An Introduction to the Mapper Algorithm and Its Open-Source Implementations

Youjia Zhou

Agenda

Part I

- The Mapper Algorithm
- Kepler Mapper

Part II

- Giotto-TDA
- Mapper Interactive

Part I

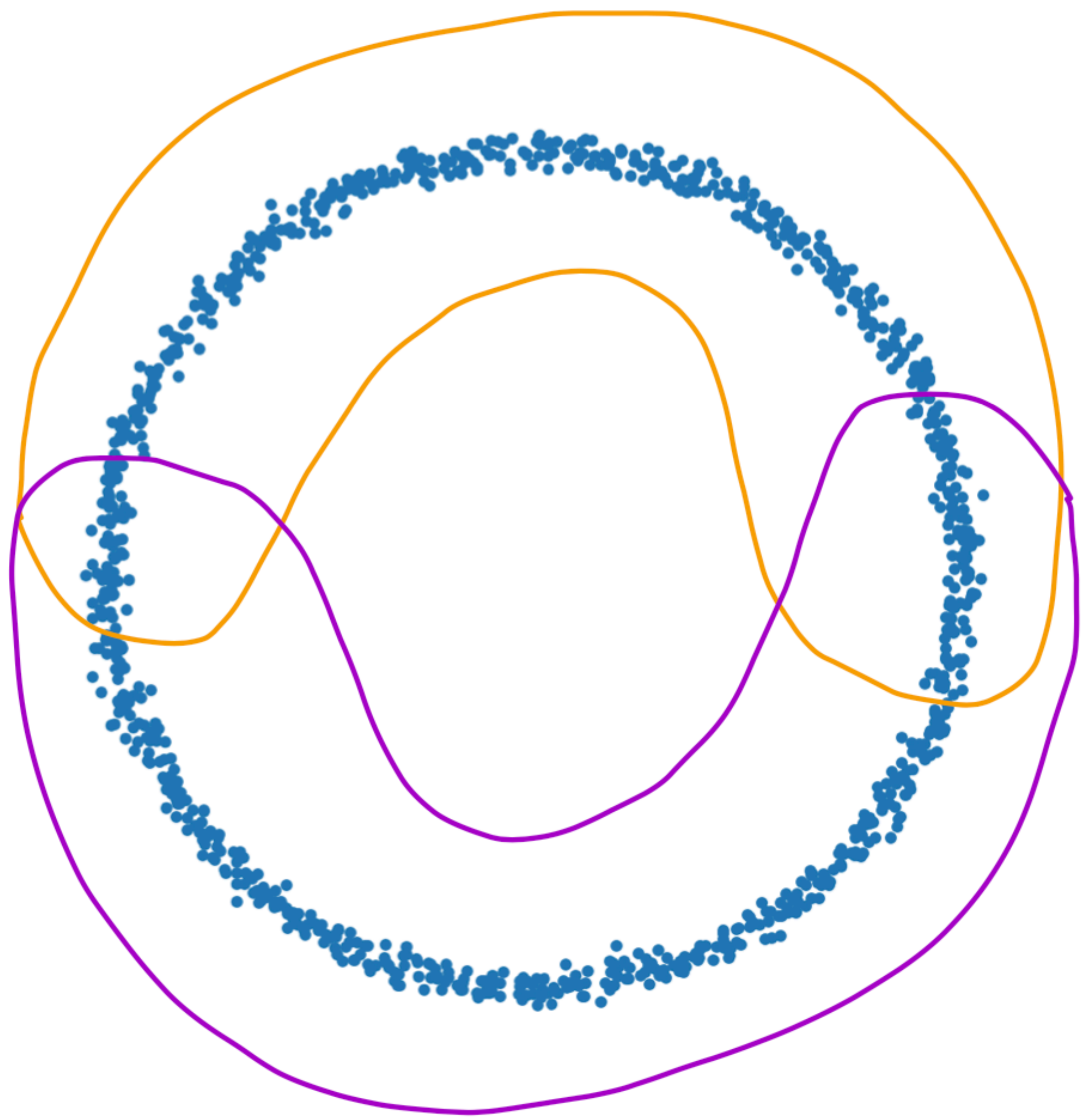
Mapper Algorithm

- First introduced by Singh et al. in 2007 [1].
- A popular framework from topological data analysis for **extracting topological summaries** of high-dimensional datasets.
- “Partial clustering of the data guided by a set of functions defined on the data”, which captures both **clusters** and **cluster relations**.
- Combining **dimensionality reduction** with **graph visualization**
- Qualitative understanding of high-dimensional point cloud data through visualization

Mapper Algorithm: Cover and Nerve

Given a high-dimensional point cloud $\mathbb{X} \subset \mathbb{R}^d$

- A *cover* of $\mathbb{X}$ is defined as a set of open sets in $\mathbb{R}^d$, $\mathscr{U} = \{U_i\}_{i \in I}$ such that $\mathbb{X} \subset \cup_{i \in I} U_i$ (I being the index set).



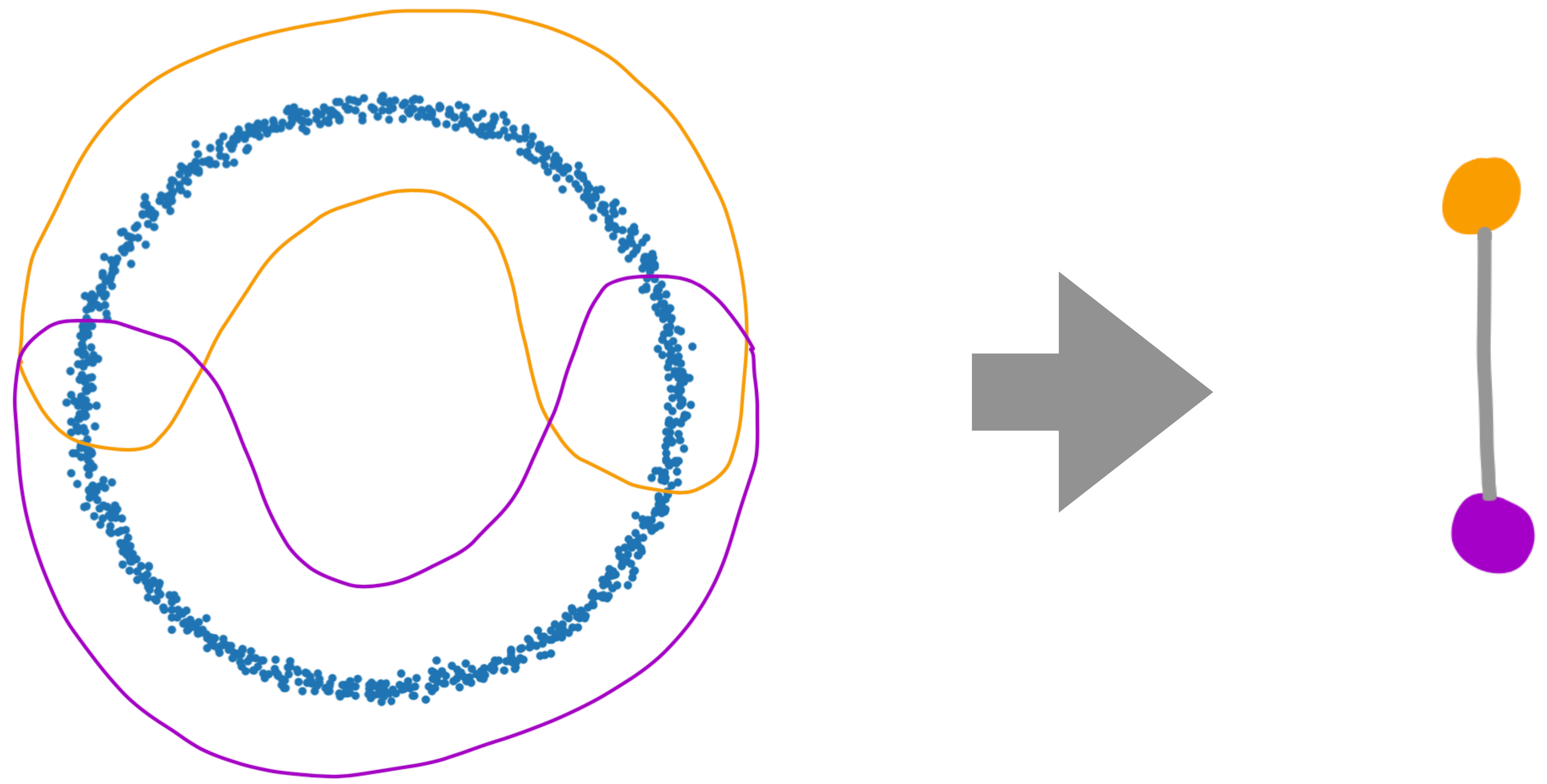
Mapper Algorithm: Cover and Nerve

- Given a cover $\mathscr{U} = \{U_i\}_{i \in I}$ of $\mathbb{X}$, let $\mathscr{N}(\mathscr{U})$ denote the simplicity complex that corresponds to the *nerve* of the cover $\mathscr{U}$

$$\mathscr{N}(\mathscr{U}) = \{\sigma \subset I \,|\, \cap_{i \in \sigma} U_i \neq \phi\}$$

- The 1D nerve of $\mathscr{U}$, denoted as $\mathscr{N}_1(\mathscr{U})$, is a graph.

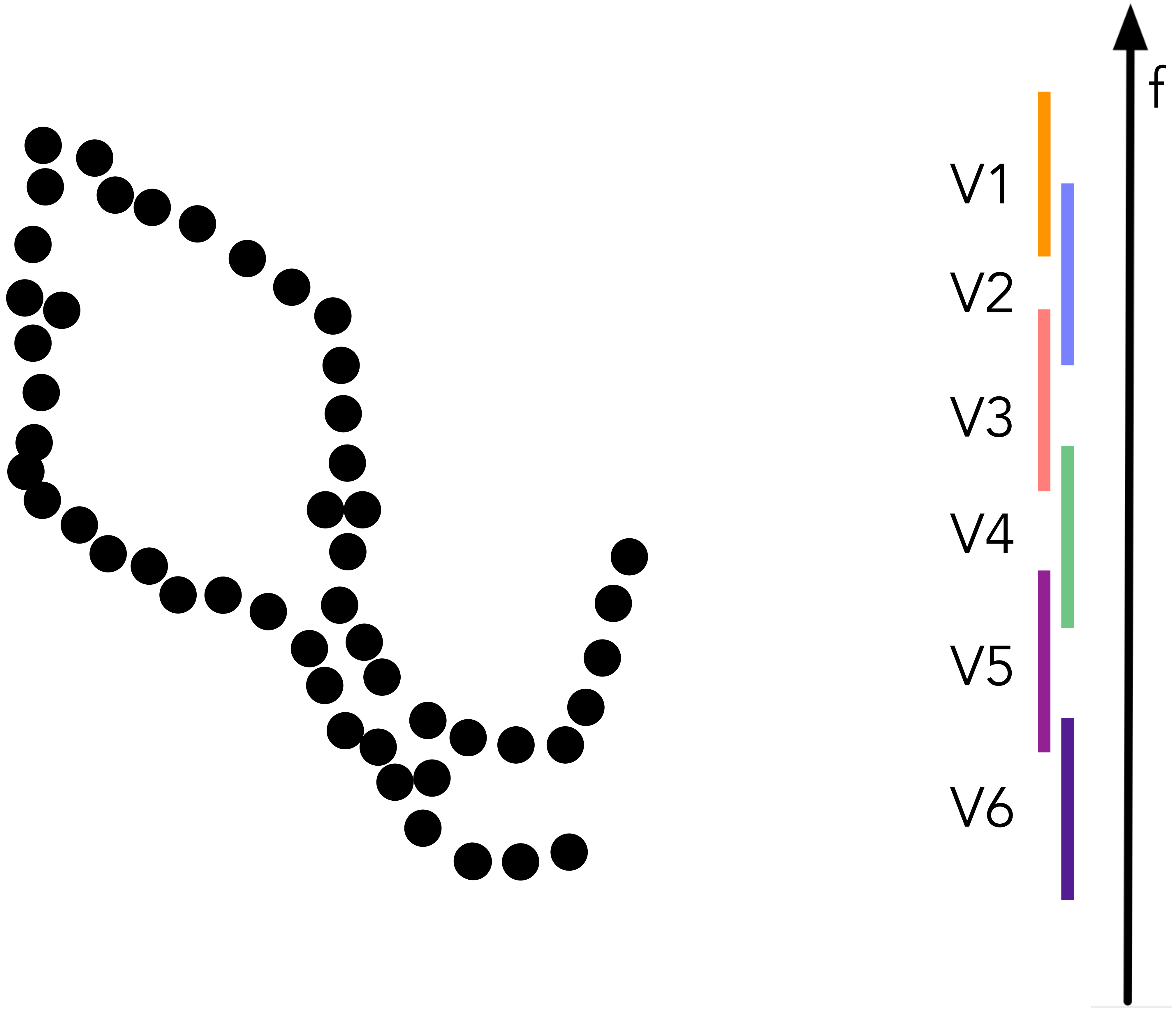
- Each node $i \in I$ in $\mathscr{N}_1(\mathscr{U})$ represents a cover element U_i, and there is an edge between $i, j \in I$ if $U_i \cap U_j$ is nonempty.



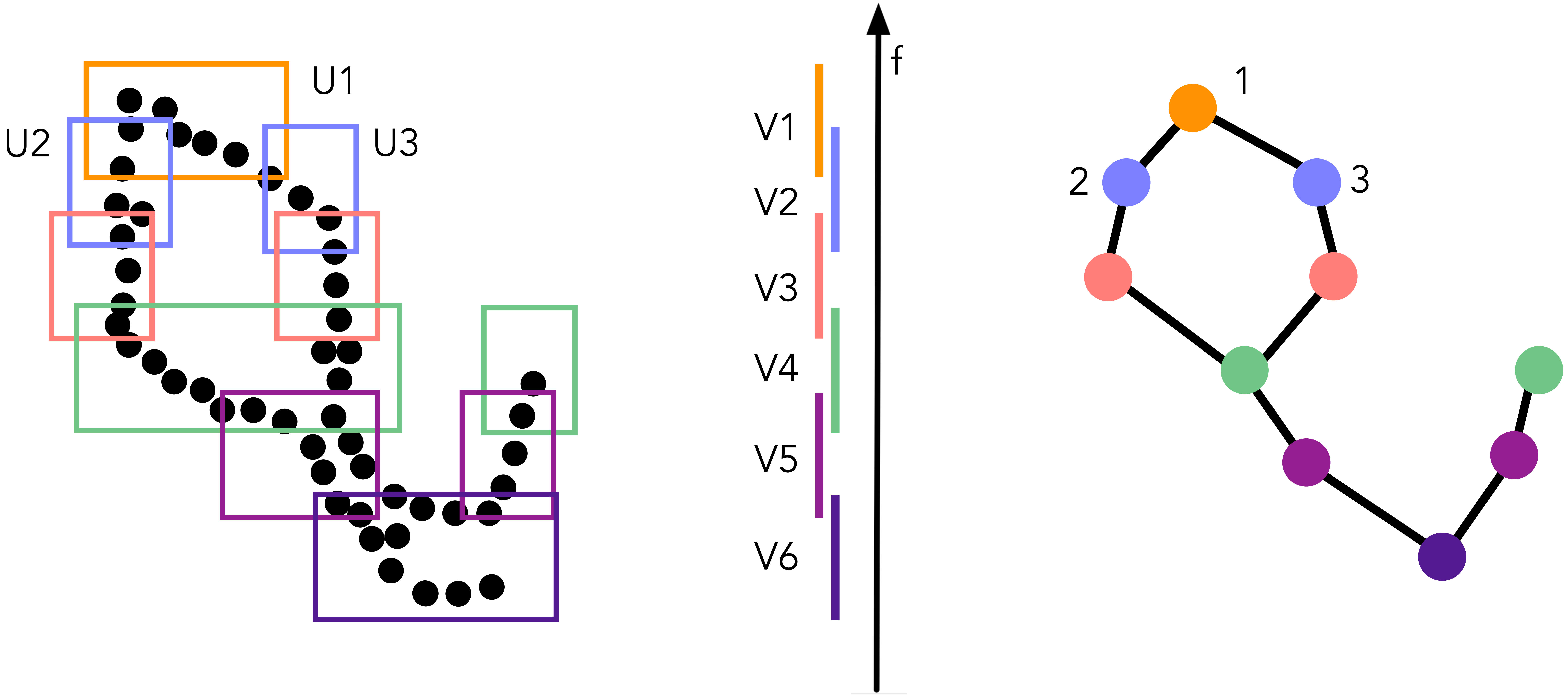
Mapper Algorithm: Filter Function and Clustering

- In the classic mapper construction [1], a set of scalar functions $f : \mathbb{X} \to \mathbb{R}$ (referred to as the filter functions) is used to guide obtaining a cover.
- Given a cover $\mathscr{V} = \{V_k\}$ $(1 \leq k \leq n)$ of $f(\mathbb{X}) \subset \mathbb{R}$, such that $f(\mathbb{X}) \subseteq \cap_k V_k$, we obtain a cover $\mathscr{U}$ of $\mathbb{X}$ by considering the clusters induced by points in $f^{-1}(V_k)$ for each V_k as cover elements.
- The 1D nerve of $\mathscr{U}$, denoted as $\mathscr{M} = \mathscr{M}(\mathbb{X}, f) := \mathscr{N}_1(\mathscr{U})$, is the mapper graph of $(\mathbb{X}, f)$.

Mapper Graph



Mapper Graph



Mapper Algorithm: Input, Output, Parameters

Input:

- Point cloud data X
- Distance metric on the point cloud d_X
- Filter functions (lens) $f : \mathbb{X} \to \mathbb{R}^d$

Output:

- The mapper graph G_X: a summary of the data as a graph (or a simplicial complex)
- Visualization, statistics and ML

Parameters:

- Parameters for the chosen clustering algorithms
- Filter functions f_1, f_2, …, etc.
- Covering parameters
- Visualization parameters: color functions, etc.

Mapper Algorithm: Clustering Details

- Almost any clustering algorithm can be used
- Assume there is a notion of distance (metric) between a pair of points in the data domain (distance can be computed or provided)
- Clustering is equivalent to **a notion of connected component** in the point cloud setting
- Commonly used clustering algorithms:
  - Density-based spatial clustering of applications with noise (DBSCAN) [2]
  - Single-linkage clustering
  - K-means, etc.
- Desirable properties:
  - Not restricted to Euclidean distance; can take distance matrix input
  - Do not require specifying the number of clusters beforehand

Mapper Algorithm: Parameters for Covering

- Number of intervals: m
- Percentage of overlap: p
- Uniform Cover: intervals have equal length, equal percentage of overlap
- Balanced Cover: approximately the same number of points are contained in each cover interval

Mapper Algorithm: Choose Filter functions

- A filter function can be a given prior, e.g. one feature of the point cloud
- It can also be derived from the properties of the point cloud it self
  - Density estimation

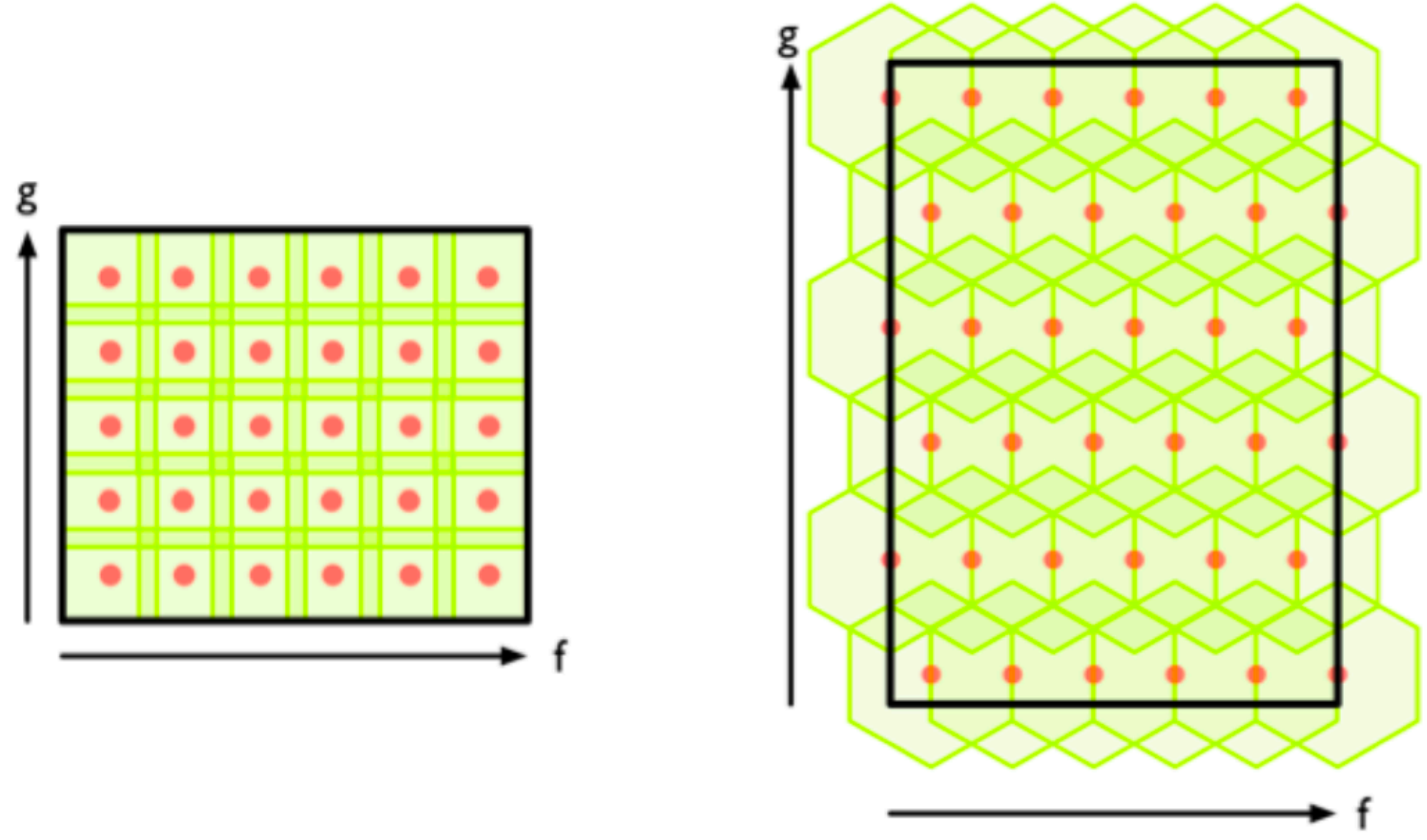
$$f_\epsilon(x) = C_\epsilon \sum_{y \in X} exp \frac{-d(x,y)^2}{\epsilon}$$

  - Eccentricity

$$E_p(x) = \left(\frac{\sum_{y \in X} d(x,y)^p}{N}\right)^{1/p}$$

  - Distance to a point in the data
  - Graph laplacians
  - …

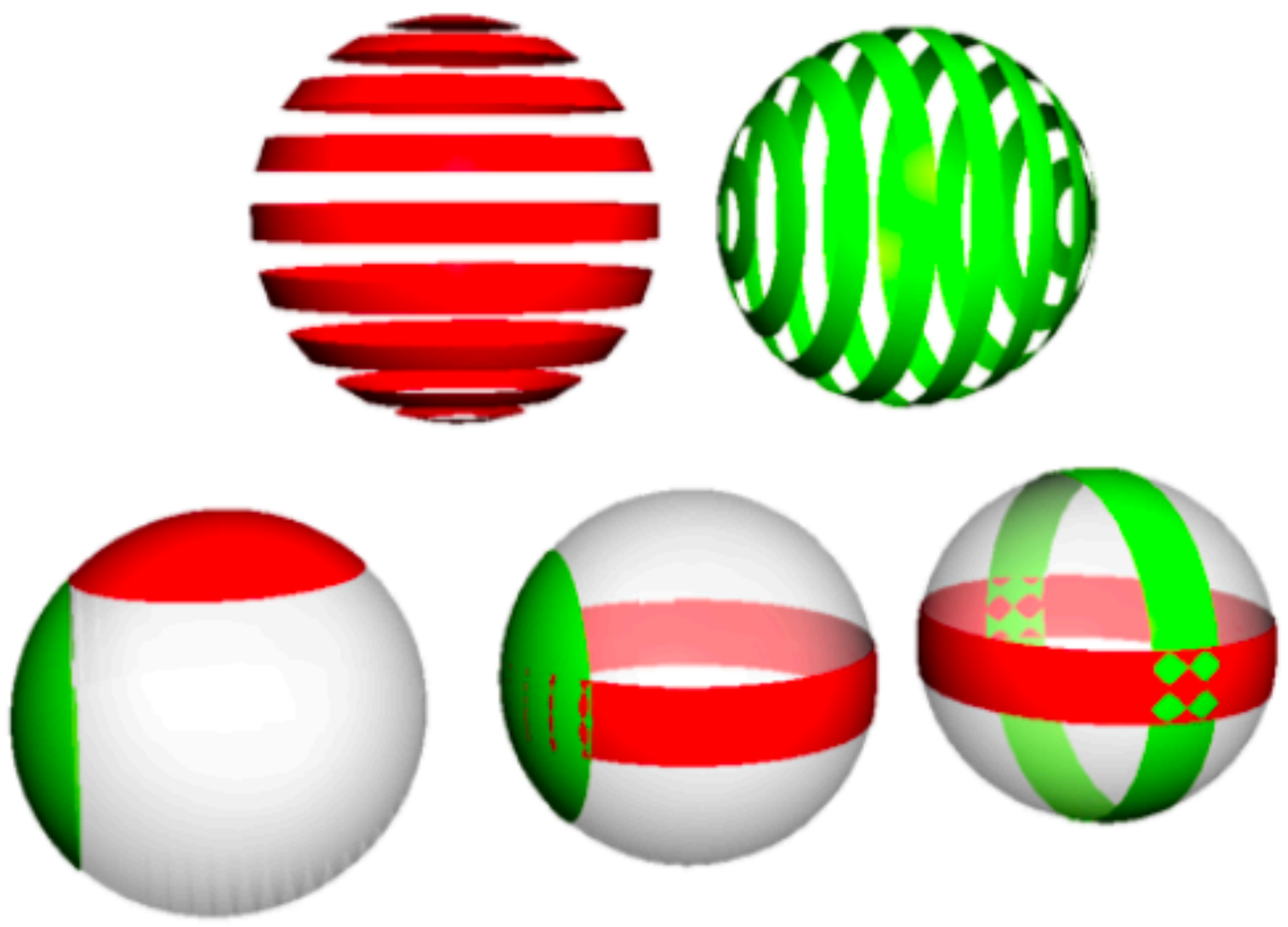
Mapper Algorithm: 1D vs 2D Mapper



source: [Singh et al. 2007]

- 1 vs. 2 filter function(s)
- 1D intervals vs. 2D intervals
- For 2D Mapper, the covering of the domain of the function is no longer by intervals, instead, by rectangles or other geometric shapes, etc.

Mapper Algorithm: 2D Mapper Example



0	0	1	1	0	0
0	1	1	1	1	0
1	1	2	2	1	1
1	1	2	2	1	1
0	1	1	1	1	0
0	0	1	1	0	0

source: [Singh et al. 2007]

State-of-the-art Tools

- **KeplerMapper [3, 4]**
- **Giotto-TDA [5]**
- **Mapper Interactive [6]**
- Gudhi [7]
- Hyppo-X [8]
- PhenoMapper [9]
- …

Kepler Mapper

- A library implementing the Mapper algorithm in Python
- Provides control over the clustering algorithm, scaling algorithms, covering scheme, and nerve scheme
- Scikit-Learn-API-compatible clustering and scaling algorithms
- Interactive capabilities for visual exploration
- Installation
  - Install with pip: `pip install kmapper`
  - Install from source:
    ```
    git clone https://github.com/MLWave/kepler-mapper
    cd kepler-mapper
    pip install -e .
    ```

Kepler Mapper: kmapper.KeplerMapper

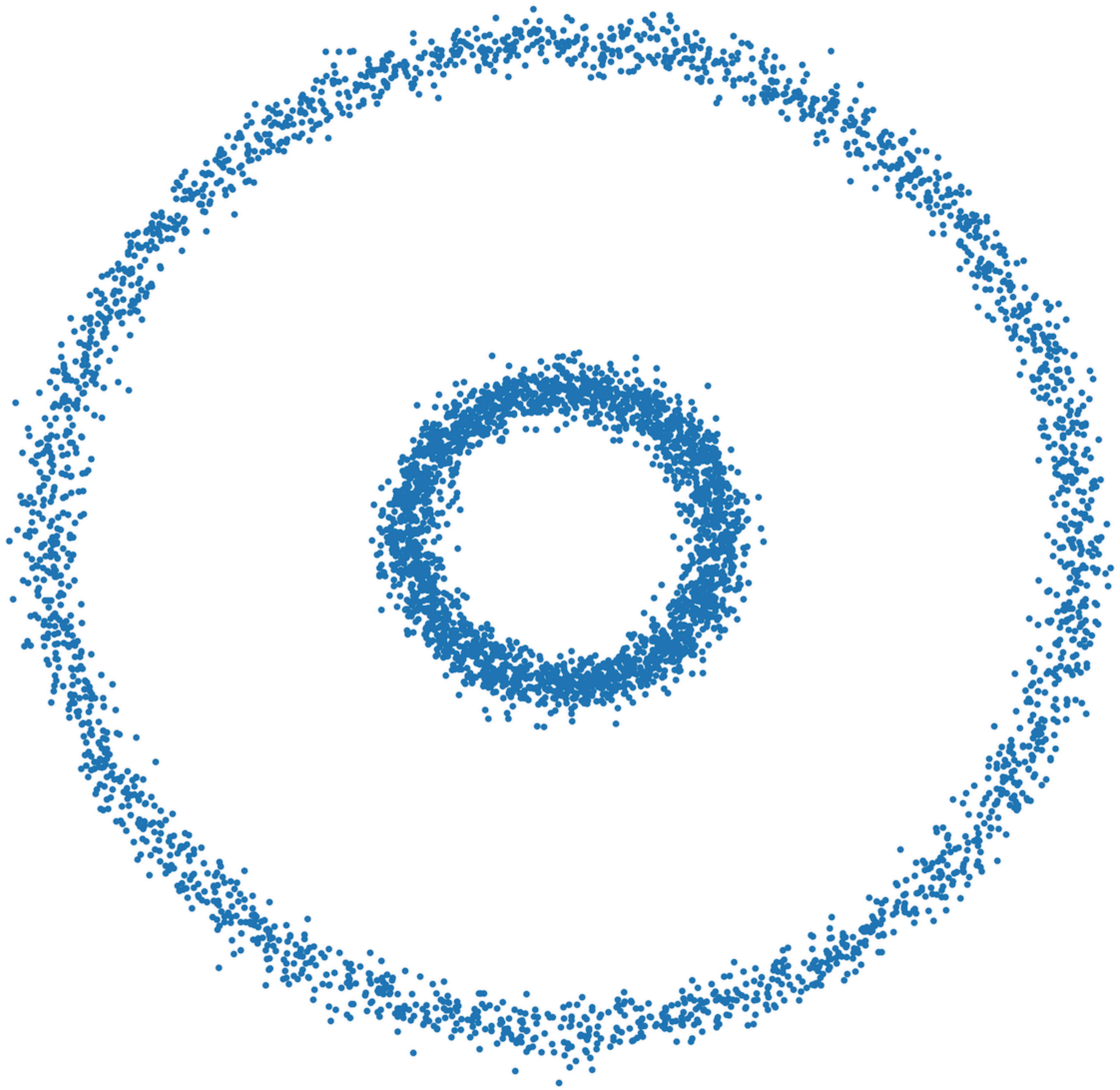
The class used to build topological networks from (high-dimensional) point cloud data

- Fit a filter function (lens) to a dataset and transform it.
- Map the filter function with overlapping intervals, cluster the points inside the interval, and connect two clusters with an edge if they intersect with each other.
- Visualize the network with HTML and D3.js

Example: two circles

- Data: 2D point cloud

```
[[ 0.79587693  0.60998219]
 [ 0.66950864 -0.74779242]
 [ 0.21097122 -0.18433569]
 ...
 [ 0.11613233 -0.29637248]
 [ 0.913603   -0.46775118]
 [ 0.26216396 -0.04880798]]
```



Kepler Mapper: Choose a lens

- fit_transform(X[, projection, scaler, distance_matrix]). The projection parameter can be:
    - a list of dimension indices
    - a string from [“sum”, “mean”, “median”, “max”, “min”, “std”, “dist_mean”, “l2norm”, “knn_distance_n”]
    - a Scikit-learn class with fit_transform, e.g., manifold.TSNE()
- An n x d array generated manually

Kepler Mapper: Visualization

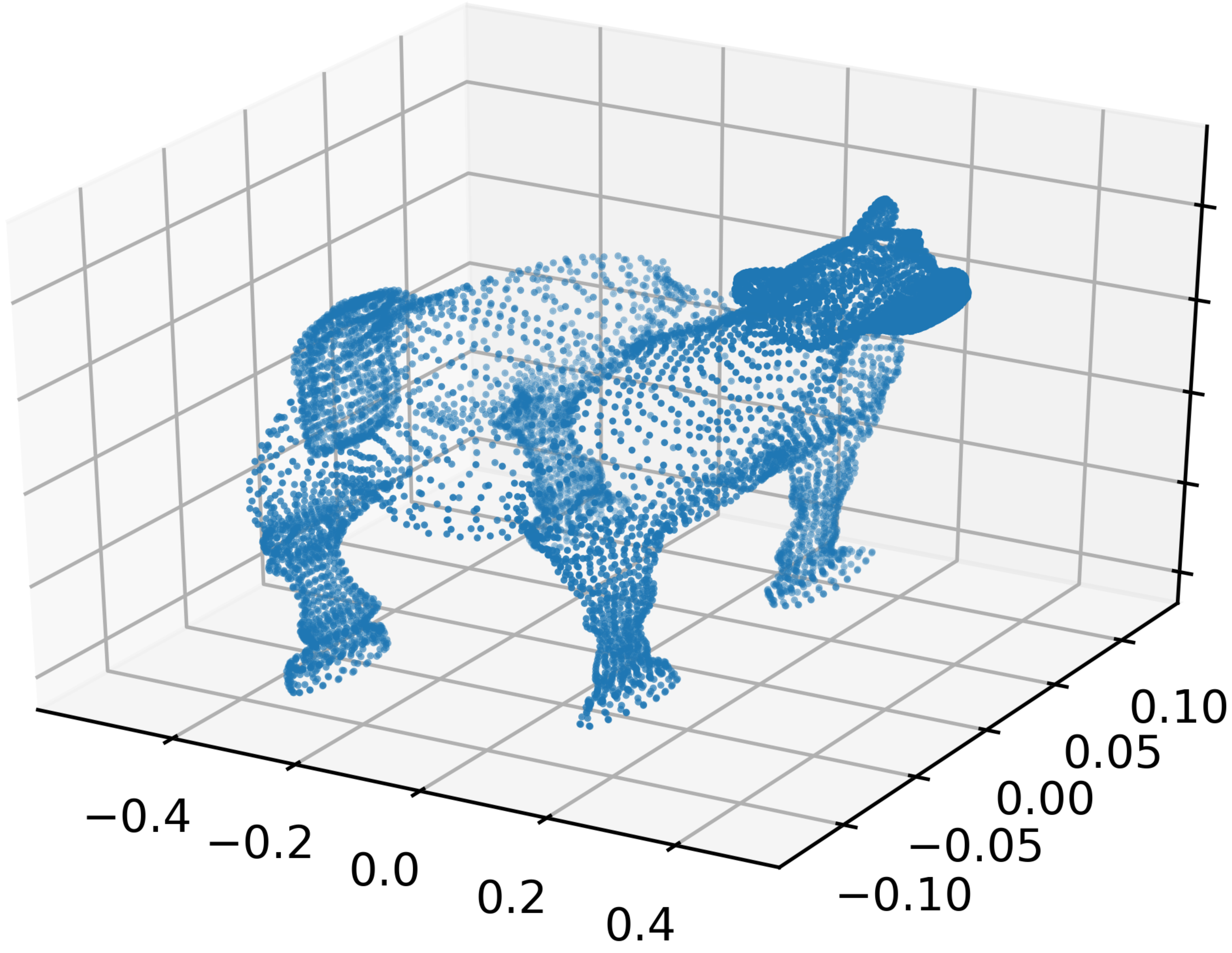
- HTML visualization
- Visualization adapters
  - networkx adapter
- Plotlyviz

Example: 3D horse

- Data: 3D point cloud

	x	y	z
0	-0.013562	0.596860	-0.379364
1	-0.008161	0.602476	-0.378438
2	-0.014804	0.590327	-0.383195
3	-0.011898	0.582940	-0.390050
4	-0.002709	0.575463	-0.396369
...	...	...	...
8426	0.027072	0.222551	-0.542017
8427	0.027611	0.220884	-0.545835
8428	0.033172	0.221070	-0.543778
8429	0.032512	0.223522	-0.540444
8430	-0.054765	0.552121	-0.378869

[8431 rows x 3 columns]



Part II

Giotto-TDA

- A high-performance topological machine learning toolbox in Python
- Create a **MapperPipeline** object that interfaces with scikit-learn for downstream analysis
- Provide an interactive visualization that can be configured in real time
- Installation: `pip install -U giotto-tda`

Giotto-TDA: MapperPipeline

Basic steps

1. Choose a filter function
   A variety of filter functions can be imported as follows:
   `from gtda.mapper.filter import FilterFunctionName`

2. Construct a cover
   A choice of cover can be imported as follow:
   `from gtda.mapper.cover import CoverName`

3. Choose a clustering algorithm

   - scikit-learn method: `from sklearn.cluster import ClusteringAlgorithm`

   - giotto-tda method: `from gtda.mapper.cluster import FirstSimpleGap`

4. The resulting mapper graph will be generated automatically by giotto-tda

Giotto-TDA: Visualization

- **Static**
  - `plot_static_mapper_graph`
  - Configure the layout, coloring, dimension, etc.
- **Interactive**
  - Compute mapper graph on-the-fly
  - Live Jupyter session needed

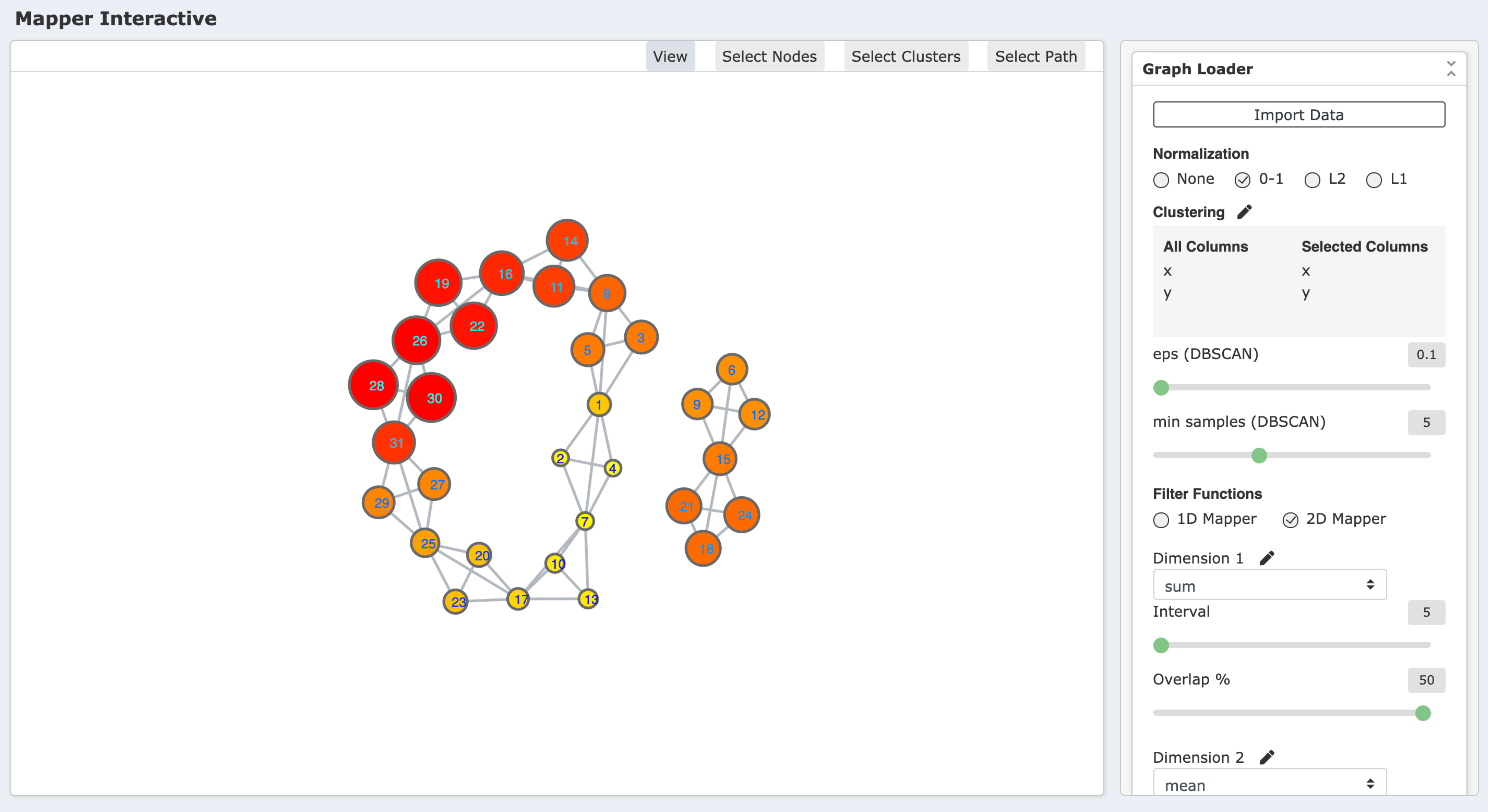
Mapper Interactive

- A scalable, extendable, and interactive toolbox for the visual exploration of high-dimensional point clouds via the mapper graph
- Web-interface for online computation
- Command-line API for offline computation
- The mapper graph computation is modified from the KeplerMapper implementation with an effective speedup strategy
- Installation

```
git clone https://github.com/MapperInteractive/MapperInteractive.git
cd MapperInteractive
python3 run.py
```

Mapper Interactive: Interactivity

- A web-based interface for on-the-fly computation and exploration.



Mapper Interactive: Extendability

- Two modes for different user groups to extend the framework
- **Novice user mode:**
  - Describe the new module information within the new modules.json file.
  - The addition of supervised and unsupervised learning algorithms is allowed via scikit-learn.
- **Expert user mode:**
  - Modify the template function for customizable and multistep analysis pipelines.

Mapper Interactive: Scalability

Key implementational idea

- The computational bottleneck happens during the DBSCAN clustering stage when querying all pairwise distances.
- Modifications:
  - Parallelize individual clustering instances.
  - Precompute the distance matrix of points within each interval.
- This strategy is applicable to any mapper framework employing DBSCAN as a clustering subroutine.

Example: COVID-19 Trends

- Daily records of COVID-19 cases in 9 selected states from April 12, 2020 to September 18, 2020
- 9240 points (rows): a daily record for a given state
- 7 columns for clustering: number of confirmed cases, death cases, active cases, people tested, as well as the testing rate, mortality rate, and incidence rate (i.e., the number of cases per 100K persons).
- Filter function: days of recording
- Parameters
  - DBSCAN: eps = 0.15, min_samples = 5
  - Covering: n = 20, p = 0.5

1	Province_Sta	Country_Reg	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	FIPS	Incident_Rat	People_Teste	People_Hosp	Mortality_Ra	UID	ISO3	Testing_Rate	Hospitalizati	start_date	days
2	Arizona	US	6/8/20 3:33	33.7298	-111.4312	26989	1051	5517	20421	4	370.793369	281621	3352	3.89417911	84000004	USA	3869.10221	12.4198748	4/12/20	57
3	California	US	6/8/20 3:33	36.1162	-119.6816	130615	4632		125983	6	330.568594	2362218		3.5463002	84000006	USA	5978.44874		4/12/20	57
4	Florida	US	6/8/20 3:33	27.7663	-81.6868	63938	2700		61238	12	297.694306	1216158	11215	4.22284088	84000012	USA	5662.4122	17.5404298	4/12/20	57
5	Georgia	US	6/8/20 3:33	33.0406	-83.6431	51898	2180		49718	13	488.800343	539884	8685	4.20054723	84000013	USA	5084.88736	16.7347489	4/12/20	57
6	Illinois	US	6/8/20 3:33	40.3495	-88.9861	127757	5904		121853	17	1008.19764	1042774		4.6212732	84000017	USA	8229.07773		4/12/20	57
7	New Jersey	US	6/8/20 3:33	40.2989	-74.521	164164	12176	27824	124164	34	1848.23788	960425	18050	7.41697327	84000034	USA	10812.9301	10.9951025	4/12/20	57
8	New York	US	6/8/20 3:33	42.1657	-74.9481	378097	30374	67544	280179	36	1943.5876	2497842	89995	8.03338826	84000036	USA	12840.0245	23.8020931	4/12/20	57
9	North Carolir	US	6/8/20 3:33	35.6301	-79.8064	35625	1032	18860	15733	37	339.671193	511226		2.89684211	84000037	USA	4874.35074		4/12/20	57
10	Texas	US	6/8/20 3:33	31.0545	-97.5635	75408	1841	49758	23809	48	260.064524	1100446		2.44138553	84000048	USA	3795.18043		4/12/20	57
11	Arizona	US	6/7/20 3:53	33.7298	-111.4312	25451	1043	5399	19009	4	349.663272	271646	3320	4.0980708	84000004	USA	3732.05882	13.0446741	4/12/20	56
12	California	US	6/7/20 3:53	36.1162	-119.6816	128593	4607		123986	6	325.45119	2308300		3.58184349	84000006	USA	5841.98971		4/12/20	56
13	Florida	US	6/7/20 3:53	27.7663	-81.6868	62758	2688		60070	12	292.200244	1174185	11163	4.28311928	84000012	USA	5466.98658	17.7873737	4/12/20	56
14	Georgia	US	6/7/20 3:53	33.0406	-83.6431	51359	2178		49181	13	483.723781	522857	8662	4.24073677	84000013	USA	4924.51888	16.8655932	4/12/20	56
15	Illinois	US	6/7/20 3:53	40.3495	-88.9861	126890	5864		121026	17	1001.35569	1022074		4.62132556	84000017	USA	8065.72315		4/12/20	56
16	New Jersey	US	6/7/20 3:53	40.2989	-74.521	163893	12106	27641	124146	34	1845.18683	919448	18023	7.38652658	84000034	USA	10351.5912	10.9968089	4/12/20	56
17	New York	US	6/7/20 3:53	42.1657	-74.9481	377316	30280	67261	279775	36	1939.57291	2437407	89995	8.02510363	84000036	USA	12520.3616	23.8513607	4/12/20	56

Example - Neuron Activations

- The Cifar dataset is created by passing input images from CIFAR-10 [10] to ResNet-18 neural network.
- Treat the activation vectors collected from the last layer of the network as the input high-dimensional point cloud
- 50K images (rows) from 10 image classes
- 512 dimensions (columns)
- Filter function: l2-norm
- Parameters
  - No normalization
  - DBSCAN: eps = 10, min_samples = 5
  - Covering: n = 40, p = 0.2

References

[1] G. Singh, F. M´emoli, and G. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symposium on Point-Based Graphics*, pages 91–100, 2007.

[2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pages 226–231, 1996.

[3] van Veen et al., (2019). Kepler Mapper: A flexible Python implementation of the Mapper algorithm. Journal of Open Source Software, 4(42), 1315, https://doi.org/10.21105/joss.01315

[4] Hendrik Jacob van Veen, Nathaniel Saul, David Eargle, & Sam W. Mangham. (2019, October 14). Kepler Mapper: A flexible Python implementation of the Mapper algorithm (Version 1.4.1). Zenodo. http://doi.org/10.5281/zenodo.4077395

[5] Tauzin, Guillaume, et al. "giotto-tda:: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration." J. Mach. Learn. Res. 22 (2021): 39-1.

[6] Youjia Zhou, Nithin Chalapathi, Archit Rathore, Yaodong Zhao, and Bei Wang. 2021. Mapper Interactive: A Scalable, Extendable, and Interactive Toolbox for the Visual Exploration of High-Dimensional Data. In IEEE 14th Pacific Visualization Symposium. 101–110.

References

[7] C. Maria, J.-D. Boissonnat, M. Glisse, and M. Yvinec, “The GUDHI library: simplicial complexes and persistent homology,” in International congress on mathematical software. Springer, 2014, pp. 167–174.

[8] Methun Kamruzzaman, Ananth Kalyanaraman, Bala Krishnamoorthy, Stefan Hey, and Pat Schnable. 2019. Hyppo-X: A scalable exploratory framework for analyzing complex phenomics data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2019).

[9] Y. Zhou, M. Kamruzzaman, P. Schnable, B. Krishnamoorthy, A. Kalyanaraman, and B. Wang. Pheno-mapper: an interactive toolbox for the visual exploration of phenomics data. In Proceedings of the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, pages 1–10, 2021.

[10] Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009): 7.