

# TopoPy: An Experimentation with Various Distance Metrics

Karenn Rodriguez

NYU Undergraduate

kr2266@nyu.edu

Alexandra Casas

NYU Undergraduate

ac7262@nyu.edu

GitHub Link: TopoPy

## ABSTRACT

Dimensionality reduction is a fundamental tool for removing multicollinearity which improves the interpretation of the parameters of any machine learning model. Multicollinearity occurs when several independent variables in a model are correlated. The presented case, TopoPy, is based on a novel projection technique that preserves components during the mapping process. A limitation produced by such technique was due to the computation of euclidean minimum spanning trees when increasing the dimensions. As a result, our presented case studies the effect of various distance metrics on the execution time when producing minimum spanning trees as a means of overcoming the main bottleneck—generating EMSTs.

## 1 INTRODUCTION

The main goal of TopoPy is to explore various distance metrics that improve the execution time of TopoMap. Since creating Euclidean Minimum Spanning Trees (EMST) were the main bottleneck of TopoMap's algorithm, we are interested in comparing the performance of our project using Manhattan or Cityblock distance with that of TopoMap's implementation which uses Euclidean distance. The Manhattan distance metric determines the absolute difference among pairs of coordinates, whereas Euclidean distance computes the ordinary distance between two points. Our interest

in TopoMap stems from wanting to learn how to implement a projection technique from scratch that focuses on preserving topological structures during the dimensionality reduction process. We'd like to modify TopoMap to use Manhattan distance due to its better performance than Euclidean distance in high dimensional applications.

Since the TopoMap research paper also mentions that EMSTs did not perform well with high point clouds, we plan on using various datasets such as Scikit Learn's Iris Classification and Breast Cancer Datasets which increase in size. The Iris Dataset consists of three classes—Setosa, Versicolour, and Virginica—, four attributes —Sepal Length, Sepal Width, Petal Length, Petal width—, and 150 instances. The Breast Cancer Dataset consists of two classes —Benign and Malignant—, thirty attributes, and 569 instances.

We plan on implementing TopoPy in a similar way to TopoMap. Therefore, we will use a minimum spanning tree framework and calculate distances. TopoMap primarily uses Euclidean distance which we will also use to test TopoPy; however, we will use Manhattan Distance as TopoPy's primary distance metric. In order to compute the convex hull, which creates a boundary that encloses all the points in a set, we plan on using SciPy's Convex-Hull spatial algorithm.

We will be comparing our running time for TopoPy with that of TopoMap for the Iris Dataset. The layout interpretation of our Iris clusters will be explored via the following guidelines provided by the research paper discussed above:

- Use a density-based colormap to visualize the projection.
- Start exploration by looking at the centers of stars with high density. These typically represent clearly distinct clusters in the data.
- Use low density stars to study “uncommon behaviors”.
- Explore branches to analyze sparse clusters and outliers.

## 2 RELATED WORK

The research paper, “TopoMap: A 0-Dimensional Homology Preserving Projection of High-Dimensional Data,” introduces TopoMap, a multidimensional projection (MDP) technique that aims to preserve the homology of 0-cycles of the Rips filtration. In simpler terms, TopoMap ensures that the topological filtrations over an original and projected dataset create the same connected components at the same instances of the filtrations. Therefore, all of the topological structures are preserved during the dimensionality reduction process. The development of TopoMap was prompted by the lack of an MDP technique that could map high-dimensional data and guarantee the preservation of geometric properties under any condition. Analyses of high-dimensional data and the identification of objects with similar features are more reliable using a technique like TopoMap. Additionally, TopoMap could be used as a metric to analyze how other MDP methods map connected components.

The concept for TopoMap stems from research in Topological data analysis and Topology-based Multidimensional Projection. When it comes to Topological data analysis, Rips filtration plays a prominent role in analyzing the topology of high dimensional point clouds by providing geometric and topological information about a given point set. Another topological analysis tool is Topology-based MDP which influenced the development of methodologies to analyze MDP distortions and facilitate extraction of information hidden in projections. An example of this is Isomap which, unlike

TopoMap, captures topological structures by computing geodesic distances and does not preserve 0-homology groups. Another MDP technique is terrain metaphors which creates intuitive visualizations of topological features in a volume scalar field, but discards information from the original space. In contrast, TopoMap’s projection layout allows for the persistent homology of the Rips filtration.

TopoMap attempts to provide a more efficient alternative to existing high dimensional data visualization techniques such as MDS, Isomap, t-SNE and more, while also preserving connected components and facilitating the analysis of high dimensional data. This research paper even includes a case study in which TopoMap was used to understand how the usage of city hotspots – Penn Station, Times Square – change based on factors such as weather. TopoMap proved successful in identifying patterns and dependent features, and in general executes faster than existing techniques.

## 3 METHODOLOGY

Before discussing our proposed methodology to improve TopoMap, it is important to explain the algorithm that constructs TopoMap. The algorithm manages all of the high dimensional points as separate components and stores the minimum spanning edges as an ordered (by length) set. As seen in Line 5, with each iteration, the smallest edge from the ordered set is added as a way to connect two components. In turn, reducing the number of maintained components by one, as well as preserving the length of the edge. This is important because the distance between the two components is equal to the length of the connecting edge that has the same length as its counterpart in the original space.

Again, with each iteration, an edge is removed until all edges are appropriately placed. As highlighted in the research paper, Line 5c places the points based on the minimum spanning tree edge lengths, more specifically, it works toward placing the two components such that the minimum distance between them is equivalent to the current edge. Line 5d indicates the preservation of the set of connected components is accomplished via the union-find data structure. At the end, a correspond-

ing set of points in the real coordinate space of dimension 2 is returned.

We aim to preserve the connected components after every iteration, but to also try and preserve the end points of the minimum spanning tree edges as much as possible. This is crucial for generating a dimensionality reduction technique. In addition, we will also incorporate convex hulls in order to envelop sets of points and rotate them. Convex hulls are beneficial in the preservation of connected components.

The following algorithm is the procedure for TopoMap. It is implemented in C++, while we have implemented it in Python with a few modifications.

Require: high dimensional points  
 $P = \{p_1, p_2, \dots, p_n\}$

1. Compute the euclidean minimum spanning tree ( $E_{mst}$ ) over  $P$ .
2. Let  $E_{mst} = \{e_1, e_2, \dots, e_{n-1}\}$  be the edges ordered by length.
3. Let  $P' = \{p'_1, p'_2, \dots, p'_n\}$  where  $p'_i = (0, 0), \forall i$ .
4. Let  $C_i = \{p'_i\}$  be the initial set of components.
5. For each  $i \in [1, n - 1]$  do:
  - a. Let  $(p_a, p_b)$  be the end points of edge  $e_i$
  - b. Let  $C_a$  be the component containing  $p'_a$  and  $C_b$  be the component containing  $p'_b$
  - c. Place  $C_a$  and  $C_b$  in  $\mathbb{R}^2$  s.t  $\min_{p'_j \in C_a, p'_k \in C_b} \{d(p'_j, p'_k)\} = \text{length}(e_i)$
  - d. Let  $C' = C_a \cup C_b$ .
  - e. Remove  $C_a$  and  $C_b$  from the set of components and add  $C'$  into this set.
6. Return  $P'$

The modification we have implemented alters the distance metric utilized in TopoMap. Rather than using the Euclidean distance, the Manhattan distance is explored. The Manhattan distance is also known as the city block distance and/or the taxicab distance, representing the sum of the absolute differences between coordinates of two points. In a

two-dimensional space, the Manhattan distance is calculated as:

$$d = |x_2 - x_1| + |y_2 - y_1| \quad (1)$$

In Equation 1, we are looking at two points  $(x_1, y_1)$  and  $(x_2, y_2)$ . In a multi-dimensional, the formula is generalized in the following way:

$$d = |p_i - q_i| \quad (2)$$

The generalized algorithm for calculating the Manhattan distance consists of a function requiring two arguments, representing the points, in any dimensional space. The function iterates over each point's dimensions, element-wise, via the zip function. The zip function is a built-in function that combines two or more lists into an object containing ordered tuples from the lists. It is important to note that the zip function can be passed any iterable object. With each iteration, we calculate the absolute difference between the two points, accumulating the sum of the differences.

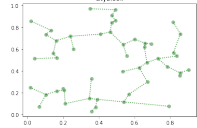
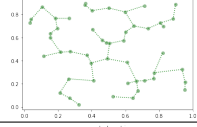
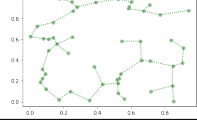
For our TopoPy research, we will focus on step 1 and 2 which computes the minimum spanning tree and orders the edges by distance.

## 4 RESULTS

### 4.1 Initial Experiments with random 2D data

Using numpy's random method, we developed a 2D dataset and called the *minimum\_spanning\_tree* method. Below are three graphs that use different distance metrics (Manhattan, Euclidean, and Mahalanobis ) and its execution time (in seconds). Based on our testing, we noticed that Manhattan and Euclidean distance perform similarly, whereas Mahalanobis had a longer execution time with the 2D data.

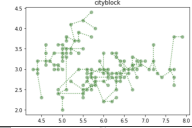
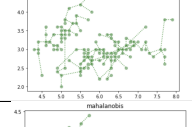
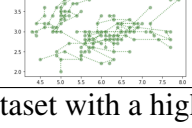
Table 1: Experimentation with Random Dataset

Distance Metric	Execution Time (s)	Graph
Manhattan	0.001494480	
Euclidean	0.001911838	
Mahalanobis	0.012272066	

## 4.2 Experiments with high dimensional data

The following charts illustrate the minimum spanning trees for high dimensional data. We first generated the minimum spanning trees for the Iris dataset. And, we then explored the Breast Cancer dataset. Our initial hypothesis that Manhattan distance will be the most efficient distance metric for higher dimensional data, held to be true. Looking at Table 2, Manhattan and Euclidean distances have similar graphs and both perform better than Mahalanobis which also produces a different MST. However, Manhattan distance still took the lead in regard to execution time.

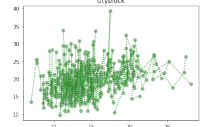
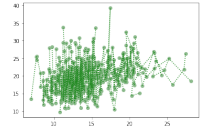
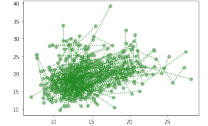
Table 2: Experimentation with Iris Dataset

Distance Metric	Execution Time (s)	Graph
Manhattan	0.008072213	
Euclidean	0.010547122	
Mahalanobis	0.015631925	

When moving onto a larger dataset with a higher dimensionality as seen in Table 3, we see similar results as the ones in Table 2. Again, Manhattan

and Euclidean distances generate similar graphs and both perform better than Mahalanobis. And yet again, Manhattan distance generated the MST edges 0.01 seconds faster than Euclidean distance.

Table 3: Experimentation with Breast Cancer Dataset

Distance Metric	Execution Time (s)	Graph
Manhattan	0.259317826	
Euclidean	0.264078501	
Mahalanobis	0.396260903	

## 5 DISCUSSION

Based on our results we noticed a consistent difference between Mahalanobis and our other distance metrics. Though Mahalanobis is not the distance metric we primarily focus on, we believe that it could also provide interesting visualizations once TopoPy is complete, despite having a longer execution time. Unlike Manhattan and Euclidean distance, Mahalanobis has a complex computation:

$$D^2 = (x - m)^T * C^{-1} * (x - m)$$

which is most likely the reason behind its lengthy runtime. The key component of Mahalanobis is that it divides by the covariance matrix. So, if the components in a dataset are strongly correlated, then the covariance will be high and dividing by a large covariance will effectively reduce the distance. Furthermore, components that are closely related will be closer to each other in the visualization.

Our focus on exploring various distance metrics provides an easier method of gauging the projection in the visualized space for users. This is something that is discussed in the research of TopoMap. However, our current state of TopoPy limits such interpretation. When implementing the various distance metrics, we utilized the `scipy.spatial.distance.pdist`

method which produces the pairwise distances between observations in  $n$ -dimensional space. A benefit of using this method is its ability to test various distance metrics by simply specifying the name of the desired metric. It returns a condensed distance matrix, which we then pass to our minimum spanning tree method. However, a limitation that we have yet to get around is retrieving the distances of the edges produced by our MST method. This is crucial for the successful implementation of the place points method used in TopoMap.

## 6 CONCLUSION

In this paper we introduced the beginning of TopoPy, a program inspired by TopoMap, that serves as a dimensionality reduction technique that preserves a dataset's original geometry or connected components. Our approach using different distance metrics provided hopeful results in regard to implementing TopoMap using a Manhattan Distance MST. In the future, we would like to further explore how different distance metrics impact TopoPy's time efficiency and visualizations when it comes to processing datasets with larger point clouds and increasing dimensions. Additionally, we would like to implement TopoMap's place points methodology, in order to have a better understanding of how our varying distance metrics impact the dimensionality reduction, component preservation, and execution time of our program.

## 7 REFERENCES

Doraiswamy, H., Tierny, J., Silva, P. J., Nonato, L. G., Silva, C. (2020). Topomap: A 0-dimensional homology preserving projection of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), 561-571. <https://arxiv.org/pdf/2009.01512.pdf>