Università degli Studi di Padova

Department Of Civil, Environmental and Architectural Engineering

Corso di Laurea Magistrale in Mathematical Engineering

Non Linear Dimensionality Reduction Techniques for Classification

Advisor:

Chiar.mo Prof. Martino Grasselli

Graduation thesis of :

Nicola Tommasini

1238784

Anno Accademico 2021/2022

Acknowledgements

I would primarily thank my family, that have given me the opportunity to follow my dreams, encouraging and supporting me day by day. Then, I would express all my thankfulness to Prof. Martino Grasselli both for having guided in this thesis project and through this Master's Degree program.

Finally, I would like to dedicate a special thanks to Beatrice for supporting me and all my friends for having accompanied me on this journey.

Abstract

This thesis project concerns on dimensionality reduction through manifold learning with a focus on non linear techniques.
Dimension Reduction (DR) is the process of reducing high dimension dataset with d feature (dimension) to one with a lower number of feature p ($p \ll d$) that preserves the information contained in the original higher dimensional space. More in general, the concept of manifold learning is introduced, a generalized approach that involves algorithm for dimensionality reduction.
Manifold learning can be divided in two main categories: Linear and Non Linear method. Although, linear method, such as Principal Component Analysis (PCA) and Multidimensional Scaling (MDS) are widely used and well known, there are plenty of non linear techniques i.e. Isometric Feature Mapping (Isomap), Locally Linear Embedding (LLE), Local Tangent Space Alignment (LTSA), which in recent years have been subject of studies.
This project is inspired by the work done by [Bahadur et Al., 2017], with the aim to estimate the US market dimensionality using Russell 3000 as a proxy of financial market.
Since financial markets are high dimensional and complex environment an approach with non linear techniques among linear is proposed.

Keywords: Dimension Reduction, Manifold Learning, Non linear techniques, Isomap, LLE.

Contents

Chapter 1

Introduction

Nowadays, working with high dimensional data, in fields like data analysis, data mining, data visualization and machine learning has become very common [Lee and Verleysen, 2007]. Just imagine data taken in various topics such as images, videos, text documents, signal analysis and financial data can only be represented with high dimensional data. Very often, working and processing them is not always easy and can lead to non optimal outcome or some difficulty to interpret results. At this point, the reduction of dimensionality becomes crucial [Cayton, 2005].

Dimension Reduction (DR) is the process of reducing a high dimension dataset, $\boldsymbol{X} = \{x_1, \cdots, x_n\} \in \mathbb{R}^d$, with n feature (dimension) to one with a lower number of feature, $\boldsymbol{Y} = \{y_1, \cdots, y_n\} \in \mathbb{R}^p$ ($p \ll n$), with p ($p \ll d$), called *Embedded Space*, that preserves some important information contained in the original higher dimensional space [Wang, 2012]. This behavior can be related to manifold hypothesis, i.e. data that are represented in high dimensional space lie on a low dimensional space [Block et Al., 2021].

The embedded data can now be used both for processing system (data mining, machine learning) and to have a better visualization and understanding of them. In Figure 1.1 there is a schematic representation of the Dimension Reduction role.

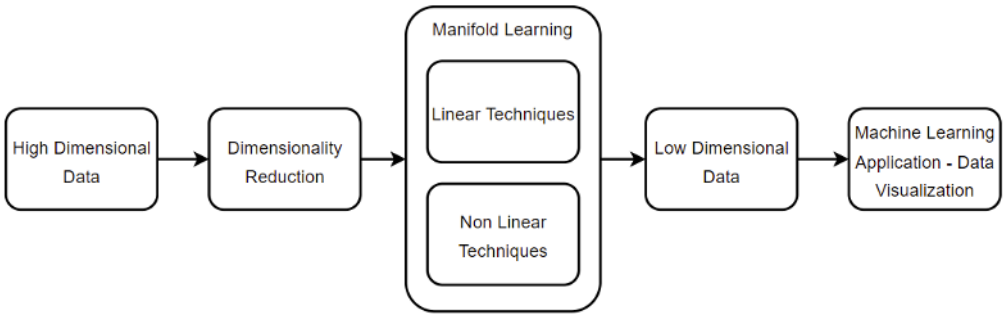


Figure 1.1: Dimensional Reduction Process.

An important research area where we focused on dimensionality reduction techniques is represented by manifold learning [Izenman, 2012]. Manifold can be divided into linear or non linear depending on the type of surface they represent. In linear techniques we find the well-known Principal Component Analysis (PCA, [Pearson, 1901]), and Multidimensional Scaling (MDS [Young, 1985]). PCA, is mainly implemented using the Singular Value Decomposition (SVD [Golub and Reinsch, (1970)]) and it reduces the dimensionality preserving the correlation structure data, conversely, MDS tries to preserve the Euclidean distance between the points. Although they are very practical and widely used techniques, given their linear nature, they cannot correctly capture complex data structures.

In 2000, two new and innovative articles published in the Science issue[1], introduce new techniques for mapping a high dimensional manifold into a low dimensional one. Specifically,the techniques proposed were Isometric Feature Mapping (Isomap) introduced by [Tenenbaum et Al., 2000] and Local Linear Embedding (LLE) introduced by [Roweis and Saul, 2000]. Isomap method relies on unfolding a manifold by keeping the geodesic metric on the original dataset applying a global approach. LLE conversely, is based on treating each point in the data set as a linearly embedding into a locally linear patch of the manifold, in a such a way to preserve the locally linear relation.

Subsequently a number of methods were presented, to name a few we have Local Tangent Space Alignment (LTSA [Zhang and Zha, 2004]), similar to LLE method with the difference that the locally linear patch is constructed by apply-

[1] *Science*, vol. 290, no. 5500

ing PCA on the neighbors, and Hessian Eigenmaps (know also as Hessian LLE - HLLE [Donoho and Grimes, 2003]) which achieves linear embedding by minimizing the *curviness* of high dimensional data by the Hessian minimization [Van der Maaten et Al., 2009].

Contextualizing Dimensional Reduction to the world of financial markets, we can see them as complex systems, expressed in a high dimensions dataset and in continuous change. Therefore, financial markets can be represented in a lower dimensional manifold having the main characteristics of the starting manifold [Huang et Al., 2016]. As an estimator of the US market the index Russell 3000 has been used, in particular its constituents represent the starting dimensionality. The idea of estimating the dimensionality of the Russell 3000 index is based on [Bahadur et Al., 2017]. In addition to the implementation of both metrics, Euclidean (MDS) and geodesic (Isomap), local methods such as LLE and LTSA, have also been implemented. Furthermore, it is referred to the original work as a stressful situation in the market therefore a drop in the index, is related to a drop in dimensionality. We will check whether this pattern has also been repeated in more recent events, such as Covid-19.

In Figure 1.2 there is an representative, but not exhaustive list of Dimensional Reduction Techniques. Covering all the techniques is almost impossible, so it was decided to focus on the first and main techniques invented, in order to have a a complete view at the first approach. For the techniques not exhibited, there is a list of references to draw from in case of deepening.

This thesis is organized as follow. In Chapter 2 are introduced a series of general concept of statistics, linear algebra,graph theory and topology. In chapter 3 and 4 linear and nonlinear reduction techniques respectively are exposed. For the nonlinear ones, where foreseen, we have chosen to expose the basic technique, followed by its kernel generalization and finally the landmark method.

The chapter 5 shows, after an introductory part useful to visually and practically show the difference between linear and non-linear techniques, the work done in

calculating the dimensionality of the US market.

Finally, 6 conclusions and some considerations are presented.

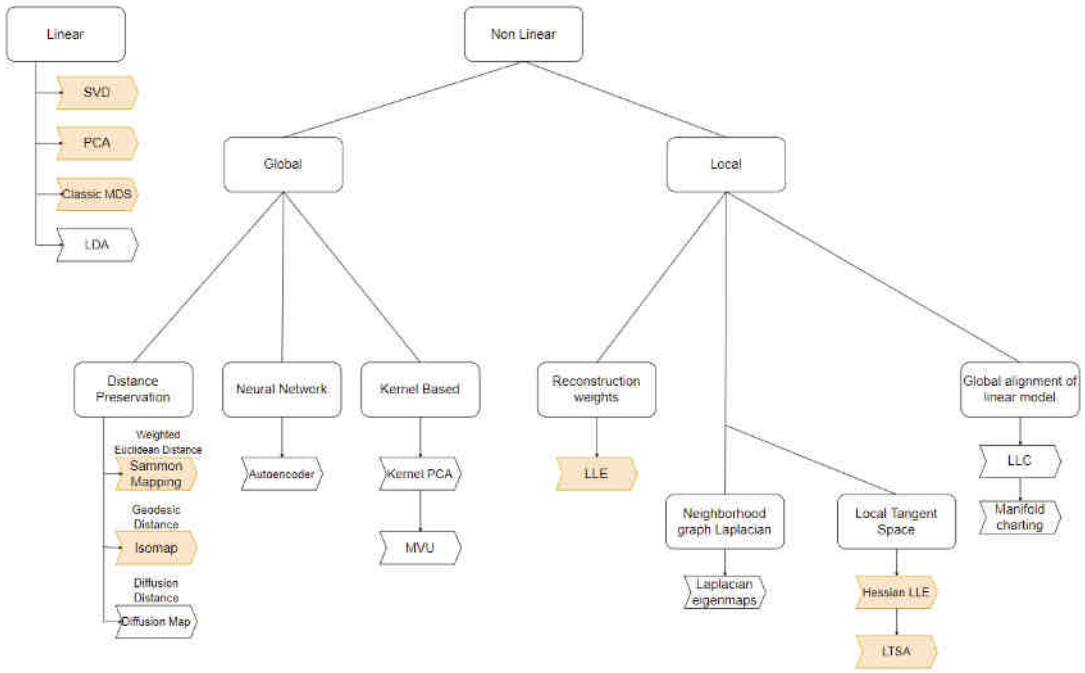


Figure 1.2: Representative, but not exhaustive list of Dimensional Reduction Techniques. The techniques discussed in this document are highlighted.

Chapter 2

Background

In this preliminary chapter some concepts graph theory and topology will be exposed and recalled in order to have a better understanding of the algorithms and techniques subsequently explained. For a more detailed introduction to graph theory, topology and differential geometry, we suggest [Wilson, 1996] and [Wang, 2012] respectively.

2.1 Graph Theory

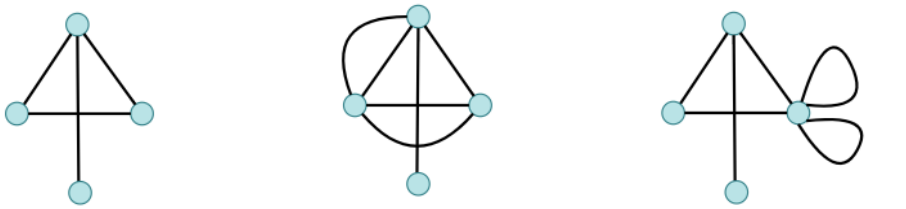
Graphs are mathematical structure used to model relations between object, and as suggest from the name, have a *graphical representation* too [Friesz Bernstein, 2016]. This section is dedicated to give some preliminary concept. As exposed later, graph theory is an essential part for the discretization of the problem [Cheng et Al., 2021]. A simple *undirected* graph $G = (V, E)$ is formed by an non empty set $V(G)$ of elements called *vertices* and $E(G)$ a finite set of distinct unordered pairs of distinct elements of $V(G)$ called *edges* [Kairanbay and Mat, 2013]. Each edge $e \in E$ is said to *join* two vertices, which are called *end points*. If e joins $u, v \in V$ then $e = \langle u, v \rangle$ [1] is said to be *incident* and vertex u and v to be *adjacent*. [Wilson, 1996]

A graph that does not have loop or multiple edge is called *simple*, instead a graph

[1] $e = \langle v, u \rangle$ since the pairs are *unordered*

that admits them, in called *non simple graph*(general graph) [Bondy and Murty, 1982].

As shown in 2.1, there the example of a simple, nonsimple with multiple edges and nonsimple with loops graph.



(a) Simple Graph

(b) Non Simple Graph with multiple edges

(c) Non simple Graph with loops

Figure 2.1: Simple, Non simple and loops graphs

Is called *weighted graph* a graph $G = (V, E)$ such that for each arc e there is a weight associated, $w : E \to \mathbb{R} | w(e) = w_e$.

Can be introduced the concept of *neighbor set* as follows:

For any graph $G(V, E)$ and vertex $v \in V(G)$ the *neighbor set* $N(v)$ of v is the set of vertices adjacent to v, i.e.

$$N(v) = \{w \in V(G) | v \neq w, \exists \in E(G) : e = \langle u, v \rangle\} \tag{2.1}$$

and we define the number of edges incident with a vertex v as *degree* of v, denoted as $\delta(v)$ (to the count of the degree, loops are counted twice). Then, for all graphs G, the sum of the vertex degree is twice the number of the edges [Wilson, 1996]

$$\sum_{c \in V(G)} \delta(v) = 2|E(G)| \tag{2.2}$$

As consequence, the number of vertices with an odd degree must be even.

A useful and appealing way to represent a graph is using the Adjacency matrix. The adjacency matrix $\boldsymbol{A}$ of a graph G is the $n \times n$ matrix $\boldsymbol{A}(G)$ whose entries

are given by [Wilson, 1996]

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

It follows that an adjacency matrix has the following properties [Chung, 1996]:

- $\boldsymbol{A}(G)$ is symmetric, that means $\forall i, j \ \boldsymbol{A}[i,j] = \boldsymbol{A}[j,i]$ due to fact that the edges are represented as an unordered pair of vertices ($e = \langle v_i, v_j \rangle = \langle v_j, v_i \rangle$)
- a graph G is simple if and only if $\forall i, j \ \boldsymbol{A}[i,j] \leq 1$ and $\boldsymbol{A}[i,i] = 0$
- the degree of vertex v_i is equal to the sum of value in row i, $\delta(v_i) = \sum_{j=1}^{n} \boldsymbol{A}[i,j]$

In figure 2.2 is shown an example of complete graph with its adjacent matrix

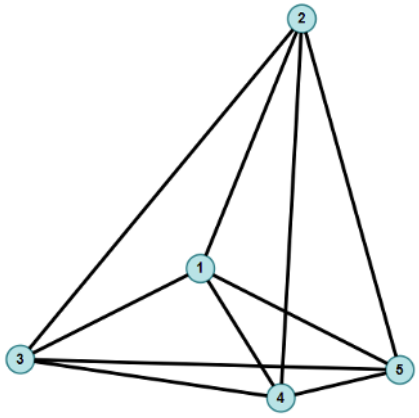


Figure 2.2: A complete Graph with 5 vertices

$$\boldsymbol{A}(G) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{2.4}$$

In relationship to the adjacent can be introduced the concept of *Spectrum* of graph [Biggs, 1993]. Let $\lambda_0 > \lambda_1 > \cdots > \lambda_{s-1}$ be the eigenvalues of $\boldsymbol{A}(G)$ and let $m(\lambda_0) > m(\lambda_1) > \cdots > m(\lambda_{s-1})$ be their multiplicities, then the spectrum of $\boldsymbol{A}(G)$ is given by

$$\text{Spec G} = \begin{pmatrix} \lambda_0 & \lambda_1 & \lambda_{s-1} \\ m(\lambda_0) & m(\lambda_1) & m(\lambda_{s-1}) \end{pmatrix} \tag{2.5}$$

Thus, considering our example in Figure 2.2 our spectrum is given by

$$\text{Spec G} = \begin{pmatrix} 4 & -1 \\ 1 & 4 \end{pmatrix} \tag{2.6}$$

since $\lambda_0 = 4$ with $m(\lambda_0) = 1$ and $\lambda_1 = -1$ with $m(\lambda_1) = 4$.

2.1.1 Nearest Neighbor Methods

Nearest Neighbor algorithms, first developed by [Fix and Hodges, 1951], is a non parametric[2] and supervised machine learning widely used in *classification*, *outlier detection* and *feature extraction* problem. During the classification stage for a given testing example, the kNN algorithm directly searches through all the examples in the set by calculating the distances between the testing example and all of the training data to identify its nearest neighbors and produce the classification output [Mitchell, 1997]. If $k = 1$, then the object is simply assigned to the nearest class, this case is simply called nearest neighbors. The core of the model relies on the definition of distance (weighted or not) and the numbers k of the neighbors.

The process of the kNN can be summed up as follows:

1. Set the number k of neighbors and d type of metric distance

2. Add new data to classify

[2]There is no assumption of the functional form of kNN

3. Calculate the distance

4. Finds neighbors and vote for labels

The distance function allows calculating the distance between point x and y in a feature space. There are many types of distance that are used in the literature, with the Euclidean one is widely used.
Let $\boldsymbol{A} = (x_1, ..., x_n)$ and $\boldsymbol{B} = (y_1, ..., y_n)$, then the most common distance metrics are [Witten et Al., 2011]:

$$\begin{aligned} \text{Euclidean} \qquad & d(\mathbf{A}, \mathbf{B}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}} \\ \text{Cosine Similarity} \qquad & d(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}||\mathbf{B}|} \\ \text{Minkowsky} \qquad & d(\mathbf{A}, \mathbf{B}) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}) \\ \text{Correlation} \qquad & d(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{n}(x_i - \mu_i)(y_i - \mu_i)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_i)^2 \sum_{i=1}^{n}(y_i - \mu_i)^2}} \end{aligned} \tag{2.7}$$

Weighted k-nearest neighbors

The Weighted k-nearest neighbors ($WkNN$) gives weights w_{ni} for each neighbor n_i. There are several way to choose the weights function [Dudani, 1976], but one of the most used is inversely proportional to the distance, that is:

$$w_{ni} = \frac{1}{d(x, n_i)} \qquad d(x, n_i) \neq 0 \tag{2.8}$$

This kind of weight function takes very large value for distances value close to zero, and thus leads in many case to the simple nearest-neighbor rule ($k = 1$) [Bicego ad Loog, 2016].

2.1.2 Shortest path problem

In many dimensional reduction algorithm (Isomap, MDS, LLE) the computation of the graph distances between points is needed. Computationally cost speaking, this process is time consuming, since we need to compute all possible path

distances (in particular for those nodes that are not directly connected by an edge). A well know iterative method develop by [Floyd, 1962] and presented in Algorithm 2.1.4 is optimally to solve the shortest path problem for dense graphs (graph with a dense adjacency matrix). However, a lot of graph related to dimensional reduction problem are constructed by neighborhood method (like kNN), therefore their adjacency matrix are sparse. In this case, for sparse matrix, the Dijkstra's Algorithm ([Dijkstra, 1959]) is proposed in Algorithm 2.1.3.

2.1.3 Dijkstra's Algorithm

Let $G = (V, E)$ a directed weighted graph and $v_0 \in V$ the initial vertex, then, the shortest path (lowest effort) from v_0 to all vertices can be found using the Dijkstra's Algorithm ([Dijkstra, 1959]).
Let the weight of path p be the sum of the weights of its edges, that is

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \tag{2.9}$$

Now, we define the *shortest path weight* as

$$\delta(u, v) = \begin{cases} min\{w(p)\} & \text{for a path pfrom}u\text{to}v \\ \infty & \text{otherwise} \end{cases} \tag{2.10}$$

and the *shortest path* from a vertex u to vertex v is any path p with weight $w(p) = \delta(u, v)$
The main steps of the algorithm can be summarized as follows [Wang, 2012]:

1. Initialize the algorithm by assigning to every node a distance value. The starting node, namely Node 1 (v_0) is set to have a distance zero,while the distances of all other nodes are set to infinity

2. calculate all tentative distances from initial node to its neighbors and mark out the starting node (is not required anymore)

3. select the node with the shortest distance and update (add to the previous one) the distance calculation from this node

4. once we have moved to the next node we compute and update all the distance of the neighbors node, storing them as tentative distance. Then, select the node with the shortest distance and re-update all the distances from this point. If the new tentative distance are smaller then the previous one select the node, otherwise select the previous node with the shorter distance.

5. iterate the process until to reach the desire node.

Algorithm 1 Dijkstra's Algorithm

Input: $G(V, E)$ graph, W weight matrix, v_0 starting point
Output: S Shortest path
1: Initialize: **for each** $v \in V$: $dist(v) := \infty$ and $precedent(v) := null$
2: $dist(s) := 0$, $Q := 0$
3: **for each** $v \in V$ insert v in Q setting $dist(v)$ as key
4: **while** $Q \neq 0$ **do**:
4.1 $u := min_{v \in Q} dist(v)$, $Q := Q - \{u\}$
4.2 **for each** neighboor v of u:
4.2.1 **if** $dist(u) + w(u, v) < dist(v)$ **then**
1. $dist(v) := dist(u) + w(u, v)$
2. $precedent(v) := u$
3. Update Q by decreasing the key $dist(v)$ of node v

The approximately computational cost is $O(V^2 \log V)$.

2.1.4 Floyd's algorithm

The Floyd algorithm ([Floyd, 1962] also known as Floyd–Warshall algorithm) is used to find out all pairs shortest path in a given directed weighted graph. As the output result, we have a minimum distance matrix , which represent the minimum distance from any node to all other nodes in the graph. In contrast to Dijkstra's Algorithm a starting point is not needed.
Floyd algorithm can be summarized as follows [Wang, 2012]:

1. Let $d_{ij}^{(k)}$ be the weight of a shortest path from vertex i to vertex j for which all intermediate vertices are in the set $\{1, 2, \cdots, k\}$

2. Then

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\left(d_{ij}^{k-1}, d_{ik}^{(k-1)+d_{kj}^{(k-1)}}\right) & \text{if } k \geq 0 \end{cases} \tag{2.11}$$

Algorithm 2 Floyd Algorithm

Input: $G(V, E)$ graph, W weight matrix

Output: D Shortest distances matrix

1: Initialize: **for each** $v \in V$ and $u \in V$:

$d[v, u] = w[v, u]$, $precedent[v, u] := null$

2: **for** k in V:

for v in V:

for u in V:

if $d[u, v] > d[u, k] + d[k, v]$:

$d[u, v] = d[u, k] + d[k, v]$

$precedent[u, v] = k$

3: return D

The computational cost for the Floyd's algorithm is $O(V^3)$.

2.2 Topology Concepts

Intuitively, a manifold $\mathcal{M}$ is a generalization of curves and surfaces to higher dimensions. It is locally Euclidean in that every point has a neighborhood, called a chart, homeomorphic to an open subset of $\mathbb{R}^n$ [Robbin and Salamon, 2021]. The coordinates on a chart allow one to carry out computations as though in a Euclidean space, so that many concepts from $\mathbb{R}^n$, such as differentiability, point-derivations, tangent spaces, and differential forms, carry over to a manifold [Tu,

2010].

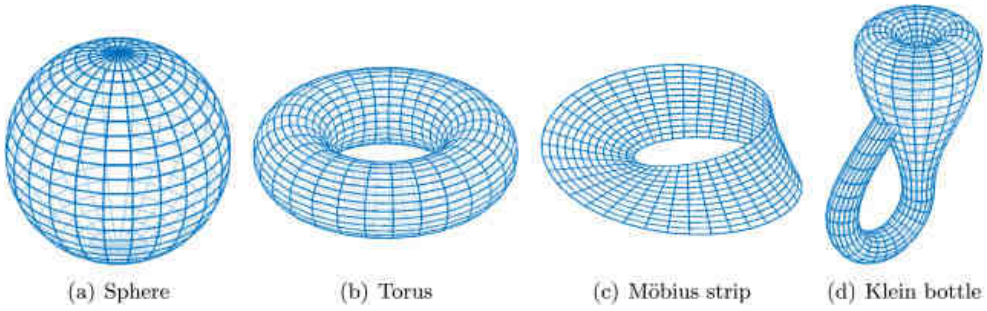


Figure 2.3: Various examples of manifolds [Deng et Al., 2020]

2.2.1 Linear Manifold

The simplest manifold is a linear manifold, often called a hyperplane. A linear manifold is a local enlargement of a nonlinear manifold. Indeed, at each point of a nonlinear manifold, there exists a tangent space $T_{\mathcal{M}}$, which locally approximates the manifold.2.4. The *Tangent Space* is in reality a linear manifold.

Linear Dimensional Reduction methods are based on the assumption that the

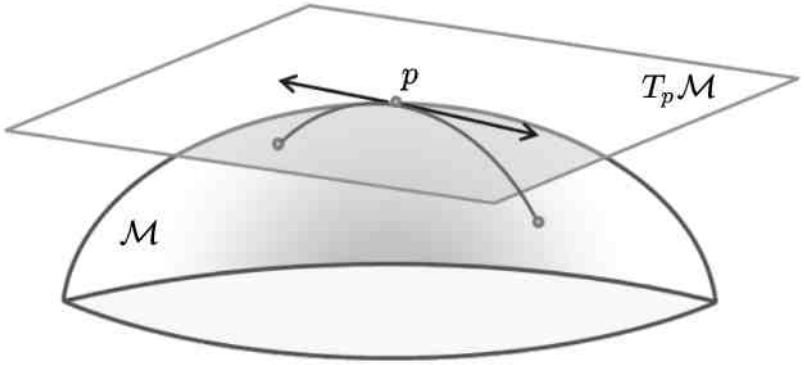


Figure 2.4: Tangent Space $T_p\mathcal{M}$ of manifold $\mathcal{M}$ on point p

observed data set resides on a linear manifold.

A briefly introduction and review of some notions and notations is presented. The coordinates of a point $\boldsymbol{x} \in \mathbb{R}^D$ in the Euclidean space are denoted by $\boldsymbol{X} = [x_1, \cdots, x_D]^\top$. A finite set of point $\{\boldsymbol{x_1},\ cdots, \boldsymbol{x_n}\} \subset \mathbb{R}^D$ is mainly expressed in

its matrix form, that is $\boldsymbol{X} = [\boldsymbol{x_1}, \cdot, \boldsymbol{x_n}] = [x_{ij}]_{i,j=1}^{D,n}$. $\boldsymbol{X}$ also spans a subspace of $\mathbb{R}^D$ denoted by $S = span\{\boldsymbol{x_1}, \cdots, \boldsymbol{x_n}\}$, a column space of the matrix $\boldsymbol{X}$.

2.2.2 Differentiable Manifold

Differentiable manifold generalizes differentiable curves and surfaces to n dimensional space. Let $X \subset \mathbb{R}^k$ and $Y \subset \mathbb{R}^l$. We define a *smooth map* $f : X \to Y$, if all partial derivatives $\frac{\partial^s f}{\partial x_{i_1} \cdots \partial x_{i_s}}$ $s \in \mathbb{Z}$ exists and they are continuous, $f \in C^\infty$ [Boothby, 1975].
Consequently, f is said to be a *differentiable homeomorphism* if both
$f : X \to Y$ and its inverse f^{-1} are smooth, and, X is said to be *diffeomorphic* to T.
Now consider for $M \subset \mathbb{R}^m$ a non empty set e let's consider that for each point $\boldsymbol{x} \in \mathcal{M}$ there is an open set $W \subset \mathcal{M}$ such that W is diffeomorphic to an open set $U \subset \mathbb{R}^k$. $\mathcal{M}$ is said to be a k-dimensional differentiable manifold and the diffeomorphism $g : U \to W$ is called parameterization of W. The inverse of g exist, denoted by $h(g^{-1}) : W \to U$, and i called *coordinate mapping*, W is called *coordinate neighborhood* and the couple (W, h) is the local coordinate system or *chart* on $\mathcal{M}$ (see Fig. 2.5) [Robbin and Salamon, 2021].
In a local coordinate system (W, h) a point $\boldsymbol{x} \in W$ is expressed as

$$h(\boldsymbol{X}) = \left[h^1(\boldsymbol{x}) \cdots, h^m(\boldsymbol{x})\right]$$

Finally, a differential structure or *Atlas* on a k-manifold can be seen as a "glued" collection of all coordinate systems $\{(W_i, h_i)\}$ on $\mathcal{M}$ and satisfying the following condition [Wang, 2012]:

- The union of W_i covers $\mathcal{M}$: $\mathcal{M} \subset \cup_i W_i$
- For each pair (i, j),$h_j \circ h_i^{-1}$ is a smooth mapping on $h_i(W_i \cap W_j)$

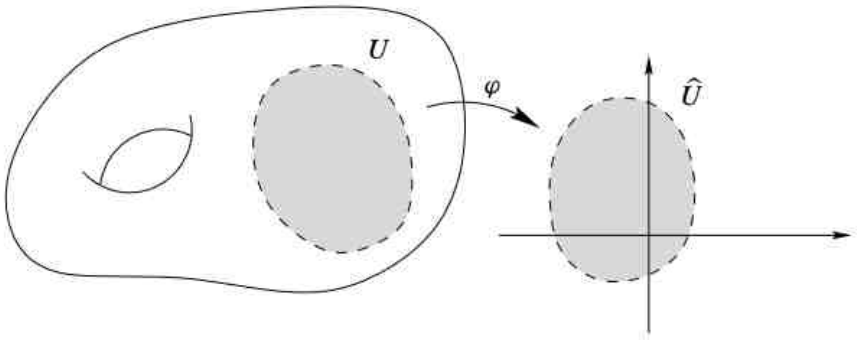


Figure 2.5: Chart on a manifold

2.2.3 Tangent Spaces

Let $\mathcal{M} \subset \mathbb{R}^m$ be a differentiable smooth manifold. The best approximation of $\mathcal{M}$ in a neighborhood of a point $\boldsymbol{p} \in \mathcal{M}$ is given by the hyperplane H [Boothby, 1975]. Recalling the derivative of a smooth mapping and identifying with gradient, i.e.

$$df_{\boldsymbol{x}} = \nabla f_{\boldsymbol{x}} = \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \cdots, \frac{\partial f(\boldsymbol{x})}{\partial x_k}\right] \tag{2.12}$$

then the following properties holds [Tu, 2010]:

- If $f : U \to V$ and $q : V \to W$ are smooth mapping and $f(\boldsymbol{x}) = \boldsymbol{y}$ then

$$d(q \circ f)_{\boldsymbol{x}} = dq_{\boldsymbol{y}} df_{\boldsymbol{x}}$$

- If $U \subset \tilde{U}$ and $i : U \to \tilde{U}$ is the inclusion map, then, $di_{\boldsymbol{x}}$ is the identity mapping on $\mathbb{R}^k$

- If $L : \mathbb{R}^k \to \mathbb{R}^l$ is a linear transformation, represented by the matrix $\boldsymbol{L}$, i.e. $L(x) = \boldsymbol{L}\boldsymbol{x}$, with $\boldsymbol{x} \in \mathbb{R}^k$, then $\boldsymbol{L} : dL_{\boldsymbol{x}} = \boldsymbol{L}$ is the derivative of L at each $\boldsymbol{x}$

Now, consider $\mathcal{M} \subset \mathbb{R}^m$ a k-manifold, $U \subset \mathbb{R}^k$ an open set and $g : U \to \mathcal{M}$ the neighborhood parameterization ($g(U) \subset \mathcal{M}$). Assuming, $\boldsymbol{p} \in \mathcal{M}$, $\boldsymbol{u} \in U$ and $g(\boldsymbol{u}) = \boldsymbol{p}$. Define the tangent space of $\mathcal{M}$ at $\boldsymbol{p}$ as

$$T_{\boldsymbol{p}}\mathcal{M} \stackrel{def}{=} dg_{\boldsymbol{u}}(\mathbb{R}^k)$$

where $g_{\boldsymbol{u}}$ is the image of the linear transformation.

Then, the hyperplane $H_{\boldsymbol{p}} =^{def} \boldsymbol{p} + T_{\boldsymbol{p}}$ is exactly the tangent space of $\mathcal{M}$. A vector in $T_{\boldsymbol{p}}\mathcal{M}$ is called a tangent vector.

Chapter 3

Linear Dimensionality Reduction Techniques

3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a feature extraction, dimensional reduction, noise filter linear techniques firstly developed by Pearson in 1901 [Pearson, 1901].

It is defined as an orthogonal linear transformation to find the direction of maximum variance in high dimensional data and projects data onto a new subspace of fewer dimension. An example is shown in Figure 3.1, where data were projected in a 2 dimensional space.

First of all we need to define the *Centering Matrix*, an important matrix transformation used to center data, i.e. subtracting their mean. It's defined as following [Greene, 2012]:

$$\mathbb{1}_{(n,n)}\bar{\boldsymbol{x}} = \mathbb{1}\frac{1}{n}\mathbb{1}^{\mathsf{T}}\bar{\boldsymbol{x}} = \begin{bmatrix} \bar{x} \\ \vdots \\ \bar{x} \end{bmatrix} = \frac{1}{n}\mathbb{1}\mathbb{1}^{\mathsf{T}}\boldsymbol{x} \tag{3.1}$$

where $\mathbb{1}$ is a column vector of 1 and $\frac{1}{n}\mathbb{1}\mathbb{1}^\mathsf{T}$ is a $n \times n$ matrix with all elements equal to $\frac{1}{n}$.

The values in deviation form from the mean can be expressed as follows:

$$\begin{bmatrix} x_1 - \bar{x} \\ \vdots \\ x_2 - \bar{x} \end{bmatrix} = \left[\boldsymbol{x} - \mathbb{1}_{(n,n)}\bar{\boldsymbol{x}}\right] = \left[\boldsymbol{x} - \frac{1}{n}\mathbb{1}\mathbb{1}^\mathsf{T}\boldsymbol{x}\right] \tag{3.2}$$

Since $\boldsymbol{x} = \mathbb{1}\boldsymbol{x}$

$$\left[\boldsymbol{x} - \frac{1}{n}\mathbb{1}\mathbb{1}^\mathsf{T}\boldsymbol{x}\right] = \left[\mathbb{1}\boldsymbol{x} - \frac{1}{n}\mathbb{1}\mathbb{1}^\mathsf{T}\boldsymbol{x}\right] = \left[\mathbb{1} - \frac{1}{n}\mathbb{1}\mathbb{1}^\mathsf{T}\right]\boldsymbol{x} = C_n\boldsymbol{x} \tag{3.3}$$

where C_n is the centered matrix. Its diagonal elements are all $(1 - \frac{1}{n})$ and its off diagonal elements are $-\frac{1}{n}$. C_n is singular, Symmetric Positive Definite (SPD), idempotent (so that $C_n^k = C_n$ for $k = 1, 2, \ldots$) and can be summarized as follows:

$$C_n = \begin{cases} 1 - \frac{1}{n} & \text{if } i = j \\ -\frac{1}{n} & \text{if } i \neq j \end{cases} \tag{3.4}$$

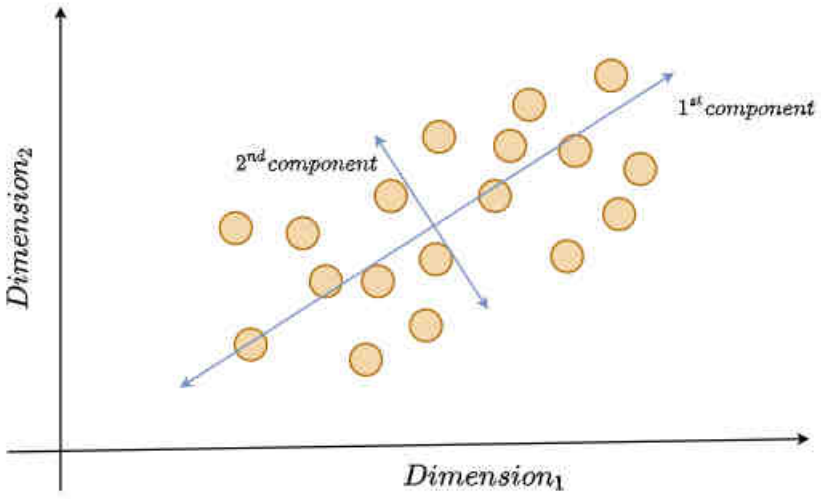


Figure 3.1: Data projection with PCA. The blue line represent the principal component (1st and 2nd component), i.e. the eigenvector of of the covariance matrix.

Let $\boldsymbol{X}$ be a $n \times p$ centered data matrix, where n is the number of samples

and p is the number of variables, then we define the covariance matrix as follows: [Granda, 2020]

$$\begin{aligned}\Sigma_{\tilde{x}} &:= \mathbb{E}\left[c(\tilde{x})c(\tilde{x}^{\mathsf{T}}\right] \\ &= \begin{bmatrix} \operatorname{Var}[\tilde{x}_1] & \operatorname{Cov}[\tilde{x}_1,\tilde{x}_2] & \cdots & \operatorname{Cov}[\tilde{x}_1,\tilde{x}_d] \\ \operatorname{Cov}[\tilde{x}_1,\tilde{x}_2] & \operatorname{Var}[\tilde{x}_2] & \cdots & \operatorname{Cov}[\tilde{x}_2,\tilde{x}_2] \\ \vdots & \vdots & \cdots & \vdots \\ \operatorname{Cov}[\tilde{x}_1,\tilde{x}_d] & \operatorname{Cov}[\tilde{x}_2,\tilde{x}_d] & \cdots & \operatorname{Var}[\tilde{x}_d] \end{bmatrix}\end{aligned} \tag{3.5}$$

The covariance matrix $\boldsymbol{C}$ of $\boldsymbol{X}$ is given by:

$$\boldsymbol{C} = \frac{1}{(n-1)}\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X} \tag{3.6}$$

where the term $\frac{1}{n-1}$ is the proper term for un unbiased normalization.
A standard and widespread decomposition used in numerical analysis, for the factorization of real or complex matrix is the Singular Value Decomposition (SVD [Golub and Reinsch, (1970)]).
It generalize the canonical eigendecomposition (i.e. $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$ where $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalue λ_i).
Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. Then the full singular value decomposition or simply SVD, of $\boldsymbol{A}$ is given by:

$$A = U\Sigma V^{\mathsf{T}} \tag{3.7}$$

$$A = \begin{bmatrix} & & & \\ U_1 & U_2 & \cdots & U_m \\ & & & \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & & \ddots & 0 \\ 0 & \cdots & & \sigma_n \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} & V_1^{\mathsf{T}} & \\ & V_2^{\mathsf{T}} & \\ & V_3^{\mathsf{T}} & \\ & V_4^{\mathsf{T}} & \end{bmatrix}$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ is an orthogonal matrix ($\boldsymbol{U}\boldsymbol{U}^{\mathsf{T}} = \mathbb{I}$) whose column $\boldsymbol{u_j}$ are

called the *left singular vectors*, $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ is orthogonal matrix ($\boldsymbol{V}\boldsymbol{V}^{\mathsf{T}} = \mathbb{I}$) whose column $\boldsymbol{v_j}$ are called the *right singular vectors* and $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ rectangular diagonal matrix. The non negative diagonal entries $\sigma_i = \Sigma_{ii}$, (with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$) of $\boldsymbol{\Sigma}$ are uniquely determined by $\boldsymbol{A}$ (are the square roots of the non-zero eigenvalues of both $\boldsymbol{A}\boldsymbol{A}^{\mathsf{T}}$ and $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$) and are known as *singular values* of $\boldsymbol{A}$.
Let perform a Singular Value Decomposition of $\boldsymbol{X}$, so

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}} \tag{3.8}$$

where $\boldsymbol{U}$ is the unitary matrix, $\boldsymbol{S}$ is the diagonal matrix of singular vales $\boldsymbol{s_i}$. The previous equation can be rearranged as

$$\boldsymbol{C} = \frac{\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^{\mathsf{T}}\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}}{\boldsymbol{n-1}} = \boldsymbol{V}\frac{\boldsymbol{S^2}}{\boldsymbol{n-1}}\boldsymbol{V}^{\mathsf{T}} \tag{3.9}$$

singular values are obtained from the eigenvalues of covariance matrix $\lambda_i \frac{s_i^2}{n-1}$.
Principal components are given by $\mathbf{XV} = \mathbf{USV}^{\mathsf{T}}\mathbf{V} = \mathbf{US}$.
The dimensionality reduction of data from dimension $\mathbf{d}$ to $\mathbf{p} < \mathbf{d}$, select first $\mathbf{p}$ column of $\mathbf{U}$ and $d \times d$ upper left part of $\mathbf{S}$. The product $\boldsymbol{U}_p\boldsymbol{S}_p$ is the $\mathbf{n} \times \boldsymbol{p}$ matrix containing first $\mathbf{p}$ principal components.
In conclusion we recall some imported assumptions [Shlens, 2003]:

- *Linearity*
- *Mean and variance are sufficient statistics*
- *The principal components are orthogonal*

The main steps can be summarized as follows [Picci, 2020]:

- *Center the data*

  First step, centering data, can be obtained by subtracting the mean of the data for each point i.e. centering matrix

- *Normalize the data*

Data normalization are useful to avoid some scale dependent side effects, especially when dimensions of data correspond to different metrics.

- *Calculate the eigendecomposition*
  Compute the eigendecomposition via Singular Value Decomposition (SVD).

- *Project the data*
  Dimension reduction can be obtained by projecting data onto the largest eigenvectors. Let $\boldsymbol{U}$ be the matrix whose columns contain the largest eigenvectors and let $\boldsymbol{X}$ be the original data whose columns contain the different observations. Then the projected data $\boldsymbol{Y}$ is obtained as $\boldsymbol{U}_p\boldsymbol{S}_p$. We can either choose the number of remaining dimensions, i.e. the columns of $\mathbf{X}$, directly, or we can define the amount of variance of the original data that needs to kept while eliminating eigenvectors.

3.2 Multidimensional Scaling MDS

In this section a brief summary of the Multidimensional scaling techniques is presented. Primarily, The Classical MDS is exposed then move towards the generalized method with the kernel or its non-metric version. Finally, the key idea for the implementation (SMACOF) of the algorithm is presented.

Classic Multidimensional Scaling (MDS) [Young, 1985], first proposed by [Torgerson, 1952], is one of the earliest proposed e developed manifold learning methods. It is a member of Multidimensional Scaling family which includes classical MDS, metric MDS, and non-metric MDS. It can be used for data visualization, data processing, data analysis and dimension reduction as well. The idea of MDS is to preserve the similarity - dissimilarity distances of a set of points in the low dimensional embedding space. The similarities between points in the original space are assumed to be represented by the Euclidean Metric in the form of a Euclidean distance. We will see, as in later approaches, as Sammon mapping [Sammon, 1969] is a special case of the MDS, with the aim of preserving a

weighted Euclidean distance. If instead of applying and considering a Euclidean distance we consider a geodesic one, we obtain the Isomap method [Tenenbaum et Al., 2000].

The goal of classical MDS is to preserve the similarity of data points in the embedding space as it was in the input space. Distance, dissimilarity and similarity (or proximity) are defined for any pair of objects in any space. They have the following properties:

- $d(x, y) \geq 0$
- $d(x, y) = 0$ iff $x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

Then, one way to measure similarity is inner product. Hence, we can minimize the difference of similarities in the input and embedding spaces [Ghojogh et Al., 2020]:

$$\underset{\{y_i\}_{i=1}^n}{minimize} \qquad c_1 := \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i^\mathsf{T} x_j - y_i^\mathsf{T} y_j) \tag{3.10}$$

or in matrix form:

$$\underset{\boldsymbol{Y}}{minimize} \qquad c_1 := \|\boldsymbol{X}^\mathsf{T}\boldsymbol{X} - \boldsymbol{Y}^\mathsf{T}\boldsymbol{Y}\|_F^2 \tag{3.11}$$

where $\|\cdot\|_F$ denote the Frobenius norm, and $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ and $\boldsymbol{Y}^\mathsf{T}\boldsymbol{Y}$ are the Gram matrices of the original data $\mathbf{X}$ and the embedded data $\mathbf{Y}$, respectively. If we decompose the Gram matrices using a SVD the objective function can be simplified as follows:

$$\begin{aligned} \boldsymbol{X}^\mathsf{T}\boldsymbol{X} &= \boldsymbol{V}\boldsymbol{\Delta}\boldsymbol{V}^\mathsf{T} \\ \boldsymbol{Y}^\mathsf{T}\boldsymbol{Y} &= \boldsymbol{Q}\boldsymbol{\Psi}\boldsymbol{Q}^\mathsf{T} \end{aligned} \tag{3.12}$$

$$\begin{aligned} \|\boldsymbol{X}^\mathsf{T}\boldsymbol{X} - \boldsymbol{Y}^\mathsf{T}\boldsymbol{Y}\|_F^2 &= \boldsymbol{tr}\left[(\boldsymbol{X}^\mathsf{T}\boldsymbol{X} - \boldsymbol{Y}^\mathsf{T}\boldsymbol{Y})^2\right] \\ &= \boldsymbol{tr}\left[(\boldsymbol{\Delta} - \boldsymbol{V}^\mathsf{T}\boldsymbol{Q}\boldsymbol{\Psi}\boldsymbol{Q}^\mathsf{T}\boldsymbol{V})^2\right] \\ &= \boldsymbol{tr}\left[(\boldsymbol{\Delta} - \boldsymbol{M}\boldsymbol{\Psi}\boldsymbol{M}^\mathsf{T})^2\right] \end{aligned} \tag{3.13}$$

where $\boldsymbol{tr}(\cdot)$ denotes the trace of matrix, $\mathbf{V}$ and $\boldsymbol{M} := \boldsymbol{V}^\mathsf{T}\boldsymbol{Q}$ where $\boldsymbol{Q}$ the right singular vectors, $\boldsymbol{\Delta}$ and $\boldsymbol{\Psi}$. (for the proof remind to the annex).
Therefore the problem of minimization becomes:

$$\begin{aligned} \underset{\boldsymbol{Y}}{minimize} \quad & \|\boldsymbol{X}^\mathsf{T}\boldsymbol{X} - \boldsymbol{Y}^\mathsf{T}\boldsymbol{Y}\|_F^2 \\ \underset{\boldsymbol{M}\boldsymbol{\Psi}}{minimize} \quad & \boldsymbol{tr}\left[(\boldsymbol{\Delta} - \boldsymbol{M}\boldsymbol{\Psi}\boldsymbol{M}^\mathsf{T})^2\right] \end{aligned} \tag{3.14}$$

As the optimization problem is unconstrained and the objective function is the trace of a quadratic function, then the minimum is non-negative. The derivative with respect to variable $\mathbf{M}$ and $\boldsymbol{\Psi}$ are:

$$\begin{aligned} &\frac{\partial c_1}{\partial \boldsymbol{M}} = 2(\boldsymbol{M}\boldsymbol{\Psi}\boldsymbol{M}^\mathsf{T})\boldsymbol{M}\boldsymbol{\Psi} - 2\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{\Psi} = 0 \\ &\boldsymbol{M}\boldsymbol{\Psi}\boldsymbol{M}^\mathsf{T} = \boldsymbol{\Delta} \end{aligned} \tag{3.15}$$

$$\begin{aligned} &\frac{\partial c_1}{\partial \boldsymbol{\Psi}} = 2M^\mathsf{T}(\boldsymbol{M}\boldsymbol{\Psi}\boldsymbol{M}^\mathsf{T})\boldsymbol{M} - 2\boldsymbol{M}^\mathsf{T}\boldsymbol{\Delta}\boldsymbol{M}^\mathsf{T} = 0 \\ &\boldsymbol{M}\boldsymbol{\Psi}\boldsymbol{M}^\mathsf{T} = \boldsymbol{\Delta} \end{aligned} \tag{3.16}$$

For the derivative with respect to $\boldsymbol{\Psi}$ some simplification has been done.
Both equations lead to the same result, that has one unique solution:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{I} \\ \boldsymbol{\Psi} &= \boldsymbol{\Delta} \end{aligned} \tag{3.17}$$

which means that the minimum value of the non-negative objective function is zero.

According to Equation 3.12 we have:

$$\begin{aligned} \boldsymbol{Y}^\intercal \boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{\Psi}\boldsymbol{Q}^\intercal &= \boldsymbol{Q}\boldsymbol{\Psi}^{\frac{1}{2}}\boldsymbol{\Psi}^{\frac{1}{2}}\boldsymbol{Q}^\intercal \\ &\implies \boldsymbol{Y} = \boldsymbol{\Psi}^{\frac{1}{2}}\boldsymbol{Q}^\intercal \\ &\implies \boldsymbol{Y} = \boldsymbol{\Delta}^{\frac{1}{2}}\boldsymbol{V}^\intercal \end{aligned} \tag{3.18}$$

where the last passage can be done since $\boldsymbol{V}^\intercal\boldsymbol{Q} = \boldsymbol{I} \implies \boldsymbol{Q} = \boldsymbol{V}$.
In summary, for embedding $\boldsymbol{X}$ using classical MDS, the eigenvalue decomposition of $\boldsymbol{X}^\intercal\boldsymbol{X}$ is obtained. Then, using Equation 3.18, $\boldsymbol{Y} \in \mathbb{R}^{n\times n}$ is obtained. Selecting the first top p row to have $\boldsymbol{Y} \in \mathbb{R}^{p\times n}$ that is the p-dimensional embedding of the n points.

3.2.1 Kernel Classical MDS

We expand the concept of Classic MDS by generalizing it introducing Kernel Method [Ghojogh et Al., 2020].
Let $d_{ij}^2 = \|x_i - x_j\|_2^2$ be the squared Euclidean distance between x_i and x_j, we have:

$$\begin{aligned} d_{ij}^2 = \|x_i - x_j\|_2^2 &= (x_i - x_j)^\intercal (x_i - x_j) \\ &= x_i^\intercal x_i - x_i^\intercal x_j - x_j^\intercal x_i + x_j^\intercal x_j \\ &= x_i^\intercal x_i - 2x_i^\intercal x_j + x_j^\intercal x_j \\ &= G_{ii} - 2G_{ij} + G_{jj} \end{aligned} \tag{3.19}$$

where $G := \boldsymbol{X}^\intercal\boldsymbol{X} \in \mathbb{R}^{n\times n}$ is the Gram matrix. If $g := [g1, \cdots, g_n] = [G_{11}, \cdots, G_{nn}] = \text{diag}(G)$, we have:

$$\begin{aligned} d_{ij}^2 &= g_i - 2G_{ij} + g_j \\ D &= g1^\intercal - 2G + g1^\intercal \end{aligned} \tag{3.20}$$

where 1 is the vector of one and D is the distance matrix with squared Euclidean distance.

Let $H := \mathbb{1} - \frac{1}{n}11^\mathsf{T} \in \mathbb{R}^{n\times n}$ the centering matrix. We double center the matrix D as follows [Oldford, 2018]:

$$\begin{aligned}
\boldsymbol{HDH} &= (\mathbb{1} - \frac{1}{n}11^\mathsf{T})\boldsymbol{D}(\mathbb{1} - \frac{1}{n}11^\mathsf{T}) \\
&= (\mathbb{1} - \frac{1}{n}11^\mathsf{T})(g1^\mathsf{T} - 2G + g1^\mathsf{T})(\mathbb{1} - \frac{1}{n}11^\mathsf{T}) \\
&= \left[\underbrace{(\mathbb{1} - \frac{1}{n}11^\mathsf{T})1}_{=0}\boldsymbol{g}^\mathsf{T} - 2(\mathbb{1} - \frac{1}{n}11^\mathsf{T})\boldsymbol{G} + (\mathbb{1} - \frac{1}{n}11^\mathsf{T})\boldsymbol{g}1^\mathsf{T}\right](\mathbb{1} - \frac{1}{n}11^\mathsf{T}) \\
&= -2(\mathbb{1} - \frac{1}{n}11^\mathsf{T})\boldsymbol{G}(\mathbb{1} - \frac{1}{n}11^\mathsf{T}) + (\mathbb{1} - \frac{1}{n}11^\mathsf{T})\boldsymbol{g}\underbrace{1^\mathsf{T}(\mathbb{1} - \frac{1}{n}11^\mathsf{T})}_{=0} \\
&= -2(\mathbb{1} - \frac{1}{n}11^\mathsf{T})\boldsymbol{G}(\mathbb{1} - \frac{1}{n}11^\mathsf{T}) = -2\boldsymbol{HGH}
\end{aligned} \tag{3.21}$$

so

$$\boldsymbol{HGH} = \boldsymbol{HX}^\mathsf{T}\boldsymbol{XH} = -\frac{1}{2}\boldsymbol{HDH} \tag{3.22}$$

Note that $1^\mathsf{T}(\mathbb{1} - \frac{1}{n}11^\mathsf{T}) = 0$ and $(\mathbb{1} - \frac{1}{n}11^\mathsf{T})1 = 0$ because removing the row mean of 1 and the column mean of 1^T results the zero vector.

Can be noticed that the classical Multidimensional scaling uses the Euclidean distance as its metric. Because of using Euclidean distance the classical MDS using Gram matrix is a linear subspace learning method.

Equation 3.22 can be rewritten as a general kernel matrix rather than the double-centered Gram matrix, in order to have:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{K} = -\frac{1}{2}\boldsymbol{HDH} \tag{3.23}$$

Note that the classical MDS is using a linear kernel $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ for its kernel. In summary, for embedding $\boldsymbol{X}$ using classical MDS, the decomposition of the kernel matrix $\boldsymbol{K}$ is obtained as follows:

$$\boldsymbol{K} = \boldsymbol{V}\boldsymbol{\Delta}\boldsymbol{V}^\mathsf{T} \tag{3.24}$$

the, recalling eq 3.18, $\boldsymbol{Y} \in \mathbb{R}^{n \times n}$ is obtained.
Replacing $\boldsymbol{X}^\intercal \boldsymbol{X}$ with kernel $\boldsymbol{K} = \boldsymbol{\Phi}(\boldsymbol{X})^\intercal \boldsymbol{\Phi}(\boldsymbol{X})$, then we have:

$$\boldsymbol{K} = \boldsymbol{Y}^\intercal \boldsymbol{Y} \tag{3.25}$$

Truncation of $\boldsymbol{Y}$ in order to obtain $\boldsymbol{Y} \in \mathbb{R}^{p \times n}$ with the first (top) p rows, gives us the p-dimensional embedding of the n points.
Classical MDS with Euclidean distance is equivalent to Principal Component Analysis (PCA). Moreover, the generalized classical MDS is equivalent to kernel PCA. [Ghojogh et Al., 2020]

3.3 Metric Multidimensional Scaling

The classical Multidimensional Scaling tries to preserve the similarities of points in the embedding space. In later approaches, [Bunte et Al., 2012] the objective has been changed to the preservation of distances rather than similarities, minimizing the difference of distances o points in the input and embedding spaces. In the Metric Multidimensional Scaling the cost function is usually named as *stress function.*
In this case the optimization problem is:

$$\underset{\{y_i\}_{i=1}^n}{minimize}\, c_2 := \left(\frac{\sum_{i=1}^n \sum_{j=1, j<i}^n w_{i,j} (d_x(x_i, x_j) - d_y(y_i, y_j))^2}{\sum_{i=1}^n \sum_{j=1, j<i}^n d_x^2(x_i, x_j)} \right)^{\frac{1}{2}} \tag{3.26}$$

or without the normalization factor:

$$\underset{\{y_i\}_{i=1}^n}{minimize}\, c_2 := \left(\sum_{i=1}^n \sum_{j=1, j<i}^n w_{i,j} (d_x(x_i, x_j) - d_y(y_i, y_j)^2 \right)^{\frac{1}{2}} \tag{3.27}$$

where $w_{i,j}$ are the weights (in this case $w_{i,j} = 1$), $d_x(\dots)$ and $d_y(\dots)$ are the distance metrics in the input and the embedded spaces (usually d_x is any valid metric distance and $d_y = \|y_i - y_j\|$).
Despite the Classical MDS, the metric MDS is *non linear* methods but, in this

case the optimization problem do not have a closed form solution. and should be solved iteratively. One of the first inspiration to solve this problem is given by Sammon [Sammon, 1969] where diagonal quasi Newton's method is used.
Considering the component-wise vector, diagonal quasi Newton's method updates the solution as [Lee and Verleysen, 2007]:

$$y_{i,k}^{(\nu+1)} := y_{i,k}^{(\nu)} - \eta \left| \frac{\partial^2 c_2}{\partial y_{i,k}^2} \right|^{-1} \frac{\partial c_2}{\partial y_{i,k}} \tag{3.28}$$

where η is the learning rate, $y_{i,k}$ is the k-th element of i-th embedded point. The absolute value is needed in order to guarantee the minimum research.
Using the gradient descent, the updating process of the solution is:

$$y_{i,k}^{(\nu+1)} := y_{i,k}^{(\nu)} - \eta \frac{\partial c_2}{\partial y_{i,k}} \tag{3.29}$$

3.3.1 Majorization Algorithm

An elegant algorithm for computing an MDS solution, more powerful than traditional techniques such as gradient descent, called Scaling by MAjorizing a COmplicated Function (SMACOF) is presented in this section. This optimization strategy is based on the concept of *stress majorization*. This section is exhaustive covered by [Leeuw and Mair,2009] and [Borg and Groenen, 2005].

Principle of Majorization

Optimization problem often required to find a minimum of a function $f(x)$. In the simplest case putting derivative $f'(x) = 0$ and solving for x is enough. In more complex case, a different approach is required. A useful method,called *Iterative Majorization (IM)* consists of trying to get increasingly better estimates of the minimum.
The main idea behind majorization is to replace iteratively the original function $f(x)$ by an auxiliary function $g(x, z)$ where z is some fixed value. In order to call $g(x, z)$ a *majorizing function* of $f(x)$ it must fulfill the following require-

ments [Borg and Groenen, 2005]:

- the auxiliary function $g(x, z)$ should be more simple to minimize than $f(x)$. For example, if $g(x, z)$ is a quadratic function in x, then the minimum of $g(x, z)$ over x can be computed in one step;
- the original function must always be smaller than or at most equal to the auxiliary function, i.e., $f(x) \leq g(x, z)$;
- the auxiliary function should touch the surface at the so called supporting point z, i.e., $f(z) = g(z, z)$.

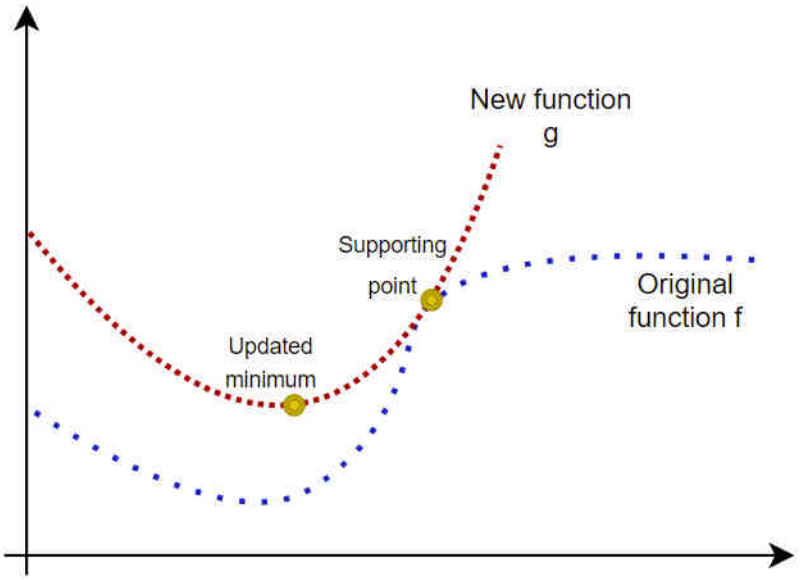


Figure 3.2: Iterative Majorization process [Groenen and Nalbantov, 2008]

Let $x^\star$ be the minimum of $g(x, z)$ over x. The last two requirements implies the chain of inequalities:

$$f(x^\star) \leq g(x^\star, z) \leq g(z, z) = f(z) \tag{3.30}$$

and its graphically represented in 3.2.

The steps of of the iterative algorithm is given by:

1. set $z = z_0$ where z_0 is the starting value

2. find the update value x^u for which $g(x^u, z) \leq g(z, z)$

3. check if $f(z) - f(x^u) \leq \varepsilon$ then stop, $\varepsilon > 0$

4. set $z = x^u$ and go to step 2

sometimes a weaker condition[1] for minimum is needed, so that $g(x^u, y) = f(y)$ and $x^u = y$.

Components of stress function

MDS inputs are usually a $n \times n$ matrix of *dissimilarities* of data. Recalling the stress function 3.27, saying $d_x(x_i, x_j) = \delta_{ij}$ the dissimilarity (ideal distances) and $d_y(y_i, y_j) = d_{ij}(\boldsymbol{X})$ the Euclidean distance (actual distance) we can rewrite it as:

$$\begin{aligned} \sigma_r(X) &= \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(\boldsymbol{X}))^2 \\ &= \sum_{i<j} w_{ij}\delta_{ij}^2 + \sum_{i<j} w_{ij} d^2(\boldsymbol{X}_{ij}) - 2\sum_{i<j} w_{ij}\delta_{ij} d_{ij}(\boldsymbol{X}) \\ &= \eta_\delta^2 + \eta^2(\boldsymbol{X}) - 2\rho(\boldsymbol{X}) \end{aligned} \tag{3.31}$$

The first part of the stress η_δ^2 depends on fixed value (fixed weights and dissimilarities). The second part $\eta^2(\boldsymbol{X})$ is a weighted sum of the squared distances, and thus a convex quadratic, and depend on $\mathbf{X}$. The last part $-2\rho(\boldsymbol{X})$, is a negative weighted sum of the Euclidean distances, consequently concave.

Assuming, without loss of generality, that $\boldsymbol{W}$ is *irreducible* (da citare de Leeuw) so the minimization problem does not need to be divide in separate problem in order to be solved.

The squared term $\eta^2(\boldsymbol{X})$ can be write in a more compact way. Let define the

[1] weaker than imposing the zero derivative

matrix $A_{ij} = (e_i - e_j)(e_i - e_j)^\mathsf{T}$ whose element are:

$$A_{ij} = \begin{cases} 1 & \text{if } a_{ii} = a_{jj} \\ -1 & \text{if } a_{ij} = a_{ji} \\ 0 & \text{elsewhere} \end{cases} \tag{3.32}$$

we define:

$$V = \sum_{i<j} w_{ij} A_{ij} \tag{3.33}$$

as the weighted sum of row and column centered matrices A_{ij} ($A_{ij}\mathbb{1} = 0$ and $\mathbb{1}^\mathsf{T} A_{ij} = 0^\mathsf{T}$). We can rewrite the second term as:

$$\eta^2(\boldsymbol{X}) = \operatorname{tr} X^\mathsf{T} V X \tag{3.34}$$

The third element, the minus weighted sum of distances, we start using the *Cauchy-Schwarz inequality* in order to have a majorization of $-d_{ij}(\mathbf{X})$. Let $\mathbf{Z}$ and $\mathbf{X}$ be a $n \times n$ input matrices, then:

$$\sum_{a=1}^{n} (x_{ia} - x_{ja})(z_{ia} - z_{ja}) \leq \left(\sum_{a=1}^{n} (x_{ia} - x_{ja})^2 \right)^{\frac{1}{2}} \left(\sum_{a=1}^{n} (z_{ia} - z_{ja})^2 \right)^{\frac{1}{2}} \tag{3.35}$$

with equality if $\mathbf{X} = \mathbf{Z}$. Dividing both sides by $d_{ij}(\boldsymbol{Z})$ and multiplying by -1 we obtain:

$$-d_{ij}(\boldsymbol{X}) \leq -\frac{\sum_{a=1}^{n} (x_{ia} - x_{ja})(z_{ia} - z_{ja})}{d_{ij}(\mathbf{Z})} \tag{3.36}$$

which is undefined if distance between point i and j is zero, but since $d_{ij}(\boldsymbol{X}) \geq 0$ then is true that $-d_{ij}(\boldsymbol{X}) \leq 0$.

Replicating the previous passages a simpler matrix expression can be obtained:

$$\sum_{a=1}^{n} (x_{ia} - x_{ja})(z_{ia} - z_{ja}) = \operatorname{tr} \boldsymbol{Z}^\mathsf{T} \boldsymbol{A_{ij}} \boldsymbol{Z} \tag{3.37}$$

Combining 3.36 and 3.37, multiplying for $w_{ij}\delta_{ij}$ and summing over indices $i < j$ gives:

$$\begin{aligned} -\rho \boldsymbol{X} &= \sum_{i<j}(w_{ij}\delta_{ij})d_{ij}(\boldsymbol{X}) \\ &\leq -\operatorname{tr} \boldsymbol{X}^{\mathsf{T}} \left(\frac{w_{ij}\delta_{ij}}{d_{ij}\boldsymbol{Z}} \boldsymbol{A}_{ij} \right) \boldsymbol{Z} \\ &= -\operatorname{tr} \boldsymbol{X}^{\mathsf{T}} \boldsymbol{B}(\boldsymbol{Z}) \boldsymbol{Z} \end{aligned} \tag{3.38}$$

where $\mathbf{B(Z)}$ is defined as follow:

$$\begin{aligned} b_{ij} &= \begin{cases} -\frac{w_{ij}\delta_{ij}}{d_{ij}(\boldsymbol{Z})} & \text{for } i \neq j and\ d_{ij}(\boldsymbol{Z}) \neq 0 \\ 0 & \text{for } i \neq j and\ d_{ij}(\boldsymbol{Z}) = 0 \end{cases} \\ b_{ii} &= -\sum_{j=1, j\neq i}^{n} b_{ij} \end{aligned} \tag{3.39}$$

Since the equality holds true when $\mathbf{Z} = \mathbf{X}$, we have obtained a majorization inequality

$$-\rho(\boldsymbol{X}) = -\operatorname{tr} \boldsymbol{X}^{\mathsf{T}} \boldsymbol{B}(\boldsymbol{X}) \boldsymbol{X} \leq -\operatorname{tr} \boldsymbol{X}^{\mathsf{T}} \boldsymbol{B}(\boldsymbol{Z}) \boldsymbol{Z} \tag{3.40}$$

The SMACOF Algorithm for Majorizing Stress

The majorization inequality for the stress function is given by:

$$\begin{aligned} \sigma_r(\boldsymbol{X}) &= \eta_\delta^2 + \operatorname{tr} \mathbf{X}^{\mathsf{T}} \mathbf{V} \mathbf{X} - 2 \operatorname{tr} \boldsymbol{X}^{\mathsf{T}} \mathbf{B}(\mathbf{X}) \mathbf{X} \\ &\leq \eta_\delta^2 + \operatorname{tr} \mathbf{X}^{\mathsf{T}} \mathbf{V} \mathbf{X} - 2 \operatorname{tr} \boldsymbol{X}^{\mathsf{T}} \mathbf{B}(\mathbf{Z}) \mathbf{Z} = \tau(\boldsymbol{X}, \boldsymbol{Z}) \end{aligned} \tag{3.41}$$

where $\tau(\boldsymbol{X}, \boldsymbol{Z})$ is the majoring function, quadratic in $\mathbf{X}$ of the stress. The minimum can be obtained setting the derivative equal 0, i.e.

$$\nabla \tau(\boldsymbol{X}, \boldsymbol{Z}) = 2\boldsymbol{X}\boldsymbol{V} - 2\boldsymbol{B}(\boldsymbol{Z})\boldsymbol{Z} = 0 \tag{3.42}$$

In order to solve the $\boldsymbol{VX} = \boldsymbol{B(Z)Z}$, the calculation of $\boldsymbol{V^{-1}}$ is needed, but, since $\mathbf{V}$ is not full rank the inverse should be calculated using the *Moore-Penrose*

inverse i.e.

$$V^{+} = (\boldsymbol{V} + \mathbb{1}\mathbb{1}^{\mathsf{T}})^{-1} - n^{-2}\mathbb{1}\mathbb{1}^{\mathsf{T}} \tag{3.43}$$

that leads to the updating formula of the SMACOF algorithm:

$$\boldsymbol{X}^{u} = \boldsymbol{V}^{+}\boldsymbol{B}(\boldsymbol{Z})\boldsymbol{Z} \tag{3.44}$$

or, in a simplified way, if $w_{ij} = 1$

$$\boldsymbol{X}^{u} = n^{-1}\boldsymbol{B}(\boldsymbol{Z})\boldsymbol{B} \tag{3.45}$$

called *Guttam Transform* ([De Leeuw and Heiser, 1980]).

Chapter 4

Non Linear Dimensionality Reduction Techniques

4.1 Isomap

Isomap is a special case of the generalized classical MDS, where the Euclidean distance has been replaced with an approximation of the geodesic distance. Following the method proposed by [Tenenbaum et Al., 2000], Isomap is based on the following assumption:

1. *Isometry* The mapping ψ preserves the geodesic distances, namely

$$\mathcal{D}(m, m') = |\theta - \theta'| \quad \forall m \leftarrow \theta, m' \leftarrow \theta' \tag{4.1}$$

2. *Convexity* The parameter space Θ is a convex subset of $\mathbb{R}^d$, that means that if θ,θ' are point in Θ, then, the entire line segment $\{(1-t)\theta+t\theta' : t \in (0,1)\}$ lies in Θ.

Geodesic Distance

The *Geodesic Distance* is the length of the shortest path between two points on the manifold. Mathematically it is defined as follows:
Let γ be a differentiable map from an interval $[a, b]$ of the real line into the surface

$\mathcal{M}$, and let $\dot{\gamma} := \frac{d}{dt}\gamma(t)$; we say that γ is *arc-length parameterized* if $\|\dot{\gamma}(t)\| = 1$ for al $t \in [a, b]$. The length associated at any curve γ is:

$$L(\gamma) = \int_a^b \|\dot{\gamma}(t)\|dt = \int_a^b \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))}dt \tag{4.2}$$

The *Geodesic distance* $\phi(p, q)$, between two points $p, q \in \mathcal{M}$ is the infimum of length over all curve $\gamma(a) = p$ and $\gamma(a) = q$. To summarize a geodesic is a curve that cannot be made shorter by adjusting any small piece of it [Wang, 2012].
Although the geodesic distance is the ideal metric to use, computationally speaking it cost a lot. In order to overcome to this limit, Isomap approximate the geodesic distance by piecewise Euclidean distances. Then, in order to find out the shortest path, two algorithm can be used: Dijkstra algorithm or Floyd-Warshal algorithm [Cormen et Al., 2009].
Following [Bengio et Al., 2006], the approximated geodesic distance can be defined as:

$$\boldsymbol{D}_{ij}^{(g)} := \min_r \sum_{i=2}^{l} \|r_i - r_{i+1}\|_2 \tag{4.3}$$

where $l \geq 2$ is the length of the sequence $r_i \in \{x_i\}_{i=1}^n$ and $\boldsymbol{D}^{(g)} \in \mathbb{R}^{n \times n}$ is the geodesic distance matrix.

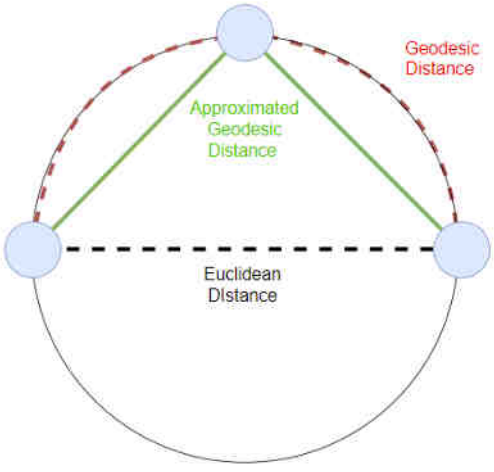


Figure 4.1: Approximate Geodesic distance - Schematic representation

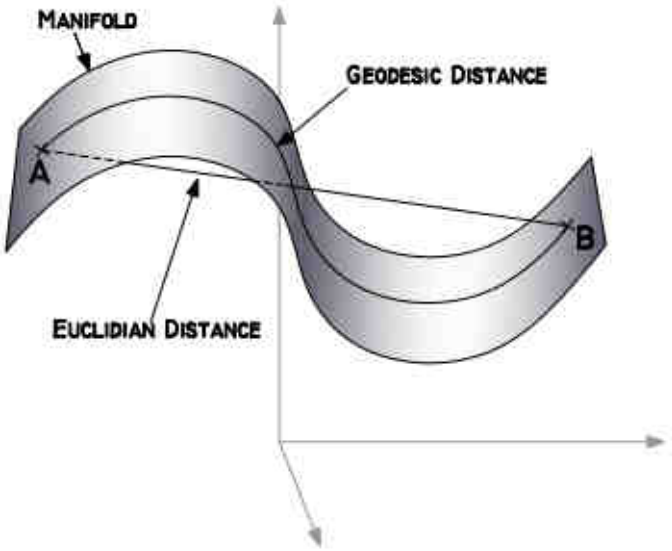


Figure 4.2: Visual difference between Euclidean and Geodesic distance [Karam and Campbell, 2013]

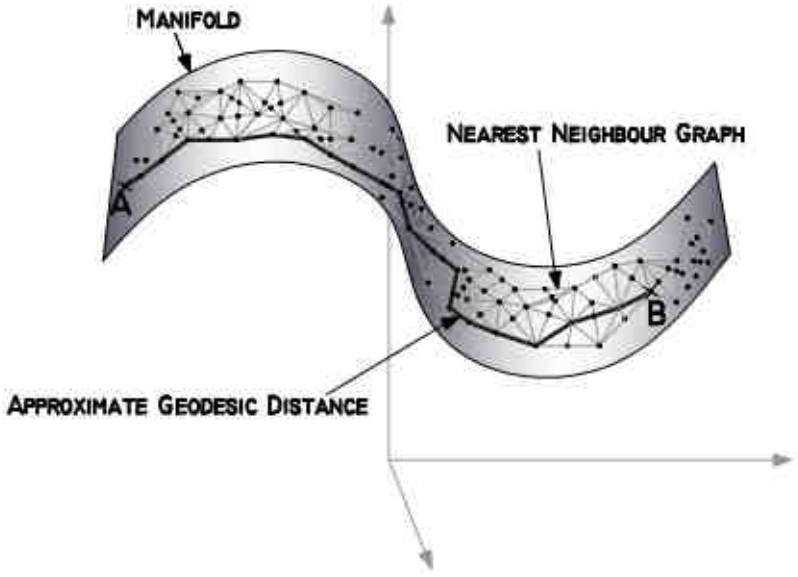


Figure 4.3: Approximate Geodesic distance calculated as the shortest path along a graph [Karam and Campbell, 2013]

Isomap Formulation

As previuosly announced, Isomap is a particular case of MDS, where the geodesic distance is used. Recall the Equation 3.23 it becomes:

$$\boldsymbol{K} = -\frac{1}{2}\mathbf{H}\ \mathbf{D}^{(g)}\mathbf{H} \tag{4.4}$$

it is a **non linear** method since the distance used in the kernel is non linear.
For the embedding we have:

$$\boldsymbol{K} = \boldsymbol{V}\boldsymbol{\Delta}\boldsymbol{V}^{\mathsf{T}} \tag{4.5}$$

$$\boldsymbol{Y} = \boldsymbol{\Delta}^{\frac{1}{2}}\boldsymbol{V}^{\mathsf{T}} \tag{4.6}$$

The result is that we have embedded a set of data $\boldsymbol{X} = [x_1, \cdots, x_n] \in \mathbb{R}^{d\times n}$ in $\boldsymbol{Y} = [y_1, \cdots, y_n] \in \mathbb{R}^{p\times n}$. **Kernel Isomap**
Reproducing the approach of the MDS is possible to find out a generalize version of Isomap: Kernel Isomap. Let $\mathbf{K(D)}$ to be as 4.4, thn we have:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{K}(\boldsymbol{D^2}) = -\frac{1}{2}\mathbf{H}\ \mathbf{D}^2\mathbf{H} \tag{4.7}$$

where $\mathbf{D}$ is the geodesic distance matrix 4.3.
Can be rewrite as [Cox and Cox, 2008]:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{K'} := \boldsymbol{K}(\boldsymbol{D}^2) + 2c\boldsymbol{K}(\boldsymbol{D}) + \frac{1}{2}c^2\boldsymbol{H} \tag{4.8}$$

$\boldsymbol{K}'$ is positive semi-definie if $c \geq c^\star$, where $c^\star$ is the largest eigenvalue of the following matrix [Cayton, 2005]:

$$\begin{bmatrix} 0 & 2K(D^2) \\ -I & -4K(D) \end{bmatrix} \in \mathbb{R}^{2n\times 2n} \tag{4.9}$$

Then, in order to obtain embedded $\mathbf{Y}$ data Equation. 3.18 should be used.

Landmark MDS and Isomap

Landmark MDS and, consequently, landmark Isomap are techniques that relies on *Nystrom approximation* [Wu and Chan, 2004] to allow the scalability to large dataset of this techniques.

Nystrom approximation is used to approximate a positive semi-definite matrix

using a subset of its column (or rows).

Consider $\mathbb{R}^{n\times n} \ni \boldsymbol{K} \succeq 0$, positive semi-definite define as follow:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{K} = \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{B}^\top & \boldsymbol{C} \end{array}\right] \tag{4.10}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times m}$, $\boldsymbol{B} \in \mathbb{R}^{m\times(n-m)}$, and $\boldsymbol{C} \in \mathbb{R}^{(n-m)\times(n-m)}$ in which $m \ll n$.

The key idea behind *Nystrom approximation* is to approximate the matrix **C** and then **K**, knowing **A** and **B**, since **K** is positive semi-definite, once we know the similarity of points from each other in **A** and **B**($n - m$ respect to m), then we don't have much possibilities to set the similarities of $n-m$, so the matrix **C** since **K** is positive semi-definite. The set of points, selected randomly from matrix **A** rows or columns are called *landmarks*.

Since **K** is positive semi-definite,it admits a form such as $\boldsymbol{K} = \boldsymbol{O}^\top\boldsymbol{O}$, where $\boldsymbol{O} =$ can be rewritten as follow:

$$\boldsymbol{K} = \boldsymbol{O}^\top\boldsymbol{O} = \begin{bmatrix} \boldsymbol{R}^\top \\ \boldsymbol{S}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{S} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^\top\boldsymbol{R} & \boldsymbol{R}^\top\boldsymbol{S} \\ \boldsymbol{S}^\top\boldsymbol{R} & \boldsymbol{S}^\top\boldsymbol{S} \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{B}^\top & \boldsymbol{C} \end{array}\right] \tag{4.11}$$

Hence, $\boldsymbol{A} = \boldsymbol{R}^\top\boldsymbol{R}$ and its eigendecomposition is given by [Ghojogh and Karray, 2019]:

$$\begin{aligned} &\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top \\ &\implies \boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top \implies \boldsymbol{R} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{U}^\top \end{aligned} \tag{4.12}$$

and, since $\boldsymbol{B} = \boldsymbol{R}^\top\boldsymbol{S}$ and $\boldsymbol{U}$ orthogonal, we have:

$$\begin{aligned} &\boldsymbol{B} = \left(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{U}^\top\right)^\top \boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{S} \\ &\implies \boldsymbol{S} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{U}^\top\boldsymbol{B} \end{aligned} \tag{4.13}$$

In conclusion, $\boldsymbol{C}$ can be obtained as:

$$\begin{aligned}\boldsymbol{C} =& \boldsymbol{S}^\intercal \boldsymbol{S} = \boldsymbol{B}^\intercal \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{U}^\intercal \boldsymbol{B} \\ =& \boldsymbol{B}^\intercal \boldsymbol{U} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma}^{-1} \boldsymbol{U}^\intercal \boldsymbol{B} = \boldsymbol{B}^\intercal \boldsymbol{A}^{-1} \boldsymbol{B}\end{aligned} \tag{4.14}$$

and equation 4.10 becomes:

$$\boldsymbol{K} \approx \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{B}^\intercal & \boldsymbol{B}^\intercal \boldsymbol{A}^{-1} \boldsymbol{B} \end{array}\right] \tag{4.15}$$

The approximation of $\boldsymbol{K}$ becomes more accurate by increasing m, and, if it is at most m, the approximation is almost exact. The usual case implies $m \ll n$, then the Nystrom approximation works better with low-rank matrices [Kumar ad Schneider, 2017].

The kernel approximation in the presence of big data can be obtained using 4.12, decomposing it in a $m \times m$ kernel submatrix. Then, combining equation 4.11 and 3.25 the approximation of embedded data can be obtained:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{Y} = [\boldsymbol{R}, \boldsymbol{S}] = \left[\boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{U}^\intercal, \boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{U}^\intercal \boldsymbol{B}\right] \tag{4.16}$$

As for the previous cases, truncating the $\boldsymbol{Y}$ matrix with p top rows, the p dimensional embedding of n points ($\boldsymbol{Y} \in \mathbb{R}^{p \times n}$) is obtained.

The *Distance Matrix* $\boldsymbol{D}$ in Landmark MDS and Isomap, can be partitioned (following the previous approach) as follow:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{D} = \left[\begin{array}{c|c} \boldsymbol{E} & \boldsymbol{F} \\ \hline \boldsymbol{F}^\intercal & \boldsymbol{G} \end{array}\right] \tag{4.17}$$

where $\boldsymbol{E} \in \mathbb{R}^{m \times m}$, $\boldsymbol{F} \in \mathbb{R}^{m \times (n-m)}$, and $\boldsymbol{G} \in \mathbb{R}^{(n-m) \times (n-m)}$ in which $m \ll n$. Since the kernel matrix is related to distance matrix by the following relationship:

$$\boldsymbol{K} = -\frac{1}{2}\left(\boldsymbol{D}_{ij}^2 - \mathbb{1}_j \boldsymbol{\Sigma}_i \boldsymbol{c}_i \boldsymbol{D}_{ij}^2 - \mathbb{1}_i \boldsymbol{\Sigma}_j \boldsymbol{c}_i \boldsymbol{D}_{ij}^2 + \boldsymbol{\Sigma}_{i,j} \boldsymbol{c}_i \boldsymbol{c}_j \boldsymbol{D}_{ij}^2\right) \tag{4.18}$$

where $\boldsymbol{\Sigma}_i \boldsymbol{c}_i = 1$. Indeed, the partition of the kernel matrix can be obtained from the partitions of the distance matrix as [Platt, 2015]:

$$\boldsymbol{A}_{ij} = -\frac{1}{2}(\boldsymbol{E}_{ij}^2 - \mathbb{1}_i \frac{1}{m}\sum\nolimits_p \boldsymbol{E}_{pj}^2 - \mathbb{1}_j \frac{1}{m}\sum\nolimits_q \boldsymbol{E}_{qj}^2 + \frac{1}{m^2}\sum\nolimits_{p,q} \boldsymbol{E}_{pq}^2) \tag{4.19}$$

$$\boldsymbol{B}_{ij} = -\frac{1}{2}(\boldsymbol{F}_{ij}^2 - \mathbb{1}_i \frac{1}{m}\sum\nolimits_p \boldsymbol{F}_{pj}^2 - \mathbb{1}_j \frac{1}{m}\sum\nolimits_q \boldsymbol{E}_{qj}^2) \tag{4.20}$$

and $\boldsymbol{C}$ is easily obtained using 4.14. Nystrom approximation result as a general method for speed up and reduce the execution time in some manifold learning techniques, but on the other hand it reduces the overall accuracy of the algorithm The Isomap main steps are presented in Algorithm 6:

4.1.1 Sammon Mapping

A special case of metric MDS, introduced mainly for historical reasons, can be found in Sammon mapping (originally known as Nonlinear Mapping [Sammon, 1969], but it is also known as Sammon mapping or Sammon's nonlinear mapping). It is a non linear method and the optimization problem is based on the weighted version of equation 3.26

$$\underset{\{y_i\}_{i=1}^n}{minimize} := \frac{1}{a}\sum_{i=1}^{n}\sum_{j=1,j<i}^{n} w_{ij}(d_x(x_i,x_j) - d_y(y_i,y_j))^2 \tag{4.21}$$

where w_{ij} is the weight, a the normalization factor, and d_x[1] and d_y are usually the Euclidean distances. The idea behind Sammon mapping is to penalize more big distance and to give more credit to the smaller one (neighbors point) in order to preserve the local structure of the manifold (to fit the manifold locally).
It can be achieved by set weight and normalization factor as follow:

$$w_{ij} = \frac{1}{d_x(x_i,x_j)} \tag{4.22}$$

[1] d_x can be any metric, but a common choice is the Euclidean distance

$$a = \sum_{i=1}^{n} \sum_{j=1, j<i}^{n} d_x(x_i, x_j) \tag{4.23}$$

Now, substituting equations 4.22 and 4.23 in equation 4.21 we obtain:

$$\underset{\boldsymbol{Y}}{minimize}\, c_4 := \frac{1}{\sum_{i=1}^{n} \sum_{j=1, j<i}^{n} d_x(x_i, x_j)} \times \sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \frac{(d_x(x_i, x_j) - d_y(y_i, y_j))^2}{d_x(x_i, x_j)} \tag{4.24}$$

This optimization problem has been solved by Sammon [Sammon, 1969] using the diagonal quasi-Newton's method, i.e. equation 3.28 [Lee and Verleysen, 2007]. This technique involves gradient and the second derivative of the cost function. The gradient is:

$$\frac{\partial c_4}{\partial y_{i,k}} = -\frac{2}{a} \sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(y_i, y_j)} (y_{i,k} - y_{j,k}) \tag{4.25}$$

Instead, the second derivative is given by:

$$\frac{\partial^2 c_4}{\partial y_{i,k}^2} = -\frac{2}{a} \sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \left(\frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(x_i, x_j)} - \frac{(y_{i,k} - y_{j,k})^2}{d_y^3(y_i, y_j)} \right) \tag{4.26}$$

In order to optimize the time complexity of the Sammon Mapping, the k Nearest Neighbors (kNN) can be used

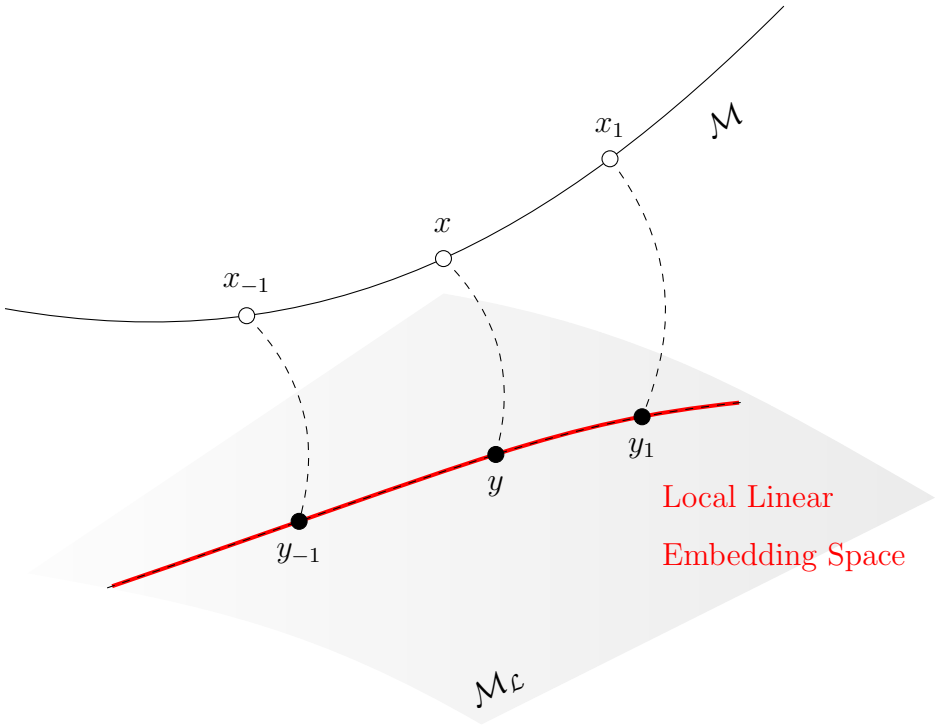
$$\underset{\{y_i\}_{i=1}^{n}}{minimize} := \frac{1}{a} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i}^{n} w_{ij} (d_x(x_i, x_j) - d_y(y_i, y_j))^2 \tag{4.27}$$

where $\mathcal{N}_i$ denotes the i-th point of kNN.

To conclude, Sammon mapping can be particularly expensive from a computation point of view since convergence is not always guaranteed and the number of iterations must be determined experimentally.

4.2 Local Linear Embedding (LLE)

Locally Linear Embedding (LLE) is another *Non Linear* techniques of dimensional reduction. In contrast to Multidimensional Scaling (MDS) and Isomap, where the aim is to preserve a global structure, LLE, as suggested by the name, aims to preserve and fitting the local structure of manifold in the embedding space [Ghojogh et Al., 2018]. Introduced by [Roweis and Saul, 2000], LLE relies on the same concept of preservation of distances of points in the high dimensional input space respect to the low dimensional embedded space. The process evolves by unfolding the non linear manifold by a sequence of locally sub unfolds, a similar idea of a piece-wise spline regression [Marsh and Cormier, 2001] (see Figure 4.4 for visual interpretation).



$\mathcal{M}$ Non Linear Manifold

$\mathcal{M}_{\mathcal{L}}$ Embedded space

• Projection in a lower dimensional embedding space

Figure 4.4: Piece-wise local manifold unfolding

The core concept of this technique can be summarize, and subsequently explained

in details, in the following three steps (see Figure 4.5):

1. Find k-nearest neighbors
2. linear reconstruction by the neighbors
3. linear embedding using the previously calculated weights.

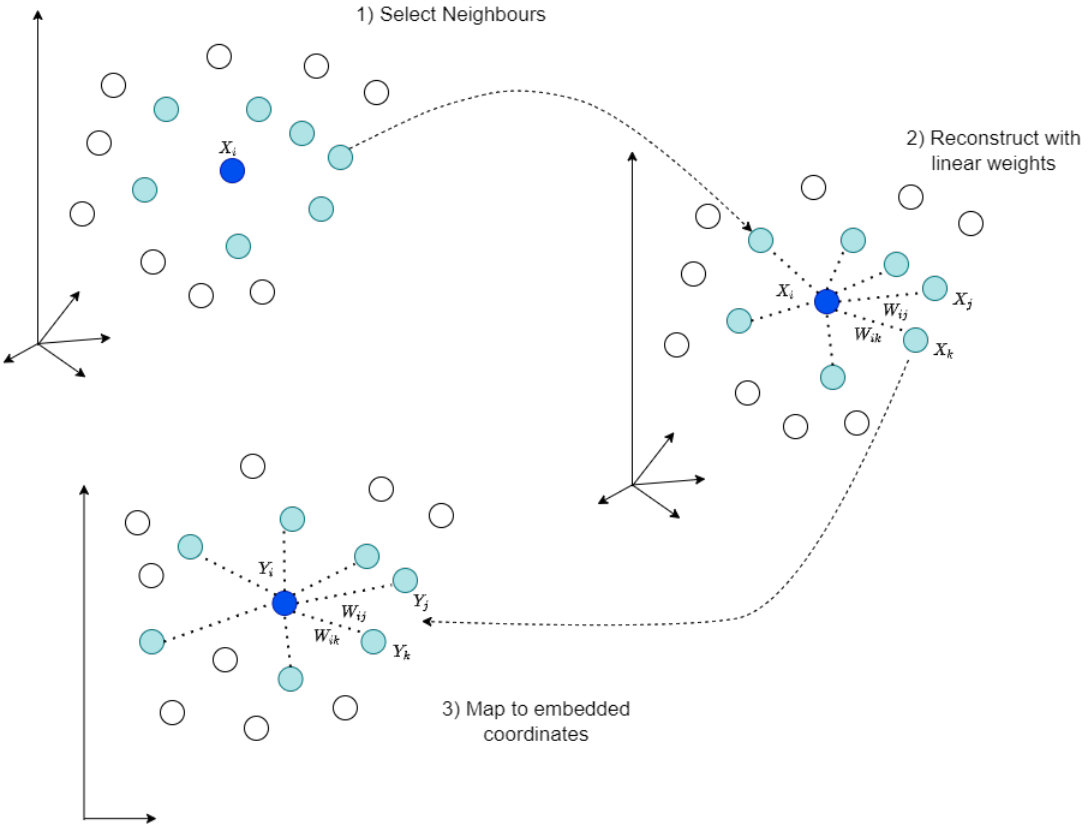


Figure 4.5: Main steps of LLE: (1) find K nearest neighbors (cyan dots) of the point (blue), (2) compute the weights (W_i) of the linear reconstruction by the neighbors, (3) linear embedding in a low dimensional manifold using the W_i weights.

4.2.1 Linear Reconstruction using k-nearest neighbors

We use kNN for the linear reconstruction. The notation adopted is the following:

1. $x_{ij} \in \mathbb{R}^d$ denote the j-neighbor of x_i
2. $\mathbb{R}^{d \times k} \ni \boldsymbol{X_i} = [x_{i1}, \cdots, x_{ik}]$ is the matrix formed by the k neighbors of x_i

Then, the second step is to find out the weights of every points, for the linear reconstruction using its kNN. The related optimization problem, associated to cost function $\varepsilon(\tilde{\boldsymbol{W}})$, is formulated as follow:

$$\begin{aligned} \underset{\tilde{W}}{minimize} \quad & \varepsilon(\tilde{\boldsymbol{W}}) := \sum_{i=1}^{n} \|x_i - \sum_{j=1}^{k} \tilde{w}_{ij} \boldsymbol{x_{ij}}\|_2^2 \\ subject\ to \quad & \sum_{j=1}^{k} \tilde{w}_{ij} = 1, \quad \forall i \in \{1, \cdots, n\} \end{aligned} \tag{4.28}$$

where $\mathbb{R}^{n\times k} \ni \tilde{\boldsymbol{W}} := [\tilde{\boldsymbol{w}}_1, \cdots, \tilde{\boldsymbol{w}}_1]^\mathsf{T}$ include the weights ($\mathbb{R}^k \ni \tilde{\boldsymbol{w}} := [\tilde{w}_{i1}, \cdots, \tilde{w}_{ik}]^\mathsf{T}$) of linear reconstruction of i-th point using k-neighbors,then, $\boldsymbol{x_{ij}} \in \mathbb{R}^d$ is the j-th neighbors of the i-th point.

The weight matrix $\tilde{\boldsymbol{W}}$ shows the local geometry of the embedded manifold, and it is invariant to rotations, rescaling and translation of data point and its neighbors point. The required constrain,$\sum_{j=1}^{k} \tilde{w}_{ij} = 1$ assures the invariant to global translation. For instance:

$$\begin{aligned} \|(x_i + c) - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij}(x_j + c)\| &= \|x_i + c - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} x_j - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} c\| \\ &= \|x_i + c - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} x_j - c\| \\ &= \|x_i - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} x_j\| \end{aligned} \tag{4.29}$$

If T is the rotation operator we have:

$$\begin{aligned} \|T(x_i) - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij}(Tx_j)\| &= \|T\left(x_i - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} x_j\right)\| \\ &= \|x_i - \sum_{j\in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} x_j\| \end{aligned} \tag{4.30}$$

where has the linearity and orthonormality of T been used.
Similarly, for the dilatation operator $D_\lambda x = \lambda x$ we have:

$$\begin{aligned} \|\lambda(x_i) - \sum_{j \in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij}(\lambda x_j)\| &= \|\lambda \left(x_i - \sum_{j \in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} x_j \right) \| \\ &= |\lambda| \|x_i - \sum_{j \in \boldsymbol{X}_i} \tilde{\boldsymbol{W}}_{ij} x_j\| \end{aligned} \tag{4.31}$$

In order to solve the minimization problem the equation 4.28 can be rewritten as follow:

$$\begin{aligned} \varepsilon(\tilde{\boldsymbol{W}}) : &= \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{X}_i \tilde{\boldsymbol{w}}_i\|_2^2 \\ &= \sum_{i=1}^{n} \|\boldsymbol{x}_i \mathbb{1}^\mathsf{T} \tilde{\boldsymbol{w}}_i - \boldsymbol{X}_i \tilde{\boldsymbol{w}}_i\|_2^2 \\ &= \sum_{i=1}^{n} \|(\boldsymbol{x}_i \mathbb{1}^\mathsf{T} - \boldsymbol{X}_i) \tilde{\boldsymbol{w}}_i\|_2^2 \\ &= \sum_{i=1}^{n}) \tilde{\boldsymbol{w}}_i^\mathsf{T} (\boldsymbol{x}_i \mathbb{1}^\mathsf{T} - \boldsymbol{X}_i)^\mathsf{T} (\boldsymbol{x}_i \mathbb{1}^\mathsf{T} - \boldsymbol{X}_i) \tilde{\boldsymbol{w}}_i \\ &= \tilde{\boldsymbol{w}}_i^\mathsf{T} \boldsymbol{G}_i \tilde{\boldsymbol{w}}_i \end{aligned} \tag{4.32}$$

where $\boldsymbol{G}_i$ is the Gram matrix defined as:

$$\mathbb{R}^{k \times k} \ni \boldsymbol{G}_i := (\boldsymbol{x}_i \mathbb{1}^\mathsf{T} - \boldsymbol{X}_i)^\mathsf{T} (\boldsymbol{x}_i \mathbb{1}^\mathsf{T} - \boldsymbol{X}_i) \tag{4.33}$$

Equation 4.28 becomes:

$$\begin{aligned} &\underset{\{\tilde{\boldsymbol{w}}\}_{i=1}^n}{minimize} \quad \sum_{i=1}^{n} \tilde{\boldsymbol{w}}_i^\mathsf{T} \boldsymbol{G}_i \tilde{\boldsymbol{w}}_i \\ &subject\ to \quad \mathbb{1}^\mathsf{T} \tilde{\boldsymbol{w}}_i = 1 \quad \forall \in \{1, \cdots, n\} \end{aligned} \tag{4.34}$$

To solve the optimization problem expressed by equation 4.34 the Lagrangian Multiplier method, as proposed in [Boyd et Al., 2004], is used.

The Lagrangian of Equation 4.34 is:

$$\mathcal{L} = \sum_{i=1}^{n} \tilde{\boldsymbol{w}}_i^{\mathsf{T}} \boldsymbol{G}_i \tilde{\boldsymbol{w}}_i - \sum_{i=1}^{n} \lambda_i (\mathbf{1}^{\mathsf{T}} \tilde{\boldsymbol{w}} - 1) \tag{4.35}$$

Now, setting the derivatives to zero:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{w}}_i} &= 2\boldsymbol{G}_i \tilde{\boldsymbol{w}}_i - \lambda_i \mathbf{1} \overset{set}{=} 0 \\ \implies \tilde{\boldsymbol{w}}_i &= \frac{1}{2} \boldsymbol{G}_i^{-1} \lambda_i \mathbf{1} = \frac{\lambda_i}{2} \boldsymbol{G}_i^{-1} \mathbf{1} \end{aligned} \tag{4.36}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \lambda} &= \mathbf{1}^\top \tilde{\boldsymbol{w}}_i \overset{set}{=} 0 \\ \implies \mathbf{1}^\top \tilde{\boldsymbol{w}}_i &= 1 \end{aligned} \tag{4.37}$$

Equations 4.36 and 4.37 allow to obtain the multipliers:

$$\frac{\lambda_i}{2} \mathbf{1}^\top \boldsymbol{G}_i^{-1} \mathbf{1} = 1 \longrightarrow \lambda_i = \frac{2}{\mathbf{1}^\top \boldsymbol{G}_i^{-1} \mathbf{1}} \tag{4.38}$$

Therefore, the weights of the linear reconstruction of high dimensional space are:

$$\tilde{\boldsymbol{w}}_i = \frac{\lambda_i}{2} \boldsymbol{G}_i^{-1} \mathbf{1} = \frac{\boldsymbol{G}_i^{-1}}{\mathbf{1}^\top \boldsymbol{G}_i^{-1} \mathbf{1}} \tag{4.39}$$

Since the rank of the Gramiam matrix $\boldsymbol{G}$ is given by $min(k, d)$, a problem can arise when the d dimensional data in input are lower then k numbers of neighbors. In this case, $\boldsymbol{G}_i$ should be replaced by $\boldsymbol{G}_i + \varepsilon \boldsymbol{I}$ where ε is a small positive number. This trick allow the computation of the $\boldsymbol{G}_i^{-1}$,but generally speaking, $k \ll d$ implying that $\boldsymbol{G}_i$ is full rank (k) and the inverse can be computed without any problem.

The third step, consists to embed data in the low dimensional space preserving the same weights as in the input space, originating the following linear embedding

optimization problem:

$$\begin{aligned} \underset{\boldsymbol{Y}}{minimize} \quad & \sum_{i=1}^{n} \|\boldsymbol{y}_i - \sum_{j=1}^{n} w_{ij}\boldsymbol{y}_j\|_2^2 \\ subject\ to \quad & \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i\boldsymbol{y}_i^\top = \boldsymbol{I} \\ & \sum_{i=1}^{n} \boldsymbol{y_i} = 0 \end{aligned} \tag{4.40}$$

where $\boldsymbol{I}$ is the identity matrix, the rows of $\boldsymbol{Y} \in \mathbb{R}^{n\times p}$ are the $[y_1, \cdots, y_n]^\top$ embedded data points and w_i are the weights obtained from the linear reconstruction as follows:

$$w_{ij} = \begin{cases} \tilde{w}_{ij} & \text{if } \boldsymbol{x}_j \in kNN(\boldsymbol{x_i}) \\ 0 & \text{otherwise} \end{cases} \tag{4.41}$$

The constrains in 4.40 assure the zero mean and the unit covariance of the embedded points.

Now, let $\mathbb{R}^n \ni \boldsymbol{w}_i := [w_{i1}, \cdots, w_{in}]$ be n dimensional embedded weights vector and $\mathbb{R}^n \ni \mathbf{1}_i := [0, \cdots, 1, \cdots, 0]$ be n dimensional vector whose i-th element is one and 0 the others.

The cost function expressed in 4.40 can be rewritten as:

$$\sum_{i=1}^{n} \|\boldsymbol{y}_i - \sum_{j=1}^{n} w_{ij}\boldsymbol{y}_j\|_2^2 = \sum_{i=1}^{n} \|\boldsymbol{Y}^\top\mathbf{1}_i - \boldsymbol{Y}^\top\boldsymbol{w}_i\|_2^2 \tag{4.42}$$

and the corresponding matrix form is:

$$\begin{aligned} \sum_{i=1}^{n} \|\boldsymbol{Y}^\top\mathbf{1}_i - \boldsymbol{Y}^\top\boldsymbol{w}_i\|_2^2 &= \|\boldsymbol{Y}^\top\boldsymbol{I} - \boldsymbol{Y}^\top\boldsymbol{W}\|_F^2 \\ &= \|\boldsymbol{Y}^\top(\boldsymbol{I} - \boldsymbol{W})^\top\|_F^2 \end{aligned} \tag{4.43}$$

where the i-th row of $\mathbb{R}^{n\times n} \ni \boldsymbol{W} := [w_1, \cdots, w_n]^\top$ are the weights of the i-th point and $\|\cdot\|_F$ is the Frobenius norm.

The 4.43 can be simplified as follows:

$$\begin{aligned} \|\boldsymbol{Y}^\top(\boldsymbol{I}-\boldsymbol{W})^\top\|_F^2 &= \operatorname{tr}\left((\boldsymbol{I}-\boldsymbol{W})\boldsymbol{Y}\boldsymbol{Y}^\top(\boldsymbol{I}-\boldsymbol{W})^\top\right) \\ &= \operatorname{tr}\left(\boldsymbol{Y}^\top(\boldsymbol{I}-\boldsymbol{W})^\top(\boldsymbol{I}-\boldsymbol{W})\boldsymbol{Y}\right) \\ &= \operatorname{tr}(\boldsymbol{Y}^\top\boldsymbol{M}\boldsymbol{Y}) \end{aligned} \tag{4.44}$$

where

$$\mathbb{R}^{n\times n} \ni \boldsymbol{M} := (\boldsymbol{I}-\boldsymbol{W})^\top(\boldsymbol{I}-\boldsymbol{W}) \tag{4.45}$$

Can be notice that $(\boldsymbol{I}-\boldsymbol{W})$ is the Laplacian of the matrix $\boldsymbol{W}^2$ (see paragraph ???), hence $\boldsymbol{M}$ is the Gram matrix over the Laplacian of weight matrix.
Finally, the optimization problem takes the following form:

$$\begin{aligned} \underset{\boldsymbol{Y}}{minimize} \quad & \operatorname{tr}(\boldsymbol{Y}^\top\boldsymbol{M}\boldsymbol{Y}) \\ subject\ to \quad & \frac{1}{n}\boldsymbol{Y}^\top\boldsymbol{Y} = \boldsymbol{I} \\ & \boldsymbol{Y}^\top\mathbf{1} = \mathbf{0} \end{aligned} \tag{4.46}$$

where $\mathbf{1} \in \mathbb{R}^n$ and $\mathbf{0} \in \mathbb{R}^p$.
Now, if we ignore the second constrain, that will turn out to be implicitly satisfied, the Lagrangian of eq. 4.46 is [Boyd et Al., 2004]:

$$\mathcal{L} = \operatorname{tr}(\boldsymbol{Y}^\top\boldsymbol{M}\boldsymbol{Y}) - \operatorname{tr}(\boldsymbol{\Lambda}^\top(\frac{1}{n}\boldsymbol{Y}^\top\boldsymbol{Y} - \boldsymbol{I})) \tag{4.47}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ is the diagonal matrix of the Lagrange multipliers. Again, setting the derivative of $\mathcal{L} = 0$, we have:

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\boldsymbol{Y}} &= 2\boldsymbol{M}\boldsymbol{Y} - \frac{2}{n}\boldsymbol{Y}\boldsymbol{\Lambda} \overset{set}{=} 0 \\ \longrightarrow \boldsymbol{M}\boldsymbol{Y} &= \boldsymbol{Y}(\frac{1}{n}\boldsymbol{\Lambda}) \end{aligned} \tag{4.48}$$

where the columns of $\boldsymbol{Y}$ are the eigenvectors of $\boldsymbol{M}$ whose eigenvalue are the diagonal elements of $\frac{1}{n}\boldsymbol{\Lambda}$, therefore, this is the eigenvalue problem for $\boldsymbol{M}$. Thus,

[2]The column of $\boldsymbol{W}$ add to one, hence the sum over the column of $(\boldsymbol{I}-\boldsymbol{W})$ add to zero.

the columns of $\boldsymbol{Y}$ should be sorted from the smallest to largest eigenvalues, but since if a graph has k disjoint connected path, its Laplacian matrix ($(I - W)$) has k zero eigenvalues, namely one zero eigenvalue whose eigenvector is $\mathbf{1} = [1, 1, \cdots, 1]^\top$ ([Marsden, 2013]).

Now, after sorting, the first eigenvector (having zero eigenvalue) can be ignore and the p smallest eigenvector can be selected in order to obtain $\boldsymbol{Y} \in \mathbb{R}^{n \times p}$.

4.3 Locally Tangent Space Alignment (LTSA)

In this section another local non linear dimensional reduction technique is presented, namely Local Tangent Space Alignment (LTSA). This method can be considered part of the LLE class of method since it is relies on the idea that if data is sampled from a smooth manifold, then the neighbors of each point remain close and similarly co-located in the low dimensional embedded space. The key idea in LTSA is to use PCA on the neighbors in order to create a locally linear patch. This patch should be considered as an approximation of the tangent space at that point. The coordinates on the tangent space provides a representation on the low dimensional space. LTSA wa introduced by in [Zhang and Zha, 2004] and exposure proposed follow the Section 11 of [Wang, 2012]. Preliminary concepts of Tangent Coordinates and Manifold coordinates can be found in section 2.2.

4.3.1 Local Coordinate Representation

Assume that $\boldsymbol{X} = \{x_1, \cdots, x_n\} \subset \mathcal{M} \subset \mathbb{R}^D$ a neighborhood well defined and $\boldsymbol{G} = [\boldsymbol{X}, \boldsymbol{A}]$ a graph generated by the system. The neighborhood of x_i is denoted by $O(i)$, and the non zero entries of $\boldsymbol{A}$ are defined by $N(i) = \{i_1, \cdots, i_k\}$. Now, let $k = |N(i)|$ and define

$$\hat{x} \overset{def}{=} \frac{1}{k} \sum_{j \in (i)} x_j \tag{4.49}$$

the geometric center of the set $O(i)$[3]

Let $\boldsymbol{T}_i$ be the tangent space of $\mathcal{M}$ at $\hat{x}$, and $\mathfrak{H}_i$ the tangent hyperplane $\mathfrak{H}_i = \hat{x} + \boldsymbol{T}_i$. Now, let $F : O_i \to \mathfrak{H}_i$ be the orthogonal projection such that

$$F(x_j) = \hat{x} + p_j, \quad j \in N(i), p_j \in \boldsymbol{T}_i \tag{4.50}$$

[3]we assume that $\hat{x} \in \mathcal{M}$,but $\hat{x}$ may not be on the $\mathcal{M}$, in this case the nearest point, the project is used.

and due to the properties of linear approximation of $\boldsymbol{T}_i$

$$\frac{1}{k}\sum_{j\in N(i)} p_j = \frac{1}{k}\sum_{j\in N(i)} F(x_j) - \hat{x} \approx 0 \tag{4.51}$$

hence the vector set $\{p_j\}_{j\in N(i)}$ is centered, meaning that

$$\boldsymbol{P}_i = \boldsymbol{P}_i\boldsymbol{C} \tag{4.52}$$

where $\boldsymbol{C}$ is the centering matrix $(k \times k)$ and $\boldsymbol{P}_i = [p_{i_1}, \cdots, p_{i_k}], \quad i \in N(i)$ can be seen as the *span* of $\boldsymbol{T}_i$ for $k > d$.
The next step is to decompose, using SVD, the matrix $\boldsymbol{P}_i$, which takes the following form:

$$\boldsymbol{P}_i = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{4.53}$$

where the column vectors of $\boldsymbol{U} = [u_1, \cdots, u_d] \in \mathbb{R}^{D\times d}$ form an orthonormal basis of $\boldsymbol{T}_i$. From eq. 4.53 we define the *local coordinate matrix* of $\boldsymbol{P}_i$ as

$$\boldsymbol{\Theta}_i \overset{def}{=} \boldsymbol{\Sigma}\boldsymbol{V} = [\tau_{s,j}]_{s,j=1}^{d,k} \tag{4.54}$$

this lead to te following representation of p_j

$$p_j = \sum_{s=1}^{d} \boldsymbol{\tau}_{s,j}\boldsymbol{u}_s, \quad j \in N(i) \tag{4.55}$$

Now, in order to compute the local coordinates $\tau_{s,j}$ we replace p_j by its approximation retrieved by eq. 4.51, $\{x_j - \hat{x}\}_{j\in N(i)}$. The SVD decomposition of the centered matrix $\boldsymbol{X}_i\boldsymbol{C}$ is:

$$\boldsymbol{X}_i\boldsymbol{C} = \boldsymbol{U}_D\boldsymbol{\Sigma}_D\boldsymbol{V}_D^\top \tag{4.56}$$

where $\boldsymbol{U}_D = [u_1, \cdots, u_D]$ is an orthonormal matrix, $\boldsymbol{\Sigma}_D = [\sigma_1, \cdots, \sigma_D]$ is a $D\times D$ diagonal matrix of singular value of $\boldsymbol{X}_i\boldsymbol{C}$. This equation can be reset as follows:

$$\boldsymbol{X}_i\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{U}\boldsymbol{\Theta} \tag{4.57}$$

4.3.2 Global Alignment

The main phase after the representation on the local coordinate is to have a global alignment for each local tangent approximation. It consist to convert a local relation into a global one. Some restrictions on the embedded matrix $\boldsymbol{Y}$ are required:

1. $\boldsymbol{Y}$ has zero mean

2. $\boldsymbol{Y}\boldsymbol{Y}^\top = \boldsymbol{I}$

The reasons of this constrains,namely *centralization constraint* and *orthogonality constraint* on the output data $\boldsymbol{Y}$, relies on the fact that often $\boldsymbol{Y}$ is obtained via maximization of the variance with respect the input similarity matrix.
If we rewrite the local representation in relationship with the manifold we have

$$\boldsymbol{X}_i\boldsymbol{C} = df(\hat{y})\boldsymbol{Y}_i\boldsymbol{H} \tag{4.58}$$

where $df(\hat{y})$ is the derivative at $\hat{y}$, and, since it is invertible $(d(\hat{y}))^{-1} = dh(\hat{x})$ leads to:

$$\boldsymbol{Y}_i\boldsymbol{C} = dh(\hat{x})\boldsymbol{X}_i\boldsymbol{H} = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \quad \boldsymbol{L} \stackrel{def}{=} dh(\hat{x})\boldsymbol{U} \tag{4.59}$$

which leads to express $\boldsymbol{L}$ as follows:

$$\boldsymbol{L} = \boldsymbol{Y}_i\boldsymbol{H}\boldsymbol{V}\boldsymbol{\Sigma}^{-1} \tag{4.60}$$

and, in general,

$$\boldsymbol{Y}_i\boldsymbol{C}(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V}) = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{V}^\top - \boldsymbol{L}\boldsymbol{\Sigma}^\top\boldsymbol{V}\boldsymbol{V}^\top = 0 \tag{4.61}$$

Now, consider a matrix $\boldsymbol{W}_i = \boldsymbol{C}(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V})^\top$ and by 4.61 we can say

$$\boldsymbol{Y}_i \boldsymbol{W}_i = \boldsymbol{0} \tag{4.62}$$

It is possible to create an extended version of them, for $\boldsymbol{W}_i$ we have an $n \times n$ $\boldsymbol{W}^i$ matrix, which is defined by

$$\boldsymbol{W}^i(j,k) = \begin{cases} \boldsymbol{W}_i(s,l) & \text{if} j = i_s, k = i_l \in N(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.63}$$

and for $\boldsymbol{Y}_i$ we have an $k \times n$ $\boldsymbol{Y}^i$ matrix, which is defined by

$$\boldsymbol{Y}^i(j,k) = \begin{cases} \boldsymbol{Y}_i(s,l) & \text{if} j = i_s, k = i_l \in N(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.64}$$

These extension yields

$$\boldsymbol{Y}\boldsymbol{W}^i = 0 \tag{4.65}$$

The kernel is obtained by

$$\boldsymbol{K} = \sum_{i=1}^{n} \boldsymbol{W}^i \tag{4.66}$$

Implementing eq.4.65we have

$$\boldsymbol{Y}\boldsymbol{K} = \sum_{i=1}^{n} \boldsymbol{Y}\boldsymbol{W}^i = \boldsymbol{0} \tag{4.67}$$

and

$$\frac{1}{n}\boldsymbol{1}^\top \boldsymbol{K} = \boldsymbol{0} \tag{4.68}$$

since $\boldsymbol{1}^\top \boldsymbol{W}^i = \boldsymbol{0}$ Thank to symmetry from eqs. 4.67 and 4.3.2 we have

$$\boldsymbol{K}\left[\frac{1}{n}\boldsymbol{1}\boldsymbol{Y}^\top\right] = 0 \tag{4.69}$$

which indicates that we have an orthonormal basis of the null space of the kernel $\boldsymbol{K}$. Finally, we need to prove that $\boldsymbol{K}$ is positive semi definite. This, can easily seen since $\boldsymbol{K}$ is a sum $\boldsymbol{K} = \sum_{i=1}^{n} \boldsymbol{W}^i$ and this sum is positive semi definite if $\boldsymbol{W}^i$ is SPD. $\boldsymbol{W}^i$ is SPD if and only if $\boldsymbol{W}_i$ is SPD, then

$$\begin{aligned} \boldsymbol{W}_i = \boldsymbol{H}(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V}^\top) &= \boldsymbol{H} - \boldsymbol{H}\boldsymbol{V}\boldsymbol{V}^\top\boldsymbol{H} \\ &= \boldsymbol{H}\boldsymbol{I}\boldsymbol{H} - \boldsymbol{H}\boldsymbol{V}\boldsymbol{V}^\top\boldsymbol{H} \\ &= \boldsymbol{H}(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V}^\top)\boldsymbol{H} \end{aligned} \tag{4.70}$$

where $\boldsymbol{V}\boldsymbol{V}^\top$ is the projection matrix, therefore $\|\boldsymbol{V}\boldsymbol{V}^\top\| \leq 1$. This implies that $\boldsymbol{W}_i$ is positive semi definite.

Finally, the embedded and dimensional reduced data set $\boldsymbol{Y}$ is obtained by the solution of the following minimization problem

$$\boldsymbol{Y} = \underset{Y \in \mathbb{R}^{d \times n}}{argmin}\, \mathrm{tr}(\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}^\top) \quad \text{such that} \varepsilon(\boldsymbol{Y}) = 0 \tag{4.71}$$

Then,in order to obtain the reduced data set, select the 2nd -$(d+1)$ smallest eigenvalue of $\boldsymbol{K}$, that corresponding the columns of the matrix $\boldsymbol{Y}$.

To summarize the steps LTSA algorithm we have

4.4 Hessian Local Linear Embedding (HLLE)

An interesting variant of Locally Linearly Embedding (LLE)[4], Hessian locally linear embedding (HLLE), introduced by [Donoho and Grimes, 2003], achieves the linear embedding by minimization of the Hessian Functional on the manifold of the input data (points). In HLLE, a similar approach to Laplacian eigenmaps is used, where a quadratic form based on the Hessian, substitute the Laplacian. The justification and motivation behind the use of the Hessian are based on the concept of linearity, in fact: a linear function has everywhere vanishing Laplacian, but the vice - versa does not hold true, in fact a function with everywhere vanish-

[4]In particular, is more closer to Laplacian eigenvalue techniques than LLE

ing Laplacian is not assure to be linear [Wang, 2012]. Hessian approach overcome to this problem since a function is linear if and only if Hessian is vanishing everywhere. The result from this method is a global embedding which is linear in the local coordinates. The aforementioned property remarks the usability of HLLE in not convex manifold (unlike Isomap).

4.4.1 Hessian on Manifold

Dimensional reduction using HLLE is obtained by setting $\boldsymbol{Y}$ the minimization of the Hessian function on $\mathcal{M}$.
Let $f \in C^2(\mathcal{M}) : \mathcal{M} \longrightarrow \mathbb{R}$ be a smooth function defined on $\mathcal{M}$ and $h = \left[h^1, \cdots, h^d\right]$ the Hessian coordinate of f on $\mathcal{M}$.
The manifold Hessian matrix of f at $\boldsymbol{x}$ with respect to the manifold coordinate is given by:

$$\boldsymbol{H}^{iso}_{\boldsymbol{x}}(f) = \left[\boldsymbol{H}^{iso}_{\boldsymbol{x}}(f)_{ij}\right]^d_{i,j=1}, \quad \boldsymbol{H}^{iso}_{\boldsymbol{x}}(f)_{ij} =^{def} \frac{\partial^2(f \circ g)}{\partial y_i \partial y_j}(\boldsymbol{y}) \tag{4.72}$$

where the derivative is define as

$$\frac{\partial f}{\partial y_i}(\boldsymbol{y}) = \lim_{t \to 0} \frac{f(\boldsymbol{y} + te_i) - f(\boldsymbol{y})}{t} \tag{4.73}$$

where $\{e_i\}^d_{i=1}$ are the canonic coordinate basis of $\mathbb{R}^d$ and $\boldsymbol{y} \in \mathbf{R}^d$.
The corresponding manifold Hessian functional on $C^2(\mathcal{M})$ is defined as:

$$\mathcal{H}^{iso}(f) = \int_{\mathcal{M}} \|\boldsymbol{H}^{iso}_x(f)\|^2_F \tag{4.74}$$

where $\|\cdot\|_F$is the Frobenius norm.
For each $\boldsymbol{x} \in \mathcal{M}$ we have:

$$\begin{aligned} \boldsymbol{H}^{iso}_{\boldsymbol{x}}(h^i) &= \mathbf{0} \quad 1 \leq i \leq d \\ \boldsymbol{H}^{iso}_{\boldsymbol{x}}(e) &= \mathbf{0} \end{aligned} \tag{4.75}$$

where e is constant function on $\mathcal{M}$ i.e. $e(x) = 1$.

The pair of pairwise relation expressed in 4.75 can be reformulated to the global one, namely:

$$\mathcal{H}^{iso}(f) = 0, \quad f = e, h^1, \cdots, h^d \tag{4.76}$$

and, in order to have a global minimization problem, equation 4.76 can be modified as follows:

$$h = \underset{\langle \boldsymbol{F}, \boldsymbol{F} \rangle}{argmin}\, \mathcal{H}^{iso}(h) \quad \boldsymbol{F} = \left[e, h^1, \cdots, h^d\right], \ h^i \in C^2(\mathcal{M}) \tag{4.77}$$

where $\langle \boldsymbol{F}, \boldsymbol{F} \rangle = \boldsymbol{I}$ means that the function set $\left[e, h^1, \cdots, h^d\right]$ forms an orthogonal system in the C^2 space.

The dimensional reduction of the input space is obtain from the solution of 4.77.

Hessian on Tangent Space

Due to fact the the parameterization g in the manifold Hessian functional, cannot be evaluated directly, since the underlying $\mathcal{M}$ is unknown, a computable alternative representation is needed.

Let $T_x\mathcal{M}$ be the tangent space of $\mathcal{M}$ at x and $L_x = x + T_x\mathcal{M}$ be the tangent hyperplane on the vector x. LEt $P = P_X$ be the orthogonal projection from O_x to L_x, where $q = P(p) \in L_x$ is an approximation of p[5].

Define an orthonormal basis of $T_x\mathcal{M}$:

$$\mathcal{B} = \{b_1, \cdots, b_d\}, \quad b_i \in \mathbb{R}^d$$

which lead to have a tangent coordinate representation for each point $t \in T_x\mathcal{M}$

$$t = \sum_{i=1}^{d} t_i b_i \tag{4.78}$$

[5]Since L_x is close to O_x

Now, the directional derivatives in the direction $b_i \in T_x\mathcal{M}$ is:

$$\frac{\partial f}{\partial t_i}(x) = \lim_{s \to 0} \frac{f(x + sb_i) - f(x)}{s} \tag{4.79}$$

for the function $f \in C^2$.

The tangent Hessian functional of $f \in C^2$ at x is defined by

$$\begin{aligned} \boldsymbol{H}_x^{tan}(f) &= \left[\boldsymbol{H}_x^{tan}(f)_{ij}\right] \\ \boldsymbol{H}_x^{tan}(f)_{ij} &= \frac{\partial^2 f}{\partial t_i \partial t_j}(x) \end{aligned} \tag{4.80}$$

and the corresponding Hessian function in the tangent coordinates is defined by:

$$\mathcal{H}^{tant}(f) = \int_{\mathcal{M}} \|\boldsymbol{H}_x^{tan}(f)\|_F^2 \tag{4.81}$$

We can assume the following equivalent relation[6] [Wang, 2012]

$$\mathcal{H}^{tan}(f) = \mathcal{H}^{iso}(f) \tag{4.82}$$

Hence, the minimization problem 4.77 is updated as the following:

$$h = \underset{\langle \boldsymbol{F}, \boldsymbol{F} \rangle}{argmin}\, \mathcal{H}^{tan}(h) \quad \boldsymbol{F} = \left[e, h^1, \cdots, h^d\right], \; h^i \in C^2(\mathcal{M}) \tag{4.83}$$

since using the tangent functional Hessian, compare the tangent hyperplane L_x which can "learned" from the neighborhood of x, the Hessian function is now computable (still in a approximated form).

4.4.2 Discrete form of Hessian functional

The Hessian functional $\mathcal{H}(f)$ can be represented as the quadratic form of f

$$\mathcal{H}(f) = \boldsymbol{f'Kf} \tag{4.84}$$

[6]The formal proof is quite technical and is not presented here, we remand the reader to [Wang, 2012]

where $\boldsymbol{f} = [f(x_1), \cdots, f(x_n)]'$ is a vector representation of a function f and $\boldsymbol{K}$ is an $n \times n$ positive semidefinite matrix, called Kernel (of HLLE).

An important characterization of the Hessian functional [Donoho and Grimes, 2003] is that if $\mathcal{M}$ is locally-isometric to an open connected subset of $\mathbb{R}^d$, then $\mathcal{H}(f)$ has a $(d+1)$ dimensional null space, consisting of the constant functions and a d-dimensional space of functions spanned by the original isometric coordinates. The process to obtain a discrete form of the Hessian require a neighborhood structure on $\mathcal{M}$. As usual, we assume also that the neighborhood system on $\mathcal{X}$ is consistent with the neighborhood structure on $\mathcal{M}$ such that each $\boldsymbol{O}_i$ is a subset of the manifold neighborhood $W_i \subset \mathcal{M}$ with $O_i = \mathcal{X} \cap W_i$ and $\cup_{i=1}^n W_i = \mathcal{M}$ [Wang, 2012].

Let $\{\phi_i\}_{i=1}^m$ be a partition of unity of $\mathcal{M}$ with support $\phi \subset W_i$. The integral over $\mathcal{M}$ can be expressed in the discrete form as follows:

$$\mathcal{H}(f) = \int_{\mathcal{M}} \|\boldsymbol{H}_x^{tan}(f)\|_F^2 = \sum_{i=1}^{m} \int_{W_i} \phi_i \|\boldsymbol{H}_x^{tan}(f)\|_F^2 \tag{4.85}$$

Hence, the discrete form of the integral expressed by the quadratic form is:

$$\mathcal{H}(f)|_{W_i} = \sum_{i=1}^{m} \int_{W_i} \phi_i \|\boldsymbol{H}_x^{tan}(f)\|_F^2 = (\boldsymbol{f_i'} W_i \boldsymbol{f_i}) \tag{4.86}$$

where $\boldsymbol{W}$ is a $k \times k$ positive semidefinite matrix whose null space consists of constant function and coordinate functions on the tangent space $T_{x_i}\mathcal{M}$

The tangent coordinates for $T_{x_i}\mathcal{M}$ are obtained by computing te SVD decomposition of the matrix $\boldsymbol{M}$ defined as follows:

$$\boldsymbol{M}^i = [x_{i_1} - \bar{x}, \cdots, x_{i_k} - \bar{x}] \tag{4.87}$$

where $\bar{x} = \frac{1}{k} \sum_{j \in N(i)} x_j$

The SVD decomposition is:

$$\boldsymbol{M}^i = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{4.88}$$

where $\boldsymbol{U} in \mathbb{R}^{D\times D}$,$\boldsymbol{v} in \mathbb{R}^{k\times k}$ $\sigma \in \mathbb{R}^{D\times k}$ having singular values on its diagonal.
Let $\boldsymbol{U}^d = [u_1, \cdots, u_d]$, $\boldsymbol{V}^d = [v_1, \cdots, v_d]$ be the first d column extracted from $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively. By 4.87 $\boldsymbol{M}^i$ is centered so that $\langle v_j, \mathbf{1}\rangle = 0$ for $j \in [0, d]$ so the column of $\boldsymbol{L}^i = \left[\mathbf{1}, \boldsymbol{V}^d\right]$ form an orthonormal basis for the space of linear function on O_i. Our aim is to obtain an orthonormal basis of the space of all homogeneous quadratic functions.
The column of the matrix

$$\boldsymbol{Q}^i = [v_l \boxtimes v_j]_{1\leq l\leq j\leq d} \tag{4.89}$$

form a basis of all homogeneous quadratic functions on O_i, while a basis of all quadratic functions on O_i is given by:

$$\boldsymbol{B}^i = \left[\boldsymbol{L}^i \;\; \boldsymbol{Q}^i\right] \tag{4.90}$$

If we orthonormalize $\boldsymbol{B}^i$ we obtain:

$$\boldsymbol{B}^i_{on} = \left[\boldsymbol{L}^i \;\; \boldsymbol{Q}^i_{on}\right] \tag{4.91}$$

we obtain an orthonormal basis of the space of quadratic function. The result sought,an orthonormal basis of the space of all homogeneous quadratic functions, is given precisely by the column of the $\boldsymbol{Q}$ matrix,

$$\boldsymbol{Q}^i_{on} = [q_1, \cdots, q_r], \quad r = \frac{d(d+1)}{2} \tag{4.92}$$

Finally, we can define:

$$\boldsymbol{H}^i = (\boldsymbol{Q}^i_{on}) \tag{4.93}$$

this lead to the discrete local Hessian form, which is, $\boldsymbol{W}_i$

$$\boldsymbol{W}_i = (\boldsymbol{H}^i)^\top \boldsymbol{H}^i \tag{4.94}$$

matched with the following result:

$$\begin{aligned} \mathbf{1}^\top \boldsymbol{W}_i \mathbf{1} &= 0 \\ \boldsymbol{v}_j^\top \boldsymbol{W}_i \boldsymbol{v}_j &= 0, \quad 1 \leq j \leq d \\ \boldsymbol{q}_j^\top \boldsymbol{W}_i \boldsymbol{q}_l &= \delta_{ij}, \quad 1 \leq j \leq l \leq d \end{aligned} \tag{4.95}$$

The final step is to construct the Kernel previously defined 4.84.
Reproducing the trick used in LTSA we can define a $n \times n$ matrix $\boldsymbol{W}^i$ with all zero entries expect for

$$\boldsymbol{W}^i(N(i), N(i)) = \boldsymbol{W}_i \tag{4.96}$$

where $\boldsymbol{W}^i(N(i)N(i))$ denote a $k \times k$ submatrix of $\boldsymbol{W}^i$. Then, the kernel of HLLE is

$$\boldsymbol{K} = \sum_{i=1}^{n} \boldsymbol{W}^i \tag{4.97}$$

Lastly, to obtain the embedded data $\boldsymbol{Y} = \left[Y^0, \cdots, Y^d\right]$ we pick up $d+1$ eigenvectors corresponding to the $d+1$ smallest eigenvalue of $\boldsymbol{K}$.
To summarize the algorithm of HLLE consist of the following step.

Chapter 5

Application to Financial Data

5.1 Synthetic Data

In this section some examples of Dimensional Reduction on synthetic data set are exposed. The idea to use some, well known, synthetic dataset is to give a simple insight into the concept of dimensionality reduction and to point out the difference between linear and non-linear techniques.
We define the following synthetic surfaces:

- **Swiss Roll**: the parametric formula presented here is from [Marsland, 2009]:

$$\begin{cases} x = -\phi cos(\phi) \\ y = \psi \\ z = \phi cos(\phi) \end{cases} \tag{5.1}$$

  where ϕ is a random number sampled uniformally from $[1.5\pi, 4.5\pi]$ and ψ is a random number sampled uniformally from $[0, n - samples]$.(Figure 5.1a) [1]

- **S Curve**: another popular manifold learning artificial dataset is given by

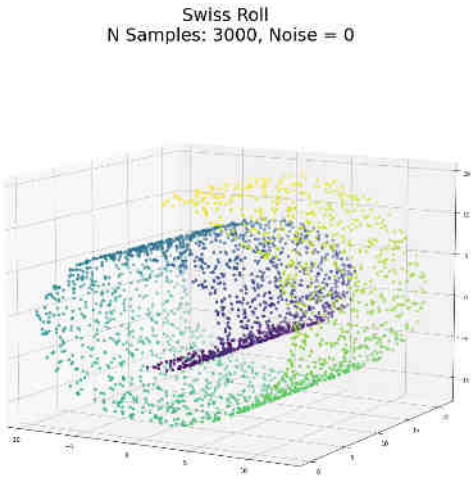
[1] The exact formula is not stated in various papers [Tenenbaum et Al., 2000] that have implemented it and a slightly modify version can be found in literature

the S curve. It's define as follow:

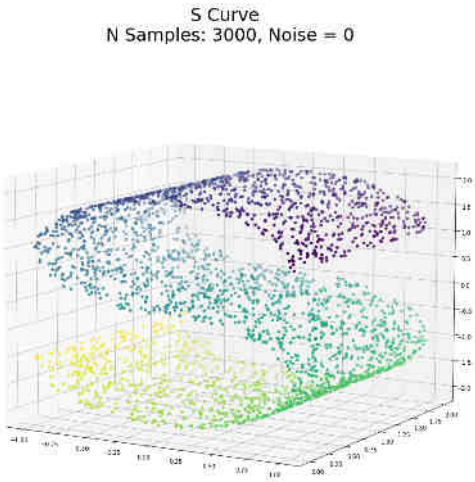
$$\begin{cases} x = sin(\phi) \\ y = 2 \cdot \psi \\ z = |\phi| \cdot (cos(\phi) - 1) \end{cases} \tag{5.2}$$

where ϕ is a random number sampled uniformally from $[1.5\pi, 4.5\pi]$ and ψ is a random number sampled uniformally from $[0, n - samples]$ (Fig. 5.1b)

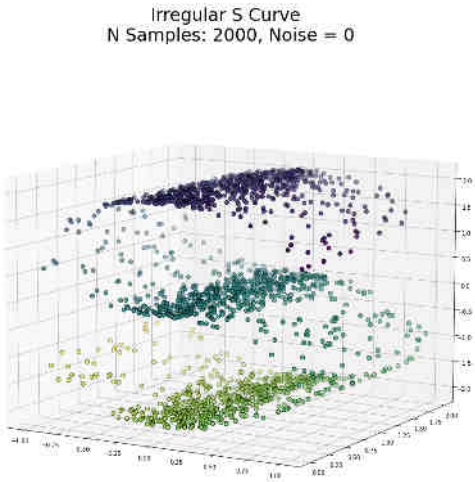
In addition to these dataset can be created their own variant version by adding Gaussian noise, holes (in order to obtain a non convex surface) or with different distribution of data on surface in order to obtain an irregular dataset.



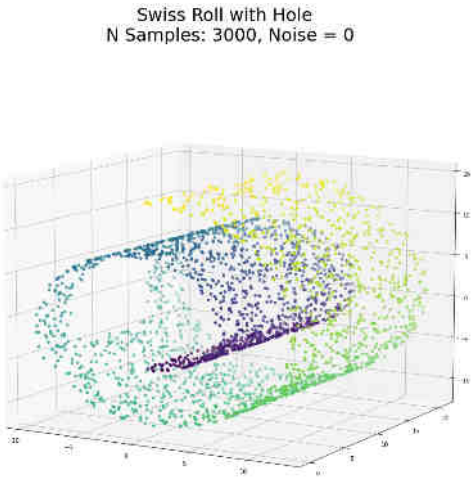
(a) Swiss Roll



(b) S Curve



(c) Irregular S curve



(d) Swiss Roll with hole

Figure 5.1: Synthetic Dataset

5.1.1 Dimensional Reduction

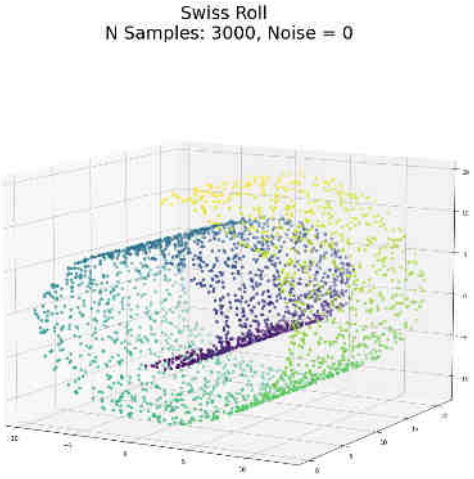
In this section are exposed results of dimensional reduction techniques on synthetic data. More in detail, the surfaces from 3 dimensions have been projected into 2 dimensional space. Methods and techniques that has been used for this test are:

- **Linear:** PCA, MDS

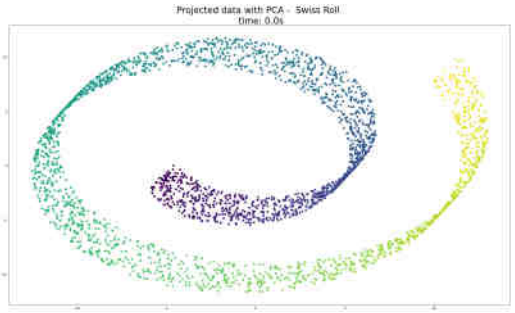
- **Non Linear:** Isomap,LLE,MLLE,HLLE,LTSA

with the number of neighbors k (for Isomap) fixed at 10. The plot of the starting surface in displayed at the top while the reduced form on the bottom line. Dataset were generated by using 3000 random point on selected surfaces.

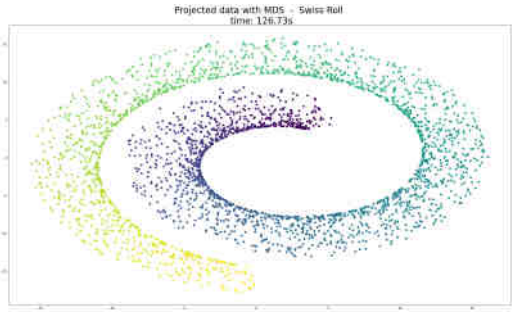
Swiss Roll



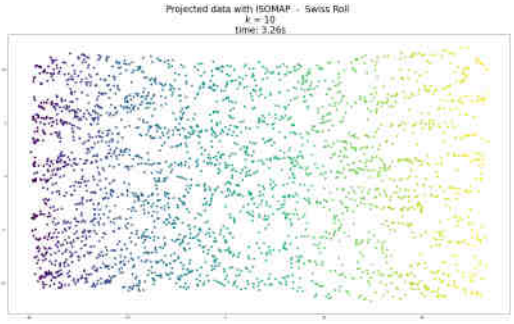
(a) Swiss Roll



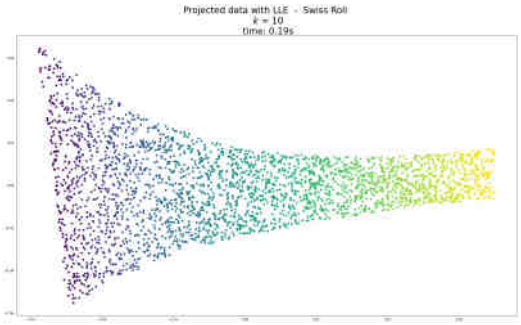
(b) PCA - 2d dimensional reduction



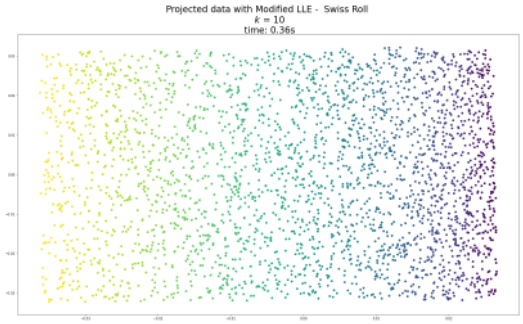
(c) MDS - 2d dimensional reduction



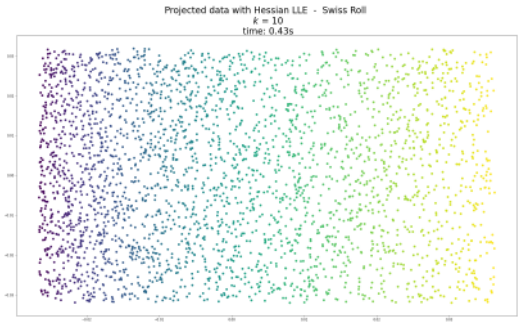
(d) Isomap - 2d dimensional reduction



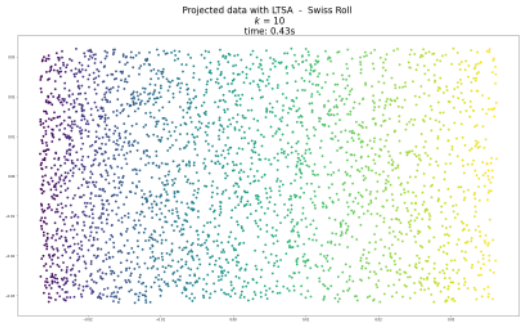
(e) LLE - 2d dimensional reduction



(f) MLLE - 2d dimensional reduction



(g) Hessian LLE - 2d dimensional reduction



(h) LTSA- 2d dimensional reduction

Figure 5.2: 2 dimensional reduced swiss roll data set. Linear techniques such as PCA and MDS are not able to correctly unfold the dataset. Global non linear method like Isomap are able to unfold and give a correct representation of low embedding data. LLE, a classic local non linear method suffers and has some issues to mapping correctly, issues that are overcomes by the other non vanilla local methods.

Without loss in generality we can assume, since the surface is a convex set, that techniques like PCA and MDS will not perform well as the other one. Conversely, techniques such as Isomap that are expected to work on convex set and expandable surfaces, should work better.The results show that Isomap worked very well and unfolded correctly the surface. Local method such as LLE suffers to represent correctly the global surface,issues that are overcomes by the other non vanilla local methods. Linear techniques, PCA and MDS, are not able to correctly unfold the dataset.

In Figure 5.3 is shown how the reconstruction error of Isomap decreases as the number of neighbors k increases. The reconstruction error is defined as follow:

$$E = \|\boldsymbol{K}(D) - \boldsymbol{K}(D_{fit})\|_2^2 \tag{5.3}$$

where D is the distance matrix of the input data $\boldsymbol{X}$, D_{fit} is the distance matrix

of the embedding data $\boldsymbol{Y}$ and $\boldsymbol{K}$ is the Isomap Kernel.

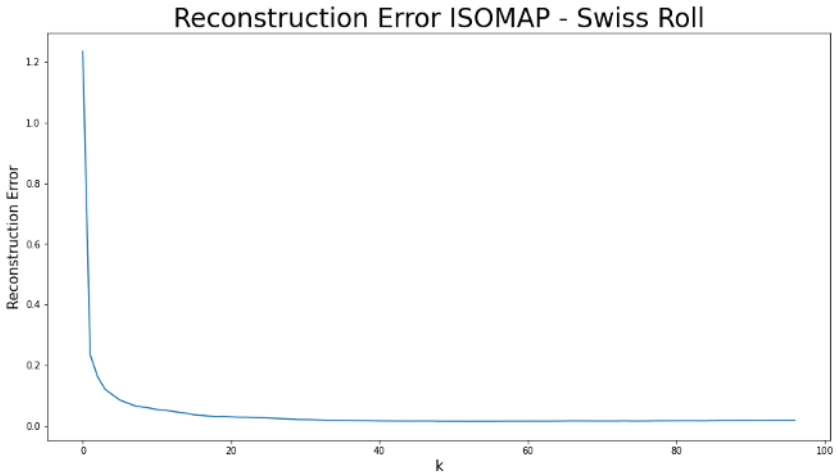


Figure 5.3: Isomap Reconstruction Error in fuction of k nearest neighbors

In Figure 5.4 there is the graph based representation of the embedded surface obtained using Isomap.

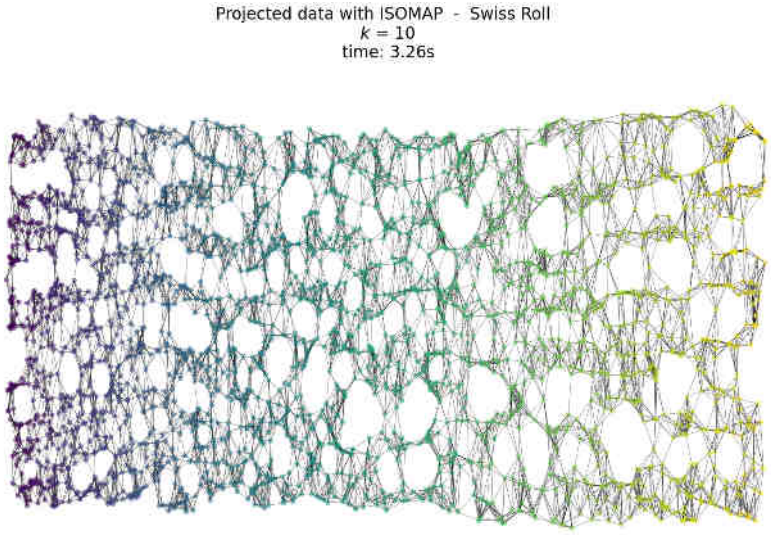
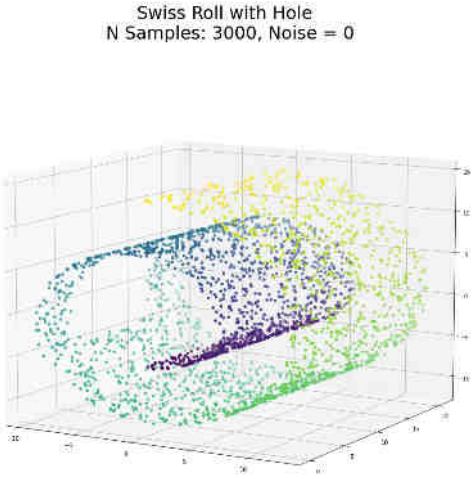
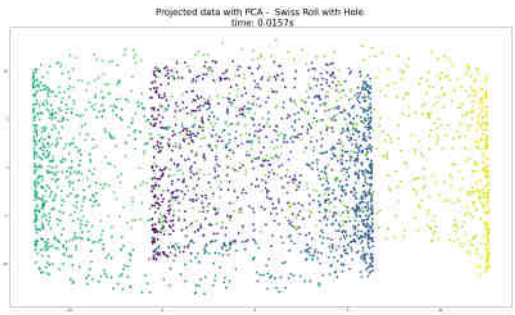


Figure 5.4: Isomap embedded surface connected with graph.

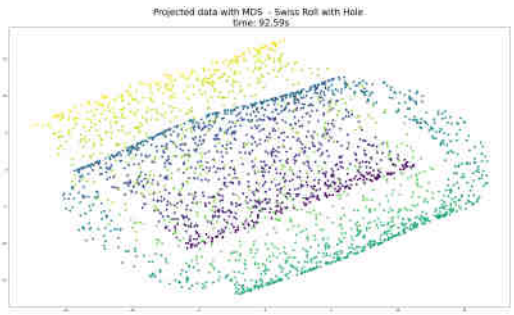
Swiss Roll with Hole In this case the variant swiss roll with a hole is presented. This variant serves to highlight some limits of global methods (Isomap), in the presence of a not well defined global structure.



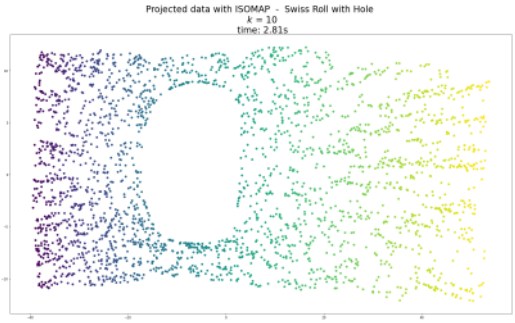
(a) Swiss Roll with Hole



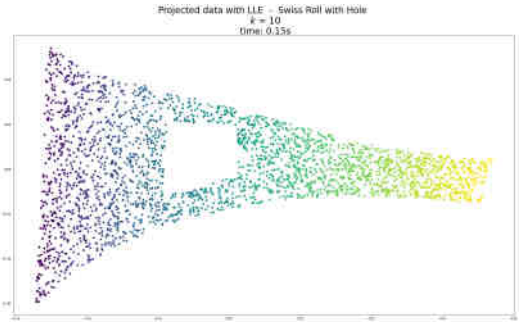
(b) PCA - 2d dimensional reduction



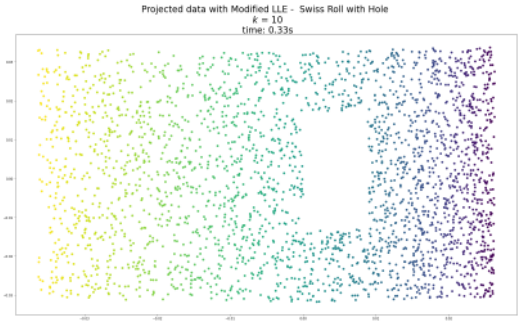
(c) MDS - 2d dimensional reduction



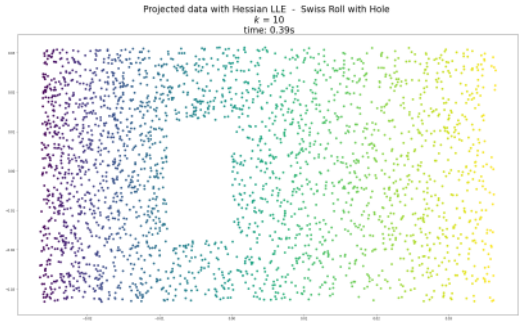
(d) Isomap - 2d dimensional reduction



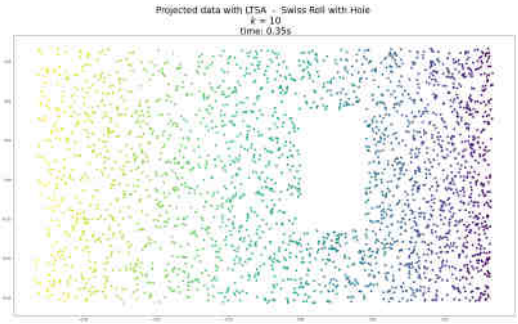
(e) LLE - 2d dimensional reduction



(f) MLLE - 2d dimensional reduction



(g) Hessian LLE - 2d dimensional reduction



(h) LTSA- 2d dimensional reduction

Figure 5.5: 2 dimensional reduced Swiss roll with Hole data set. Linear techniques such as PCA and MDS are not able to correctly unfold the dataset. Global non linear method like Isomap are able to unfold and give a partially correct (the hole is correctly mapped) representation of low embedding data. LLE, suffers and has some issues to mapping correctly, the other local techniques give the best representation in 2 dimension

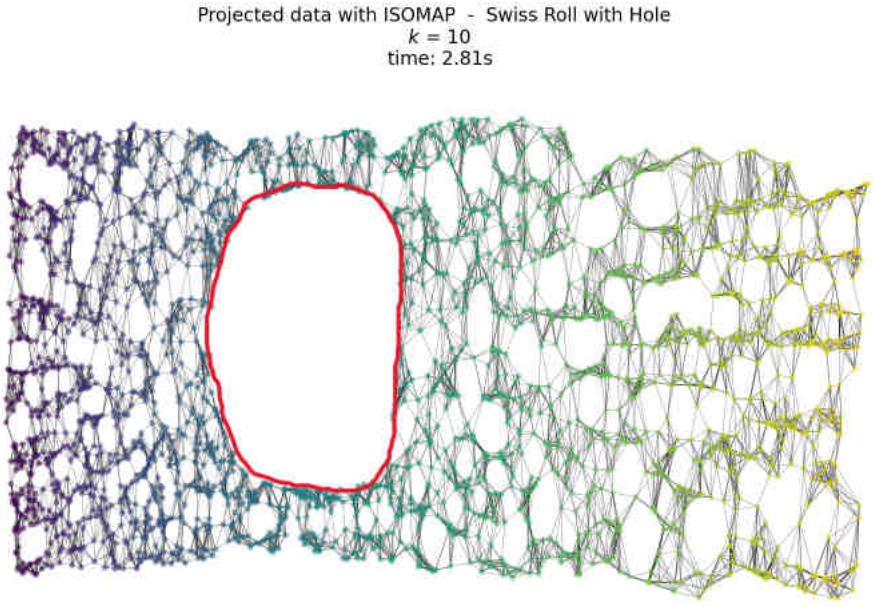


Figure 5.6: Isomap embedded surface connected with graph. Area near the hole is highlighted in red

In this case, Swiss Roll with Hole, one of the main assumption of Isomap, the geodesically convexity of data set, is not guaranteed. In fact, as you can see from the Figure 5.6, Isomap causes a distortion near the hole. Distortion that does not occur in local techniques such as LTSA, Figure 5.7.

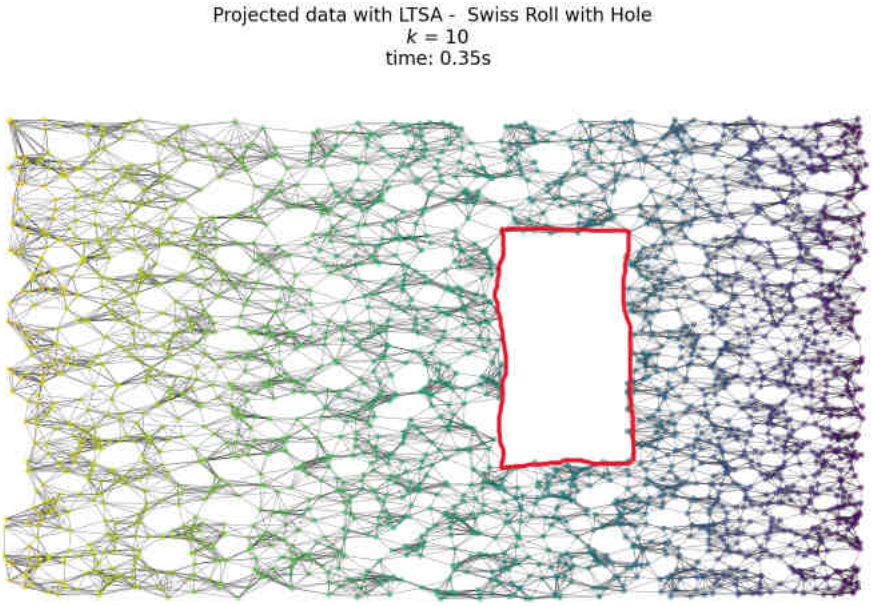


Figure 5.7: LTSA embedded surface connected with graph. Area near the hole is highlighted in red

Further examples regarding others synthetic datasets are presented in Annex B.

5.2 Application to Financial Data

This chapter implements and extends the intuition and the work proposed by [Bahadur et Al., 2017] in *A Study of Russell 3000 Dimensionality Using non-linear Dimensionality Reduction Techniques.*

The goal is to compute a dimension reduction of the US market using the Russell 3000[2] index as a proxy. The idea is that financial markets are a complex object described on high-dimensional manifolds and it is possible to represent them in low dimensional manifolds that preserves the characteristic of high dimensional data [Huang et Al., 2016]. Starting from this assumption, combined with the hypothesis that manifold where data lies is indeed not linear [Christofides et Al., 2016], therefore an approach using nonlinear techniques among linear is presented. In addition to find the expected results (consistent with the original work) we have extended the temporal window, and we are going to investigate if in this temporal extension leads to confirm of what has been achieved previously: in presence of situations of market stress, the dimensionality drops. To decide a common metric on how many dimensions retain we have followed what proposed in [Bahadur et Al., 2017], that is to take the first p components that express 90% of the variance (See Algorithm 9).

5.2.1 Description of Data

The dataset that has been used for this application are the daily Adjusted Close prices Russell 3000 constituents. These data are collected from 5958 trading days[3] from January 1^{st}, 2000 to March 24, 2022. We denote the dataset as $\boldsymbol{X}_{ij} \in \mathbb{R}^{n \times d}$ where n are the samples and d the feature (i.e. the constituents).

Russell 3000 is a stock market index composed by the 3000 largest publicly held companies based in USA. It represent, approximately 97% of the American public equity market [FTSE Russell - An LSEG Business]. The motivation to use this index is that it can be seen as benchmark of the US Stock Market.

[2] [FTSE Russell - An LSEG Business]

[3] trading days are 5 days a week excluding some holidays.

The collecting and data cleaning process involved the following step:

- Collect all the historical constituent (tickers) for both indices from January 2000 to March 2023. The constituent list is issued by [FTSE Russell - An LSEG Business]

- collect and download all *Adjusted Close* price of tickers historical series. If data were unavailable for a certain period, the column was dropped out. The data provider is `http://finance.yahoo.com/`

- data quality controls: simple data quality checks have been implemented to verify the consistency of the data. For example, it has been verified that the data are not *nan* and that they are positive ($\boldsymbol{X}_{ij} > 0$)

- compute the log return:

$$R_j[i] = \frac{\log(Close_{adj}[i])}{\log[Close_{adj}[i-1]]} \tag{5.4}$$

  where i is the ith row of dataset for a fixed j column.

- then, the final dataset denoted by $\boldsymbol{X}_{ij} \in \mathbb{R}^{n \times d}$, is composed by n samples and d feature. In Table 5.1 there is an example of the final dataset.

Russell 3000 Adj Close from 01/01/2000 to 24/03/2022					
Date	A	AA	$\cdots$	ZY	ZION
03/01/2000	-0.0711	-0.0251	$\cdots$	-0.0381	-0.06432
04/04/2000	-0.0794	0.0046	$\cdots$	-0.0145	-0.0496
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
22/03/2020	0.0082	-0.0089	$\cdots$	0.02621	0.0340
23/03/2022	-0.0373	0.0471	$\cdots$	0.0138	0.0441

Table 5.1: Russell 3000 dataset. Constituents are retrieved from [FTSE Russell - An LSEG Business] and data from `http://finance.yahoo.com/`.

5.2.2 Dimensionality Reduction

In this section is proposed a comparison with various techniques for dimensionality reduction, both linear and non linear. The proposed approach is similar to [Bahadur et Al., 2017], and it can be summarize in the following steps:

- Dimensionality reduction is calculated over 60 days moving window (move by 1 day) of daily log return. See Fig 5.8 for a schematic chart.

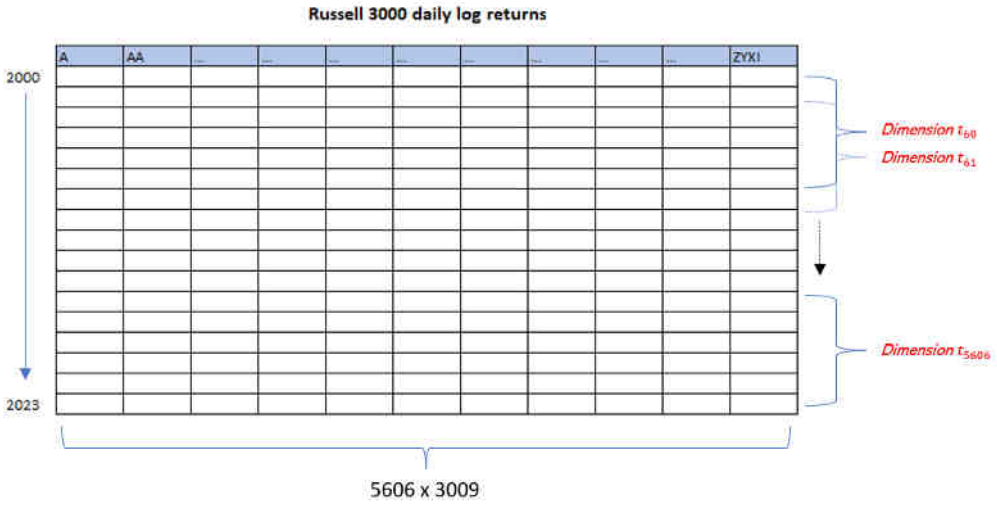


Figure 5.8: Dimension estimation of the time series using 60 days moving window (move by 1 day) (Inspired by [Bahadur and Paffenroth, 2019]).

- Check if the input data is a non linear manifold. If the manifold $\mathcal{M}$ is linear then the Euclidean distances between each pair of points should be equal to geodesic distance. If the manifold $\mathcal{M}$ is non linear then the Euclidean distance should be always less then the geodesic one (Figure 5.9).

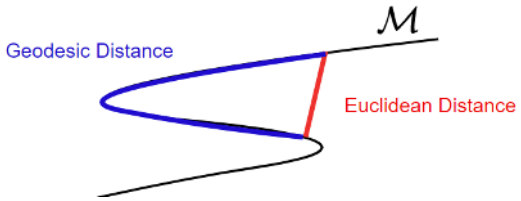


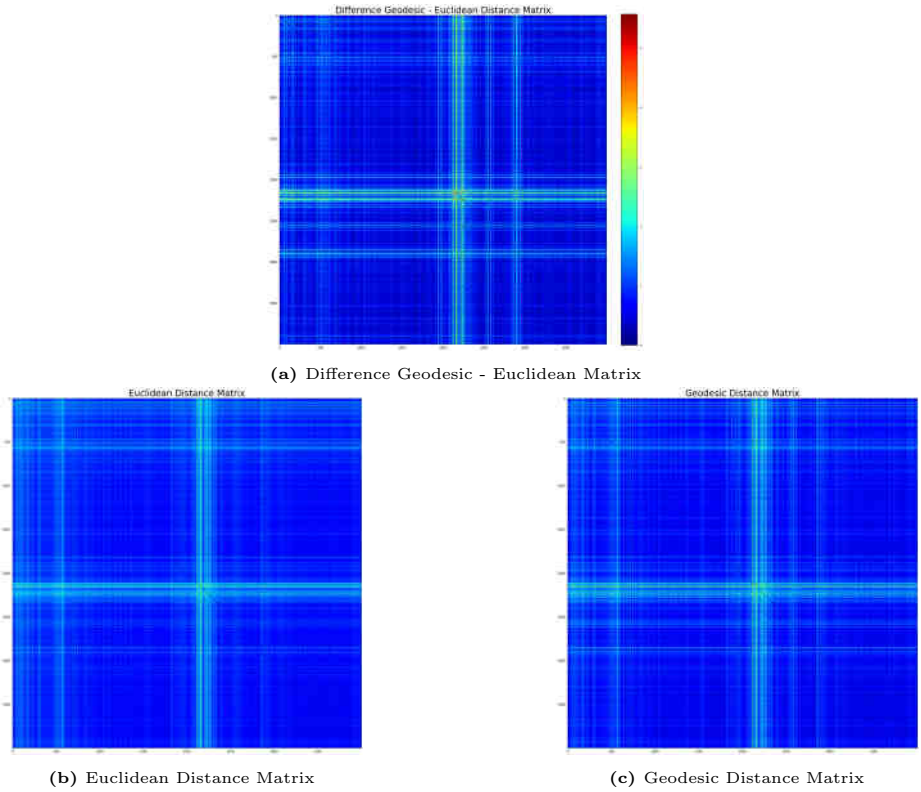
Figure 5.9: Difference Geodesic - Euclidean Distance. Can easily seen why Geodesic distance should be greater than the Euclidean

In Figure 5.14 are exposed the Distance matrix Euclidean (Figure 5.10b) and Geodesic (Figure 5.10c). Delta Matrix (Figure 5.10a), is obtained by

the difference of G (the Geodesic Distance matrix) and E (Euclidean Distance Matrix). Since

$$\Delta_{ij} = G_{ij} - E_{ij} > 0 \tag{5.5}$$

then $\mathcal{M}$ is indeed non linear [Falasca and Bracco, 2021].



(a) Difference Geodesic - Euclidean Matrix

(b) Euclidean Distance Matrix

(c) Geodesic Distance Matrix

Figure 5.10: Distance Matrix of Russell 3000 dataset.

- Now, we run the dimensionality reduction techniques, namely: PCA, MDS, Isomap, LLE, HLLE and LTSA to obtain a lower dimensional dataset. In the next bullets the pseudo codes used for the implementation are shown.

– *PCA implementation*: in Algorithm 3 is shown the PCA pseudo code [Shlens, 2003]

Algorithm 3 PCA

Input: p number of feature of reduced space, $\boldsymbol{X}^{n\times d}$ input data

Output: $\boldsymbol{Y}^{n\times p}$ reduced space

1: Compute the product $\boldsymbol{X}^\top\boldsymbol{X} = \sum_{i=1}^{N}(x_i-\mu)^\top(x_i-\mu)$

2: Eigen Analysis $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$

3: Compute eigenvector $\boldsymbol{U} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}$

4: Select p numbers of component $\boldsymbol{U}^d = [u_1, \cdots, u_d]$

5: Compute p feature $\boldsymbol{Y} = \boldsymbol{U}^d\boldsymbol{X}$

– *SMACOF implementation*: in Algorithm 4 is shown the SMACOF pseudo code [Leeuw and Mair,2009]

Algorithm 4 SMACOF Algorithm

Input: $\boldsymbol{X}$ dissimilarity matrix

Output: $\sigma_r(\boldsymbol{X}^{[k]})$ Stress minimization

1: Set $\boldsymbol{Z} = \boldsymbol{X}^{[0]}$, where $\boldsymbol{X}^{[0]}$ is a random configuration.

2: Set $k = 0$

3: Compute $\sigma_r(\boldsymbol{X}^{[0]})$

4: Compute *Guttam Transform* given by 3.44 or 3.45 depending on w_{ij}

5: If $\sigma_r(\boldsymbol{X}^{[k-1]}) - \sigma_r(\boldsymbol{X}^{[k]}) < \varepsilon$ (ε small, positive, constant) or k = maximum number of iterations, stop. If not set $\mathbf{Z} = \boldsymbol{X}^{[k]}$ and go to step 3.

– *MDS implementation*: in Algorithm 5 is shown the MDS pseudo code [Borg and Groenen, 2005]

Algorithm 5 Classic and generalized MDS

Input: p number of feature of reduced space, $\boldsymbol{X}^{n\times d}$ input data

Output: $\boldsymbol{Y}^{n\times p}$ reduced space

1: Compute the dissimilarity matrix $d_{ij} = \|x_i^2 - x_j^2\|_2^2$
2: Calculate the matrix $\boldsymbol{D} = -\frac{1}{2}d_{ij}^2$ and $\boldsymbol{K} = \boldsymbol{HDH}$ where $\boldsymbol{H}$ is the centering matrix
3: Compute the spectral decomposition of $\boldsymbol{K} = \boldsymbol{V\Delta V}^\top$ and $\boldsymbol{U}$
4: Select p greatest eigenvalue from $\boldsymbol{\Sigma}$
5: Construct the $n \times p$ dimensional embedding matrix $\boldsymbol{Y} = \boldsymbol{V\Delta}^{\frac{1}{2}}$ or using the SMACOF algorithm (4) $\boldsymbol{Y} = SMACOF(\boldsymbol{D})$

– *Isomap implementation*: in Algorithm 6 is shown the Isomap pseudo code [Ghojogh et Al., 2020]

Algorithm 6 Isomap algorithm

Input:

Input: p number of feature of reduced space, $\boldsymbol{X}^{n\times d}$ input data, number of nearest neighbors k

Output: $\boldsymbol{Y}^{n\times p}$ reduced space

1: Fore each point in $\mathbf{X}$ select the k nearest point as neighbors
2: Compute the Euclidean distance between all neighbor nodes, $\boldsymbol{D} = \left[d_{ij}^2\right]_{n\times n}$ in order to convert the data set into a graph
3: For each pairs of nodes in the graph, finds the points $\mathcal{G} = \{x_i \| i = 1, \cdots, k\}$ in the shortest paths using the Dijkstra or Floyd-Warshal algorithm, and, assign to $\mathbf{D}$ distance matrix.
4: Convert matrix of distance into a Gram matrix by double centering $\boldsymbol{K} = -\frac{1}{2}\boldsymbol{HDH}$
5: Compute the spectral decomposition $\boldsymbol{K} = \boldsymbol{V\Delta V}^\mathsf{T}$
6: Estimate $\mathbb{R}^{p\times n} \ni \boldsymbol{Y} = \boldsymbol{\Delta}^{\frac{1}{2}}\boldsymbol{V}^\mathsf{T}$

– *HLLE implementation*: in Algorithm 7 is shown the HLLE pseudo code [Donoho and Grimes, 2003]

Algorithm 7 HLLE

Input: p number of feature of reduced space, $\boldsymbol{X}^{n\times d}$ input data

Output: $\boldsymbol{Y}^{n\times p}$ reduced space

1: Neighborhood and neighbors k definition. Constrain on $K \geq r$ where $r = \frac{(d+2)(d+1)}{2}$
2: Compute local tangent coordinate functional using PCA to local data $\boldsymbol{X}$
3: Local Hessian functional construction $\boldsymbol{W}_i = \boldsymbol{Q}^i(\boldsymbol{Q}^i)^\top$
4: HLLE kernel construction by setting $\boldsymbol{K} = n\ timesn$ matrix and update by $\boldsymbol{K}(N(i), N(i)) = \boldsymbol{K}(N(i), N(i)) + \boldsymbol{W}_i$
5: Eigen decompositio of kernel. Select $\boldsymbol{Y}$ as the eigenvector matrix corresponding to the $d+1$ smallest eigenvalue of $\boldsymbol{K}$

– *LTSA implementation*: in Algorithm 8 is shown the LTSA pseudo code [Zhang and Zha, 2004]

Algorithm 8 LTSA

Input: p number of feature of reduced space, $\boldsymbol{X}^{n\times d}$ input data

Output: $\boldsymbol{Y}^{n\times p}$ reduced space

1: Neighborhood and neighbors k definition of the $G = [\boldsymbol{X}, \boldsymbol{A}]$ data graph
2: Find local coordinate relation by applying PCA on $\hat{\boldsymbol{X}}$. Compute $\boldsymbol{W}_i = \boldsymbol{I} - \boldsymbol{G}_i\boldsymbol{G}^\top$
3: LTSA kernel construction via global alignment. Set $\boldsymbol{K} = 0$ and update it by $\boldsymbol{K}(N(i), N(i)) = \boldsymbol{I} - \boldsymbol{G}_i\boldsymbol{G}_i^\top$ for $i = 1, 2, \cdots, n$
4: Eigen decomposition of kernel by the spectral decomposition $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$. Select $\boldsymbol{Y}$ as the eigenvector matrix corresponding to the 2nd -$(d+1)$ smallest eigenvalue of $\boldsymbol{K}$

- The selecting criteria to determine p dimension of the $\boldsymbol{Y}$ embedded dataset, is the following: *select all the component until reach the 90 % of variance*

[Bahadur et Al., 2017]. Dimensionality of the embedded dataset, using this criteria should be interpret as the minimum number of instruments that contribute to reach 90% of the index variance. (cumulative variance threshold). A sketch of implementation and pseudocode can be found in Algorithm 9.

Algorithm 9 Dimensionality using up to 90% variance

Input: $(\sigma_1, \sigma_2, \cdots, \sigma_n)$ - singular values

Output: p, the number of largest squared singular values that explains 90% of variance in $(\mathbf{X})$.

Sort $(\sigma_1, \sigma_2, \cdots, \sigma_n)$ in descending order, where σ_i are singular values.

Calculate $\sigma_{sum} = \Sigma_{i=1}^{n} \sigma_i^2$

Calculate $\sigma_{i\%} = \frac{\sigma_i^2}{\sigma_{sum}}$

Dimensionality p is the value of l where $\Sigma_{l=1}^{w} \sigma_{i\%} \geq 90\%$

- Then, the reduced dataset $\boldsymbol{Y}_{ij} \in \mathbb{R}^{n \times p}$, where $p \leq d$ is the number of the lower embedded dimension, is obtained.

5.2.3 Dimensionality Reduction Results

The Figure 5.11 displays the time series dimensionality reduction of Russell 3000. As can be seen and clearly understood from the chart 5.11, linear techniques such as PCA (blue) and MDS (green) have a greater dimensionality than the nonlinear technique such as Isomap (in orange). This means that with the same variance, the result of the nonlinear techniques is to be preferred. The trend of the index is plotted in red, and as we can see in the proximity of a drop of the index corresponds to a decrease in dimensionality.

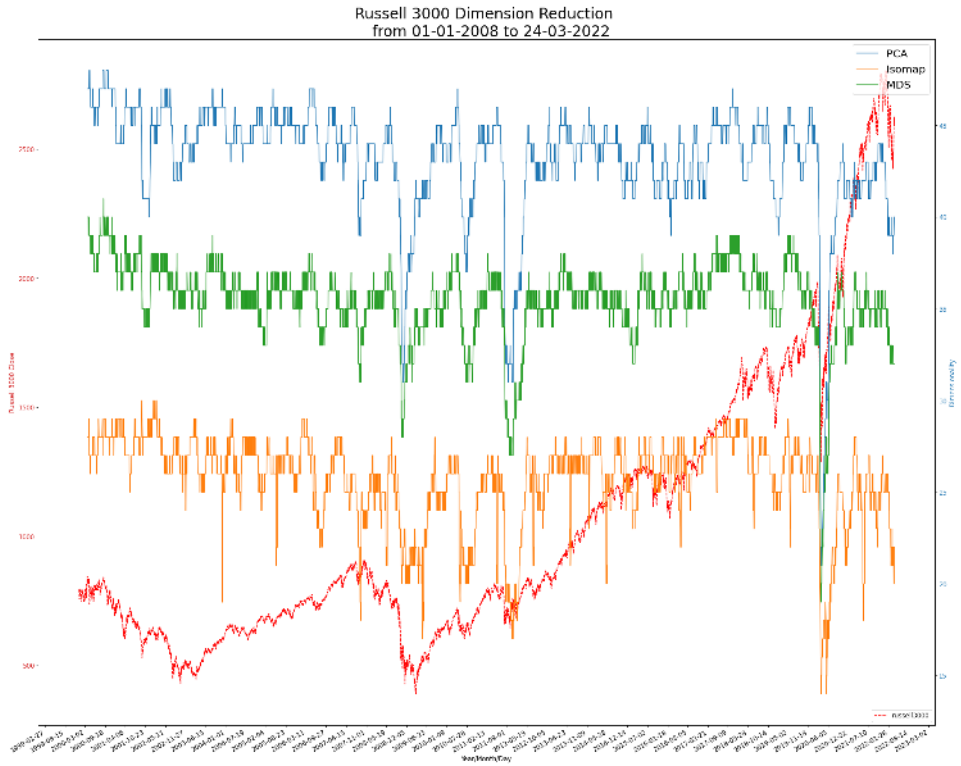


Figure 5.11: Russel 3000 dimension reduction from 2000 to 23/03/2022. PCA in blue, MDS in green, Isomap in orange, Russell3000 index in red.

In Table 5.2 are exposed the average, maximum and the minimum value of the dimensionality reduction that we have obtained. As we can see the spread PCA-Isomap is thinned in the minimums while it is greater in the maximum.

	Techniques		
Dimension	PCA	MDS	ISOMAP
Average	43.2	35.4	25.6
Max	48	41	30
Min	21	19	14

Table 5.2: Average, Max and Min value of dimension over the time horizon.

Leaving aside the various drops, the dimensionality of the PCA fluctuates between 40-45, MDS between 30 and 35 and the Isomap between 22 and 27.
A landscape image in Figure 5.14 is proposed to capture more details.
Locally Linear family methods, such as LLE, HLLE and LTSA, has been implemented and tested to estimate the dimensionality too. However, Figure 5.12

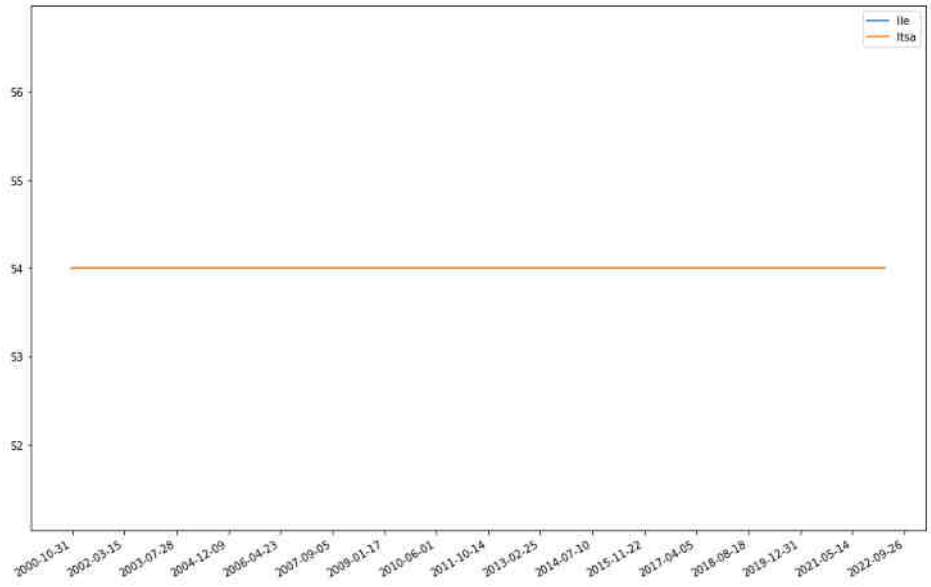


Figure 5.12: Russel 3000 dimension reduction with Locally Linear method: LLE and LTSA. Local method fails to unfold correctly the manifold and 90% variance dimensionality is stuck at 54 for both techniques. Also HLLE has been tested producing the same result.

shows how local model fails, since the dimensionality is constant and fixed over the time at 54. Recalling the toy example, local method acts well when hypothesis of convexity fails and there is a well defined locally structure, but for complex dataset, verifying which manifold assumptions have been respected is not easy. In Figure 5.13 is shown the difference between PCA-Isomap and MDS-Isomap.

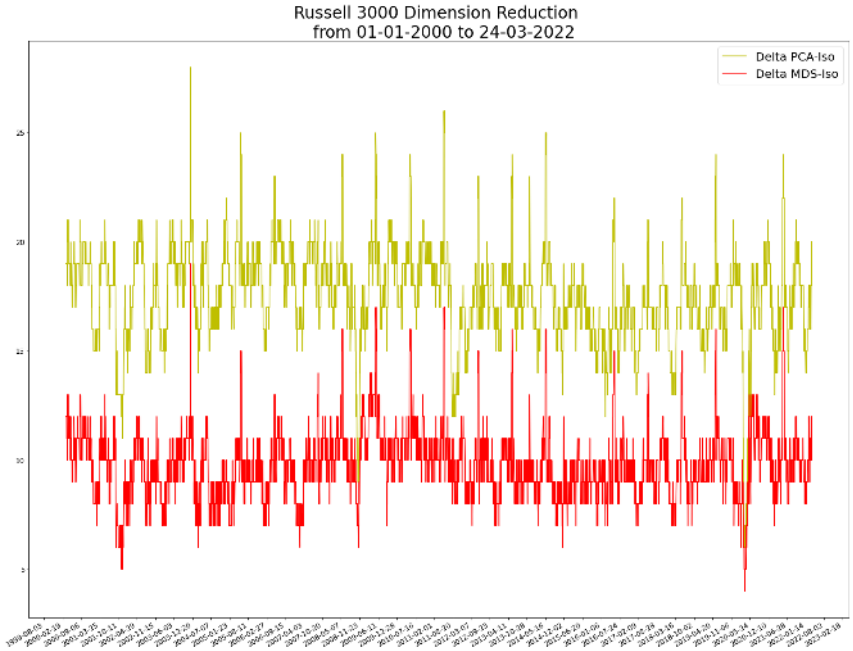


Figure 5.13: Russel 3000 dimension reduction. Difference between PCA-Isomap and MDS-Isomap.

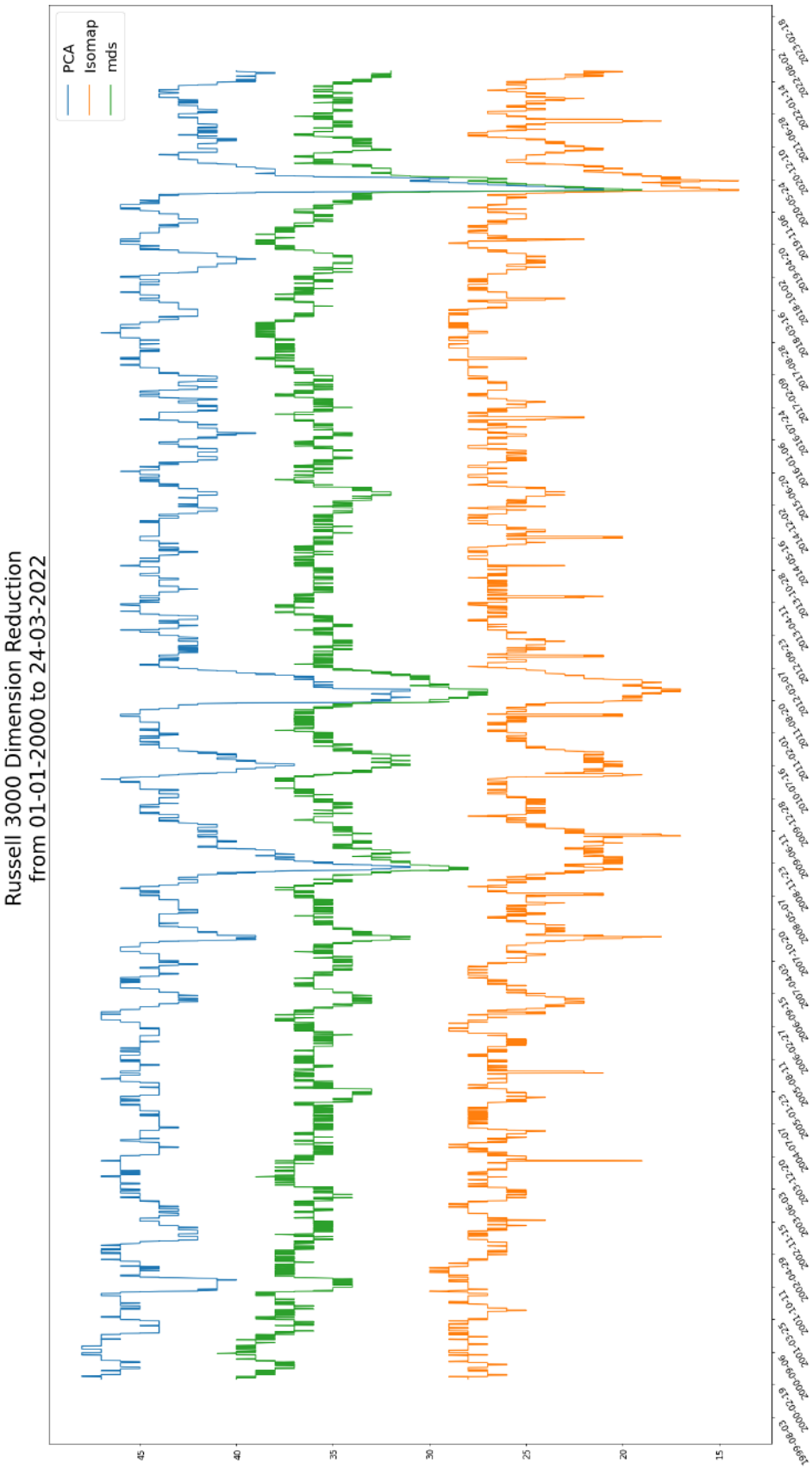


Figure 5.14: Russel 3000 dimension reduction from 01/01/2000 to 23/03/2022. PCA in blue, MDS in green, Isomap in orange

5.2.4 Crash Analysis

Financial and Banking Crisis - 2008

The Global financial crisis of 2008 , related to US housing bubbling, an excessive risk-taking and a lack of risk management by global financial institutions culminates with Lehman Brother collapse in September 15,2008. Russell 3000 index drops nearly 7.5% in a couple of days (15-17 Septmber 2008) as illustrated in Figure 5.16. In this scenario, the change of dimensionality is less accentuate in non linear (Isomap) rather than the linear (PCA - MDS). The "slope" (rate of change) of the Isomap dimensionality indicates that drops beforehand linear dimensionality drops. In Table 5.3 are shown dimension reduction of Isomap, MDS and PCA respectively over the period: 01/09/2008 - 31/12/2008.

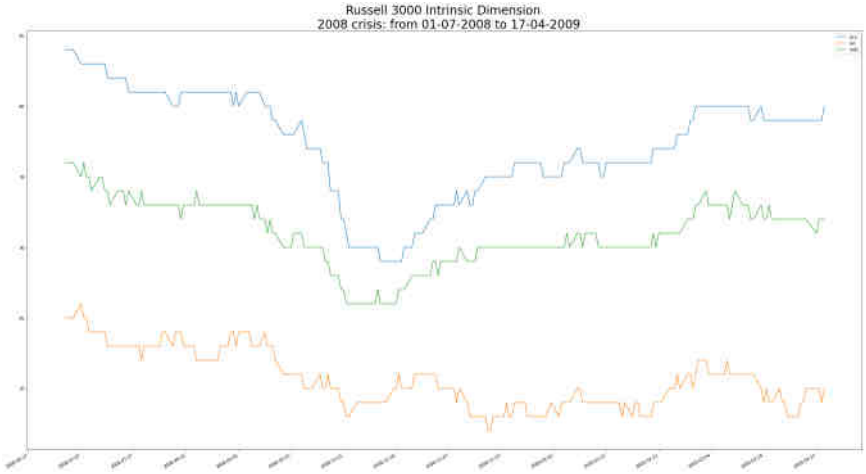


Figure 5.15: Russel 3000 dimension reduction during 2008 crisis. Period from 01/07/2008 to 14/04/2009. PCA in blue, MDS in green, Isomap in orange

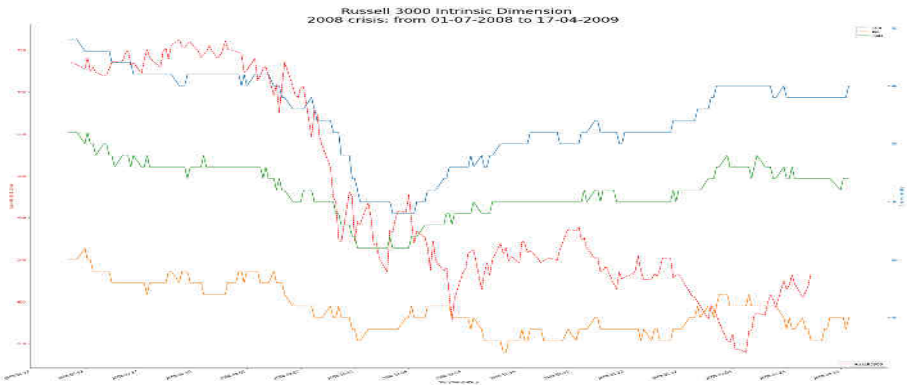


Figure 5.16: Russel 3000 intrinsic dimension estimation during 2008 crisis. Period from 01/07/2008 to 14/04/2009. PCA in blue, MDS in green, Isomap in orange,Russell 3000 in red

Date	Isomap	MDS	PCA
01/09/2008	24	35	43
02/09/2008	25	35	43
03/09/2008	25	35	42
04/09/2008	25	35	42
05/09/2008	26	35	42
08/09/2008	25	35	42
09/09/2008	25	35	42
10/09/2008	25	35	42
11/09/2008	25	35	42
12/09/2008	25	35	42
15/09/2008	25	34	42
16/09/2008	25	34	42
17/09/2008	25	35	42
18/09/2008	25	34	42
19/09/2008	24	33	41
22/09/2008	24	33	41
23/09/2008	23	33	40
24/09/2008	23	33	40
25/09/2008	23	33	40
26/09/2008	23	33	40
29/09/2008	24	33	40
30/09/2008	23	33	41
01/10/2008	23	32	40
02/10/2008	22	32	39
03/10/2008	23	32	40
06/10/2008	22	33	39
07/10/2008	23	32	39
08/10/2008	23	32	38
09/10/2008	22	32	38
10/10/2008	23	32	38
13/10/2008	22	31	37
14/10/2008	22	31	36
15/10/2008	21	29	35
16/10/2008	21	30	34
17/10/2008	21	29	33
20/10/2008	20	28	33
21/10/2008	21	29	33
22/10/2008	20	28	33
23/10/2008	21	29	33
24/10/2008	21	29	32
27/10/2008	21	28	32
28/10/2008	21	28	32
29/10/2008	21	29	32
30/10/2008	21	28	32
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
31/10/2008	21	28	31
03/11/2008	21	28	31
04/11/2008	22	29	31
05/11/2008	22	29	31
06/11/2008	22	29	31
07/11/2008	22	29	31
10/11/2008	23	29	32
11/11/2008	23	29	32
12/11/2008	23	30	32
13/11/2008	23	30	33
14/11/2008	23	30	33
17/11/2008	22	30	33
18/11/2008	22	31	33
19/11/2008	23	31	34
20/11/2008	23	31	34
21/11/2008	23	31	35
24/11/2008	21	31	35
25/11/2008	21	31	35
26/11/2008	20	31	35
27/11/2008	21	31	36
28/11/2008	20	31	36
01/12/2008	20	31	35
02/12/2008	21	32	36
03/12/2008	20	31	35
04/12/2008	21	31	36
05/12/2008	21	32	36
08/12/2008	21	32	36
09/12/2008	21	32	36
10/12/2008	20	31	37
11/12/2008	20	32	37
12/12/2008	20	32	37
15/12/2008	20	32	37
16/12/2008	20	32	37
17/12/2008	21	32	37
18/12/2008	20	32	37
19/12/2008	20	33	37
22/12/2008	20	33	38
23/12/2008	21	33	38
24/12/2008	21	33	38
26/12/2008	20	33	38
29/12/2008	20	33	38
30/12/2008	20	33	38
31/12/2008	20	33	37

Table 5.3: Financial and Banking Crisis - 2008 dimension reduction. Dimensionality of Isomap, MDS and PCA respectively. Period: 01/09/2008 - 31/12/2008

European debt crisis - 2011

The European debt crisis, erupted when several eurozone state were unable to pay their government debt. The climax occurs on 8 August 2011, when Athens stock market crashed, in that day Russell3000 drops nearly 7%. The large drop, illustrated in Figure 5.18 (red dashed line represent the Russell3000 index) is accompanied by a decline in dimensionality. The blue line, which represents the PCA, decrease with a similar rate to Isomap (but with different magnitude), more tenuous instead MDS. In Table 5.4 are shown dimension reduction of Isomap, MDS and PCA respectively over the period: 01/07/20011 - 31/10/2011.

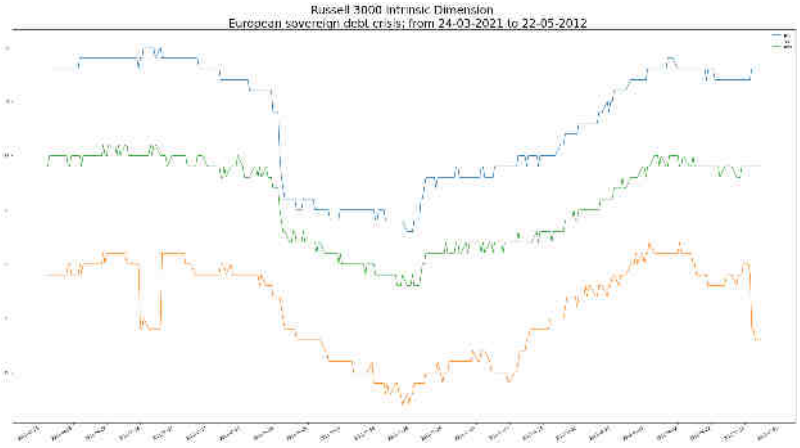


Figure 5.17: Russel 3000 intrinsic dimension estimation during European debt crisis. Period from 19/03/2011 to 22/05/2012. PCA in blue, MDS in green, Isomap in orange

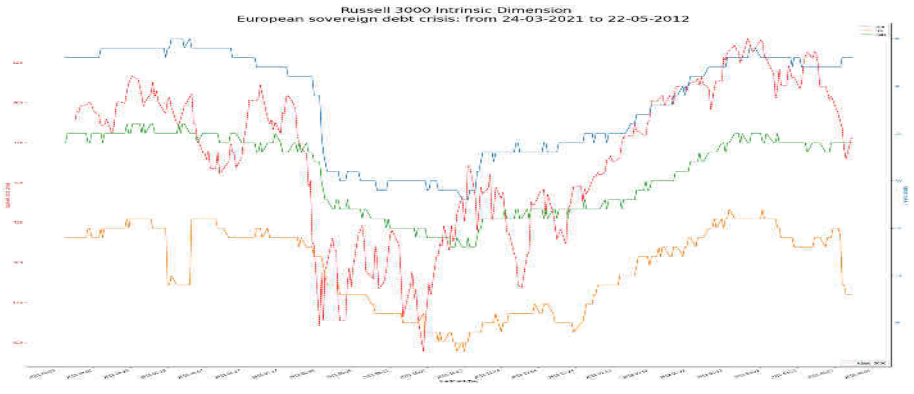


Figure 5.18: Russel 3000 intrinsic dimension estimation during European debt crisis. Period from 19/03/2011 to 22/05/2012. PCA in blue, MDS in green, Isomap in orange,Russell3000 index in red

Date	Isomap	MDS	PCA
01/07/2011	25	36	44
04/07/2011	26	36	44
05/07/2011	26	36	44
06/07/2011	26	35	43
07/07/2011	26	35	43
08/07/2011	26	35	43
11/07/2011	26	36	43
12/07/2011	26	36	43
13/07/2011	25	35	43
14/07/2011	25	36	43
15/07/2011	25	35	43
18/07/2011	25	34	43
19/07/2011	25	35	43
20/07/2011	25	35	43
21/07/2011	25	35	43
22/07/2011	25	35	43
25/07/2011	25	35	43
26/07/2011	24	35	43
27/07/2011	25	34	43
28/07/2011	25	34	43
29/07/2011	24	34	42
01/08/2011	25	35	42
02/08/2011	25	34	42
03/08/2011	25	35	42
04/08/2011	24	34	42
05/08/2011	24	34	42
08/08/2011	24	33	40
09/08/2011	24	34	40
10/08/2011	23	32	37
11/08/2011	21	31	35
12/08/2011	21	30	34
15/08/2011	21	29	33
16/08/2011	21	30	33
17/08/2011	21	29	33
18/08/2011	21	30	33
19/08/2011	21	30	33
22/08/2011	20	30	32
23/08/2011	19	30	32
24/08/2011	20	30	33
25/08/2011	19	30	33
26/08/2011	20	30	33
29/08/2011	20	30	33
30/08/2011	20	30	33
31/08/2011	20	29	32
⋮	⋮	⋮	⋮

Date	Isomap	MDS	PCA
⋮	⋮	⋮	⋮
01/09/2011	20	29	32
02/09/2011	20	29	32
05/09/2011	20	29	32
06/09/2011	20	29	32
07/09/2011	20	28	32
08/09/2011	20	29	32
09/09/2011	19	28	32
12/09/2011	19	29	31
13/09/2011	19	28	31
14/09/2011	19	28	31
15/09/2011	19	28	32
16/09/2011	20	28	32
19/09/2011	20	28	32
20/09/2011	19	28	32
21/09/2011	20	28	32
22/09/2011	19	28	32
23/09/2011	19	28	32
26/09/2011	19	27	32
27/09/2011	19	28	32
28/09/2011	19	28	32
29/09/2011	19	28	32
30/09/2011	19	28	32
03/10/2011	19	28	33
04/10/2011	19	28	33
05/10/2011	18	28	32
06/10/2011	18	27	32
07/10/2011	18	28	32
10/10/2011	19	28	32
11/10/2011	19	27	32
12/10/2011	18	28	32
13/10/2011	18	27	32
14/10/2011	18	27	32
17/10/2011	18	28	32
18/10/2011	18	27	32
19/10/2011	17	27	32
20/10/2011	18	27	31
21/10/2011	18	27	31
24/10/2011	17	27	31
25/10/2011	18	27	31
26/10/2011	17	27	31
27/10/2011	17	27	31
28/10/2011	17	27	31
31/10/2011	17	27	31

Table 5.4: European debt crisis - 2011 dimension reduction. Dimensionality of Isomap, MDS and PCA respectively. Period: 01/07/20011 - 31/10/2011

Covid-19 Pandemic

Due to the increasing growing and instability of the spread of Covid-19 around the world, on 20 February 2020 stock market crashed. Russell 3000 drops 12.5% in 7 days as shown in Figure 5.20. Also in this situation, see Figure 5.19, can be recognized the pattern in the change of dimensionality, with a more moderate drop in the case of Isomap with respect the linear techniques. The difference in dimensionality between these category becomes thinner in case of drops, but is still notable. In Table 5.5 are shown dimension reduction of Isomap, MDS and PCA respectively over the period: 01/02/2020 - 31/05/2020.

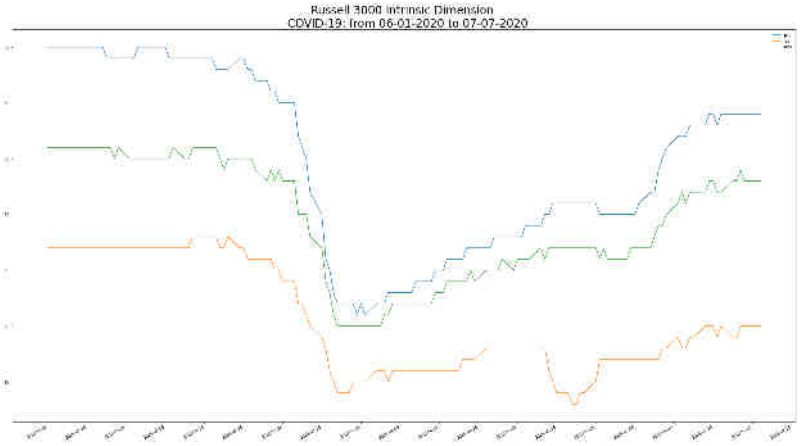


Figure 5.19: Russel 3000 intrinsic dimension estimation during the initial spread of Covid-19. Period from 06/01/2020 to 07/07/2020. PCA in blue, MDS in green, Isomap in orange

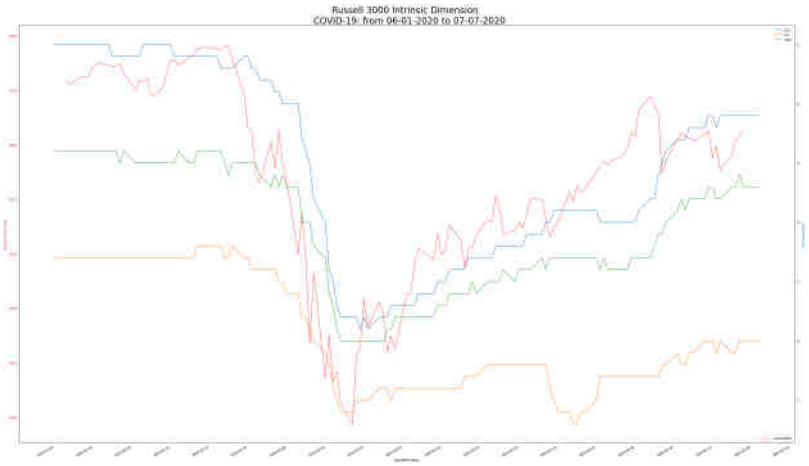


Figure 5.20: Russel 3000 intrinsic dimension estimation during the initial spread of Covid-19. Period from 06/01/2020 to 07/07/2020. PCA in blue, MDS in green, Isomap in orange,Russell3000 index in red

Date	Isomap	MDS	PCA
01/02/2021	25	34	40
02/02/2021	25	33	40
03/02/2021	26	35	42
04/02/2021	26	35	42
05/02/2021	26	35	42
08/02/2021	26	35	42
09/02/2021	26	35	42
10/02/2021	27	35	42
11/02/2021	27	34	42
12/02/2021	27	35	42
15/02/2021	27	36	42
16/02/2021	28	36	41
17/02/2021	28	36	41
18/02/2021	28	36	42
19/02/2021	28	36	42
22/02/2021	27	37	42
23/02/2021	28	37	42
24/02/2021	27	37	43
25/02/2021	28	36	43
26/02/2021	28	36	42
01/03/2021	28	37	42
02/03/2021	28	36	42
03/03/2021	28	37	42
04/03/2021	28	36	42
05/03/2021	28	36	42
08/03/2021	28	36	42
09/03/2021	28	36	42
10/03/2021	27	36	41
11/03/2021	27	36	42
12/03/2021	27	36	41
15/03/2021	27	36	41
16/03/2021	27	35	42
17/03/2021	27	36	41
18/03/2021	27	35	41
19/03/2021	27	35	41
22/03/2021	26	35	42
23/03/2021	26	35	42
24/03/2021	25	35	41
25/03/2021	26	35	41
26/03/2021	25	35	42
29/03/2021	25	35	42
30/03/2021	24	35	42
31/03/2021	25	35	42
⋮	⋮	⋮	⋮

Date	Isomap	MDS	PCA
⋮	⋮	⋮	⋮
01/04/2021	25	35	42
05/04/2021	25	35	42
06/04/2021	25	35	42
07/04/2021	25	36	42
08/04/2021	25	35	42
09/04/2021	25	35	41
12/04/2021	26	35	41
13/04/2021	26	35	41
14/04/2021	25	35	42
15/04/2021	25	34	41
16/04/2021	25	35	41
19/04/2021	26	35	41
20/04/2021	26	35	41
21/04/2021	26	34	41
22/04/2021	25	34	41
23/04/2021	25	35	41
26/04/2021	25	35	41
27/04/2021	25	35	41
28/04/2021	25	35	41
29/04/2021	24	34	41
30/04/2021	25	35	41
03/05/2021	24	35	42
04/05/2021	24	35	42
05/05/2021	24	35	42
06/05/2021	24	35	42
07/05/2021	24	35	42
10/05/2021	24	35	42
11/05/2021	24	34	42
12/05/2021	20	35	42
13/05/2021	20	35	42
14/05/2021	20	35	42
17/05/2021	20	35	42
18/05/2021	19	35	42
19/05/2021	19	35	42
20/05/2021	19	34	42
21/05/2021	18	35	42
24/05/2021	20	35	42
25/05/2021	20	35	42
26/05/2021	20	35	42
27/05/2021	20	35	42
28/05/2021	20	35	42
31/05/2021	20	35	42

Table 5.5: Covid-19 dimension reduction. Dimensionality of Isomap, MDS and PCA respectively. Period: 01/02/2020 - 31/05/2020

Russian invasion of Ukraine - 2022

After a long and prolonged tense atmosphere, Russia invaded Ukraine on 24 February 2022. Russell 3000 drops almost 6% in 6 days as shown in Figure 5.22. Also in this situation, see Figure 5.21, can be recognized the pattern in the change of dimensionality, with a more moderate drop in the case of MDS, almost imperceptible decline in PCA case. Isomap dimensionality drop is more accentuate raging from 26 of 03/01/2022 to 21/03/2022, showing a greater sensitivity to market stess. In Table 5.6 are shown dimension reduction of Isomap, MDS and PCA respectively over the period: 01/01/2022 - 23/03/2022.

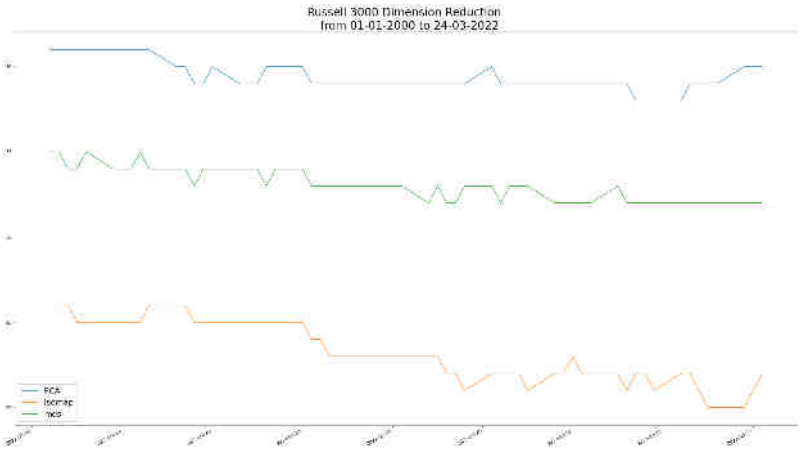


Figure 5.21: Russel 3000 intrinsic dimension estimation during the Russian invasion of Ukraine - 2022. Period from 01/01/2022 to 23/04/2022. PCA in blue, MDS in green, Isomap in orange

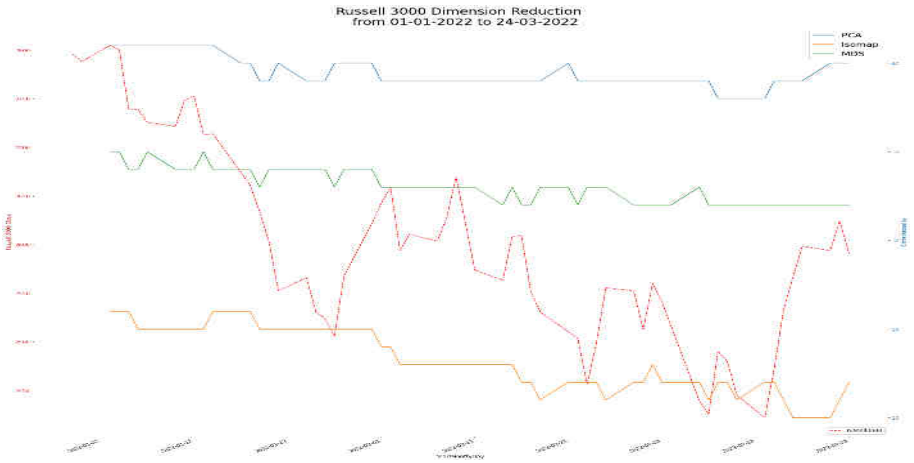


Figure 5.22: Russel 3000 intrinsic dimension estimation during the Russian invasion of Ukraine - 2022. Period from 01/01/2022 to 23/04/2022. PCA in blue, MDS in green, Isomap in orange, Russell3000 index in red

Date	Isomap	MDS	PCA
03/01/2022	26	35	41
04/01/2022	26	35	41
05/01/2022	26	34	41
06/01/2022	25	34	41
07/01/2022	25	35	41
10/01/2022	25	34	41
11/01/2022	25	34	41
12/01/2022	25	34	41
13/01/2022	25	35	41
14/01/2022	26	34	41
17/01/2022	26	34	40
18/01/2022	26	34	40
19/01/2022	25	33	39
20/01/2022	25	34	39
21/01/2022	25	34	40
24/01/2022	25	34	39
25/01/2022	25	34	39
26/01/2022	25	34	39
27/01/2022	25	33	40
28/01/2022	25	34	40
31/01/2022	25	34	40
01/02/2022	24	33	39
02/02/2022	24	33	39
03/02/2022	23	33	39
04/02/2022	23	33	39
07/02/2022	23	33	39
08/02/2022	23	33	39
09/02/2022	23	33	39
10/02/2022	23	33	39
11/02/2022	23	33	39
14/02/2022	23	32	39
15/02/2022	23	33	39
16/02/2022	22	32	39
17/02/2022	22	32	39
18/02/2022	21	33	39
21/02/2022	22	33	40
22/02/2022	22	32	39
23/02/2022	22	33	39
24/02/2022	22	33	39
25/02/2022	21	33	39
28/02/2022	22	32	39
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
01/03/2022	22	32	39
02/03/2022	23	32	39
03/03/2022	22	32	39
04/03/2022	22	32	39
07/03/2022	22	33	39
08/03/2022	21	32	39
09/03/2022	22	32	38
10/03/2022	22	32	38
11/03/2022	21	32	38
14/03/2022	22	32	38
15/03/2022	22	32	39
16/03/2022	21	32	39
17/03/2022	20	32	39
18/03/2022	20	32	39
21/03/2022	20	32	40
22/03/2022	21	32	40
23/03/2022	22	32	40

Table 5.6: Russian invasion of Ukraine - 2022, dimension reduction. Dimensionality of Isomap, MDS and PCA respectively. Period: 01/021/2022 - 23/03/2022

Chapter 6

Conclusion

In this thesis we have approached the problem of Dimensional Reduction highlighting the difference between linear and non-linear techniques, in particular, introducing the concept of *Manifold Learning*.
After a first introductory part, designed to expose and present both the key mathematical concepts and the Manifold Learning techniques, a series of visual examples is presented, in order to get a better insight into the DR process and to highlight the difference between these techniques.
Finally, an application of *Dimensionality Reduction* to financial data, in particular to the Russell 3000 index, is proposed. This last work can be seen as an extension of [Bahadur et Al., 2017].

We estimate the temporal dimensionality of the Russell 3000 index, therefor of US Market. Three techniques were implemented and used: Principal Component Analysis, Multidimensional Scaling and Isomap. The number of dimension, i.e. the constituent of the index, of reduced data set was selected by the criteria of 90% of variance.

In conclusion, we observed the benefit of using non-linear techniques compared to linear ones, in our particular case Isomap, a global non-linear technique that uses geodesic distance instead of the Euclidean one, proved to be the most efficient. The results obtained, in addition to confirming what was obtained in

the work of [Bahadur et Al., 2017], also confirmed what appears to be a well-defined pattern, namely that a decreasing in dimensionality is associated with a decreasing - stress condition of the market (then reflected to the index). We have seen how both in the recent cases of Covid-19 and with the most recent event of the Ukrainian invasion by Russia, they show the same characteristics in the change of dimensionality.

Further solution can come from Statistical Manifold Learning, where an information metric in a probabilistic space replace the geodesic metric. Since prices of financial instruments are stochastic, Statistical Learning could grasp further details and information from data.

Appendix A

Proofs

A.1 Multidimensional Scaling

Proof of relation 3.12 [Ghojogh et Al., 2020]:

Considering

$$\begin{aligned} \mathbf{X}^\mathsf{T}\mathbf{X} &= \mathbf{V}\boldsymbol{\Delta}\mathbf{V}^\mathsf{T} \\ \mathbf{Y}^\mathsf{T}\mathbf{Y} &= \mathbf{Q}\boldsymbol{\Psi}\mathbf{Q}^\mathsf{T} \\ \mathbf{M} &= \mathbf{V}^\mathsf{T}\mathbf{Q} \end{aligned} \tag{A.1}$$

$$\begin{aligned} \|\mathbf{X}^\mathsf{T}\mathbf{X} - \mathbf{Y}^\mathsf{T}\mathbf{Y}\|_F^2 &= \mathbf{tr}\left[(\mathbf{X}^\mathsf{T}\mathbf{X} - \mathbf{Y}^\mathsf{T}\mathbf{Y})^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{X} - \mathbf{Y}^\mathsf{T}\mathbf{Y})\right] \\ &= \mathbf{tr}\left[(\mathbf{X}^\mathsf{T}\mathbf{X} - \mathbf{Y}^\mathsf{T}\mathbf{Y})^2\right] \\ &= \mathbf{tr}\left[(\mathbf{V}\boldsymbol{\Delta}\mathbf{V}^\mathsf{T} - \mathbf{Q}\boldsymbol{\Psi}\mathbf{Q}^\mathsf{T})^\mathbf{2}\right] \\ &= \mathbf{tr}\left[(\mathbf{V}\boldsymbol{\Delta}\mathbf{V}^\mathsf{T} - \mathbf{V}\mathbf{V}^\mathsf{T}\mathbf{Q}\boldsymbol{\Psi}\mathbf{Q}^\mathsf{T}\mathbf{V}\mathbf{V}^\mathsf{T})^\mathbf{2}\right] \\ &= \mathbf{tr}\left[(\mathbf{V}(\boldsymbol{\Delta} - \mathbf{V}^\mathsf{T}\mathbf{Q}\boldsymbol{\Psi}\mathbf{Q}^\mathsf{T}\mathbf{V})^\mathbf{2}\mathbf{V}^\mathsf{T})^\mathbf{2}\right] \\ &= \mathbf{tr}\left[(\mathbf{V}^\mathsf{T})^\mathbf{2}\mathbf{V}^\mathbf{2}(\boldsymbol{\Delta} - \mathbf{V}^\mathsf{T}\mathbf{Q}\boldsymbol{\Psi}\mathbf{Q}^\mathsf{T}\mathbf{V})^\mathbf{2}\right] \\ &= \mathbf{tr}\left[(\mathbf{V}^\mathsf{T}\mathbf{V})^\mathbf{2}(\boldsymbol{\Delta} - \mathbf{V}^\mathsf{T}\mathbf{Q}\boldsymbol{\Psi}\mathbf{Q}^\mathsf{T}\mathbf{V})^\mathbf{2}\right] \\ &= \mathbf{tr}\left[(\boldsymbol{\Delta} - \mathbf{V}^\mathsf{T}\mathbf{Q}\boldsymbol{\Psi}\mathbf{Q}^\mathsf{T}\mathbf{V})^\mathbf{2}\right] \\ &= \mathbf{tr}\left[(\boldsymbol{\Delta} - \mathbf{M}\boldsymbol{\Psi}\mathbf{M}^\mathsf{T})^\mathbf{2}\right] \end{aligned} \tag{A.2}$$

A.2 Sammon Mapping

Proof of 4.25, gradient of cost function of Sammon mapping [Lee and Verleysen, 2007].

According to chain rule, we have:

$$\frac{\partial c_4}{\partial y_{i,k}} = \frac{\partial c_4}{\partial d_y(y_i, y_j)} \times \frac{\partial d_y(y_i, y_j)}{\partial c_4} \tag{A.3}$$

Then, the first and the second derivative are:

$$\frac{\partial c_4}{\partial d_y(y_i, y_j)} = -\frac{2}{a} \sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j)} \tag{A.4}$$

$$\frac{\partial d_y(y_i, y_j)}{\partial y_{i,k}} = \frac{\partial d_y(y_i, y_j)}{\partial^2 d_y(y_i, y_j)} \times \frac{\partial^2 d_y(y_i, y_j)}{\partial y_{i,k}} \tag{A.5}$$

respectively.

Then we have

$$\frac{\partial d_y(y_i, y_j)}{\partial d_y^2(y_i, y_j)} = 1/\frac{\partial d_y^2(y_i, y_j)}{\partial d_y(y_i, y_j)} = 1/(2d_y(y_i, y_j)) \tag{A.6}$$

and

$$d_y^2(y_i, y_j) = \|y_i y_j\|_2^2 = \sum_{k=1}^{p} (y_{i,k} - y_{j,k})^2 \tag{A.7}$$

That lead to:

$$\frac{\partial d_y^2(y_i, y_j)}{\partial y_{i,k})} = 2(y_{i,k} - y_{j,k}) \tag{A.8}$$

and

$$\frac{\partial d_y(y_i, y_j)}{\partial y_{i,k})} = \frac{y_{i,k} - y_{j,k}}{d_y(y_i, y_j)} \tag{A.9}$$

finally, the gradient is:

$$\frac{\partial c_4}{\partial y_{i,k}} = -\frac{2}{a} \sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(y_i, y_j)} (y_{i,k} - y_{j,k}) \tag{A.10}$$

Proof of 4.23.

Let's compute the second derivative now:

$$\frac{\partial^2 c_4}{\partial y_{i,k}^2} = \frac{\partial}{\partial y_{i,k}} \left(\frac{\partial c_4}{\partial y_{i,k}} \right) \tag{A.11}$$

where the gradient is given by A.10.Therefore:

$$\frac{\partial^2 c_4}{\partial y_{i,k}^2} = -\frac{2}{a} \sum_{i=1}^{n} \sum_{j=1, j<i}^{n} \frac{\partial}{\partial y_{i,k}} \left(\frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(y_i, y_j)} (y_{i,k} - y_{j,k}) \right) \tag{A.12}$$

The partial derivative is:

$$\begin{aligned} &\frac{\partial}{\partial y_{i,k}} \left(\frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(y_i, y_j)} (y_{i,k} - y_{j,k}) \right) \\ &\quad = (y_{i,k} - y_{j,k}) \frac{\partial}{\partial y_{i,k}} \left(\frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(y_i, y_j)} \right) \\ &\quad + \frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(y_i, y_j)} \underbrace{\frac{\partial}{\partial y_{i,k}} (y - i, k - y_{j,k})}_{=1} \end{aligned} \tag{A.13}$$

we note that:

$$\begin{aligned} &\frac{\partial}{\partial y_{i,k}} \left(\frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j) d_y(y_i, y_j)} \right) \\ &\quad = \frac{1}{d_x(x_i, x_j)} \frac{\partial}{\partial y_{i,k}} \left(\frac{d_x(x_i, x_j) - d_y(y_i, y_j)}{d_x(x_i, x_j)} \right) \\ &\quad = \frac{1}{d_x(x_i, x_j)} \frac{\partial}{\partial y_{i,k}} \left(\frac{d_x(x_i, x_j)}{d_y(y_i, y_j)} - 1 \right) \\ &\quad = \underbrace{\frac{d_x(x_i, x_j)}{d_x(x_i, x_j)}}_{=1} \frac{\partial}{\partial y_{i,k}} \left(\frac{1}{d_y(y_i, y_j)} \right) - \underbrace{\frac{\partial}{\partial y_{i,k}}}_{=0} \\ &\quad = \frac{-1}{d_y^2(y_i, y_j)} \frac{\partial}{\partial y_{i,k}} (d_y(x_i, x_j)) \\ &\quad = \frac{1}{d^2(x_i, x_j)} \frac{y_{i,k} - y_{j,k}}{d_y(y_{i,j})} \end{aligned} \tag{A.14}$$

Therefore:

$$\begin{aligned}&\frac{\partial}{\partial y_{i,k}}\left(\frac{d_x(x_i,x_j)-d_y(y_i,y_j)}{d_x(x_i,x_j)d_y(y_i,y_j)}(y_{i,k}-y_{j,k})\right)\\&=\frac{-(y_{i,k}-y_{j,k})^2}{d_y^3(y_i,y_j)}+\frac{d_x(x_i,x_j)-d_y(y_i,y_j)}{d_x(x_i,x_j)d_y(y_i,y_j)}\end{aligned} \tag{A.15}$$

$$\frac{\partial^2 c_4}{\partial y_{i,k}^2}=-\frac{2}{a}\sum_{i=1}^{n}\sum_{j=1,j<i}^{n}\left(\frac{d_x(x_i,x_j)-d_y(y_i,y_j)}{d_x(x_i,x_j)d_y(x_i,x_j)}-(y_{i,k}-y_{j,k})\right) \tag{A.16}$$

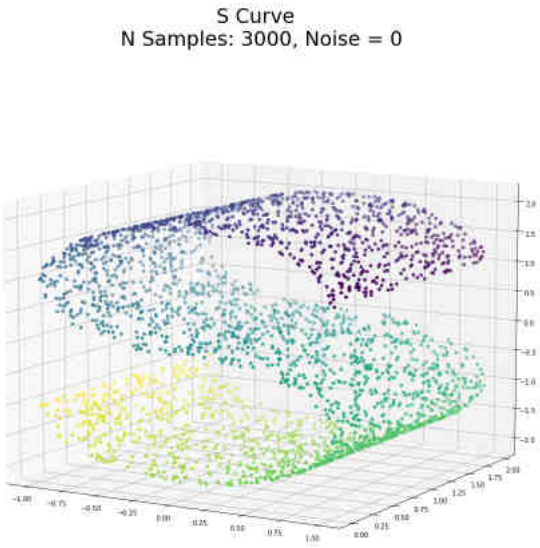
Appendix B

Synthetic Data

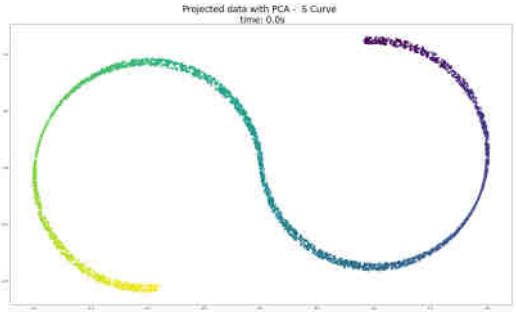
In this appendix we continue to present some application of dimensional reduction on synthetic data.The material in this appendix can be used as a supplement of Chapter 5.1.1.

B.1 S Curve

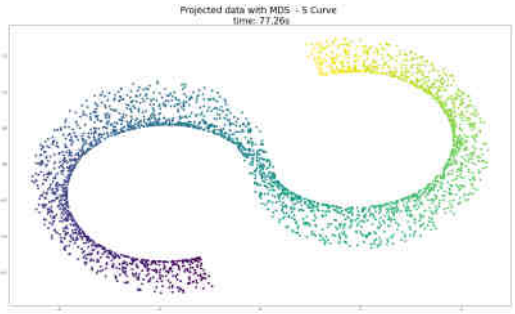
As in the previous case, S Curve is a convex ad expandable surface. The same considerations and the result made previously hold



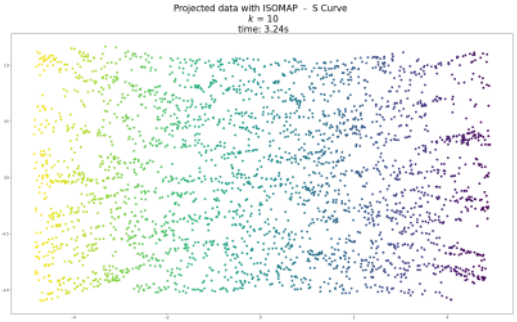
(a) S Curve



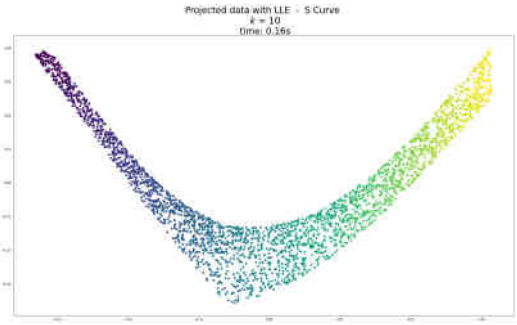
(b) PCA - 2d dimensional reduction



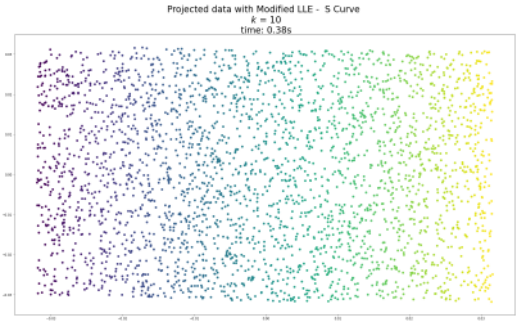
(c) MDS - 2d dimensional reduction



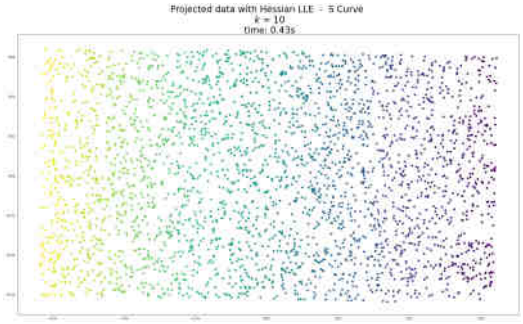
(d) Isomap - 2d dimensional reduction



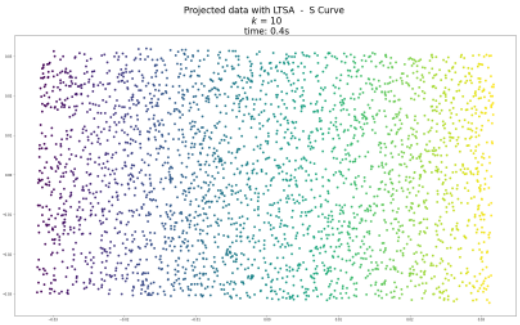
(e) LLE - 2d dimensional reduction



(f) MLLE - 2d dimensional reduction



(g) Hessian LLE - 2d dimensional reduction

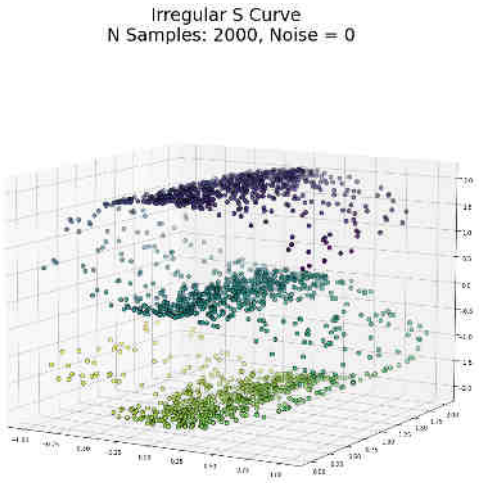


(h) LTSA- 2d dimensional reduction

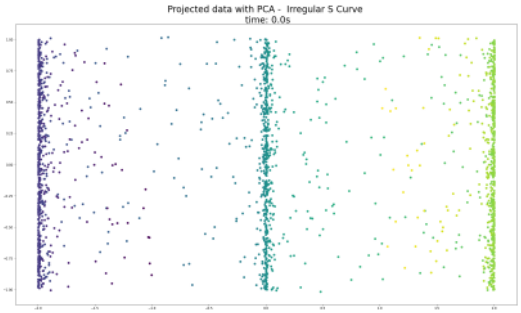
Figure B.1: 2 dimensional reduced S Curve data set. Linear techniques such as PCA and MDS are not able to correctly unfold the dataset. Global non linear method like Isomap are able to unfold and give a correct representation of low embedding data. LLE, a classic local non linear method suffers and has some issues to mapping correctly, issues that are overcomes by the other non vanilla local method.

B.2 Irregular S Curve

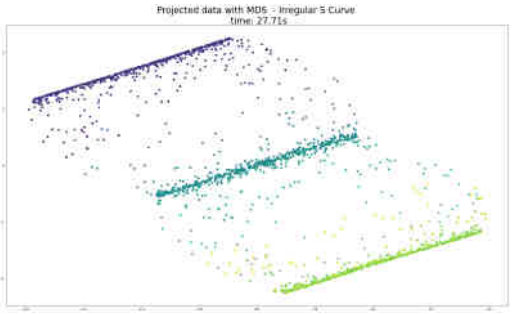
In this case the surface has an irregular distribution of the data over the curve. As we can see from the Figure 3.1, the PCA is able to represent the dataset almost well, instead method like LLE, fails to capture the geometric nature and therefore fails to represent it. Although there are scattered data, a kind of small holes, Isomap manages to represent embedding projection well, as well as LTSA, MLLE, HLLE.



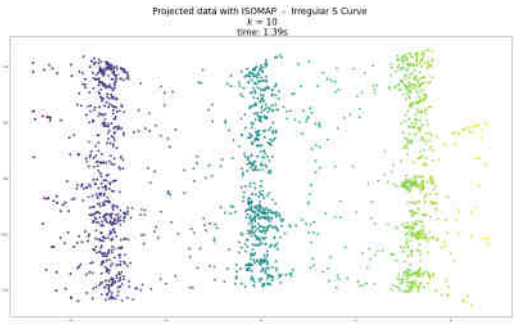
(a) Irregular S Curve



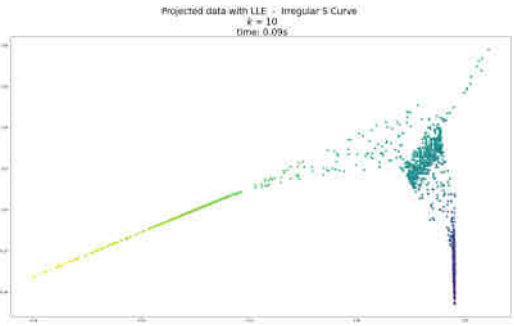
(b) PCA - 2d dimensional reduction



(c) MDS - 2d dimensional reduction



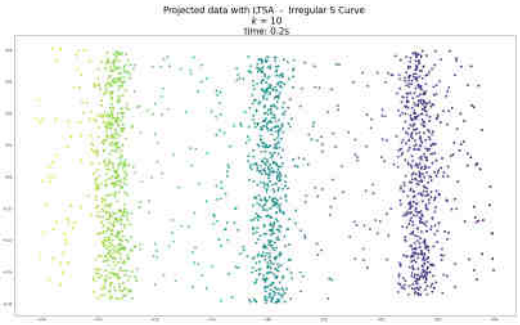
(d) Isomap - 2d dimensional reduction



(e) LLE - 2d dimensional reduction

(f) MLLE - 2d dimensional reduction

(g) Hessian LLE - 2d dimensional reduction



(h) LTSA- 2d dimensional reduction

Figure B.2: 2 dimensional reduced Irregular S Curve data set. Linear techniques such as PCA and MDS are not able to correctly unfold the dataset. Global non linear method like Isomap are able to unfold and give a correct representation of low embedding data. LLE, a classic local non linear method suffers and has some issues to mapping correctly, issues that are overcomes by the other non vanilla local method.

Appendix C

Code

```
############### RETRIEVE DATA ###############
f = open("Russel_3000_list.txt",'r')
russell = []
with open('Russel_3000_list.txt', 'r') as filehandle:
    for line in filehandle:
        currentPlace = line[:-1]
        russell.append(currentPlace)
start_1 = dt.datetime(2000,8,2)
end_1 = dt.datetime(2012,10,1)
start_2 = dt.datetime(2012,10,2)
end_2 = dt.datetime(2015,11,4)
start_3 = dt.datetime(2015,11,5)
end_3 = dt.datetime(2022,3,24)
############### DATA CLEANING ##############
def data_cleaning(df):
    new_df = np.log1p(df.pct_change())
    new_df = new_df.iloc[1:]
    new_df = new_df.dropna( axis='columns', how ='any')
    return new_df

###########FUNCTION FOR EIGENDECOMPOSITION AND DIMENSIONALITY
   ############
def svd(emb):
    u, s, vh = np.linalg.svd(emb, full_matrices=True)
```

```
24      cum_var = np.sum(s**2)
25      sigma_perc = s**2/cum_var
26      return sigma_perc
27
28  def dimensionality(sigma_perc):
29      i= 0
30      dimension = 0
31      while (dimension < 0.90):
32          dimension += sigma_perc[i]
33          i = i+1
34      return i,dimension
35
36  def dimensionality_1(sigma_perc):
37      i= 0
38      dimension = 0
39      for s in sigma_perc:
40          if sigma_perc[i] > 0.01:
41              dimension += 1
42          i = i+1
43      return i,dimension
44
45  ############MOVING WINDOW PCA,MDS,ISOMAP,LLE,HLLE#############
46
47  def rolling(df,window,i):
48      frame = df.shift(-1*(len(df)-window))
49      frame = frame.dropna()
50      frame = frame.shift(0)
51      frame = frame.dropna()
52      return frame
53
54  def roll_pca(df,window):
55      df = pd.DataFrame(df)
56      up_bound = int(df.shape[0])
57      shape = np.zeros(up_bound - window)
58      for i in range(0,up_bound - window):
59          temp_df = df.iloc[i:window +i]
```

```
60          st = ss().fit_transform(temp_df)
61          pca = PCA(0.90)
62          pc = pca.fit_transform(st)
63          s = svd(pc)
64          p = dimensionality(s)
65          shape[i] = pc.shape[1]
66      return shape
67
68  def roll_iso(df,window,k):
69      df = pd.DataFrame(df)
70      up_bound = int(df.shape[0])
71      shape = np.zeros(up_bound - window)
72      for i in range(0,up_bound - window):
73          temp_df = df.iloc[i:window +i]
74          iso = Isomap(n_neighbors=k,n_components=60,n_jobs=-1)
75          X_transformed = iso.fit_transform(temp_df)
76          s = svd(X_transformed)
77          p = dimensionality(s)
78          shape[i] = p[0]
79
80      return shape
81
82
83  def roll_mds(df,window):
84      df = pd.DataFrame(df)
85      up_bound = int(df.shape[0])
86      shape = np.zeros(up_bound - window)
87      for i in range(0,up_bound - window):
88          temp_df = df.iloc[i:window +i]
89          multids = MDS(metric=(True),n_components=60,n_jobs=-1)
90          X_transformed = multids.fit_transform(temp_df)
91          s = svd(X_transformed)
92          p = dimensionality(s)
93          shape[i] = p[0]
94      return shape
95
```

```
#Change method for 'lle', 'ltsa', 'hlle', 'mlle'
def roll_hlle(df,window,method='hlle'):
    df = pd.DataFrame(df)
    up_bound = int(df.shape[0])
    shape = np.zeros(up_bound - window)
    for i in range(0,up_bound - window):
        temp_df = df.iloc[i:window +i]
        embedding = manifold.LocallyLinearEmbedding(n_neighbors
   =60, n_components=60,method=method)
        X_transformed = embedding.fit_transform(temp_df)
        s = svd(X_transformed)
        p = dimensionality(s)
        shape[i] = p[0]
    return shape

###########DISTANCE CHECK############
dist_eucl = euclidean_distances()
geo = iso.dist_matrix_
delta = geo - dist_eucl
plt.figure(figsize=(20,20))
plt.imshow(dist_eucl, cmap='jet', interpolation='nearest')
plt.title('Euclidean Distance Matrix', fontsize=28)
plt.show()
plt.figure(figsize=(20,20))
plt.imshow(geo, cmap='jet', interpolation='nearest')
plt.title('Geodesic Distance Matrix', fontsize=28)
plt.show()
plt.figure(figsize=(20,20))
shw = plt.imshow(delta, cmap='jet', interpolation='nearest')
plt.colorbar(shw,fraction=0.046, pad=0.04)
plt.title('Difference Geodesic - Euclidean Distance Matrix',
   fontsize=28)
plt.show()

###########DISTANCE CHECK############
pca_dim_= roll_pca(df,60)
```

```
iso_dim = roll_iso(df,60)
mds_dim = roll_mds(df,60)
hlle_dim = roll_hlle(df,60,'hlle')
ltsa_dim = roll_hlle(df,60,'ltsa')
lle_dim = roll_hlle(df,60,'lle')
mlle_dim = roll_hlle(df,60,'mlle')

 ############PLOT#############
##prendere le date
date_1 = (temp_df.index.get_level_values('Date'))
data = date_1[61::]
plt.plot(data_2["Close"])
import matplotlib.dates as mdates
plt.figure(figsize=(35,20)) #(35,20)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%
    d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=300)
    )
plt.plot(data,sp_pca_dim, label = 'pca')
plt.plot(data,sp_iso_dim,label = 'iso')
PLT.plot(data,sp_mds_dim,label = 'mds')

plt.legend(loc = 'best')
plt.gcf().autofmt_xdate()
plt.title('#########ICE#########'  , fontsize=28)
plt.show()
##################################################

plt.figure(figsize=(25,20)) #(35,20)
fig, ax1 = plt.subplots(figsize=(25,20))
color = 'tab:red'
ax1.set_xlabel('Year/Month/Day')
ax1.set_ylabel('Russell 3000 Close', color=color)
ax1.plot(data_ru[1990:2180],label = 'russell3000', color='r',
    linestyle='dashed')
ax1.tick_params(axis='y', labelcolor=color)
```

```
ax1.legend(loc = 'lower right')

ax2 = ax1.twinx()  # instantiate a second axes that shares the
    same x-axis

color = 'tab:blue'
ax2.set_ylabel('Dimensionality', color=color)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%
    d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=20))
     # we already handled the x-label with ax1
plt.plot(data[1994:2200],pca_dim[1994:2200], label = 'pca')
plt.plot(data[1994:2200],iso_dim[1994:2200],label = 'iso')
plt.plot(data[1994:2200],mds_dim[1994:2200],label = 'mds')
ax2.tick_params(axis='y', labelcolor=color)
ax2.legend(loc = 'best')
plt.gcf().autofmt_xdate()
plt.title('Russell 3000 Intrinsic Dimension \n  2008 crisis: from
    01-07-2008 to 17-04-2009'  , fontsize=28)
fig.tight_layout()  # otherwise the right y-label is slightly
    clipped
plt.show()

#####################################################
#TOY MODEL
#####################################################
#S - Normal
# Style_1
X, y = make_s_curve(n_samples=3000, noise=0)
plt.figure(figsize=(28, 16))
ax = plt.axes(projection='3d')
ax.scatter3D(X[:, 0], X[:, 1], X[:, 2], c=y)
ax.view_init(10, -60)
ax.set_title('S Curve \n N Samples: 3000, Noise = 0.5', size=30)
# Irregular Swiss
with open(('./irregularS.npy'), 'rb') as f:
```

```
194     X = np.load(f)
195
196 with open(('./irregularS_exact_parametrization.npy'),
197           'rb'
198           ) as f:
199     y = np.load(f)
200 print('Input pointset shape: ', X.shape)
201 plt.figure(figsize=(28, 16))
202 ax = plt.axes(projection='3d')
203 ax.scatter3D(X[:, 0], X[:, 1], X[:, 2], c=y[:, 0],
204              cmap='viridis', s=50, edgecolors='k')
205 ax.view_init(10, -60)
206 ax.set_title('Irregular S Curve \n N Samples: 2000, Noise = 0',
    size=30)
207 a = "Irregular S Curve"
208 def residual_variance_pca(initial,emb):
209     dist_eucl = euclidean_distances(initial)
210     dist_emb  = euclidean_distances(emb)
211     residual = 1- np.corrcoef(dist_eucl,dist_emb)**2
212     return residual
213
214 plt.figure(figsize=(15,10))
215 plt.plot(k,error)
216 plt.plot(k,)
217 res = residual_variance_pca(X,X_pca)
218 # Swiss Roll - scattered
219 X, y = make_swiss_roll(n_samples=3000, noise=0)
220 plt.figure(figsize=(28, 16))
221 ax = plt.axes(projection='3d')
222 ax.scatter3D(X[:, 0], X[:, 1], X[:, 2], c=y)
223 ax.view_init(10, -60)
224 ax.set_title('Swiss Roll \n N Samples: 3000, Noise = 1', size=30)
225 # PCA
226 t = time.time()
227 X_pca = PCA(n_components=2).fit_transform(X)
228 pca_time = round(time.time()-t, 4)
```

```
229 plt.figure(figsize=(28, 16))
230 plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
231 plt.title('Projected data with PCA -  {} \n time: {}s'.format(a,
232           pca_time), fontsize=28)
233 plt.figure(figsize=(28, 16))
234 plt.title('Projected data with PCA -  {} \n time: {}s'.format(a,
235           pca_time), fontsize=28)
236 plot_nearest_neighbors_graph(X_pca, y, 10)
237 plt.show()
238
239 # MDS
240 t = time.time()
241 multids = MDS(metric=(True), n_components=2, n_jobs=-1)
242 X_transformed = multids.fit_transform(X)
243 mds_time = round(time.time()-t, 2)
244 plt.figure(figsize=(28, 16))
245 plt.scatter(X_transformed[:, 0], X_transformed[:, 1],
246             c=y[:, 0], cmap='viridis')
247 plt.title('Projected data with MDS  - {} \n time: {}s'.format(a,
248           mds_time), fontsize=28)
249 plt.figure(figsize=(28, 16))
250 plt.title('Projected data with MDS  -  {} \n time: {}s'.format(a,
251           mds_time), fontsize=28)
252 plot_nearest_neighbors_graph(X_transformed, y, 10)
253 plt.show()
254 # ISOMAP
255 er =[]
256 for i in range(3,100):
257     iso = manifold.Isomap(n_neighbors=i, n_components=2,
258                           neighbors_algorithm="auto", metric="
    euclidean", n_jobs=-1)
259     X_iso = iso.fit_transform(X)
260     er.append(iso.reconstruction_error())
261     #er[i]=iso.reconstruction_error()
262
263 plt.figure(figsize=(15, 8))
```

```
plt.plot(er)
plt.title('Reconstruction Error ISOMAP - Swiss Roll', fontsize=28)
plt.ylabel('Reconstruction Error', fontsize=15)
plt.xlabel('k', fontsize=15)
plt.figure(figsize=(28, 16))
t = time.time()
iso = manifold.Isomap(n_neighbors=14, n_components=2,
                      neighbors_algorithm="auto", metric="euclidean", n_jobs=-1)
X_iso = iso.fit_transform(X)
er_14= iso.reconstruction_error()
isomap_time = round(time.time()-t, 2)
plt.scatter(X_iso[:, 0], X_iso[:, 1], c=y, cmap='viridis')
plt.title('Projected data with ISOMAP  -  {} \n $k$ = 10 \n time: {}s'.format(a,
          isomap_time), fontsize=28)
plt.figure(figsize=(28, 16))
plt.title('Projected data with ISOMAP  -  {} \n $k$ = 10 \n time: {}s'.format(a,
          isomap_time), fontsize=28)
plot_nearest_neighbors_graph(X_iso, y, 10)
plt.show()
#################KNN - vs Reconstruction error################
error = np.zeros(20)
k = []
def iso():
    while i<10:
        i = 6
        iso = manifold.Isomap(n_neighbors=i, n_components=2, neighbors_algorithm="auto", metric="euclidean", n_jobs=-1)
        X_iso = iso.fit_transform(X)
        error[i-6] = iso.reconstruction_error()
        i = i+1
    return error

```

```
295 # LLE
296 t = time.time()
297 embedding = manifold.LocallyLinearEmbedding(n_neighbors=10,
        n_components=2)
298 X_transformed = embedding.fit_transform(X)
299 lle_time = round(time.time()-t, 2)
300 plt.figure(figsize=(28, 16))
301 plt.scatter(X_transformed[:, 0], X_transformed[:, 1],
302             c=y[:, 0], cmap='viridis')
303 plt.title('Projected data with LLE  -  {} \n $k$ = 10 \n time: {}
        s'.format(a,
304           lle_time), fontsize=28)
305 plt.figure(figsize=(28, 16))
306 plt.title('Projected data with LEE  -  {} \n $k$ = 10 \n time: {}
        s'.format(a,
307           lle_time), fontsize=28)
308 plot_nearest_neighbors_graph(X_transformed, y, 10)
309 plt.show()
310 # HESSIAN
311 t = time.time()
312 embedding = manifold.LocallyLinearEmbedding(
313     n_neighbors=10, n_components=2, method='hessian')
314 X_transformed = embedding.fit_transform(X)
315 hlle_time = round(time.time()-t, 2)
316 plt.figure(figsize=(28, 16))
317 plt.scatter(X_transformed[:, 0], X_transformed[:, 1],
318             c=y[:, 0], cmap='viridis')
319 plt.title('Projected data with Hessian LLE  -  {} \n $k$ = 10 \n
        time: {}s'.format(a,
320           hlle_time), fontsize=28)
321 plt.figure(figsize=(28, 16))
322 plt.title('Projected data with Hessian LEE  -  {} \n $k$ = 10 \n
        time: {}s'.format(a,
323           hlle_time), fontsize=28)
324 plot_nearest_neighbors_graph(X_transformed, y, 10)
325 plt.show()
```

```
# LTSA
t = time.time()
embedding = manifold.LocallyLinearEmbedding(
    n_neighbors=10, n_components=2, method='ltsa')
X_transformed = embedding.fit_transform(X)
ltsa_time = round(time.time()-t, 2)
plt.figure(figsize=(28, 16))
plt.scatter(X_transformed[:, 0], X_transformed[:, 1],
            c=y[:, 0], cmap='viridis')
plt.title('Projected data with LTSA  -  {} \n $k$ = 10 \n time:
    {}s'.format(a,
           ltsa_time), fontsize=28)
plt.figure(figsize=(28, 16))
plt.title('Projected data with LTSA -  {} \n $k$ = 10 \n time: {}
    s'.format(a,
           ltsa_time), fontsize=28)
plot_nearest_neighbors_graph(X_transformed, y, 10)
plt.show()
# Modified
t = time.time()
embedding = manifold.LocallyLinearEmbedding(
    n_neighbors=10, n_components=2, method='modified')
X_transformed = embedding.fit_transform(X)
mlle_time = round(time.time()-t, 2)
plt.figure(figsize=(28, 16))
plt.scatter(X_transformed[:, 0], X_transformed[:, 1],
            c=y[:, 0], cmap='viridis')
plt.title('Projected data with Modified LLE -  {} \n $k$ = 10 \n
    time: {}s'.format(a,
           mlle_time), fontsize=28)
plt.figure(figsize=(28, 16))
plt.title('Projected data with Modified LLE -  {} \n $k$ = 10 \n
    time: {}s'.format(a,
           mlle_time), fontsize=28)
plot_nearest_neighbors_graph(X_transformed, y, 10)
plt.show()
```

```
358 ################################################
359 plt.figure(figsize=(18, 10))
360 plot_nearest_neighbors_graph(X_iso, y, 10)
361 # Swiss Roll - Hole
362 X, y = make_swiss_roll_with_hole(n_samples=3000, noise=0)
363 plt.figure(figsize=(28, 16))
364 ax = plt.axes(projection='3d')
365 ax.scatter3D(X[:, 0], X[:, 1], X[:, 2], c=y)
366 ax.view_init(10, -60)
367 ax.set_title('Swiss Roll', size=30)
368 ax.set_title('Swiss Roll with Hole \n N Samples: 3000, Noise = 1'
    , size=30)
369
370 # Swiss Roll - Continuous
371
372 # Sphere
373 X, y = make_sphere_dataset(n_samples=3000, severed_poles=True)
374 plt.figure(figsize=(28, 16))
375 ax = plt.axes(projection='3d')
376 ax.scatter3D(X[:, 0], X[:, 1], X[:, 2], c=y)
377 ax.view_init(10, 20)
378 ax.set_title('Sphere with Severed Poles \n N Samples: 3000, Noise
     = 0', size=30)
379 ########
380 X_pca = PCA(n_components=2).fit_transform(X)
381 plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
382
383 iso = manifold.Isomap(n_neighbors=25, n_components=2,
384                       neighbors_algorithm="auto", metric="
    euclidean")
385 X_iso = iso.fit_transform(X)
386 iso_3 = manifold.Isomap(n_neighbors=15, n_components=2)
387 X_iso_3 = iso_3.fit_transform(X)
388 plt.figure(figsize=(25, 20))
389 plt.scatter(X_iso[:, 0], X_iso[:, 1], c=y)
390 plot_nearest_neighbors_graph(X_pca, y, 10)
```

```
391
392 #PLOT THE NN ON EMBEDDING SURFACE
393 def plot_nearest_neighbors_graph(X_iso, y, k):
394     nbrs = NearestNeighbors(n_neighbors=10, algorithm='ball_tree'
    ).fit(X)
395     #e = nbrs.kneighbors_graph(X).toarray()
396     e = kneighbors_graph(X_iso, k, mode='distance', include_self=
    False)
397     g = nx.Graph(e)
398     del e
399     pos = {n: location[:2] for n, location in enumerate(X_iso)}
400
401     nx.draw(g, pos=pos, node_size=60, alpha=.5, node_color=y,
402             width=0.8, with_labels=False)
403     ax.set_title('Swiss Roll', size=100)
404     plt.show()
```

Listing C.1: Python Code

List of Figures

List of Tables

References

[Bahadur and Paffenroth, 2019] N. Bahadur and R. Paffenroth, (2019). Dimension Estimation Using Autoencoders, *Department of Mathematical Sciences, Worcester Polytechnic Institute.*

[Bahadur et Al., 2017] N. Bahadur, K. Gajamannage,R. Paffenroth (2017), Study of Russell 3000 Dimensionality Using non-linear Dimensionality Reduction Techniques, *Department of Mathematical Sciences, Worcester Polytechnic Institute.*

[Bengio et Al., 2006] Y. Bengio, O. Delalleau, N. Le Roux, J.F., Paiement, P. Vincent and M. Ouimet (2006), Spectral Dimensionality Reduction, *Feature Extraction, Foundations and Applications, Springer*, pp. 519–550.

[Bicego ad Loog, 2016] M. Bicego and M. Loog (2016), Weighted K-Nearest Neighbor Revisited *23rd International Conference on Pattern Recognition (ICPR)*, pp. 1642-1647.

[Biggs, 1993] N. Biggs, (1993). Algebraic Graph Theory, *Cambridge Press.*

[Block et Al., 2021] A. Block, Z. Jia, Y. Polyanskiy and A. Rakhlin, (2021). A New Estimator of Intrinsic Dimension, *arXiv:2106.04018v1 [stat.ML].*

[Borg and Groenen, 2005] I.Borg and P. Groenen (2005), Modern Multidimensional Scaling: Theory and Applications, *Springer Series in Statistic*, pp.

[Boyd et Al., 2004] S. Boyd, S.P. Boyd and L. Vanderberghe (2004) Convex optimization *Cambridge University press.*

[Bondy and Murty, 1982] J. Bondy and U. Murty, (1982). Graph Theory with Applications Springer, *North Holland*.

[Boothby, 1975] W.M. Boothby, (1975). An Introduction to Differentiable Manifolds and Riemannian Geometry, *Academic Press*.

[Bunte et Al., 2012] K. Bunte,M. Biehl and B. Hammer (2012), A general framework for dimensionality-reducing data visualization mapping, *Neural Computation, MIT Press*, vol. 24, no. 3, pp.771–804.

[Cailliez, 1983] F. Cailliez (1983), The analytical solution of the additive constant problem, *Psychometrika*, vol. 48, no. 2, pp. 305–308.

[Cayton, 2005] L. Cayton, (2005). Algorithms for Manifold Learning, *University Of California at San Diego Technology*, vol. 12, pp. 1-17.

[Cheng et Al., 2021] F. Cheng, R. J Hyndman and A. Panagiotelis, (2021). Manifold learning with approximate nearest neighbors, `arXiv:2103.11773v1`.

[Christofides et Al., 2016] C. Christofides,T.S. Eicher, and C. Papageorgiou, (2016). Did established early warning signals predict the 2008 crises? *European Economic Review*, vol.81, pp. 103–114.

[Chung, 1996] F.R. Chung, (1996). Spectral Graph Theory. *CBMS Regional Conference Series in Mathematics*, no. 9.

[Cormen et Al., 2009] T. Cormen, C.E. Leiserson, R.L Rivest and C. Stein, (2009), Introduction to algorithms - Third edition, *MIT press*.

[Cox and Cox, 2008] M. Cox and T.Cox (2008), Multidimensional scaling, *In Handbook of data visualization, Springer* pp. 315–347.

[De Angelis and Dias, 2014] L. De Angelis and J. G. Dias, (2014), Mining categorical sequences from data using a hybrid clustering method, *European Journal of Operational Research*, vol. 234, pp. 720–730.

[De Leeuw and Heiser, 1980] J. De Leeuw and W. Heiser (1980). Multidimensional scaling with restrictions on the configuration, *North-Holland Publishing Company - Multivariate Analysis*, vol. 5.

[Deng et Al., 2020] Y. Deng, Z.Liu, J. Korvink *Topology optimization on two-dimensional manifolds* Computer Methods in Applied Mechanics and Engineering, Volume 364,2020

[Dijkstra, 1959] E.W. Dijkstra (1959). A note on two problems in connexion with graphs *Numerische Mathematik*, vol. 1, pp. 269 – 271.

[Donoho and Grimes, 2003] D. Donoho and D. Grimes (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data, *Proceedings of the National Academy of Sciences - PNAS*, vol. 100, no.10, pp. 5591-5596.

[Dudani, 1976] S. Dudani (1976). The distance-weighted k-nearest-neighbor rule *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325-327.

[Falasca and Bracco, 2021] F. Falasca, and A. Bracco, (2021). Exploring the climate system through manifold learning, *arXiv preprint arXiv:2110.03614*.

[Fix and Hodges, 1951] E. Fix and J.L. Hodges, (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties, *USAF School of Aviation Medicine, Randolph Field*.

[Floyd, 1962] R. W. Floyd, (1962). Algorithm 97: Shortest Path, *Communications of the ACM*, Vol. 5, No. 6, pp. 345.

[Friesz Bernstein, 2016] T.L. Friesz and D. Bernstein, (2016). Elements of Graph Theory *Foundations of Network Optimization and Games*, pp. 75-120.

[FTSE Russell - An LSEG Business] `https://research.ftserussell.com/products/downloads/Russell-US-indexes.pdf?552`

[Ghojogh and Karray, 2019] B.Ghojogh and F.Karray (2019). Eigenvalue and generalized eigenvalue problems: Tutorial, *arXiv preprint arXiv:1903.11240*.

[Ghojogh et Al., 2018] B. Ghojogh, A. Ghodsi,F.Karray and M.Crowley (2018). Locally Linear Embedding and its Variants: Tutorial and Survey, *Computing Research Repository - CoRR*.

[Ghojogh et Al., 2020] B. Ghojogh, A.Ghodsi,F.Karray, M.Crowley (2020). Multidimensional Scaling, Sammon Mapping, and Isomap: Tutorial and Survey, *Computing Research Repository - CoRR*.

[Golub and Reinsch, (1970)] G. Golub and C. Reinsch, (1970). Singular value decomposition and least squares solutions, *Numerische Mathematik*.

[Granda, 2020] C.F. Granda (2020). Probability and Statistics for Data Science, *New York University - NYU*.

[Greene, 2012] W. H. Greene, (2012). Econometric Analysis - 7th edition *Prentice Hall*.

[Groenen and Nalbantov, 2008] P. Groenen, G. Nalbantov and J.C. Bioch, (2008). SVM-Maj: A majorization approach to linear support vector machines with different hinge errors *Advances in Data Analysis and Classification* vol. 2, pp. 17-43.

[Horn and Johnson, 1994] R. A. Horn and C. R. Johnson, (1994). Matrix Analysis: Second Edition, *Cambridge University Press*, 1994

[Huang and Kou, 2014] Y. Huang and G. Kou,(2014). A kernel entropy manifold learning approach for financial data analysis, *Decision Support System*, vol. 64, pp. 31–42.

[Huang et Al., 2016] Y. Huang, G. Kou, and Y. Peng, (2016). Nonlinear manifold learning early warnings in financial markets, *European Journal of Operational Research*, vol. 258, pp. 692–702.

[Ioffe and Szegedy, 2015] S. Ioffe and C. Szegedy, (2015).Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *Computing Research Repository - CoRR* `arXiv:1502.03167` 2015

[Izenman, 2012] A. J. Izenman, (2012). Introduction to manifold learning, *WIREs Computational Statistics*, vol. 4, no. 5, pp. 439-446

[Jolliffe, 2002] I.Jolliffe, (2002.)Principal component analysis, *Springer Series in Statistics - Springer*.

[Kairanbay and Mat, 2013] M. Kairanbay and J.H. Mat, (2013). A Review And Evaluations Of Shortest Path Algorithms, *Int.J. of Sci. & Tech.Res* vol. 2, no. 6 pp.96-99.

[Karam and Campbell, 2013] Z. Karam and W. Campbell, (2013). Graph embedding for speaker recognition, *Graph Embedding for Pattern Analysis - Springer*, pp.229-260.

[Kumar ad Schneider, 2017] K. Kumar and J. Schneider, (2017). *Linear and Multilinear Algebra*, vol. 65, no. 11, pp. 2212-2244.

[Lee and Verleysen, 2007] J. Lee and M.Verleysen, (2007). Nonlinear dimensionality reduction, *Springer Science & Business Media*.

[Leeuw and Mair,2009] J. de Leeuw and P. Mair, (2009). Multidimensional Scaling Using Majorization:SMACOF in R, *Journal of Statistical Software*, vol. 31 no.3, pp. 1-30.

[Marsden, 2013] A. Marsden, (2013). Eigenvalues of the Laplacian and their relationship to the connectedness of a graph, *University of Chicago - REU*.

[Marsh and Cormier, 2001] L.C. Marsh and D.R. Cormier, (2001). Spline regression models, *SAGE Publications Ltd*, vol. 137.

[Marsland, 2009] S. Marsland, (2009). Machine Learning: An Algorithmic Perspective - Second Edition, *Chapman and Hall - CRC*.

[Mitchell, 1997] T.M. Mitchell, (1997). Machine Learning, *Mcgraw-Hill International*.

[Oldford, 2018] W. Oldford, (2018). Lecture: Recasting principal components. Lecture notes for Data Visualization, *Department of Statistics and Actuarial Science, University of Waterloo*.

[Pearson, 1901] K. Pearson, (1951), On lines and planes of closest fit to systems of points in space, *Philosophical Magazine* vol. 2, no.6, pp. 559 – 572.

[Picci, 2020] G. Picci, (2020). Statistical Methods for Dynamical Data Analysis Department of information Engineering, *University of Padova*.

[Platt, 2015] J. Platt, (2005). Fastmap, metricmap, and landmark mds are all Nystrom algorithms, *Microsoft Research - Technical Report*, vol. 26.

[Robbin and Salamon, 2021] J.W. Robbin and D.A. Salamon, (2021).Introduction to Differential Geometry, *ETH Zurich*

[Roweis and Saul, 2000] S.T. Roweis and L. K. and Saul, (2000). Nonlinear dimensionality reduction by locally linear embedding, *Science*, vol.290, no. 5500, pp. 2323–2326.

[Sammon, 1969] J.W. Sammon, (1969). A nonlinear mapping for data structure analysis, *IEEE Transactions on computers*, vol. 100, no. 5, pp. 401–409.

[Schwartz,2002] M. Balasubramanian and E. L. Schwartz, (2002). *The isomap algorithm and topological stability* Science, vol. 295, no. 5552, pp. 1–7.

[Shlens, 2003] J. Shlens, (2003). A TUTORIAL ON PRINCIPAL COMPONENT ANALYSIS: Derivation, Discussion and Singular Value Decomposition.

[Tenenbaum et Al., 2000] J. Tenenbaum,V. De Silva and J.C. Langford, (2000). A global geometric framework for nonlinear dimensionality reduction, *Science*, vol. 290, no. 5500, pp. 2319–2323.

[Torgerson, 1952] W.S. Torgerson, (1952). Multidimensional scaling: theory and method, *Psychometrika*, vol. 17, pp. 401–419.

[Tu, 2010] L.W. Tu, (2010). An Introduction to Manifolds, *Springer*.

[Van der Maaten et Al., 2009] L. Van der Maaten, E. Postma, J. Van den Herik, (2009). Dimensionality Reduction: A Comparative Review, *Tilburg University - Machine Learning Res*, vol. 10, pp. 66–71.

[Wang, 2012] J.Wang, (2012). Geometric Structure of High-Dimensional Data and Dimensionality Reduction, *Springer*, pp. 1-24.

[Wang, 2012] J.Wang, (2012). Geometric Structure of High-Dimensional Data and Dimensionality Reduction, *Springer*.

[Wilson, 1996] R.J. Wilson, (1996). Introduction to Graph Teory, *Prentice Hall Fourth edition*.

[Witten et Al., 2011] I. Witten,E. Frank, M. Hall and C. Pal, (2011). Data Mining: Practical Machine Learning Tools and Techniques *Morgan Kaufmann Series in Data Management Systems*.

[Wu and Chan, 2004] Y. Wu ad K.L. Chan, (2004). An extended Isomap algorithm for learning multi-class manifold, *International Conference on Machine Learning and Cybernetics*, vol. 6, pp. 3429–3433.

[Yang, 2002] M.-H. Yang, (2002). Extended isomap for pattern classification, *AAAI/IAAI*, pp. 224–229.

[Young, 1985] F.W. Young, (1985). Multidimensional scaling, *Encyclopedia of Statistical Sciences* vol. 5.

[Zhang and Wang, 2006] Z. Zhang and J.Wang, (2006). MLLE: Modified Locally Linear Embedding Using Multiple Weights, *Advances in Neural Information Processing Systems*, vol. 19.

[Zhang and Zha, 2004] Z.Y. Zhang, H.Y. Zha, (2004). Principal manifolds and nonlinear dimensionality reduction via tangent space alignment *SIAM Journal on Scientific Computing.* vol. 26, no.1, pp. 313 – 338.