Interactive Multi-Scale System for Topological Visualization

Yichen Xie*
New York University

Jingyang Men†
New York University

Houze Liu‡
New York University

ABSTRACT

In this paper we introduce a new system for topological visualization on high-dimensional data, which includes Dim-Map, Scale-Map and Force Directed TopoMap. Our method is mainly based on the idea of TopoMap, with novel features like selecting data dimension and changing data scale interactively. We translate the implement of TopoMap from C into Python without losing efficiency, and add interactive features via D3.js and Jupyter Notebook. We also present a few sample applications to illustrate the advantages in providing important information in high-dimensional data visualization tasks. Our implementation can be accessed at GitHub: https://github.com/Kurisute/TopoMap-Python

Index Terms: Topological Data Analysis; Interactive Visualization Systems; High-dimensional data

1 INTRODUCTION

Topology-based visualization methods are becoming increasing popular in the last two decades. In contrast to classical dimension reduction methods which based on geometric features such as Euclidean distance, the topological methods mainly focus on preserving topological properties during the process of mapping high-dimensional data to a 2-D or 3-D visual space. Thus topological methods can have better performance in the conditions of manifold distortion and enfoldment, and provide exact and correct projection layouts for the data analysts to make better decisions on various of tasks.

TopoMap [3] is a famous topology-based multidimensional projection method for the visualization of high-dimensional data. It guarantees to preserve topological structures during the dimensionality reduction process, therefore more precisely presenting what takes place in the high-dimensional space. TopoMap uses Rip filtration over the inputs, preserving 0-homology (Betti-0) topological persistence. On implementation, TopoMap generates a Euclidean Minimum Spanning Tree (EMST) with Dual Tree Boruvka Algorithm [11] to make topological structure for the data in original high-dimensional space, and then use multiple designed point placement rules to project the EMST as a 2D layout.

*e-mail: yx2606@nyu.edu
†e-mail: jm7828@nyu.edu
‡e-mail: hl2979@nyu.edu

However, TopoMap still has several shortcomings. First, TopoMap, like some of the other topology-based methods, can only generate static and fixed layouts for the given datasets, so the point placing rules can sometime give an unsatisfying result to the data analysts. Second, unlike UMAP [12] and tSNE [15], TopoMap only focus on preserving 0-homology topological persistence, which makes it hard to generate concentrated clusters with clear gaps. Third, since TopoMap is generated globally with all the given data, it may ignore the relationships between data from the same group in a local scale. Forth, TopoMap uses all dimensions of the data to generate the layout, while only some of the dimensions matterin constructing the manifold. Filtering out the dimensions of interest is necessary for us to understand the data distribution in high-dimensional space.

Aiming to solve the problems above, we introduce a novel way to make an interactive system for TopoMap, meanwhile adding features for multi-scale views and better clustered layouts. To make the interaction part, we choose to implement our work via Python and D3.js [8], which are widely used to realize interaction visualization systems. With more details, we first translate the original TopoMap generation part from C into Python to better fit the backend of our system. Then we add Dim-Map, Scale-Map and Force Directed TopoMap to the system. The Dim-Map is a TopoMap generated with dimension information in order to provide a 2D-visualized selection interface to the user. The Scale-Map is a localized version of TopoMap, designed to find related adjacent points in a local view. The Force Directed TopoMap is an interactive TopoMap, with the connection between points from the same group. Applying these techniques, user can select the dimension of interest and adjust the TopoMap layout interactively, and also get important information upon the points of interest.

The contribution of our work can be summarized as follows:

- We implement TopoMap in Python by generating EMST with Boruvka Algorithm.
- We propose Dim-Map, a TopoMap for input dimension selection. By making visualization for dimension relationships, our approach can help user find the dimension of interest for specific clusters.
- We provide Scale-Map, a localized TopoMap. By selecting one group of elements as an emphasis, Scale-Map will provide comparisons between local and global TopoMap projections.

- We combine TopoMap with Force Directed Map as Force Directed TopoMap to solve the problem of poor clustering and unclear connections in original TopoMap. We show that Force Directed TopoMap is an efficient method to explore relationships within and between clusters in layouts.

2 RELATED WORK

In order to show the context of our work more clearly, we categorize the related works into two main topics: Dimension Reduction and TopoMap, and the Interactive Visualization.

2.1 Dimension Reduction and TopoMap

Dimension Reduction has long been an important tool for data analysts to understand the distribution and features of high dimensional data. This reduction process performs projection from high-dimensional space into 2-D or 3-D space which can be visualized by human, meanwhile guarantees specific geometric persistence so that people can get useful information from those unchanged features. These guarantees can be divided into two categories, one is persistence on linear features in the whole space, which is used by Principle Component Analysis (PCA) [17]; the other is persistence on topological structures, which is used by Isomap [1], tSNE, UMAP, TopoMap, etc. In this paper we mainly focus on the methods based on topological persistence, especially TopoMap.

Among the topological dimension reduction methods, Isomap is often considered as the pioneer. Isomap detects neighborhood around data points by setting radius thresholds or using the principle of K-nearest neighbors [13], and generates adjacency map based on geodesic distance. The topological structure is then represented as a manifold by this adjacency map since it shows the local relationships of all the data points. As a variant of Isomap, Landmark-Isomap [18] introduces landmark points as a subset for time-consuming computation, which increase scalability and efficiency. In another research, graph tearing procedure is used to preserve essential loop structures for circular manifolds like cylinders or torus [10]. These methods are based on the idea of capture 1-dimensional homology structures in a given dataset, but hardly take the structures of 0-homology groups in to account.

T-SNE is one of the most widely used techniques for dimension reduction and visualization. It is a variation of Stochastic Neighbor Embedding (SNE) [6], and shows much stronger ability on showing different but related low dimensional manifolds. It uses random walks on adjacency graphs, allowing the implicit structure of all the data to influence every sub-structures displayed. UMAP (Uniform Mainfold Approximation and Projection) is also among the most popular techniques. It uses complex theoretical framework base in Reimannian geometry and algebraic topology, thus shows even better performance than t-SNE for visualization quality and run time efficiency. However, T-SNE and UMAP use more global topological information and more complex optimization methods than Isomap to generate their layouts, which brings much more burdens to computational resources.

TopoMap is a topological dimension reduction method that is different from the methods mentioned above in the aspect of geometric guarantees. Based on Rips filtration, it aims to find a topological structure that preserves 0-homology persistence. In other words, it tries to keep the connection relationships between data points in a neighborhood. And it also keeps the distance between connected points during the projection procedure. These features helps analysts confidently explore high-dimensional data by visualizing the groups of data that tightly connected in high-dimensional space. But they also lead to shortcomings like information lost between unconnected points, inefficient usage of visualization space, and poor scaling abilities.

2.2 Interactive Visualization

Data visualization is the practice of mapping data into visual media to assist users in exploring data or communicating about them to other uses. Therefore interaction techniques is of vital importance in visualization tasks, for it improves the user experience in data selection and comparison.

There are numbers of applications for interactive visualization designing on different platforms. In Python, Matplotlib [7] is almost the most widely used interactive visualization tool for data analysts. Matplotlib can easily make interactive figures that can zoom, pan or update. Meanwhile, D3.js is a data-driven visualization engine written in Javascript, using HTML, SVG and CSS. D3 provides lots of APIs for users to manipulate their plots directly, thus it is much more flexible for the design of interactive mechanisms. Another approach is the MATLAB [14]. As a successful software for mathematical programming, MATLAB has its own interactive visualization systems, and is famous for its tremendous performance on accuracy and speed, which makes it popular among the data scientists in companies and research institutions.

Except for choosing a good design application, users also have to focus on selecting a good interactive method that meets their need for the visualization system. Since there are too many useful techniques for different tasks, we only introduce the techniques that are used in our research here.

The first method we use is brushing. Brushing means that user can use their mouse to create a box for data selection in a given chart, so the visualization system generates results with those data in the region of interest. Brushing can not only remove the distracting data from the view of users, but also alleviate the burdens on computation resources. Moreover, brushing can also help making contrast between different selected groups of data, or showing the transformation process in layout when selection box moves.

The second method we use is the Force Directed Map [5]. Force Directed Map is mainly used to show the relationships between the data points in a network. By adding compulsory force to the points and attractive force to the edge, Force Directed Map simulates the physical procedure of particles, making a fantastic model for clustering and categorizing. Meanwhile, Force Directed Map can provide interaction mechanisms such as dragging and zooming, so users can change the layout as they want to have a better visualization performance.

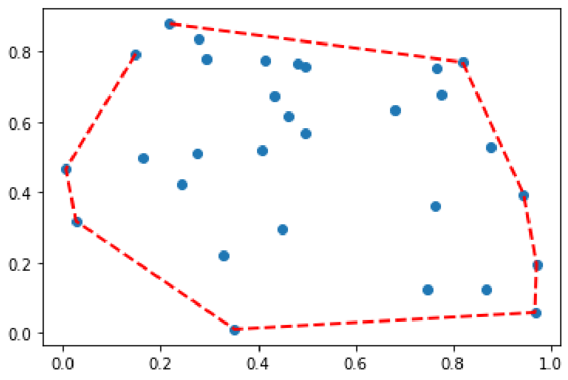

Figure 1: Convex Hull

3 METHODOLOGY

To build the interactive multi-scale visualization system base on TopoMap, we first implement the original TopoMap, which is written in C, into Python. Then we add features like Dim-Map, Scale-Map and Force-Directed Map into the system.

3.1 Re-implementation of TopoMap in Python

For the major functions and algorithm components, we have reconstructed them in Python style and optimized with efficient data structures based on Python libraries.

Disjoint set is a fundamental data structure to compute union of neighbours. We study the implementation of original CPP code and use the indexed list to save the time for searching and comparing.

For the utilities of geometric structures, we reconstruct them in Python class. We specify each class variable and its type to avoid different inputs. To compute the convex hull, we use the ConvexHull method from Scipy [16]. It is based on quick hull algorithm. To transfer data for the function, we reload our point type data into numpy array. Figure 1 shows a basic implementation of it.

In the TopoMap procedure, we implement the Python function correctly as the algorithm showed in the paper. To compute the Euclidean minimum spanning tree, we use the emst function from the Python mlpack [2] library. This function is based on fast Euclidean minimum spanning tree using a Dual-Tree Boruvka algorithm with Kd-trees to find near neighbours. For the general cases of high-dimensional points, it is the most efficient way. The transformation and union computations are based on list with point indexes. A sample mapping is shown in Figure 2.

3.2 Dim-Map

To construct Dim-Map, we combine all the data elements in the same dimension as a new vector, which represents the dimensional information of the given dataset. To more extent, we take the columns of the dataset as data vectors in the dimensional space. Then we generate TopoMap with the dimension data to make an interactive interface for the users to make selection on the dimensions of interest.

In the procedure of generating TopoMap, several metrics can be chosen, including Euclidean distance, Cosine distance, etc. The metric we choose relies heavily on the properties of given datasets. For example, on datasets with various dimensions due to null values, Euclidean metrics are not recommended because some vectors cannot be valued in specific dimensions. Instead, we would better use Chebyshev distance here. And on datasets whose dimensions each ranging different scale levels, cosine distance is much better since it is not sensitive to the scale variations.

Using Dim-Map, we can explore dimension-wise information that are not clearly displayed in the original TopoMap: data points in Dim-Map represents the statistical feature of different dimensions, and the distance between them are the similarity between different dimensions; The clusters in Dim-Map shows the group of dimensions that are high correlated, thus they possibly generate the same distribution of data in the original dataset.

We also provide brushing tools to interactively select the dimensions of interest, which is showed in Figure 3. Every time a brushing box is created and changed, the selected dimensions will be sent to the engine of main visualization system and used to build a filter on the dataset. Then the main system applies this filter, computes EMST and layout positions of data points, and re-renders TopoMap. Due to the cost of computing a new EMST, the system may have latency in each updating process, but it does not affect the user experience since the computing is very time efficient.

3.3 Scale-Map

In Scale-Map we focus mainly on the local projections. Local projections should be made to preserve important information in a small view, thus we consider to use a new criterion for judging the range of locality. With the insight that the adjacent points within a manifold in high dimensional space only have relationships in a few the dimensions, we make the locality here a n-dimension range regulated by a certain group of data, where n refers to the number of features of the dataset. With more detail , we first choose a specific group of data, calculate its maximums and minimums in all dimensions, and filter out all the points that are within this range. All those points are considered as the neighbors of the selected group and are included in the Scale-Map. In contrast to simply calculating the Euclidean distances as the criteria of neighbors in the whole space, we keep more related adjacent points in a local view by using dimension-wise criteria of ranges.

For example, there are 64 features of the famous digit

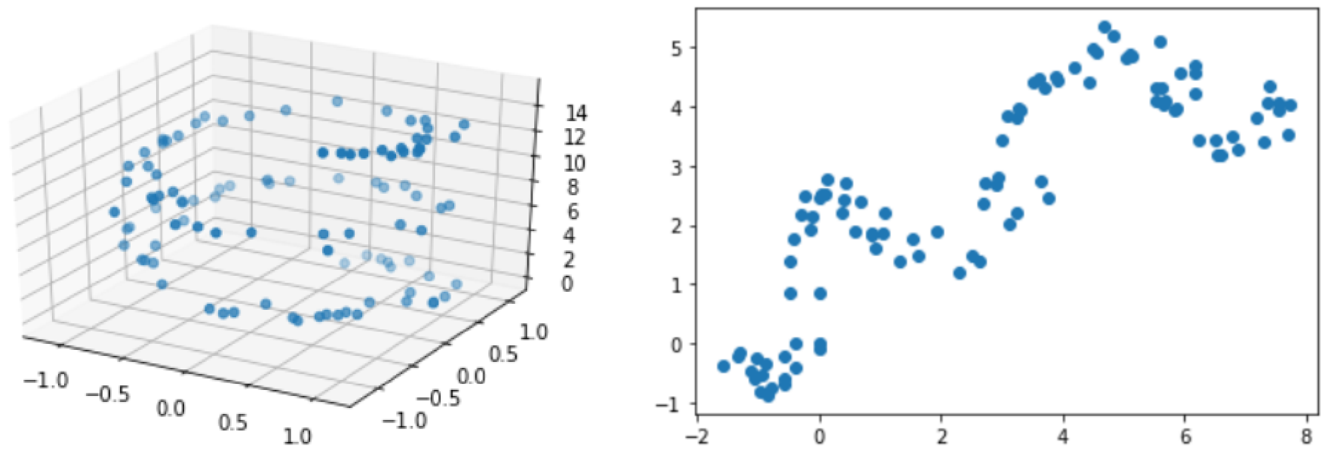

Figure 2: Sample Mapping

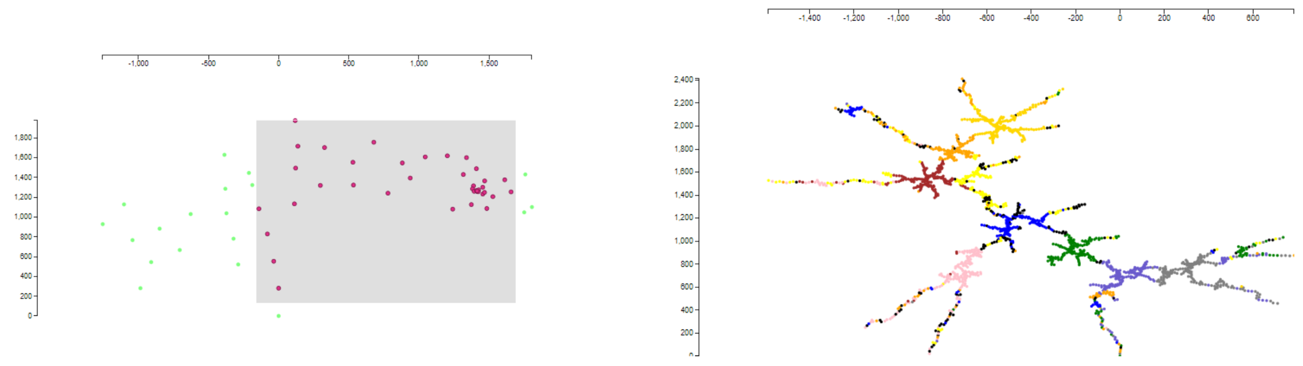

Figure 3: Dim-Map: Left is the brushing interface for users, Right is the layout of selected dimensions

dataset. In a Scale-Map we first pick an emphasis digit (e.g. digit 1). After this we create a 64-dimension range using digit 1. We first pick all data points of digit one, calculate the 64 maximums and minimums as the range. We then use this range to filter out all points in this range, simply compare if all 64 features of a data point are within the corresponding maximums and minimums, and use this new filtered subset as the data source to project using TopoMap.

3.4 Forced Directed TopoMap

The output we received from TopoMap projection algorithm is an array of 2D point coordinates, giving up all information about connections. Therefore there is no connection showed in the layout. However, it is really important to emphasize the connections since the 0-homology persistence plays the pivot role in generating TopoMap, and the connections are necessary for analysts to find which two points are indeed the nearest in high dimensional space intuitively. Fortunately, it is easy for us to retrieve the connection relationships during the Euclidean minimum spanning tree (EMST) generating process.

Since we add connections in the layout with the edges in EMST, another problem arises: If we use interactive methods like dragging or zooming in this layout, the whole TopoMap will just act like a plate of pasta because of the sparsity of connections. The unstable movement of points and edges will greatly distract the data analysts, and lead to poor performance on clustering.

To solve the problems above, we get an inspiration of introducing Force Directed Map into TopoMap. We not only preserve the connections between the adjacent points in the high dimensional space, but also introduce connections between all the points from the same group. We set weights for all these connections as attractive force, and add compulsory force between all the data points. Finally all the points in the layout will be placed into balanced positions by a optimization algorithm, and the optimization process will be revoked every time we change the position of the points. Though we may lose some information about the exactly distance of points in high dimension space in Force Directed TopoMap due to the optimization procedure, we think that the exact distance is much less useful than connection in topological projections.

As to the weight of connections, we have 2 different approaches here:

- Maintain the weights of connections in the minimum

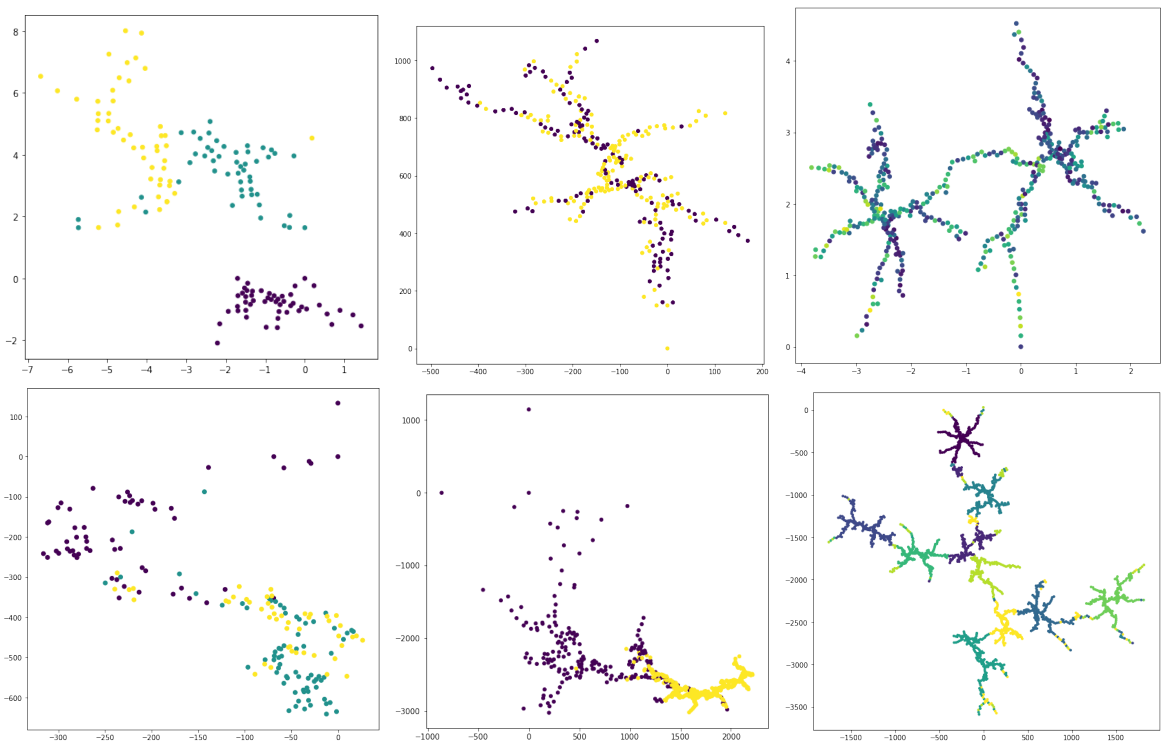

Figure 4: TopoMap Projection on Iris, Heart, Diabetes, Wine, Cancer and Digits

spanning tree.

- Use other similarity metrics based on the points connected, such as Cosine Distance.

And to get a better performance of clustering, the weights of connections between points in the same group should be set larger than those between points in different groups.

4 Results and Analysis

4.1 TopoMap in Python

We reproduced the result of TopoMap that is re-implemented in Python on Iris [4], Heart [4], Diabetes [4], Wine [4], and Digits [9]. The layouts are showed in Figure 4.

4.2 Dim-Map

We generated two groups of layouts as Figure 5 and 6 on Digit dataset to show the effects of using Dim-Map as an interactive visualization tool for TopoMap, and giving simple interpretations for the layouts with dimension selection.

To get a global scale of view, we can select most of the dimensions with a large brushing operation on the Dim-Map as Figure 5. And the difference between the first row and the second row is the position of the brushing window. Comparing the two layouts, we can find that the group in black is more concentrated in the second row layout, which indicate that the dimensions selected in the second graph contains more dimensions that have the local information for the black group. Those groups which only changes a little relies more heavily on the unchanged dimensions. In this way we can find the relationships between specific dimensions and the data clusters.

Figure 6 shows Dim-Map with a smaller brushing window. The window movement showed in Figure 6 has almost the same effects as in Figure 5. In addition, we can compare the two groups of layouts together, finding that the size of brushing window can affect the construction of clusters. More data with more dimensions can make more clusters, which fits human's intuition about the formation of data clusters in high dimensional space. The most interesting part we think is the second row in Figure 6, which generated clusters with only a small group of dimensions. We think it owes to the sparsity of the selected dimensions in Dim-Map, which means the selected dimensions have less similarity in the high dimensional space, thus contains more information to categorize the given data.

4.3 Scale Map

Figure 7 is the comparison between result of original TopoMap and Scale-Map with emphasis on digit 1. In both graph data points that represent ”1” in Digit dataset are marked larger than other points and are in the color of dark blue. We call this selected group of points as the anchor points.

Comparing the two subplots in Figure 7, we found that the there are similarities on the emphasis digit. In both plots the anchor cluster (”1” cluster) is separated as 2 parts. However, the cluster between the 2 parts are different: Cyan in original

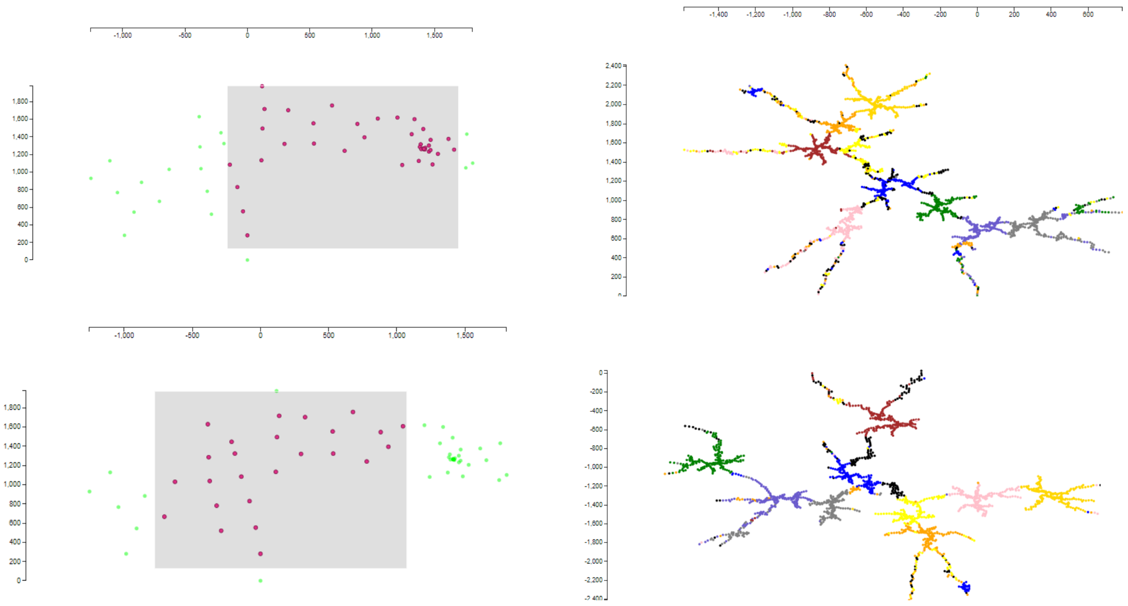

Figure 5: Dim-Map: Two brushing windows of the same size but with different dimension points contained

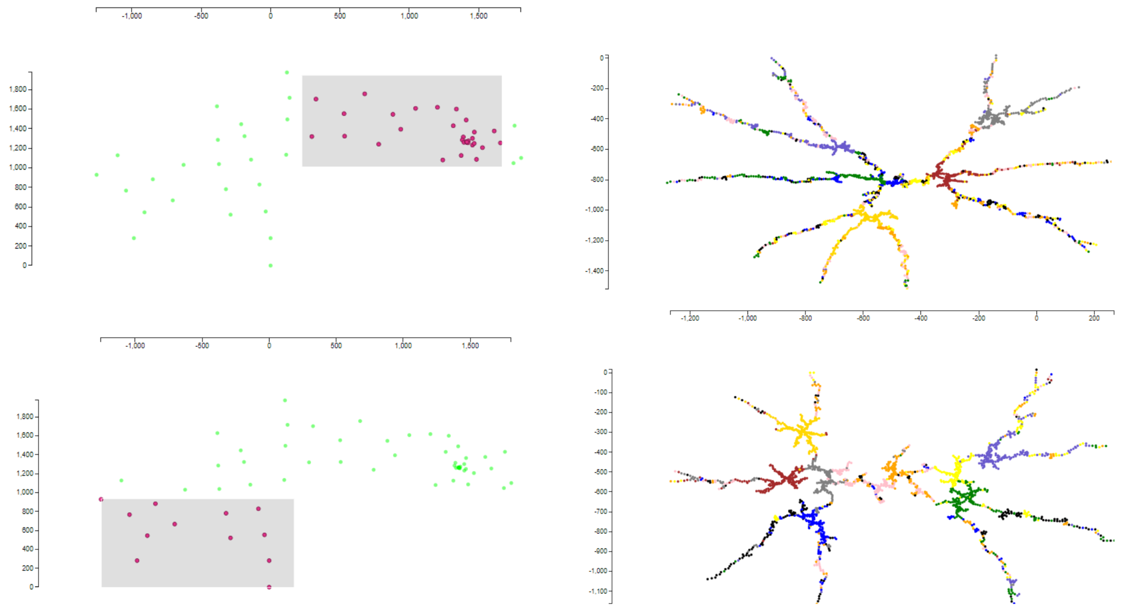

Figure 6: Dim-Map: Two smaller brushing windows of the same size but with different dimension points contained

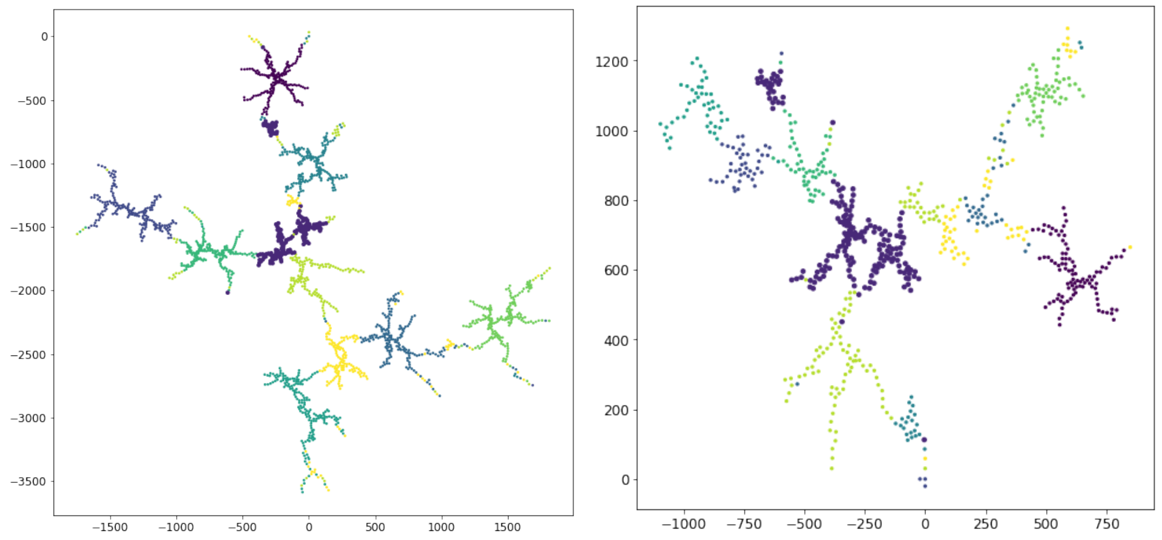

Figure 7: Scale-Map Emphasis on Digit ”1”

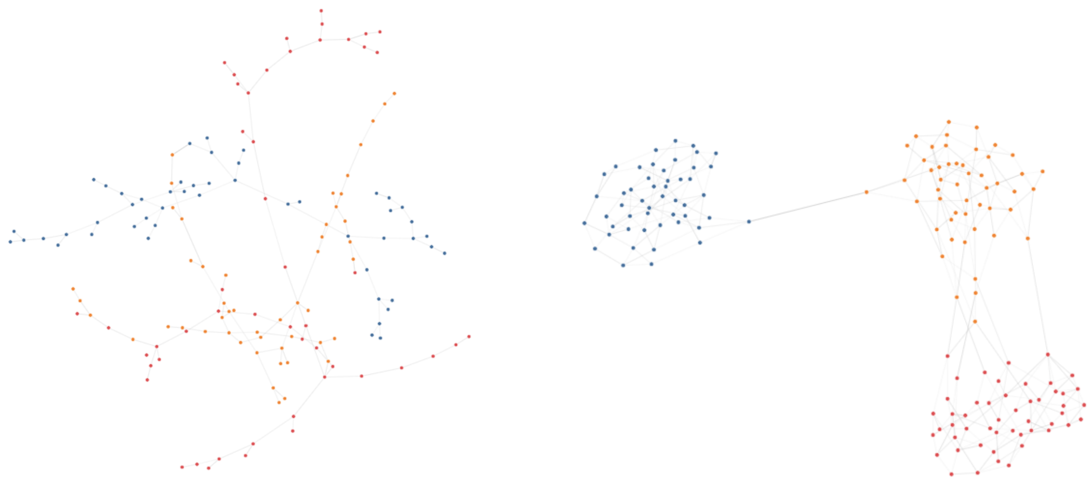

Figure 8: Force Directed Map on Iris, MST connection only vs. Group connection added

plot but green in Scale-Map. The reason of the difference is possibly caused by the scaling procedure. In the global view, the point which is closest to the small purple cluster is in the green group, but it is not included in the local view of the anchor points, which indicate the effects caused by folding and distortion in high dimensional space. The Scale-Map always shows a tighter neighborhood for the anchor group of points.

4.4 Force Directed TopoMap

We generated Force Directed Map with EMST of TopoMap and Force Directed TopoMap on Iris dataset, and the layouts are showed in Figure 8. The result is fascinating: points are more tightly clustered, and the connection between clusters are emphasized. We can know exactly how much connections are there, and which points act as the boundary of a specific cluster.

However, there are some problems when Force Directed TopoMap is applied to large datasets, such as too many edges and bad clustering. To solve these problems, several adjustments should be made: The weight of the edges within a group have to be increased, and the canvas should be larger. With these adjustments, the layouts can have the same properties as those generated by a small dataset.

5 Conclusion

In this paper we introduce an interactive multi-scale visualization system based on TopoMap, including Dim-Map, Scale-Map and Force Directed TopoMap. This interactive system shows its advantage in exploring dimensional information, keeping neighborhood relationships, and emphasizing clusters and their connections. Since the system here is not only limited to TopoMap, applying the system to other visualization methods will also be an interesting topic for future researches.

Acknowledgments

This work is submitted as the a research project to CS-GY/CS-UY 3943 Visualization for Machine Learning in 2022 Spring.

References

[1] M. Balasubramanian and E. L. Schwartz. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002.

[2] R. R. Curtin, M. Edel, M. Lozhnikov, Y. Mentekidis, S. Ghaisas, and S. Zhang. mlpack 3: a fast, flexible machine learning library. *Journal of Open Source Software*, 3(26):726, 2018.

[3] H. Doraiswamy, J. Tierny, P. J. Silva, L. G. Nonato, and C. Silva. Topomap: A 0-dimensional homology preserving projection of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):561–571, 2020.

[4] D. Dua and C. Graff. UCI machine learning repository, 2017.

[5] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[6] G. E. Hinton and S. Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15, 2002.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

[8] A. Jain. Data visualization with the d3. js javascript library. *Journal of Computing Sciences in Colleges*, 30(2):139–141, 2014.

[9] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[10] J. A. Lee and M. Verleysen. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing*, 67:29–53, 2005.

[11] W. B. March, P. Ram, and A. G. Gray. Fast euclidean minimum spanning tree: algorithm, analysis, and applications. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 603–612, 2010.

[12] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[13] E. A. Patrick and F. P. Fischer III. A generalized k-nearest neighbor rule. *Information and control*, 16(2):128–152, 1970.

[14] S. M. Toolbox et al. Matlab. *Mathworks Inc*, 1993.

[15] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[16] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2

[17] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[18] L. Yan, Y. Zhao, P. Rosen, C. Scheidegger, and B. Wang. Homology-preserving dimensionality reduction via manifold landmarking and tearing. *arXiv preprint arXiv:1806.08460*, 2018.