Topological Mode Analysis Tool (ToMATo): A Clustering Method Based on the Topological Data Analysis (TDA)

Ananya Sharma
Harshit Joshi

Department of Mathematics
Indian Institute of Technology, Delhi

April 20, 2025

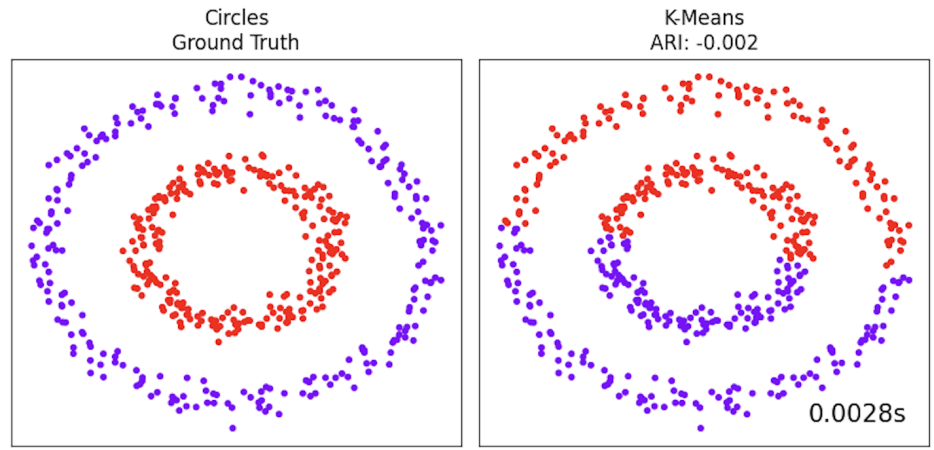
Introduction

- **ToMATo (Topological Mode Analysis Tool)** is a clustering algorithm based on topological data analysis.
- **Topological Data Analysis (TDA)** is a set of techniques that use concepts from topology to study the shape and structure of data. It helps identify patterns, clusters, and features like holes or voids in high-dimensional datasets
- It identifies dense regions in data using persistence and merges them based on topological features, making it robust to noise and shape variations.

Recall classical clustering methods

K-means:

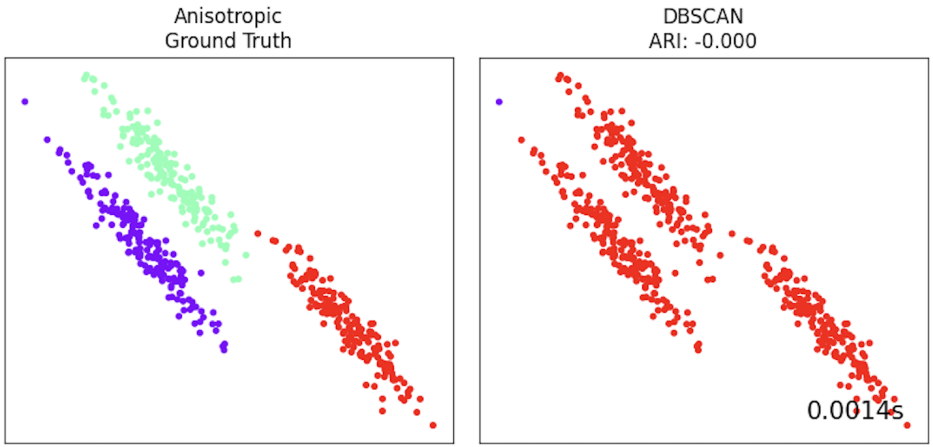
- For a fixed number k, it tries to minimize the sum of the squared distances to the center within each cluster.
- K-means assumes convex clusters and struggles with complex shapes.



Recall classical clustering methods

DBSCAN:

- It clusters data by identifying ϵ-**neighborhood** of a point p which includes all points within distance ϵ from p.
- If p has $\geq$ **MinPts** neighbors then it forms a new cluster.
- As it uses a fixed threshold, it is unsuitable for data with multiscale or hierarchical structures where clusters exist at varying densities.



Homology

To understand how the ToMATo algorithm identifies clusters using persistent homology, it's important to first grasp the fundamental concepts of homology.

Homology captures information about a space's structure by examining its holes/cavities of various dimensions.

- **0-holes** (H_0): Connected components of the space.
- **1-holes** (H_1): Non-contractible loops (e.g., circles on a torus).
- **2-holes** (H_2): Cavities (e.g., hollow interiors of a sphere or torus).

Betti numbers: It is defined as the rank of a homology group $H_k(X)$ that counts the number of independent k-dimensional holes in a space.

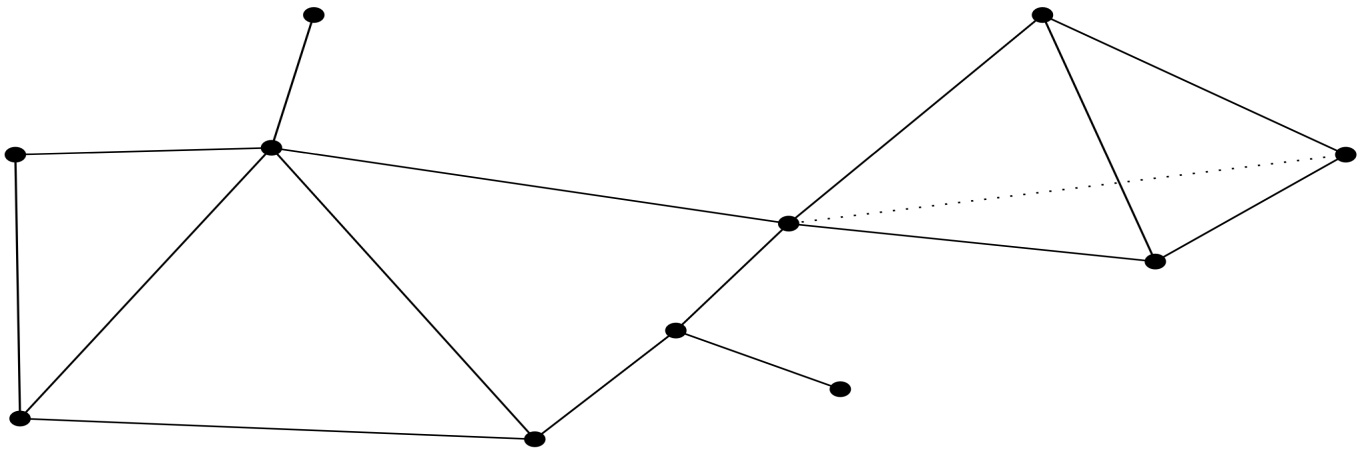
$\beta_0 = 1 \quad \beta_1 = 2 \quad \beta_2 = 1$

Computation of Homology

1. Identifying Simplices:

A Simplex complex is a discrete mathematical structure used in topology that consists of **points (0-simplices)**, **edges (1-simplices)**, and higher-dimensional simplices like **triangles (2-simplices)** and **tetrahedra (3-simplices)**, which are combined in a structured way.

Example: Given n-simplex has eleven 0-simplices, sixteen 1-simplices, seven 2-simplices and one 3-simplex.



A 1-simplex (edge) with endpoints v_1, v_2 is written as: $e = [v_1, v_2]$.

Computation of Homology

2. k-chains $C_k(X)$: A k-chain is a formal linear combination of k-simplices in the complex.

Example: Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be the k-simplices in X, and let the coefficient group be $\mathbb{Z}_2$. Then a k-chain is:

$$c = a_1\sigma_1 + a_2\sigma_2 + \cdots + a_n\sigma_n \quad \text{where } a_i \in \mathbb{Z}_2$$

and $C_k(X) \cong \mathbb{Z}_2^n$.

3. Boundary Operator ∂_k: $C_k(X) \to C_{k-1}(X)$

Example: For an edge $e = [v_0, v_1]$, the boundary is $\partial_1[e] = v_0 + v_1$. Also,

$$\partial_{k-1} \circ \partial_k = 0$$

Computation of Homology

4. k**-cycles** $Z_k(X)$**:** The kernel of the boundary operator ∂_k, meaning all k-chains with no boundary.

5. k**-boundaries** $B_k(X)$**:** It is the set of k-chains that are actual boundaries of $k+1$-chains

Example: Consider a triangle $\tau = [v_1, v_2, v_3]$ with edges e_1, e_2 and e_3. The boundary operator ∂_2 maps 2-chains (triangles) to 1-chains (edges):

$$\partial_2 \tau = e_1 + e_2 + e_3$$

This means the boundary of the triangle is the sum of its three edges.

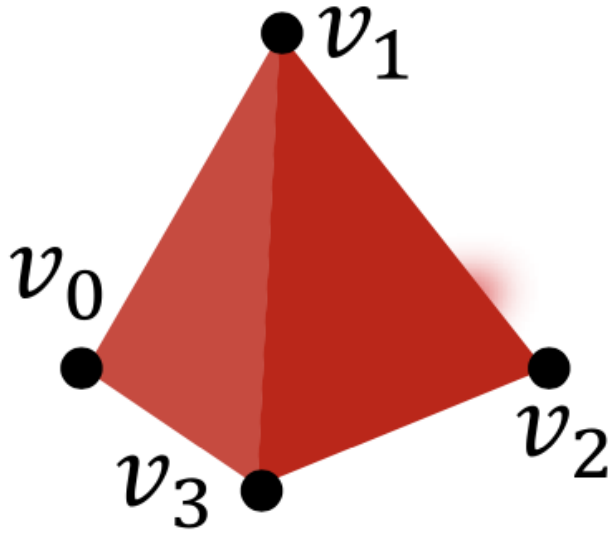
6. Homology Group $H_k(X)$**:** It is defined as the quotient group:

$$H_k(X) = Z_k(X)/B_k(X)$$

Computation of Homology (Example)

Consider a hollow tetrahedron:

- **Four 2-simplices (triangles)**: $\tau_1 = [v_0, v_1, v_2], \tau_2 = [v_1, v_2, v_3], \tau_3 = [v_0, v_1, v_3]$ and $\tau_4 = [v_0, v_2, v_3]$
- **Six 1-simplices (edges)**: $e_1 = [v_0, v_1], e_2 = [v_1, v_2], e_3 = [v_2, v_3], e_4 = [v_3, v_0], e_5 = [v_0, v_2]$, and $e_6 = [v_1, v_3]$
- **Four 0-simplices (vertices)**: v_0, v_1, v_2 and v_3

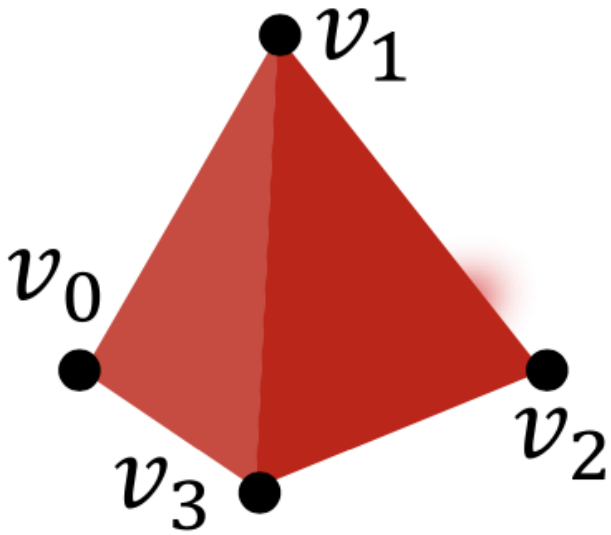


The chain groups are:

$$C_2(X) = \mathbb{Z}_2^4, \quad C_1(X) = \mathbb{Z}_2^6, \quad C_0(X) = \mathbb{Z}_2^4.$$

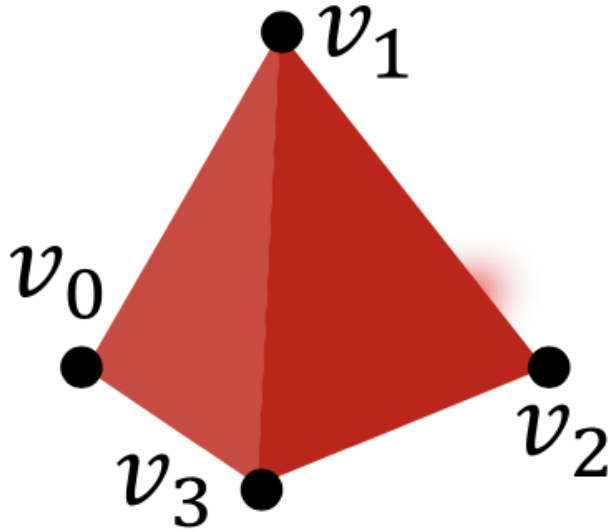
Computation of $H_2(X)$ (Detecting 2D Cavities):

- $Z_2(X) =$ Kernel of ∂_2. The only generator of $Z_2(X)$ is $\tau_1 + \tau_2 + \tau_3 + \tau_4$ so $Z_2(X) = \mathbb{Z}_2$.
- Since there is no 3-simplex in X, $B_2(X) = Img(\partial_3) = \phi$.
- Thus, $H_2(X) = Z_2(X)/B_2(X) = \mathbb{Z}_2$.
- X has **one 2-dimensional cavity**, consistent with it being a hollow tetrahedron.



Computation of $H_1(X)$ (Detecting 1D Holes):

- The 1-cycles $Z_1(X)$ consist of edge combinations whose boundaries vanish *i.e.* $Z_1(X) = \mathbb{Z}_2^3$.
- The 1-boundaries $B_1(X)$, which are the boundaries of the triangles, are $B_1(X) = Img(\partial_2) = \text{span}_{\mathbb{Z}_2}\{\partial_2(\tau_1),\ \partial_2(\tau_2),\ \partial_2(\tau_3),\ \partial_2(\tau_4)\} = \text{span}_{\mathbb{Z}_2}\{e_1 + e_2 + e_5,\ e_2 + e_3 + e_6,\ e_1 + e_4 + e_6,\ e_3 + e_4 + e_5\} = \mathbb{Z}_2^3$.
- Thus, $H_1(X) = Z_1(X)/B_1(X) = \phi$.
- Thus, there are **no Non-contractible loops** in X.

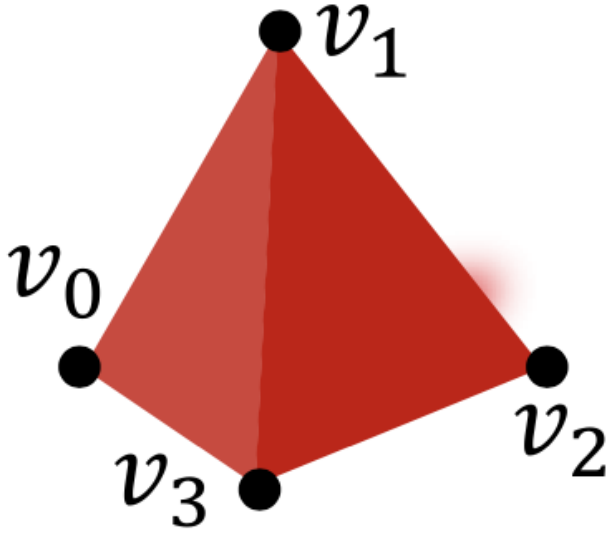


Computation of $H_0(X)$ (Detecting Connected Components):

- The 0-cycles $Z_0(X)$ consist of all vertex sums with no boundary.
  $Z_0(X) = C_0(X)$.
- The 0-boundaries
  $B_0(X) = \text{span}_{\mathbb{Z}_2}\{v_0+v_1,\ v_1+v_2,\ v_2+v_3,$
  $v_3 + v_0,\ v_0 + v_2,\ v_1 + v_3\} = \mathbb{Z}_2^3$.
- We obtain:

$$H_0(X) = \mathbb{Z}_2.$$

- This confirms that X has **one connected component**, as expected.



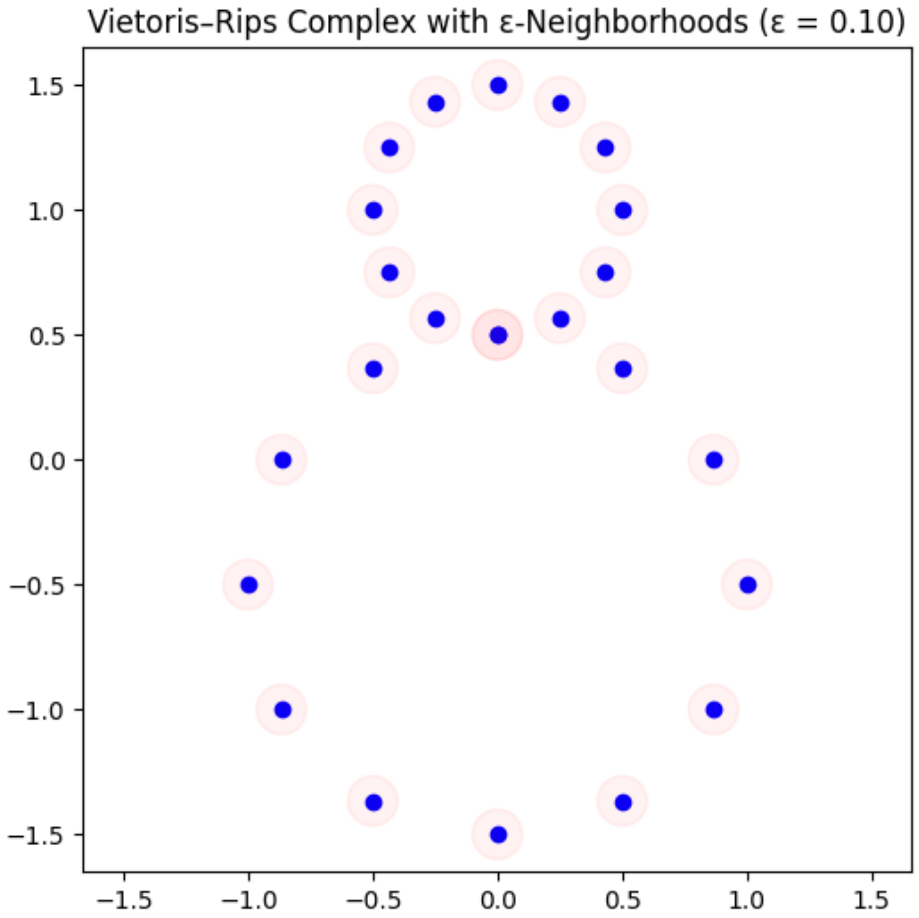
Vietoris–Rips Complex

- Given a point cloud $X \subset M$ and a fixed scale parameter $\epsilon > 0$, the Vietoris–Rips complex $VR(X, \epsilon)$ is defined by

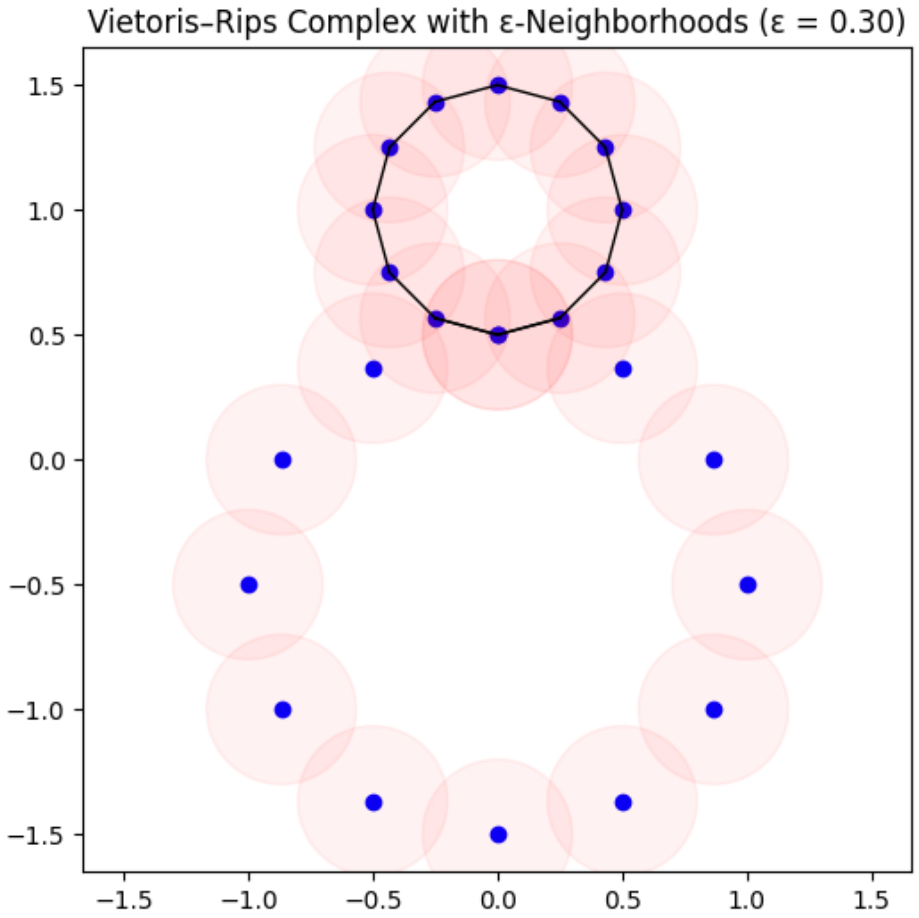
$$VR(X, \epsilon) = \{ \sigma \subset X : d(x, y) \leq \epsilon \quad \forall\, x, y \in \sigma \}.$$

- That is, any finite set of points forms a simplex if every pair of points in that set is at most ϵ apart.
- As ϵ increases, more simplices appear, capturing more "global" features of the point cloud.

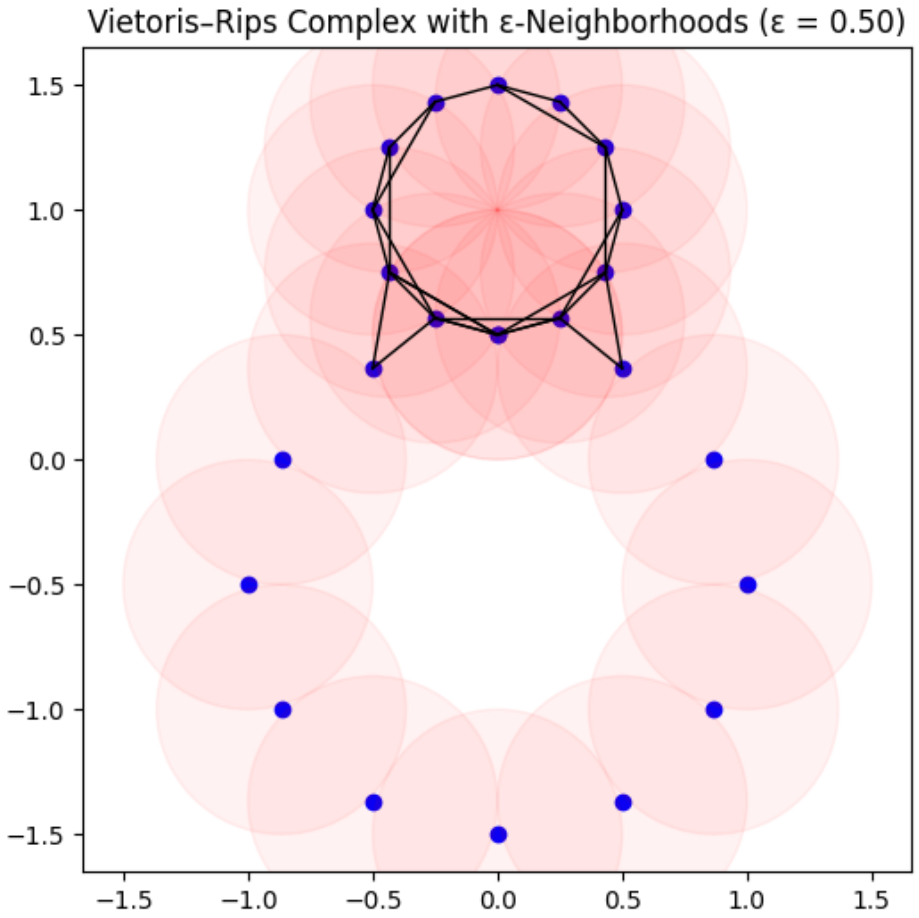
Visualization of VR Complex



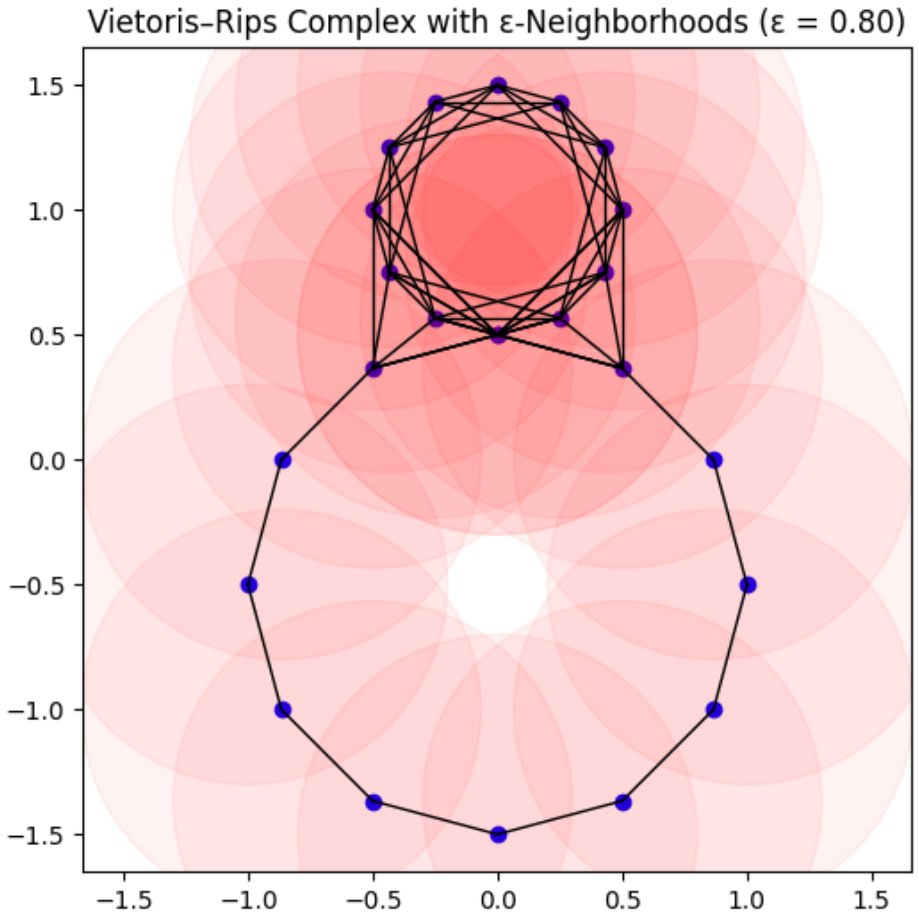
Visualization of VR Complex



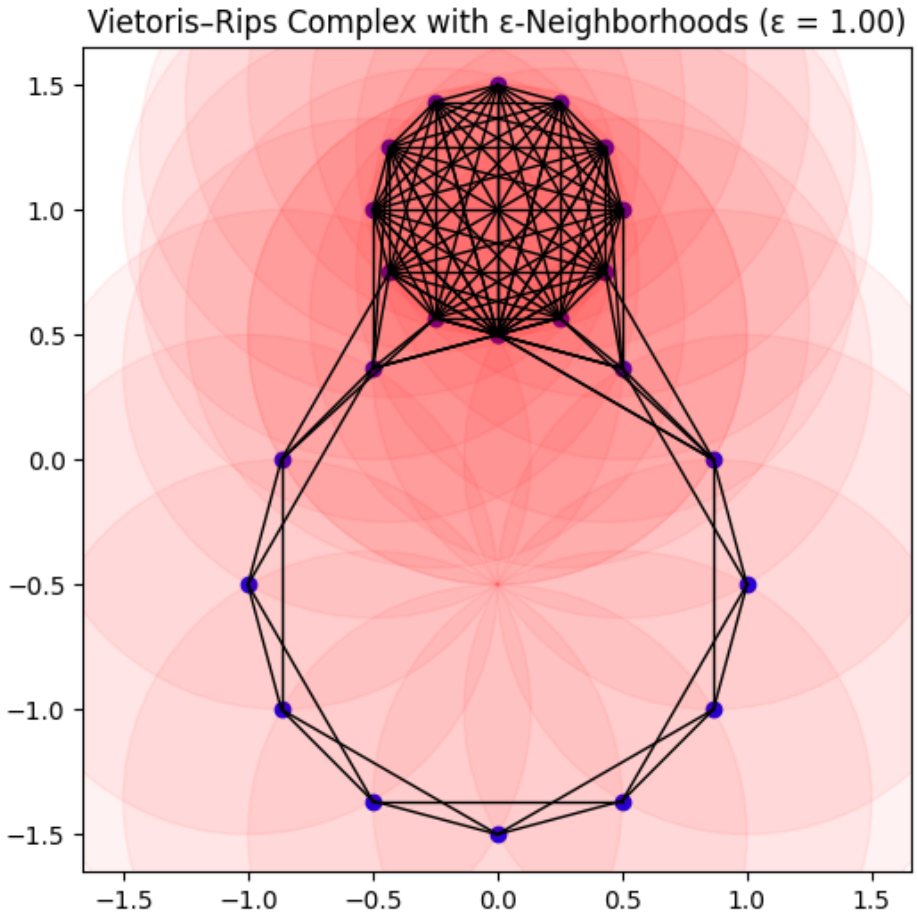
Visualization of VR Complex



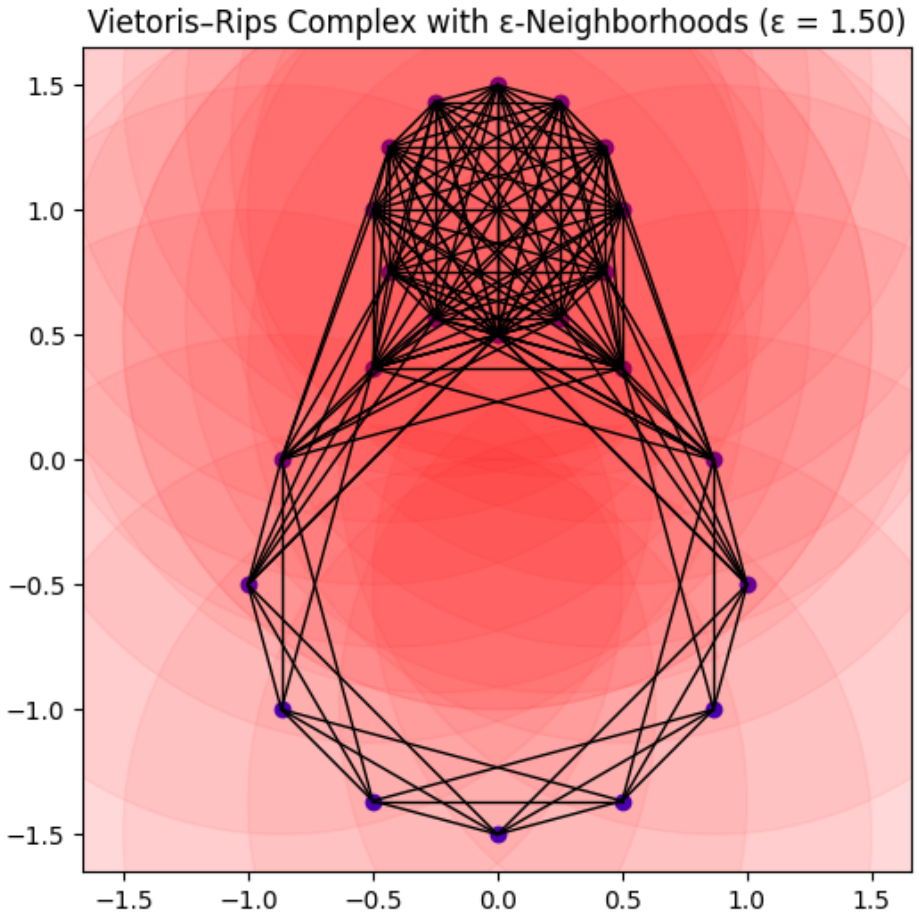
Visualization of VR Complex



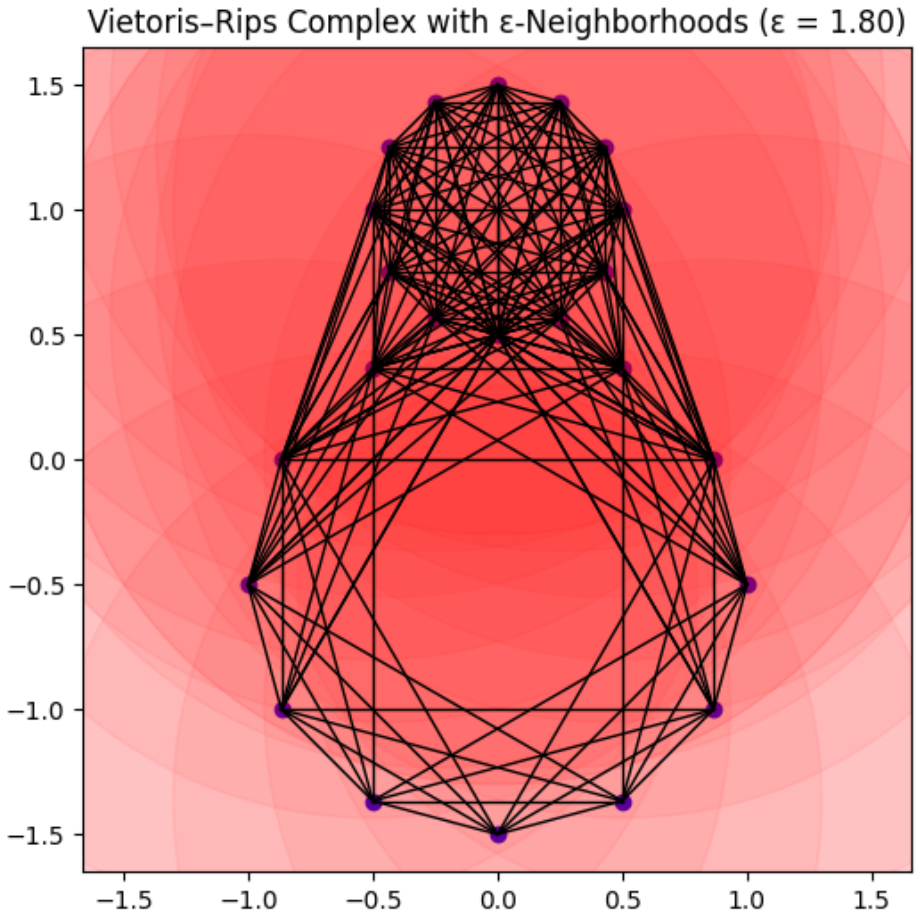
Visualization of VR Complex



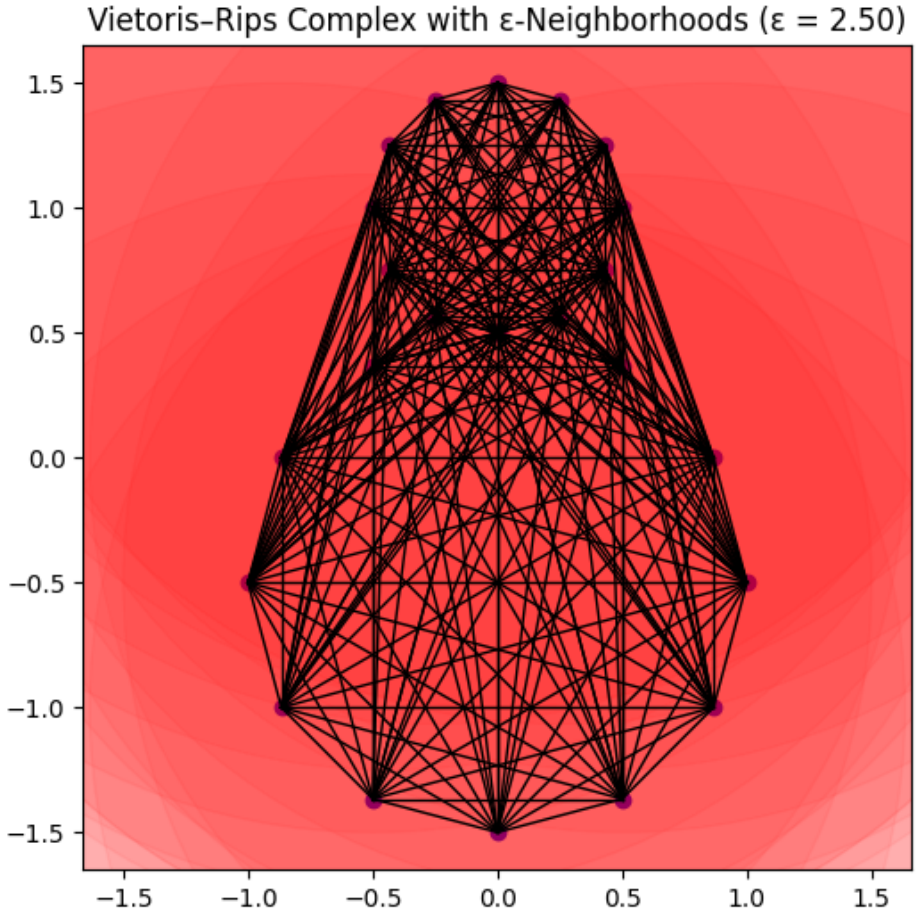
Visualization of VR Complex



Visualization of VR Complex



Visualization of VR Complex



Filtrations of VR Complex

Instead of fixing a single scale ϵ, we study how the topology of the complex changes as we vary ϵ. This gives rise to the concept of a **filtration**.

Filtration:

A filtration is a nested sequence of simplicial complexes:

$$\emptyset = K_0 \subset K_1 \subset \cdots \subset K_N = K,$$

where each $K_i = VR(X, \epsilon_i)$ for a sequence $0 \leq \epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_N$.

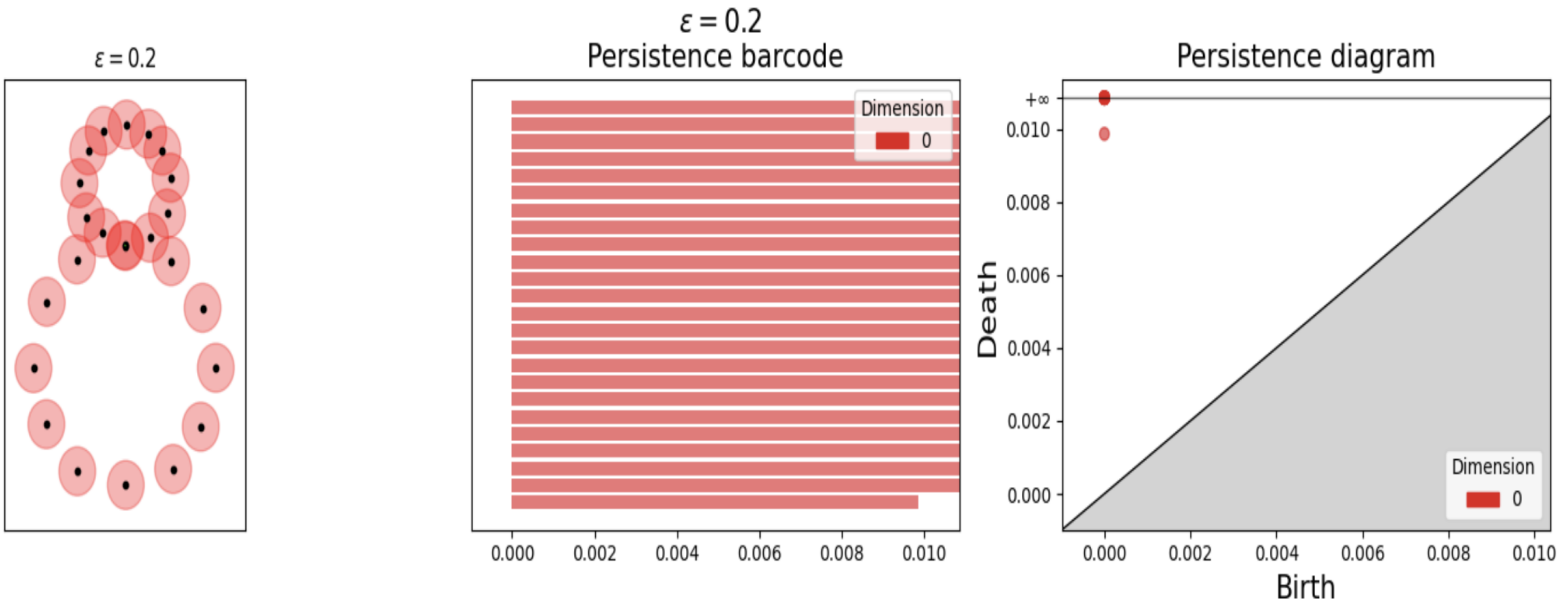
Persistence Diagrams

Evolution of persistence diagram can be understood as following:

- For each $k-$hole σ, persistence homology records its first appearance in the filtration sequence, denoted K_{io} , and its first disappearance in a later complex, K_{jo} .
- **Birth time:** We define $b_\sigma = \epsilon_{io}$ as the birth time.
- **Death time:** We define $d_\sigma = \epsilon_{jo}$ as the death time.
- **Lifespan of σ:** The difference $d_\sigma - b_\sigma$ is called the lifespan of σ.
- We represent each $k-$hole, σ with a 2-tuple (b_σ, d_σ) to denote its birth and death times in the filtration.
- The collection of all such 2-tuples is called the **persistence diagram (PD)**.

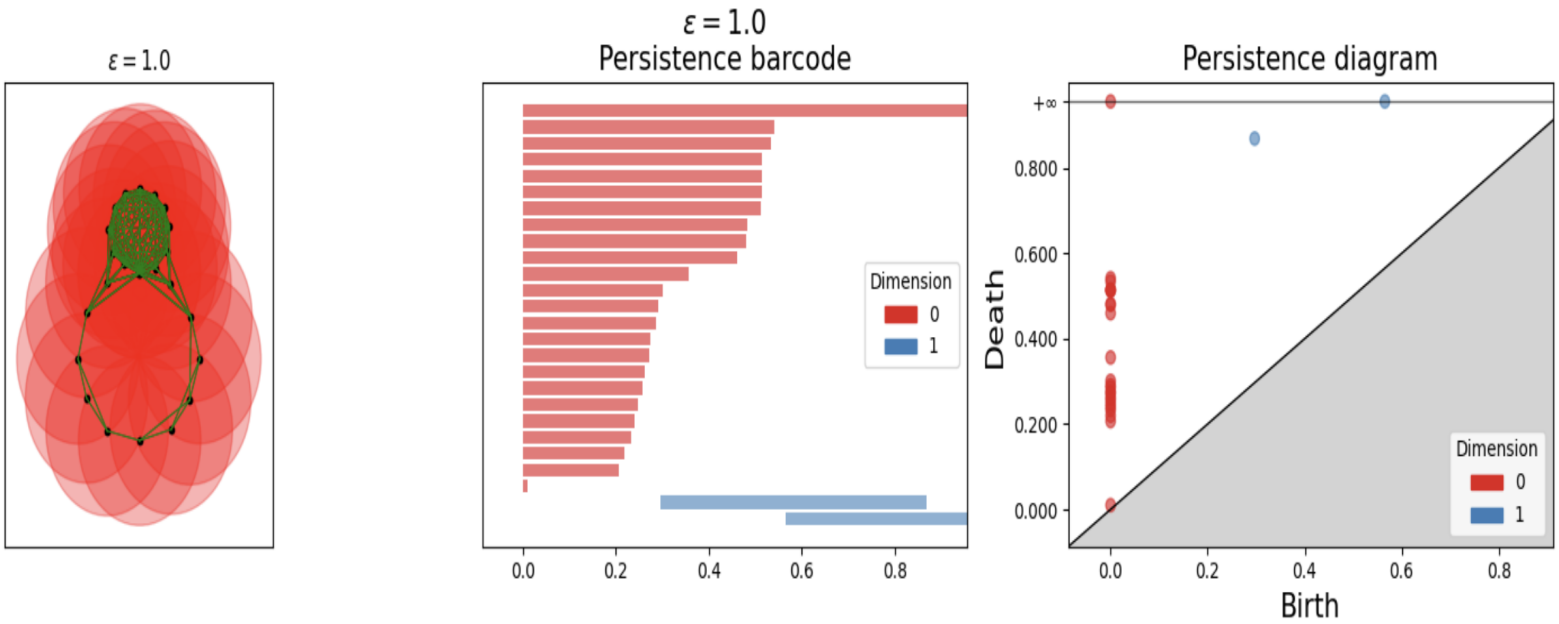
Persistence Diagrams

- 0 lifespan is considered a trivial topological feature, and they are represented with diagonal elements ($x = y$) in the PD.
- **Persistence Barcode:** It uses bars instead of 2-tuples.



How to differentiate between an important topological feature and noise?

In general, features with long lifespans are located far from the diagonal and are considered significant features. Features with short lifespans are close to the diagonal and are considered insignificant features.



ToMaTo Algorithm

Mode-Seeking

In classical Morse theory, one studies smooth functions $f : M \to \mathbb{R}$ on a manifold M. Critical points (where the gradient $\nabla f = 0$) and their indices provide deep insight into the topology of M.

In data analysis, we can often define a density estimator $f : X \to \mathbb{R}$ on a point cloud. The local maxima of this function correspond to regions where data are "concentrated."

We can then study the gradient flow induced by f; points flow toward local maxima, and the basins of attraction of these maxima naturally suggest a clustering of the data.

This approach is called **mode-seeking**

Mode-Seeking

A common issue faced by this technique is that the gradient and extremal points of a density function are notoriously unstable, so their approximation from a density estimator can lead to unpredictable results

We can use Topological Persistence to detect and merge these unstable clusters

Superlevel-Sets

Let $f : X \to \mathbb{R}$ be a density function defined on a space X. For each threshold $\alpha \in \mathbb{R}$, the *superlevel-set* is defined by

$$F_\alpha = f^{-1}([\alpha, +\infty)) = \{x \in X \mid f(x) \geq \alpha\}.$$

As α decreases from $+\infty$ to $-\infty$, the sets F_α form a nested family (or *filtration*):

$$F_{+\infty} \subseteq \cdots \subseteq F_\alpha \subseteq F_\beta \subseteq \cdots \subseteq F_{-\infty}.$$

Algorithm

INPUT

- A simple graph G with n vertices.
- A density vector $\tilde{f} \in \mathbb{R}^n$ where $\tilde{f}(i)$ is the estimated density at vertex i.
- A merging parameter $\tau \geq 0$.

NOTE

The graph G may be calculated using:

- k-Nearest Neighbors (k-NN): Each point $x \in X$ is connected to its k nearest neighbors.
- ϵ-Neighborhood: Two points $x, y \in X$ are connected if $d(x, y) \leq \epsilon$.

Sorting and Initialization

1. **Sort the vertices:** Order the indices $\{1, 2, \ldots, n\}$ such that

$$\tilde{f}(1) \geq \tilde{f}(2) \geq \cdots \geq \tilde{f}(n).$$

2. **Initialization:**
   - Initialize a union-find data structure U to track disjoint clusters.
   - Initialize two vectors g and r :
     - $g(i)$ stores the neighbor that has the highest density among those already processed **(i := data point)**.
     - $r(e)$ denotes the root (or center) of the cluster e.**(e:= cluster)**

Main Loop (Case1)

Main Loop: For $i = 1$ to n, do *(Note: vertices are ordered)*:

- Let N be the set of neighbors of vertex i in G that have indices lower than i (i.e., already processed and with higher density).
- **Case 1:** If $N = \emptyset$, then vertex i is a peak. Create a new entry e in U and assign:

$$r(e) \leftarrow i.$$

(we assign i as the root for a new cluster e)

Main Loop (Case2)

Case 2: if $N \neq \emptyset$:

- Compute

$$g(i) = \arg\max_{j \in N} \tilde{f}(j).$$

(Neighbor of i with highest density)

- Let $e_i = U.\text{find}(g(i))$
- Attach vertex i to the cluster e_i.
  - For each $j \in N$, let $e = U.\text{find}(j)$. If

$$e \neq e_i \quad \text{and} \quad \min\{\tilde{f}(r(e)), \tilde{f}(r(e_i))\} < \tilde{f}(i) + \tau,$$

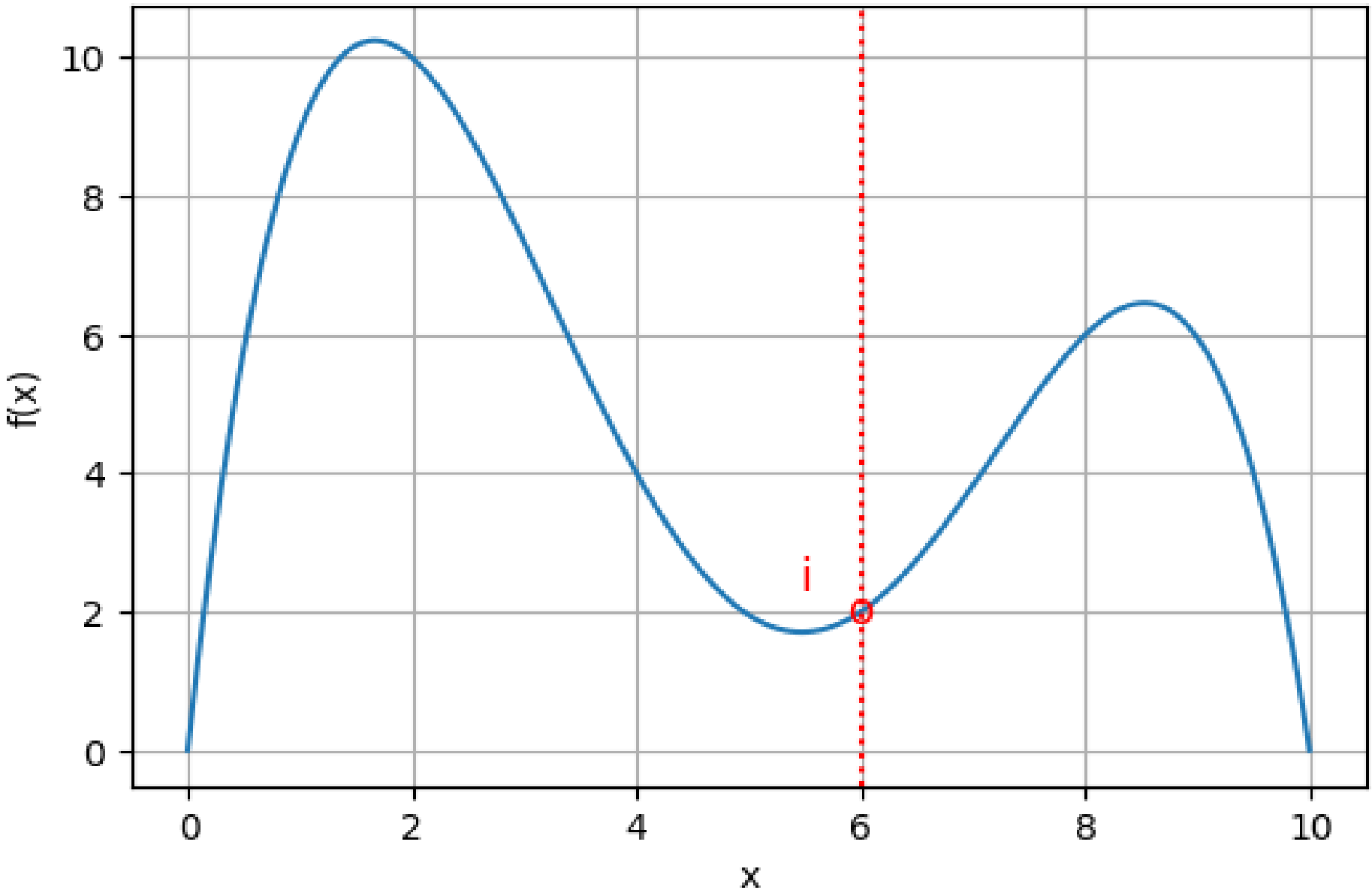
then merge the clusters:

$$U.\text{union}(e, e_i), \quad \text{and update}$$

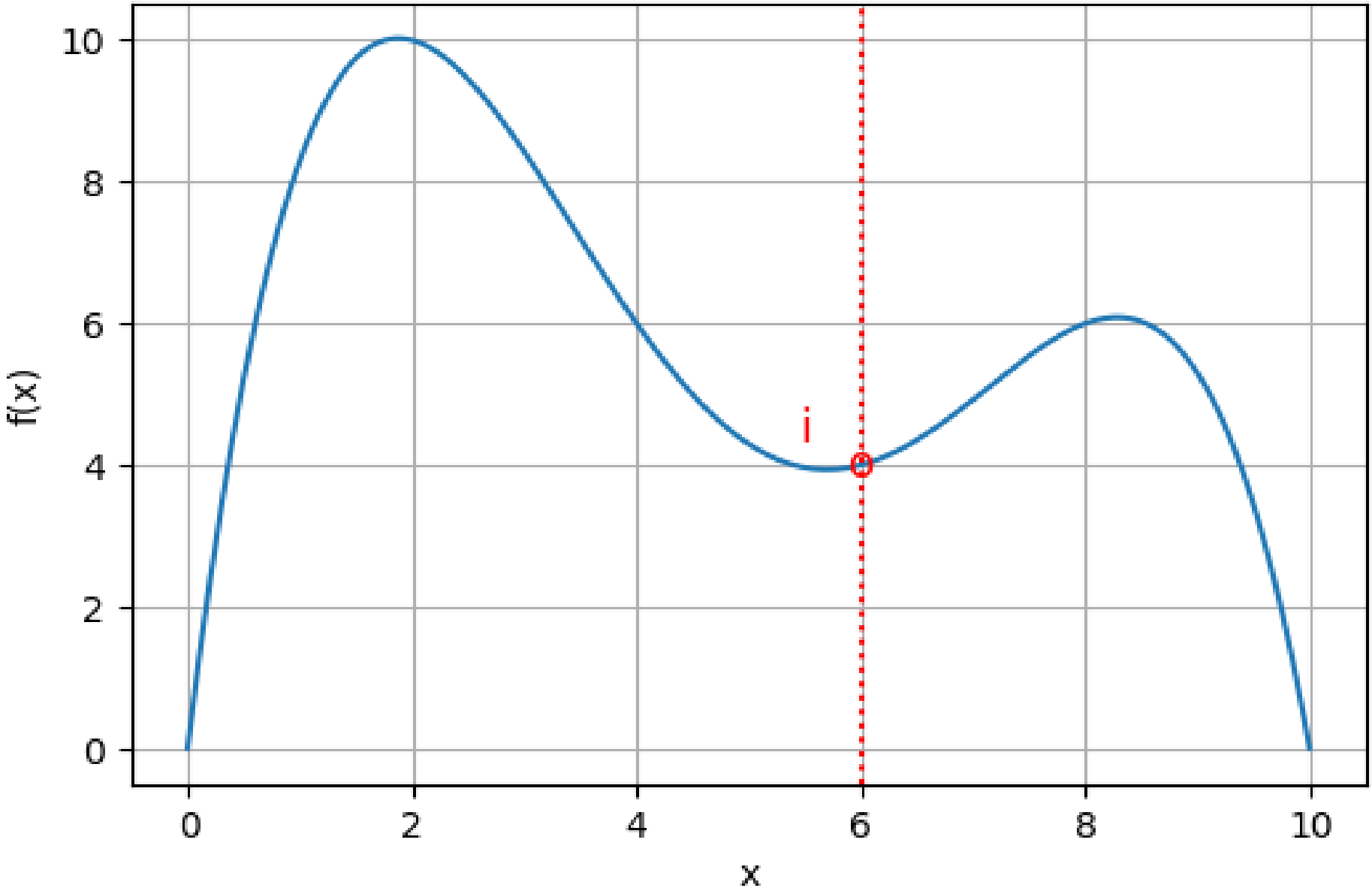
$$r(e \cup e_i) = \arg\max\{\tilde{f}(r(e)), \tilde{f}(r(e_i))\}.$$

Finally, update $e_i \leftarrow e \cup e_i$. (for each neighbor of i we find their respective clusters and merge the clusters (of j and i) if they are not prominent.

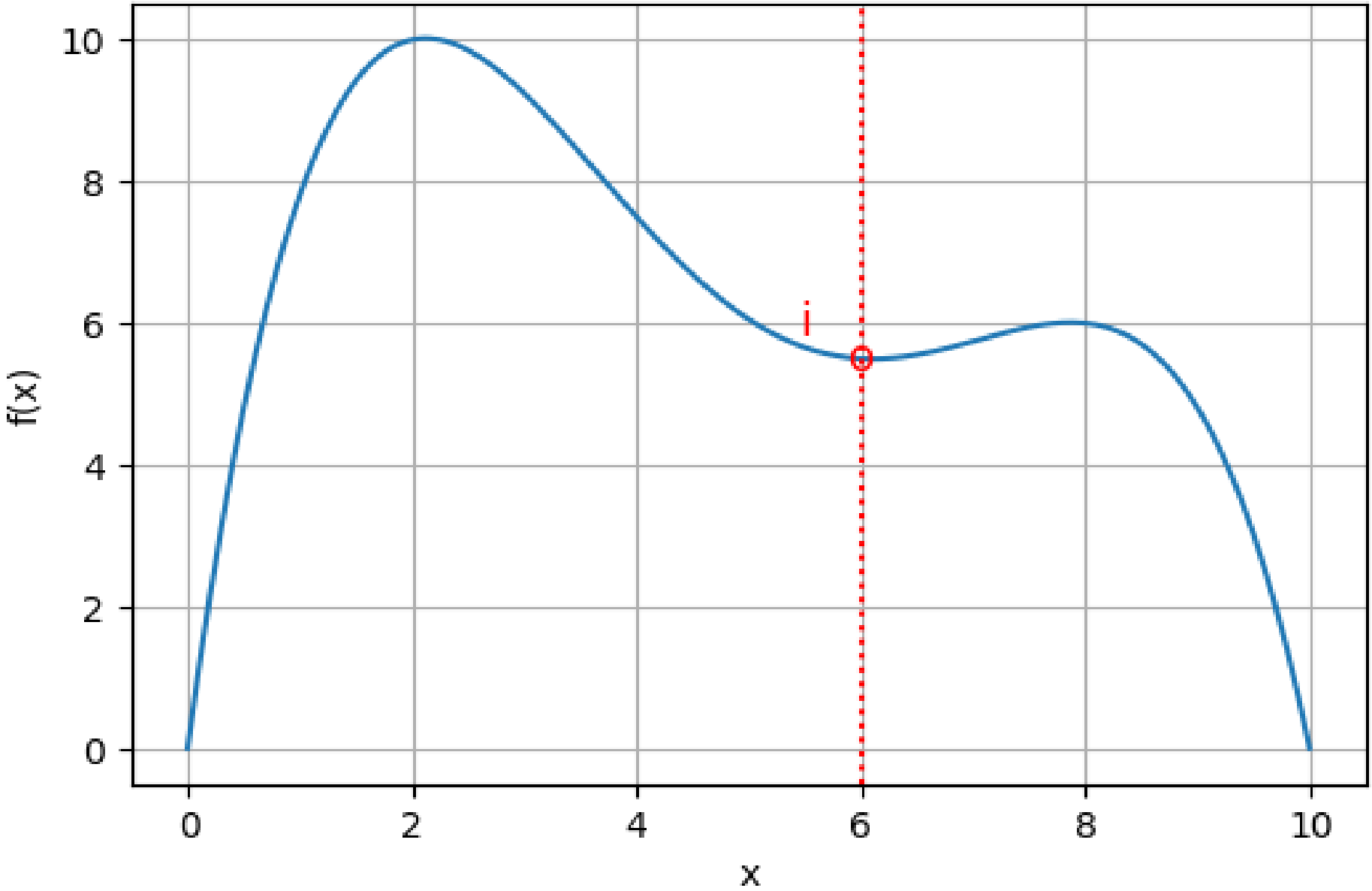
Visualization of Persistence



Visualization of Persistence

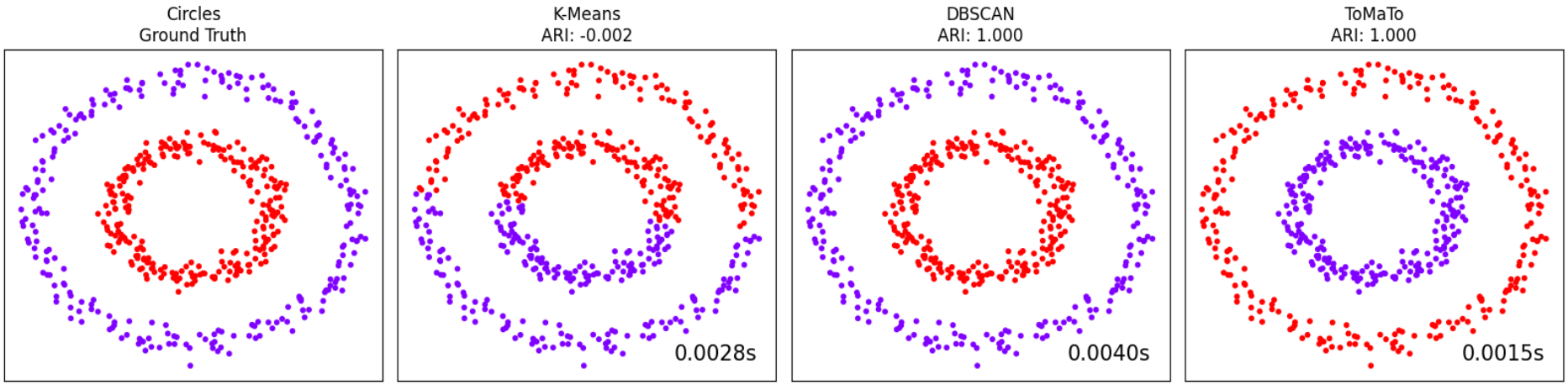


Visualization of Persistence

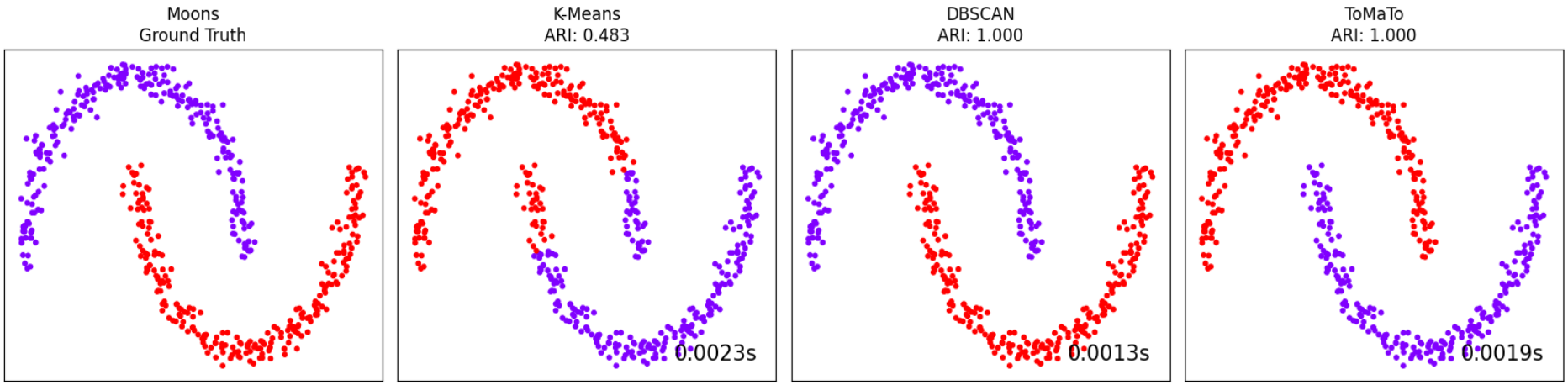


Testing on 2-D Datasets

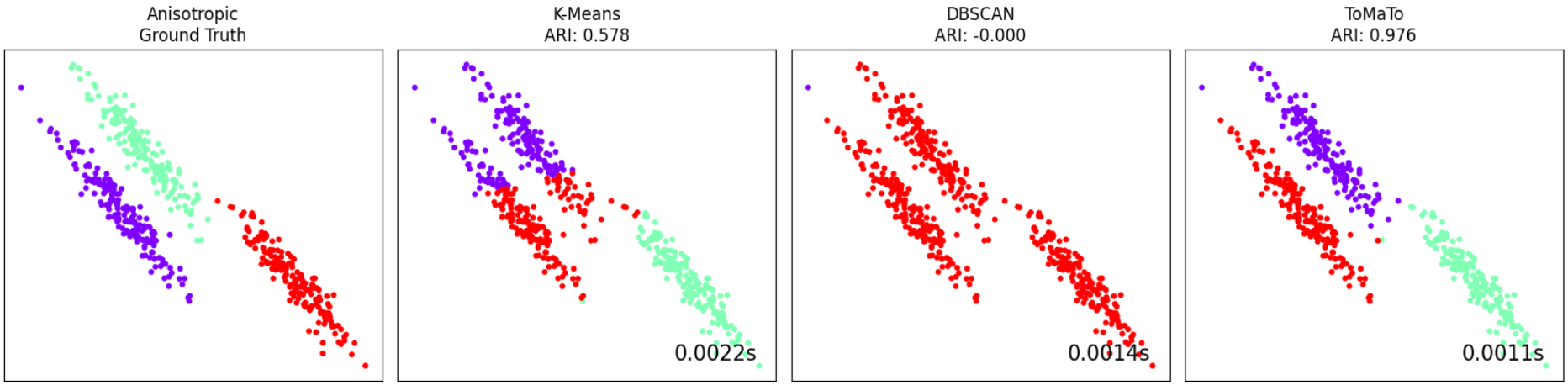
Circles


Circles
Ground Truth
K-Means
ARI: -0.002
0.0028s
DBSCAN
ARI: 1.000
0.0040s
ToMaTo
ARI: 1.000
0.0015s

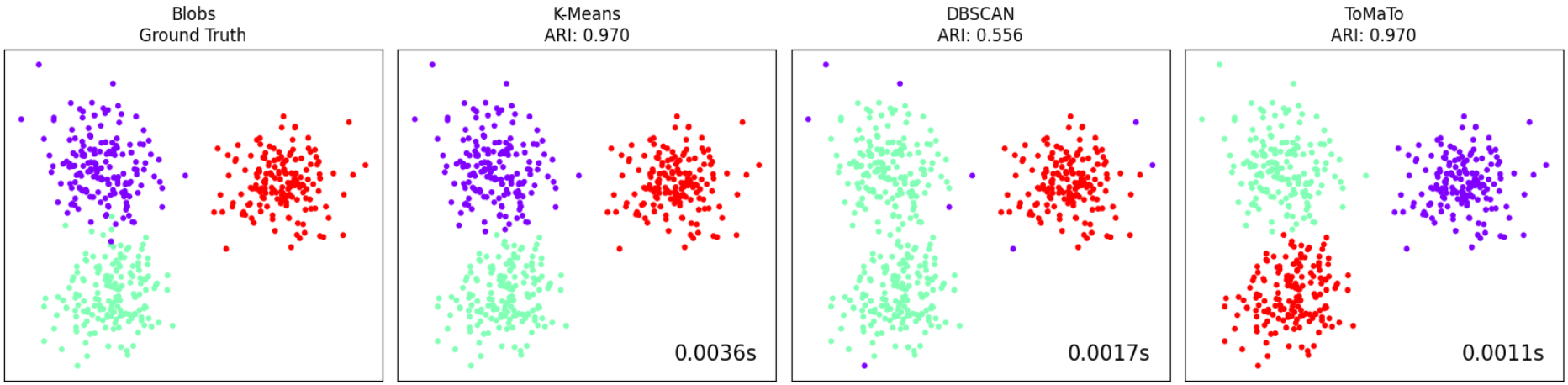
Moons


Moons
Ground Truth
K-Means
ARI: 0.483
0.0023s
DBSCAN
ARI: 1.000
0.0013s
ToMaTo
ARI: 1.000
0.0019s

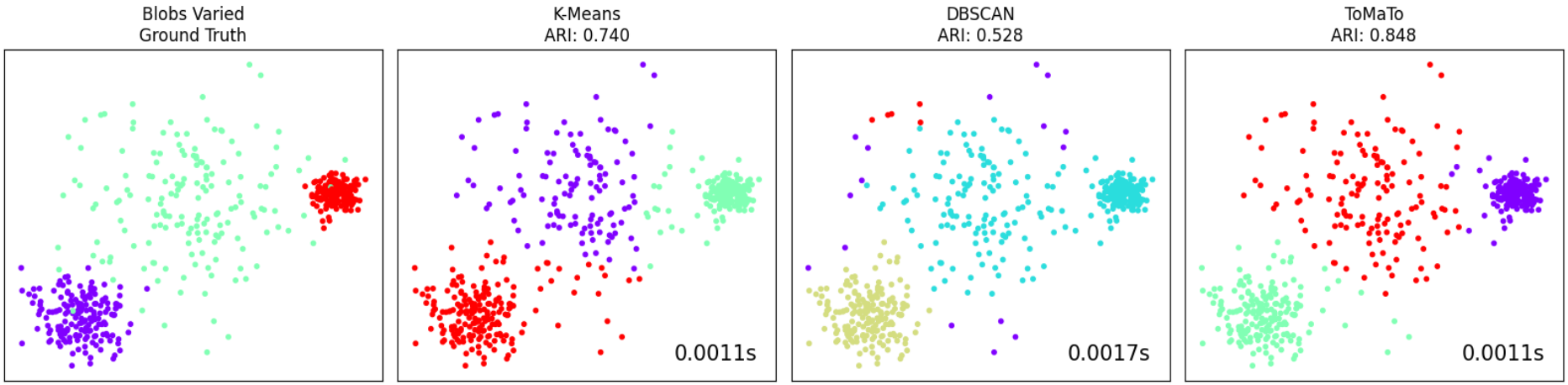
Anisotropic



Blobs


Blobs
Ground Truth
K-Means
ARI: 0.970
0.0036s
DBSCAN
ARI: 0.556
0.0017s
ToMaTo
ARI: 0.970
0.0011s

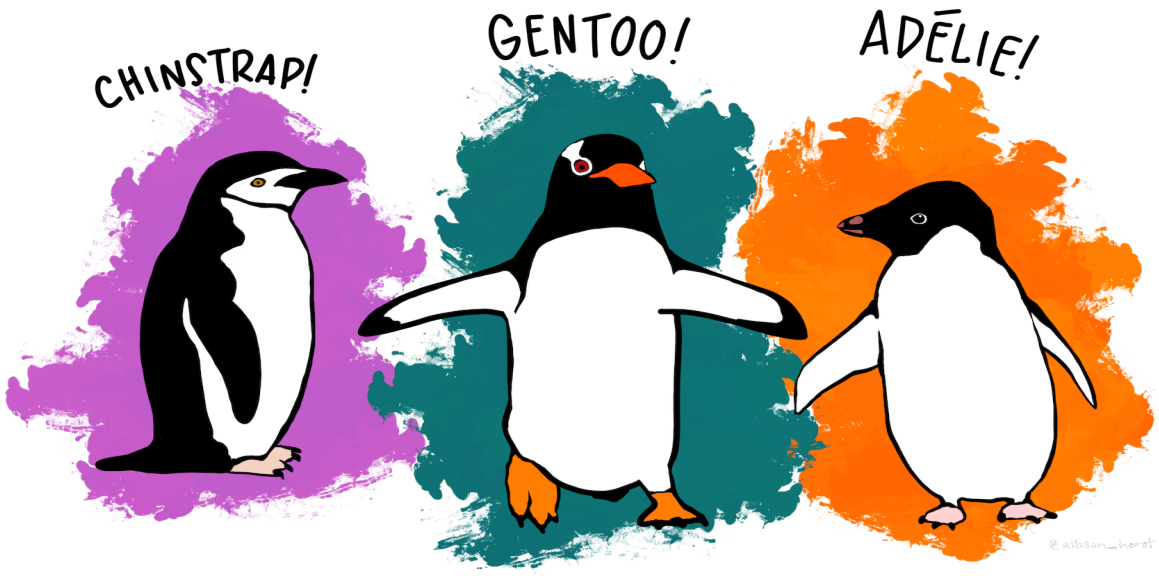
Blobs with different variences



Testing on Real World Dataset

Introduction to dataset

- We use data from the Palmer Penguins dataset.
- It includes the target variable (species) and six additional features: island (nominal; 3 levels), culmen (bill) length (continuous), culmen (bill) depth (continuous), flipper length (continuous), body mass (discrete), and sex (nominal; 2 levels).

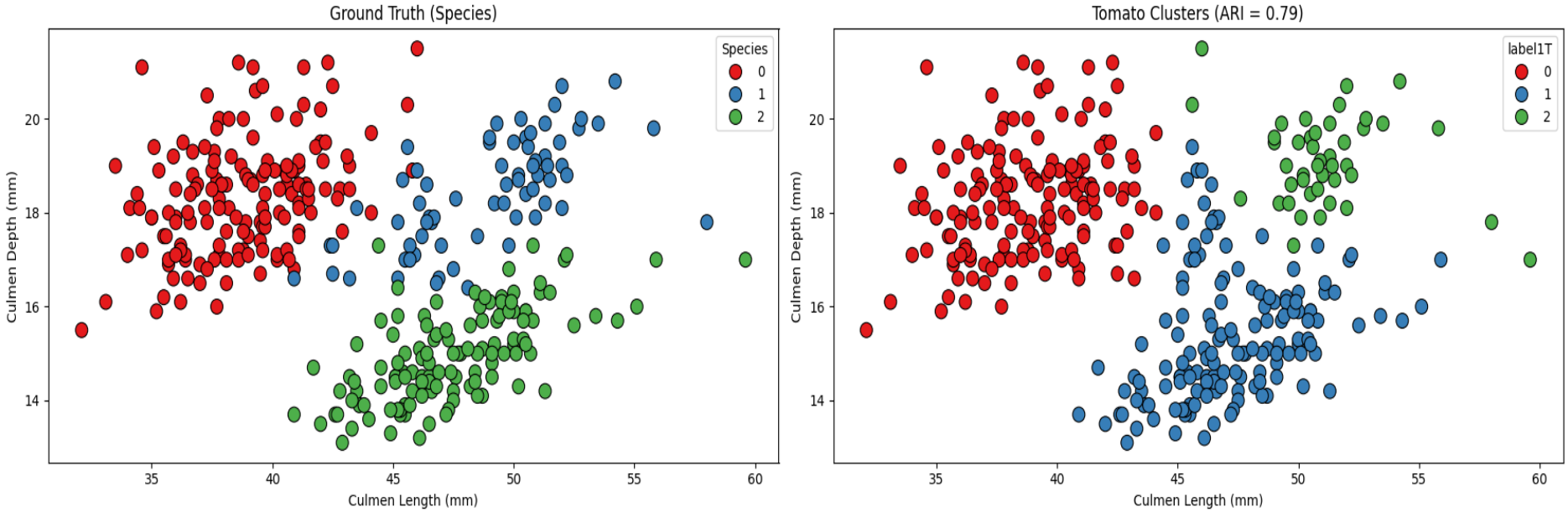


Data Pre-Processing and Cleaning

- The dataset originally contains 344 instances, but 2 with missing values were removed for this study.
- To evaluate the performance, we constructed three scatter plots from the Palmer Penguin dataset, each comparing:
  1. culmen length vs culmen depth
  2. culmen length vs flipper length
  3. culmen length vs body mass

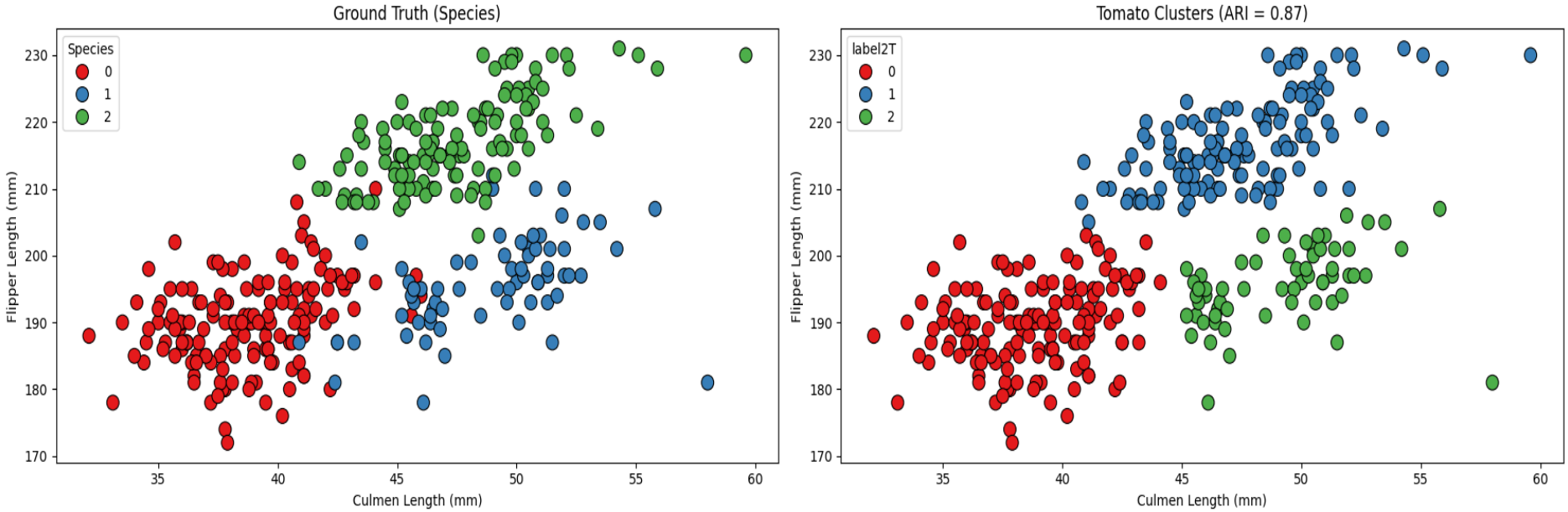
Culmen Length vs Culmen Depth

ARI = 0.79



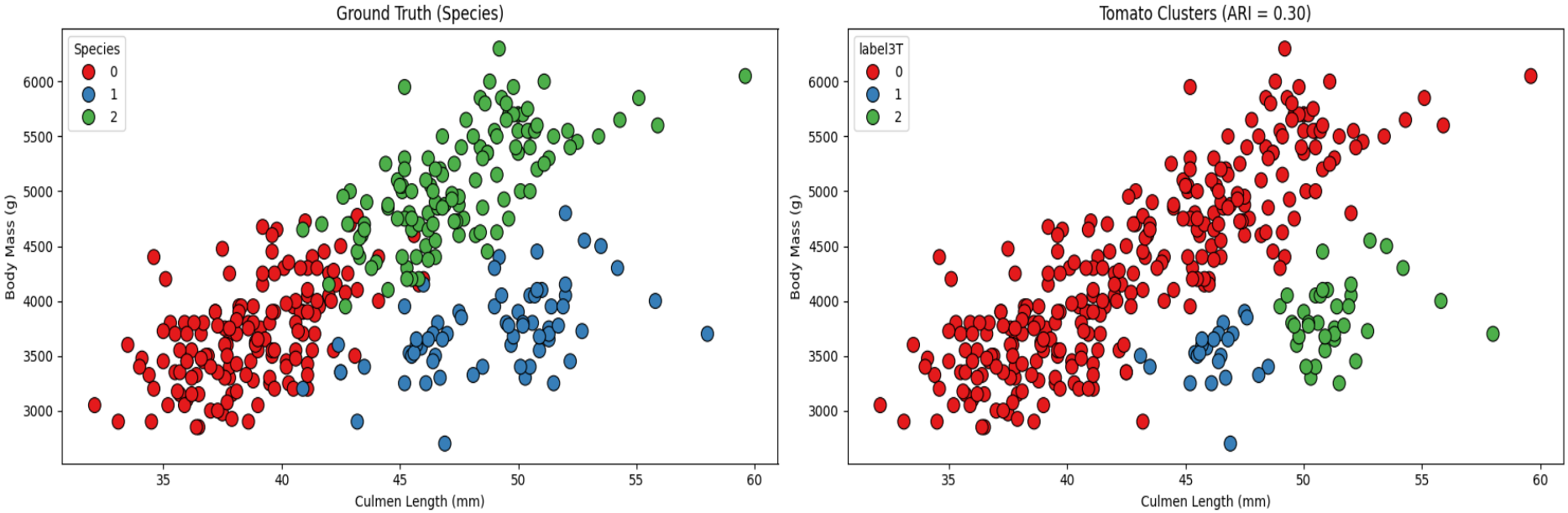
Culmen Length vs Flipper Length

ARI = 0.87



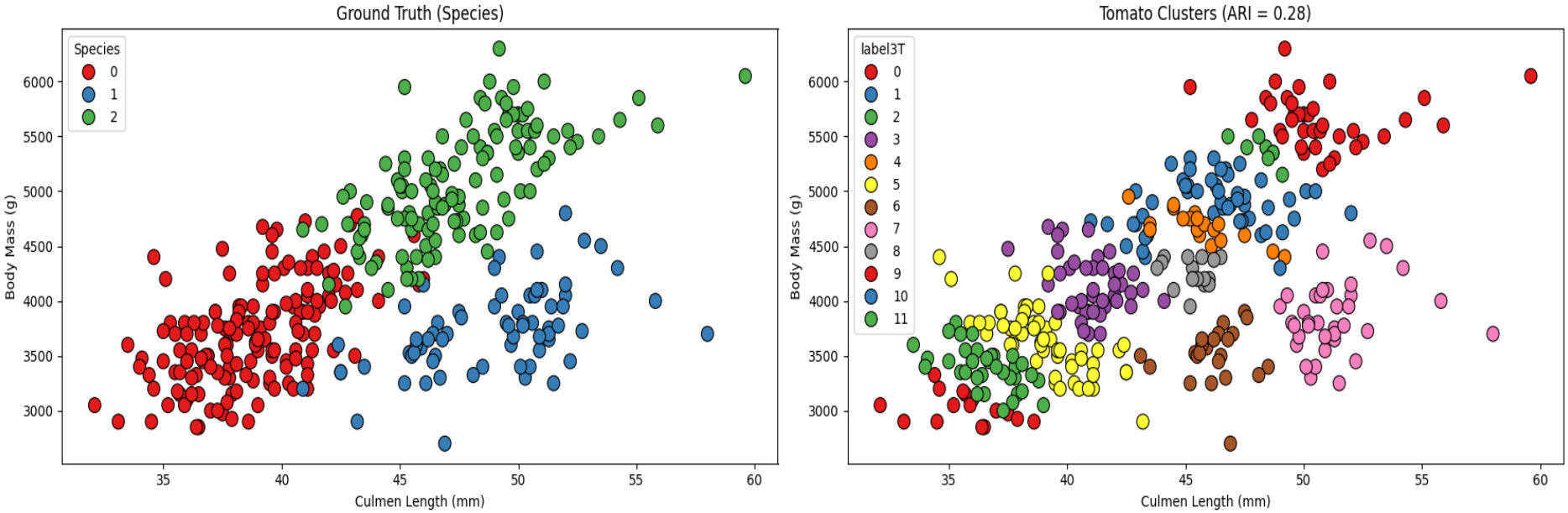
Culmen Length vs Body Mass

ARI = 0.30



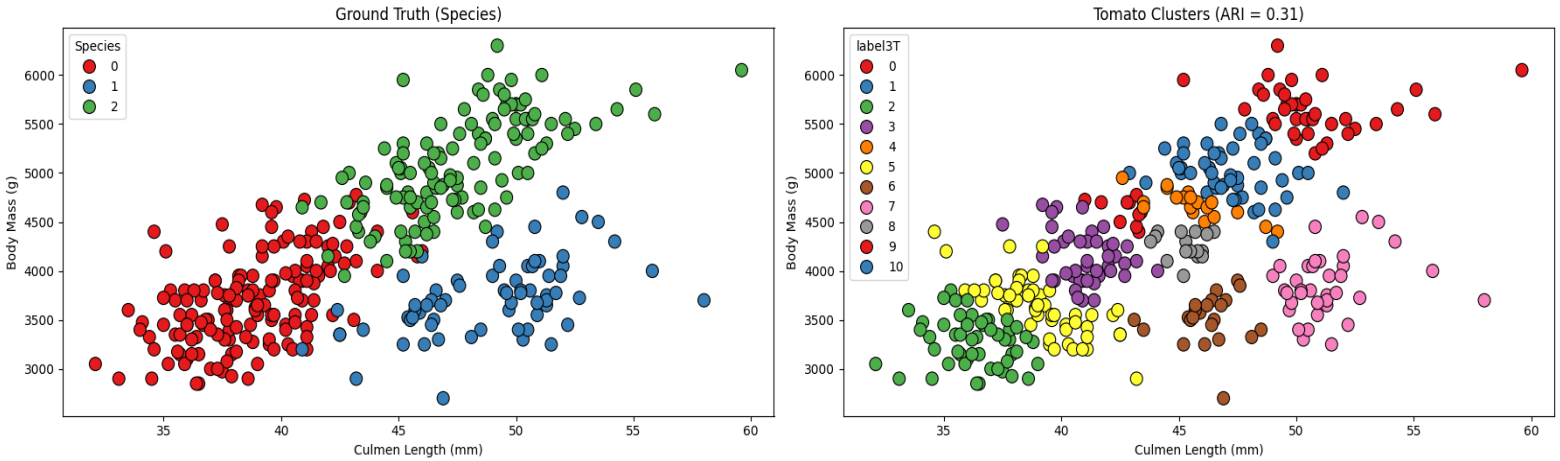
Initial Detected Clusters

ARI = 0.28

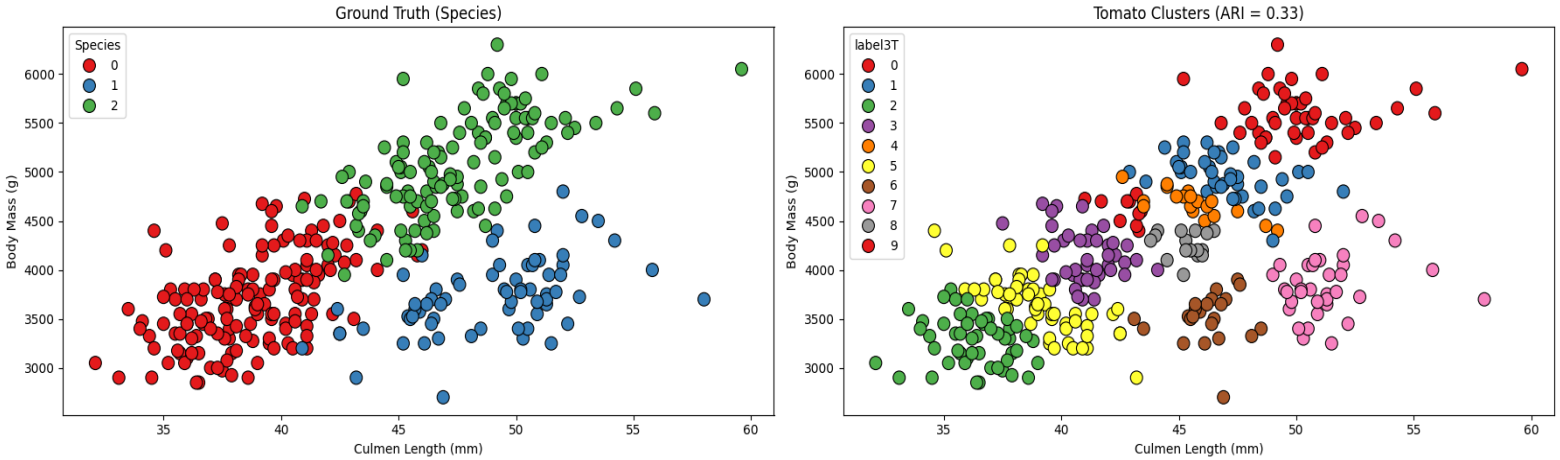


Merging

ARI = 0.31

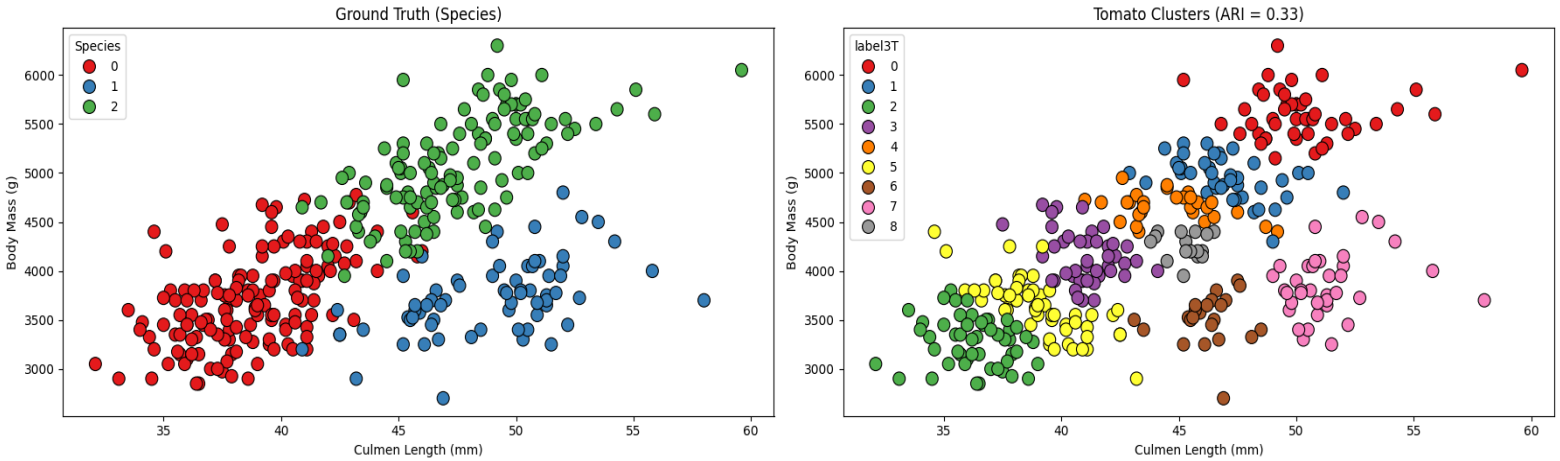


ARI = 0.33

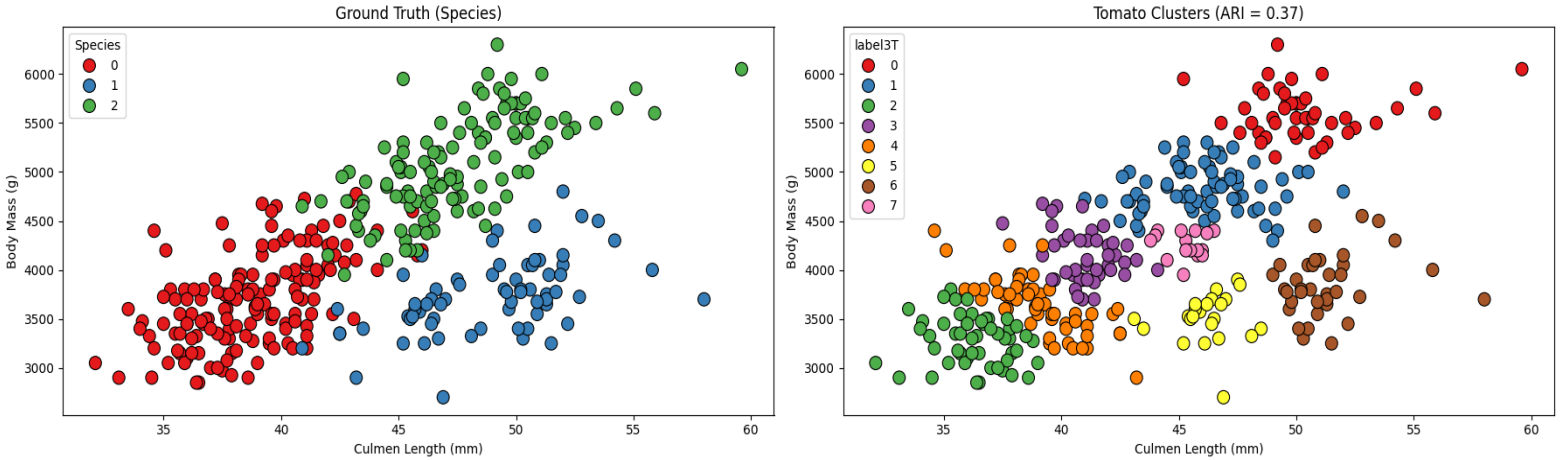


Merging

ARI = 0.33

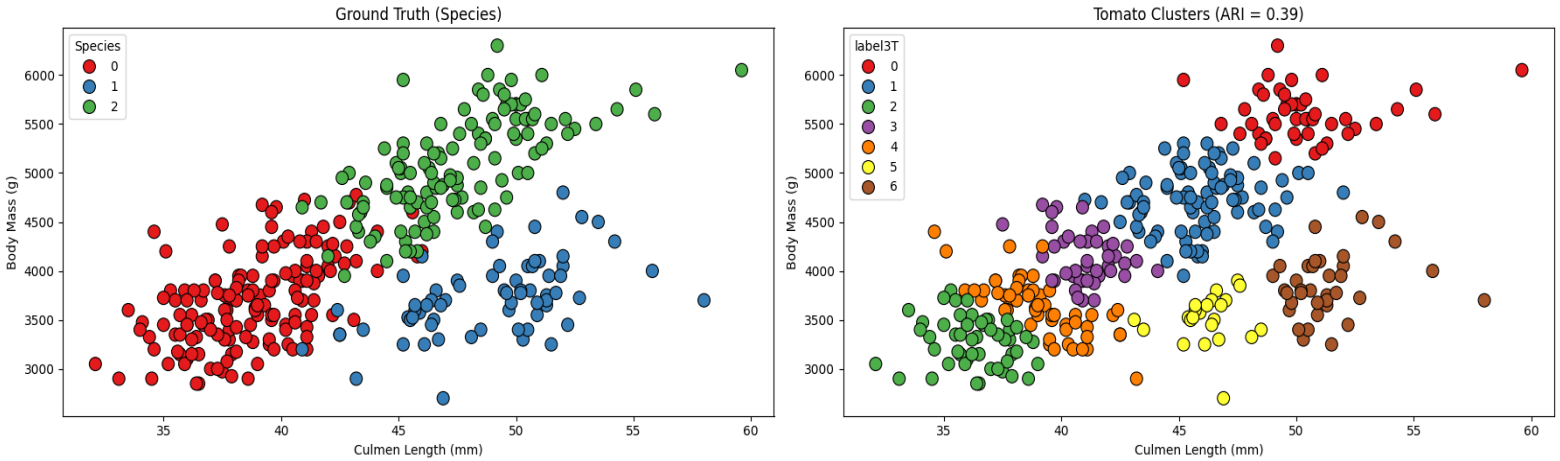


ARI = 0.37

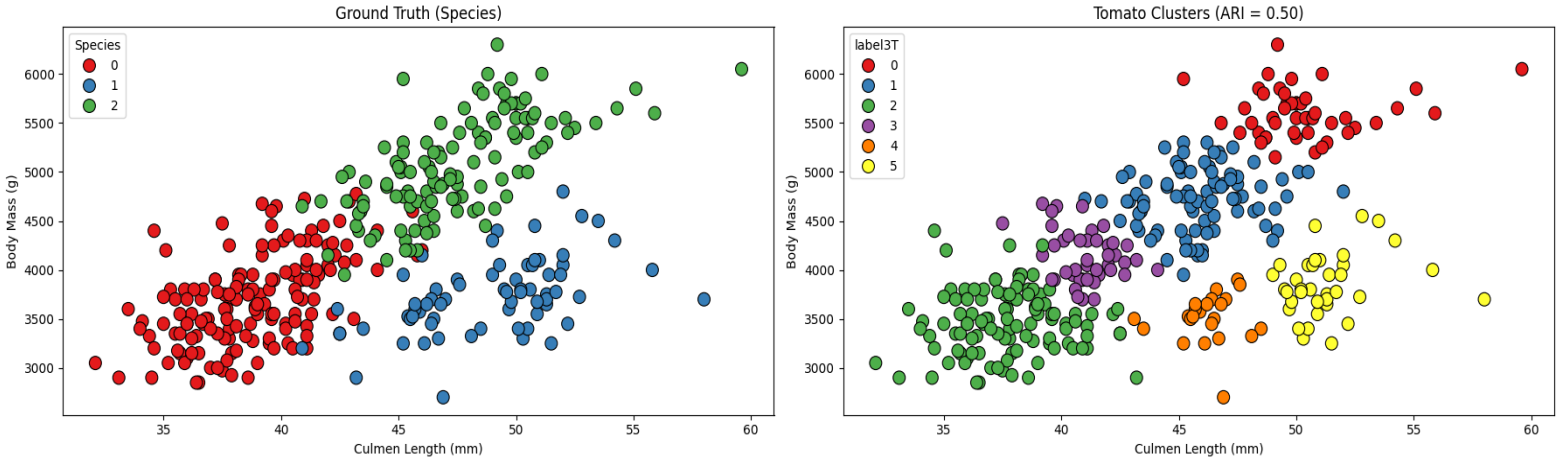


Merging

ARI = 0.39

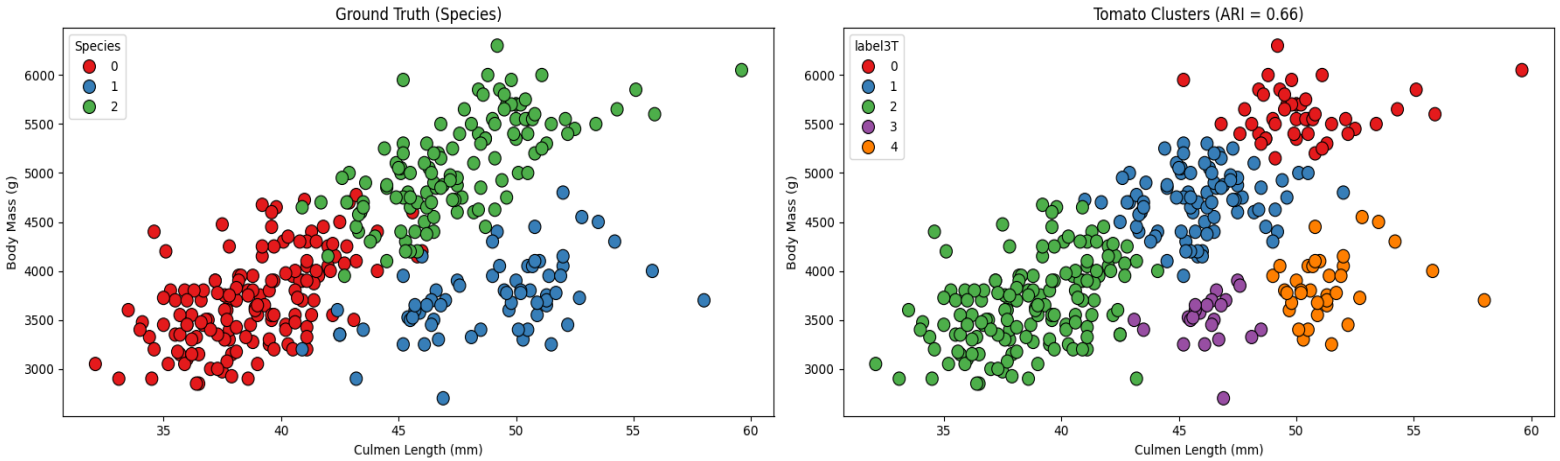


ARI = 0.50

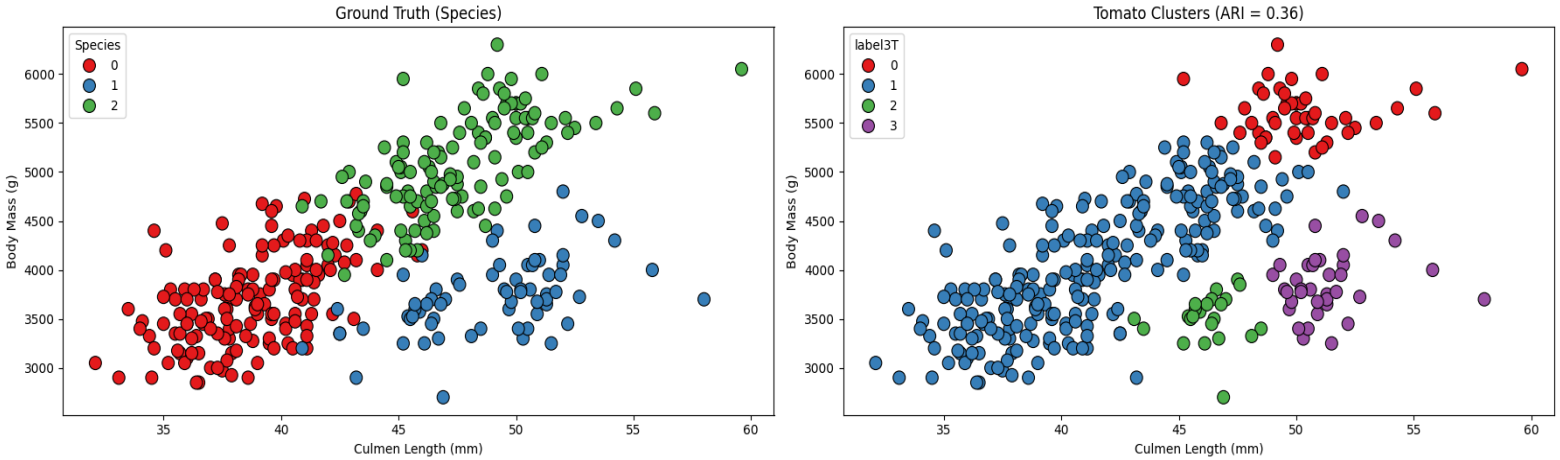


Merging

ARI = 0.66



ARI = 0.36

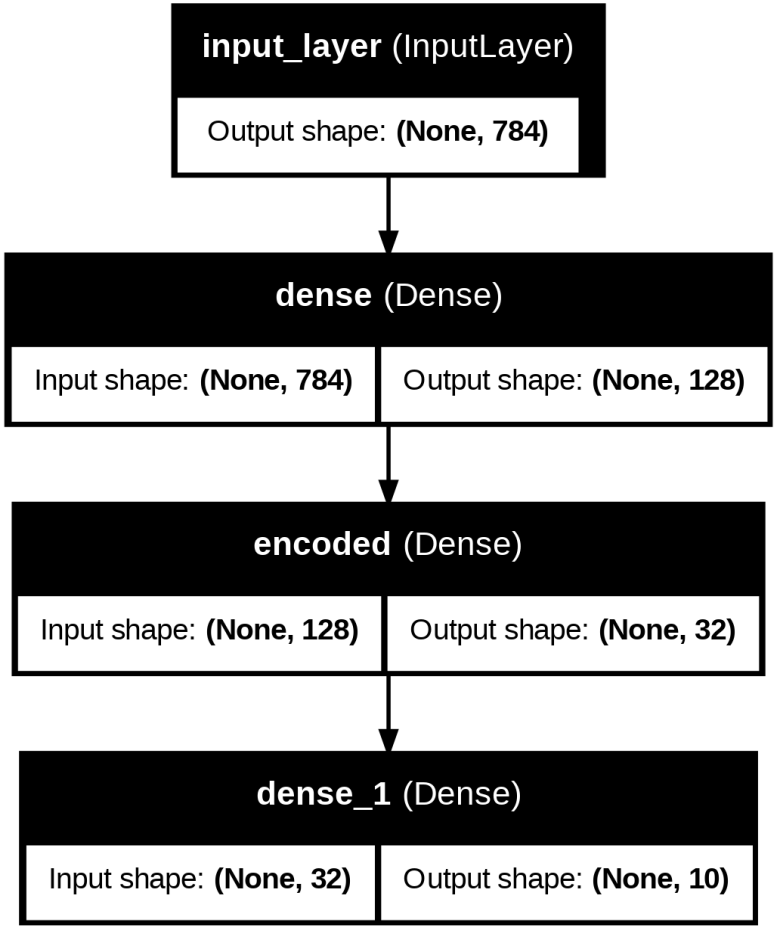


Testing on High-Dimensional Dataset

MNIST Digits

- We evaluate the performance on the MNIST Digit Dataset.
- Due to computational complexity, we first encode the 784-dimensional vectors into 32-dimensional vectors using a simple neural network with the architecture: $784 \to 128 \to \mathbf{32} \to 10$.
- And then we cluster these 32-dimensional vectors using the ToMATo algorithm.

NN Model


input_layer (InputLayer)
Output shape: (None, 784)
dense (Dense)
Input shape: (None, 784)
Output shape: (None, 128)
encoded (Dense)
Input shape: (None, 128)
Output shape: (None, 32)
dense_1 (Dense)
Input shape: (None, 32)
Output shape: (None, 10)

Applying ToMATo

- After obtaining the 32-dimensional embeddings, we apply the ToMATo algorithm for clustering.
- The algorithm is initialized using a k-NN graph with $k = 200$ and $k = 300$.

Table: Adjusted Rand Index (ARI) for Different Methods

Method	ARI
ToMATo ($k = 200$)	0.7844
ToMATo ($k = 300$)	0.7432
K-Means	0.5203

MNIST Clusters Visualization (ToMATo, $k = 200$)

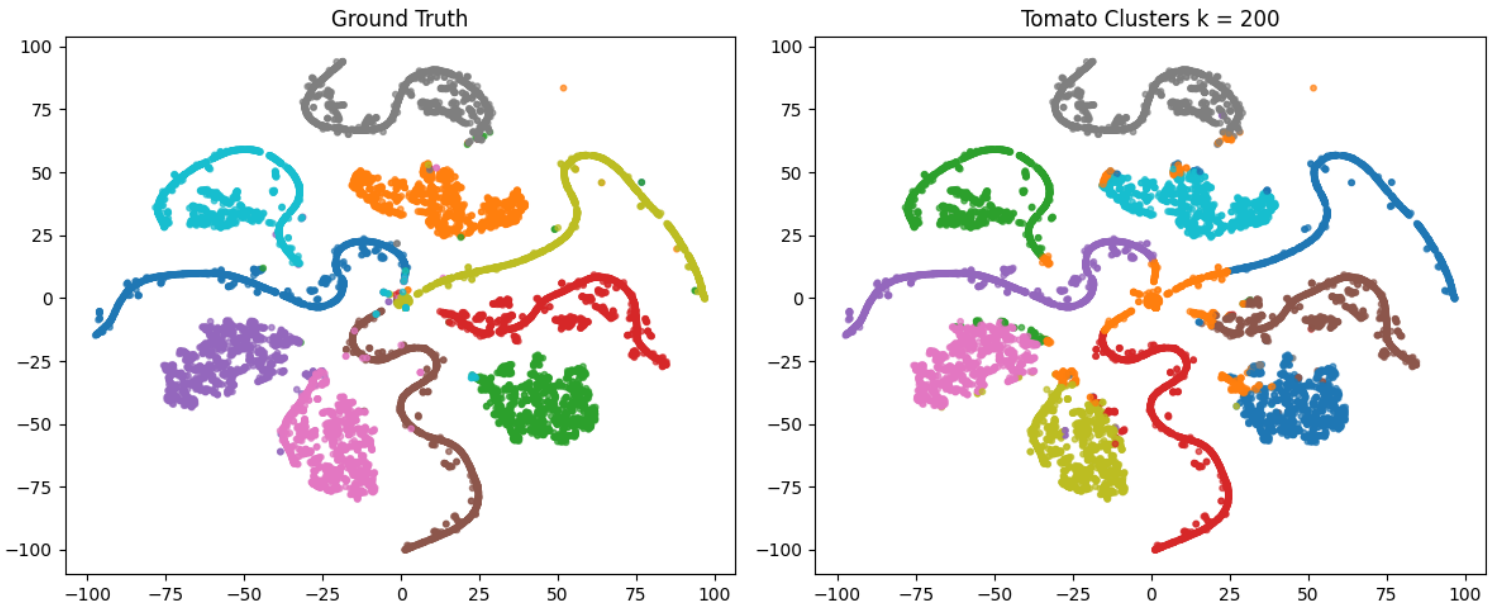


Figure: Clustering results using ToMATo ($k = 200$).

MNIST Clusters Visualization (ToMATo, $k = 300$)

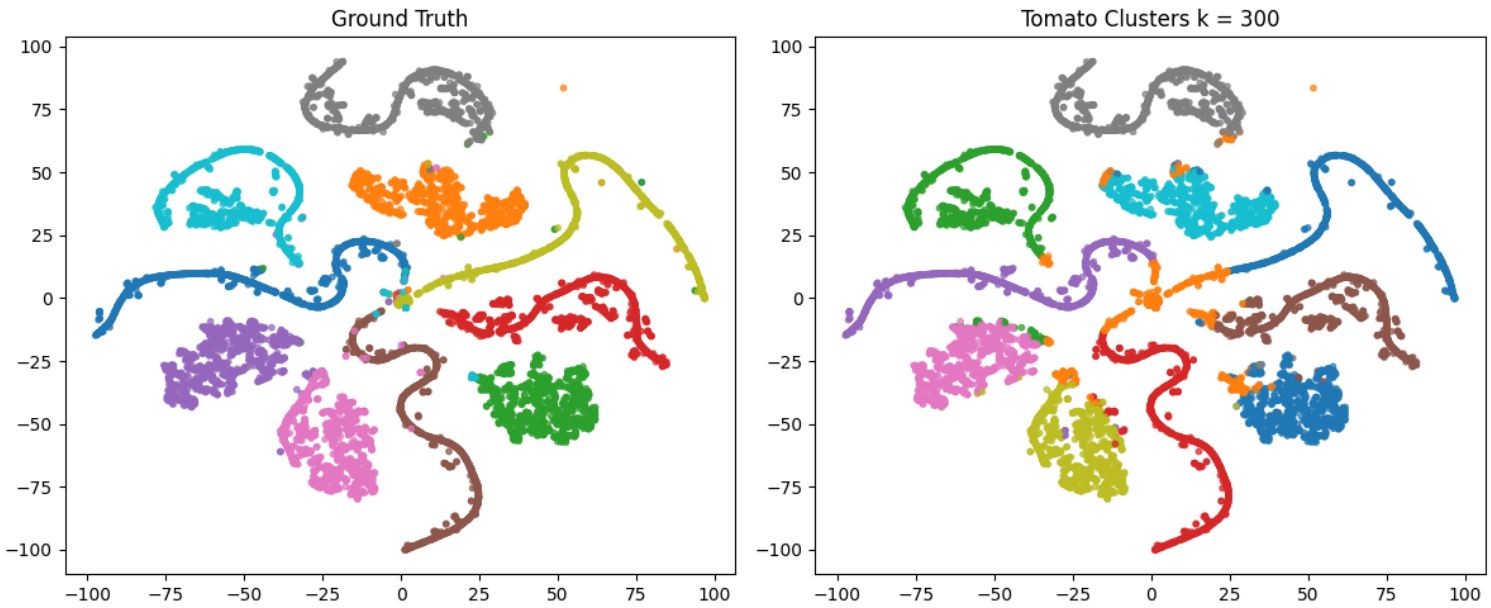


Figure: Clustering results using ToMATo ($k = 300$).

MNIST Clusters Visualization (K-Means)

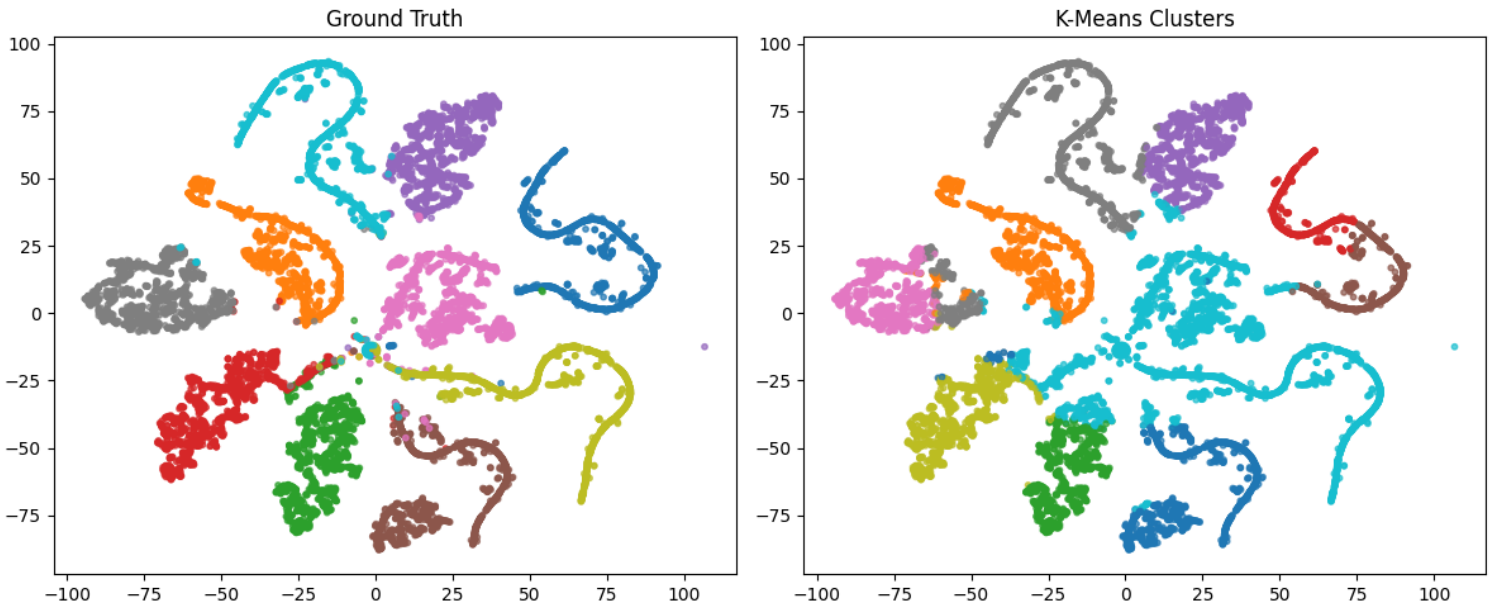


Figure: Clustering results using K-Means.

Class-Wise Accuracies

Table: Class-Wise Accuracies for ToMATo ($k = 200$) and K-Means

Class	ToMATo ($k = 200$)	K-Means
0	0.9450	0.9160
1	0.9090	0.9160
2	0.8730	0.5150
3	0.8970	0.8530
4	0.8820	0.0000
5	0.9180	0.8460
6	0.9110	0.9140
7	0.9720	0.5210
8	0.2020	1.0000
9	0.9500	0.0000
Overall	0.8459	0.6481

Cluster Analysis

Since this is an unsupervised task, we analyze the most frequent predicted cluster for each true class.

Table: Most Frequent Predicted Clusters for Each True Class

True Class	K-Means	ToMATo
0	5 (916)	4 (945)
1	8 (916)	9 (909)
2	7 (515)	0 (873)
3	4 (853)	5 (897)
4	0 (615)	6 (882)
5	2 (846)	3 (918)
6	0 (914)	8 (911)
7	9 (521)	7 (972)
8	6 (1000)	0 (798)
9	6 (1000)	2 (950)

Cluster Analysis

K-Means groups classes 4 and 6 together as class 0, and classes 9 and 8 together. In contrast, ToMATo misclassifies class 8 and class 2 together but provides a more balanced clustering overall.

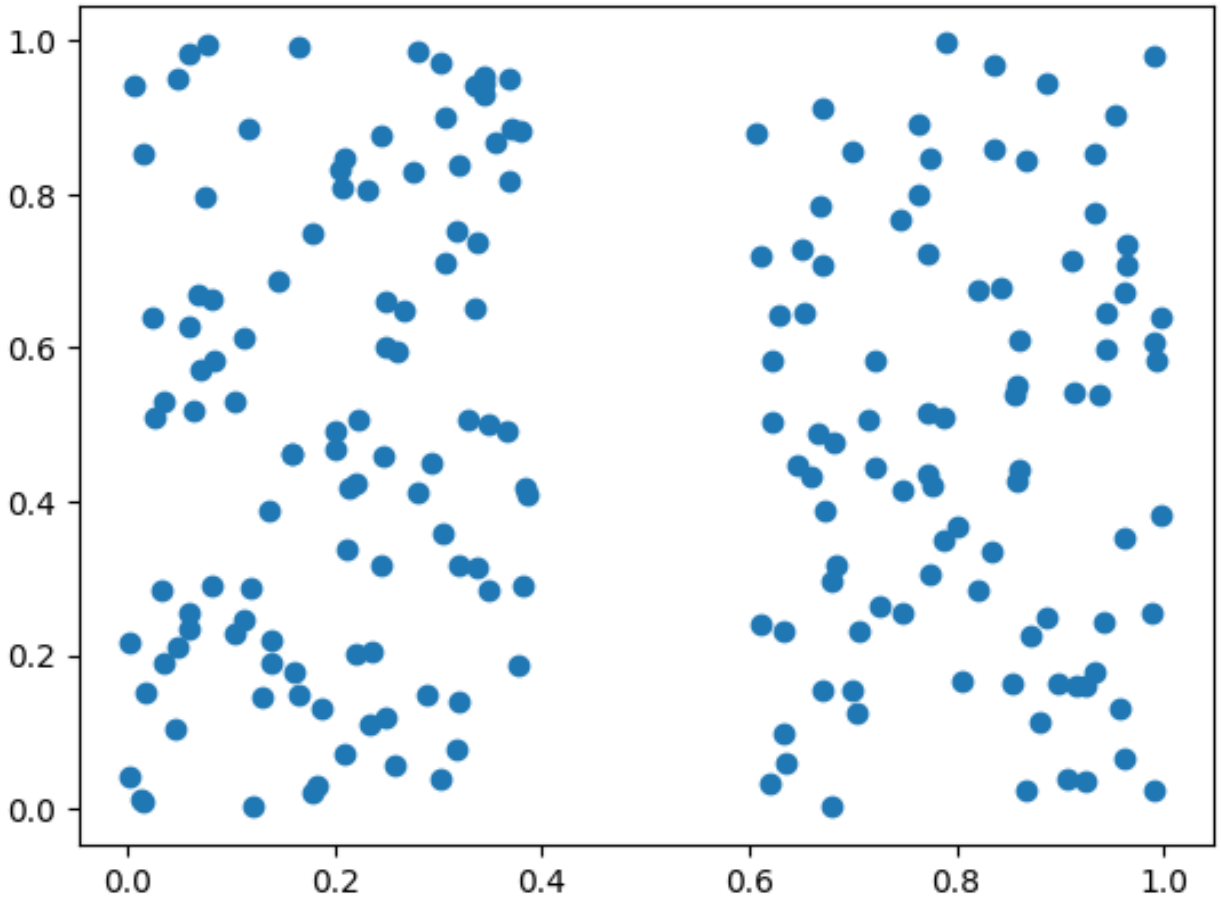
Limitations

Parameter Sensitivity and Tuning

- ToMaTo algorithm is highly sensitive to parameters
- Requires two free parameters: neighborhood scale (δ)/number of neighbours(k) and persistence threshold (τ).
- Small variations can alter merging outcomes.

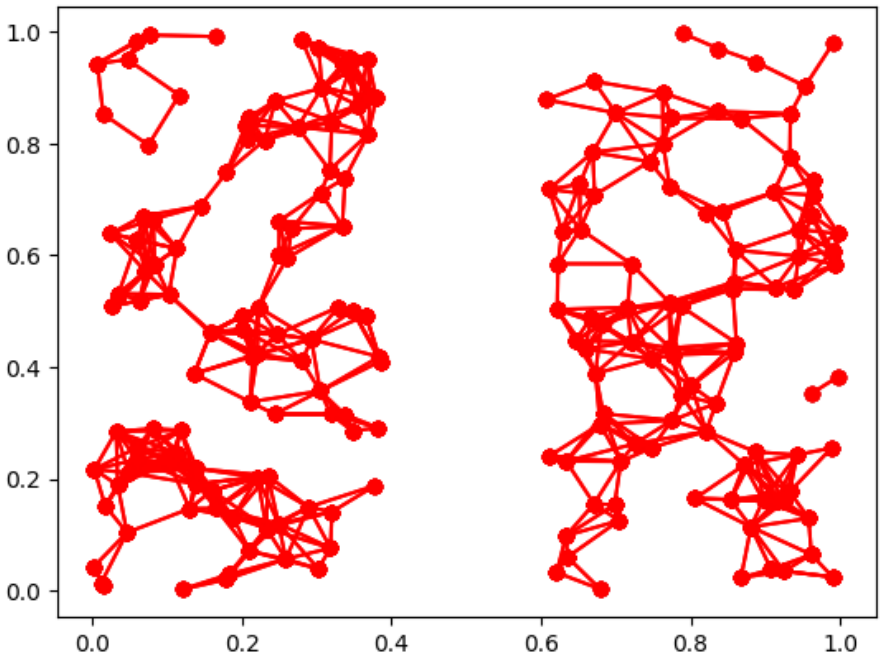
Example - 1

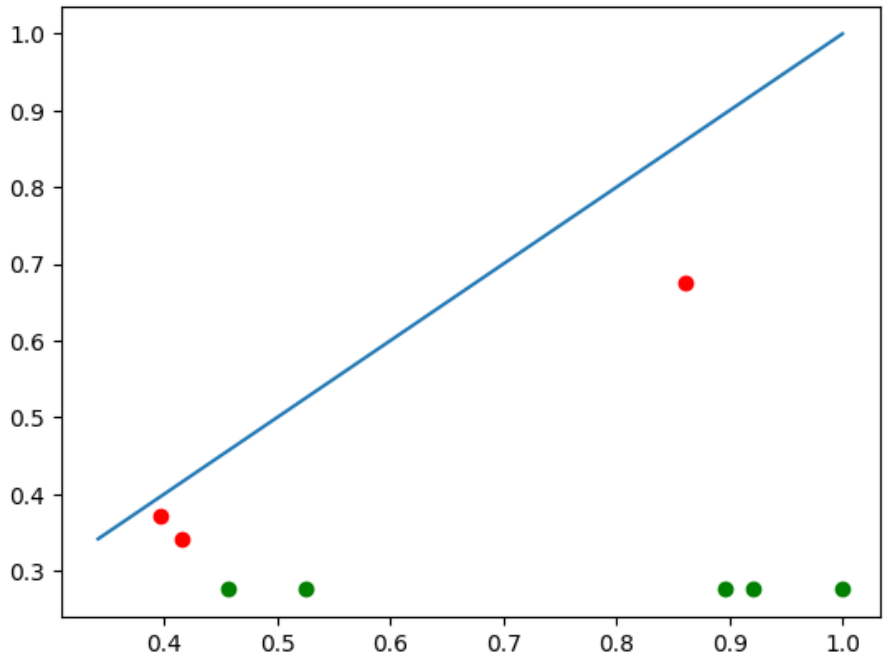
For the following point cloud data



Epsilon-Ball Neighborhood (r = 0.1)

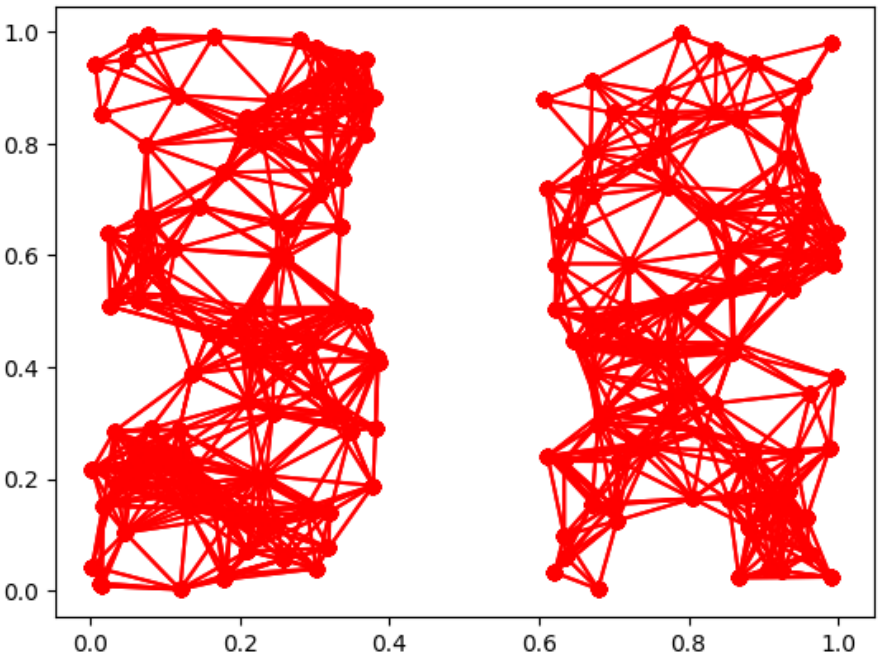
When we initialize the graph with an epsilon-ball neighborhood for $r = 0.1$, we obtain:
The algorithm 'naturally' detects 8 clusters, which can be later merged using persistence

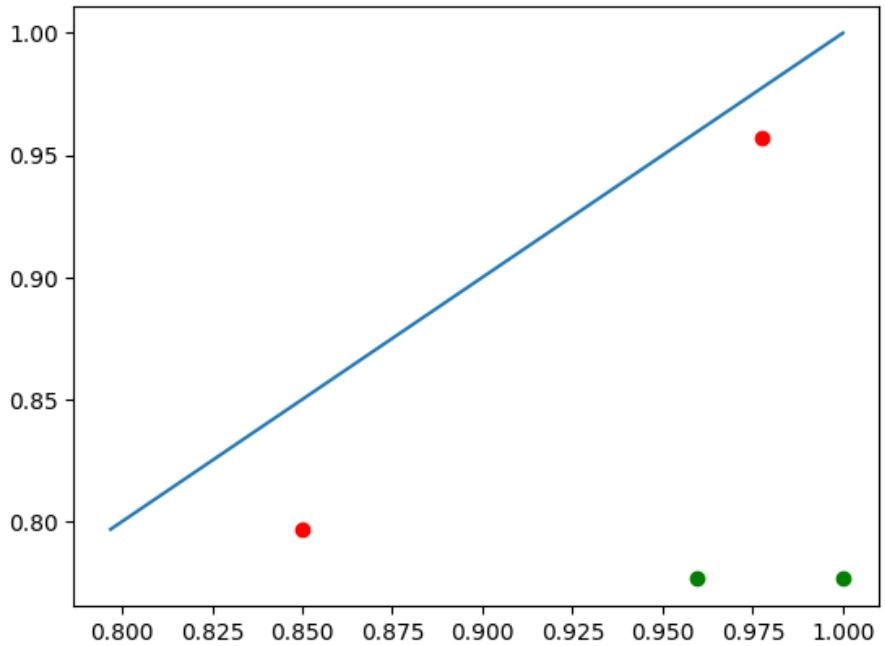




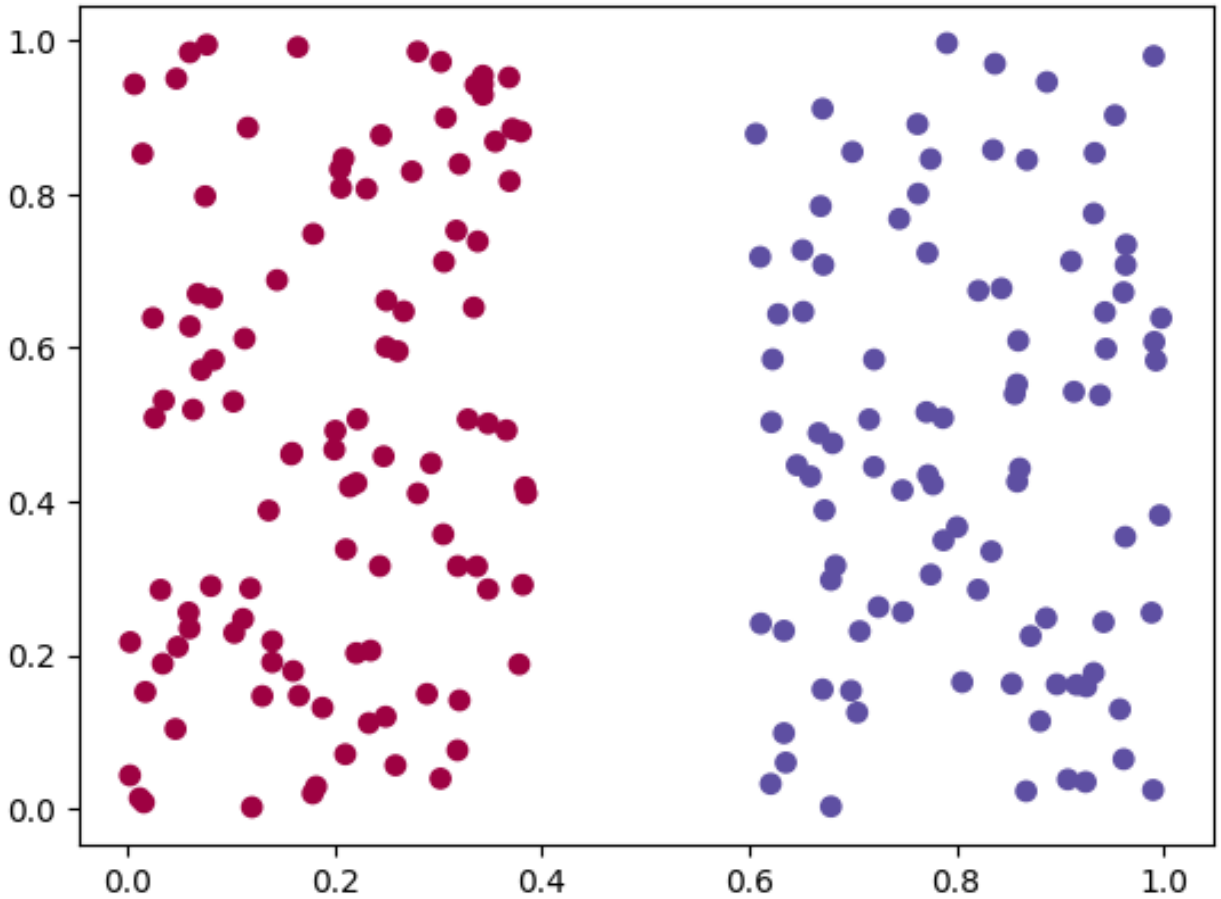
Epsilon-Ball Neighborhood (r = 0.13)

When we initialize the graph with an epsilon-ball neighborhood for $r = 0.13$, we obtain:
The algorithm 'naturally' detects 4 clusters, which can be later merged using persistence





The final clusters for r=0.13



Example - 2

While clustering MNIST digits we used kNN for initial graph:

- for k $= 5$ we get *The number of clusters required 10 is smaller than the number of connected components 46*
- for k $= 200$ we get the ARI score of 0.7844
- for k $= 300$ we get the ARI score of 0.7432
- for k $= 1000$ we get *The number of clusters required 10 is larger than the number of mini-clusters 2*

Dependence on Density Estimation

- Relies on accurate estimation of the density function f (e.g., via kernel methods).
- Misestimation can lead to incorrect clustering.

High-Dimensional Limitations

- Persistent homology via Rips complexes is practical only in moderate dimensions.
- High-dimensional data increases sparsity and computational challenges.
- This may lead to unreliable clustering outcomes.

References

- Baris Coskunuzer and Cuneyt Gürcan Akçora. *Topological Methods in Machine Learning: A Tutorial for Practitioners*. University of Texas at Dallas, USA; University of Central Florida, USA, 2023. Available at: `https://arxiv.org/abs/2409.02901`
- Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. *Persistence-Based Clustering in Riemannian Manifolds*. 2013. Available at: `https://doi.org/10.1145/2535927`
- Kara Combs and Trevor J. Bihl. *Clustering and Topological Data Analysis: Comparison and Application*. 2023. Available at: `https://aisel.aisnet.org/hicss-56/da/big_data_and_analytics/4/`
- Bastian Rieck. *TDA for ML - Lec1*. 2020. Available at: `https://bastian.rieck.me/talks/ECML_PKDD_2020_Lecture_1.pdf`
- Guide ToMATo - Jupyter Notebook. *ToMATo: Topological Mode Analysis Tool*. Available at: `https://github.com/xetaxe/ToMATo-Notebook/blob/master/Guide%20ToMATo.ipynb`