(1):209–219.

[Cruz, 2003] Cruz, M. (2003). The Spectrum of the Laplacian in Riemannian Geometry .

[Diaz-Chito et al., 2016] Diaz-Chito, K., Hernandez-Sabato, A., and Lopez, A. M. (2016). A reduced feature set for driver head pose estimation. *Applied Soft Computing*, 45:98–107.

[G. Krämer and Mahecha, 2018] G. Krämer, M. R. and Mahecha, M. (2018). dimRed and coRanking - Unifying Dimensionality Reduction . *R. R Journal*, 10(10):342–358.

[G.Chen,] G.Chen. Math 253: Mathematical Methods for Data Visualization, Laplacian Eigenmaps .

[Jolliffe and Cadima, 2016] Jolliffe, I. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.

[Kayo, 2006] Kayo, O. (2006). Locally linear embedding algorithm: Extensions and applications. pages 33–44.

[Kouropteva et al., 2002] Kouropteva, O., Okun, O., and Pietikäinen, M. (2002). Selection of the optimal parameter value for the locally linear embedding algorithm. page 359–363.

[Lee and Verleysen, 2009] Lee, J. and Verleysen, M. (2009). Quality assessment of dimensionality reduction: Rank-based criteria . *Neurocomputing*, 72(72):7–9.

[Lee et al., 2013] Lee, J. A., Renard, E., Bernard, G., Dupont, P., and Verleysen, M. (2013). Type 1 and 2 mixtures of Kull- back–Leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing*, 112(72):92–108.

[M.Belkin and P.Niyogi, 2003] M.Belkin and P.Niyogi (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation . *Neural compuration*, 15(6):1373–1396.

[Mittal,] Mittal, R. Applications of semidefinite programming in Complexity Theory, Lecture 7: Positive Semidefinite Matrices .

[Roweis and Saul, 2000] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[Saul and Roweis, 2001] Saul, L. and Roweis, S. (2001). An introduction to locally linear embedding. *Journal of Machine Learning Research*, 7.

[Spielman, 2015] Spielman, D. (2015). Spectral Graph Theory, Lecture 2: The Laplacian .

[Urakawa,] Urakawa, H. Geometry of Laplace–Beltrami Operator on a Complete Riemannian Manifold . *Progress in Differential Geometry.*