Distributed Persistence Based Dimensionality Reduction Method: DIPOLE

Rishendra Singh Chauhan

A dissertation submitted for the partial fulfillment of BS-MS dual degree in Science

Indian Institute of Science Education and Research, Behrampur

Registration Number: 19104

Department of Mathematical Sciences

April, 2024

Certificate of Examination

This is to certify that the dissertation titled **Distributed Persistence Based Dimensionality Reduction Method: DIPOLE** submitted by **Rishendra Singh Chauhan** (Roll No. 19104) for the partial fulfillment of BS-MS Dual Degree program of the institute, has been examined by the thesis committee duly appointed by the institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

G. Kasi Viswanadham

Dr. G Kasi Viswanadham
(Committee Member)

[illegible]

Dr. Amiya Mondal
(Committee Member)

[illegible]

Dr. Seshadri Chintapalli
(Co-Supervisor)

Bastian Rieck

Dr. Bastian Rieck
(Supervisor-1)

[illegible]

Dr. Amit Chattopadhyay
(Supervisor-2)

April, 2024

Declaration

I, Rishendra Singh Chauhan (Roll No: 19104), hereby declare that, this report entitled "Distributed Persistence Based Dimensionality Reduction Method: DIPOLE" submitted to Indian Institute of Science Education and Research Berhampur towards partial requirement of Master of Science in Mathematics is an original work carried out by me under the supervision of Dr. Amit Chattopadhyay and Bastian Rieck has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever an external information, statement, or result is used then, that has been duly acknowledged and cited. Berhampur - 760 010

[illegible]

Rishendra Singh Chauhan
April, 2024

Certificate from the Supervisor

This is to certify that Rishendra Singh Chauhan, student of BS-MS (10th Semester) in Indian Institute of Science Education and Research, Berhampur of Mathematical Sciences Department has carried out the dissertation titled, ”Distributed Persistence based Dimensionality Reduction Method: DIPOLE” under my supervision for the partial fulfillment of his BS-MS Degree in Department of Mathematical Science of this Institute. The thesis has been examined with the Turnitin software and is found to be within the limit of the plagiarism policy of the Department.

[illegible]

Dr. Amit Chattopadhyay
(Supervisor)
April, 2024

Acknowledgements

I would like to express my heartfelt appreciation to Dr. Amit Chattopadhyay, and Dr. Bastian Rieck for graciously allowing me to undertake my MS research project under their guidance. Dr. Chattopadhyay's invaluable mentorship, unwavering support, and encouragement have been instrumental throughout this journey. I am also grateful to Karthik and Yashwant, Dr. Chattopadhyay's PhD students, for their enriching discussions and valuable insights. Furthermore, I extend my gratitude to all the faculty members at IISER Berhampur for their exceptional coursework during the initial four years, which laid the groundwork for this project. Last but not least, I am deeply thankful to my parents, whose unwavering belief in me has served as a constant source of motivation.

Abstract

In this project, we study a dimensionality reduction method that preserves the topology of high-dimensional data, in its low-dimensional embedding. We study different dimensionality reduction methods and introduce a post-dimensionality reduction method, called DIPOLE. This method is based on distributed persistence, and it corrects the topology of low-dimensional embedding obtained through the initial dimensionality reduction method. Although it is claimed that this method can be used with any dimensionality reduction method, there are no results on this, except with Isomap. We verify this claim by using DIPOLE with a dimensionality reduction method called t-SNE. We also find certain limitations that DIPOLE presents, and then we give one way to deal with those limitations. Albeit not practical, this method might serve as the base for other methods that might be more practical, and efficient.

List of Figures

List of Tables

Contents

Chapter 1

Introduction to Computational Topology

In this chapter, we will informally introduce some concepts of computational topology. The reader is expected to have some background knowledge in algebraic topology. If not, one is advised to go through chapter 1 of [Dey 21]. We will be introducing some fundamental aspects of computational topology like simplicial complexes, chains, cycles, boundaries, homology, and persistence. We won't go into much detail. If interested in further details and definitions, one can access those from chapters 2 and 3 of [Dey 21]. We will be following these chapters for this chapter. For further information, one can also refer to [Nanda XX], which can be accessed from the following link. One can also refer to an alternate book, [Edelsbrunner 10], for understanding the concepts mentioned in this chapter.

1.1 Simplicial Complex

We use simplicial complexes to represent the topological spaces. These complexes can be decomposed into smaller fragments (or filtrations) that provide different information on the topology of the space. A simplicial complex for a point set is basically a collection of non-empty subsets of that point set, such that for every x in the point set, $\{x\}$ lies in that collection, and for every set (also known as simplex) in that collection, its subset also lies in that collection. These sets in the collection are called simplices, and all the subsets of the simplex that lie in the simplicial complexes are called faces of that simplex. The dimension of the simplex is defined as the cardinality of that simplex minus one, and the dimension of the simplicial complex is defined as the maximum dimension among all the simplices in that simplicial complex.

Any subset of a simplicial complex that satisfies the conditions required to be a simplicial complex is called a subcomplex of that simplicial complex. For any m less than or equal to the dimension of the simplicial complex K, the m-skeleton is defined as a subcomplex of K consisting of all the simplices with dimensions at most m. It is denoted by K^m.

The filtration of a simplicial complex is a nested sequence of subcomplexes, which looks like this:

$$F_1K \subset F_2K \subset F_3K \subset \cdots \subset F_NK = K.$$

The geometric simplex is the representation of the abstract simplex in some Euclidean space, $\mathbb{R}^n$. The geometric simplex formed by points $v_0, v_1, \ldots, v_k$ that lie in $\mathbb{R}^n$ is defined as the summation $\sum_{i=0}^{k} c_i v_i$, where c_i's are non-negative, and $\sum_{i=0}^{k} c_i = 1$, and the points $v_0, \ldots, v_k$ are affinely independent. This will help us define the geometric realization (or underlying space) of any simplicial complex. For simplicial complex K, and any vertex function $f : K_0 \to \mathbb{R}^n$, where K_0 is the set containing all the 0-dimensional simplices, we define the geometric realization of K with respect to f as

$$|K|_f = \bigcup |\tau|_f$$

where τ is a simplex in K, and $|\tau|_f$ is a geometric simplex spanned by the f-images of vertices in τ. If f is an affine embedding, i.e., it is injective, and the images of vertices are affinely independent then the topology of K remains the same regardless of the embedding. That is,

Proposition 1.1.1. *For any two affine embeddings $f : K_0 \to \mathbb{R}^n$ and $g : K_0 \to \mathbb{R}^n$, there is a homeomorphism between the corresponding geometric realizations in $\mathbb{R}^n$. [Dey 21]*

Therefore, we will follow the following notations: we will denote $|K|_f$ as $|K|$ where f is an affine embedding, leaving out any reference to the affine embedding. It's common to choose the endpoints of standard basis vectors in $\mathbb{R}^n$ as the targets for vertices. This choice guarantees that every simplicial complex K has a geometric realization that can be embedded in $\mathbb{R}^n$, where $n = \#K_0$.

We call a simplicial complex the triangulation of a manifold if the underlying space of that simplicial complex is homeomorphic to that manifold. If some simplicial complex is the triangulation of a manifold then the dimensions of that manifold, and that simplicial complex are also the same.

For a simplex τ in the simplicial complex K, its star is given by a set of simplices that have τ as their face. Usually, the star of a simplex is not a simplicial complex, but it can be made into a simplicial complex by adding the missing faces; we call it the *closed star*. It is, simply put, the closure of the star. The link of τ is the set of simplices in the closed star that are disjoint from τ. Intuitively, we can think of the star (and the closed star) of a vertex as an open (and closed) neighborhood around it, while the link represents the boundary of that neighborhood.

A simplicial map between two simplicial complexes, K and L, is the map from the vertex set of K to the vertex set of L, such that it sends every simplex in K to a simplex in L.

1.2 Rips and Čech Complexes

In this section, certain simplicial complexes that are featured in this thesis will be discussed. We will begin by defining a nerve. Consider a finite collection of sets $U = \{U_{\alpha_i} \mid \alpha_i \text{ with } i \in \mathbb{N}\}$. The nerve of U, denoted by $N(U)$, is defined as the simplicial complex such that $\{\alpha_0, \alpha_1, \dots, \alpha_k\}$ forms a simplex in $N(U)$ if and only if $U_{\alpha_0} \cap U_{\alpha_1} \cap \cdots \cap U_{\alpha_k} = \emptyset$.

One significant result concerning Nerves is the Nerve Theorem, presented as follows:

Theorem 1.2.1. *Given a finite cover U (open or closed) of a metric space M, the underlying space $|N(U)|$ is homotopy equivalent to M if every non-empty intersection $\bigcap_{i=0}^{k} U_{\alpha_i}$ of cover elements is homotopy equivalent to a point, i.e., contractible. [Dey 21]*

Now, the Čech complexes are for a metric space M, and finite set P are defined as the nerve of the set $\{B(p_i, r)\}$, where $r > 0$, and $B(p_i, r)$ represents a closed ball centered at p_i with radius r. If M is Euclidean, then the balls considered will be convex, resulting in their intersection being contractible. As per theorem 1.2.1, Čech complex becomes homotopy equivalent to a space formed by the union of these balls.

We define the Vietoris-Rips complex as follows: Consider a metric space M, and the finite set P, then for $r > 0$ Vietoris-Rips complex, denoted as $VR_r(P)$ is defined as a simplicial complex such that for all simplices in $VR_r(P)$ the pairwise distances between the vertices in any simplex is always less than $2r$, and vice versa.

We have the following relationship between Čech and Rips Complexes:

Proposition 1.2.1. *Given a finite subset P of a metric space (M, d). Then, $VR_r(P) \subseteq C^r(P) \subseteq VR_{2r}(P)$. [Dey 21]*

One can further read about other complexes, like alpha complexes, and graph-induced complexes, used in the field of computational topology from [Dey 21]. These complexes might prove useful in enhancing the DIPOLE pipeline, which we will discuss in later chapters of this thesis.

1.3 Chains, Boundaries, and Cycles

To understand this section, one is expected to have some understanding of Groups, Rings, Modules, Vector Spaces, and Fields. For a brief revision, one can refer to [Dey 21]. In this section, we will define Chains, cycles, and Boundaries, which are essential in understanding Homology.

For a simplicial complex K of dimension k, we define a p-chain, where $0 \leq p \leq k$, as the formal sum of p dimensional simplices, i.e., $c = \sum_{i=1}^{m} \alpha_i \sigma_i$, where σ_i's are p-simplices, and m is the number of p-simplices. And α_i's are the corresponding coefficient, obtained from the field $\mathbb{Z}_2$. The addition of two chains involves the addition of the corresponding coefficients, and since we are considering $\mathbb{Z}_2$, we will be having the following coefficient additions:

$$0+0=0,\ 1+0=0+1=1,\ 1+1=0$$

For instance, consider the following example taken from [Dey 21] of two p-chains, $c = e_1+e_3+e_4$, and $c' = e_1+e_2+e_3$, then $(e_1+e_3+e_4)+(e_1+e_2+e_3) = e_2+e_4$. This also leads to $c+c=0$, for any p-chain c.

Referencing Figure 1.1, we identify the following chains from the figure:-

0-chain: $(\{b\}+\{d\})+(\{d\}+\{e\}) = \{b\}+\{e\}$ (left)

1-chain: $(\{a,b\}+\{b,d\})+(\{b,c\}+\{b,d\}) = \{a,b\}+\{b,c\}$ (left)

2-chain: $(\{a,b,c\}+\{b,c,e\})+(\{b,c,e\}) = \{a,b,c\}$ (right)

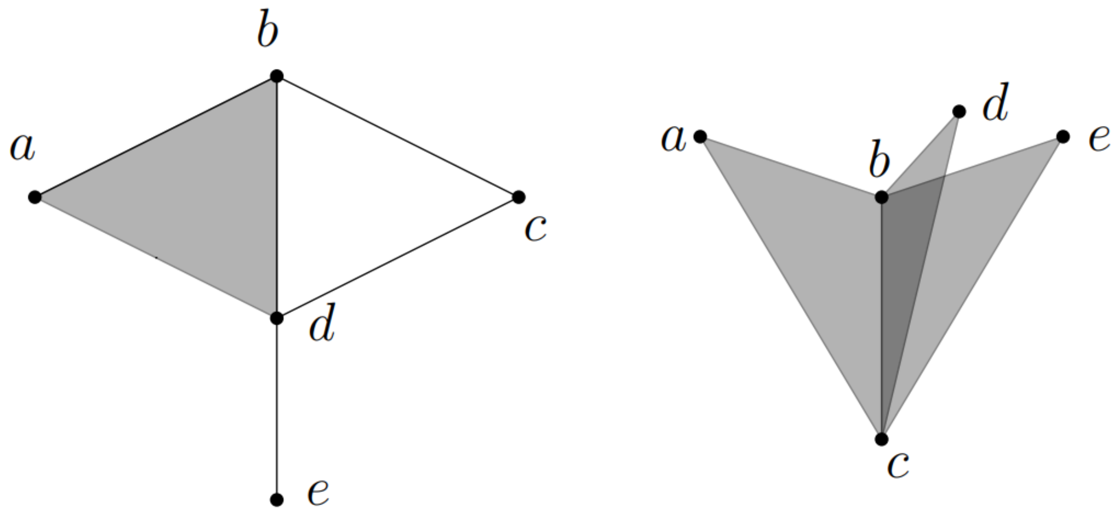


Figure 1.1: Chains, boundaries, and cycles [Dey 21].

The set of p-chains form a group called the p-th chain group, denoted by $C_p(K)$, where K is the simplicial complex. The identity is the 0-chain, i.e., a chain with all the coefficient values equal to zero, and the inverse of the chain is the chain itself.

Now, let τ be any p-dimensional simplex in a simplicial complex K. Then we define the boundary operator, δ_p, such that $\delta_p\sigma = \sum_{i=0}^{p} v_1 \dots \hat{v}_i \dots v_p$, where $\hat{v}_i$ means that vertex v_i is not included in the simplex. Thus, δ_p of any p-simplex returns a $(p-1)$-chain, which is the sum of all $(p-1)$-faces of the p-simplex. The boundary of any vertex is defined as empty. We can extend the boundary operator to a p-chain as follows: $\delta_p c = \sum_{i=1}^{m} \alpha_i(\delta_p\sigma_i)$ for a p-chain $c = \sum_{i=1}^{m} \alpha_i\sigma_i \in C_p$. Therefore, the boundary operator acts as a homomorphism between two chain groups, i.e., $\delta_p : C_p \to C_{p-1}$. It should be noted that for $p=0$, $\delta_p c = \emptyset$.

Thus, the chain group C_{-1} consists solely of one element, its identity 0. Furthermore, if K is a k-complex, then C_p is 0 for $p > k$.

We also have the following proposition, proof of which can be referred from [Dey 21].

Proposition 1.3.1. *For $p > 0$ and any p-chain c, $\delta_{p-1} \circ \delta_p(c) = 0$. [Dey 21]*

By extending the boundary operator to the chain groups, we obtain the following sequence of homomorphisms satisfying proposition 1.3.1 for a simplicial k-complex; such a sequence is also referred to as a chain complex:

$$0 \to C_{k+1} \xrightarrow{\delta_{k+1}} C_k \xrightarrow{\delta_k} C_{k-1} \xrightarrow{\delta_{k-1}} \dots \xrightarrow{\delta_1} C_1 \xrightarrow{\delta_1} C_0 \xrightarrow{\delta_0} C_{-1} \to 0.$$

A p-chain is called a p-cycle if its boundary equals zero, i.e., $\delta_p\, c = 0$. All p-cycles together form a group, Z_p, called the p-th cycle group.

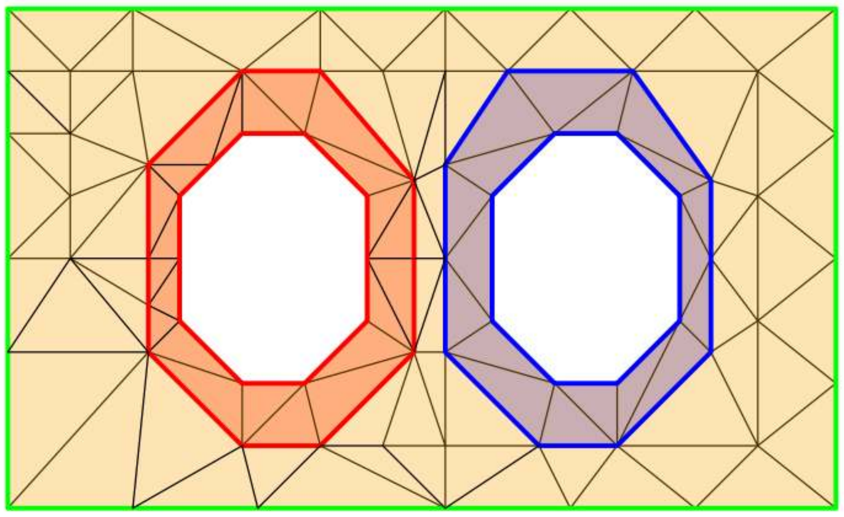


Figure 1.2: Boundaries and Cycles [Dey 21].

1.4 Homology

In the homology group, all the cycles that differ by a boundary form the same class. This means that a homology group is defined as a quotient group. The p-th homology group is defined as $H_p = Z_p/B_p$. H_p forms a vector space, and dimension$(H_p) = \beta_p$, where β_p is called the p-th Betti number.

In the homology group H_p, every element is derived by adding a p-cycle $c \in Z_p$ to the complete boundary group, noted as $c + B_p$, producing a coset of B_p in Z_p. All cycles formed by adding an element of B_p to c comprise the class $[c]$, recognized as the homology class of

c. Two cycles c and c_0 within the same homology class are named homologous, signifying that $[c] = [c_0]$. By definition, $[c] = [c_0]$ if and only if $c \in c_0 + B_p$. With Z_2 coefficients, this condition also implies that $c + c_0 \in B_p$. For example, in Figure 1.2, the outer cycle c_5 is homologous to the sum $c_2 + c_4$ because they together enclose the 2-chain comprising all triangles. Additionally, it's noteworthy that the group operation for H_p is characterized by $[c] + [c_0] = [c + c_0]$.

1.5 Topological Persistence

Topological persistence is a technique in topological data analysis that captures and quantifies the lifespan of topological features within a dataset. It identifies and monitors features like connected components, loops, and voids across various levels of resolution or scale. These are essentially the homological features of a metric space. At the heart of topological persistence lies filtration, which we'll now discuss.

1.5.1 Persistent Homology

Persistent Homology captures the evolution of the topology of a metric space in a parametrized sequence of spaces. This parametrized sequence is obtained by a pre-processing step that converts the finite metric space into a sequence of nested simplicial complexes called filtration. Two of the most common filtrations used are Rips and Čech filtration. We have used Rips filtration in our experiments. Rips filtration is easier to compute than Čech filtration, but it may contain less geometric information. Both filtrations involve the addition of simplices into the metric space at the parameter value equal to the proximity of the corresponding vertex point. As we move ahead in this sequence, the addition of certain simplices might lead to a change in the homological type of the metric space, we call such simplices critical simplices. Whenever the critical simplices appear, the corresponding parameter values are recorded via persistent homology. Persistent homology also records the dimensional change in homology that is caused by the critical simplex and pairs the critical value at which the new homological feature appears with the critical value at which that homological feature disappears. In short, we try to capture how long a certain homological feature persists in these filtrations. Persistence is basically the difference between these paired critical values. This information is organized in the form of persistence diagrams.

One can learn further about the topological persistence from Chapter 3 of [Dey 21].

1.5.2 Persistence Diagram

As discussed in the previous section, persistence diagrams note the pairs of critical values at which certain dimensional homological features (homological classes) appeared and then disappeared. The birth of such a feature is represented by the X-axis, and the death of such a feature is represented by the Y-axis. There are also certain features (or classes) that never die; their y-coordinate is kept as infinity; these are called essential points. While there are certain features that die at the same time that they are born, they are represented in the diagonal. Their multiplicity is infinity since they can keep on appearing and disappearing an infinite number of times. Have a look at Figure 1.3. It shows an example of a persistence diagram, alongside that it also shows the Barcode diagram of that persistence diagram, which captures the persistence of homological features.

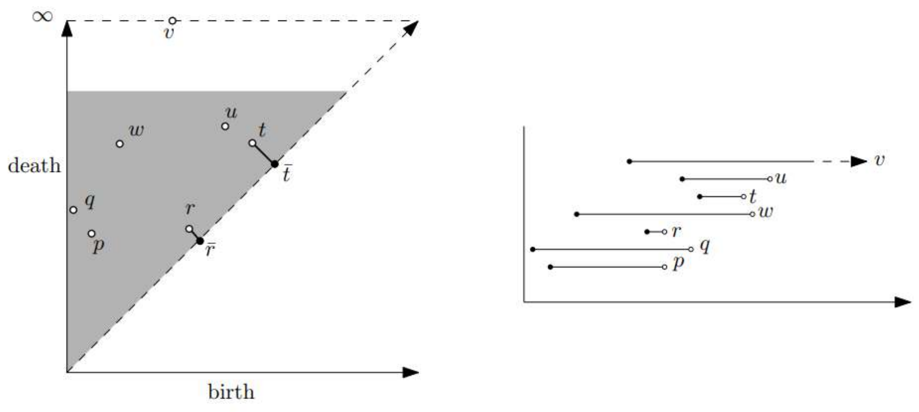


Figure 1.3: Left: Persistence Diagram, Right: Corresponding Barcode

Now we will define one kind of distance that we can calculate between two persistence diagrams. Two spaces having similar topology are bound to have lower such distance. That distance is called the Bottleneck distance.

Let $\Pi = \{\pi : PD_1 \to PD_2\}$ denote the set of all bijections from persistence diagram PD_1 to persistence diagram PD_2. Consider the distance between two points $x = (x_1, x_2)$ and $y = (y_1, y_2)$ in L^∞-norm $\|x - y\|_\infty = \max\{|x_1 - y_1|, |x_2 - y_2|\}$ with the assumption that $\infty - \infty = 0$. The bottleneck distance between the two diagrams is:

$$d_b(PD_1, PD_2) = \inf_{\pi \in \Pi} \sup_{x \in PD_1} \| x - \pi(x) \|_\infty.$$

We have the following theorem from [Dey 21]:

Theorem 1.5.1. *Let $f, g : K \to \mathbb{R}$ be two simplex-wise monotone functions giving rise to two simplicial filtrations F_f and F_g. Then, for every $p \geq 0$,*

$$d_b(Dgm_p(F_f), Dgm_p(F_g)) \leq \|f - g\|_\infty.$$

[Dey 21]

There's another distance called the Wasserstein distance. One can look for the Wasserstein distance and corresponding stability theorems in [Dey 21].

Chapter 2

Dimensionality Reduction

As the dimensions of data increase, it becomes more and more difficult to interpret and analyze, visualization gets close to impossible, and storage becomes expensive. One example of high-dimensional data is the image. For example, consider a 28×28 pixel image. A 28x28 pixel image has a total of 784 pixels. Each pixel corresponds to a single value representing the intensity of light or color at that particular position in the image. Therefore, a 28x28 pixel image can be represented as a vector with 784 dimensions, where each dimension represents the intensity of a pixel. Since high dimensions present many problems, it would be better to get a low-dimensional representation for such datasets, with certain characteristics that we want to study in high-dimensional datasets preserved. Dimensionality reduction presents one such way. It is a process of getting a low-dimensional representation for high-dimensional data, while also preserving those certain characteristics of the high-dimensional dataset.

In the following chapter, we will discuss three dimensionality reduction methods: PCA, Isomap, and t-SNE. The latter two methods are often used in experiments done in the later chapters of this thesis.

2.1 Principal Component Analysis

Principal Component Analysis (PCA), first introduced by Pearson [Pearson 01], and Hotelling [Hotelling 33], is one of the most commonly used dimensionality reduction methods. It's a linear dimensionality reduction method that aims to preserve the highest amount of variance possible in the low-dimensional embedding of the high-dimensional dataset. Figure 2.1 shows how PCA works. In the figure, PCA has been implemented to reduce the dimensions of the point cloud from 2 dimensions to 1 dimension. It is clear from 2.1(a) that the point cloud data is not varying much in x_2 axis, while it has more variance in x_1 axis, so while performing dimensionality reduction, we have only considered the axis in which

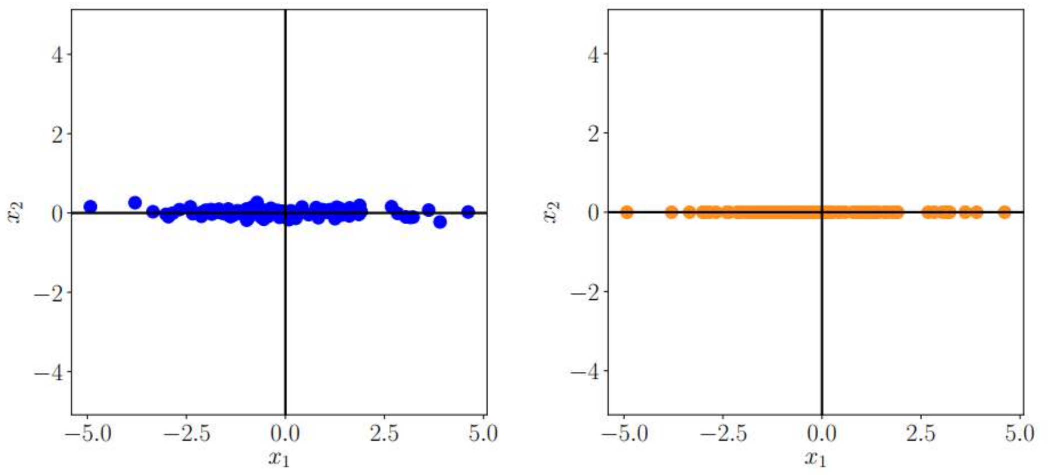


Figure 2.1: Illustration on how PCA works. [Deisenroth 19]

most amount of variance is preserved (i.e. x_2 axis). This is how PCA works, it first finds the axis (or axes, based on target dimensions) that shows the maximum amount of variance in the data, and then it returns the projection of the point cloud on that axis (or those axes). We won't be using PCA in our thesis so for further details one can refer to chapter-10 of [Deisenroth 19].

2.2 Isomap

Isomap, short for Isometric Mapping, is a nonlinear dimensionality reduction technique used in machine learning and data analysis. It was first introduced in [Tenenbaum 00]. It aims to preserve the intrinsic geometric structure of high-dimensional data by representing it in a lower-dimensional space while maintaining the pairwise geodesic distances between all points. Unlike linear methods such as PCA, Isomap captures the underlying manifold structure of the data, making it particularly useful for tasks involving nonlinear relationships and manifold learning.

Before discussing Isomap, we will be understanding Multi-Dimensional Scaling (MDS), because Isomap is a particular type of MDS.

2.2.1 Multi-Dimensional Scaling

The first paper on Multidimensional Scaling (MDS) is often attributed to Shepard, R. N [Shepard 62]. Multidimensional Scaling (MDS) is a dimensionality reduction method used for visualizing the structure of pairwise dissimilarities or distances between data points.

The goal of MDS is to find a configuration of points in a low-dimensional space such that the pairwise distances or dissimilarities in that space approximate those in the original high-dimensional space as closely as possible.

Assume a collection of n objects, $X = \{x_1, \dots, x_n\}$, which can be points, images, documents, people, etc. Suppose also that we are given the pairwise dissimilarities of the objects in the set x,

$$D = (d_{ij}) \in \mathbb{R}^{n \times n}, \text{ where } d_{ij} = \text{dissimilarity}(x_i, x_j), \quad 1 \leq i, j \leq n,$$

which specify in some way how different the objects are from each other.
Note that these dissimilarities are not necessarily computed based on a valid distance metric and may violate some of the conditions required by a distance metric (e.g., the triangle inequality).

Given the matrix of pairwise dissimilarities $D = (d_{ij}) \in \mathbb{R}^{n \times n}$, the goal of Multidimensional Scaling (MDS) is to represent the objects as points in some Euclidean space, say $y_1, \dots, y_n \in \mathbb{R}^k$, such that

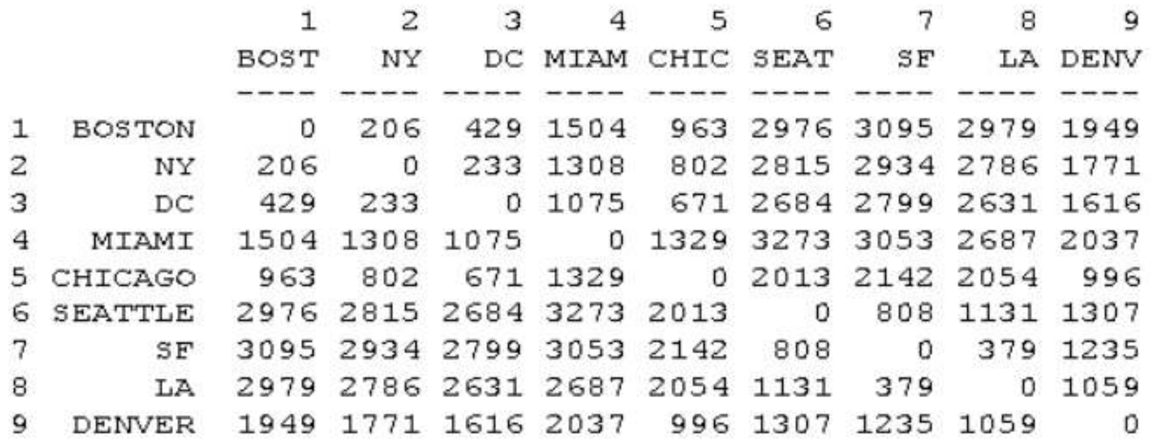
$$\|y_i - y_j\|_2 = d_{ij} \quad \text{(or as closely as possible)}, \quad \forall i, j,$$

so as to visualize their proximity relationships and global appearance. One can refer to the following algorithm 2.2.1 of MDS [Borg 05]. Note that when the dissimilarity is Euclidean distance then MDS as dimensionality reduction is identical to the PCA.

Algorithm 1 Classical Multidimensional Scaling (MDS)

```
Require: Dissimilarity matrix $D$ of size $n \times n$
Ensure: Low-dimensional representation $X$ of size $n \times k$
1: Construct the Dissimilarity Matrix: Compute the dissimilarity matrix $D$ based on the given data.
2: Centering the Dissimilarity Matrix: Center the dissimilarity matrix $D$ by double centering: $B = -\frac{1}{2}(I - \frac{1}{n}J)D(I - \frac{1}{n}J)$
3: Eigenvalue Decomposition: Compute the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_m$ and eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m$ of $B$.
4: Selecting Dimensions: Choose the number of dimensions $k$ for the low-dimensional representation.
5: Constructing the Low-dimensional Embedding: Form the low-dimensional representation $X$: $X_{ik} = \sqrt{\lambda_k} v_{ik}$
6: Visualization: Visualize the low-dimensional embedding $X$ to understand the underlying structure of the data.
```

Have a look at the figure 2.2 and 2.3. They explain how MDS works.



		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0

Figure 2.2: Dissimalirity Matrix based on distances between some places in the United States

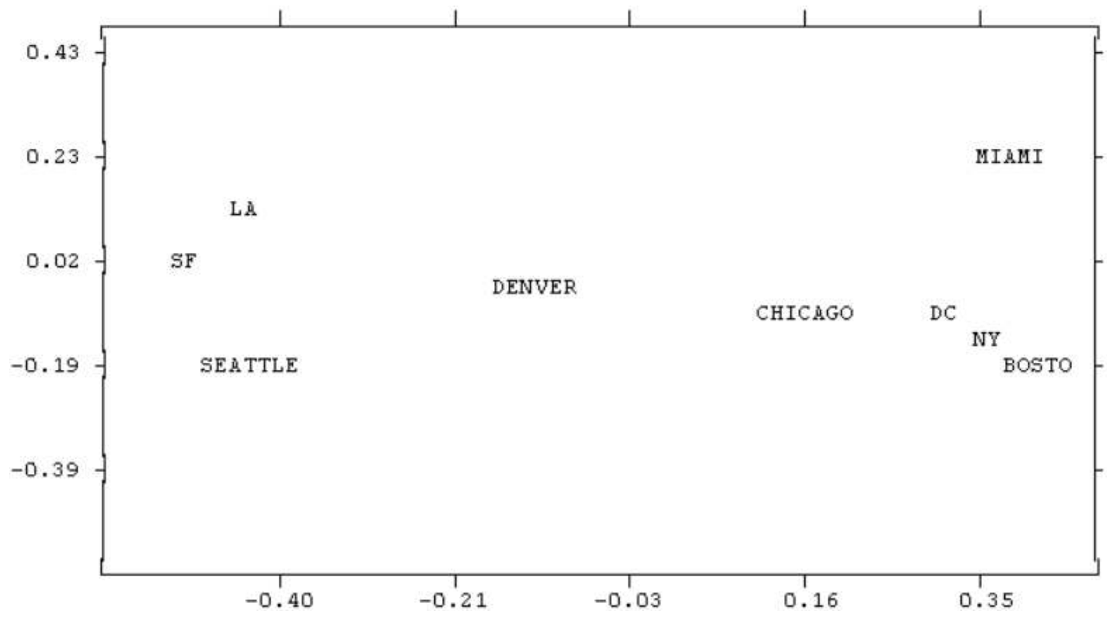


Figure 2.3: MDS preserves the dissimilarity matrix as shown in figure 2.2

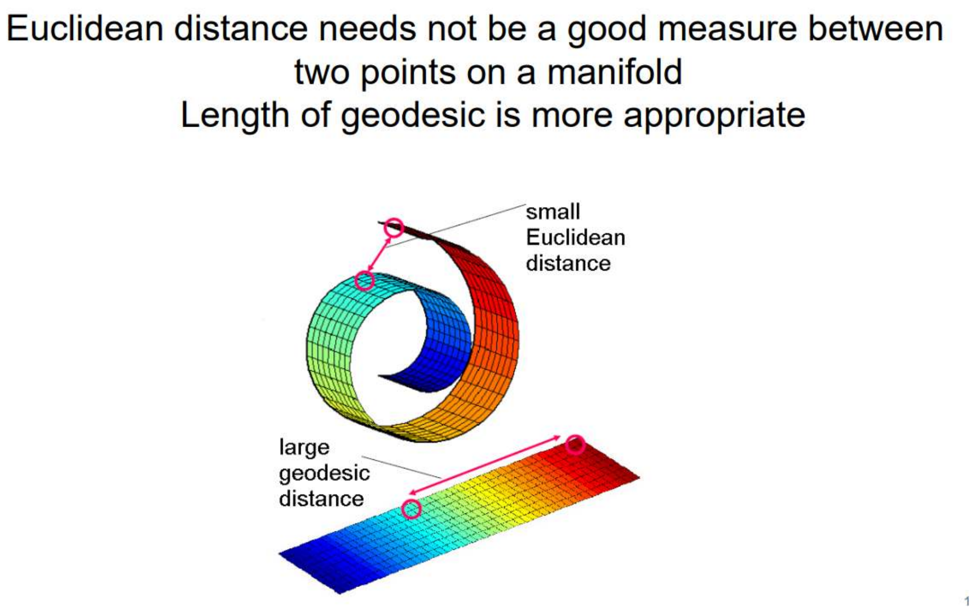

Euclidean distance needs not be a good measure between two points on a manifold
Length of geodesic is more appropriate
small Euclidean distance
large geodesic distance
1

Figure 2.4: Why Euclidean Distance might not work? [Borne 21]

2.2.2 Isomap

We noted that when dissimilarity is equal to Euclidean distance then MDS is identical to doing a PCA. When dissimilarity is equal to Geodesic distance we get Isomap.

If we apply PCA (or MDS with Euclidean Distance) to the Swissroll dataset. We will get the following drawbacks:

- PCA may show distant points along the manifold near each other, and to rectify this we might need to keep the dimensions as high as possible;
- PCA will fail to represent the curved dimensions.

So, how about we try another distance? Specifically, the Geodesic distance. The geodesic distance between two points on a manifold is the shortest distance along that manifold. In practical scenarios, where we're given a dataset X sampled from an unknown manifold M, we can approximate the true geodesic distances $d_M(i,j)$ by the shortest-path distances $d_G(i,j)$ on a nearest-neighbor graph G built on the dataset.

So, we will now try to preserve the geodesic distance instead of Euclidean distance, as the geodesic distance is basically the distance on the manifold and thus it will be able to capture the true, non-linear geometry and the global structure of the manifold. This is what Isomap does, it tries to preserve the geodesic distance, and thus in turn capture the manifold on which the high-dimensional dataset lies. One can further read about Isomap, and how it works in [Chen 20].

2.3 t-SNE

t-distributed Stochastic Neighbor Embedding (t-SNE) is a popular nonlinear dimensionality reduction method used, primarily, to visualize clusters in high-dimensional datasets. It tries to preserve the local, and global structure of data as much as possible.

Let $X = \{x_1, x_2, ..., x_n\}$ be the high-dimensional input data, where x_i represents the i-th data point with d-dimensional features. t-SNE constructs two probability distributions over pairs of data points: a conditional probability distribution $p_{j|i}$ that represents the similarity between data point x_i and x_j in the original high-dimensional space, and a similar conditional probability distribution $q_{j|i}$ in the lower-dimensional space. By similarity, here, we mean the probability of two points being the neighbor of each other.

The similarity between data points in the original space is often defined using a Gaussian kernel:

$$p_{j|i} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-||x_i - x_k||^2/2\sigma_i^2\right)}$$

where σ_i is the variance of the Gaussian kernel that is chosen based on the perplexity parameter, a hyperparameter that controls the effective number of neighbors considered for each data point.

In the lower-dimensional space, t-SNE defines the conditional probabilities using the Student's t-distribution:

$$q_{j|i} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||y_i - y_k||^2)^{-1}}$$

where y_i represents the corresponding lower-dimensional representation of data point x_i. The goal of t-SNE is to minimize the mismatch between these two distributions by adjusting the lower-dimensional embeddings y_i iteratively using gradient descent (Appendix A). The Kullback-Leibler divergence (loss function) between $p_{j|i}$ and $q_{j|i}$ is minimized with respect to the lower-dimensional embeddings y_i, leading to a configuration where similar data points are mapped close to each other in the lower-dimensional space. This process effectively captures the local and global structure of the data in a lower-dimensional representation suitable for visualization.

This link gives some idea on how t-SNE gives the results. One can further understand on how t-SNE functions from [Starmer 14].

Chapter 3

Distributed Persistence

3.1 Distributed Persistence Diagram

To capture the topology of any point cloud data, one calculates the *full* persistence diagram of that dataset. However, it becomes expensive to compute for a dataset with high cardinality. Also, it is unstable to outliers. One other limitation of the persistence diagrams is that they are not injective, i.e., there might exist two very different datasets that may give the same persistence diagram. In the following chapter, a new topological invariant is introduced that deals with all the previous limitations. In this invariant, we compute the collection of the persistence diagram of subsets of the dataset instead of taking the persistence diagram for the whole dataset. This invariant is parallelizable and is robust against outliers. Also, it has some very nice inverse properties, that is, if two datasets give the same invariant, then that would mean two datasets being isometric to each other. This is the result of its kind in TDA literature, and the invariant is called the distributed persistence diagram.

In the following chapter, we will begin by introducing distributed persistence invariant, then we will go through the inverse properties of distributed persistence, and in the end, we will show some experimental results based on distributed persistence and talk about some future directions for distributed persistence.

3.2 Distributed Persistence Diagram

Let (X, d_X) be a finite metric space. We define λ as a map that sends X to its persistence diagram $\lambda(X)$. Then for any $k \in \mathbb{Z}_+$, we define distributed persistence diagram as follows,

$$\lambda_k(X) := \{(S, \lambda(S)) \mid S \subset X \text{ and } |S| = k\}.$$

This here means that λ_k will record the persistence diagrams of all subsets of X of cardinality k.

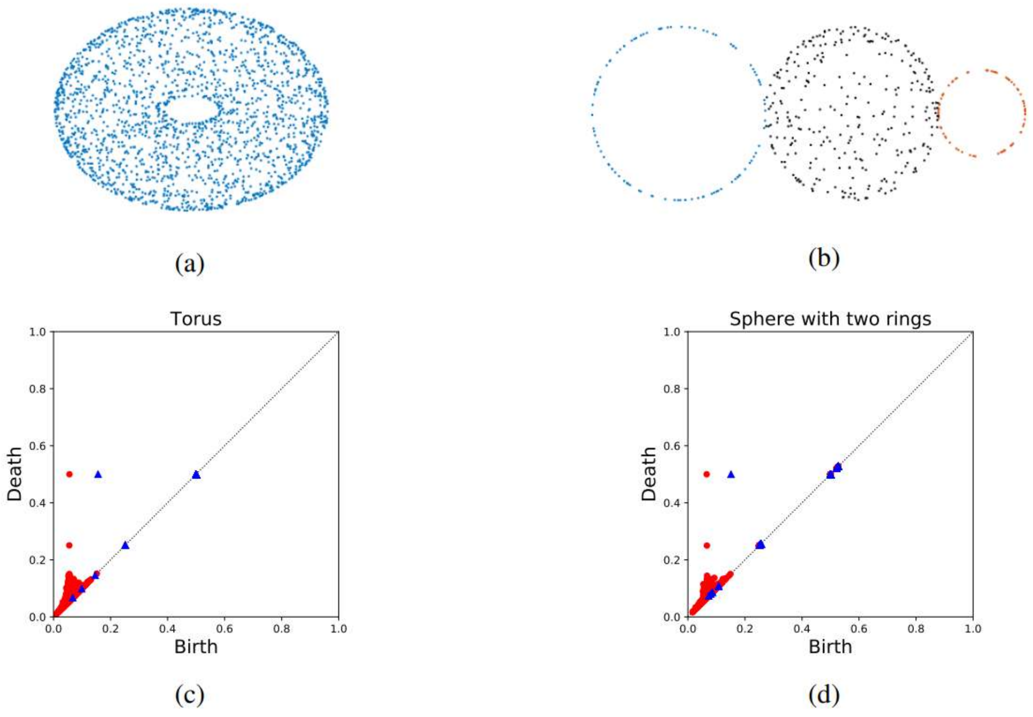


Figure 3.1: Different topological spaces giving similar persistence diagrams [Reani 21].

We know that the computational complexity of λ increases with the size of the dataset X, but in the case of λ_k, we are computing λ for many subsets of X of size k. Also, we can perform all such calculations parallelly, since each computation is independent of the other. Therefore, the size of X won't have any impact on the calculation of λ_k. This, in turn, increases the computation speed.

Also, consider the case when there are outliers in the dataset. When calculating λ for the whole dataset X, our lambda is bound to include those outliers in its computation. But when we are computing λ for many small subsets of a certain fixed size, there might be some subset that won't have any outlier in it. Therefore, λ_k will be more robust to outliers when compared with λ.

Alongside all these advantages λ_k also possess some nice inverse properties, which we will see in a while.

3.3 Non-Injectivity of Persistence Diagram

Persistence diagrams are not injective. It is possible that two very different topological spaces might give the same persistence diagram. Have a look at Figure 3.1 [Reani 21]. It can be seen in the figure that two very different topological spaces (a torus and a sphere with two rings attached to its sides) have given very similar persistence diagrams. If we

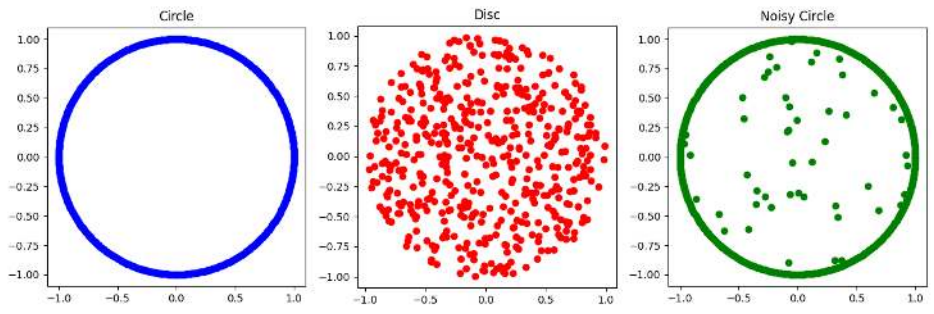


Figure 3.2: Three point clouds: Circle, Disc, and Noisy Circle.

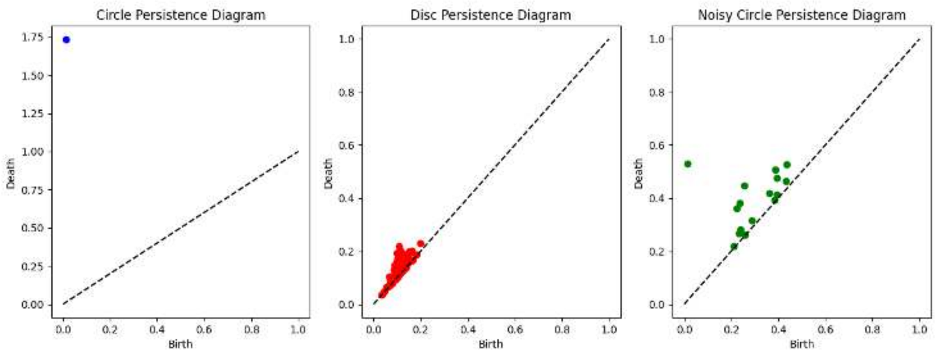


Figure 3.3: Persistence diagrams of the datasets.

calculate the bottleneck distance between these two persistence diagrams it will be around zero, but their corresponding metric spaces are very different in terms of topology.

Now we will have a look at the case study of a noisy circle [Solomon 22]. We will be comparing three datasets, shown in Figure 3.2.

1. Circle: In this dataset, 500 points are sampled from a unit circle.
2. Disc: In this dataset, 500 points are sampled from a unit disc.
3. Noisy Circle: In this dataset, 450 points are sampled from a circle, and 50 points are sampled from the unit disc.

Now, we will compute their respective 1-dimensional persistence diagrams, shown in Figure 3.3. We have additionally calculated the bottleneck distance between each pair of these persistence diagrams. The bottleneck distances come as follows:-

- Circle and Disc: 0.8618206433020532
- Circle and Noisy Circle: 0.8618206433020532
- Disc and Noisy Circle: 0.3744002003222704

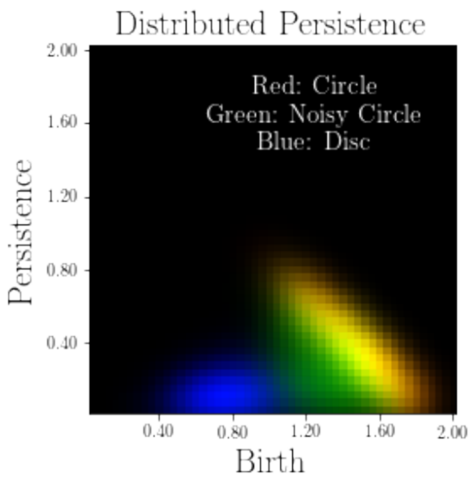


Figure 3.4: Average of distributed persistence images of the three datasets [Solomon 22].

From the above results, we can concur that the persistence diagram sees the noisy circle as more similar to a disc. This tells us that the ordinary persistence diagram fails to recognize the circle around which most data points are clustered in the noisy circle dataset. One thing to note is that the result is quite evident from Figure 3.3 itself; we have computed the bottleneck distance as an additional measure to confirm the result.

Now, for the case of the distributed persistence diagram of the above point clouds, we have taken the results obtained in [Solomon 22]. The following steps are performed to get the results shown in Figure 3.4 :-

1. Sample 1000 subsets of size 10 from the dataset.
2. Compute 1000 1-dimensional persistence diagrams for those subsets.
3. Vectorize them as persistence images.
4. Average the results to generate the final persistence image.

We have also shown the persistence diagram that was obtained in [Solomon 22], that we can also get by merging the diagrams we obtained in Figure 3.3, and then using the diagonal as x-axis, in Figure 3.5.

In Figure 3.4, the circle is represented by the red region, the disc is represented by the blue region, and the noisy circle is represented by the green region. The yellow/orange region is due to the overlay of the circle's red distribution and the noisy circle's green distribution. In the figure, it is visible that the noisy circle, although it interpolates between the circle and the disc, is much closer to the circle than it is to the disc. We haven't defined the bottleneck distance between distributed persistence diagrams, so we haven't calculated

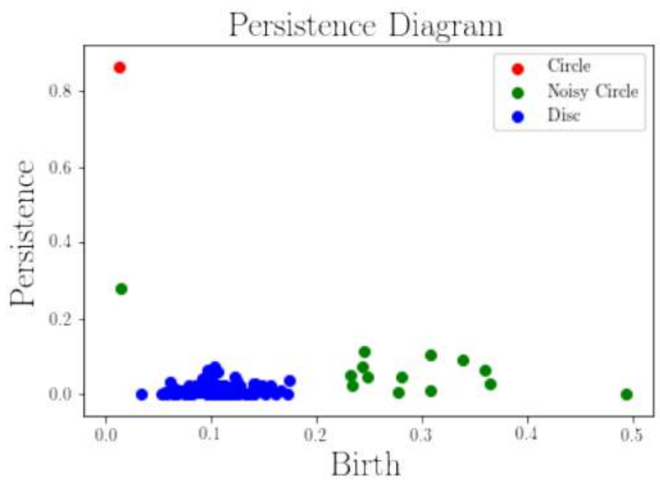


Figure 3.5: Persistence diagram of three datasets [Solomon 22].

the distance for these diagrams. As we move forward, we will see that it will be defined as the average of the bottleneck distances between the persistence diagrams of all the subsets.

3.4 Inverse Theorems

In this section, we will show some theoretical results obtained in [Solomon 22]. We will be focusing on the stability and inverse stability theorems, and we will conclude this section with probabilistic results that will help in the realistic application of inverse stability theorems in DIPOLE.

3.4.1 Stability Theorem

First, let's define quasi-isometry and quasi-isometry distances which will be useful in later theorems. One thing to note is that from here on we will be restricting λ to a map that sends a metric space to its persistence diagram based on Rips filtration, similarly for λ_k.

Definition 3.4.1. *Consider two metric spaces, (X, d_X) and (Y, d_Y). Let there be a map $f : (X, d_X) \to (Y, d_Y)$, then this map is called **ε-quasi-isometry** if $|d_X(x,y) - d_Y(f(x), f(y))| \leq \varepsilon \; \forall \, x,y \in X$.*

Definition 3.4.2. *We define quasi-isometry distance between two metric spaces X and Y, as the smallest ε among ε-quasi-isometries between X and Y. That is,*

$$d_q(X,Y) = \inf\{\varepsilon \,|\, \forall f : X \to Y, \text{ such that } |d_X(x,y) - d_Y(f(x), f(y))| \leq \varepsilon \;\; \forall x,y \in X\}.$$

Theorem 3.4.1. *Let $f : X \to Y$ be an ε-quasi-isometry, and let's denote the Bottleneck distance between two persistence diagrams with $d_B(\cdot\,,\cdot)$, then*

$$d_B(\lambda(S), \lambda(f(S))) \leq \varepsilon \;\; \forall S \subseteq X.$$

One thing to note from the previous theorem is that for all subsets of X of cardinality k, the bottleneck distance between their respective persistence diagram and that of their image in Y will be less than ε. We can take the average of all such bottleneck distances for all subsets of cardinality k, and this average can be used as a distance between two distributed persistence diagrams ($\lambda_k(X)$ and $\lambda_k(f(X))$).

3.4.2 Inverse Stability Theorems

The inverse theorems tell us that if we have two distributed persistence diagrams that are very similar but not identical then there ought to be a quasi-isometry between the metric spaces which gave those distributed persistence diagrams. Also, if two metric spaces give the same persistence diagram, then the two metric spaces will be isometric. To understand this, one needs to look at the definition 3.4.2, now suppose if $\varepsilon = 0$, then pair-wise distances will be preserved in the map $f : X \to Y$, in turn implying that two metric spaces, X and Y, are isometric.

Now let's define a space of metric spaces M with quasi-isometry distance as d_q, and a space of distributed persistence diagrams P with distance d_b. One thing to note is that this is not the regular bottleneck distance that we use for persistence diagrams. This distance will be properly defined when we discuss the theorems.

Now, let $\varphi : M \to P$ be a map that sends a metric space to its distributed persistence diagram. Then φ will be bi-lipschitz map. That is, for any $K \geq 1$, we have

$$\frac{1}{K} d_q(X_1, X_2) \leq d_b(\varphi(X_1), \varphi(X_2)) \leq K d_q(X_1, X_2)$$

for all $X_1, X_2 \in M$. Which further implies,

$$d_q(\varphi^{-1}(D_1), \varphi^{-1}(D_2)) \leq K d_b(D_1, D_2)$$

where D_1 and D_2 are distributed persistence diagrams in P. Now suppose if $D_1 = D = D_2$, then $d_b(D_1, D_2) = 0$, implying that quasi-isometry distance between the inverse images of D_1 and D_2 (or the metric spaces that gave these diagrams) is equal to 0. This tells us that there exists an isometry between the two spaces. Therefore all the metric spaces that will be giving the same distributed persistence diagrams will be isometric to each other.

Before we move to the main theorems we define the following notations:-

- λ^m:= λ restricted to m-skeleton of Rips complex. Also note that the output of λ^m is a collection of all persistence diagrams, one for each degree up to m.

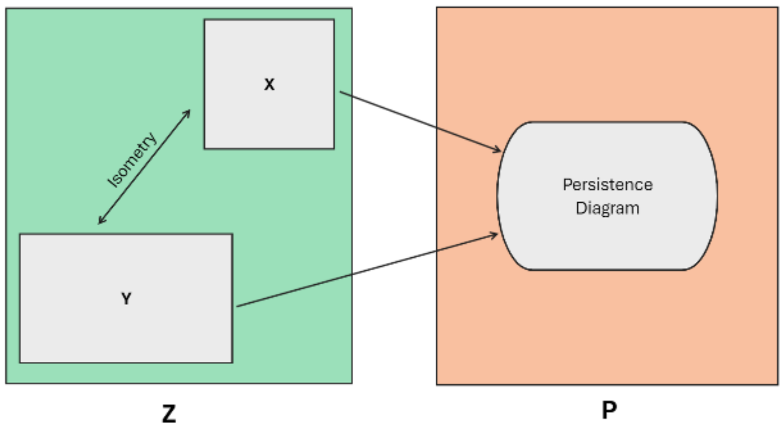


Figure 3.6: Metric spaces giving the same distributed persistence diagrams are isometric.

- The bottleneck distance is the maximum of all bottleneck distances across all degrees.

Theorem 3.4.2. *Let Z, and Y, be two metric spaces with a mapping $\varphi : Z \to Y$, and $\varphi|_{X \subset Z} : X \to Y$ being onto. Let $k > m > 0$, and let $\Gamma \subset P(Z)$ be a collection of subsets with cardinalities $\{k, k-1, \ldots, k-m-1\}$ and satisfying following properties:-*

- ***X-Covering Property**: $\forall \{x_1, x_2\} \subseteq X$, $\exists\, S \in \Gamma$ such that $|S| = k$ and $\{x_1, x_2\} \subseteq S$.*
- ***Closure Property**: $S \in \Gamma$ such that $|S| = k$, and $S_0 \subset S$ such that $|S_0| \geq k-m-1 \Rightarrow S_0 \in \Gamma$.*

Then if $d_B(\lambda^m(S), \lambda^m(\varphi(S))) \leq \varepsilon$ then $\varphi|_X$ is a $112k^2\varepsilon$ quasi-isometry, meaning

$$|d_X(x,y) - d_Y(\phi(x), \phi(y))| \leq 112k^2\varepsilon, \forall x, y \in X.$$

In order to satisfy X-Covering Property, and Closure Property, we can do the following:
(1) $X = Z$,
(2) $\Gamma = \{S \subseteq X \mid |S| = k, k-1, \ldots, k-m-1\}$.
Point (2) will make sure all the subsets S of required cardinalities are there in Γ, satisfying closure property, and points (1), and (2), will together ensure X-covering property. This will give a modified theorem where $X = Z$, and Γ contains all the subsets of X of cardinality k through $k-m-1$, and we would not require extra conditions of X-covering, and closure property. However, it will be much easier to satisfy these conditions with a smaller collection of subsets, which we will see in the probability theorems. But for now, for simplicity purposes, we will be considering all the subsets.

Theorem 3.4.3. *Let X and Y be the metric spaces such that $\varphi : X \to Y$ is onto, and let $k > m > 0$. If for all $S \subseteq X$ with $|S| = k, k-1, \ldots, k-m-1$, $d_B(\lambda^m(S), \lambda^m(\varphi(S))) \leq \varepsilon$, then $\varphi|_X$ is a $112k^2\varepsilon$ quasi-isometry.*

The above theorem can be further improved to give linear dependence on subset size k, only if φ preserves the pair-wise distances on some small subset of X. The resultant theorem is the one on which DIPOLE is based.

Theorem 3.4.4. *Let X, and Y be metric spaces, and let $k > m = 1$.*
Then if $\varphi : X \longrightarrow Y$ is a surjection such that $\forall\, S \subseteq X$ with $|S| \in \{k, k-1\}$,

$$W_1(\lambda^1(S), \lambda^1(\varphi(S))) \leq \varepsilon_1$$

and alongside this suppose $\exists X' \subset X$ of cardinality $(k-1)$ with

$$\sum_{(x_i,x_j)\in X'\times X'} |d_X(x_i,x_j) - d_Y(\varphi(x_i),\varphi(x_j))| \leq \varepsilon_2$$

then φ is $56(k+1)\varepsilon_1 + 28\varepsilon_2$ quasi-isometry, i.e. $\forall\, x_1, x_2 \in X$

$$|d_X(x_1,x_2) - d_Y(\varphi(x_1),\varphi(x_2))| \leq 56(k+1)\varepsilon_1 + 28\varepsilon_2$$

Note:-
W_1 distance here is the sum of W_1 distances for zero and first-degree persistent homology.

The implication of the above theorem is that by maintaining a bound over the distance between the persistence diagram of subsets of the metric space and their respective images, whilst also maintaining a bound over the difference between pair-wise distances of points in a small subset of that metric space, and pair-wise distances between the images of those points in co-domain, we are able to get a linear bound over the difference between pair-wise distances of points in whole metric space and pair-wise distances between their images in co-domain. This helps us in two ways:-

1. We are able to maintain quasi-isometry between the metric space and its image.

2. We are able to maintain a similar topology between the metric space and its image.

These two implications will help in setting up of DIPOLE, which we will see in the next chapter.

3.4.3 Probability Theorem

In theorem 3.4.3, and theorem 3.4.4 we have considered the collection of all the subsets so that it would be bound to satisfy covering and closure properties of theorem 3.4.2. We cannot use such kind of collection in real experiments because it would be very time-consuming. In fact, in the coming theorem, we will show that a relatively small collection of randomly chosen size-k subsets is likely to satisfy the covering property. Therefore we can generate a collection that is both covering and closed. We will assume $Z = X$ and $p = 2$ for the upcoming theorem.

Theorem 3.4.5. *Let $|X| = n$, and the collection $\Gamma = \{S_1, S_2, \cdots, S_M \mid |S_i| = k \; \forall i \in \{1, 2, \cdots, M\}\}$ be chosen by uniform sampling without replacement. Let $p < k$, and A be the outcome that every set of p points $\{x_1, x_2, \cdots, x_p\} \subseteq S_i$ for some $i \in \{1, 2, \cdots, M\}$. Then*

$$P(A) \geq 1 - \binom{n}{p} \left(1 - \left(\frac{k-p+1}{n-p+1}\right)^p\right)^M.$$

Proof of this theorem can be found in the appendix of [Solomon 22]. This theorem gives the lower bound on the probability of a certain number of subsets of a certain size satisfying the covering property.

Theorem 3.4.6. *Let A be the outcome as defined in the previous theorem. For any $\varepsilon \in (0, 1)$, if*

$$M \geq \left(p \log \frac{ne}{p} - \log(1-\varepsilon)\right) \left(\frac{n-p+1}{k-p+1}\right)^p$$

then $P(A) \geq \varepsilon$.

Proof of this theorem can be found in the appendix of [Solomon 22]. With this theorem, we can find the size of the collection that we would need for the given cardinality of the dataset, and cardinality of subsets to be considered in the collection.

3.5 Experiment

The goal of the following experiment is to confirm the theoretical results discussed before. This experiment is taken from [Solomon 22], and you can find the code at `https://github.com/aywagner/DIPOLE`. Now let's have a look at the experiment.

Let X and Y be finite subsets of the Euclidean space, and $\varphi : X \to Y$ be a surjection. According to theorem 3.4.2, we can check if φ is a quasi-isometry by evaluating $d_B(\lambda^m(S), \lambda^m(\varphi(S)))$ for a certain collection of subsets $S \subset X$. Now suppose if X is fixed and Y is variable, then if we can minimize $d_B(\lambda^m(S), \lambda^m(\varphi(S)))$, it should have the effect of bringing the alignment of Y closer to that of X. For the given experiment, we take $X = S^1$ (circle) with $|X| = 100$. Y will be initialized as X with independent Gaussian noise added coordinate-wise. Our goal will be to minimize the loss functional based on distributed persistence by gradient descent and check whether by doing so we are able to correct the geometric distortion that happened due to the addition of Gaussian noise. We define the loss functional as $W_2^2(D_0(S), D_0(\varphi(S))) + W_2^2(D_1(S), D_1(\varphi(S)))$, where W_2^2 is the Wasserstein distance with $p = 2$, and D_i represents the i-th degree persistence diagram of the Rips filtration of the subset S of size $k = 25$.

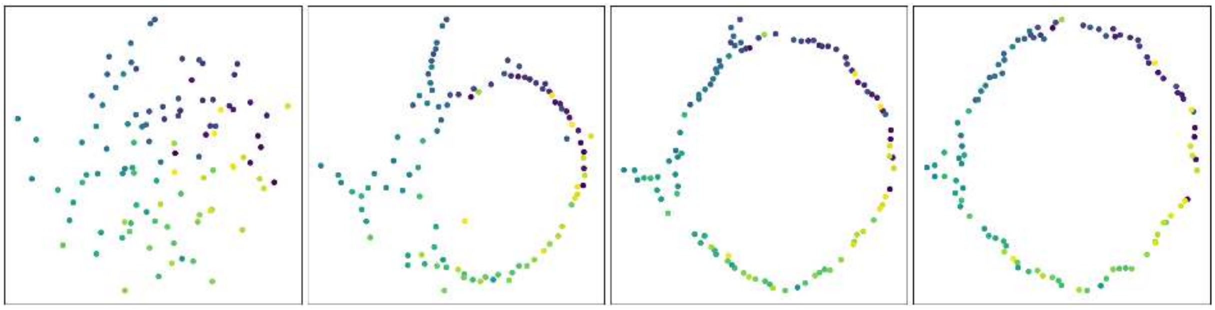


Figure 3.7: Synthetic dataset optimization experiment.

Look at the figure 3.7, each panel from left to right shows Y after successive multiples of 2^{11} gradient descent iterations. It can be seen that all the points reorganize themselves in the shape of X, therefore correcting the topology of Y. But also, the color gradient of those points becomes continuous. The color gradient is based on the distance between two points, i.e., the closer the points, the more similar the color. The fact that the color gradient becomes continuous implies that not only are we able to maintain the topology but we are also able to maintain the pair-wise distance between the points, meaning we are able to maintain the quasi-isometry.

3.6 Conclusion

Computational complexity and sensitivity to outliers have been major issues with the standard persistence homology application in computational topology. Along with that, the lack of strong inverse properties has led to confusion on what geometric properties are retained and what is forgotten in persistence diagrams. All these issues are dealt with through the introduction of distributed persistence. It decreases the computation time for high-dimensional datasets because of its parallelizability and is more robust against outliers because it considers the subsets of the dataset instead of the whole dataset, and there might exist a subset that doesn't have any outlier in it. Also, it has strong inverse properties, meaning two metric spaces giving the same distributed persistence diagrams are bound to be isometries. In other words, it restricts the inverse image of persistence diagrams to isometries.

Some interesting problems on distributed persistence are presented in [Solomon 22]. Alongside all those, one can also try to get the inverse theorem for different types of filtrations like alpha complexes, Ĉech complexes, graph-induced complexes that are introduced in [Dey 21]. We will see one of the applications of distributed persistence in the next chapter. In this application, distributed persistence is used alongside the dimensionality reduction method to preserve the topology of the high-dimensional dataset in its low-dimensional embedding.

Chapter 4

DIPOLE

Dimensionality reduction methods are used to reduce the dimensions of any high-dimensional dataset for better interpretation of certain characteristics that we want to study. They work by preserving those characteristics of high-dimensional datasets when reducing the dimensions. Some dimensionality reduction methods are PCA, Isomap, and tSNE. PCA aims to preserve the variance, while Isomap aims to preserve the geodesic distances, and tSNE aims to preserve the clusters. DIPOLE is a dimensionality reduction post-processing step, that aims to preserve the topology of the high-dimensional dataset in its low-dimensional embedding. It takes in a certain initial embedding, that can be obtained by performing any dimensionality reduction method, and then it tries to preserve its topology, making it similar to that of the high-dimensional dataset from which the initial embedding was obtained. DIPOLE is based on strong theoretical properties of distributed persistence, and it is shown in [Wagner 21] that DIPOLE outperforms many conventional dimensionality reduction methods in terms of preserving topology, and many other conventional tests. We will see some results of [Wagner 21] in this chapter.

4.1 What is DIPOLE?

DIPOLE stands for DIstributed Persistence Optimized Local Embedding. It combines the techniques of metric geometry and distributed persistence in the form of gradient descent-based optimization, aiming to preserve the topology of the high-dimensional dataset in its low-dimensional embedding. It is a post-dimensionality reduction step meaning that we require an initialization embedding, which can be given by any dimensionality reduction method (in [Wagner 21] they have used Isomap). Once we have this initialization embedding, DIPOLE tries to correct the topology of this embedding with respect to its high-dimensional dataset (1). Alongside this, it also corrects the pairwise distances of points on a small subset in this embedding with respect to the pre-image of that subset (2). These two steps are required to satisfy the hypothesis of theorem 3.4.4, which will then result in the quasi-isometry between the high-dimensional dataset, and its low-dimensional embedding. We will state theorem 3.4.4 again here to explain its importance in defining DIPOLE pipeline.

Theorem 4.1.1. *Let X, and Y be metric spaces, and let $k > m = 1$.*
Then if $\varphi : X \longrightarrow Y$ is a surjection such that $\forall S \subseteq X$ with $|S| \in \{k, k-1\}$,

$$W_1(\lambda^1(S), \lambda^1(\varphi(S))) \leq \varepsilon_1 \tag{1}$$

and alongside this suppose $\exists X' \subset X$ of cardinality $(k-1)$ with

$$\sum_{(x_i,x_j)\in X'\times X'} |d_X(x_i,x_j) - d_Y(\varphi(x_i),\varphi(x_j))| \leq \varepsilon_2 \tag{2}$$

then φ is $56(k+1)\varepsilon_1 + 28\varepsilon_2$ quasi-isometry, i.e., $\forall x_1, x_2 \in X$

$$|d_X(x_1,x_2) - d_Y(\varphi(x_1),\varphi(x_2))| \leq 56(k+1)\varepsilon_1 + 28\varepsilon_2$$

In the above theorem, φ is a dimensionality reduction method that will give us initialization embedding $\varphi(X) = Y$, since φ is a surjection. Then based on equation (1), and equation (2), we will define two loss functionals. The loss functional based on equation (1) will be the global topological loss functional based on distributed persistence instead of standard persistence. The loss functional based on equation (2) will be loss functional based on local metric (i.e. metric geometry of some small subset of X and its image in Y). We will then try to minimize these two loss functionals by gradient descent. By minimizing these two loss functionals we are also minimizing the values of ε_1, and ε_2 in equation (1), and equation (2) respectively. This will in turn minimize the quasi-isometry distance between X, and Y, based on above theorem. This also means that if we are able to make values of ε_1, and ε_2 equal to zero, X and Y will become isometric.

4.2 DIPOLE Pipeline

Let (X, d_X) be a finite metric space, and we want to get its low-dimensional embedding while also preserving the topology. Suppose that the target dimension is $n \in \mathbb{N}$. Let $f : X \to \mathbb{R}^n$ be a mapping that sends elements from X to $\mathbb{R}^n$. Our aim is to optimize this f's embedding of X so that it preserves the topology of X, while also maintaining quasi-isometry with X. This can be achieved by defining two loss functionals, one local metric and another global topological based on the Inverse Theorem that we discussed, which are sufficiently differentiable to allow for gradient descent optimization.

4.2.1 Local Metric Regularizer

First, we will define local metric loss functional. This loss functional is based on equation (2). Let $P \subset X \times X$. This P is $X' \times X'$ from the theorem 4.1.1. It would be better if we consider pair of points in P to be very close to each other because equation (2) contains the summation of pairwise distances, and the smaller the subset the smaller the value of summation. This intuition leads to two of the following definitions of P:

D-1: For some $\delta > 0$ define $P = \{(x_i, x_j) \in X \times X \,|\, d_X(x_i, x_j) \leq \delta\}$.

D-2: For any $l \in \mathbb{N}$ define $P = \{(x_i, x_j) \in X \times X \,|\, x_i$ is x_j's l-nearest neighbor, and vice versa$\}$.

Now we can define **Local Metric Regularizer** functional as:

$$LMR_P(f) = \sum_{(x_1,x_2)\in P} (d_X(x_1,x_2) - ||f(x_1) - f(x_2)||)^2$$

4.2.2 Global Topological Loss Functional

We define **Global Topological Loss** functional based on distributed persistence, and it is as follows:

$$DP_k^m(f) = \frac{1}{\binom{|X|}{k}} \sum_{S\subset X,\, |S|=k} W_p^p(\lambda^m(S), \lambda^m(f(S)))$$

W_p is the p-Wasserstein distance between the distributed persistence diagrams, and W_p^p sums up all the p-Wasserstein distances among all the degrees that we are considering. Here, we have taken the average of all the Wasserstein distances between the persistence diagrams of all the possible subsets of fixed size to define our loss functional.

4.2.3 Net Loss

We define our final loss functional as:

$$\Phi(f_0) = \alpha LMR_p(f_0) + \frac{1-\alpha}{2} DP_k^m(f)$$

where α is the trade-off parameter.

In viewing the embedding f as a matrix of shape $n \times |X|$, this functional is differentiable almost everywhere concerning the entries of the matrix. Hence, gradient descent optimization can be applied. Stochastic gradient descent is preferred because of the large number of terms in the loss functional. In fact gradient descent on the functional Φ converges almost surely [Wagner 21].

4.2.4 The Pipeline

The pipeline is as follows:-

1. We get our initializing embedding f_0, then
2. We perform gradient descent optimization on the loss functional:

$$\Phi(f_0) = \alpha LMR_p(f_0) + \frac{1-\alpha}{2} DP_k^m(f_0)$$

3. This returns us the new embedding that is quasi-isometric to the original high-dimensional dataset, and also has a similar topology with the original high-dimensional dataset.

4.2.5 The Parameters

Default choices for DIPOLE hyperparameters are as follows:-

- For the initialization embedding authors of [Wagner 21] have used Isomap, for which the n-nearest neighbor graph has been set for $n = 5$.
- When input data is the point cloud in Euclidean space it is transformed into a distance matrix by taking geodesic distances from m-nearest neighbor graph, where $2 \leq m \leq 10$. This distance matrix is used for calculating the distributed persistence diagrams.
- Learning rate for gradient descent $lr \in \{0.1, 1\}$.
- Trade-off parameter $\alpha \in \{0.01, 0.1\}$.
- The l-nearest neighbors graph of points in X is used for setting P, where $2 \leq l \leq 5$.
- For Wasserstein distance we take $p = 2$.
- For gradient descent 2500 descent steps are run with annealing factor of $\frac{1000}{1000+step}$.

Target dimensions, and subset size, k, are user-defined parameters.

4.3 Experimental Results

In this section, we will show DIPOLE implementations on the following datasets:-

- Mammoth dataset taken from [smi 20].
- Dataset sampled from Brain Artery Tree taken from [Bullitt 05].

- And a dataset sampled from a Swiss-roll with a solid cylindrical section removed from it.

These results are mentioned in [Wagner 21], we will be presenting these results here, and performing qualitative and quantitative analysis on the above datasets. For quantitative analysis the following three tests are performed:-

1. **ijk-Test**
   - Test: Measure preservation of numerical order of pairwise distances between 3 points.
   - Points: x, y, z in high-dimensional dataset.
   - Let φ be the map that high-dimensional dataset to its low-dimensional embedding.
   - If $d(x,y) > d(y,z) \Rightarrow d_{\varphi}(\varphi(x), \varphi(y)) > d(\varphi(y), \varphi(z))$ then $Z = 1$, else $Z = 0$.
   - Calculate Z for 1000 samples.
   - Measure: $1 - E[Z]$.

2. **Residual Variance Test**
   - Test: Measures the unexpressed variance in low-dimensional embedding.
   - Evaluate the success of Isomap vs. PCA, MDS.
   - Calculate residual variance as $1 - R^2$.
   - R: Pearson correlation coefficient between distance matrices (d_H, d_L) of high and low-dimensional datasets.
   - Treat distance matrices as mean-centered vectors and measure the cosine of the angle between them.
   - Lower residual variance indicates more successful dimensionality reduction.

3. **Global Persistence Test**
   - Test involves computing Wasserstein distance of persistence diagrams.
   - Applied to high-dimensional dataset and its low-dimensional embedding.
   - Due to computational cost, use farthest point sampling to obtain subsets of size 256.
   - Compute Rips persistence on subsets.
   - Calculate Wasserstein distance between persistence diagrams.

- This test computes the distance between both zero-dimensional and one-dimensional persistence diagrams.

The first two tests are the common tests that were used to test the performance of dimensionality reduction methods like PCA, and Isomap. The residual variance test was used in the first paper that introduced [Tenenbaum 00] Isomap to test the performance of Isomap.

After performing the above tests with different parameter combinations we will get a table similar to the table 4.3.

Table 4.1: Test scores for different parameter combinations.

target_dim	lmr_edges	alpha	k	lr	ijk	residual variance	globalh0	globalh1
2	3	0.01	32	0.1	0.0754	0.07601823	211.3758653	47.71194012
2	5	0.01	32	0.1	0.0762	0.075810123	205.5937559	51.60334119
2	3	0.1	32	0.1	0.0755	0.075728166	214.3416324	48.42901669
2	5	0.1	32	0.1	0.0744	0.075480934	208.9097285	52.79834364
2	3	0.01	64	0.1	0.0734	0.075383581	194.7403123	49.65537223
2	5	0.01	64	0.1	0.0772	0.075335344	198.6620336	48.66771107

The scores in the above table will be represented by a histogram. Each score will be ranged in the X-axis, while parameter combination (i.e. each row will have a 1-unit value in the Y-axis). For example, consider the first row of the table 4.3, and consider the score of ijk-test, then this row will be represented by the cuboid of the height of 1-unit in the Y-axis, and lie over the range of 0.0 to 0.1.

4.3.1 Visualization Results

From these visualizations: 4.3.1, 4.3.1, and 4.3.1 , we can conclude that DIPOLE is able to maintain the topology of the datasets in their low-dimensional embedding better than the other dimensionality reduction methods that were used. The local metric regularizer is degenerate version of DIPOLE with $\alpha = 1$ (and also l is increased to 10) in the net loss 4.2.3. That is we are only considering local metric loss functional during the optimization. Local metric regularizer visualization is shown because we need to show the importance of minimizing both the loss functional in order to maintain the topology of high-dimensional datasets in their low-dimensional embeddings. It is clear from these visualizations that if we try to minimize the local metric loss functional only, then the low-dimensional embedding is not able to capture as much topology of high dimensional dataset as it is able to when we minimize both local metric and global topological loss functionals. In both mammoth and brain artery tree datasets, the appendages are not visible in the local metric regularizer visualizations, while they are visible in DIPOLE visualizations.

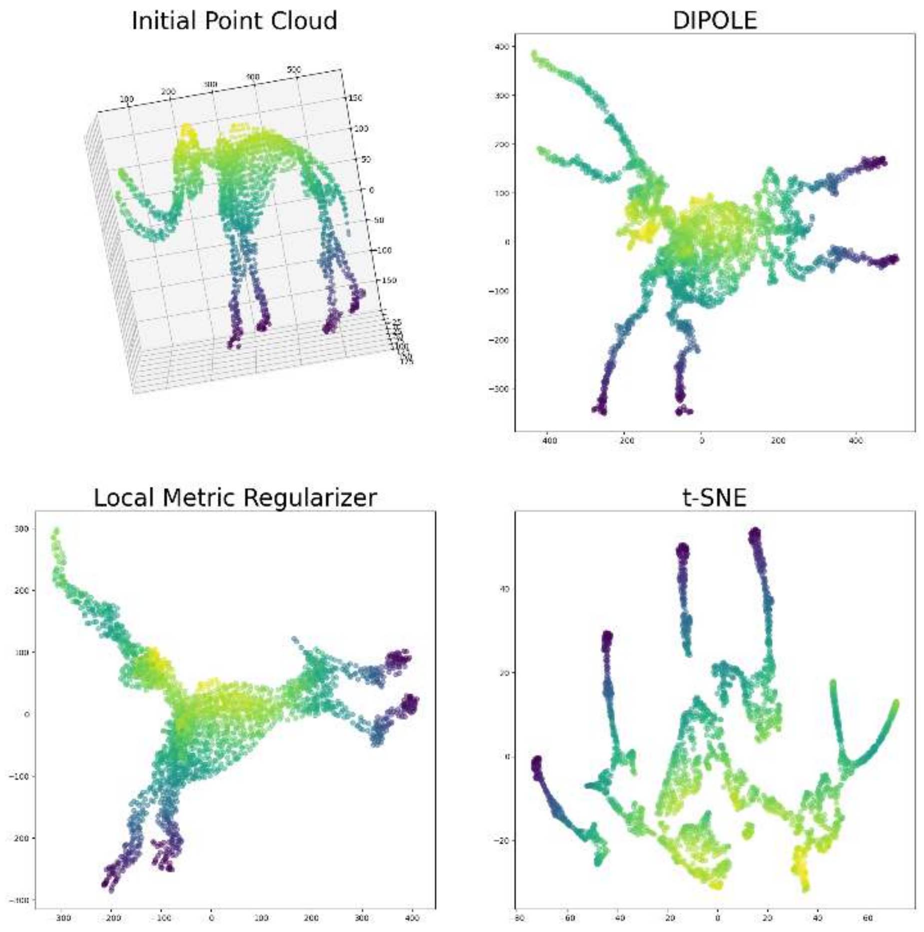


Figure 4.1: Mammoth Dataset

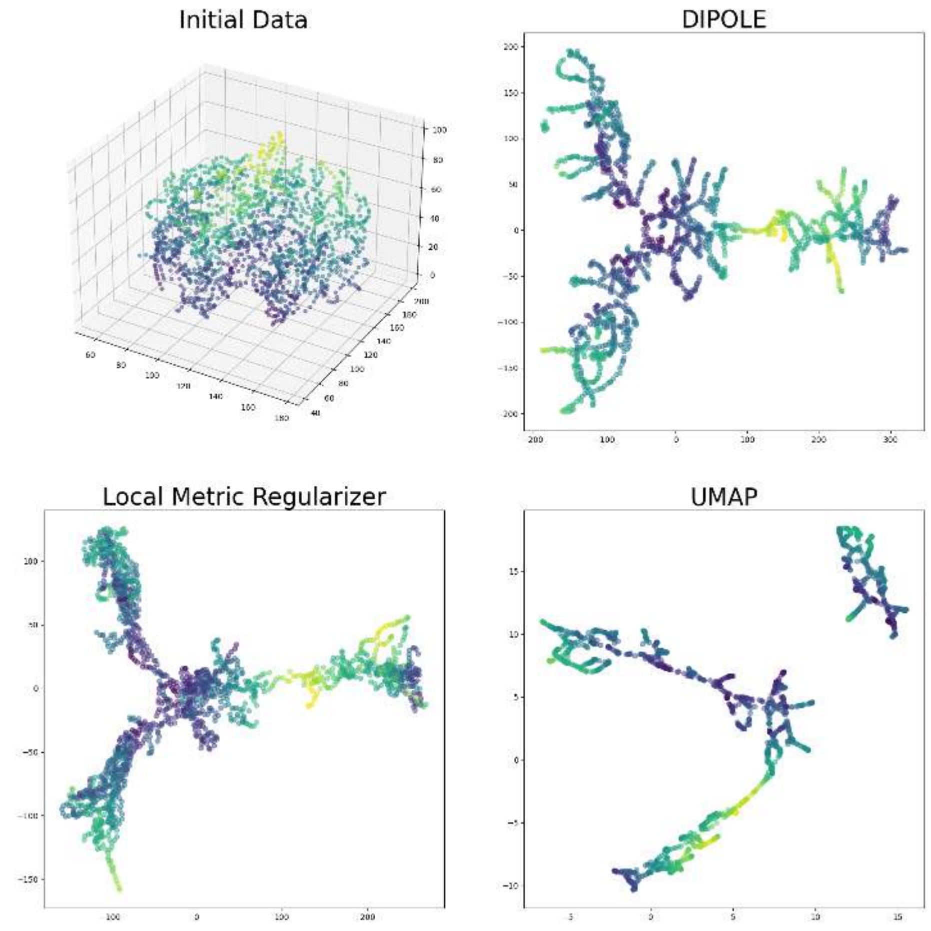


Figure 4.2: Brain Artery Tree Dataset

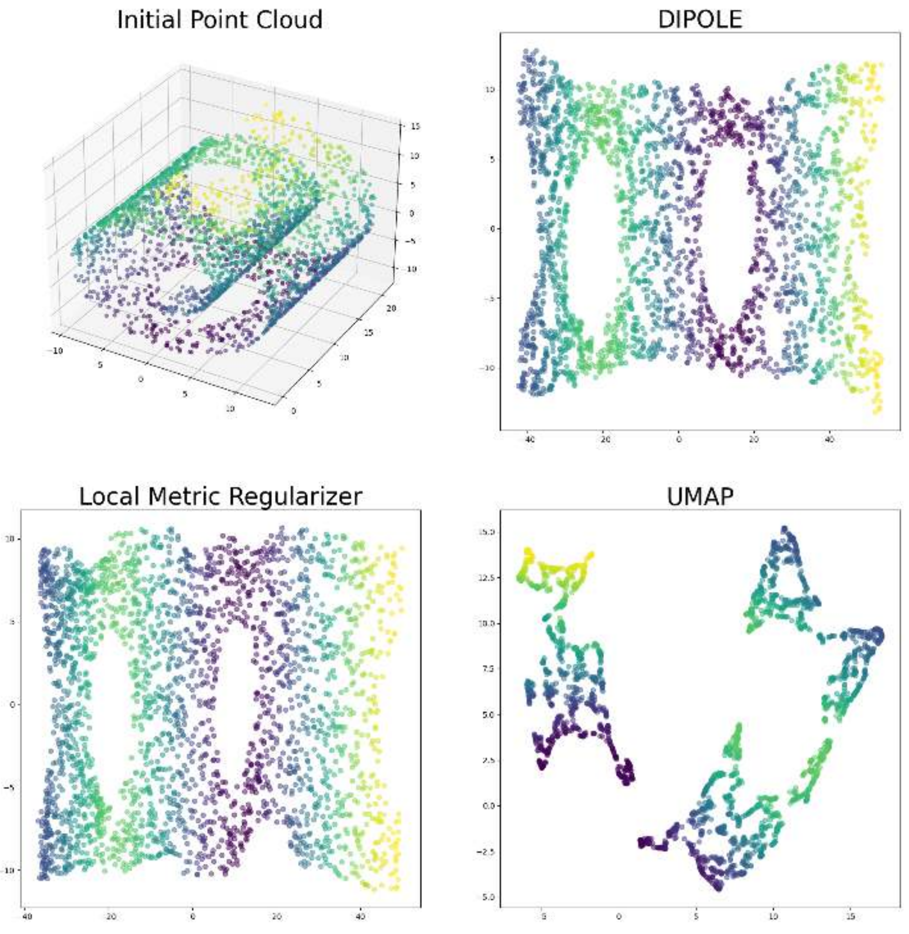


Figure 4.3: Swiss-roll Dataset

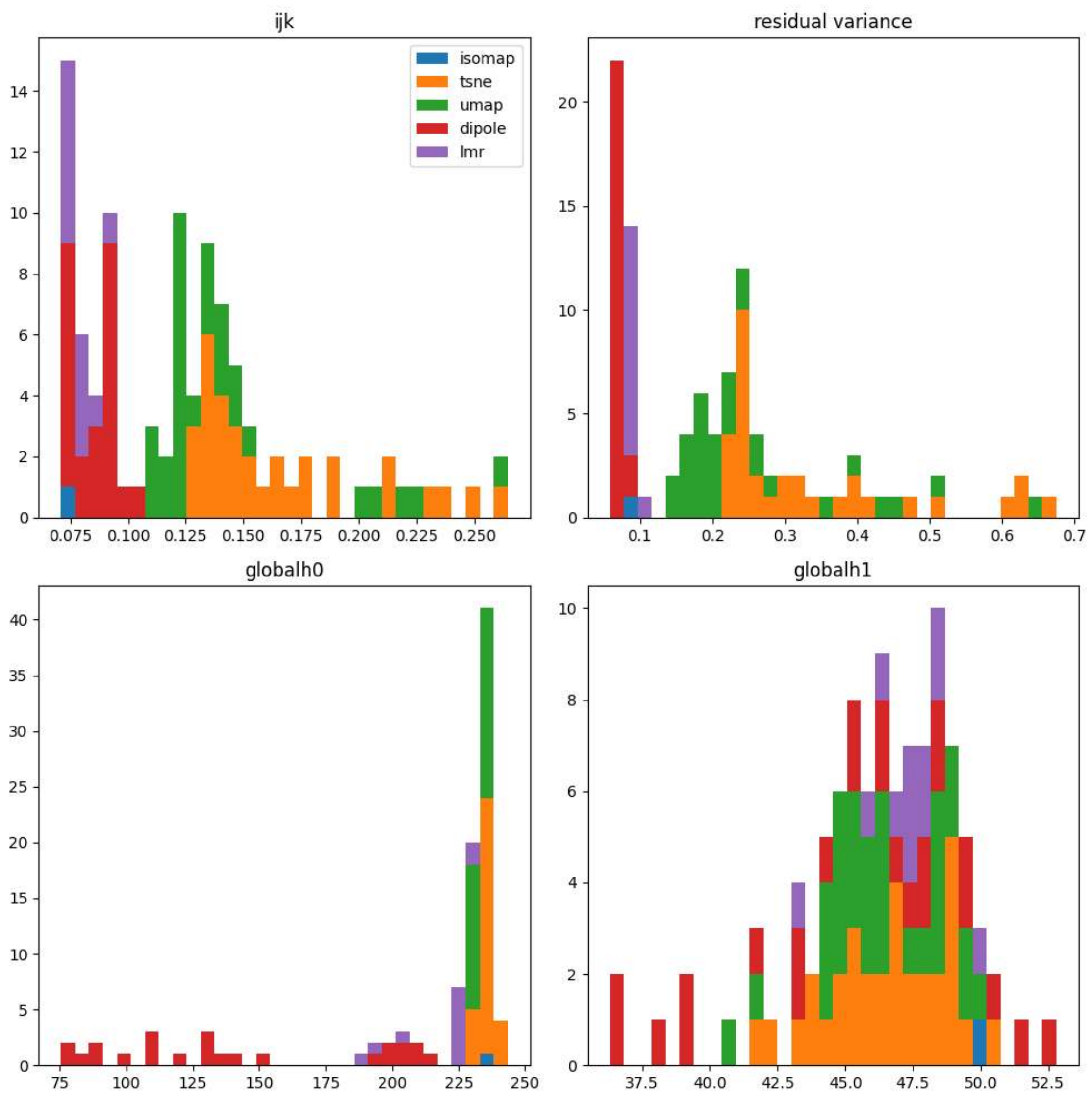


Figure 4.4: Mammoth Dataset: Metric Analysis

4.3.2 Quantitative Results

The tests are conducted in such a way that the lower the score, the better the performance. It can be seen from the histograms [4.4, 4.5, 4.6] showing scores for different parametric combinations that DIPOLE performs on par, and better than many conventional dimensionality reduction methods in tests such as ijk-test, and residual variance tests. DIPOLE shows significantly better results than other dimensionality reduction methods when it comes to preserving the topology.

Readers interested in comparing the scores in more details for different parameter combinations can access the original DIPOLE code from `https://github.com/aywagner/DIPOLE.git`.

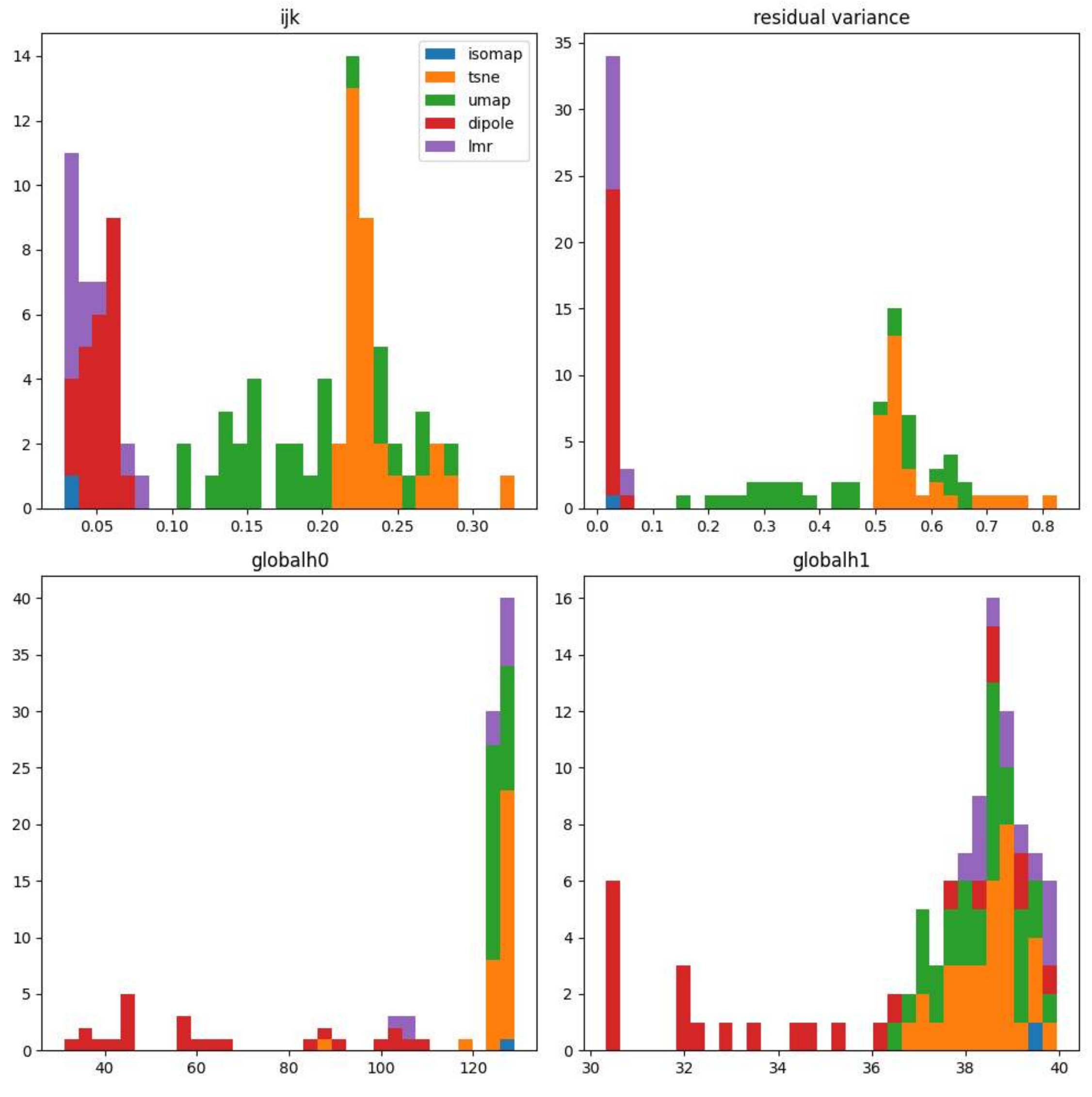


Figure 4.5: Brain Artery Tree Dataset: Metric Analysis

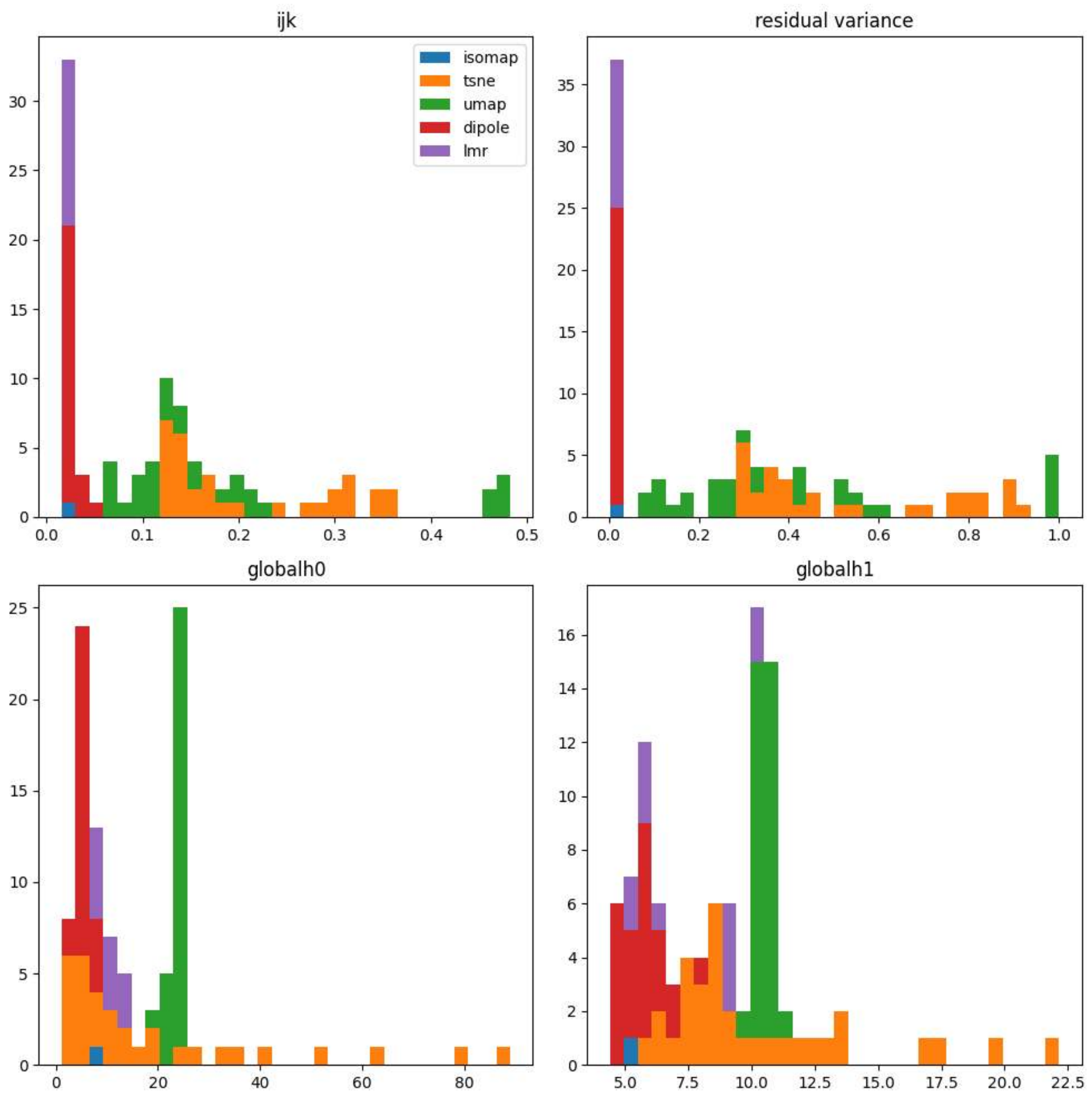


Figure 4.6: Swiss-roll Dataset: Metric Analysis

4.4 Conclusion

Distributed persistence introduced in the previous chapter is used in dimensionality reduction to give the DIPOLE pipeline. The inverse properties of distributed persistence not only help in maintaining the topology of high-dimensional data while performing dimensionality reduction, but they also help in maintaining the quasi-isometry between the high-dimensional dataset, and its low-dimensional embedding. Distributed persistence also gives an added advantage of being scalable, and robust against outliers, and these properties make the DIPOLE dimensionality reduction methods even more efficient method.

The DIPOLE pipeline is very flexible and can be subjected to many modifications and improvements. At the moment we can preserve only zero, and one-dimensional homological features while performing the dimensionality reduction method. Trying to preserve the additional two-dimensional features will increase the time complexity. But with efficient methods, we might be able to tackle this issue. Also, one can try to come up with the inverse theorems for different simplicial complexes, and then use those theorems to define a DIPOLE pipeline that uses a different simplicial complex other than the Rips complex that we used.

Chapter 5

Special Orthogonal Group Implementation

DIPOLE is a novel dimensionality reduction method that can be used to preserve the topology of high-dimensional datasets in their low-dimensional embeddings. It performs gradient descent optimization on distributed persistence-based loss functional and a local metric loss functional to achieve this objective. Distributed persistence, not standard persistence, is used because distributed persistence presents the advantage of being scalable, robust against outliers, and having some nice inverse properties. Due to these inverse properties, the high-dimensional dataset and its low-dimensional embedding will have similar topology and will also be quasi-isometric. Refer to chapter 4 for further details on DIPOLE.

In this chapter, we perform DIPOLE dimensionality reduction on certain datasets and verify the status of DIPOLE as a topology-preserving dimensionality reduction method.

5.1 Special Orthogonal Group: $SO(2)$

In the following section, we will show DIPOLE implementation on a dataset sampled from the special orthogonal group. Then we will explain the metric analysis that we have done to compare different dimensionality reduction methods for this dataset. In the end, we will give the results of metric analysis. This implementation verifies the topology-preserving nature of the DIPOLE dimensionality reduction method.

5.1.1 $SO(2)$ Group

A general linear group over the space $\mathbb{R}$ is defined as $GL_n(\mathbb{R}) = \{A \in M_n(\mathbb{R}) | \exists B$, and $AB = I\}$. Note that here $M_n(\mathbb{R})$ represents a space containing all $n \times n$ square matrices. Therefore, a general linear group can be considered as a group that contains all the invertible $n \times n$ matrices over $\mathbb{R}$. The orthogonal group $O_n(\mathbb{R}) \subseteq GL_n(\mathbb{R})$ is defined as $O_n(\mathbb{R}) = \{A \in GL_n(\mathbb{R}) \mid A \cdot A^T = A^T \cdot A = I\}$. This means $det(A^T \cdot A) = det(A \cdot A^T) = det(I) = 1$. This further implies $det(A) = \pm 1$. Now, if we further restrict our determinant to $+1$, we get a subgroup of $O_n(\mathbb{R})$ called a special orthogonal subgroup.

Special Orthogonal Group is a group that contains $n \times n$ rotation matrices. These rotation matrices perform rotation transformation on n-dimensional vectors in $\mathbb{R}^n$ in the anti-clockwise direction. The special orthogonal group is denoted by $SO_n(\mathbb{R})$ or for simplicity $SO(n)$. For this particular experiment, we have restricted ourselves to $n = 2$.

Every $A \in SO(2)$ can also be represented as

$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{5.1}$$

This can be understood as follows: Whenever we rotate a vector in $\mathbb{R}^2$, we always rotate it by a certain angle, and that angle here is θ. Also, note that $2\pi + \theta$ will give the same rotation as θ. Therefore, we can parameterize every matrix in $SO(2)$ with respect to this θ. Also, since we only need one parameter to represent matrices in $SO(2)$, we can consider the dimension of $SO(2)$ to be equal to 1. In fact, it can be easily shown that $(SO(2), \cdot)$ is isomorphic to $(S^1, \cdot)$. Here $S_1 = \{z \in \mathbb{C} : |z| = 1\}$, or simply said, S^1 is a circle in $\mathbb{R}^2$. S^1 is also called a one-dimensional circle because it is a one-dimensional manifold that is embedded in $\mathbb{R}^2$.

Therefore, it can be understood that if we perform a dimensionality reduction method on $SO(2)$ group from 4 dimensions to 2 dimensions, while also preserving the topology of the group (i.e. $\cong S^1$), then our lower 2-dimensional embedding should represent a circle.

5.1.2 Experiment

We generated this dataset by varying sampling different θ values from $[0, 2\pi)$, and putting this value in A from 5.1. We performed three dimensionality reduction methods on this dataset:-

1. t-SNE with target dimension equal to 2, and perplexity equal to 30.
2. UMAP with target dimension equal to 2, nearest neighbors equal to 15, and minimum distance equal to 0.1.
3. DIPOLE (using Isomap initialization embedding) with target dimension equal to 2, l equal to 3, α equal to 0.1, k equal to 64, and m equal to 10.

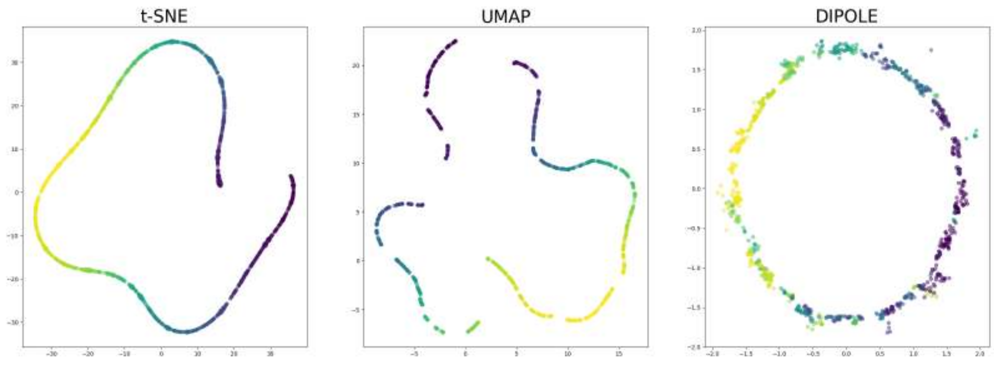


Figure 5.1: t-SNE, UMAP, and DIPOLE implementation on dataset sampled from $SO(2)$ group.

The visualization results are shown in figure 5.1. It can be seen that DIPOLE is able to maintain the topology of a circle when reducing the dimensions to 2. The metric analysis in figure 5.2 also shows that DIPOLE, while performing on par with other dimensionality reduction methods in the ijk-test, and residual variance test, outperforms other dimensionality reduction methods when it comes to preserving the topology.

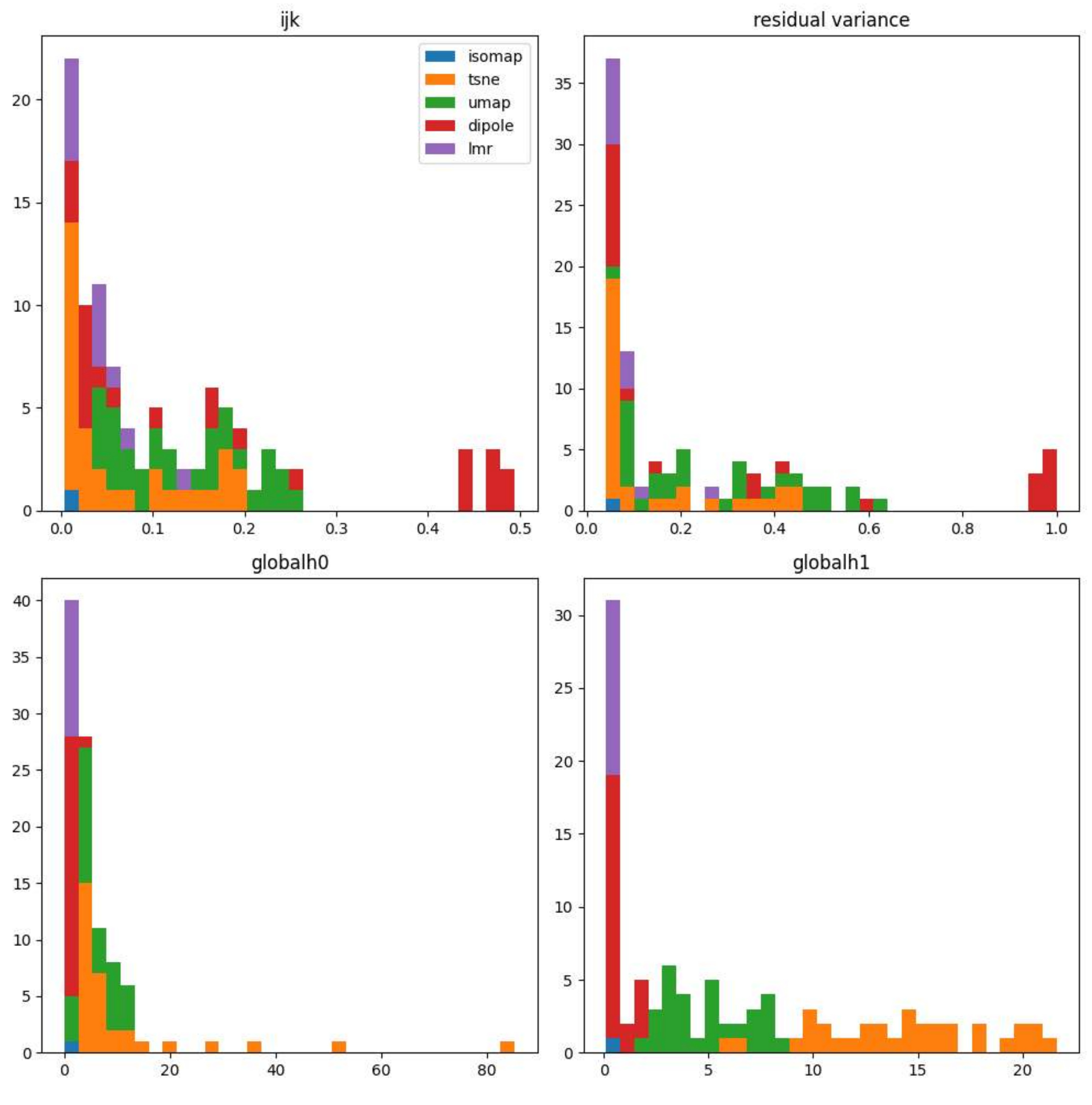


Figure 5.2: $SO(2)$ Group: Metric Analysis

Chapter 6

DIPOLE + tSNE

DIPOLE needs an initialization embedding; it then creates the loss functional that we discussed in Chapter 4. After that, it performs gradient descent optimization on that loss functional, minimizing it, and giving us a low-dimensional embedding that is topologically similar and quasi-isometric to the high-dimensional dataset. The authors of [Wagner 21] chose Isomap to give the initialization embedding because Isomap can help satisfy certain conditions required in Theorem 4.1.1. In fact, we can use any dimensionality reduction method to get an initialization embedding because the conditions of Theorem 4.1.1 are bound to be satisfied when minimizing the loss functional. This presents an advantage because this way we can get a low-dimensional embedding, with initialization embedding obtained from any dimensionality reduction method, that is also true in terms of topology to its high-dimensional dataset. And also, it is quasi-isometric. Since different dimensionality reduction methods aim to preserve different characteristics in low-dimensional embedding, this will ensure that we are able to preserve those characteristics 'as much as possible,' while also ensuring similar topology and maintaining a quasi-isometry.

In the following chapter, we will be using t-SNE dimensionality reduction method to obtain our initialization embedding and then perform the optimization through DIPOLE to get our final low-dimensional embeddings. We will try to see how DIPOLE can improve the topology for different initialization embeddings (specifically, t-SNE).

6.1 Cylinder$\times 4$

In this section, we will show how DIPOLE can help improve the topology of low-dimensional embeddings we get with a dimensionality reduction method other than Isomap. We will be considering t-SNE. First, we will introduce the dataset on which we will implement these dimensionality reduction methods.

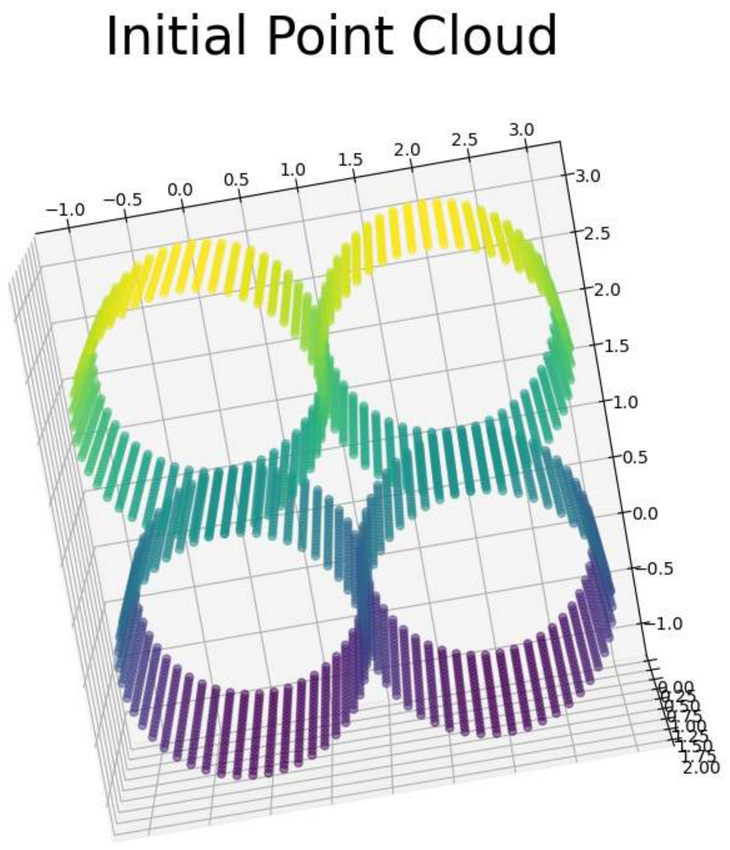


Figure 6.1: Point cloud sampled from Cylinder$\times$4 dataset.

6.1.1 Cylinder$\times$4 Dataset

This dataset consists of four cylinders that are touching each other laterally. See Figure 6.1 to see the point cloud sampled from this dataset. Each cylinder is of radius 1 unit and height 2 units, and they are arranged in such a way that they touch each other's sides. Looking at the figure 6.1, one can conclude that there are five one-dimensional holes in the point cloud. If we perform dimensionality reduction on this point cloud, from three dimensions to two dimensions, aiming to preserve the topology of the high-dimensional dataset, the low-dimensional embedding must contain these five one-dimensional holes.

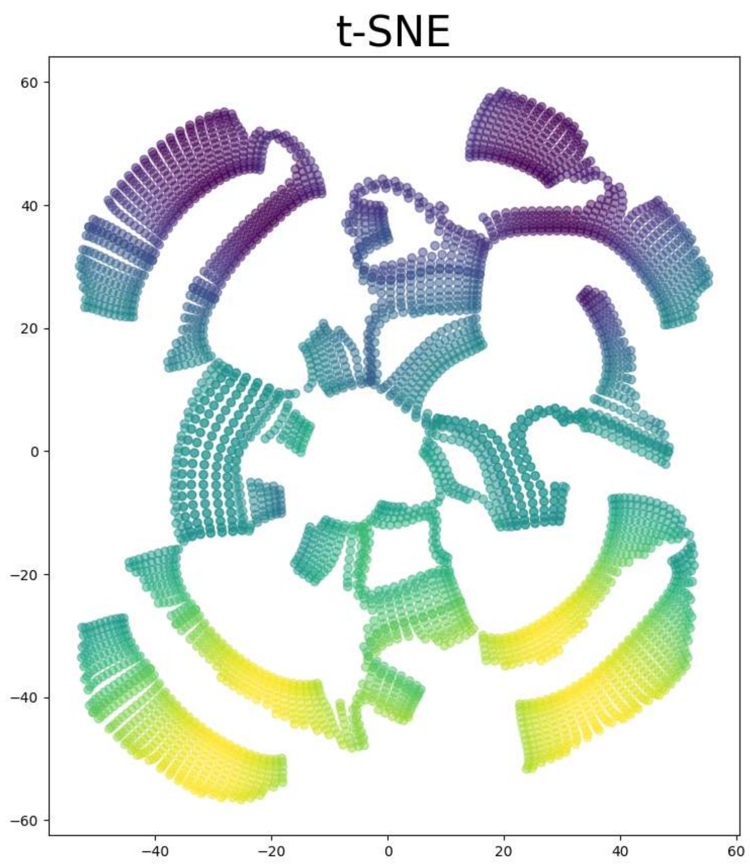


Figure 6.2: 2-dimensional embedding of Cylinder×4 obtained through t-SNE.

6.1.2 Experiment

First, we perform t-SNE dimensionality reduction on the point cloud sampled from the Cylinder×4 dataset. We keep the perplexity equal to 30, and target dimensions equal to 2. We get the low-dimensional embedding as shown in Figure 6.2. It can be seen that the five one-dimensional holes that we discussed in the previous section weren't preserved in this low-dimensional embedding. Now, we take the point cloud obtained in Figure 6.2 as our initialization embedding for DIPOLE. After passing through the DIPOLE pipeline, we obtain the embedding shown in Figure 6.3. All the five one-dimensional holes are preserved in this embedding.

It's clear from the figures that DIPOLE helps in improving the topology of the embedding obtained through t-SNE. In fact, DIPOLE would help improve the topology of the embedding obtained through any dimensionality reduction method as per [Wagner 21]. The metric analysis for this case is shown in Figure 6.4.

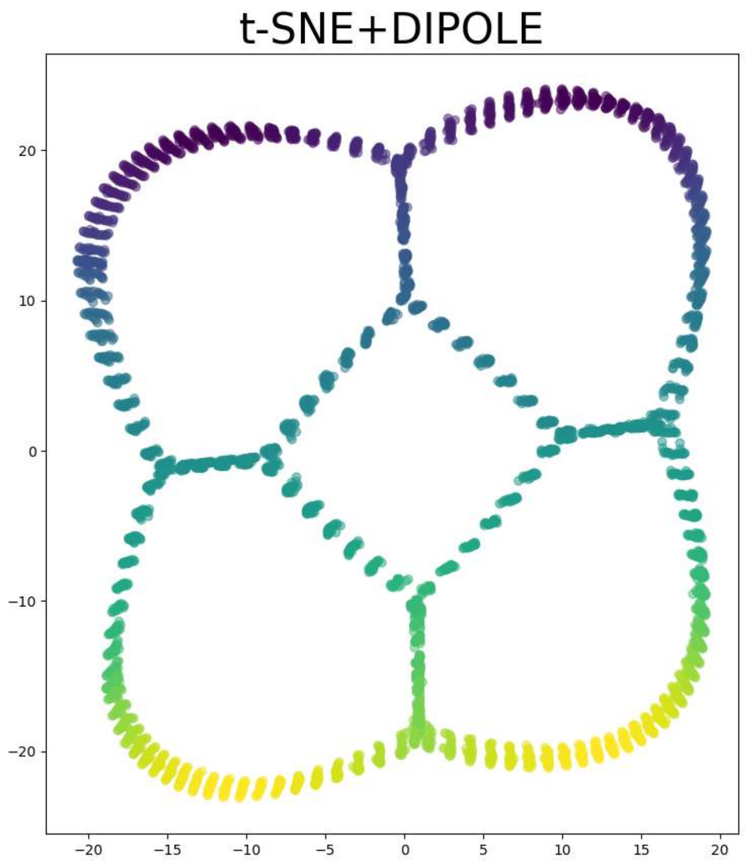


Figure 6.3: 2-dimensional embedding of Cylinder×4 obtained through t-SNE + DIPOLE.

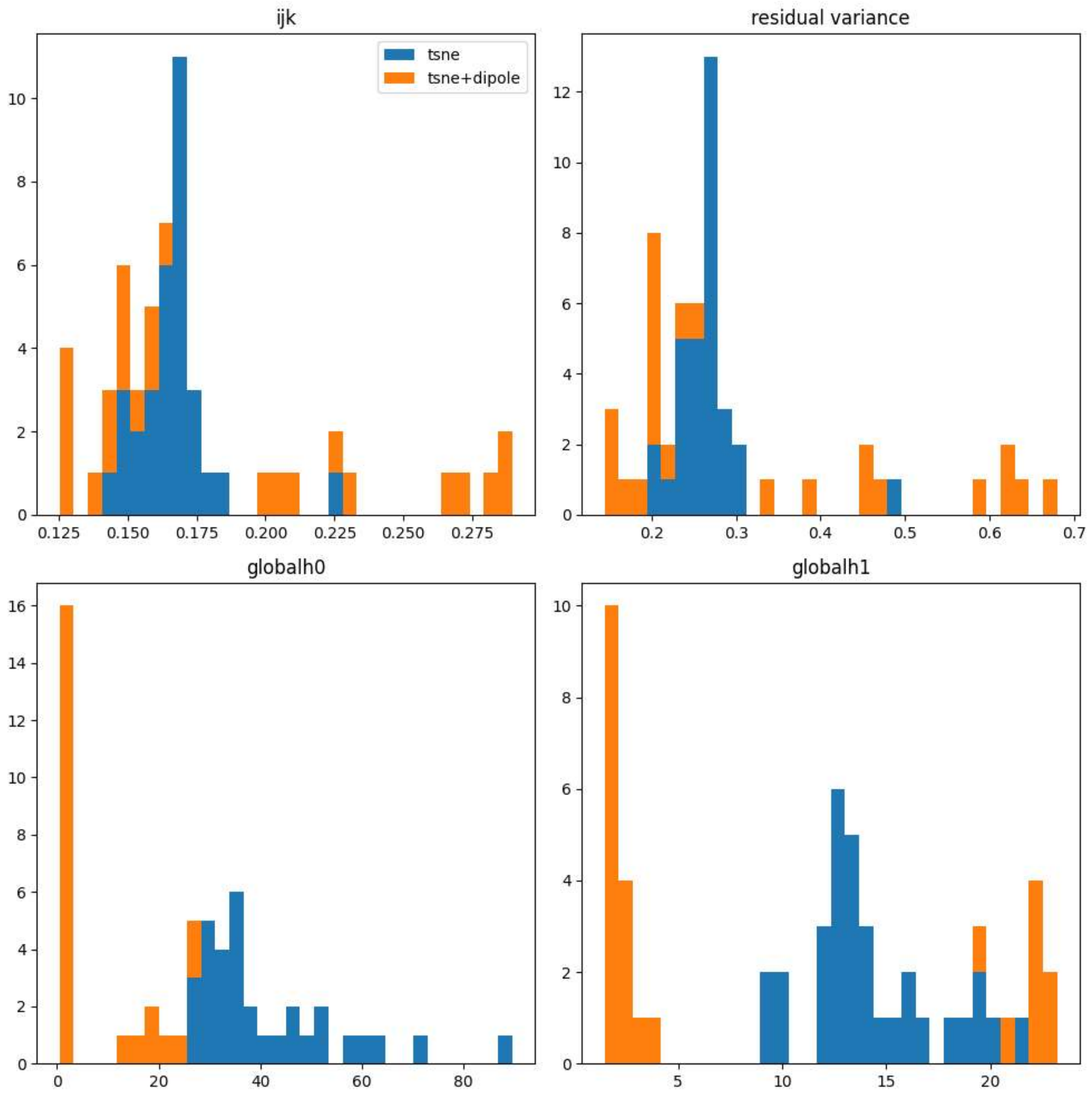


Figure 6.4: Cylinder×4: Metric Analysis

6.2 $S^1 \times S^1$

In the following section, we will implement t-SNE + DIPOLE on a classical topological space, $S^1 \times S^1$, that is also called a torus.

6.2.1 $S^1 \times S^1$ Dataset

The product space of two one-dimensional circles denoted as $S^1 \times S^1$ is one of the fundamental topological spaces. And it is homeomorphic to a torus.

When we take the product of two such circles, $S^1 \times S^1$, we essentially create a space where each point is uniquely identified by an ordered pair (x,y), where x and y are points on the two individual circles, respectively. In simpler terms, it's like taking a sheet of graph paper and identifying each point by its row and column.

Now, let's understand why $S^1 \times S^1$ is homeomorphic to a torus. Imagine taking a doughnut shape, which is essentially a torus, and imagining it as made up of infinitely many circles, both around its central hole and around its outer edge. If you consider the coordinates of each point on the surface of the torus, they can be represented by two angles: one representing the rotation around the hole and the other representing the rotation around the center. These angles correspond to the points on the two circles in the product space $S^1 \times S^1$.

In conclusion, $S^1 \times S^1$ represents a space where each point is identified by an ordered pair of points from two individual circles, and it is homeomorphic to a torus due to the geometric similarity between the two structures.

Since $S^1 \times S^1$ lies in the ambient space of 4 dimensions and it is homeomorphic to a torus that lies in 3-dimensional space. Performing dimensionality reduction on the dataset sampled from $S^1 \times S^1$ from 4 dimensions to 3 dimensions must result in an embedding that has a shape similar to that of a torus when we are aiming to preserve the topology as in DIPOLE. In fact, performing Isomap results in a torus-shaped embedding, as shown in Figure 6.5.

DIPOLE, at the moment, can only preserve the zero and one-dimensional features; therefore, using DIPOLE as a dimensionality reduction method on $S^1 \times S^1$ from 4 dimensions to 3 dimensions will not result in a torus-shaped embedding. Since introducing two-dimensional features (that will represent the void of the torus) alongside zero and one-dimensional features will increase the time complexity, Isomap can be preferred over DIPOLE in this case. Have a look at Figure 6.6 to see how DIPOLE (with initialization embedding obtained through Isomap) performs in this case (we haven't considered two-dimensional homological features in this case). We get a cylindrical-shaped embedding,

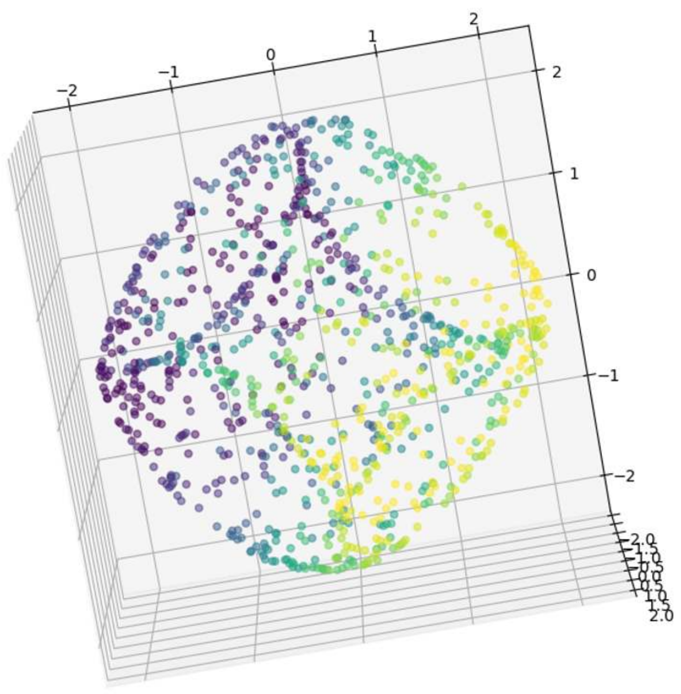


Figure 6.5: $S^1 \times S^1$ with Isomap, from 4-dimensions to 3-dimensions.

which is basically a torus that has been cut along its bigger loop.

6.2.2 Experiment

We generated this dataset by first forming two datasets sampled from the same circle. This will give two datasets with two columns in each dataset. Then we combined those two datasets to get a final dataset in which the first two columns come from the first dataset, and the next two columns come from the second dataset. Due to the reasons discussed in the previous section, we performed DIPOLE (with initialization embedding obtained through t-SNE) dimensionality reduction from 4 dimensions to 2 dimensions. Visualization results are shown in Figures 6.7 and 6.8. Metric analysis is shown in Figure 6.9.

6.3 MNIST

In the following section, we will implement t-SNE + DIPOLE on one of the most standard datasets, MNIST Handwritten Digits.

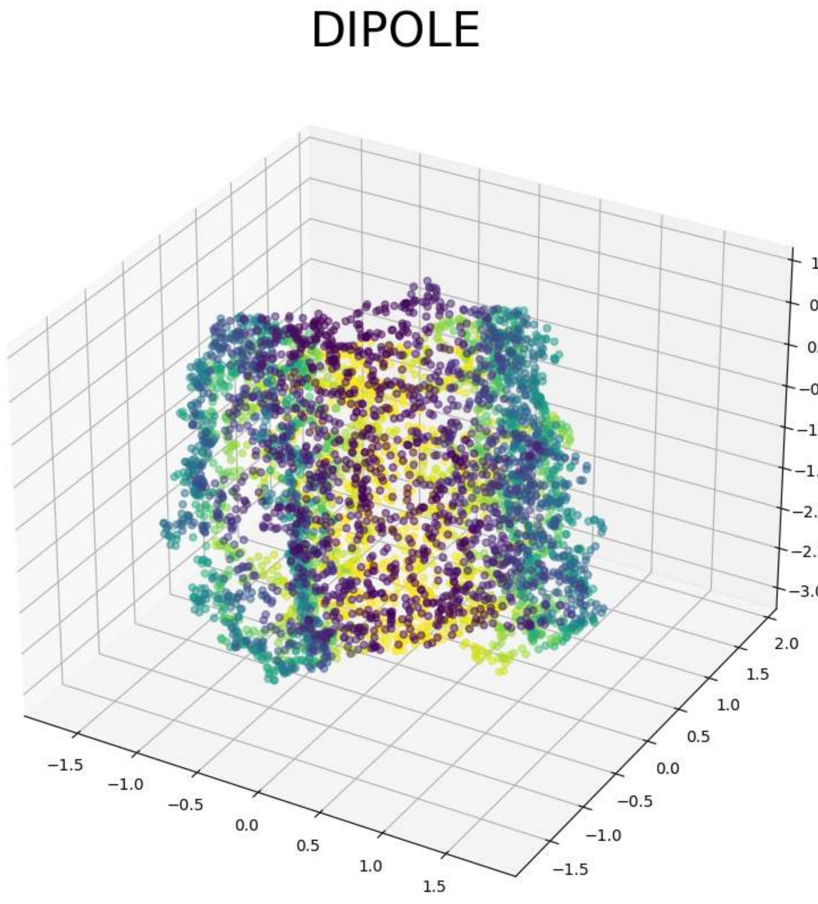


Figure 6.6: $S^1 \times S^1$ with DIPOLE, from 4-dimensions to 3-dimensions.

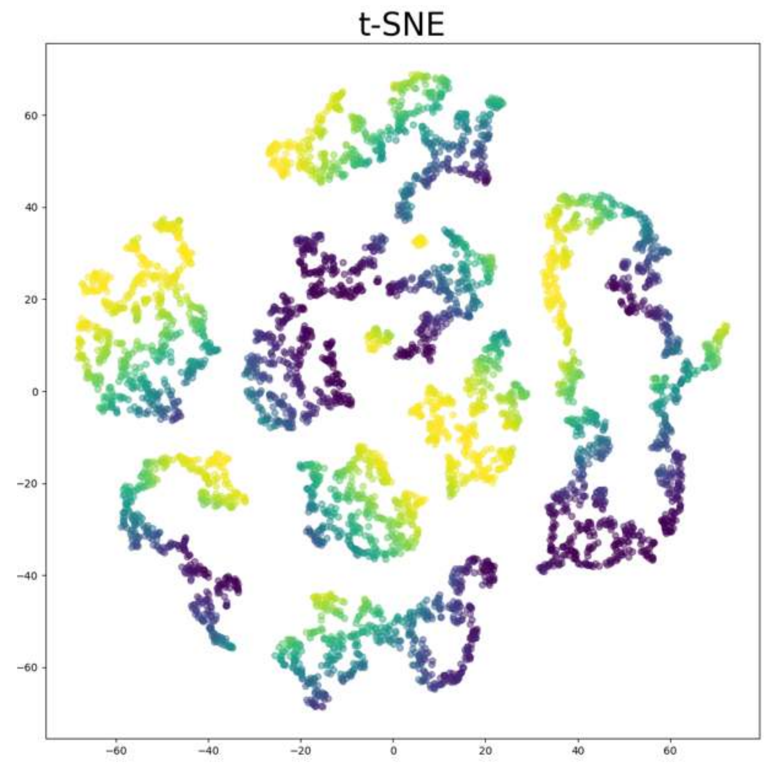


Figure 6.7: 2-dimensional embedding of $S^1 \times S^1$ obtained through t-SNE.

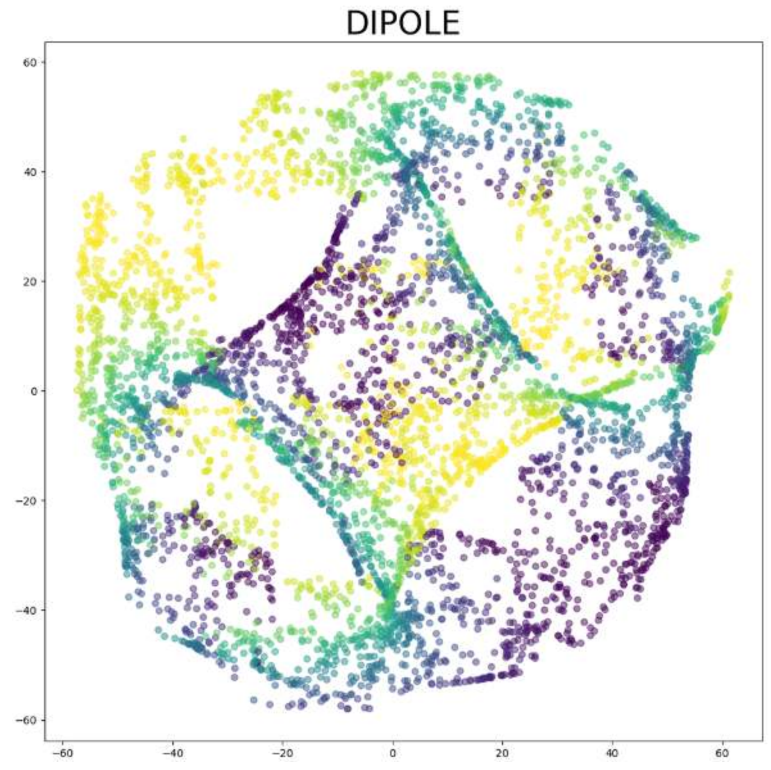


Figure 6.8: 2-dimensional embedding of $S^1 \times S^1$ obtained through t-SNE + DIPOLE.

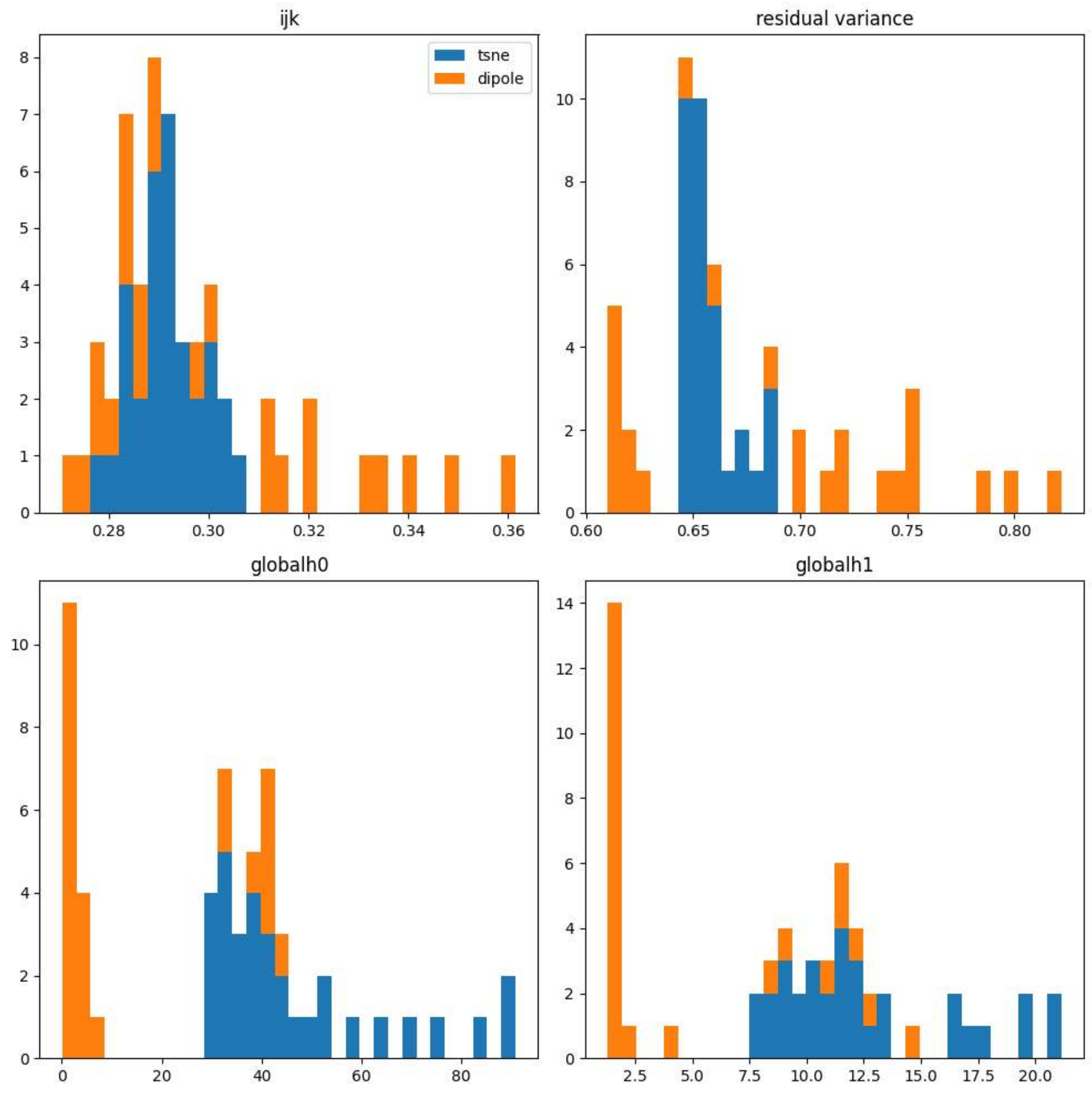


Figure 6.9: $S^1 \times S^1$: Metric Analysis

6.3.1 MNIST Handwritten Digits Dataset

The MNIST dataset is a collection of grayscale images showing handwritten digits from 0 to 9, each in a 28x28 pixel format. It's widely used in machine learning and computer vision to test algorithms. MNIST helps researchers understand how well their models can recognize handwritten characters. In addition to its utility in algorithm testing, the MNIST dataset serves as an educational resource for aspiring machine learning practitioners. Its straightforward structure and clear labeling make it an ideal starting point for beginners to grasp fundamental concepts such as data preprocessing, model building, and evaluation.

For experiment purposes, we have sampled 6000 points from the dataset and performed a dimensionality reduction method on this sampled dataset.

6.3.2 Experiment

We have shown dimensionality reduction visualization results on this dataset, with t-SNE method, and t-SNE + DIPOLE method. Numbers and the color they are represented by are shown below:-
0: Blue
1: Orange
2: Green
3: Red
4: Purple
5: Brown
6: Pink
7: Gray
8: Olive
9: Cyan

Look at Figure 6.10 to see the visualization results. Look at Figure 6.11 for the metric analysis. One thing we can note from the metric analysis is that DIPOLE is not able to make much impact on the low-dimensional initialization embedding.

6.4 SPHERES

In this section we perform t-SNE, and t-SNE + DIPOLE on the SPHERES dataset that was used in [Moor 21] to test the performance of Topological Autoencoder, a dimensionality reduction method aiming to preserve topology based on autoencoder.

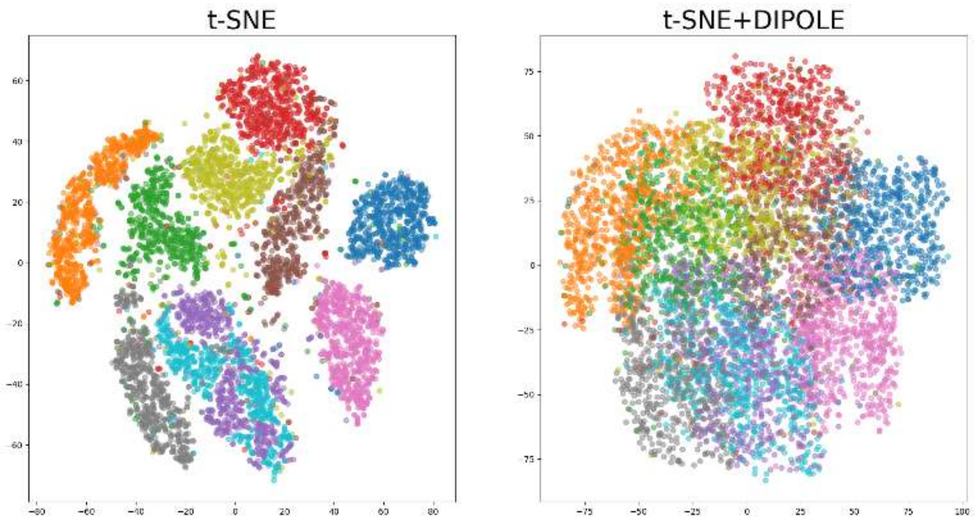


Figure 6.10: Dimensionality reduction performed on MNIST dataset from 784 to 2 dimensions: t-SNE, and t-SNE + DIPOLE

6.4.1 SPHERES Dataset

This dataset consists of ten 100-dimensional spheres enclosed by a larger 100-dimensional sphere. We perform dimensionality reduction methods, namely t-SNE, and t-SNE + DIPOLE to see if the nesting property of the space is maintained in the lower dimensions. In [Moor 21] only TopoAE was able to maintain this property as you can see in the figure 6.12. We will be performing t-SNE, and t-SNE + DIPOLE with the similar objective of maintaining this nesting property.

6.4.2 Experiment

Look at the figure 6.13 to see the visualizations. One can see that while t-SNE tears the original data apart by forming clusters of points obtained from the larger sphere around the smaller sphere, t-SNE + DIPOLE spreads out these clusters giving the impression that smaller spheres lie inside the larger sphere.

6.5 Original Datasets

In this section we will perform t-SNE, and t-SNE + DIPOLE on the original datasets that were used in [Wagner 21], namely Mammoth Dataset, Brain Artery Tree Dataset, and Swisshole Dataset.

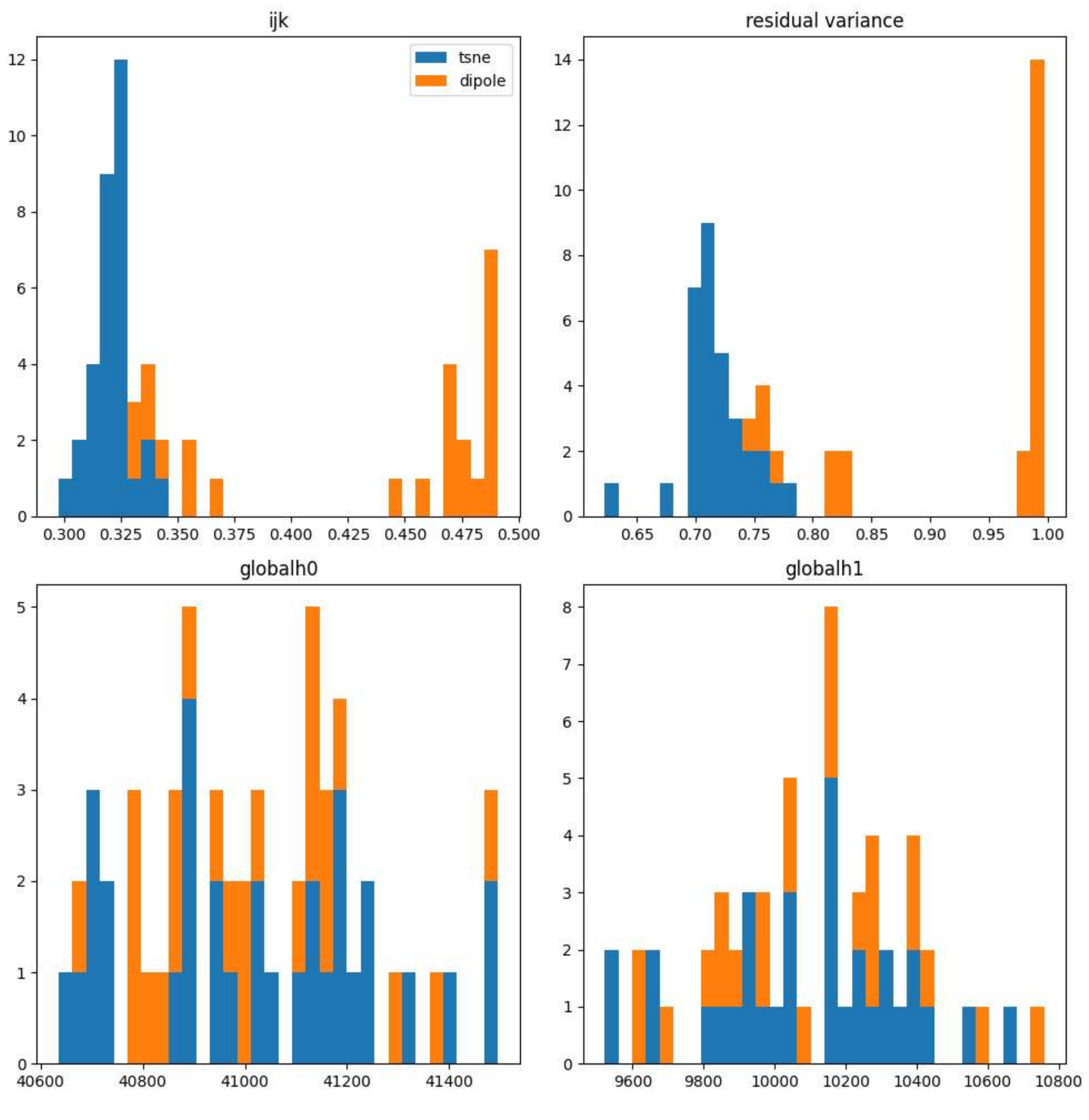


Figure 6.11: MNIST Dataset: Metric Analysis

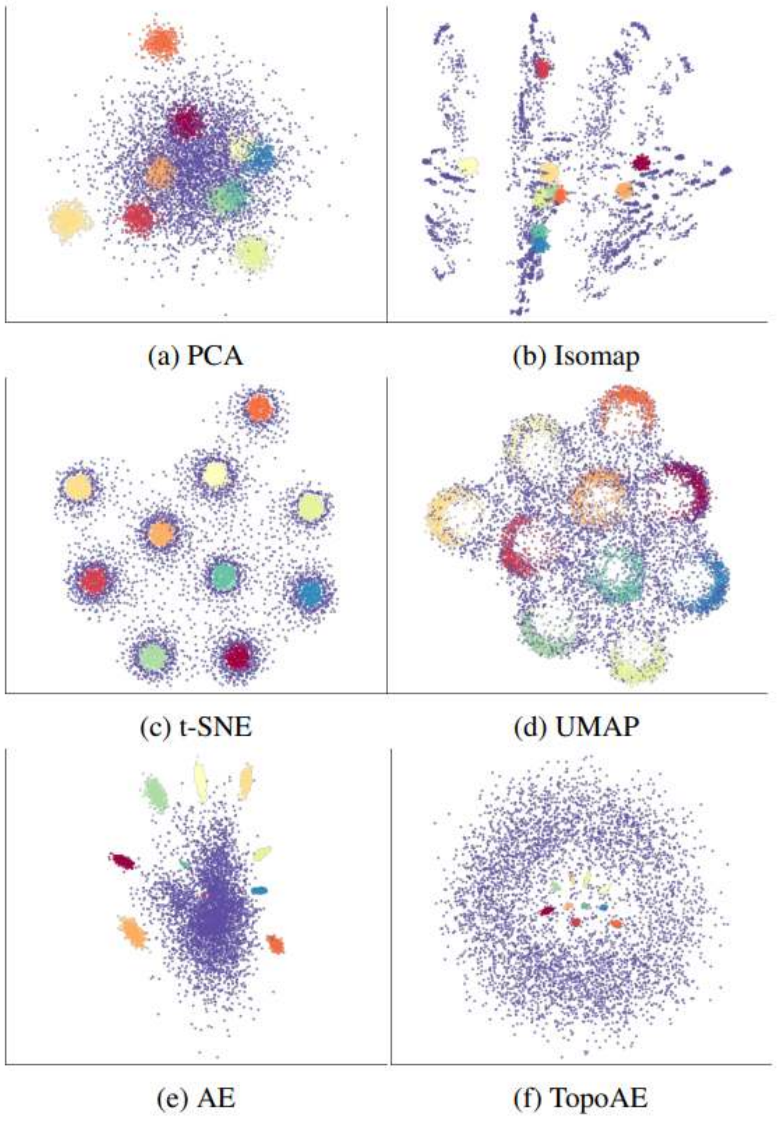


Figure 6.12: TopoAE introduced in [Moor 21] is able to maintain the nesting property of the dataset.

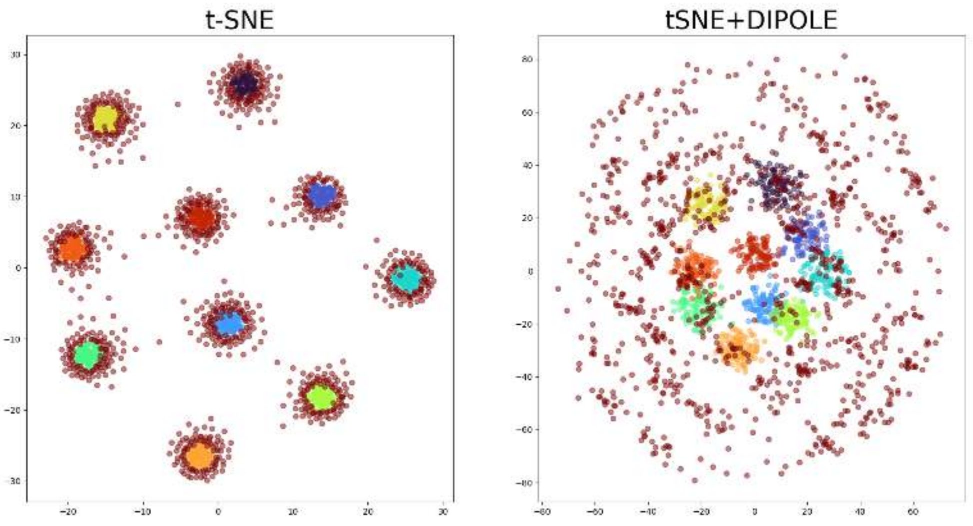


Figure 6.13: t-SNE + DIPOLE is able to maintain the nesting property of the dataset.

6.5.1 Datasets

The following are the datasets used for this experiment:-

- Mammoth dataset taken from [smi 20].
- Dataset sampled from Brain Artery Tree taken from [Bullitt 05].
- And a dataset sampled from a swiss-role with a solid cylindrical section removed from it.

6.5.2 Experiment

We perform t-SNE, and t-SNE + DIPOLE on the above datasets. The visualization results and their respective metric analysis are shown in the figures below. In visualization results, it can be seen that t-SNE + DIPOLE gives an embedding that is similar in topology (connectivity) to the high-dimensional dataset. While metric results don't give a very nice bigger picture when it comes to ijk-test, and residual variance test.

6.6 Conclusion

From all the experiments shown in this chapter one can conclude that DIPOLE can be used as a post dimensionality reduction method with t-SNE too. That is, t-SNE can also provide initialization embedding, and then DIPOLE can help improve the topology of that initialization embedding, and make it similar to that of high-dimensional data. In fact

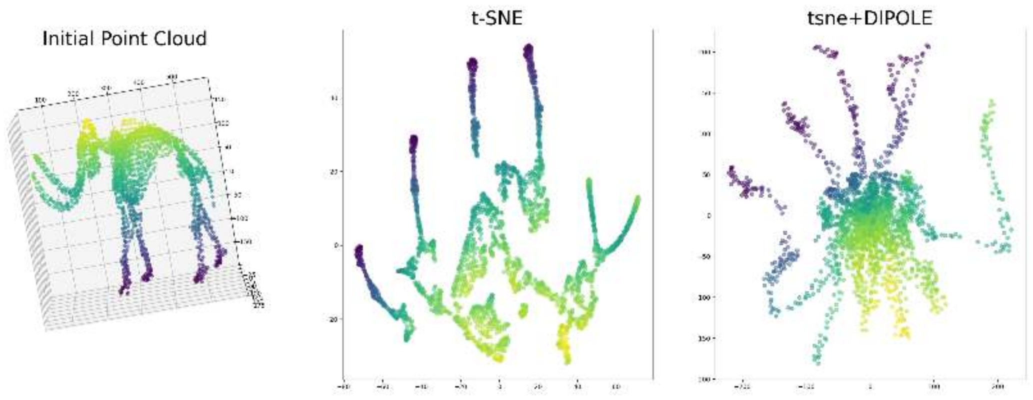


Figure 6.14: Mammoth Dataset: Visualization

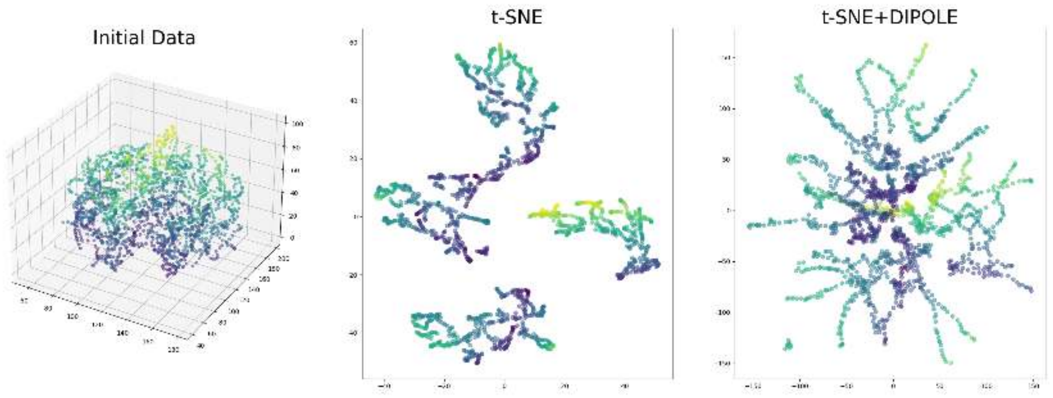


Figure 6.15: Brain Artery Tree Dataset: Visualization

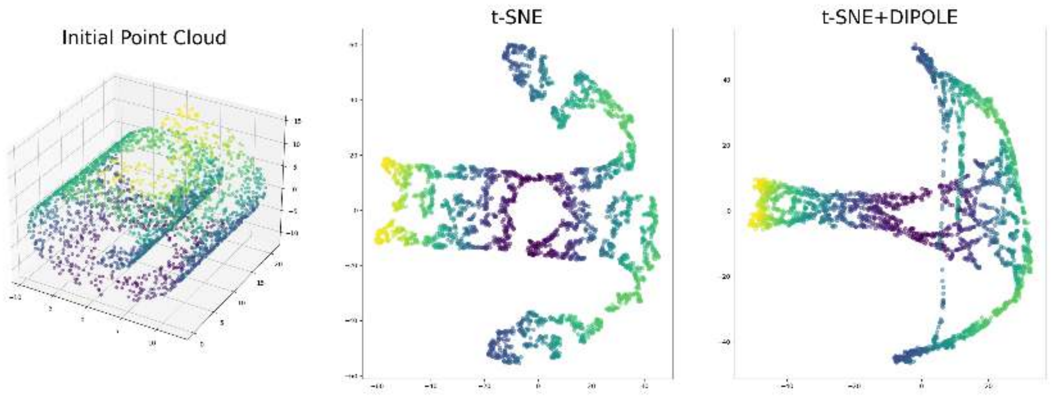


Figure 6.16: Swisshole Dataset: Visualization

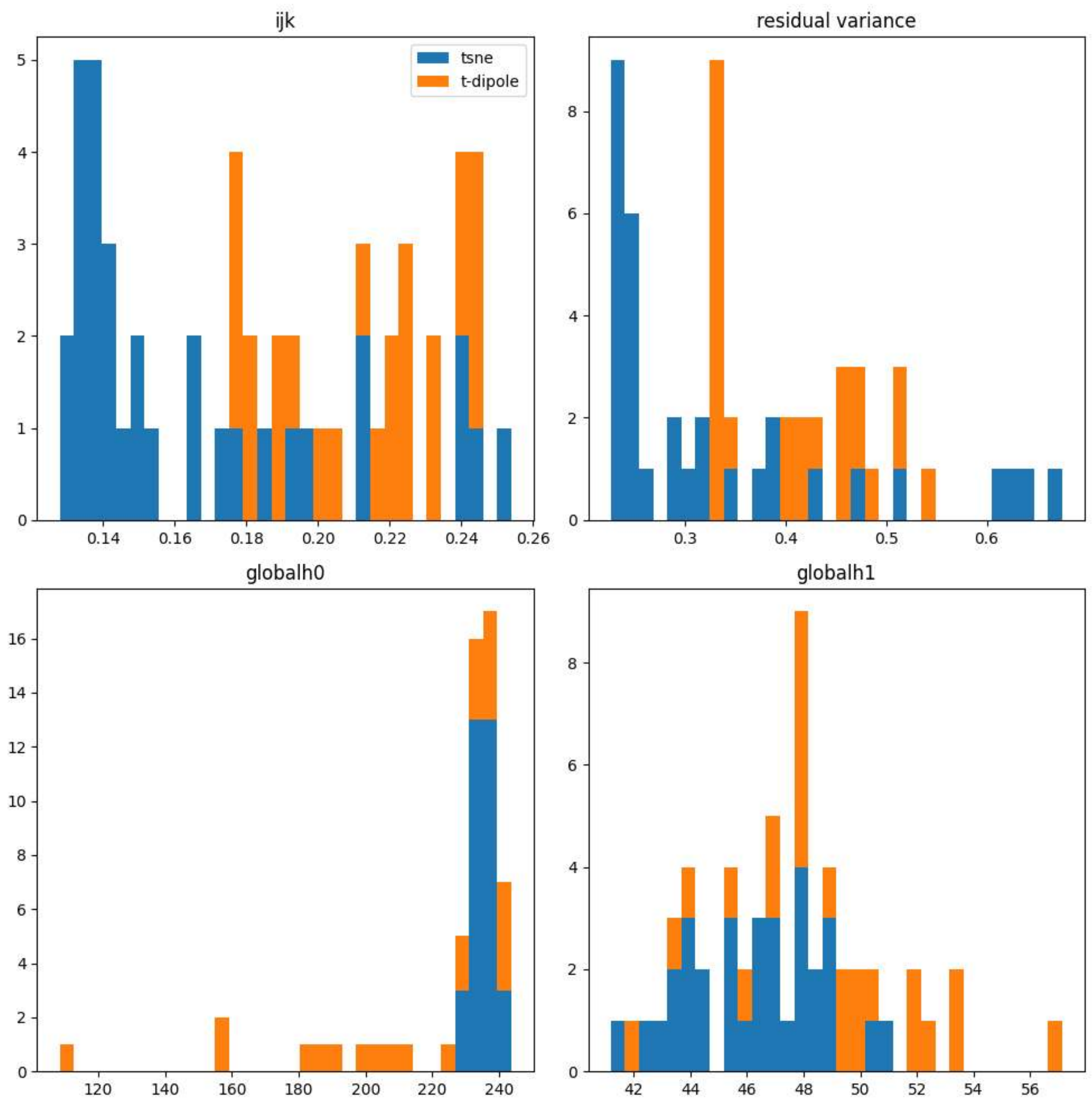


Figure 6.17: Mammoth Dataset: Metric Analysis

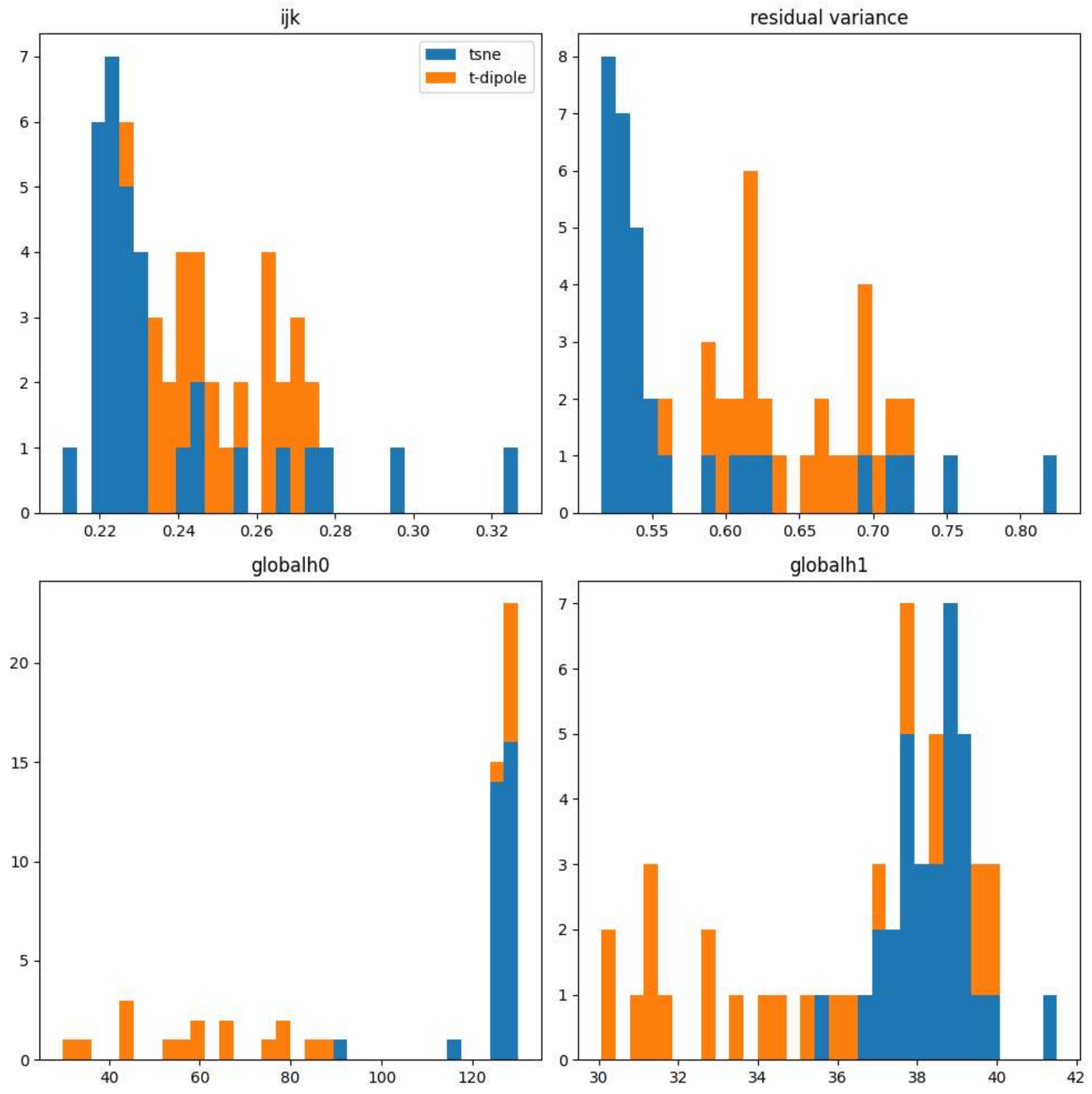


Figure 6.18: Brain Artery Tree Dataset: Metric Analysis

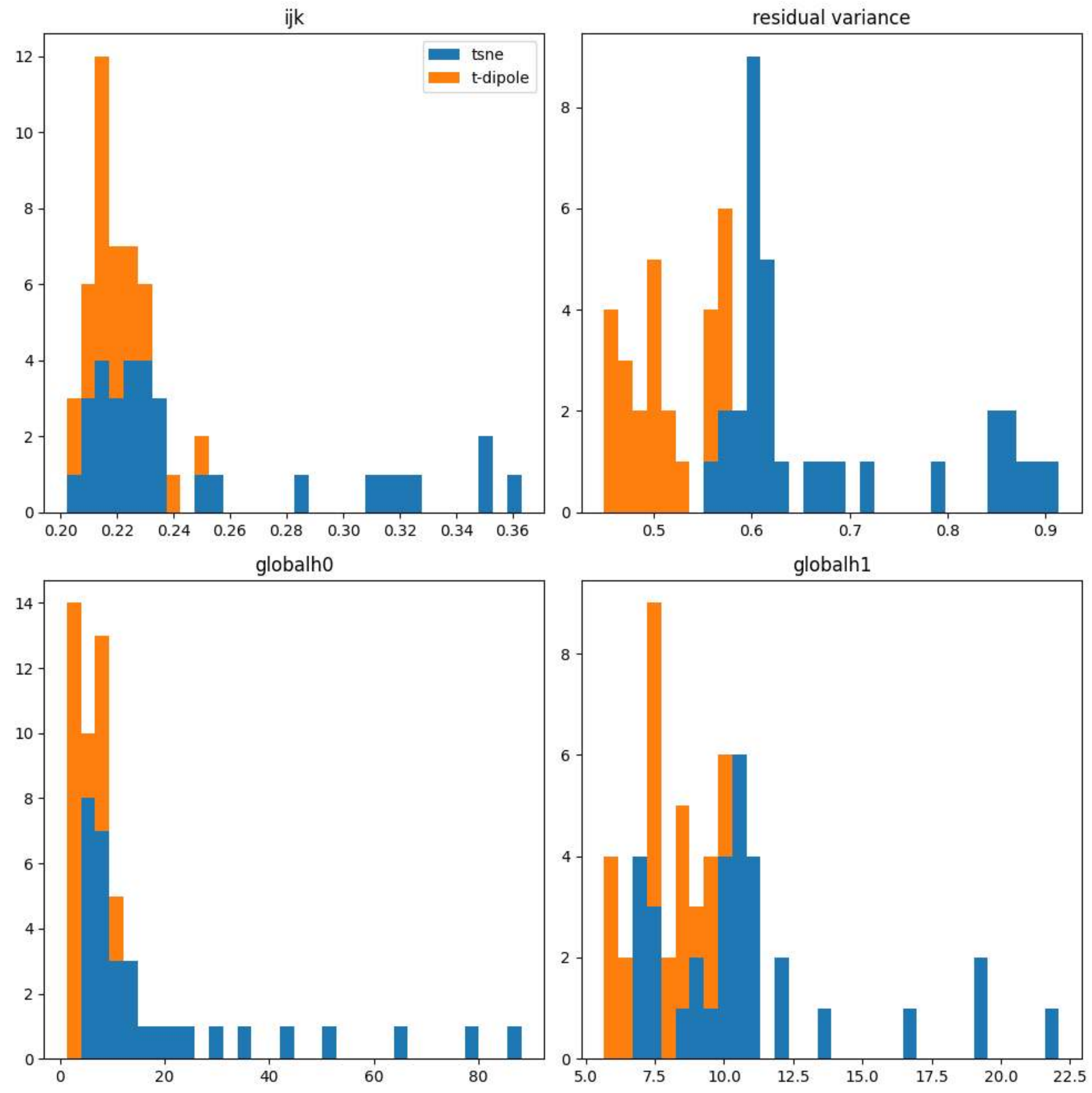


Figure 6.19: Swisshole Dataset: Metric Analysis

DIPOLE can take initialization embeddings from any dimensionality reduction method as per [Wagner 21]. This gives an advantage of preserving the characteristics that different dimensionality reduction methods aim to preserve, while also maintaining the topology and quasi-isometry.

One thing to note is the case of $S^1 \times S^1$, where Isomap presents a better way to preserve the topology of a four-dimensional dataset sampled $S^1 \times S^1$, since with DIPOLE we will need to consider two-dimensional homological features, and it will increase the time complexity, while with Isomap nothing as such is required. DIPOLE still needs modification, and improvements if it has to deal with such problems. This is one of the limitations of DIPOLE, we won't be discussing it much. In the next chapter, we will see further limitations that come with DIPOLE, and how we have tried to deal with those limitations.

Chapter 7

DIPOLE Limitations

DIPOLE presents us with a dimensionality reduction pipeline that helps in preserving the topology of the high-dimensional dataset in its low-dimensional embedding. It also provides an additional advantage of maintaining the quasi-isometry between the high-dimensional dataset and its respective low-dimensional embedding. Inverse properties also ensure that the inverse image of a respective distributed persistence diagram is restricted to a set of isometries, unlike the case shown in Figure 3.1. Distributed persistence also offers the advantages that come with sub-sampling, such as reduced computational complexities and robustness against outliers.

However, with all these advantages that DIPOLE presents, there also exist certain limitations, and those limitations will be discussed in this chapter. We won't be discussing the limitation that we introduced in the previous chapter, namely that computational complexity is bound to increase if we include 2-dimensional homological features to be also preserved. We might be able to deal with this limitation by making the code more efficient.

7.1 LIMITATION-1: DIPOLE is sensitive to noisy features.

The presence of noise might disrupt the performance of DIPOLE since DIPOLE will consider that noise when defining the loss functional and then minimizing it. One thing to note is that we are not talking about some specific noise because outliers also act as noise, but DIPOLE is robust against outliers due to the use of distributed persistence. We are considering the noise in general, and we will discuss one specific case of such noise here, namely noisy features, which are features (or dimensions) that contain irrelevant information with no particular pattern.

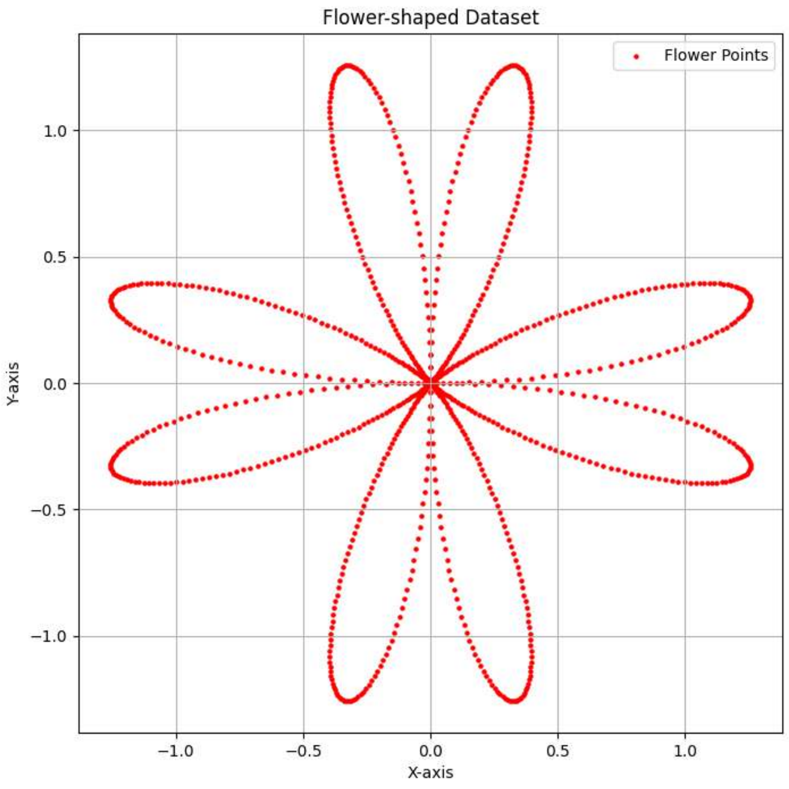


Figure 7.1: Step-1

7.1.1 Dataset-1: Flower in 3D

This dataset is generated by first sampling the dataset in 2 dimensions from the structure of a flower, then adding a third random variable (uniformly sampled from $[0,1]$) to each point to increase the dimensions to three. Have a look at Figures 7.1 and 7.2 to see how this dataset is generated. If you look at Figure 7.2 from the top, you will see the shape of the flower as shown in Figure 7.1. What we have done is we have added an extra feature that is irrelevant to the dataset and is acting as noise. We then perform dimensionality reduction with target dimension $= 2$.

We perform two dimensionality reduction methods on the above-defined dataset. The first one is Isomap, and the second one is DIPOLE, which takes the initialization embedding obtained through Isomap. We then compare the results obtained. Results are shown in Figures 7.3 and 7.4.

Although DIPOLE does its job of preserving the topology and maintaining the quasi-isometry, the final embedding can be affected by the presence of noisy features in its high-dimensional dataset since DIPOLE relies on optimization of loss functionals based on a high-dimensional dataset and its low-dimensional embedding. If the high-dimensional

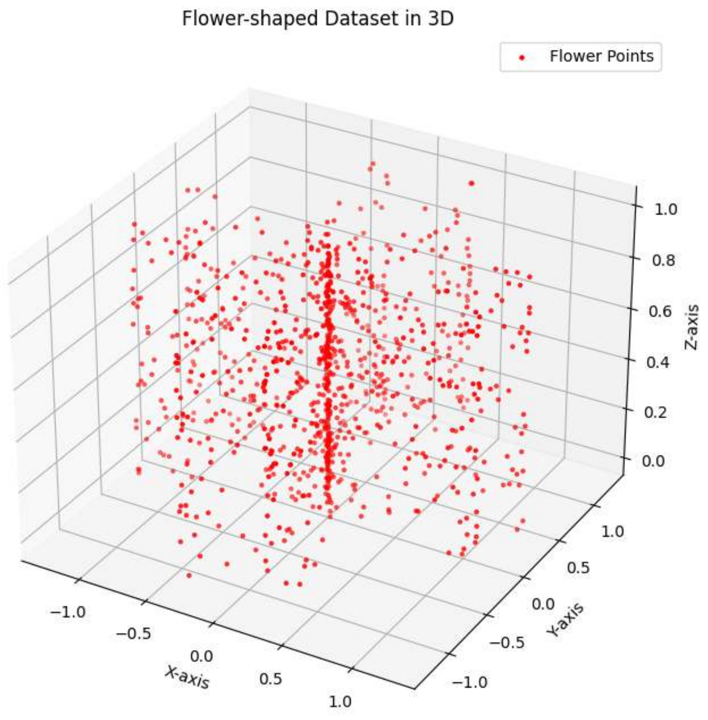


Figure 7.2: Step-2

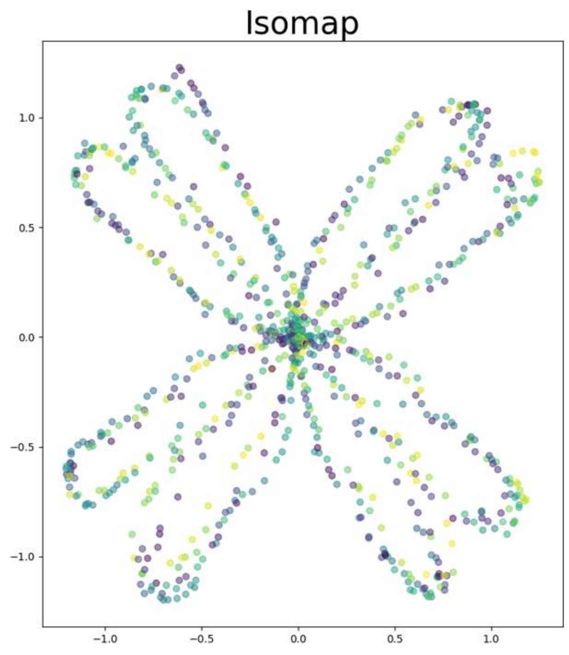


Figure 7.3: Results obtained on Flower dataset through Isomap.

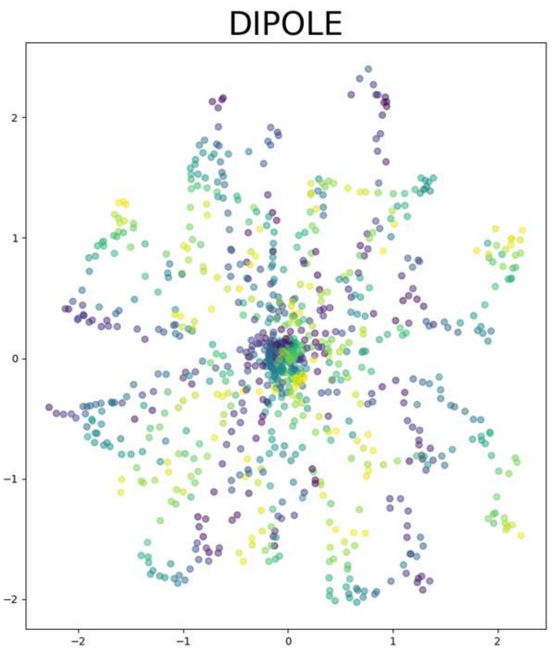


Figure 7.4: Results obtained on Flower dataset through DIPOLE.

dataset has noise similar to the one we discussed, it will be included in the loss functional, and DIPOLE will preserve the topology based on that noise.

This can be dealt with by incorporating any conventional denoising method with DIPOLE to obtain some intermediate dataset and performing DIPOLE on that intermediate dataset. In the above-given specific cases, however, even many denoising methods might not give a better result. We have to use a denoising method that will remove the type of noise we are dealing with, and there is no generalized denoising method.

In the previous case, we took the uniform sampling between 0 and 1, the variance would be around $\frac{(1.0-0)^2}{12} = \frac{1}{12}$. It is similar to that of other features. Now, we will reduce this variance to $\frac{1}{1200}$ by taking uniform sampling from 0 and 0.1 and perform a similar experiment. It can be observed that the lesser the variance in noisy features, the better the results.

7.1.2 Dataset-2: Figure-8 in 4D

In this dataset, we generate a dataset sampled from the structure of figure-8, or infinity as shown in Figure 7.6. Now we will add two random variables to each point in this dataset, increasing the dimension from 2 to 4, in the same way as we did in the case of the flower dataset. Now instead of one noisy feature as in the previous case, this dataset has two noisy

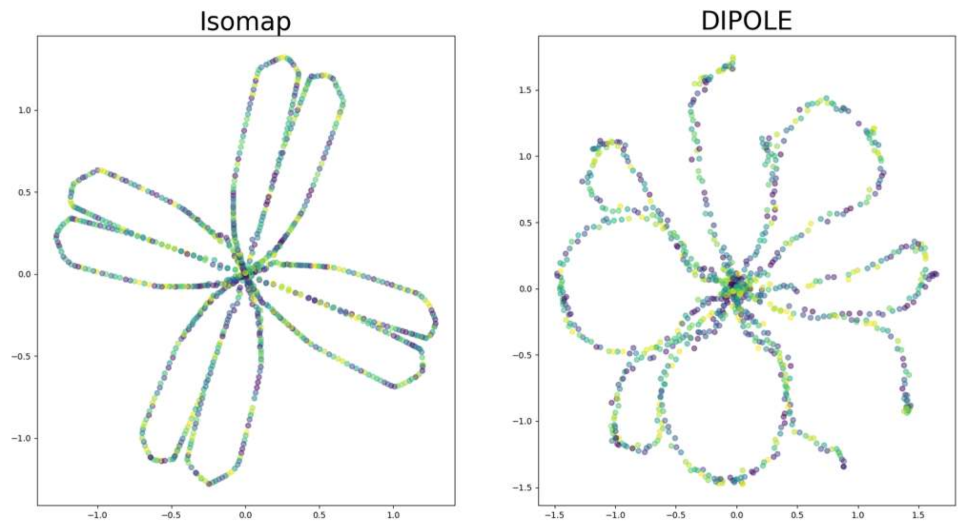


Figure 7.5: Results obtained on Flower Dataset with low-variance noisy feature.

features. Then we will perform different dimensionality reduction methods and compare the results. The results are shown in Figure 7.7.

7.1.3 Conclusion from the Results

It can be concluded from the above results that while Isomap succeeds in identifying the hidden topological structure and showing that structure in low-dimensional embedding, DIPOLE fails in this regard. This is mainly because we are relying on gradient descent optimization, which in turn relies on loss functionals that rely on the high-dimensional data containing noise.

One way to deal with this problem is by using some conventional denoising methods. But as discussed previously, there is no general denoising method that can be applied to all kinds of noises. Another way is to completely throw gradient descent off the hook and rely on some other optimization method that doesn't depend on loss functional. We will have a look at this approach later in this chapter.

7.2 Limitation-2: DIPOLE might disturb the characteristics that were preserved in the initialization embedding.

This limitation presents no big problem, but it's worth noting. Different dimensionality reduction methods aim to preserve different characteristics of high-dimensional datasets

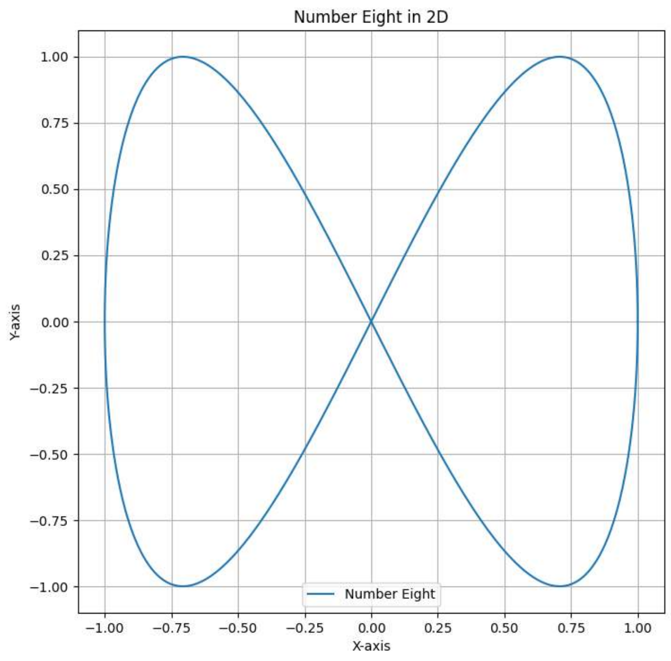


Figure 7.6: Figure-8

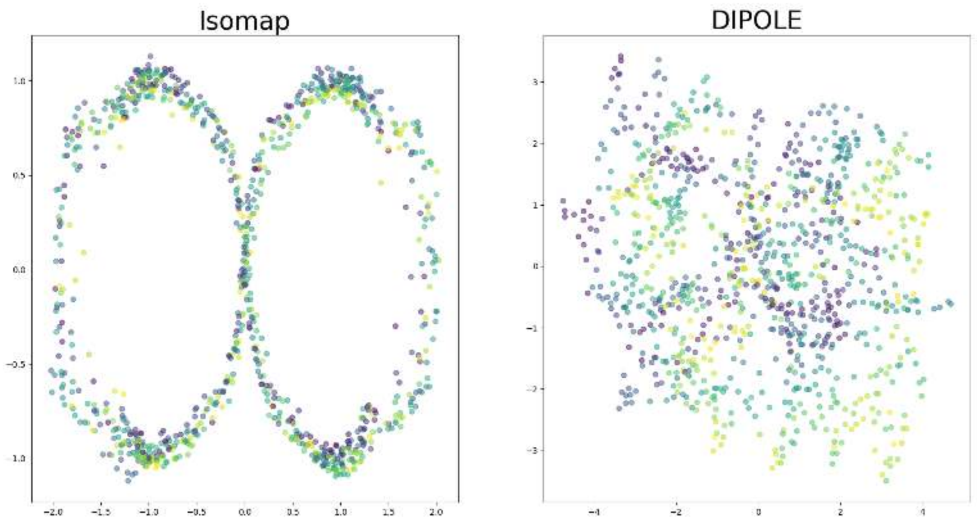


Figure 7.7: Results obtained on performing different dimensionality reduction methods on the figure-8 dataset.

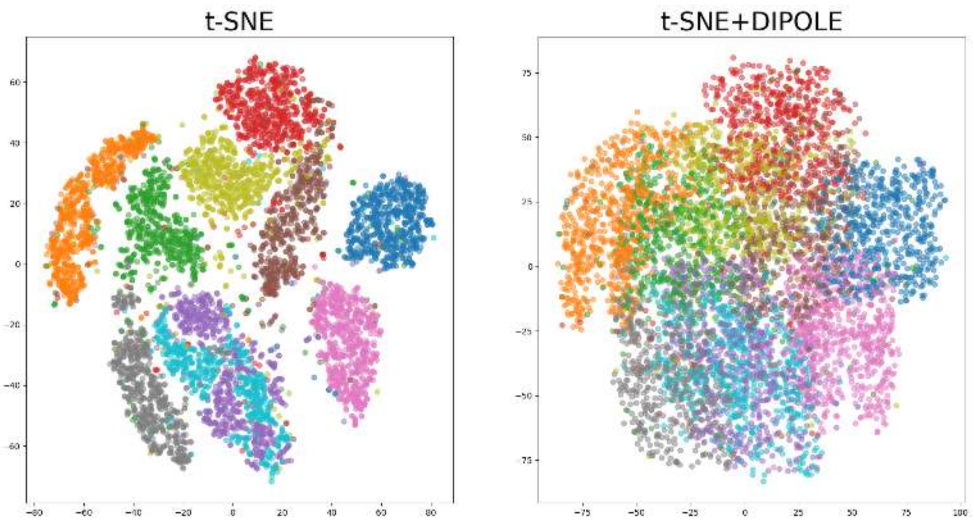


Figure 7.8: DIPOLE disrupts the clustering that t-SNE gave.

in their low-dimensional embeddings. DIPOLE acts as a post-dimensionality reduction method that can improve the topology of the low-dimensional embedding (initialization embedding) obtained through different dimensionality reduction methods and make it more similar to that of the high-dimensional dataset. However, in doing so, it might disrupt the characteristics that were preserved by the earlier dimensionality reduction method that gave the initialization embedding. Have a look at the following Figure 7.8. In the figure, we can see that while t-SNE forms clusters of different digits, DIPOLE disrupts those clusters in order to optimize the embedding based on topology.

This can be dealt with by using an optimization method that isn't based on gradient descent on the loss functional. In the next section, we present an optimization method other than gradient descent that can be used to get an embedding with similar topology. This optimization method will use distributed persistence because of the advantages it presents.

7.3 Grid Search Optimization Method

There are many optimization methods other than gradient descent that can be used. The simplest one is random search optimization, in which we randomly select one parameter value and find the function value based on that parameter value. Then there is simulated annealing, in which we probabilistically accept worse solutions to explore the search space and transition to better solutions. Other optimization methods include genetic algorithms, particle swarm optimization, Nelder-Mead method, Bayesian optimization with Gaussian search, etc. The one we will be using here is the most basic; it's called the grid search optimization method. In this optimization method, we define a particular grid of parameter values, calculate the function value at each point in the grid, and then return the most appropriate function value among all the function values. Before we begin, let's define some notations that will be helpful in understanding all these concepts.

7.3.1 Grid Search Method

At the moment, let's consider Isomap dimensionality reduction method. We will be following a similar process with t-SNE dimensionality reduction method, but for now, let's represent our function by the Isomap method.

Now, let's denote our function by $f_n : X \to \mathbb{R}^D$, where $\dim(X) = N > D$, and $n =$ n-nearest neighbors parameter value for Isomap. And let $d_B(X, f_n(X))$ be the *average bottleneck distance* between the distributed persistence diagrams of X and $f_n(X)$. This bottleneck distance is the average of bottleneck distances between the subsets of $S \subset X$ of some fixed size and their respective images $f_n(S) \subset \mathbb{R}^D$. This will ensure that we use the concept of distributed persistence here instead of standard persistence diagrams.

Now, for different $n_1, n_2, \cdots, n_k$, we will be calculating the bottleneck distances between the distributed persistence diagram of X and $f_{n_i}(X)$ where $i \in \{1, 2, 3, ..., k\}$.

$$\begin{tikzcd}
X \arrow[d, dash] & X \arrow[d, dash] & \cdots \arrow[d, dash] & X \arrow[d, dash] \\
d_B \arrow[d, dash] & d_B \arrow[d, dash] & d_B \arrow[d, dash] & d_B \arrow[d, dash] \\
f_{n_1}(X) & f_{n_2}(X) & \cdots & f_{n_k}(X)
\end{tikzcd}$$

We can then create a graph of Bottleneck Distance vs Neighbourhood Size. Have a look at the figures 7.9, and 7.10. Figure 7.9 shows the average Bottleneck Distance vs Neighbourhood Size graph of the Mammoth Dataset, and Figure 7.10 shows the average Bottleneck Distance vs Neighbourhood Size graph of the Brain Artery Tree Dataset introduced in 4.3.

Now we have a number of average bottleneck distances between the original space and its image with Isomap with different parameter values. Amongst all these parameter values, we choose the parameter value for which the bottleneck distance is minimum, and return the embedding with respect to this parameter value. This way, we haven't changed the final embedding obtained through Isomap, implying that the characteristics that we were trying to preserve aren't disturbed. Also, since we have used Isomap directly, and we have already noted that Isomap isn't much disturbed by the presence of the noisy feature, this method will give an embedding unaffected by the presence of such noises. Also, since we are

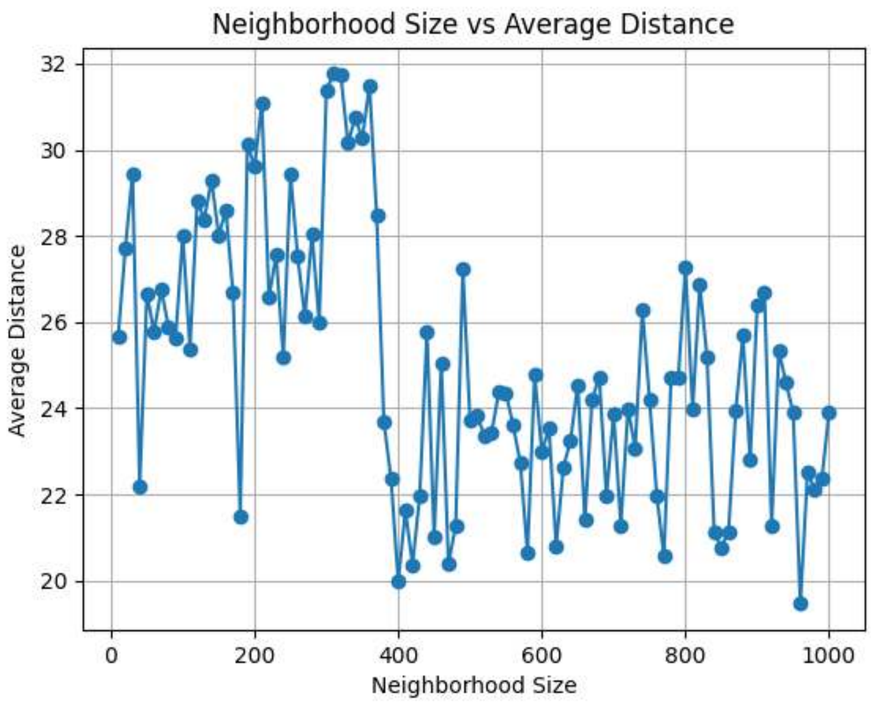


Figure 7.9: Mammoth Dataset

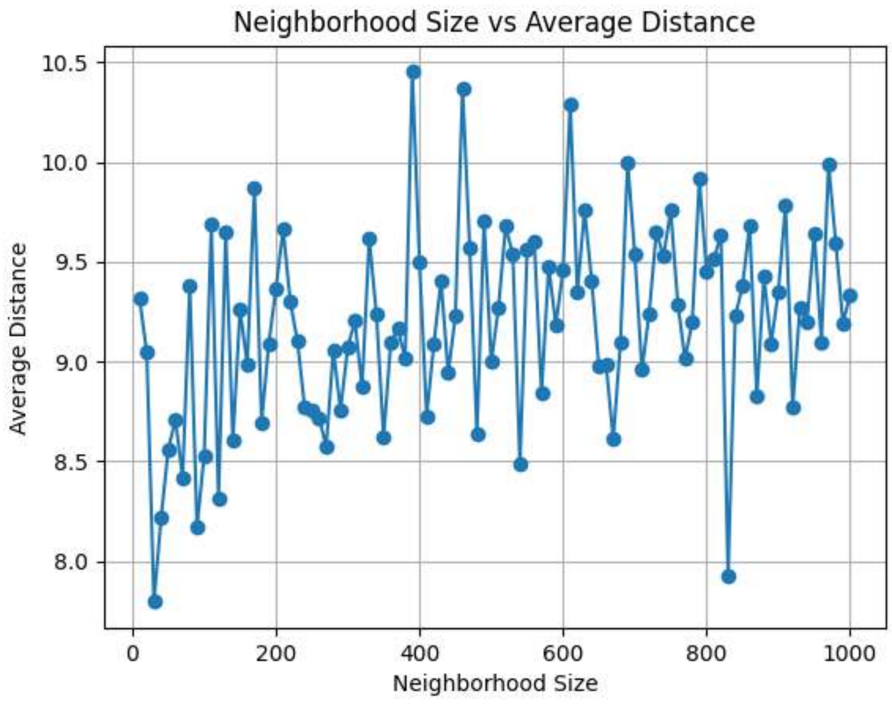


Figure 7.10: Brain Dataset

considering average bottleneck distances of subsets, we are using distributed persistence. This also gives the advantages that come with distributed persistence. However, it will still be computationally expensive since we are performing the dimensionality reduction again and again, n number of times, where n equals the number of parameter values in the grid. This is one example of using optimization methods other than gradient descent, that is based on distributed persistence, there might be a faster and more efficient optimization method than this.

7.3.2 Algorithm

To understand better, let's define everything:-

- Let's represent Isomap by, $f_n : X \to \mathbb{R}^D$, where $\dim(X) = N > D$, and $n =$ n-nearest neighbor.
- $\mathscr{I} = \{n_1, n_2, \cdots, n_k \,|\, n_i \in \mathbb{N} \text{ for all } i \in \{1, 2, \cdots, k\}\}$.
- Let $d_B(X, f_n(X))$ be the bottleneck distance between the distributed persistence diagrams of X and $f_n(X)$.
- Now we take m such that,

$$d_B(X, f_m(X)) = \text{Inf}\,\{d_B(X, f_{n_i}(X)) \,|\, \forall\, n_i \in \mathscr{I}\}.$$

- And we return $f_m(X)$ as our final embedding.

The diagram representation of our algorithm is as follows:-

$$\begin{tikzcd}
\text{High-Dimensional Dataset} \arrow[r] \arrow[d, "\mathscr{I}"'] \arrow[r, phantom, shift right=6ex, "\text{Repeat}"] & \text{D-Persistence Diagram} \arrow[d] \\
\text{DR-Method} \arrow[d] & \text{Distance} \arrow[dd, <->, bend left=60] \\
\text{Low-Dimensional Embedding} \arrow[r] & \text{D-Persistence Diagram} \arrow[u] \\
 & \text{Minimum such distance with its corresponding embedding will be returned.}
\end{tikzcd}$$

The algorithm can be given as follows:-

Best Topological Embedding (BTE)

Input: High-dimensional dataset HD, parameters n, A, B

Output: Low-dimensional embedding LD

Algorithm 2 Best Topological Embedding (BTE)

```
1: Initialization: Set $d_m = \infty$.
2: Parameter Selection: Fix $(A,B)$ such that $A < B$ with $A, B \in \mathbb{Z}_+$, and also fix $k \in \mathbb{Z}$.
3: Calculate Distributed Persistence Diagram: Calculate the distributed persistence diagram of $HD$, denote it by $D(HD)$.
4: Isomap Embedding: Perform Isomap with $n = k$ on $HD$ to obtain $LD_k$.
5: Calculate Distributed Persistence Diagram of Embedding: Calculate the distributed persistence diagram of $LD_k$, denote it by $D(LD_k)$.
6: Calculate Distance: Calculate the distance between $D(HD)$ and $D(LD_k)$ and denote it by $d_k$.
7: Update Minimum Distance: Update $d_m$ as $\min(d_m, d_k)$.
8: Update Embedding: Update $LD$ as $LD_m$.
9: Update Parameter: Update $k$ as $k = k + A$.
10: Iteration: Repeat Steps 5 to 10 until $k > B$.
11: Output: Return $LD$.
```

7.3.3 Experimental Results

We performed BTE on mammoth, brain, and Swiss hole datasets. The visualization results are shown in the following figures: 7.11, 7.12, 7.13, and 7.14. d_m mentioned above the BTE figure is the average bottleneck distance of that embedding with the original high-dimensional dataset. BTE returns the embedding that has a minimum average bottleneck distance among all the parameter value embeddings.

Now, let's have a look at the datasets that were used in limitation-1 to see whether BTE is able to deal with the limitations that were mentioned. Have a look at the figures: 7.15, and 7.16. It is clear that BTE is not affected by the noise in the datasets.

Now let's see whether BTE is able to deal with limitation-2 of DIPOLE. Now instead of Isomap, we will consider t-SNE, and instead of n-nearest neighbors, we will consider the perplexity parameter. BTE will then create a grid with different perplexity parameter values and choose the perplexity parameter value that gives an embedding with minimum average bottleneck distance and return its corresponding embedding. Have a look at the figure: 7.17. It can be seen that BTE is able to maintain the clustering property of t-SNE while also giving an embedding that is closest in terms of topology to the high-dimensional dataset.

7.3.4 Conclusion on BTE

Now we present some benefits and limitations of BTE:-

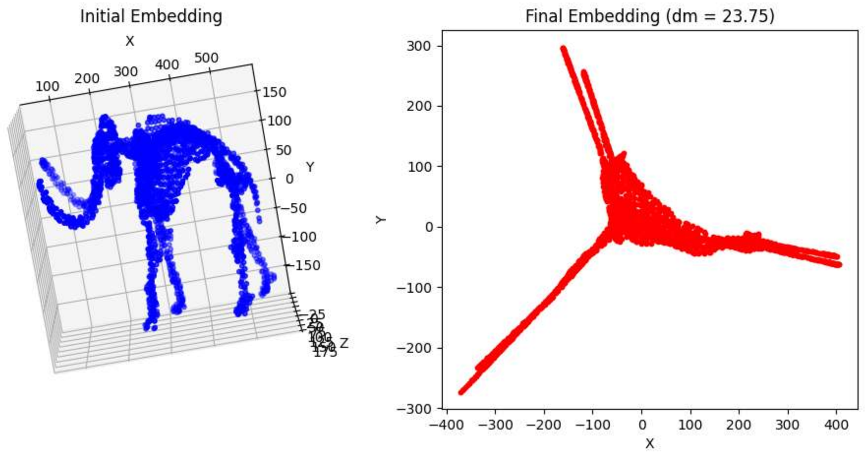


Figure 7.11: Mammoth Dataset: Initial Neighbourhood Size = 10, Increment = 10, Max Neighbourhood Size = 100

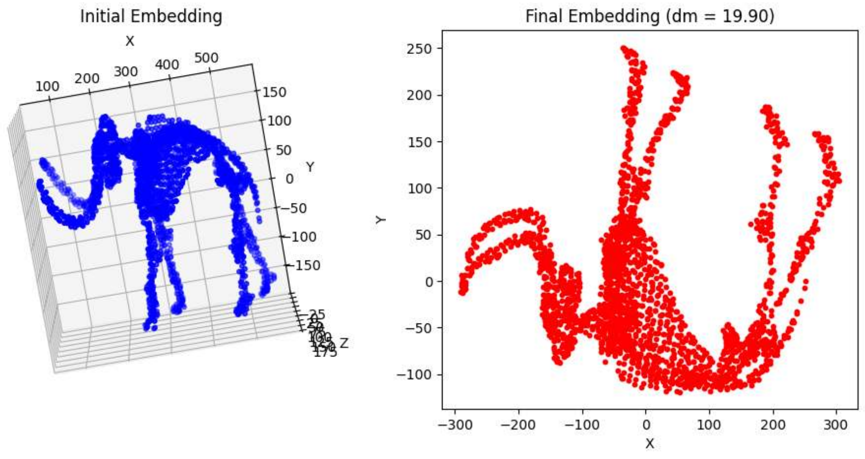


Figure 7.12: Mammoth Dataset: Initial Neighbourhood Size = 10, Increment = 10, Max Neighbourhood Size = 500

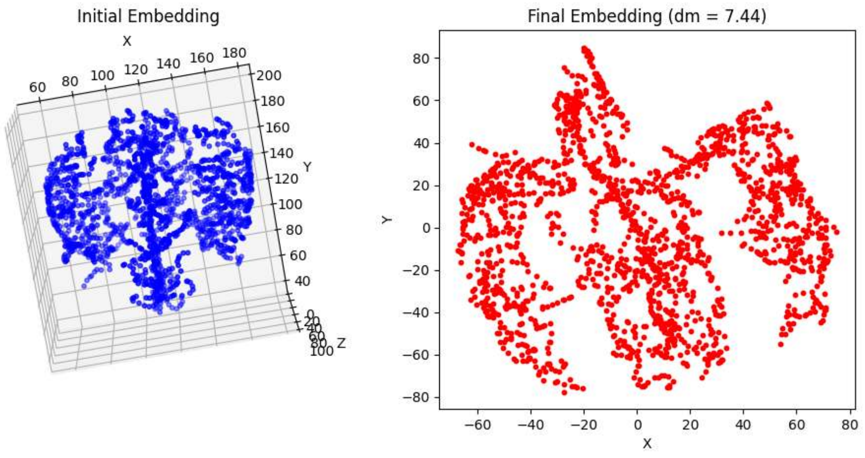


Figure 7.13: Brain Dataset: Initial Neighbourhood Size = 10, Increment = 10, Max Neighbourhood Size = 100

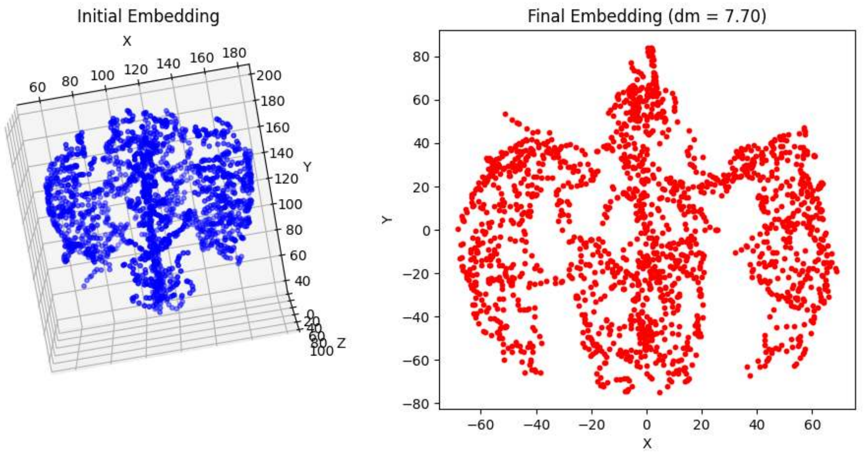


Figure 7.14: Brain Dataset: Initial Neighbourhood Size = 10, Increment = 10, Max Neighbourhood Size = 1000

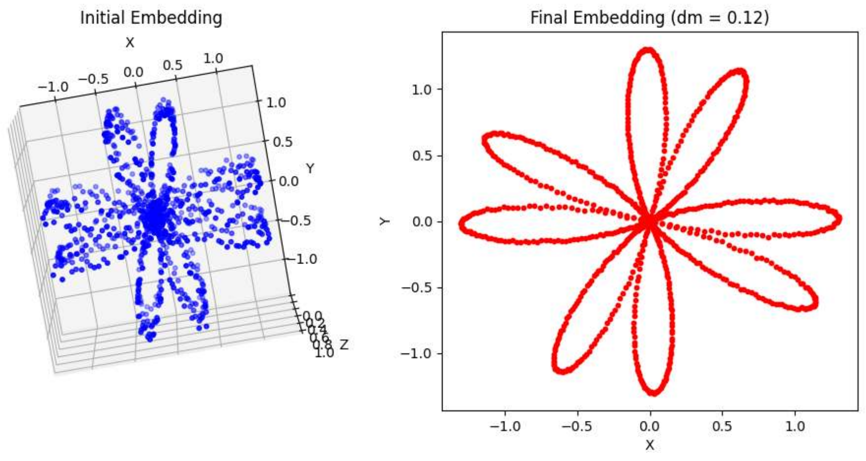


Figure 7.15: Flower Dataset: Initial Neighbourhood Size = 10, Increment = 10, Max Neighbourhood Size = 500

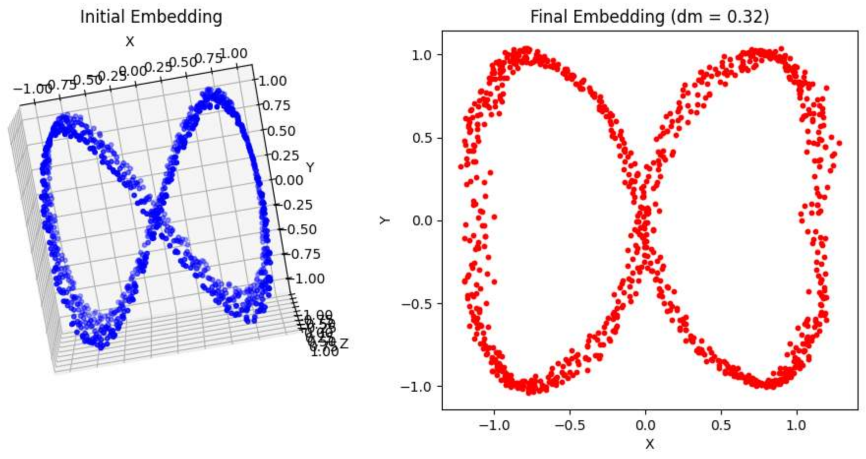


Figure 7.16: Figure-8 Dataset: Initial Neighbourhood Size = 10, Increment = 10, Max Neighbourhood Size = 500

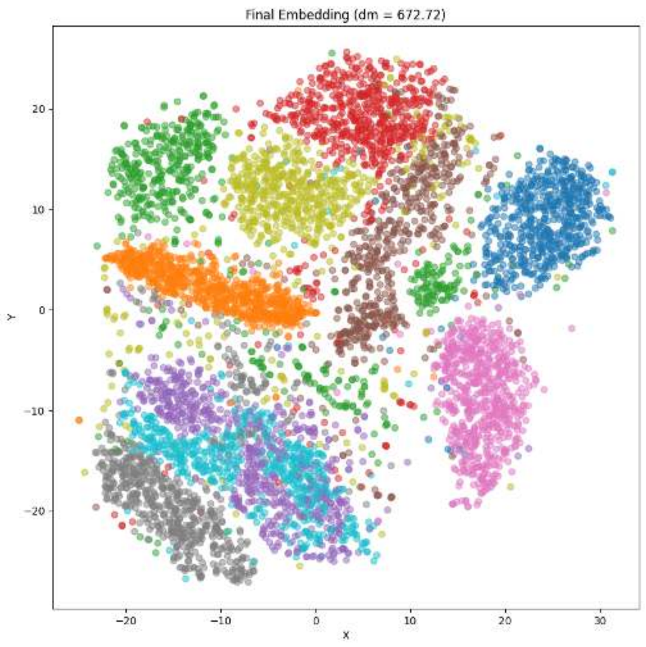


Figure 7.17: MNIST Dataset: Initial Neighbourhood Size = 10, Increment = 20, Max Neighbourhood Size = 500

Benefits:-

- We get the best low-dimensional embedding in terms of topology in which the characteristics that we were trying to preserve are not disturbed.
- Since we have used the Distributed Persistence diagrams, we have also weakly maintained quasi-isometry between the high-dimensional dataset and its low-dimensional embedding.
- Also, the use of Distributed Persistence diagrams instead of persistence diagrams presents the advantage of faster computation. However, this process will be computationally expensive as the range of parameter value increases, but it will not depend on the cardinality of the dataset.
- This is also not affected by noise since we are using the dimensionality reduction method directly.

Limitations:-

- Computationally expensive. As we increase the max-parameter value, the time for computation will also increase. This can be, however, dealt with by making the algorithm and computation process more efficient.

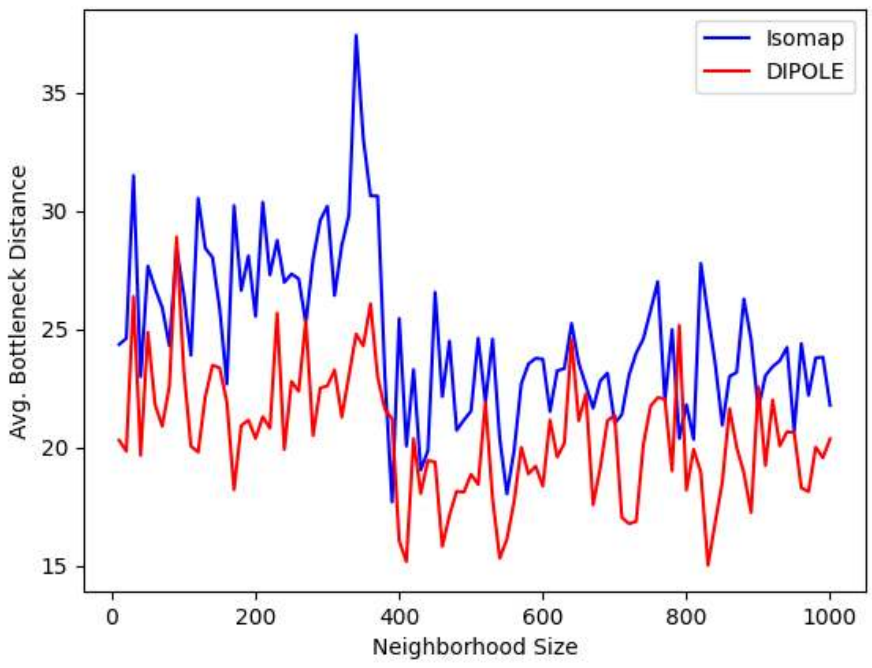


Figure 7.18: Isomap vs DIPOLE

7.4 Isomap vs DIPOLE

DIPOLE Pipeline:

$$HD \longrightarrow f_n \longrightarrow IE \longrightarrow \text{DIPOLE} \longrightarrow FE$$

Previously, we tried to calculate the bottleneck distance between high-dimensional datasets, and their initialization embedding that we obtained through Isomap. Now we will perform DIPOLE on that initialization embedding to see how DIPOLE improves the topology. We will be calculating the average bottleneck distance vs neighborhood size graph for Isomap, as we did previously, and additionally, we will also do it for DIPOLE, in which all the parameters will be fixed, and we will be varying the neighborhood size of the Isomap method that will give the initialization embedding. We will be performing this for the Mammoth dataset. We get the graph shown in figure 7.18.

It can be seen that adding DIPOLE to the initialization embedding obtained via Isomap reduces this average bottleneck, implying that it improves the topology of this initialization embedding.

7.5 Conclusion

Although DIPOLE presents us with an excellent post-dimensionality reduction method that can be used with any dimensionality reduction method giving the initialization embedding, DIPOLE does have certain limitations. We discussed two of those limitations in this chapter and tried to overcome those limitations by using a different way of optimization. This gave decent results but it also came with the cost of the increase in computational complexity. There are other optimization methods that can use the principle of distributed persistence, and then be used in dimensionality reduction methods and help in preserving the topology in lower dimensions. This can help in improving dimensionality reduction methods that preserve the topology. After that, we also saw how DIPOLE can help improve topology for different parameter values of the initialization embedding, namely in the case of Isomap, with the Mammoth dataset.

In the end, DIPOLE can be considered an excellent tool for preserving the topology of high-dimensional datasets in lower dimensions. Albeit with certain limitations, it can be improved, so that it can deal with those limitations.

Chapter 8

Conclusion

In this thesis, we discussed a novel dimensionality reduction method that maintains the topology of high-dimensional data in its low-dimensional representation. This method is called DIPOLE. DIPOLE uses distributed persistence instead of standard persistence. We discussed distributed persistence and showed certain experiments with distributed persistence. DIPOLE is a post-dimensionality reduction method that needs initialization embedding obtained from any conventional dimensionality reduction method, while the authors of the paper, that introduced DIPOLE, used Isomap since it helps in satisfying certain conditions, we considered another dimensionality reduction method called t-SNE for this initialization embedding and produced certain experimental results. We also looked over certain limitations that DIPOLE presents and discussed some ways of dealing with those limitations. We also presented one way, albeit impractical, for dealing with those limitations.

In conclusion, we can say that DIPOLE is an excellent way to preserve the topology of high-dimensional datasets in their low-dimensional embedding. It is flexible and can be improved upon.

Appendix A

Appendix

A.1 Gradient Descent

We have briefly summarised gradient descent here. A better understanding can be developed from the following video: https://www.youtube.com/watch?v=sDv4f4s2SB8&ab_channel=StatQuestwithJoshStarmer.

Gradient descent is an optimization algorithm used to minimize a function by iteratively moving in the direction of the negative gradient of the function at a given point. Let $J(\theta)$ represent the cost function to be minimized, where θ denotes the parameters of the function. Now,

Objective function: $J(\theta)$
Gradient: $\nabla J(\theta) = \left[\frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \ldots, \frac{\partial J}{\partial \theta_n}\right]$

- Gradient descent minimizes $J(\theta)$ by updating parameters iteratively.
- Gradient $\nabla J(\theta)$ points in the direction of steepest increase.
- Move opposite to the gradient to reach a local minimum.

Update rule for parameter θ_i:

$$\theta_i = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

- α is the learning rate, and controls step size in each iteration.
- Large α may cause overshooting, small α leads to slow convergence.
- Iterate until convergence or the predefined number of iterations.
- Gradient descent converges to local minimum of $J(\theta)$.

Have a look at the figure A.1.

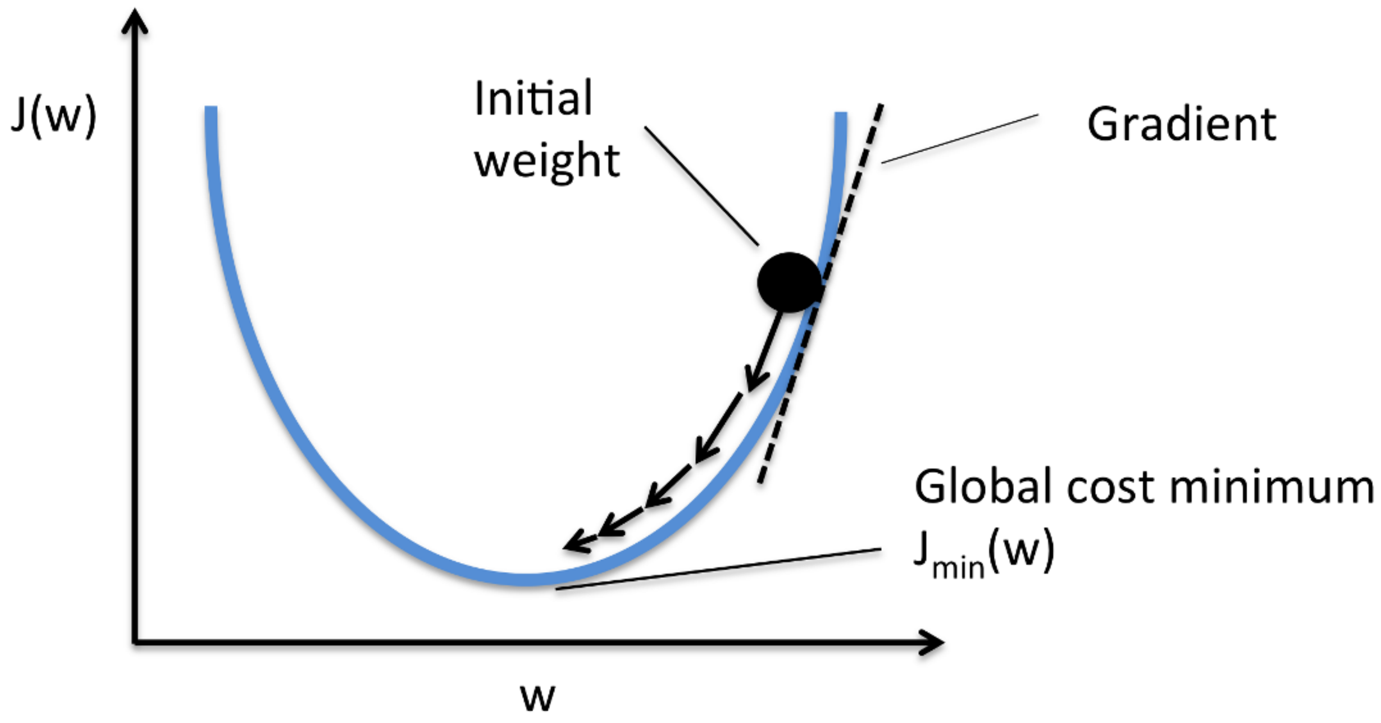


Figure A.1: Gradient Descent

Bibliography

[Borg 05] I. Borg & P.J.F. Groenen. Modern multidimensional scaling: Theory and applications. Springer, 2005.

[Borne 21] Kirk Borne. *When Big Data Goes Local, Small Data Gets Big*, 2021.

[Bullitt 05] Elizabeth Bullitt, Donglin Zeng, Guido Gerig, Stephen Aylward, Sarang Joshi, J. Keith Smith, Weili Lin & Matthew G. Ewend. *Vessel tortuosity and brain tumor malignancy: A blinded study*. Academic Radiology, vol. 12, no. 10, pages 1232–1240, 2005.

[Chen 20] Guangliang Chen. *Lecture 10: Isometric Embedding*, 2020.

[Deisenroth 19] M. P. Deisenroth, A. A. Faisal & C. S. Ong. *Mathematics for Machine Learning*, 2019. Draft (August 20, 2019). To be published by Cambridge University Press. `https://mml-book.com`.

[Dey 21] Tamal Krishna Dey & Yusu Wang. Computational topology for data analysis. Cambridge University Press, Cambridge, 2021. Prepublication version.

[Edelsbrunner 10] Herbert Edelsbrunner & John Harer. Computational topology: An introduction. American Mathematical Society, Providence, Rhode Island, 2010.

[Hotelling 33] H. Hotelling. *Analysis of a complex of statistical variables into principal components*. Journal of Educational Psychology, vol. 24, no. 6, pages 417–441, 1933.

[Moor 21] Michael Moor, Max Horn, Bastian Rieck & Karsten Borgwardt. *Topological Autoencoders*, 2021.

[Nanda XX] Vidit Nanda. Computational algebraic topology: Lecture notes. XXXX. With title illustrations by Robert Ghrist.

[Pearson 01] 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, pages 559–572, 1901.

[Reani 21] Yohai Reani & Omer Bobrowski. *Cycle Registration in Persistent Homology with Applications in Topological Bootstrap*, 2021.

[Shepard 62] Roger N Shepard. *The analysis of proximities: Multidimensional scaling with an unknown distance function. I.* Psychometrika, vol. 27, no. 2, pages 125–140, 1962.

[smi 20] *Smithsonian. Mammuthus primigenius (Blumbach), 2020*, 2020.

[Solomon 22] Elchanan Solomon, Alexander Wagner & Paul Bendich. *From Geometry to Topology: Inverse Theorems for Distributed Persistence*, 2022.

[Starmer 14] Josh Starmer. *Principal Component Analysis (PCA) clearly explained (2014)*, 2014.

[Tenenbaum 00] Joshua B Tenenbaum, Vin de Silva & John C Langford. *A global geometric framework for nonlinear dimensionality reduction.* Science, vol. 290, no. 5500, pages 2319–2323, 2000.

[Wagner 21] Alexander Wagner, Elchanan Solomon & Paul Bendich. *Improving Metric Dimensionality Reduction with Distributed Topology*, 2021.