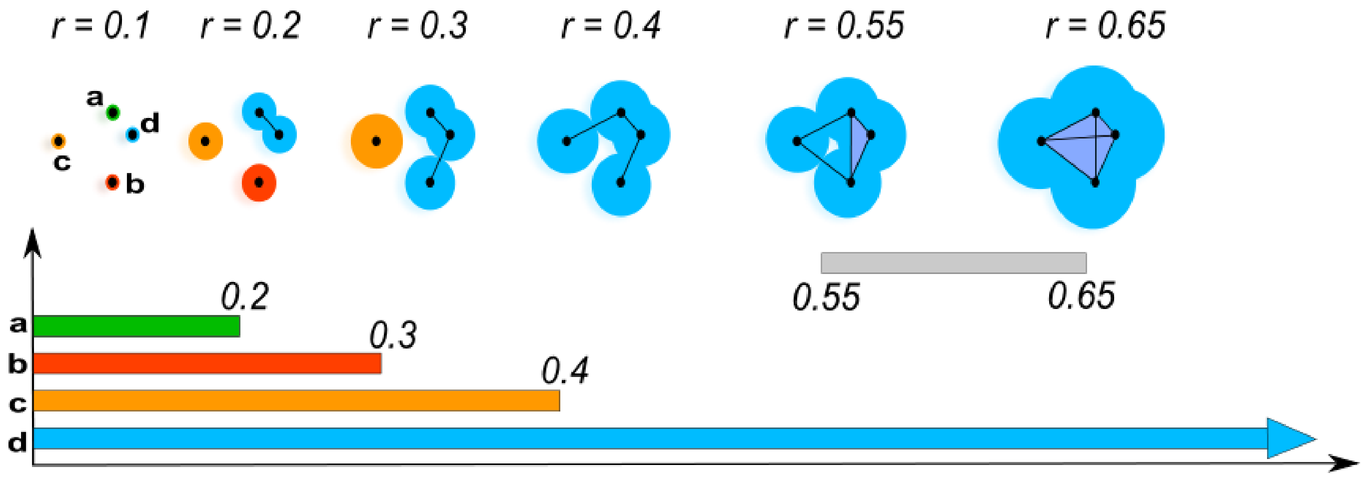
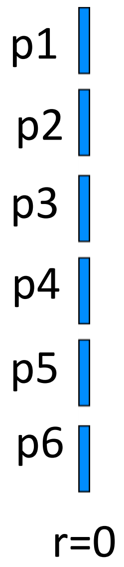
An introduction to persistent homology
Computing the 0-barcodes



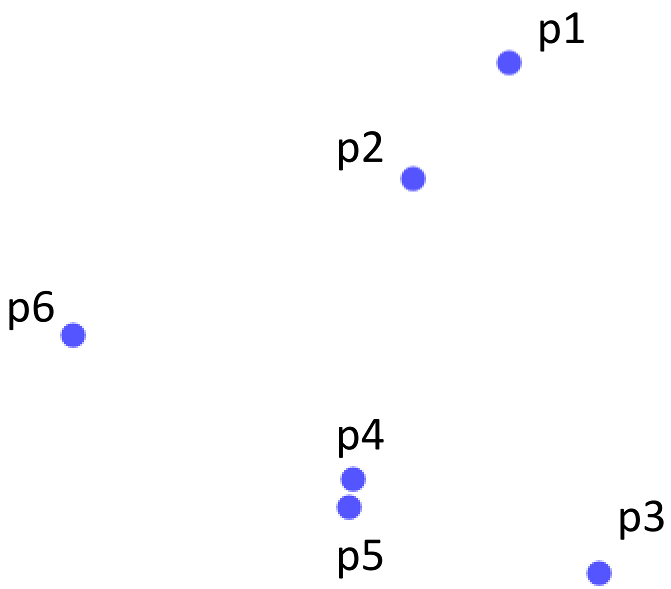
MUSTAFA HAJIJ

Persistence diagram

- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.

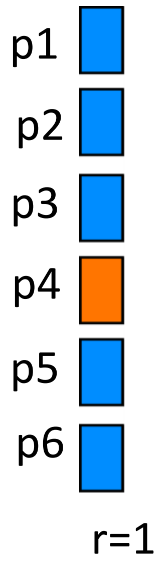


r=0

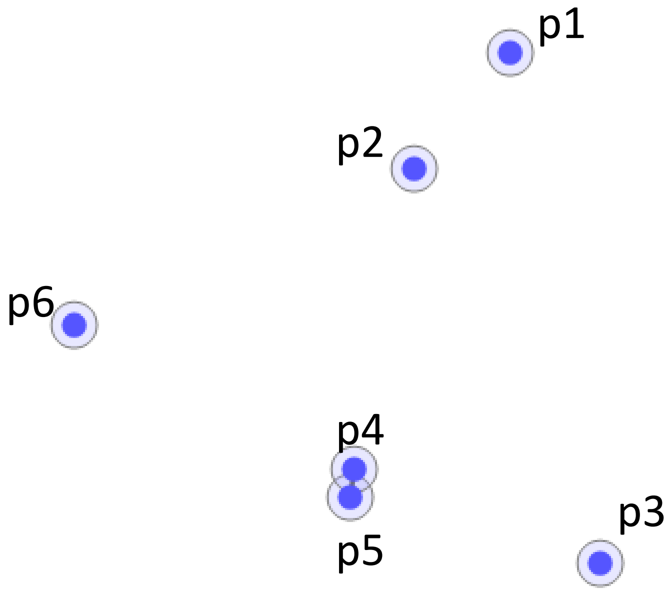


Persistence diagram

- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



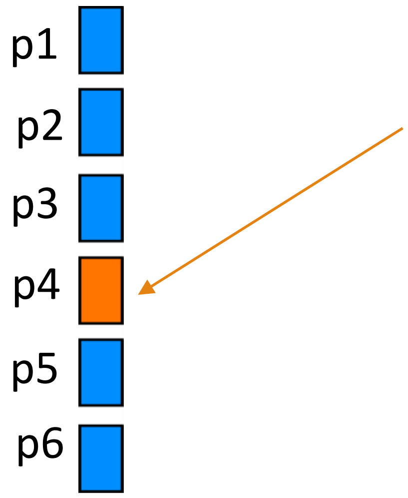
r=1



p4 and p5 merge.
We record this event in the bars
by deciding not to grow one of them anymore (death event)

Persistence diagram

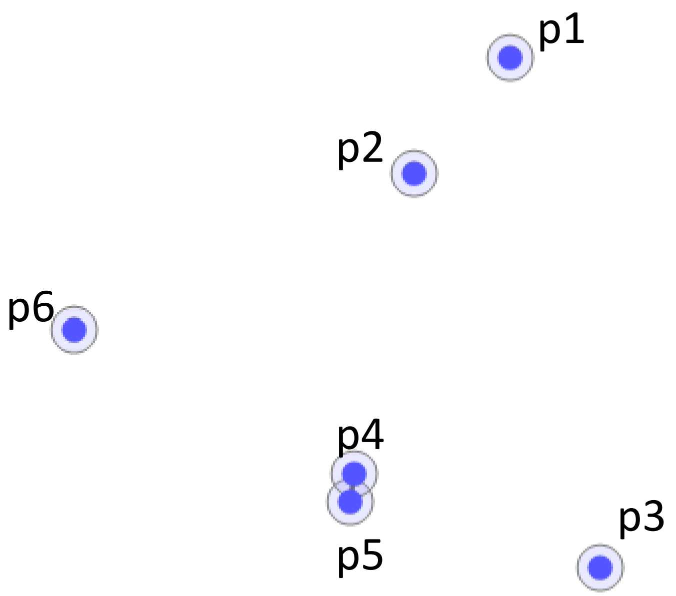
- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



r=1

Here we chose the connected component of p4 to die and p5 to live (the choice is random). Now p5 represents the connected component of p5 and p4 but we will still call it p5

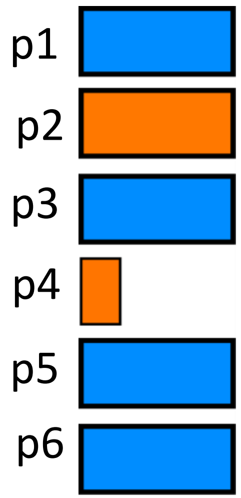
r=1



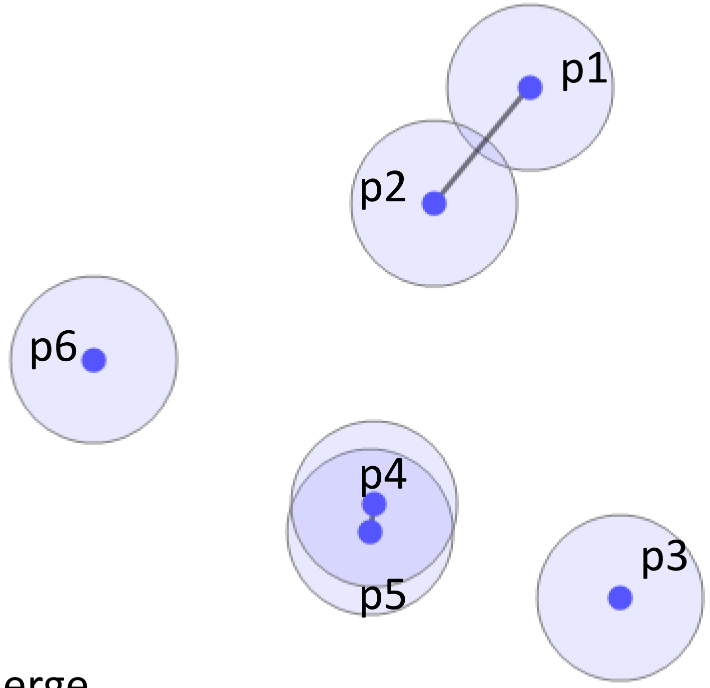
p4 and p5 merge.
We record this event in the bars
by deciding not to grow one of them anymore (death event)

Persistence diagram

- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



r=3

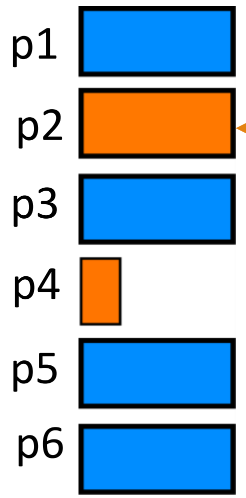


r=3

p1 and p2 merge.
We record this event in the bars
by deciding not to grow one of them anymore (death event)

Persistence diagram

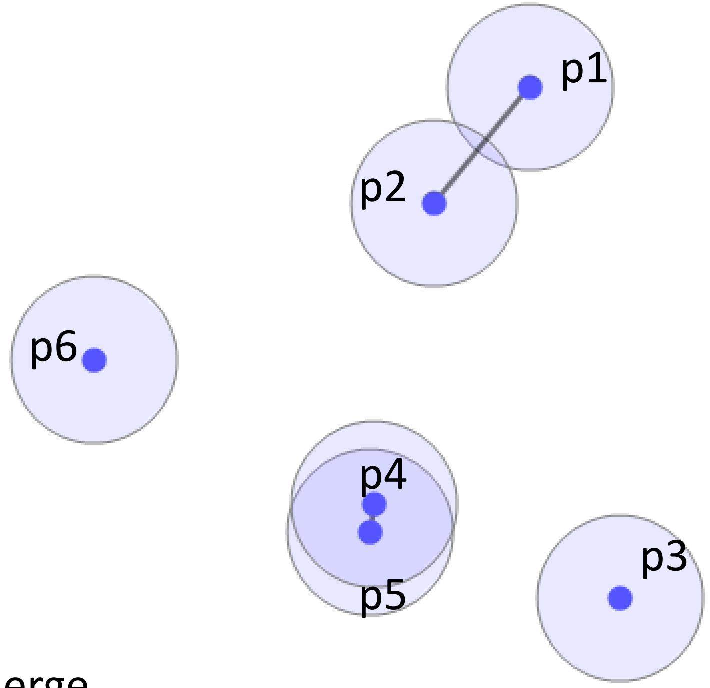
- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



r=3

Here we chose the connected component of p2 to die and the connected component of p1 to live. Now p1 (the one that lives) represents the connected component of p1 and p2.

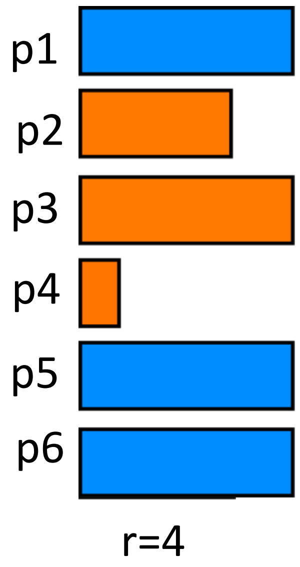
r=3



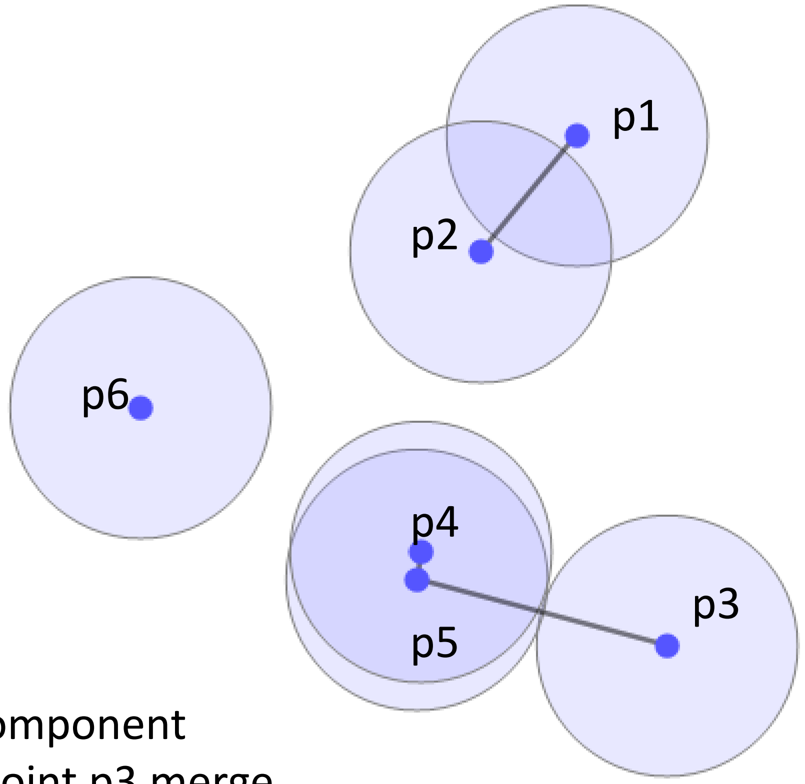
p1 and p2 merge.
We record this event in the bars
by deciding not to grow one of them anymore (death event)

Persistence diagram

- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



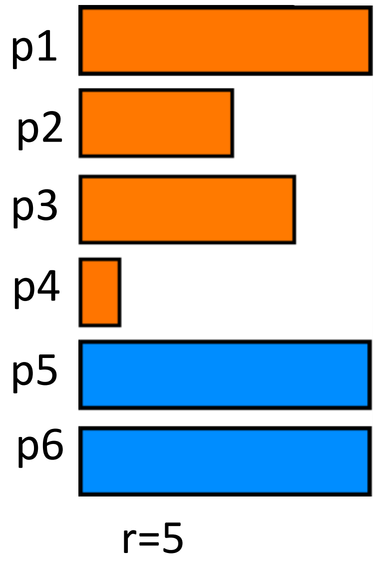
r=4



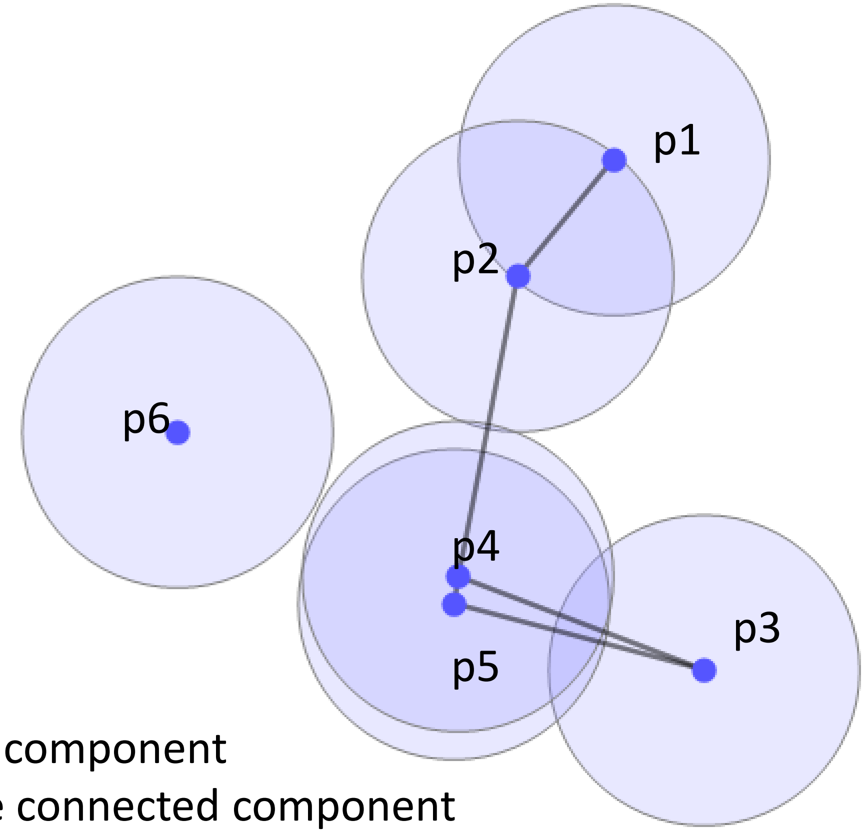
The connected component (p5,p4) and the point p3 merge we record this event in the bars by deciding to stop growing one bar of these connected components

Persistence diagram

- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



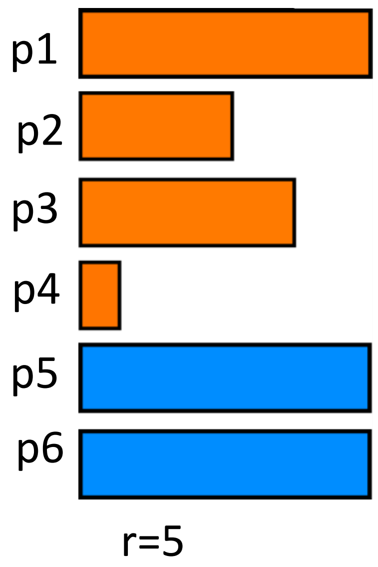
r=5



The connected component (p1,p2) and the connected component (p3,p4,p5) merge we record this event in the bars by deciding to stop growing one bar of these connected components

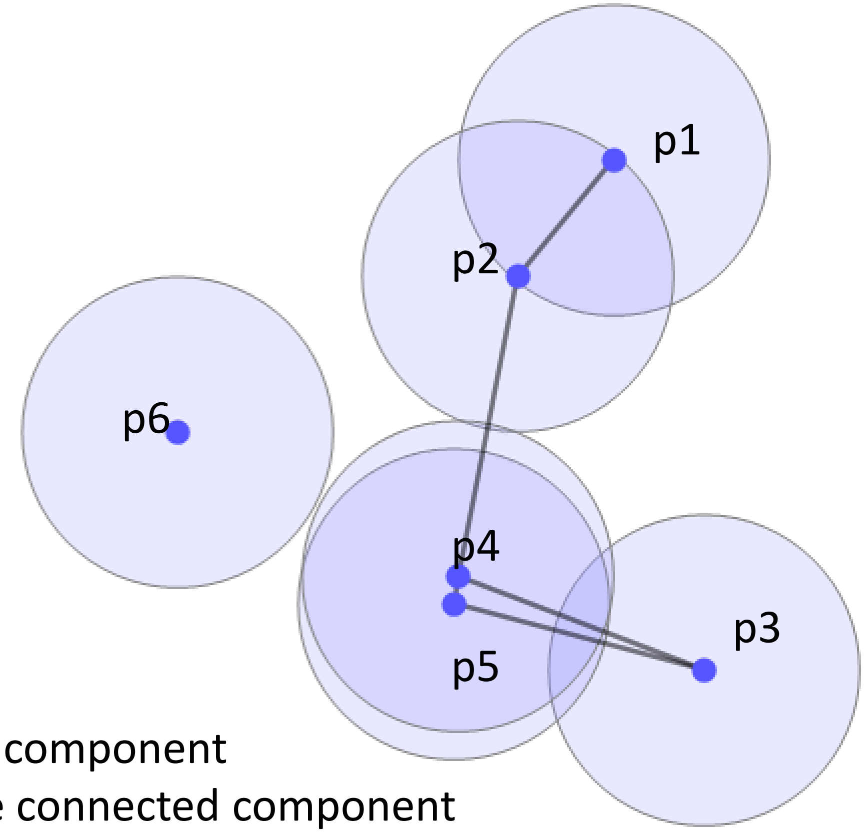
Persistence diagram

- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



Here we chose the connected component of (p1) to die and the connected component of p5 to live. Now p5 (the one that lives) represents the connected component of p1, p2,p3,p4 and p5.

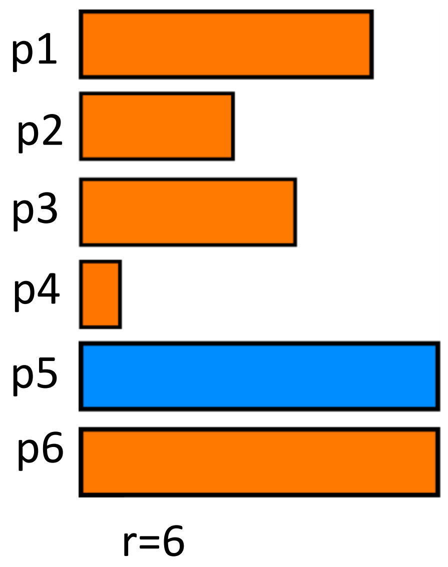
r=5



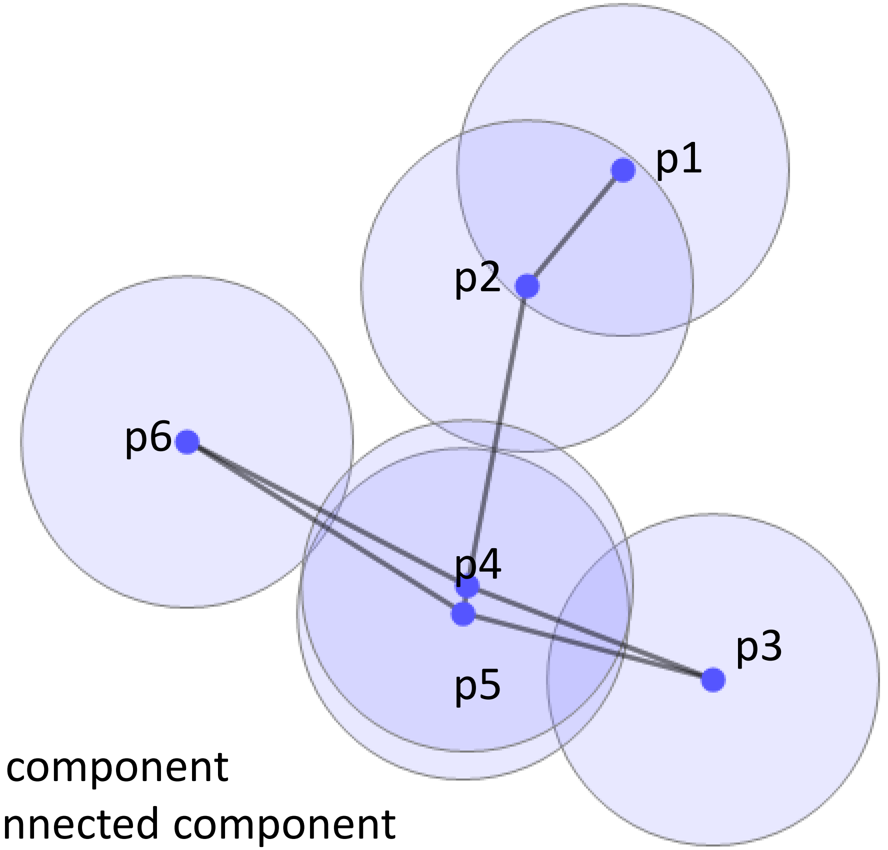
The connected component (p1,p2) and the connected component (p3,p4,p5) merge we record this event in the bars by deciding to stop growing one bar of these connected components

Persistence diagram

- Consider the following point cloud
- Around each point we will grow a disk centered at each point just like what we did the h-clustering single linkage clustering algorithm
- We also create a bar for each point and the length of that bar represents the radius of the disk around each point.



r=6

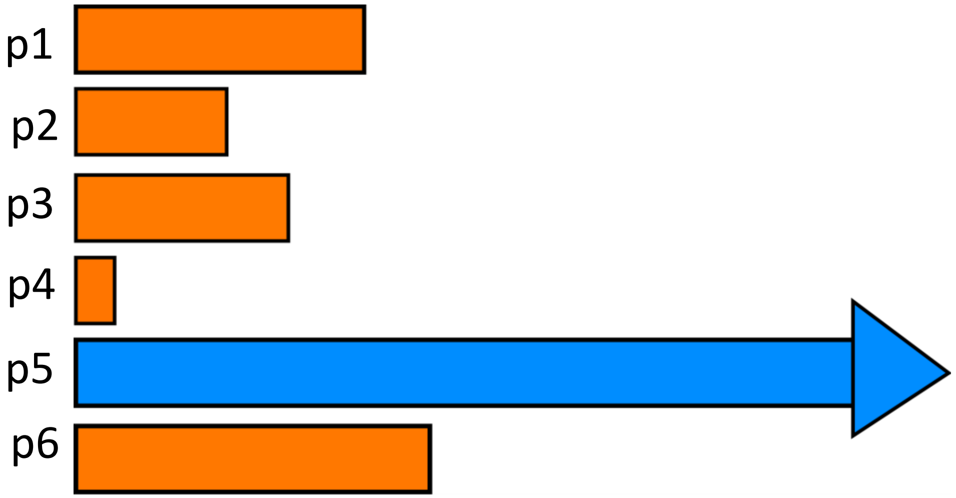


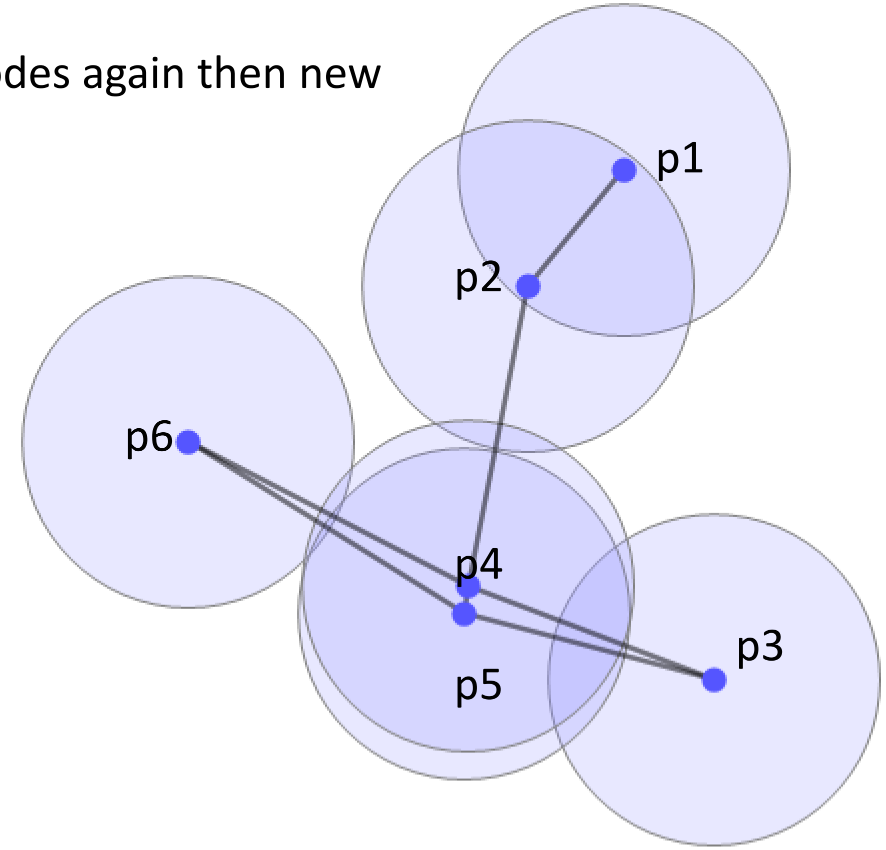
The connected component
(p6) and the connected component
(p1,p2 p3,p4,p5) merge we record this event in the bars
by deciding to stop growing one bar of these connected components

Persistence diagram

- The resulting diagram barcode is called the 0-barcode.
- The 0-barcode is a signature of the point cloud. It encodes topological and geometrical information of the point cloud in meaningful way.
- Long bars represents natural connected components.
- Short bars represent points that are close to each other.
- If we change the point cloud by a little bit, and recompute the barcodes again then new

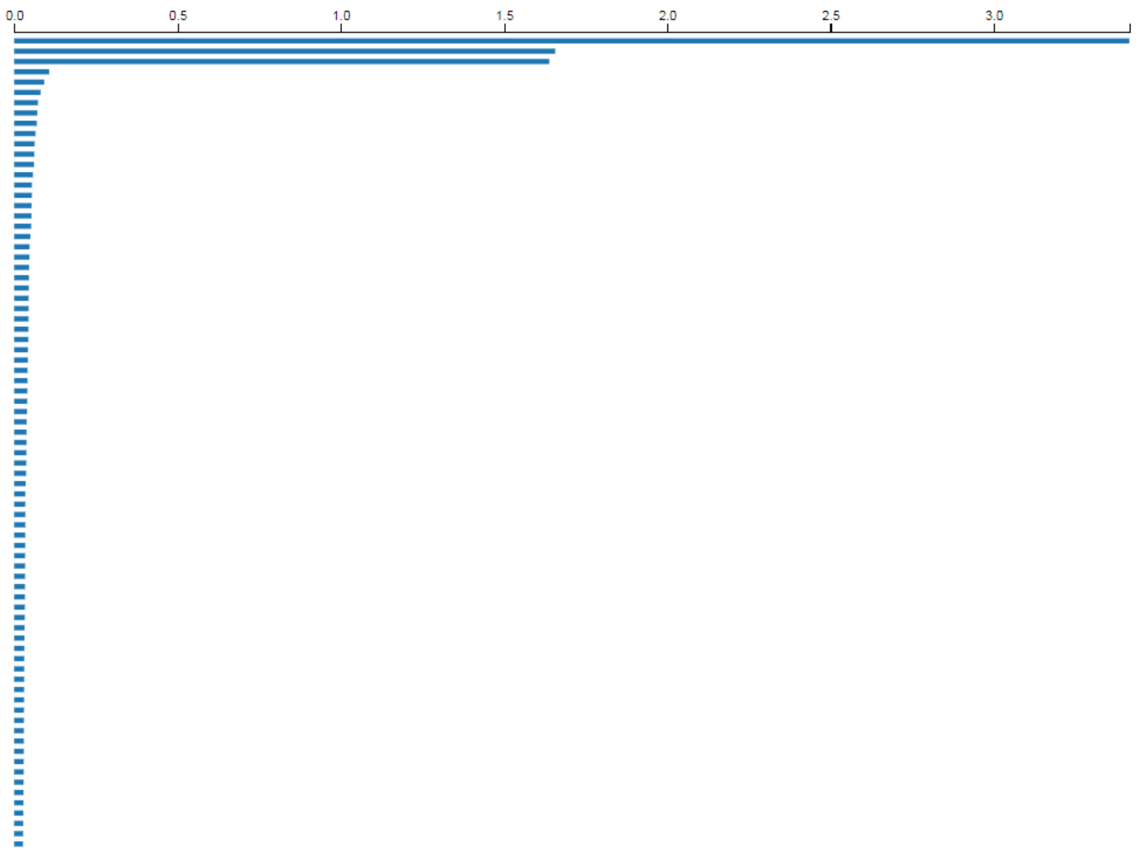
barcode is very close to the old one.

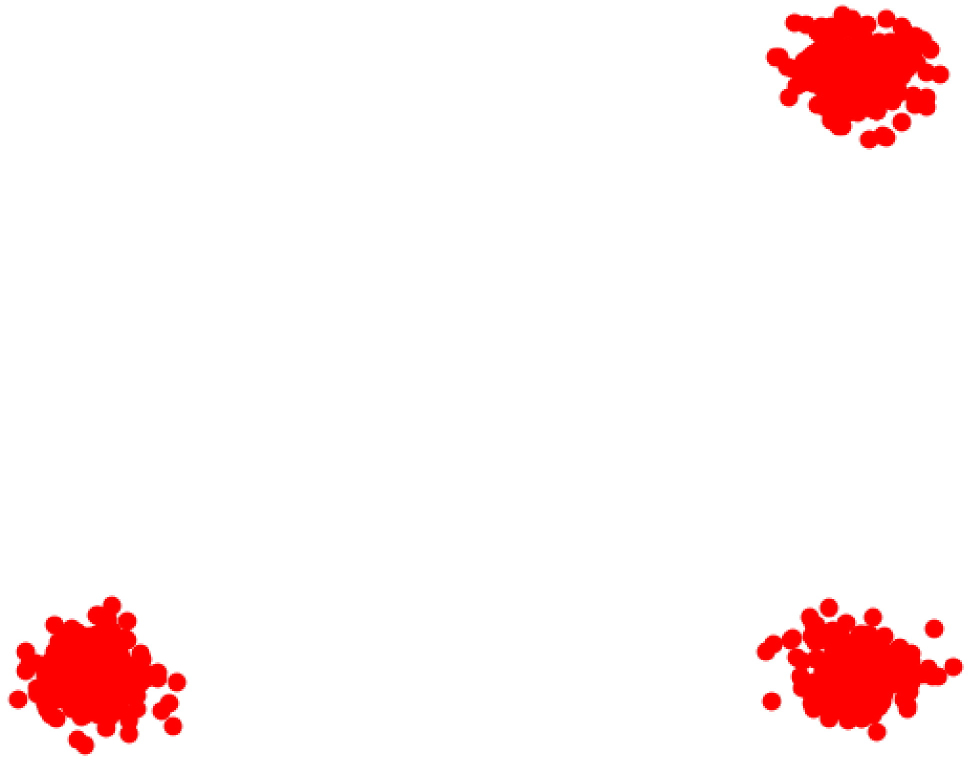




Examples

- 3 long bars, everything else represent noise

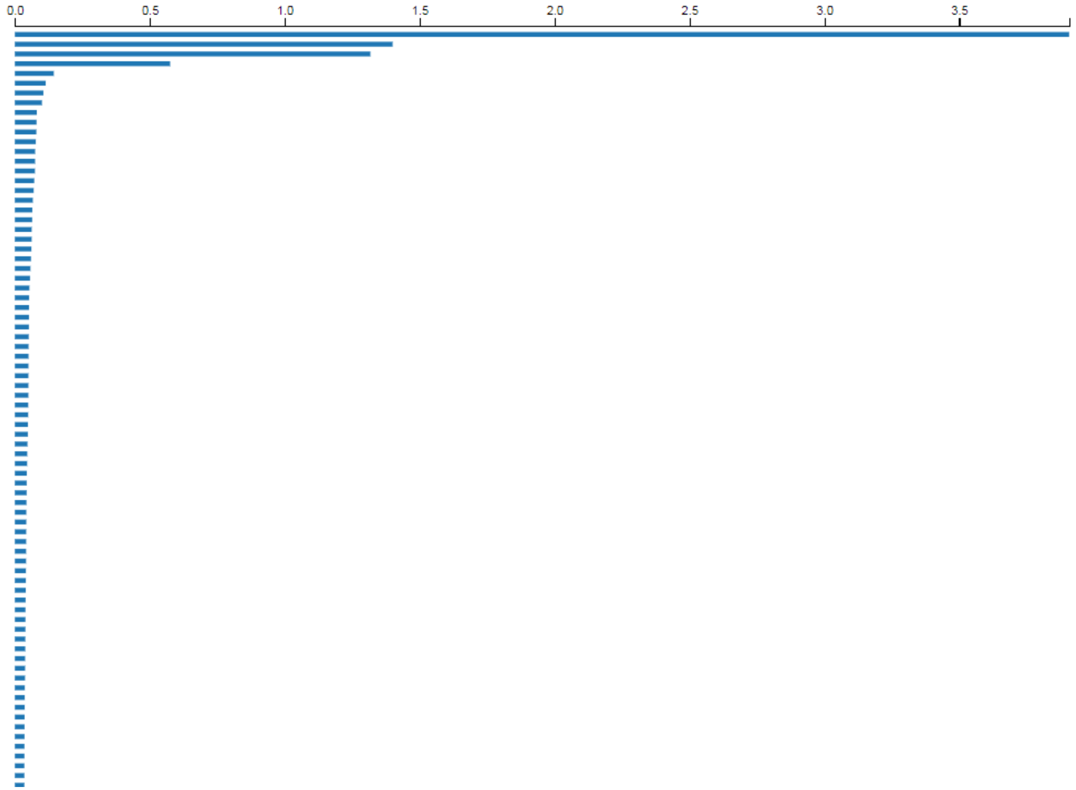


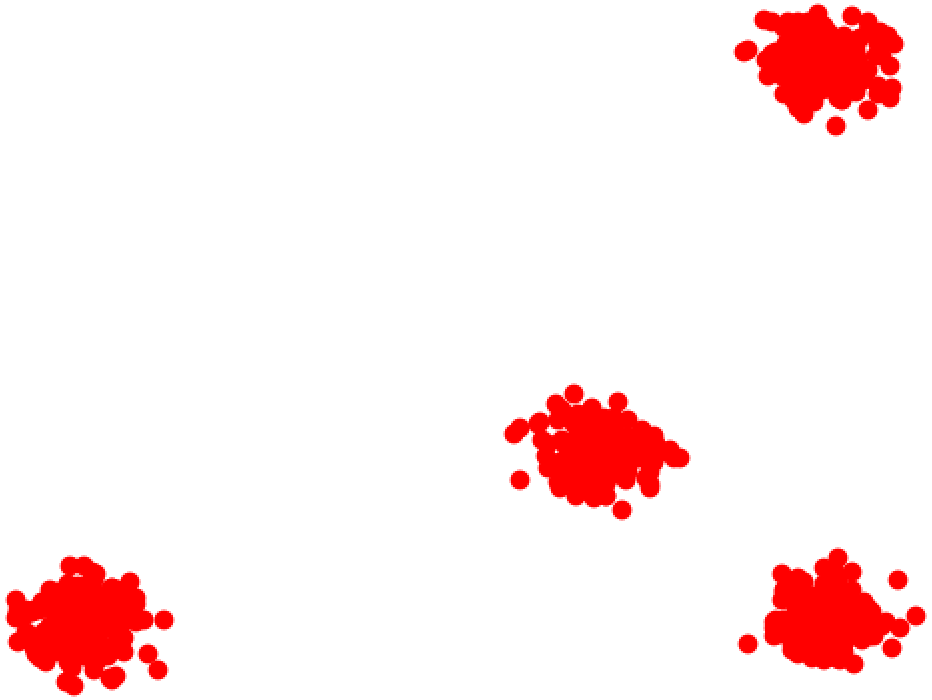


Computed using [ripser]

Examples

- 4 long bars, everything else represent noise





Computed using [ripser]

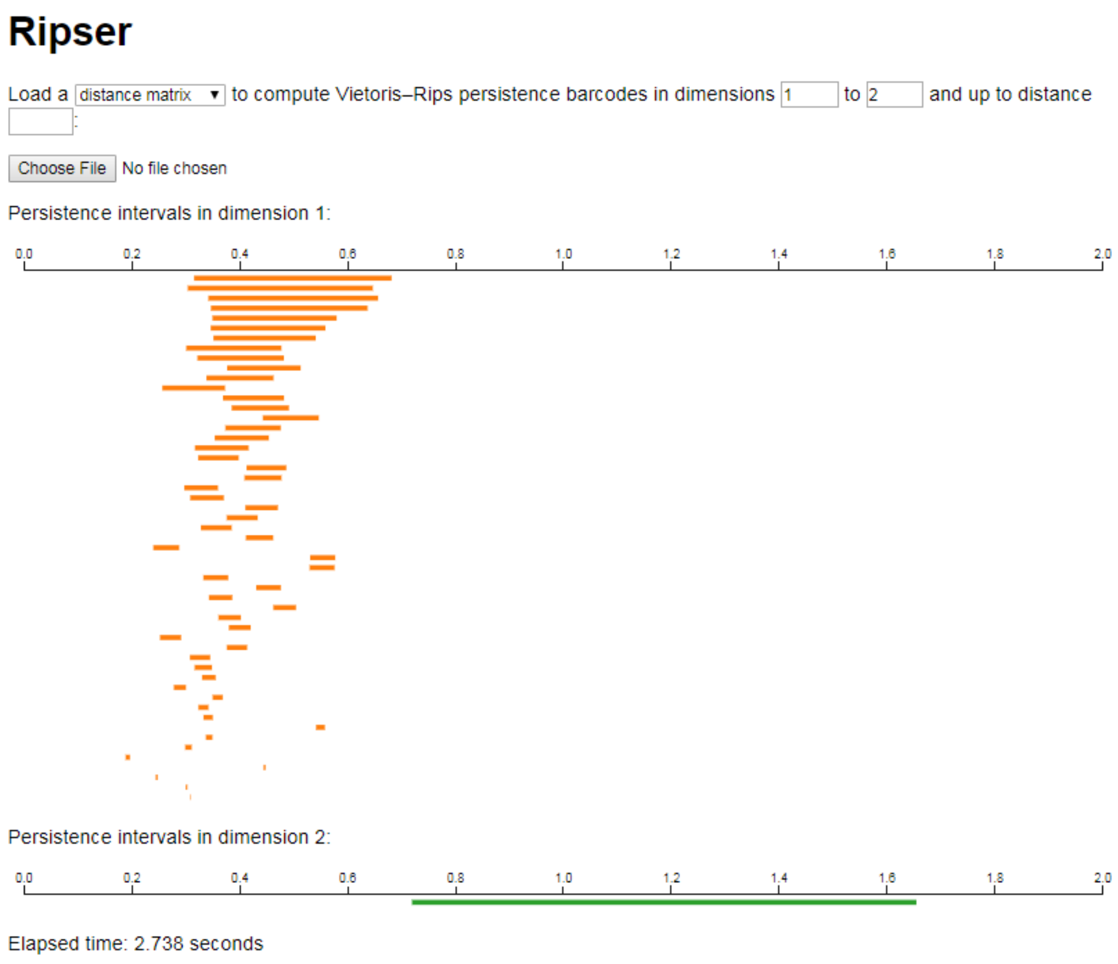
Examples

- 4 long bars, everything else represent noise

Computed using ripser

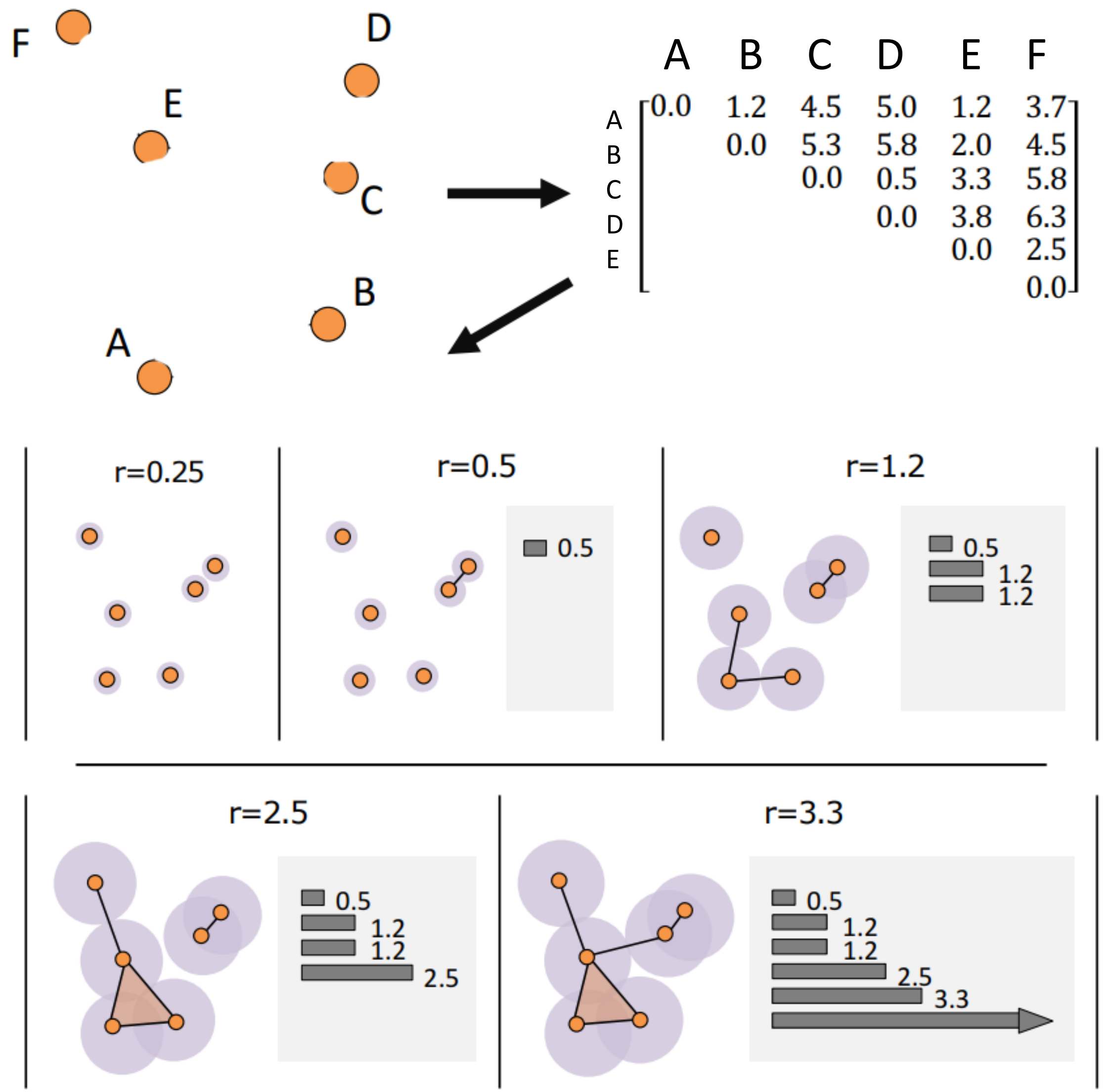
Ripser

http://live.ripser.org/



Barcode when a distance matrix is given

In this case, the points Coordinate are not given Explicitly. Only the distance between the points are given



The same computation can be carried out

Recall : Kruskal's Algorithm

Let $G = (V, E, w)$ be a connected weighted graph.

Informally, the algorithm can be given by the following three steps :

1. Set V_T $to\ be\ V, Set\ E_T = \{\}$. Let $S = E$
2. While S is not empty and T is not a spanning tree
   1. Select an edge e from S with the minimum weight and delete e from S.
   2. If e connects two separate trees of T then add e to E_T

Algorithm for computing the 0-barcode

Data: A distance matrix M

Result:
1-Create the complete graph G associated with the matrix M
2-Initiate an empty UnionFind U.
3- for each node vi in G :

1. U.add(vi)
2. Create a bar Bi with birth = 0 and death = ∞

4-Sort the edges of G in increasing order
5-for each edge ei in G do:

1. If ei connects two different sets C1 and C2 then
   1. Join C1 and C2
   2. Set the death of B1 to w(ei)

The complete graph associated with a distance matrix M : complete graph with e(i,j)=M(i,j).

Algorithm for computing the 0-barcode

Data: A distance matrix M

Result:
1-Create the complete graph G associated with the matrix M
2-Initiate an empty UnionFind U.
3- for each node vi in G :

1. U.add(vi)
2. Create a bar Bi with birth = 0 and death = ∞

4-Sort the edges of G in increasing order
5-for each edge ei in G do:

1. If ei connects two different sets C1 and C2 then
   1. Join C1 and C2
   2. Set the death of B1 to w(ei)

The complete graph associated with a distance matrix M : complete graph with e(i,j)=M(i,j).

This is essentially Kruskal's algorithm

Algorithm for computing the 0-barcode with a given max value

Data: A distance matrix M, maximal value **ε**

Result:
1-Create the **ε**-neighborhood graph of M
2-Initiate an empty UnionFind U.
3- for each node vi in G :

1. U.add(vi)
2. Create a bar Bi with birth = 0 and death = ∞

4-Sort the edges of G in increasing order
5-for each edge ei in G do:

1. If ei connects two different sets C1 and C2 then
    1. Join C1 and C2
    2. Set the death of B1 to w(ei)

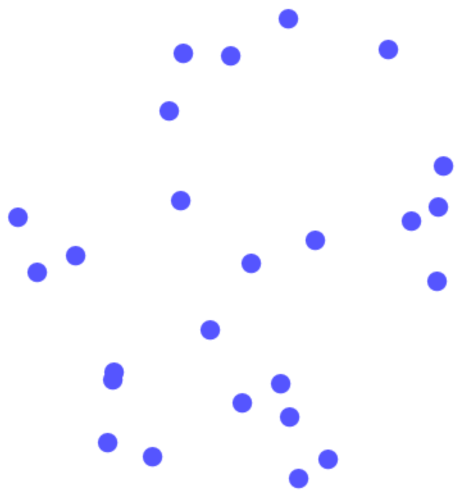
This is essentially Kruskal's algorithm

The relationship between 0-persistent homology and single linkage clustering

Recall: Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \ldots, p_n\}$ in R^d with a distance function d defined one them.

Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.

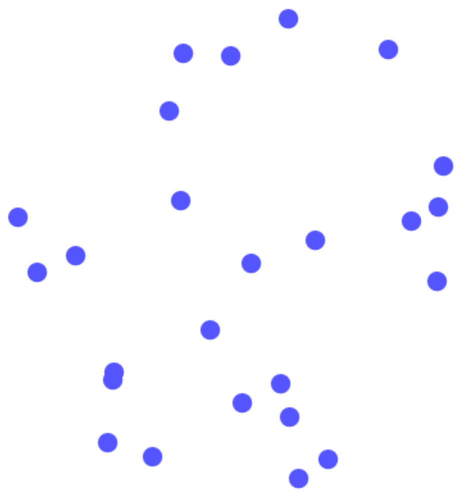


Every point is a connected component

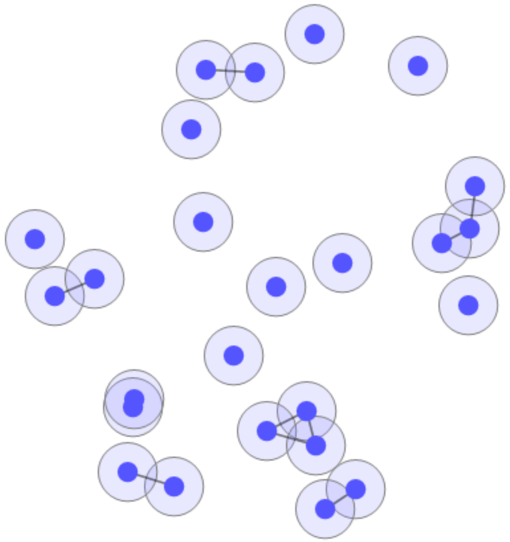
Recall: Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \dots, p_n\}$ in R^d with a distance function d defined one them.

Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.



Every point is a connected component

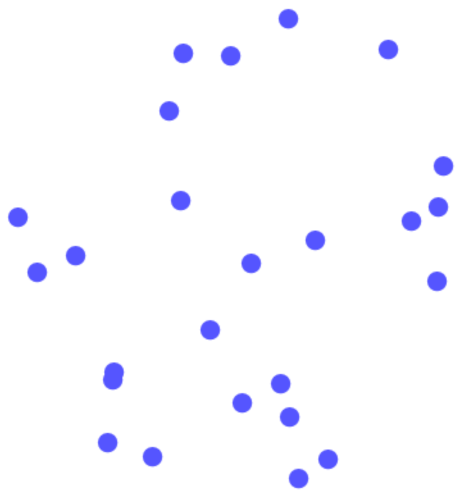


When ε is a little larger some clusters start to get form

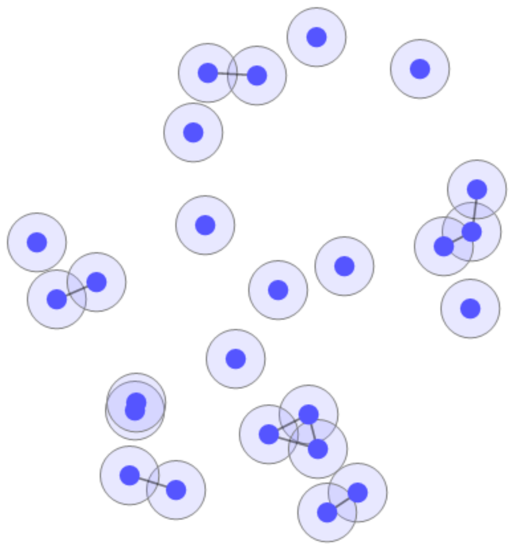
Recall: Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \dots, p_n\}$ in R^d with a distance function d defined one them.

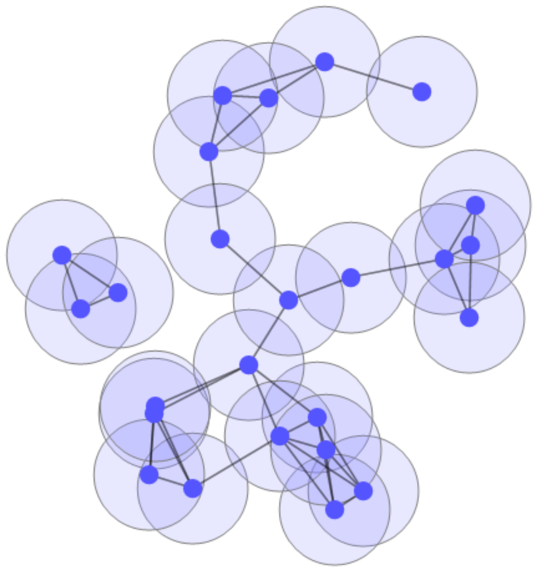
Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.



Every point is a connected component



When ε is a little larger some clusters start to get form

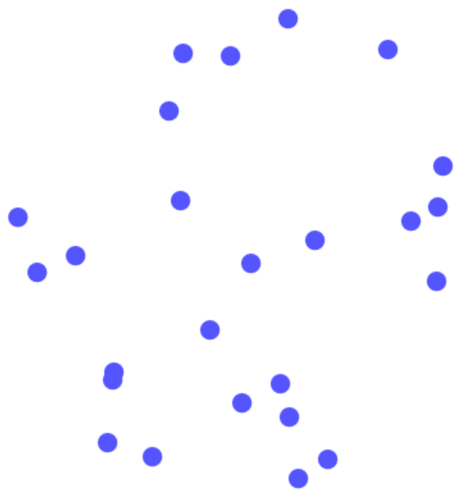


When ε is even larger we have fewer clusters

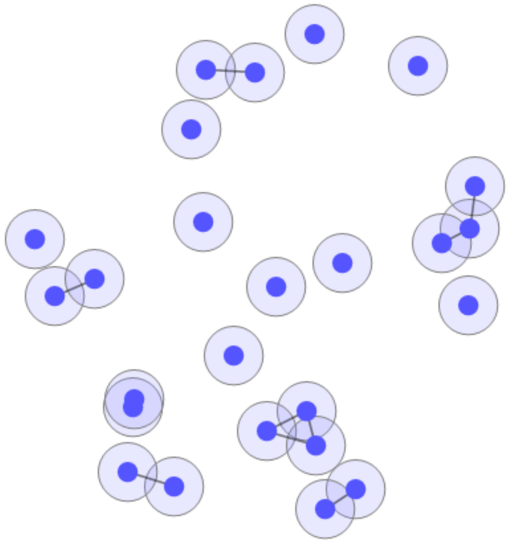
Recall: Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \dots, p_n\}$ in R^d with a distance function d defined one them.

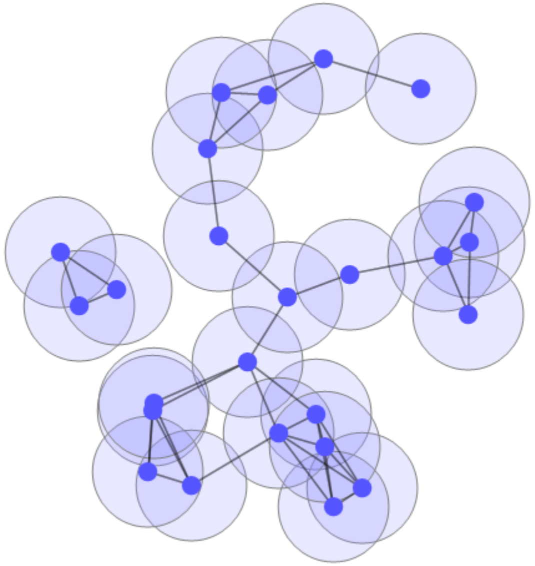
Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.



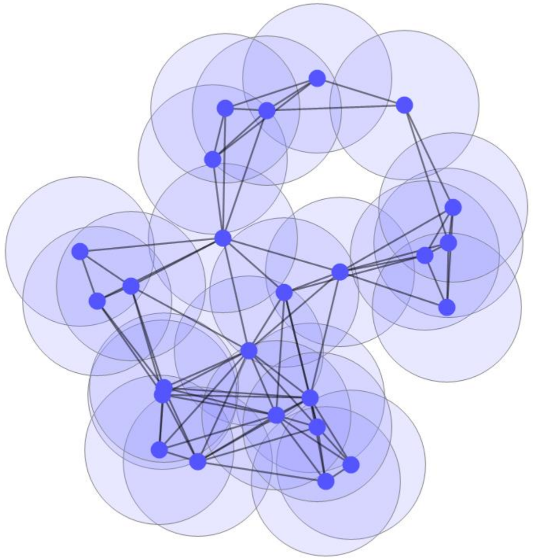
Every point is a connected component



When ε is a little larger some clusters start to get form

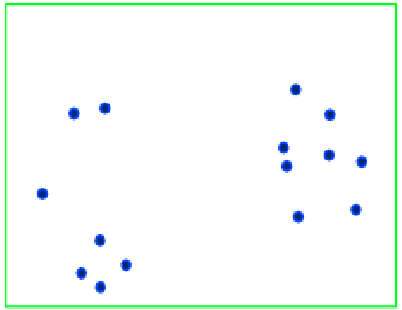


When ε is even larger we have fewer clusters

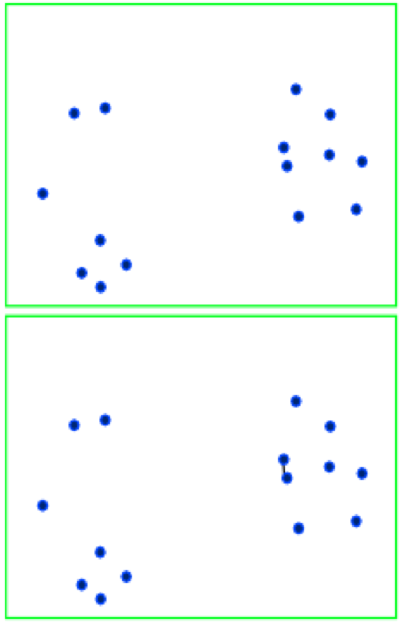


When ε is large enough all points become a part of a single cluster

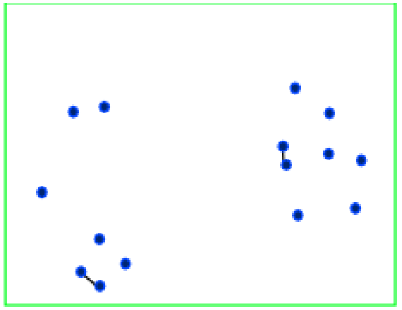
Single Linkage Hierarchical Clustering and the and Kruskal’s algorithm

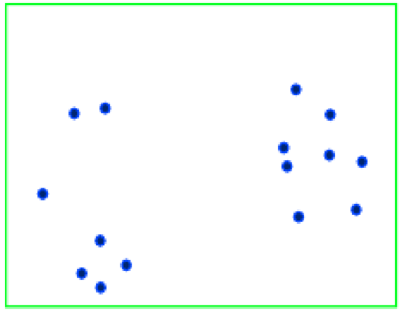


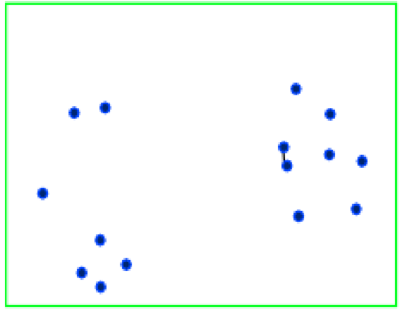
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



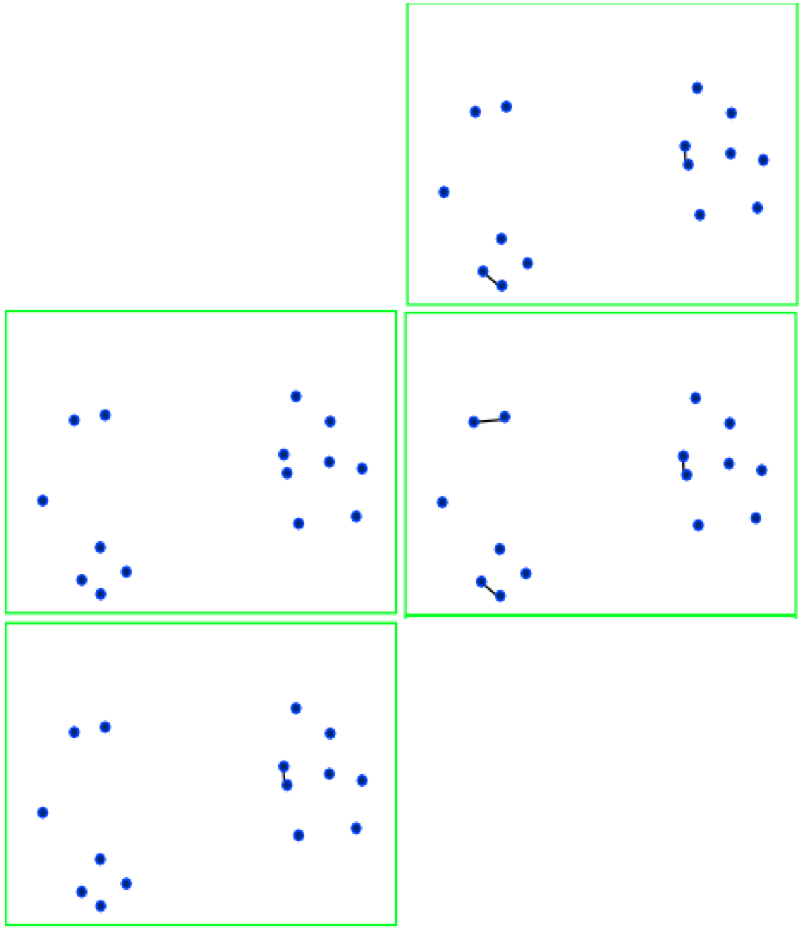
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



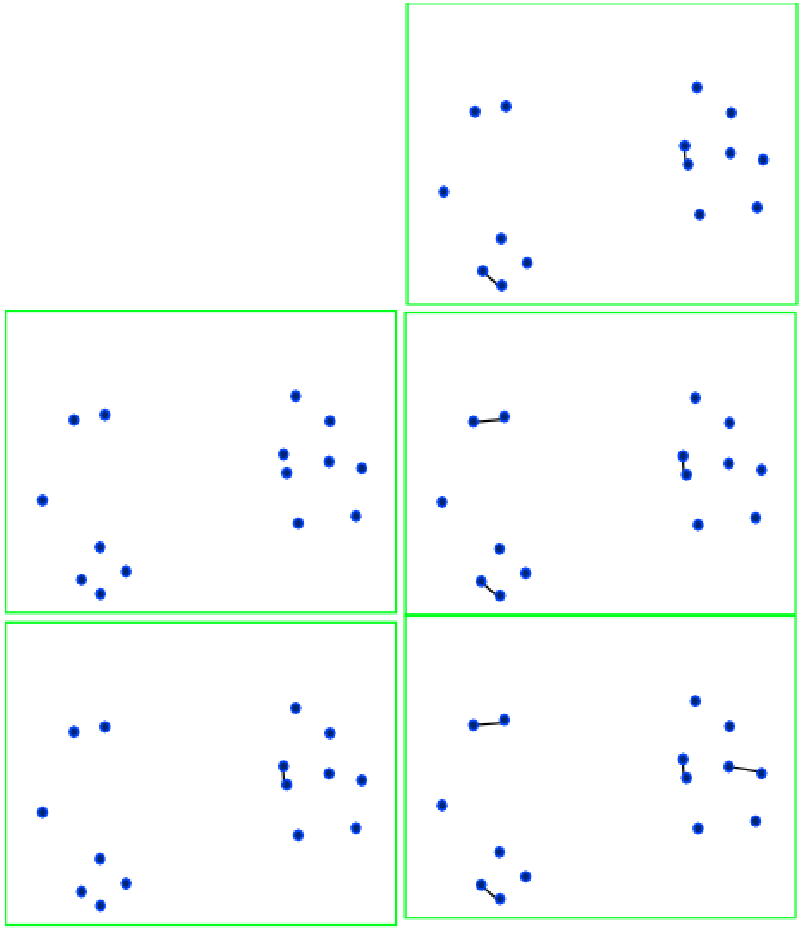




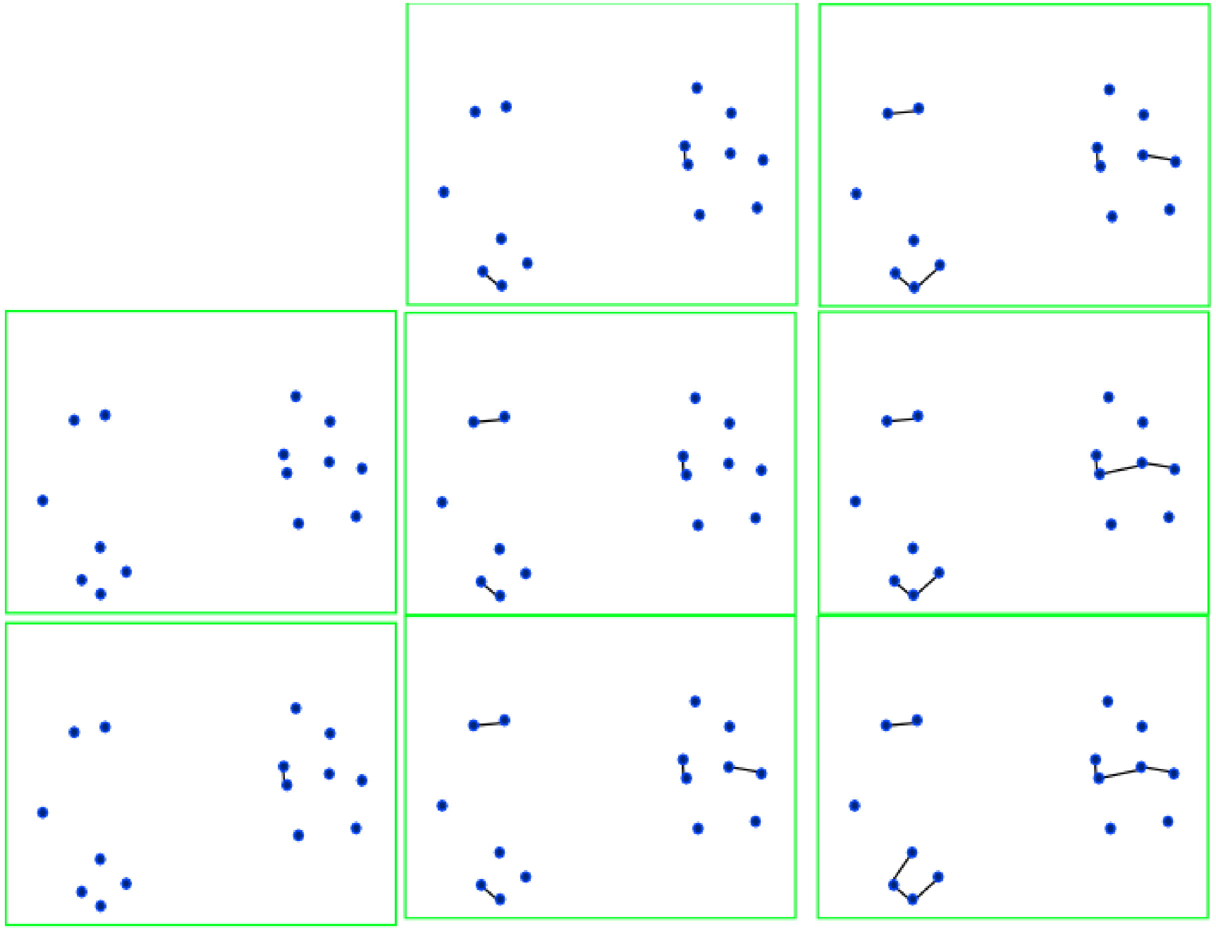
Single Linkage Hierarchical Clustering and the and Kruskal’s algorithm



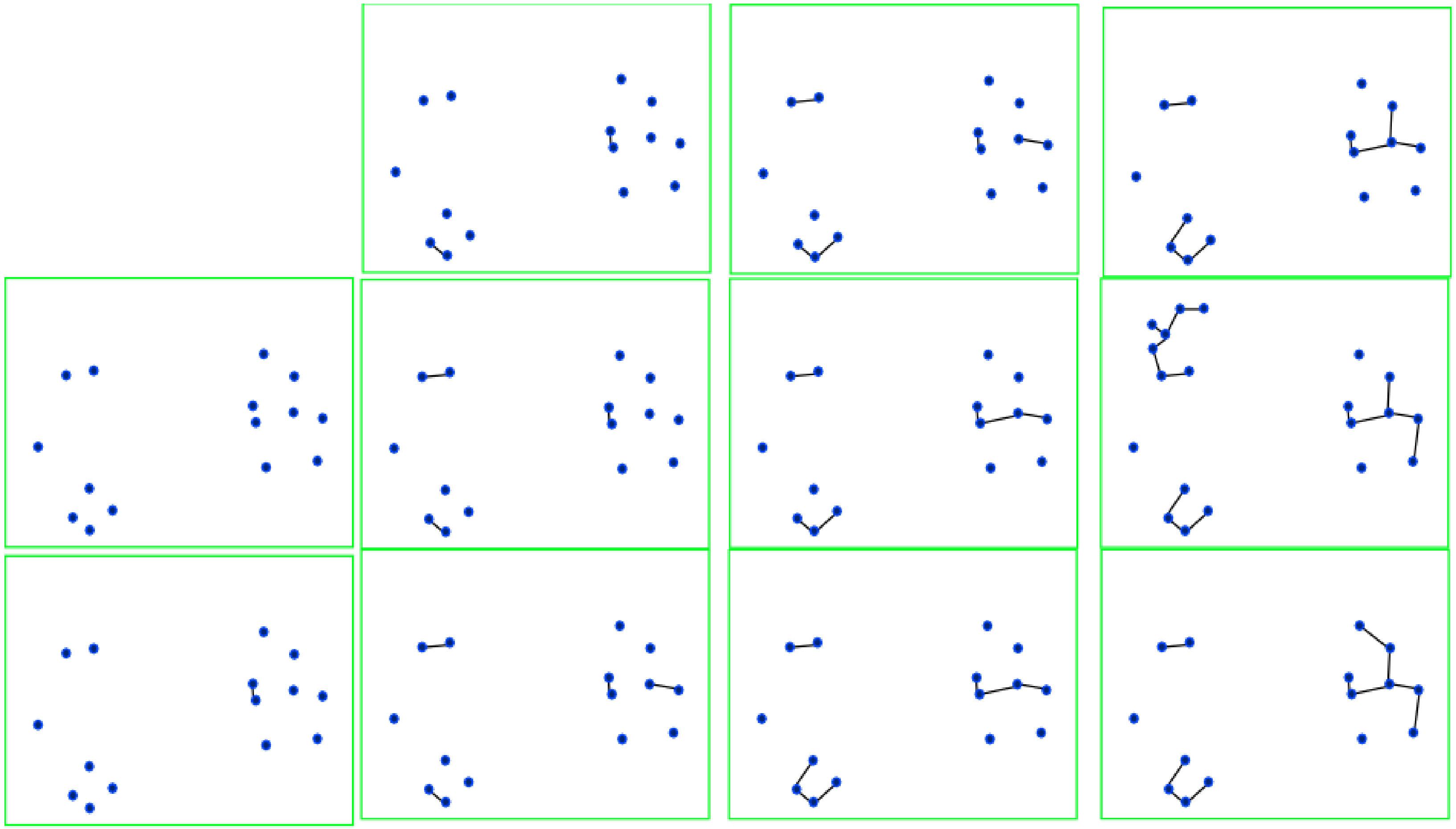
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



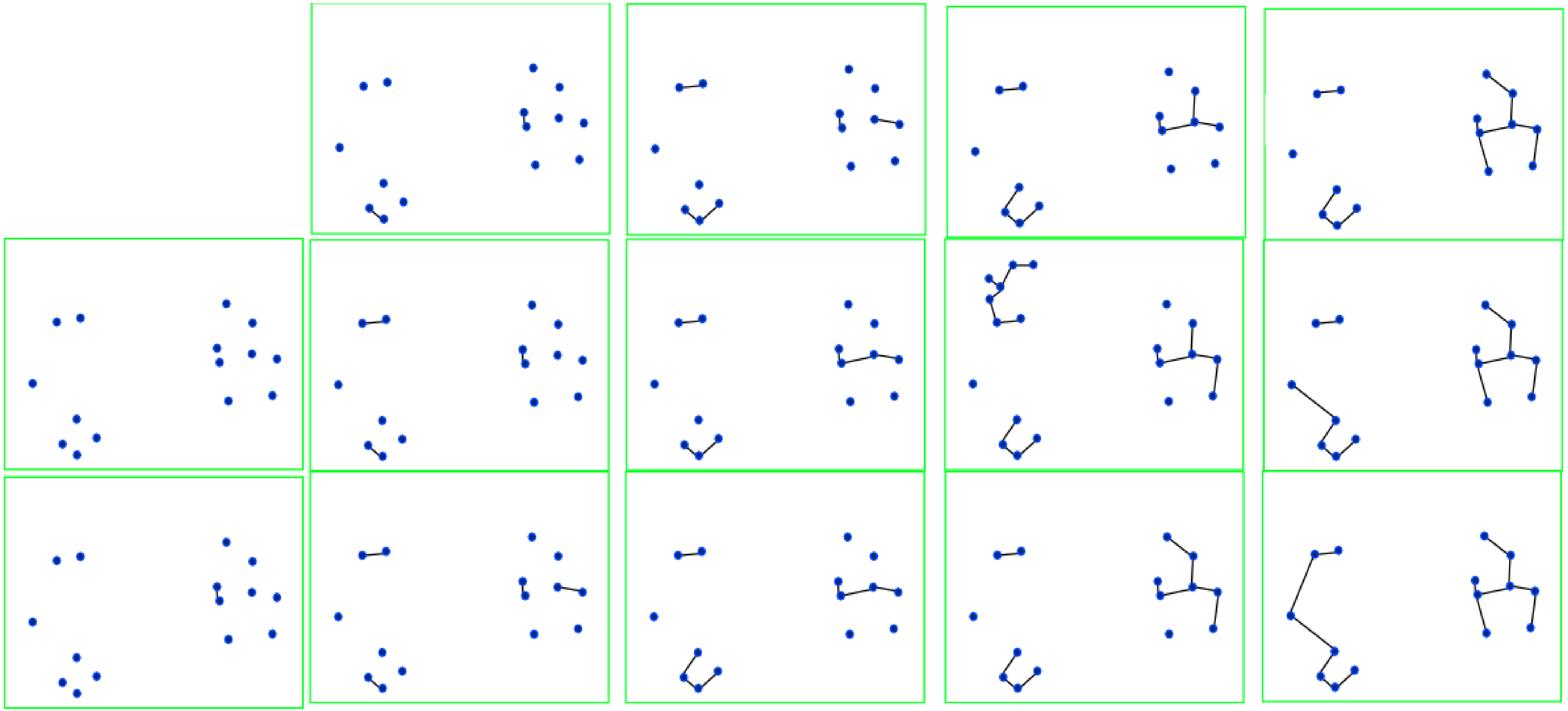
Single Linkage Hierarchical Clustering and the and Kruskal’s algorithm



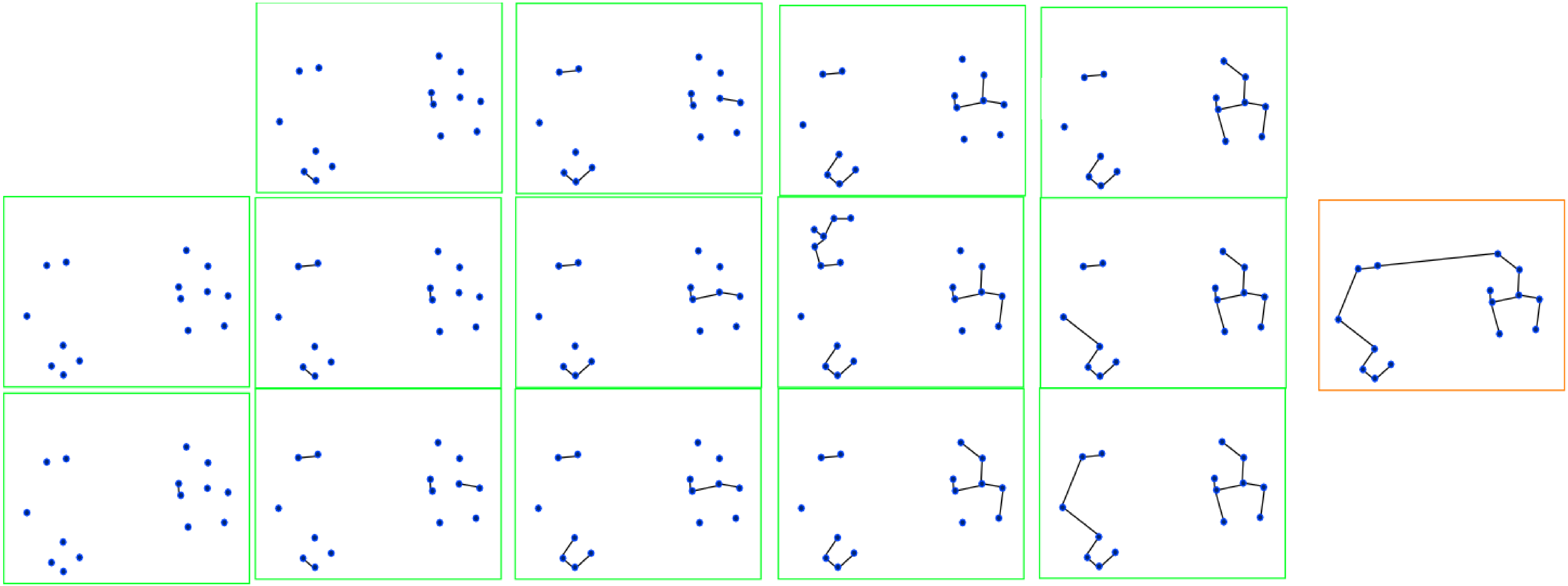
Single Linkage Hierarchical Clustering and the and Kruskal’s algorithm



Single Linkage Hierarchical Clustering and the and Kruskal’s algorithm



Single Linkage Hierarchical Clustering and the and Kruskal's algorithm

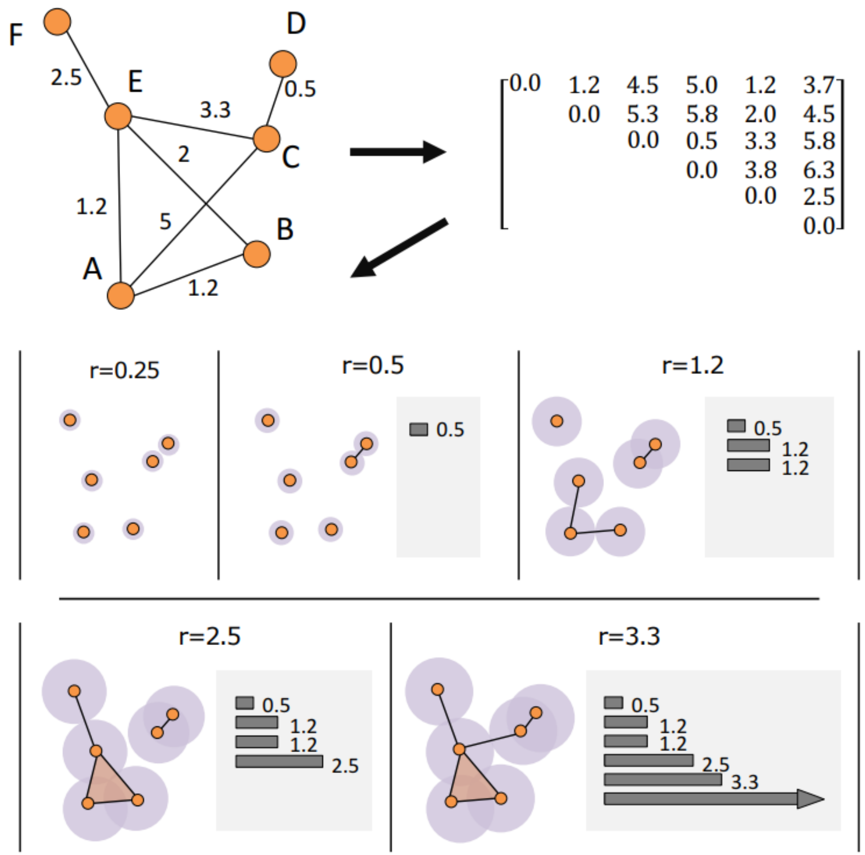


Relationship between 0-persistent homology and single linkage clustering

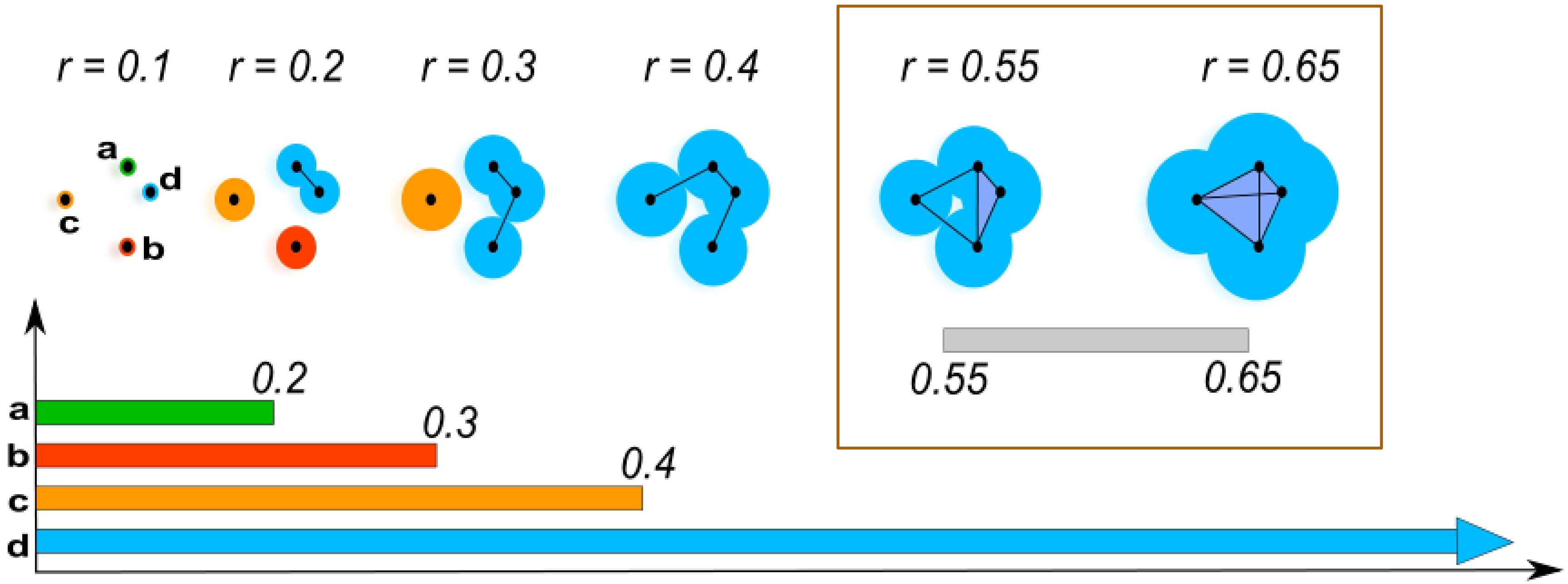
Essentially dendrogram of a data set in the single linkage clustering at a specific distance **ε** and the 0-barcode of a data set at a certain max distance encode the exact same information (just represented differently).

0-barcode of a weighted graph

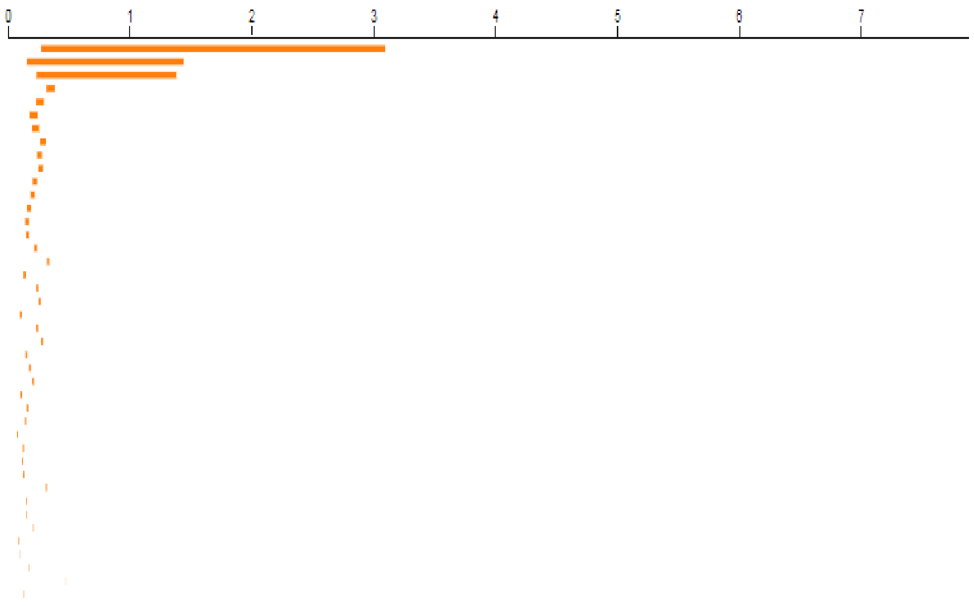
Weighted graph -> distance matrix using Dijekstra algorithm -> 0-barcode

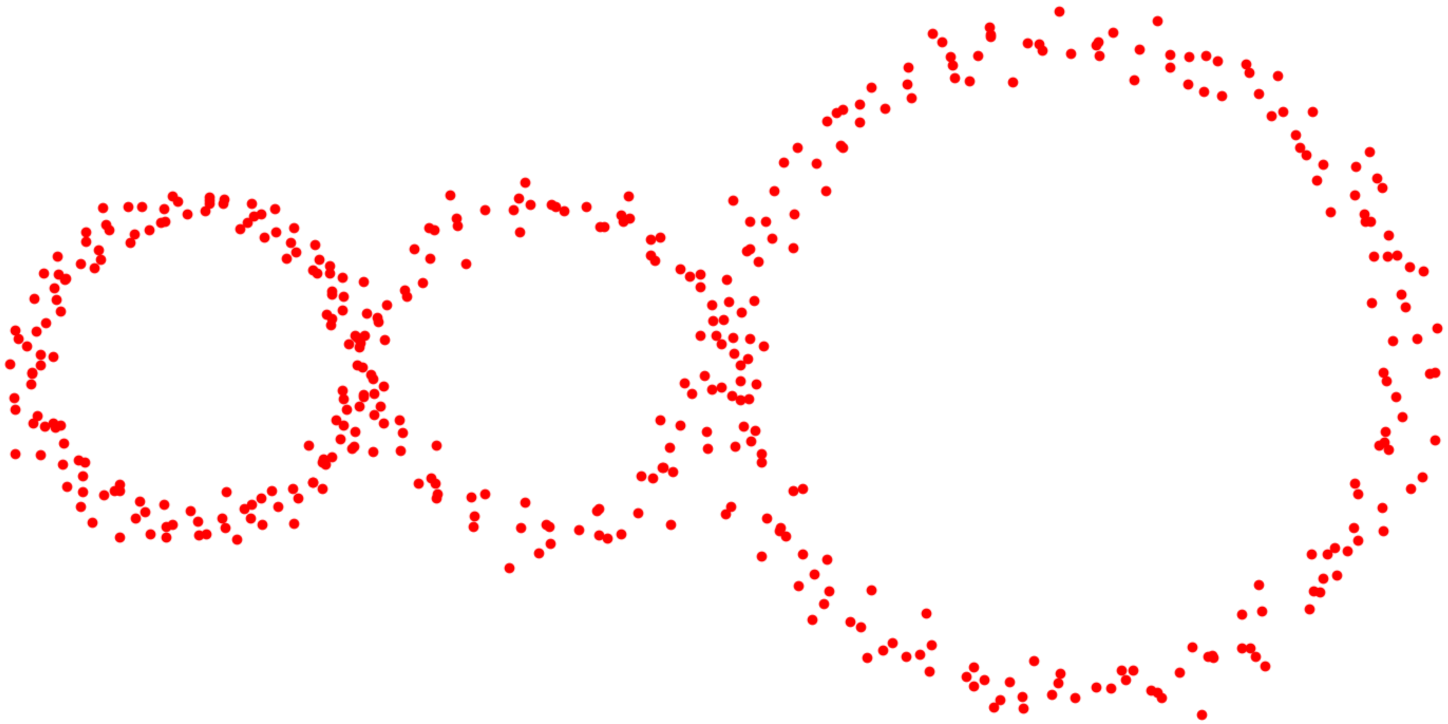


Higher dimensional barcodes



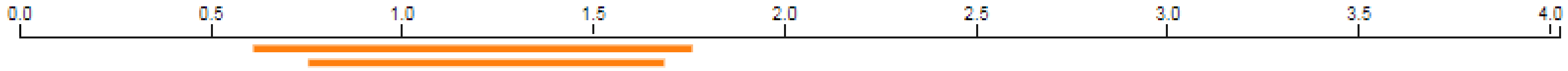
1-barcodes-examples

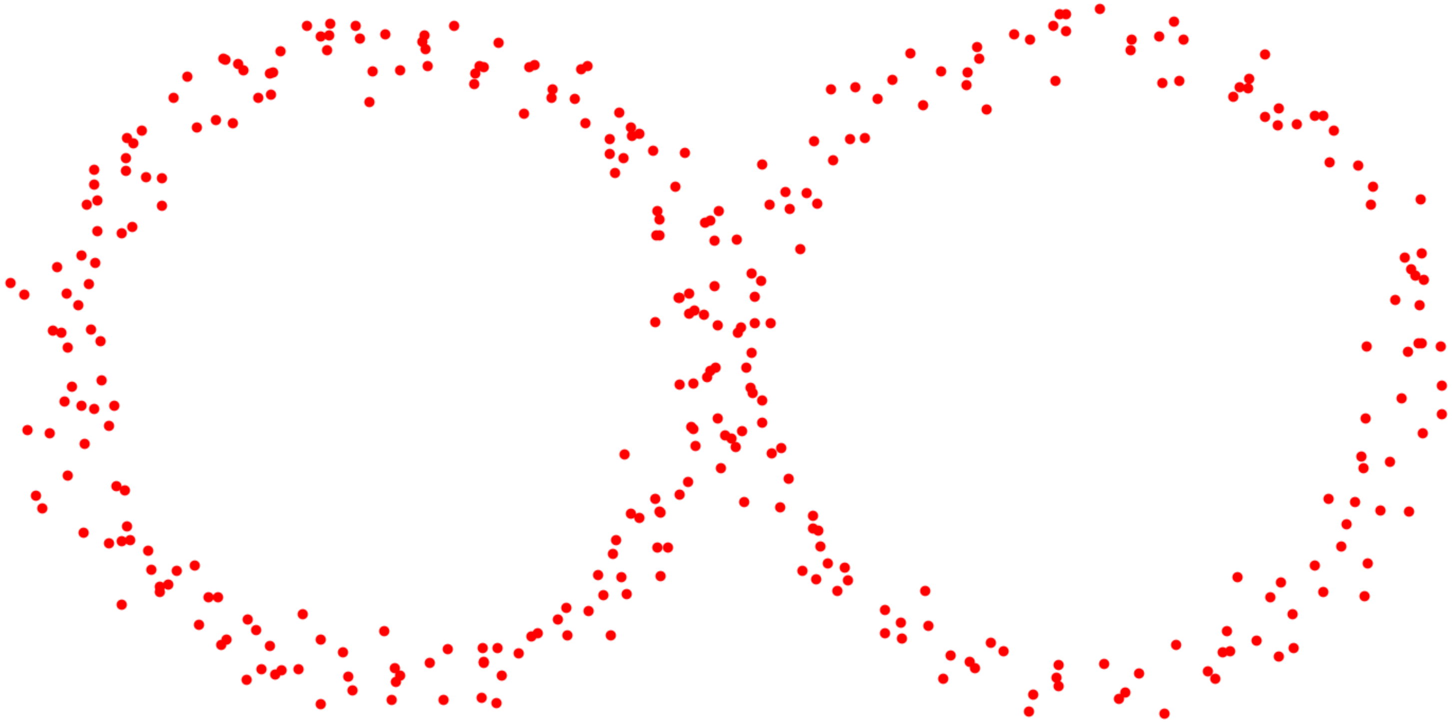




1-barcodes-examples

Persistence Diagram of a scalar function





1-barcodes-examples

Persistence Diagram of a scalar function