Manifold Learning: Comparison of Dimensionality Reduction Methods appropriate for non-linear data

Jazmin Inés Teng, *Student, TUM*

Abstract

Reducing the dimensionality of the feature space is essential for many application areas, such as computer vision, bioinformatics, neuroinformatics, climate predictions, among others. This is because its application not only allows us to optimize the learning process but also to reduce the complexity of the model. At the same time, it is widely used in Exploratory Data Analysis to visualize data from a high dimensionality to a lower dimensionality which is more familiar to the human eye. The most common techniques are those of linear projections in which it seeks to project data from one dimensional space to another. However, real world data tends to be non-linear and applying this technique can get results that are not optimal. This is why manifold learning, which focuses on nonlinear data, has gained popularity in recent years and has supported the community in solving problems with high-dimensional data. In this paper we will introduce the motivation of dimensionality reduction. At the same time, we will provide an overview of different well-known techniques and talk about the differences between linear and nonlinear dimensionality reduction methods, highlighting their strengths and the challenges they face, and finally comparing their performance on a nonlinear dataset.

Index Terms

Manifold Learning, Dimensionality Reduction, PCA, ISOMAP, LLE, Diffusion maps, MDS.

I. Introduction

NOWADAYS, given the technological advances, big data and improvements in computational capacities, Machine Learning techniques are being massively applied in different fields in order to solve problems that were previously considered impossible. Advances also bring challenges, one of the problems studied in recent times is dedicated to the feature space dimensionality reduction of the datasets used to feed machine learning models. With dimensionality reduction we refer to the process of reducing the number of attributes in a dataset while keeping as much of the variation in the original dataset as possible. In other words, convert the data from a high-dimensional space into a low-dimensional space.

The reason we seek to reduce dimensionality is because usually it is computationally expensive to work with the full dimensionality of the data taken from observations. In the majority of cases, data with a complex feature set will slow down the training of the model and it also complicates the search of the global optimum, such as the case of the classifiers that have low performance when it is trained with few training samples, in which each of them is located in a high-dimensional space. Reducing the dimensionality of the data will train the model faster and with fewer resources.

On the other hand, we try to reduce the dimensionality in order to keep only the most significant attributes for the model. Removing noisy data improves the accuracy of the model and also avoids the problem of overfitting. One of the strengths that it brings us is also its ability to transform non-linear data into a linearly separable form, which in turn allows us to visualize high-dimensional data into something that is humanly understandable.

Within dimensionality reduction, you can find a variety of techniques that emerged to solve situations that were not covered. This is also the case of Manifold Learning, which was born with the purpose of dealing with non-linear data, that is how real life data tends to be, that linear methods did not generate good performance. Therefore, to evaluate the performance of the different dimensionality reduction techniques and to prove the superiority of manifold learning techniques over linear techniques in non-linear data, we will apply the techniques on the swiss roll dataset, which consists of observations of three variables that represent three-dimensional coordinates in the shape of a swiss roll dessert and we aim to unroll the swiss roll and represent it in a two-dimensional subspace while maintaining the local geometry of nearby data points.

In this paper, we discuss different manifold learning techniques that work on non linear high-dimensional data structure. In Section II, we provide the background of the linear dimensionality reduction with the description of some well known techniques. In Section III, we introduce different manifold learning methods that are widely used these days and we detail their strengths and challenges. In Section IV, we focus on the differences presented in the techniques mentioned in previous sections. Finally, we make a brief conclusion in Section V.

S. Reiz was with the Department of Informatics, Technical University of Munich, Germany e-mail: reiz@in.tum.de.
Manuscript received Dec 12, 2022; revised Jan 11th, 2023.

II. Linear Dimensionality Reduction Methods

Firstly, we describe the different linear dimensionality reduction methods in the present section. The linear dimensionality reduction methods aim to project the data from a high to a low dimensional space through linear transformations. That is to say, they reduce the dimensionality of the data through a linear function. The most common methods are Principal Component Analysis (PCA) [1], Linear Discriminant Analysis (LDA) [2] and Metric Multidimensional Scaling (Metric MDS).

Regarding the linearity of the data, it means that the data lies in a affine space and that is a geometric structure that generalizes some of the properties of Euclidean spaces keeping only the properties related to parallelism. Thus, linear dimensionality reduction is appropriate for data that is on or close to a linear manifold, and on the other hand, manifold learning focuses on data on a non-linear manifold that refers to a manifold that is not an affine space.

A. PCA

Principal Component Analysis (PCA), introduced by Karl Pearson in 1901 [1] and named by Harold Hotelling in 1933 [3] is a powerfull dimensionality reduction technique that is widely used in several areas. The main idea of PCA is to find a linear mapping between a high dimensional space D and a subspace d that captures most of the variability in the data. With subspace or vector subspace we refer to the vector space that is contained in another vector space and contains the same properties of the vector space in which it is contained. In this case, the subspace is specified by d orthogonal vectors called the principal components and it indicates the directions of maximum variance of the data, in which the information is more dispersed. Therefore, the PCA mapping is a projection into that space [4] and we can say that the principal components are the underlying structure of the data.

For a better understanding of how PCA works, we introduce some mathematical concepts. An eigenvector is a vector that does not change its direction when you apply a linear transformation. Each eigenvector is associated with a corresponding eigenvalue, which is used to transform an eigenvector and indicates the variation of the data and how the data is spread out in that direction. The eigenvector whose direction shows the maximum variance of the dataset (highest eigenvalue) is called the first eigenvector or in other words the principal component.

According to Lawrence K. Saul [5], the PCA can be interpreted as the calculation of the linear projections of greatest variance from the top eigenvectors of the data covariance matrix. As we know, the covariance matrix defines the spread of the data as the variance and the orientation as the covariance.Therefore, we can find a vector that indicates the direction of the greatest dispersion of the data and whose magnitude is equal to the dispersion in this direction. This vector is called the top or first eigenvector and that magnitude, which represents the importance of the direction, is the corresponding eigenvalue.

1) Algorithm:

The algorithm of the PCA is described below [6].

Algorithm 1 PCA Algorithm

1: Standardize the D-dimensional dataset X.
2: Construct the covariance matrix.
3: Decompose the covariance matrix into its eigenvectors and eigenvalues.
4: Select d eigenvectors which correspond to the d largest eigenvalues. (d is the dimension of the new feature subspace).
5: Construct a projection matrix W from the top d eigenvectors.
6: Transform the D-dimensional input dataset X using the projection matrix W to obtain the new d-dimensional feature subspace.

B. Metric MDS

Multidimensional Scaling (MDS) is one of the eigenvector methods designed to model linear variabilities in high dimensional data and it is also used to create mapping of items based on distance. MDS arranges the low-dimensional points to minimize the discrepancy between the pairwise distances in the original space and the pairwise distances in the low-D space. That is, it computes the low dimensional embedding that best preserves pairwise distances between data points.

The classical MDS (cMDS) preserves the original distance metric between points, that's why it belongs to the metric MDS category. If these distance correspond to Euclidean distances, the results of metric MDS are equivalent to PCA. Fot that reason, the well-known PCA can actually be considered as Metric MDS. Note that the mechanism of both are different but it is possible to obtain the same result. In contrast, nonmetric MDS is used to deal with non-numerical distances between items, in which there is no measurement error. Lastly, it is interesting to mention the existence of MDS for non-linear data such as the Sammon mapping [7] where small distances are emphasized.

In MDS with classical scaling (cMDS), the inputs are projected into the subspace that best preserves their pairwise squared distances $|\vec{X_i} - \vec{X_j}|$, while the outputs of MDS are computed from the top eigenvectors of the n x n Gram matrix B. Instead of use the matrix of input X, we use the pairwise dissimilarity matrix D in order to reconstruct X in a lower dimension keeping

the original distances. In cMDS, the distance matrix D is given by the euclidean pairwise distances between data points and the goal is to find the matrix X, in which the distances between data points in the dimension d is similar to the distances in the original dimension. The entries of the gram matrix $B = X.X^t$ are given by the inner products of the rows of X:

$$b_{ij} =< x_i, x_j >= \sum_{k=1,K} x_{ik}.x_{jk}$$

$$d_{ij}^2 = \sum_{k=1,K} (x_{ik}.x_{jk})^2 = \sum_{k=1,K} (x_{ik}.x_{ik}) + (x_{jk}.x_{jk}) - 2(x_{ik}.x_{ik})$$

$$= \sum_{k=1,K} x_{ik}.x_{ik} + \sum_{k=1,K} x_{jk}.x_{jk} - 2 \sum_{k=1,K} x_{ik}.x_{jk} = b_{ii} + b_{jj} - 2b_{ij}$$

We can obtain the entries of B from the matrix D and once we obtain B we can find the X in dimension d preserving the original distances. The algorithm of the cMDS is described below [8].

1) Algorithm:

Algorithm 2 classical MDS Algorithm

1: Set up the squared proximity matrix $D^2 = [d_{ij}^2]$
2: Apply the double centering: $B = -\frac{1}{2}CD^2C$ using the centering matrix $C = I - \frac{1}{n}J_n$, where n is the number of objects, I is the nxn identity matrix and J_n is an nxn matrix of all ones.
3: Extract the m largest positive eigenvalues $\lambda_1 \dots \lambda_m$ of B and the corresponding m eigenvectors $e_1 \dots e_m$.
4: A m-dimensional spacial configuration of the n objects is derived from the coordinate matrix $X = E_m \Lambda_m^{1/2}$, where E_m is the matrix of m eigenvectors and Λ_m is the diagonal matrix of m eigenvalues of B, respectively.

III. Manifold Learning: Non-linear Dimensionality Reduction Methods

In this section, we introduce some background regarding manifold learning and describe the different non-linear dimensionality reduction methods. In most of the cases, the data obtained from the observation of the real world tends to be non-linear and processing it with linear methods does not bring good results. This is because they fail to detect the intrinsic geometric dimensionality of non-linear data. Often, those observed high dimensional data are the consequences of a small number of factors. In this case, it is reasonable to assume the high-dimensional data lie approximately on a manifold.

As a consequence, manifold learning describes non-linear datasets as low-dimensional manifolds embedded in high-dimensional spaces. The assumption is that all nonlinear data resides in a high-dimensional space but can actually be represented by a low-dimensional manifold. Therefore, when we place the manifold inside the higher dimensional spaces, we have an embedding of a manifold in a higher-dimensional Euclidean space. In order to better understand this concept, it would be better to start with some important definitions.

1) Manifold:

A d-dimensional manifold M is a set that is locally homeomorphic with R^d. That is, for each $x \in M$, there is an open neighborhood around x called N_x, and a homeomorphism $f : N_x \to R^d$. These neighborhoods are referred to as coordinate patches, and the map is referred to a coordinate chart. The image of the coordinate charts is referred to as the parameter space.

2) Geodesic distance:

The geodesic represents a straight line in curved space or the shortest curve along the geometric structured defined by our data points. The geodesic between two points is defined as the shortest curve on the manifold connecting the two points. The length of the geodesic is the geodesic distance. Due to the complexity of non-linear data, the distance between the data can only be reflected with the geodesic distance and not the Euclidean distance. In other words, the true low-dimensional geometry of the manifold is reflected in the geodesic distances. Linear methods often fail to obtain the real distance since they only see the Euclidean structure. For that reason, non-linear methods tend to calculate the geodesic distance of the data structure or using local euclidean distance to approximate it.

A. ISOMAP

In this subsection, we introduce ISOMAP as non-linear dimensionality reduction methods and its working mechanism. ISOMAP (isometric mapping) introduced by Tenenbaum [9] is a manifold learning method based on the spectral theory [10], which tries to preserve the geodesic distances in the lower dimension by creating a neighborhood network and uses its graph distance to approximate the geodesic distance between all pairs of point.

As mentioned above, non linear data are reflected in geodesic distance. For this reason, the principle of ISOMAP is to use the graph distance between two point to approximate the geodesic distances between those two points. This is made by creating a similarity matrix for eigenvalue decomposition and through that, it can find the low dimensional embedding of the dataset.

Unlike other non-linear dimensionality reduction like LLE (Subsection B) which only uses local information, ISOMAP uses the local information to create a global similarity matrix and its algorithm uses euclidean metrics to prepare the neighborhood graph. Then, it approximates the geodesic distance between two points by measuring shortest path between these points using graph distance. Thus, it approximates both global as well as the local structure of the dataset in the low dimensional embedding.

ISOMAP is base on MDS but, unlike MDS, it preserves geodesic distances between data points instead of Euclidean distance. ISOMAP seeks a mapping so that the geodesic distance between data points match the corresponding Euclidean distance in the transformed space in order to preserve the true geometric structure of the data. To approximate the geodesic distance between points without knowing the geometric structure of the data we can assume that, in a small neighbourhood, the Euclidean distance is a good approximation for the geodesic distance. So, for points further apart, the geodesic distance is approximated as the sum of Euclidean distances along the shortest connecting path. Once the geodesic distance has been obtained, MDS is performed in order to recreate the map.

1) Algorithm:

Given a set of data points $y_1, \ldots, y_n$ in a D-dimensional space R^D, ISOMAP assumes that the data lie on a manifold and maps y_i to its d-dimensional representation x_i in such a way that the geodesic distance between y_i and y_j is as close to the euclidean distance between x_i and x_j in R^d as possible. The ISOMAP algorithm has three stages:

Algorithm 3 ISOMAP Algorithm

1: A neighborhood graph is constructed.
2: ISOMAP proceeds with the estimation of geodesic distances.
3: ISOMAP recovers x_i by using the classical MDS on the geodesic distances.

We start calculating the distance between all the points and then we construct the neighborhood graph, in which, the weight of the edges are the distance. The neighborhood graph can be constructed using KNN or ϵ-neighborhood. In the case of ϵ-neighborhood, two points are neighbors if the distance between those two points is less than the parameter ϵ. On the other side, with knn-neighborhood, by specifying the value of k, the points x and y are neighbors if y is one of the k nearest neighbors of x. After that, it estimates the geodesic distances by computing the shortest path between the corresponding vertices in the neighbourhood graph. The distance is stored in a matrix that is ultimately used as input to the cMDS to obtain the embedding.

B. LLE

Continuing with the nonlinear methods, we present a topology preservation manifold learning method called Locally Linear Embedding and with topology preservation we mean the neighborhood structure is intact. This implies that the points which are nearby in higher dimension should be close in lower dimension.

The Local Linear Embedding (LLE) introduced by Roweis and Saul in 2000 [5] is an unsupervised learning algorithm that looks for an embedding that preserves the local geometry in the neighborhood of each data point. In detail, LLE embeds data points in a low-dimensional space by finding the optimal linear reconstruction in a small neighborhood. Roweis and Saul [11] also said that the inspiration behind LLE is "Think globally fit locally". Therefore, we can say that LLE attempts to discover nonlinear structure in high dimensional data by exploiting the local symmetries of linear reconstructions.

1) Assumptions:

LLE makes some assumptions:

- Data is well sampled. That means the density of the dataset is high. In other words, for every point we have at least 2d points in its neighborhood.
- Dataset lies on a smooth manifold. That assume that the point and its neighbors lie on the locally linear manifold (Without sharp bends or holes)

The LLE algorithm is based on simple geometric intuitions. Suppose the data consist of N real-valued vector $\vec{X}i$, each of dimensionality D, sampled from some underlying manifold. Provided sufficient data, we expect each data point and its neighbors to lie on or close to a locally linear patch of the manifold and we can characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. In others words, we want to map data point, in this case $\vec{X}i$, from the original high dimension D to a lower dimension d with vectors $\vec{Y}i$ that preserve the structure of the data. In the simplest formulation of LLE, one identifies k nearest neighbors per data point, as measured by Euclidean distance. After that, we construct the weights W_{ij} that minimize the cost function $\xi(W)$ in where we measure the reconstruction error. The resulting weights are those that obtain the lowest cost function, that is, those that characterize the intrinsic geometric properties of each neighborhood and best reconstructs the data structure in both dimensions.

$$\xi(W) = \sum_i \left| \vec{X}i - \sum_j W_{ij} \vec{X}j \right|^2$$

The reconstruction cost function adds up the squared distances between all the data points and their reconstructions. While the weight W_{ij} summarizes the contribution of the j^{th} data point to the i^{th} reconstruction . To compute the weights W_{ij} we minimize the cost function subject to two constraints:

1) Each data point X_i is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if X_j does not belong to this set.
2) The rows of the weight matrix sum to one $\sum_j (W_{ij} = 1)$.

The optimal weight W_{ij} subject to these constraints are found by solving a least squares problem. With these constraints, the data points are invariant to rotations, rescalings and translation of that data point and its neighbors. And as consequence, the reconstruction weights characterize intrinsic geometric properties of each neighborhood that are invariant to those transformations. Finally, LLE constructs a neighborhood preserving mapping where each high dimensional observation $\vec{X}i$ is mapped to a low dimensional vector $\vec{Y}i$ representing global internal coordinated on the manifold. This is done by choosing d-dimensional coordinates $\vec{Y}i$ to minimize the embedding cost function $\Phi(Y)$ in where the vectors $\vec{Y}i$ with the lowest embedding cost function is those best reconstruct the coordinates in the lower dimensions with the weights W_{ij}. This cost function is based on locally linear reconstruction errors, but fixing the weights W_{ij} while optimizing the coordinates $\vec{Y}i$.

$$\Phi(Y) = \sum_i \left| \vec{Y}i - \sum_j W_{ij} \vec{Y}j \right|^2$$

2) Algorithm:

The algorithm has only one free parameter and that is the number of neighbors per data point, k. Once neighbors are chosen, the optimal weight W_{ij} and coordinates $\vec{Y}i$ are computed by standard methods in linear algebra. The algorithm involves a single pass through the three steps and finds global minima of the reconstruction and embedding costs in both equations. The algorithm takes as input the N high dimensional vectors, $\vec{X}i$.

Algorithm 4 Locally Linear Embedding Algorithm

1: Compute the neighbors of each data point, $\vec{X}i$
2: Compute the weights W_{ij} that best reconstruct each data point $\vec{X}i$ from its neighbors, minimizing the reconstruction cost function $\xi(W)$ by constrained linear fits.
3: Compute the vectors $\vec{Y}i$ best reconstructed by the weights W_{ij}, minimizing the embedding cost function $\Phi(Y)$

C. Diffusion Maps

Diffusion maps is a non-linear dimensionality reduction method introduced by Coifman and Lafon in 2006 [12] that maps coordinates between data and diffusion space in order to reorganize the data according to parameters of its underlying geometry called diffusion metric. The diffusion map preserves a data set's intrinsic geometry, and since the mapping measures distances on a lower-dimensional structure, we expect to find that fewer coordinates are needed to represent data points in the new space.

In order to achieve it, diffusion map computes a family of embeddings of a dataset into a lower dimension euclidean space whose coordinates can be computed from the eigenvectors and eigenvalues of a diffusion operator on the data. Note that the Euclidean distance between points in the embedded space is equal to the diffusion distance between probability distributions centered at those points and it is known that the calculation of diffusion distance is computationally expensive, therefore, we can map data points into a Euclidean space according to the diffusion metric and the diffusion distance in data space simply becomes the Euclidean distance in this new diffusion space. That is to say, diffusion map embeds data in a lower dimensional space, such that the Euclidean distance between points approximates the diffusion distance in the original feature space. As result, diffusion maps give a global description of the data set by integrating local similarities at different scales.

The connectivity of the data set, measured using a local similarity measure, is used to create a time-dependent diffusion process. As the diffusion progresses, it integrates local geometry to reveal geometric structures of the dataset at different scales. By defining the time-dependent diffusion metric, we can then measure the similarity between two points at a specific scale (or time), based on the revealed geometry. With the concept of random walks, diffusion map reveal the underlying geometric structure of the dataset so they integrate local feature to build a global picture.

1) Diffusion distance:

$D_t(x_i, x_j)$ is called diffusion distance after t time steps between points x_i and x_j.

$$D_t(x_i, x_j)^2 = \sum_u |p_t(x_i, x_u) - p_t(x_j, x_u)|^2 = \sum_u |P^t_{iu} - P^t_{ju}|^2$$

The diffusion distance between x_i and x_j points is zero if $x_i = x_j$ and in the case when x_i and x_j are in the same cluster $p_t(x_i, x_u)$ and $p_t(x_j, x_u)$ will always have similar magnitude. That is because if x_u, x_i and x_j are in the same cluster, both

probabilities will be high and otherwise both will be low. Therefore the probabilities always cancel out and the distance will be small. On the other hand, if x_i and x_j are in separate cluster then the probabilities will have different magnitudes. The therms in the sum are therefore all large in magnitude leading to a large diffusion distance.

By definition, the notion of proximity reflects the connectivity in the graph of the data. Indeed, $D_t(x, y)$ will be small if there is a large number of short paths connecting x and y, that is, if there is a large probability of transition from x to y and viceversa. In addition, as previously noted, t plays the role of a scale parameter [12].

2) Kernel function:

The kernel defines a local measure of similarity within a certain neighbourhood of the data points. This means that outside the neighbourhood, the function quickly goes to zero [4].

The diffusion kernel k satisfies two properties:

- k is symmetric: $k(x, y) = k(y, x)$.
- k is positivity preserving: $k(x, y) \geq 0$.

3) Random Walk:

The random walk process determines the probable location of a point subject to random motions, given the probabilities of moving some distance in some direction. The time-dependent diffusion process consist of a random walk on the dataset where each jump has a probability associated with it. When the diffusion process runs for a time t, we get different probabilities of various paths it can take to calculate the distance over the underlying geometric structure.

The connectivity between two data points x and y is defined as the probability of jumping from x to y in one step of the random walk and is $connectivity(x, y) = p(x, y)$ and it is useful to express this connectivity in terms of a non-normalised likelihood function k known as the diffusion kernel:

$$connectivity(x, y) \propto k(x, y).$$

If the diffusion kernel satisfies the second properties, then it is allowed to be interpreted as a scaled probability, so that $\frac{1}{dx} \sum k(x, y) = 1$. Therefore, with $\frac{1}{dx}$ as the normalisation constant, the relation between the kernel function and the connectivity is then:

$$connectivity(x, y) = p(x, y) = \frac{1}{dx} k(x, y)$$

4) Algorithm:

The diffusion map consist of the following steps described in the algorithm 5.

Algorithm 5 Diffusion Map Algorithm

Input High dimensional data set x_i, $i = 0...N - 1$

1: Define a kernel, $k(x, y)$ and create a kernel matrix, K, such that $K_{ij} = k(x_i, x_j)$.
2: Create the diffusion matrix by normalising the rows of the kernel matrix.
3: Calculate the eigenvectors of the diffusion matrix.
4: Map to the d-dimensional diffusion space at time t, using the d dominant eigenvectors and eigenvalues.

Output Lower dimensional data set Y_i, $i = 0..N - 1$

IV. Comparison

With the purpose of analysing the performance of each discussed methods under non-linear data, we will use the Swiss roll dataset, which is a 2-dimensional submanifold embedded in 3-dimensional space [Figure 1]. The Swiss roll dataset, created by Dinoj Surendran in 2004, is used widely as benchmark dataset for testing dimensionality reduction techniques and it consists of 2000 observations of an 3 variables which represent 3 dimensional coordinates. After applying the dimensionality reduction techniques, we expect an unrolled 2d representation of the Swiss roll dataset, maintaining the underlying structure of the original manifold like the variability of the data, global and local geometry, the neighbourhood or distance between data points. Furthermore, we will consider certain aspects such as the speed, computational complexity and their sensitivities to outliers, noises and parameter configurations.

Since the main objective is to unroll the Swiss roll, we start evaluating the geometry structure of the recovered embedding. With linear methods like PCA and MDS, we apply both scaled and unscaled data. As observed in Figure 2, with unscaled data, both methods fail to unroll the Swiss roll, while with scaled data, they are able to flatten the Swiss roll so we can visualize the form of a flatten roll but they cannot unroll it and thus they fail to preserve the geodesic distance between data points. Notice that with scaled data MDS gives a better result than PCA representing the depth or width of the roll.

In contrast, non linear methods are able to unroll the data set. In the case of ISOMAP, the resulting embedding is uniformly distributed in 2d space with the correct order of the data as displayed with the colors. Thus, we conclude that it succeeds in

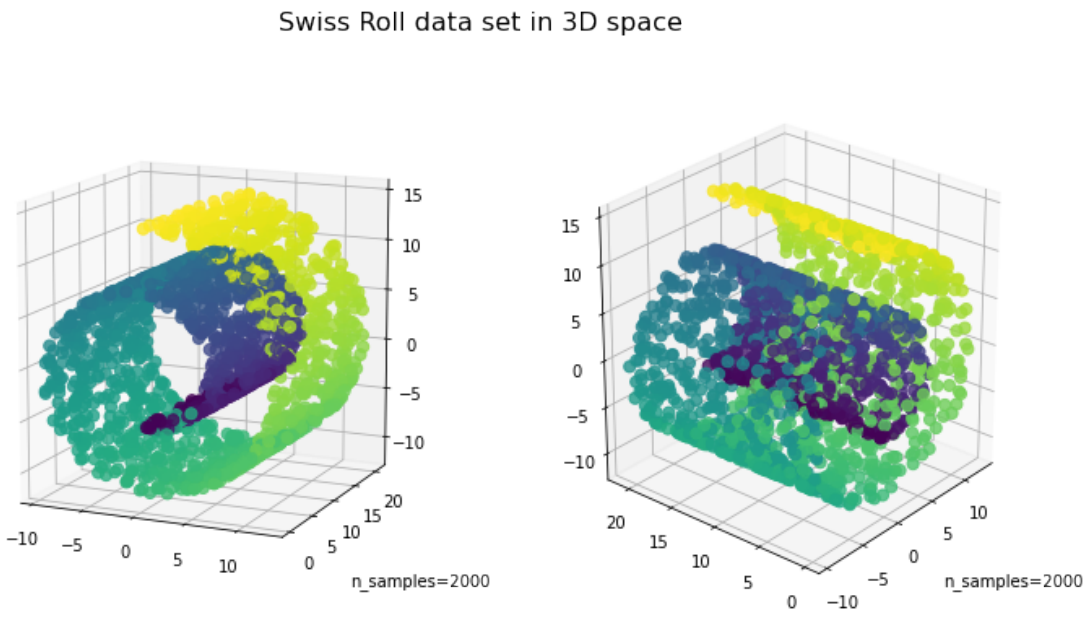

Fig. 1: 3D Representation of the Swiss Roll dataset.

preserving the global and local structure of the data, in addition to conserving the distance between the data points through the neighbourhood graph, which is constructed with local information. Meanwhile, LLE returns an embedding that preserves the local distance between points but fails to recover the global structure. This is because it belongs to conformal mappings and that means it only preserves local angles but not necessarily global distance and this can be easily observed with the deformation of the width of the roll, which is supposed to be uniform. Unlike the others methods, Diffusion Maps is able to recognize the global structures while maintaining local order. The Figure 2 demonstrates the embedding of the Swiss Roll using different sets of eigenvectors after applying Diffusion Map, starting with the one-dimensional embedding, which only use the eigenvector Ψ_1. The following uses the eigenvector Ψ_1 and $\Psi 5$ and the resulting embedding shows the unrolled Swiss Roll in its own way and the last one, which use the eigenvector Ψ_1 and Ψ_8, also recover the height of the data in the original dimension.

As mentioned in previous section, cMDS can obtain the same result as PCA and both depend on the same eigenvalues, but they use eigenvectors to construct different matrix. It is important to remark that MDS is much slower as PCA but its advantages lies in the fact that MDS only needs to know the pairwise distance matrix D and with PCA we also need to know the matrix X. Nonlinear methods are sensitive to their parameters, that means a small change in the configuration of these hyper parameters can drastically affect the results. ISOMAP and LLE are sensible to the parameter K since they use the KNN algorithm, whereas the performance of Diffusion Map depends on their parameters, for example, it is highly sensible to the parameter sigma which represents the width of the Gaussian kernel.

Respecting to the speed and the computational complexity which are highly related, the slower techniques are MDS and ISOMAP and this is because they have dense matrix to compute the spectral decomposition. Particularly, ISOMAP is slower than MDS and the reason is because it is based on MDS but it needs to precompute the neighborhood graph to get the geodesic distance matrix, which is the input of MDS. In contrast, PCA, LLE and Diffusion map are considered as fast methods. LLE use simple linear algebra operations and it is incremental, the W matrix is very sparse therefore the spectral decomposition can be performed much quickly. Unlike LLE, PCA compute the eigenvalue decomposition of its covariance matrix, thus it is more expensive than LLE. Diffusion map algorithm is simple and efficiently computable, because it involves one sparse eigenvalue problem and few local computations. However the distance computations in search of neighbouring points is a seemingly difficult task depending on the dimensionality of the data [13].

Finally, it is important to notice that these methods may act in an undesired way when faced with non-convex data, with noise or outliers. That is the case of ISOMAP and LLE, despite the fact that the result obtained with ISOMAP is optimal according to Figure 2, where the embedding results are uniformly distributed and well organized. However, if the neighborhood graph has disconnected components or the data is non-convex (with holes) or contains high level of noise or outliers, it can return a result that deforms the global structure of the data, emphasizing the anomalous data. Diffusion Maps is able to recover the hole in the resulting embedding and it is the only one that is robust to noise and outliers. The reason is because, instead of using one shortest path like ISOMAP, Diffusion map takes averages of different paths on the neighborhood graph.

A summary of the advantages and disadvantages of each method can be seen in Table I and for more detail one can visit the referenced bibliographies [14] [15] [16] [17].

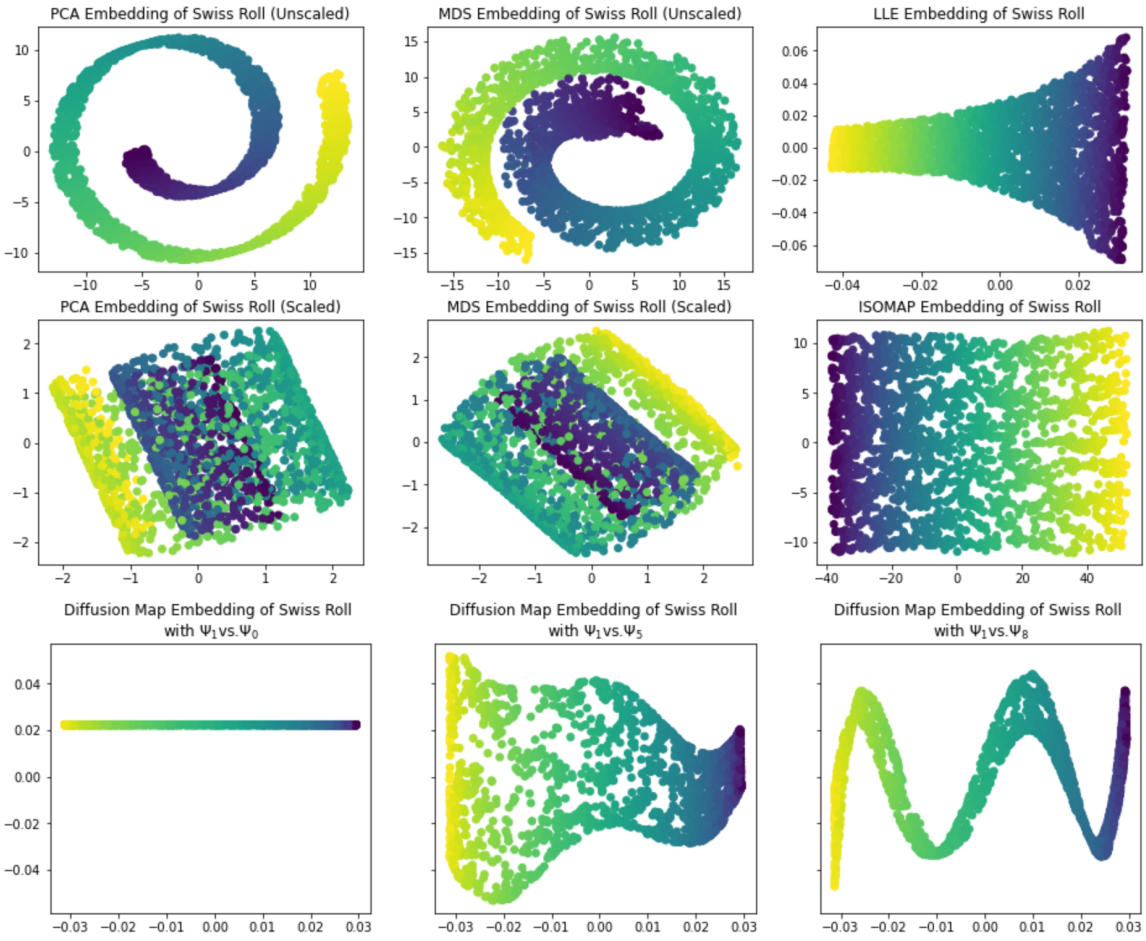

Fig. 2: Embedding of Swiss roll in a 2-dimensional subspace with different dimensionality reduction methods.

TABLE I: Comparative study of the methods

	Advantages	Disadvantages
PCA	Simple to implement. The optimizations do not involve local minima. Fast.	Not suitable for non-linear data. Only see Euclidean structure. Can't recover the underlying manifold of the data. Can't preserve the neighborhood structure of the data.
cMDS	Simple to implement. The optimizations do not involve local minima. Use pairwise distances between data points. Only need the pairwise distance matrix.	Not suitable for non-linear data. Only see Euclidean structure. Can't recover the underlying manifold of the data. Computationally expensive and slow.
ISOMAP	Can find globally meaningful coordinates and nonlinear structure. Preserves geodesic distances between data points.	Poor performance when manifold is not well sampled and contains holes. Sensible to the parameter. Not robust to noise perturbation and outliers. Computationally expensive and slow.
LLE	Simple to implement and only needs to store the weight matrix. The optimization is convex so it doesn't involve local minima. Capable of generating highly nonlinear embeddings preserving the topology of the dataset. Involve only one free parameter. Fast and computationally efficient.	Sensitive to outliers and noise. Often distort the global structure. Gives poor result in dataset that haven't a smooth manifold (contain holes). Sensible to the parameter.
Diffusion Maps	Robust to noise perturbation. Preserves the topology of the dataset. Able to handle dataset with holes.	Very sensible to the parameter. Can't deal with sparsity.

V. Conclusion

This paper introduces how useful manifold learning is within the dimensionality reduction problem environment. Furthermore, an overview of five dimensionality reduction techniques is presented, covering both linear and non-linear. In detail, the definition and algorithm of each algorithm is described. In the end, a comparative study between the techniques is demonstrated with a non-linear data set, the swiss roll dataset.

According to the results of the comparison, PCA is the fastest method of all but it is not appropriate for data belonging to a non-linear manifold. On the other hand, cMDS only deals with linear data and is much slower than PCA. However, when we only have information about the distance between pairs in the data set, it is much more convenient to apply cMDS. For non-linear data, LLE is fast, easy to apply, and requires little computational power compared to ISOMAP, but it has the disadvantage that it usually does not maintain the global structure of the dataset. ISOMAP is the method that best preserves the global and local structure of the data, but it must be kept in mind that it is slower and requires a lot of computational capacity. These methods are easily affected by noise and outliers in the data, except for diffusion map which performs well in this situation. Properly choosing the parameter can return a decent embedding.

Summarily, we can conclude that each method has its pros and cons, so before choosing one of them, it is important to analyze the properties of the data to be processed, to know the computational limitations of the problem and, based on that, choose the most appropriate method to treat it.

References

[1] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[2] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[3] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[4] J. De la Porte, B. Herbst, W. Hereman, and S. Van Der Walt, "An introduction to diffusion maps," in *Proceedings of the 19th symposium of the pattern recognition association of South Africa (PRASA 2008), Cape Town, South Africa*, 2008, pp. 15–25.

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

[6] L. Li, "Principal component analysis for dimensionality reduction," https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad.

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

[8] F. Wickelmaier, "An introduction to mds," *Sound Quality Research Unit, Aalborg University, Denmark*, vol. 46, no. 5, pp. 1–26, 2003.

[9] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[10] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee, "Spectral methods for dimensionality reduction." *Semi-supervised learning*, vol. 3, 2006.

[11] L. K. Saul and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *Journal of machine learning research*, vol. 4, no. Jun, pp. 119–155, 2003.

[12] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and computational harmonic analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[13] B. Bah, "Diffusion maps: analysis and applications," 2008.

[14] P. N. T. W. B. Póczos, M. Belkin, "Manifold learning," https://www.cs.cmu.edu/~bapoczos/Classes/ML10715_2015Fall/slides/ManifoldLearning.pdf.

[15] J. D. C. M. Alaiz, A. Fernandez, "Diffusion maps parameters selection based on neighbourhood preservation," *Computational Intelligence*, 2015.

[16] T. Lin, H. Zha, and S. U. Lee, "Riemannian manifold learning for nonlinear dimensionality reduction," in *European Conference on Computer Vision*. Springer, 2006, pp. 44–55.

[17] L. Cayton, "Algorithms for manifold learning," *Univ. of California at San Diego Tech. Rep*, vol. 12, no. 1-17, p. 1, 2005.

Jazmin Ines Teng is an exchange student at the Technical University of Munich (TUM) in the department of Informatics. She is currently finishing her bachelor's degree in Information Systems Engineering at the National Technological University (UTN) in Argentina. At the same time, she works as a researcher at the CInApTIC research center of UTN.