GESDA Introductory School
5-9 September 2022
IESC Cargèse, Corsica

Topological Clustering with Statistical Guarantees

Mathieu Carrière
INRIA Sophia-Antipolis
mathieu.carriere@inria.fr

I. ToMATo algorithm

1. Introduction to hierarchical and mode-seeking clustering

2. ToMATo algorithm and guarantees

II. Mapper algorithm

1. Reeb spaces and Mappers

2. Confidence intervals

I. ToMATo algorithm

1. **Introduction to hierarchical and mode-seeking clustering**
2. **ToMATo algorithm and guarantees**

II. Mapper algorithm

1. **Reeb spaces and Mappers**
2. **Confidence intervals**

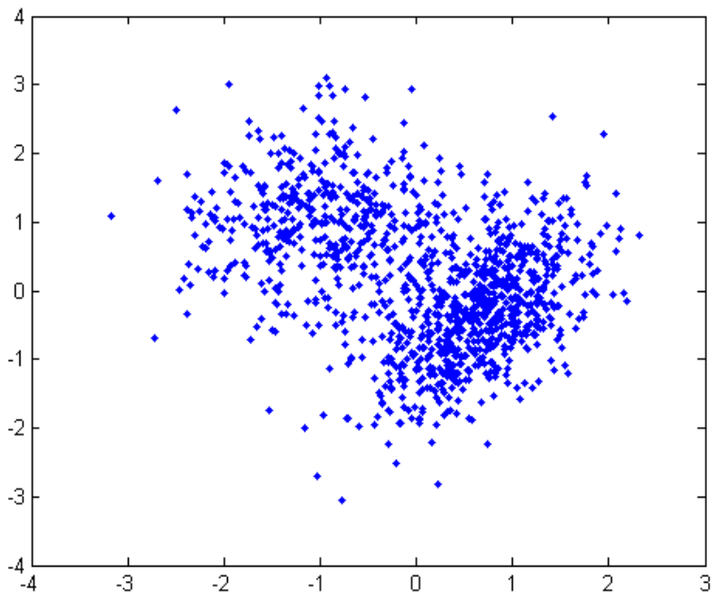
General clustering

Def: A partition of data into groups of similar data points. The data points in each group, or cluster, are similar to each other and dissimilar to the ones from other clusters.

General clustering

Def: A partition of data into groups of similar data points. The data points in each group, or cluster, are similar to each other and dissimilar to the ones from other clusters.

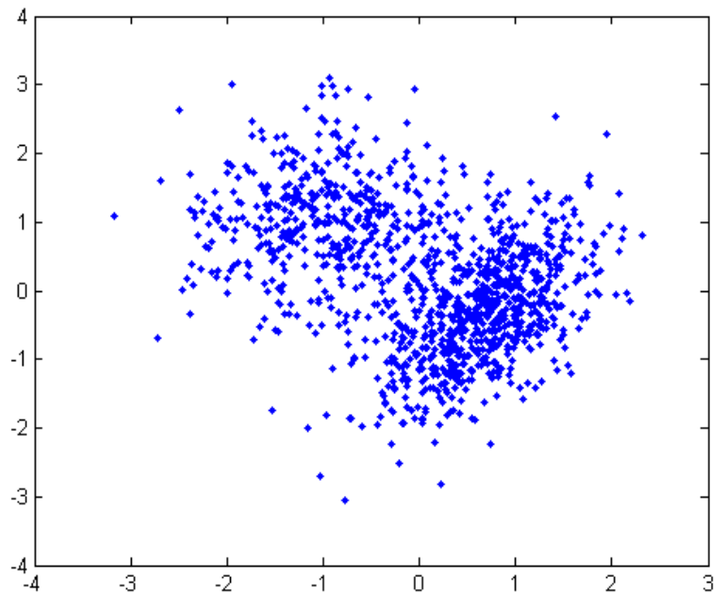
Input: a finite set of observations: point cloud embedded in an Euclidean space (i.e., with well-defined coordinates) or a more general metric space (pairwise distance or similarity) matrix.

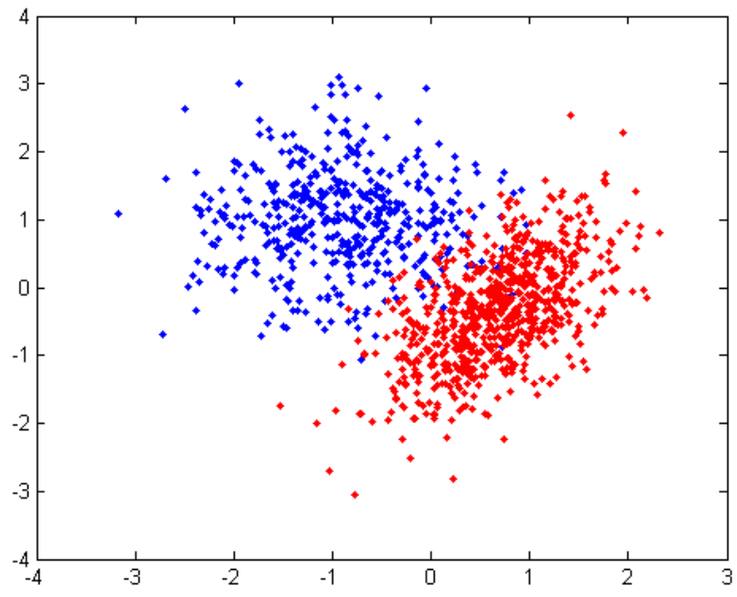


General clustering

Def: A partition of data into groups of similar data points. The data points in each group, or cluster, are similar to each other and dissimilar to the ones from other clusters.

Input: a finite set of observations: point cloud embedded in an Euclidean space (i.e., with well-defined coordinates) or a more general metric space (pairwise distance or similarity) matrix.





Goal: partition the data into a relevant family of clusters.

General clustering

Def: A partition of data into groups of similar data points. The data points in each group, or cluster, are similar to each other and dissimilar to the ones from other clusters.

Not a single or universal notion of cluster.

A variety of approaches:

- Variational (Bayes priors)
- Spectral (eigenvalues of Laplacian)
- Density-based (KDE, DTM)
- Hierarchical (dendrograms)
- etc...

General clustering

Def: A partition of data into groups of similar data points. The data points in each group, or cluster, are similar to each other and dissimilar to the ones from other clusters.

Not a single or universal notion of cluster.

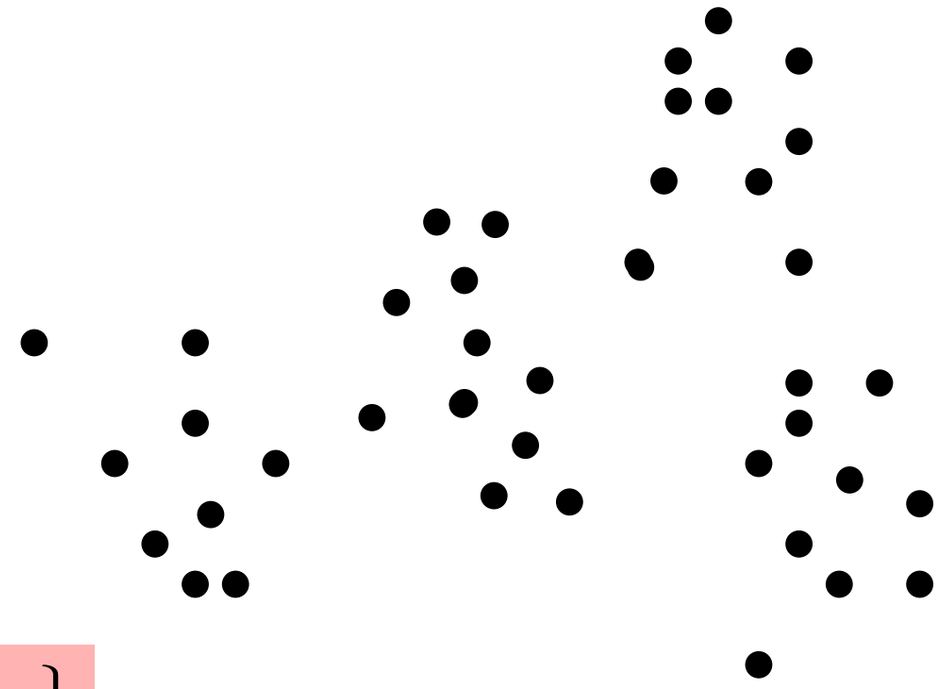
A variety of approaches:

- Variational (Bayes priors)
- Spectral (eigenvalues of Laplacian)
- Density-based (KDE, DTM)
- Hierarchical (dendrograms)
- etc...

We will see a few standard algorithms and how they can be improved with (0-dimensional) persistent homology.

The k-means algorithm

Input: A (large) set of n points X and an integer $k < n$.

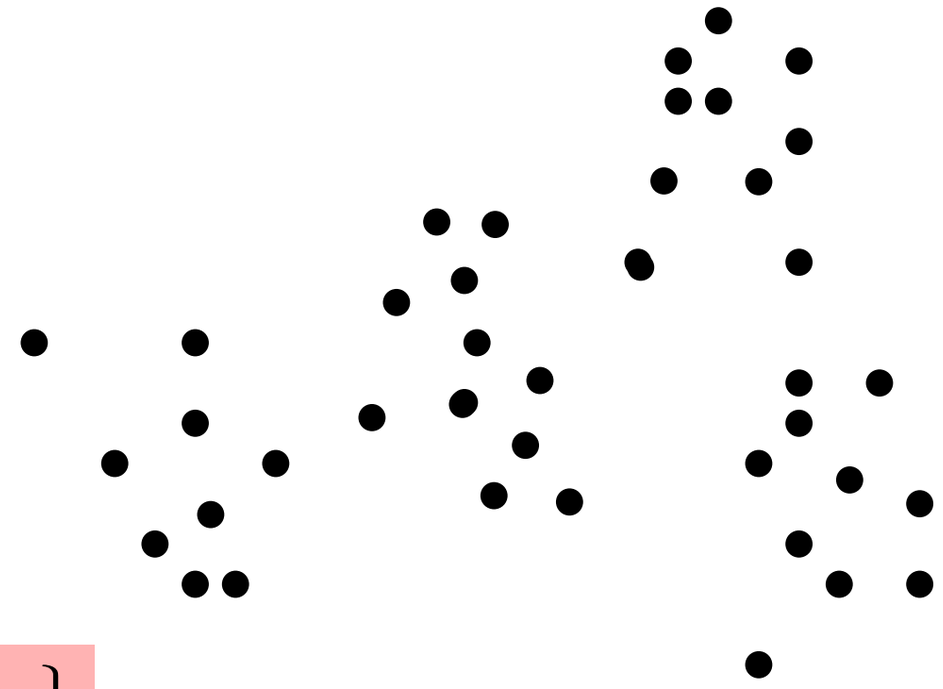


Goal: Find a set of k points $L = \{y_1, \ldots, y_k\}$ that minimizes

$$E = \sum_{i=1}^{n} d(x_i, L)^2$$

The k-means algorithm

Input: A (large) set of n points X and an integer $k < n$.



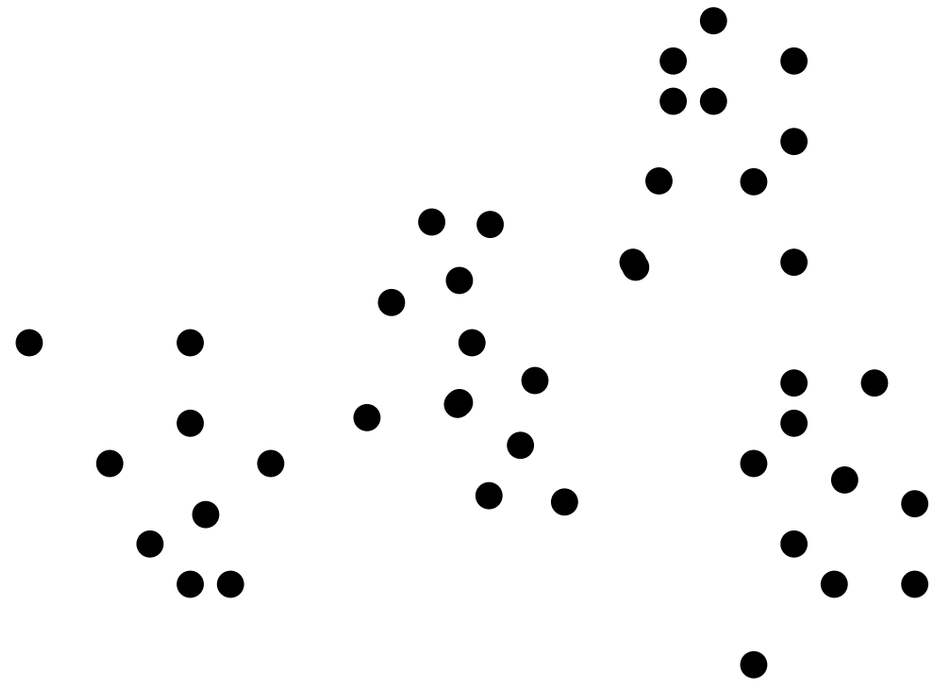
Goal: Find a set of k points $L = \{y_1, \ldots, y_k\}$ that minimizes

$$E = \sum_{i=1}^{n} d(x_i, L)^2$$

This is a NP hard problem!

Lloyd's algorithm: a very simple local search algorithm.

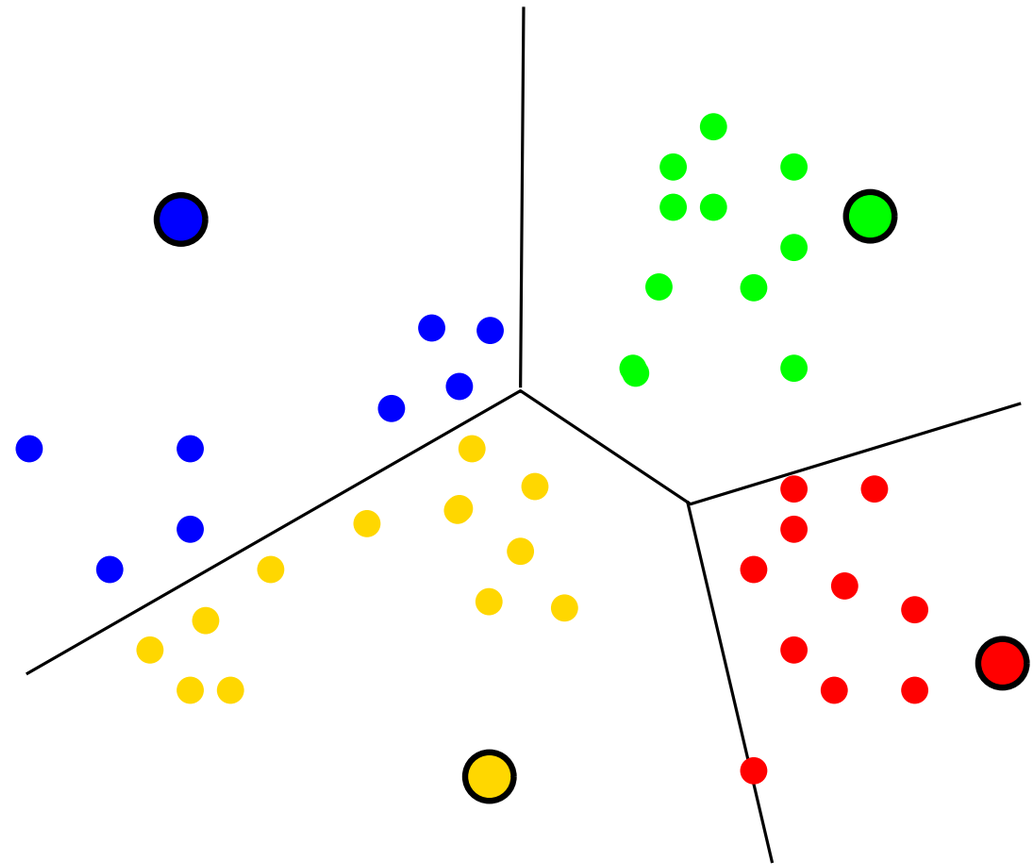
The k-means algorithm



Lloyd's algorithm

$L^1 \leftarrow \{y_1^1, \ldots, y_k^1\}$ (initial seeds)
$i \leftarrow 1$
`while` convergence not reached:
    `for` $j \in \{1, \ldots, k\}$:
        $S_j^i \leftarrow \{x \in X : d(x, y_j^i)$ achieves $d(x, L^i)\}$
    `for` $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$

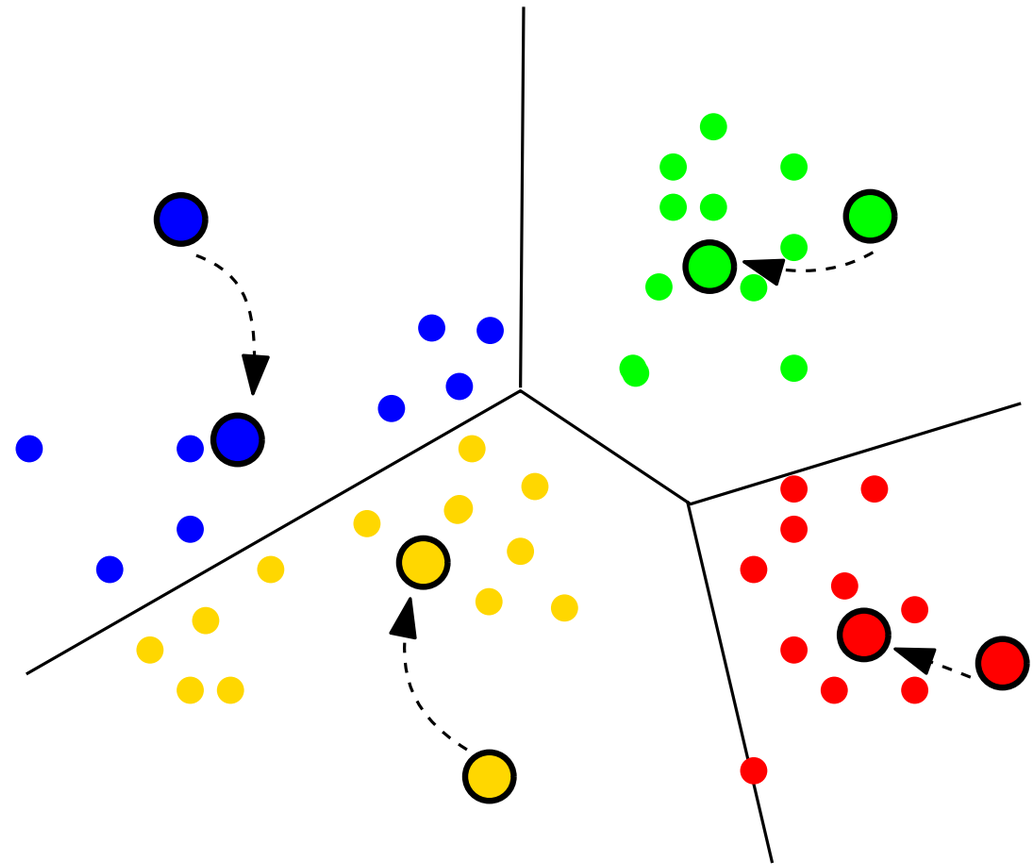
The k-means algorithm



Lloyd's algorithm

$L^1 \leftarrow \{y_1^1, \ldots, y_k^1\}$ (initial seeds)
$i \leftarrow 1$
`while` convergence not reached:
    `for` $j \in \{1, \ldots, k\}$:
        $S_j^i \leftarrow \{x \in X : d(x, y_j^i)$ achieves $d(x, L^i)\}$
    `for` $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$

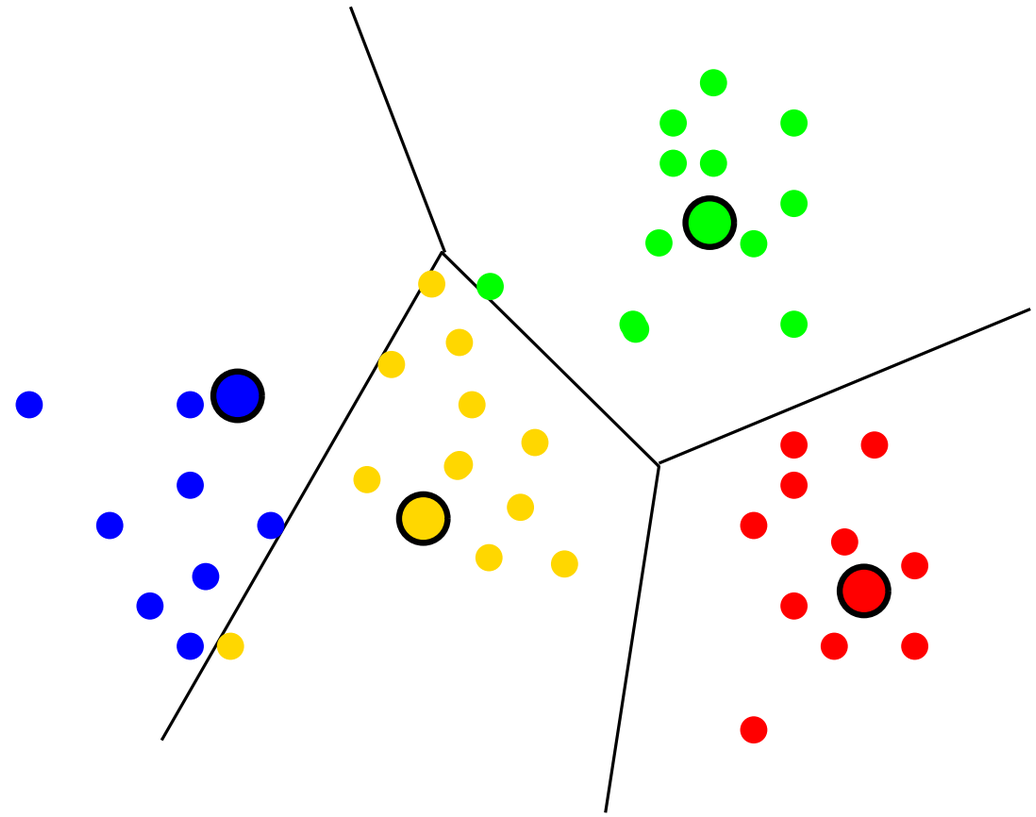
The k-means algorithm



Lloyd's algorithm

$L^1 \leftarrow \{y_1^1, \ldots, y_k^1\}$ (initial seeds)
$i \leftarrow 1$
`while` convergence not reached:
    `for` $j \in \{1, \ldots, k\}$:
        $S_j^i \leftarrow \{x \in X : d(x, y_j^i)$ achieves $d(x, L^i)\}$
    `for` $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$

The k-means algorithm



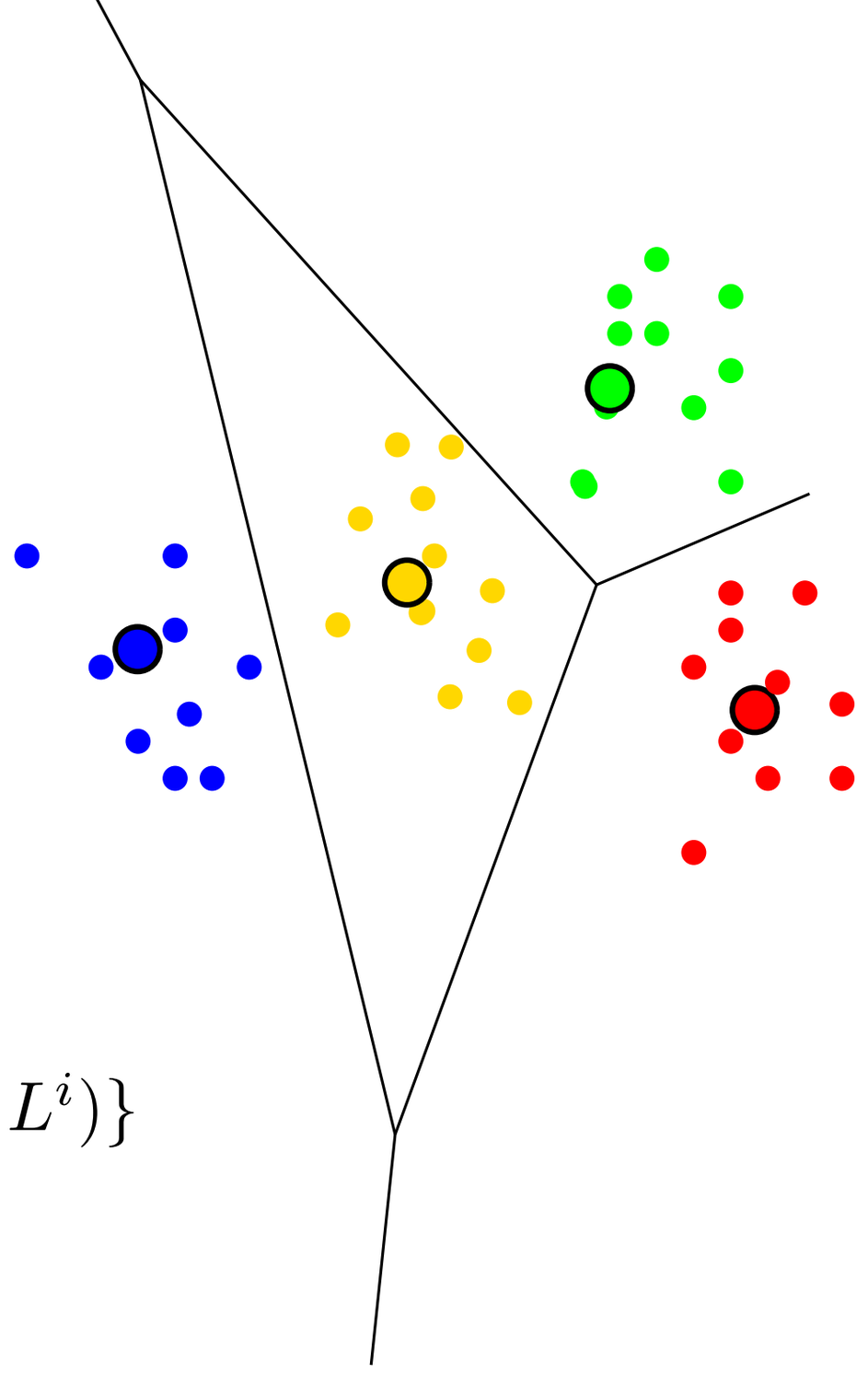
Lloyd's algorithm

$L^1 \leftarrow \{y_1^1, \ldots, y_k^1\}$ (initial seeds)
$i \leftarrow 1$
`while` convergence not reached:
    `for` $j \in \{1, \ldots, k\}$:
        $S_j^i \leftarrow \{x \in X : d(x, y_j^i)$ achieves $d(x, L^i)\}$
    `for` $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$

The k-means algorithm

Lloyd's algorithm

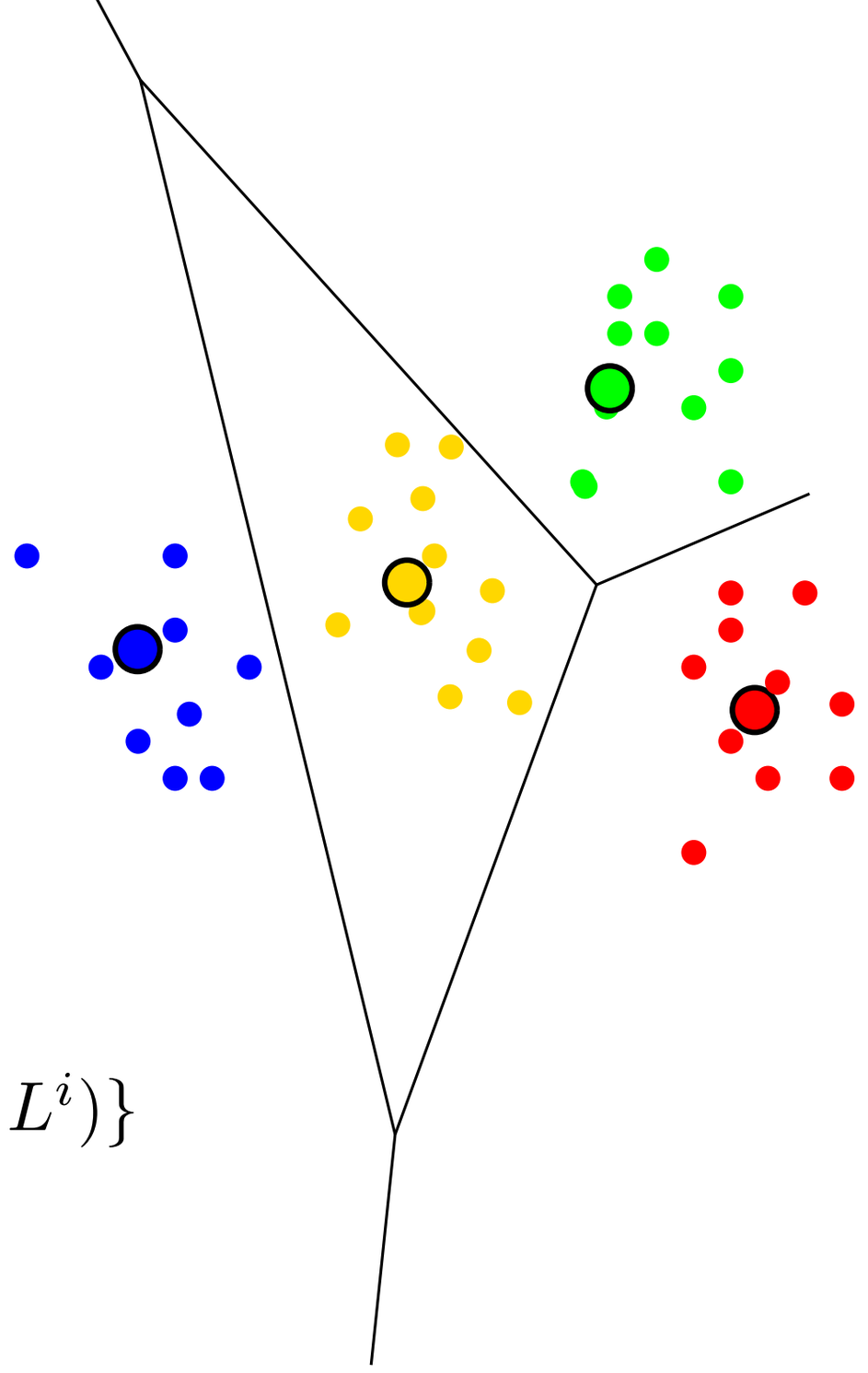
$L^1 \leftarrow \{y_1^1, \ldots, y_k^1\}$ (initial seeds)
$i \leftarrow 1$
`while` convergence not reached:
    `for` $j \in \{1, \ldots, k\}$:
        $S_j^i \leftarrow \{x \in X : d(x, y_j^i)$ achieves $d(x, L^i)\}$
    `for` $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$



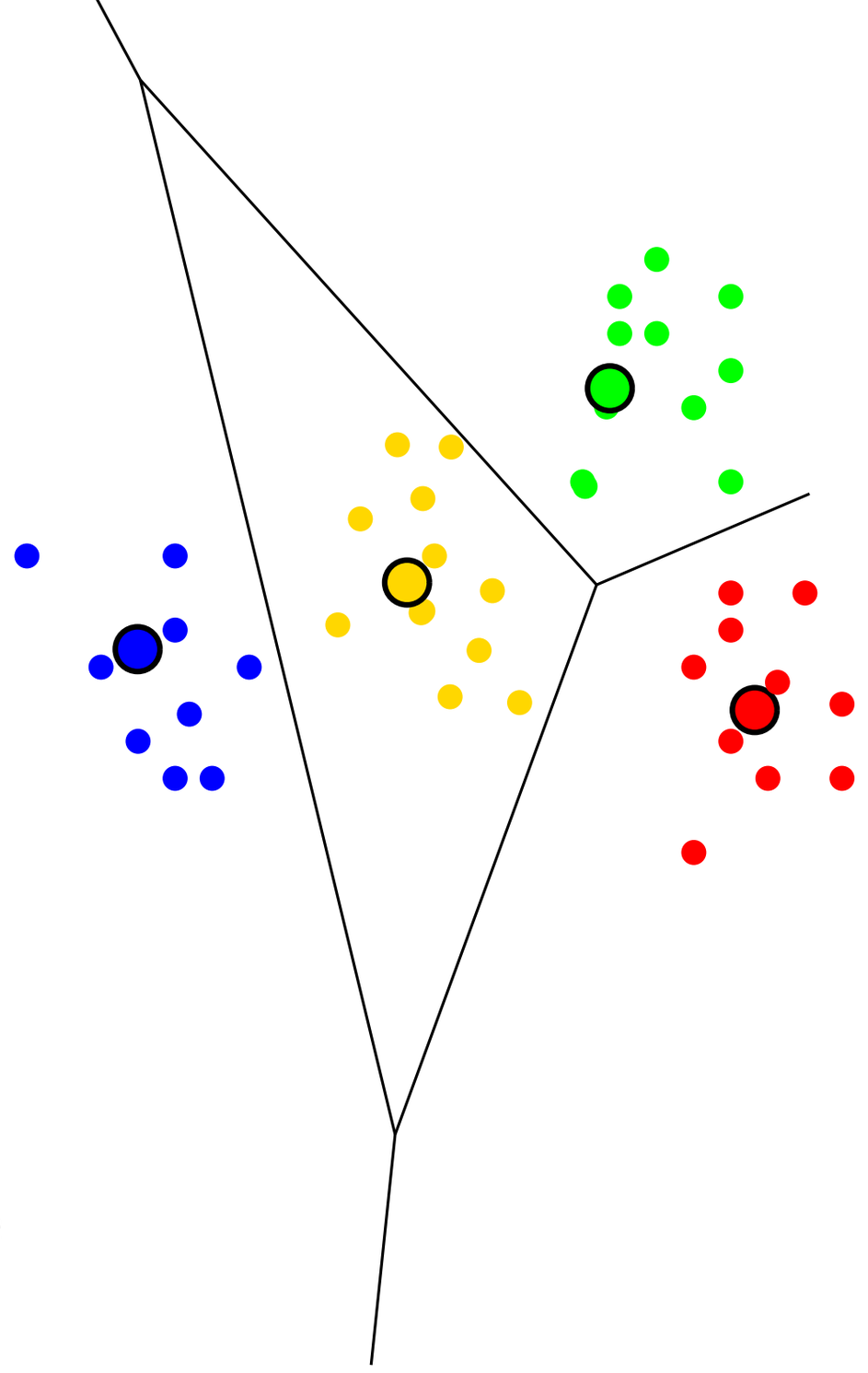
The k-means algorithm

Lloyd's algorithm

$L^1 \leftarrow \{y_1^1, \ldots, y_k^1\}$ (initial seeds)
$i \leftarrow 1$
`while` convergence not reached:
    `for` $j \in \{1, \ldots, k\}$:
        $S_j^i \leftarrow \{x \in X : d(x, y_j^i)$ achieves $d(x, L^i)\}$
    `for` $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$



The k-means algorithm



Warning:

- Minimum is not necessarily global!
- Speed of convergence not guaranteed.
- Lack of stability: output is very sensitive to initial seeds.

Hierarchical clustering algorithms

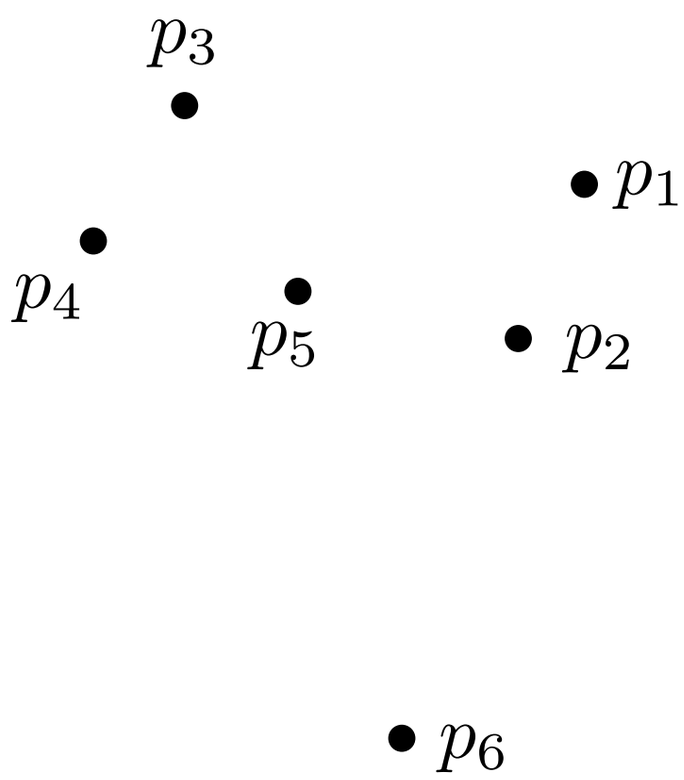
Goal: Build a hierarchy of clusters (nested family of partitions).

Hierarchical clustering algorithms

Goal: Build a hierarchy of clusters (nested family of partitions).

Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).

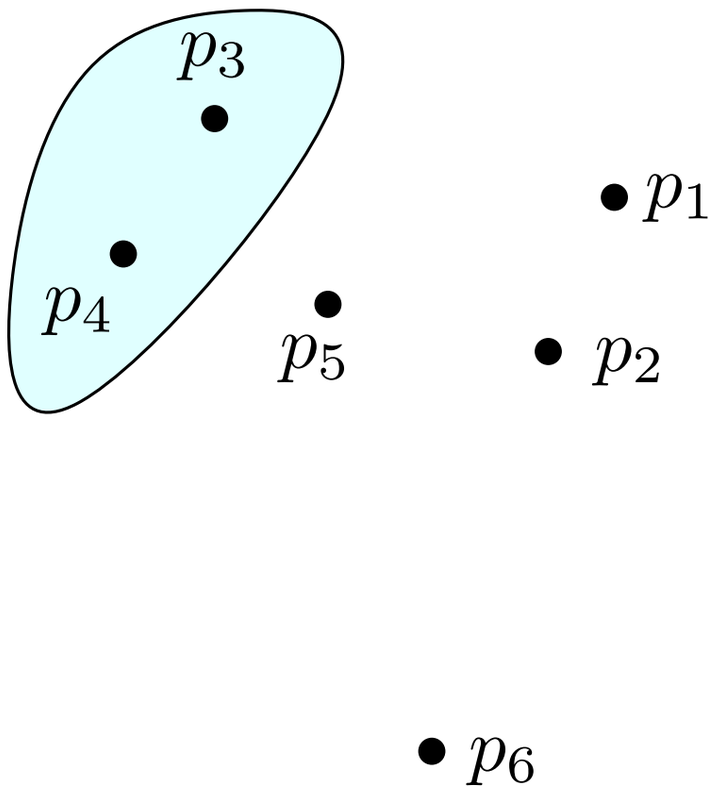


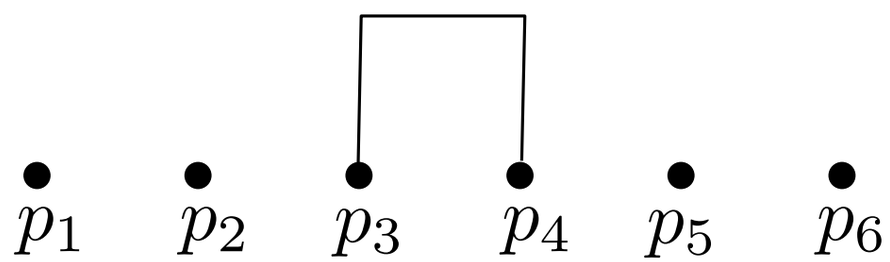
Hierarchical clustering algorithms

Goal: Build a hierarchy of clusters (nested family of partitions).

Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).



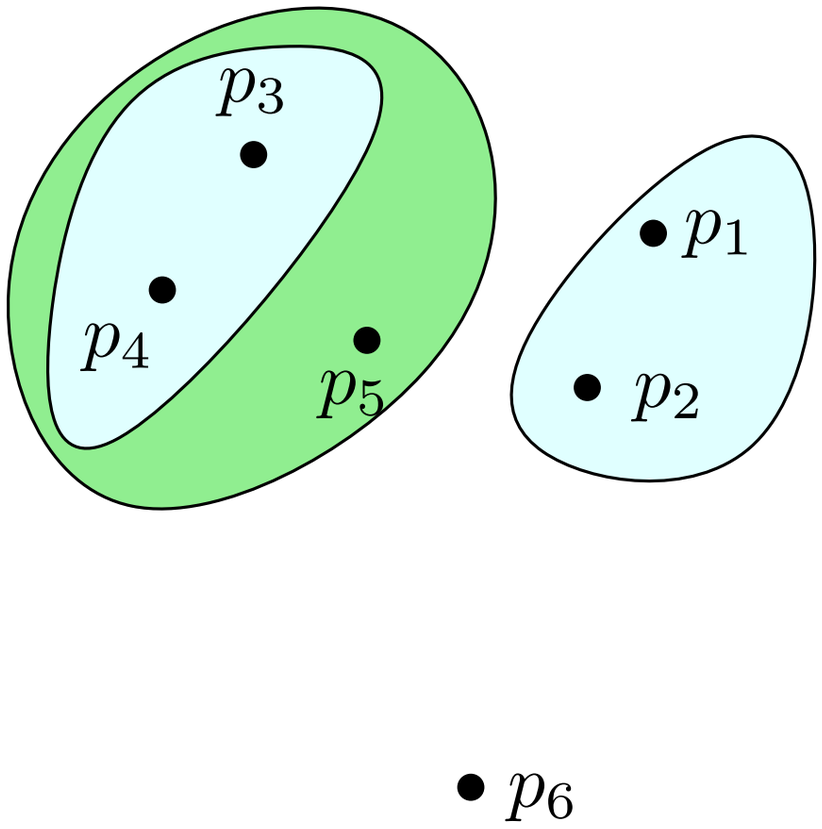


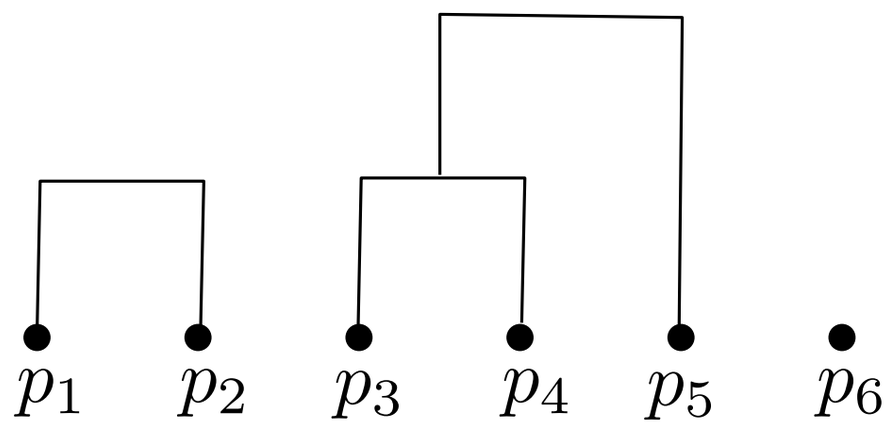
Hierarchical clustering algorithms

Goal: Build a hierarchy of clusters (nested family of partitions).

Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).



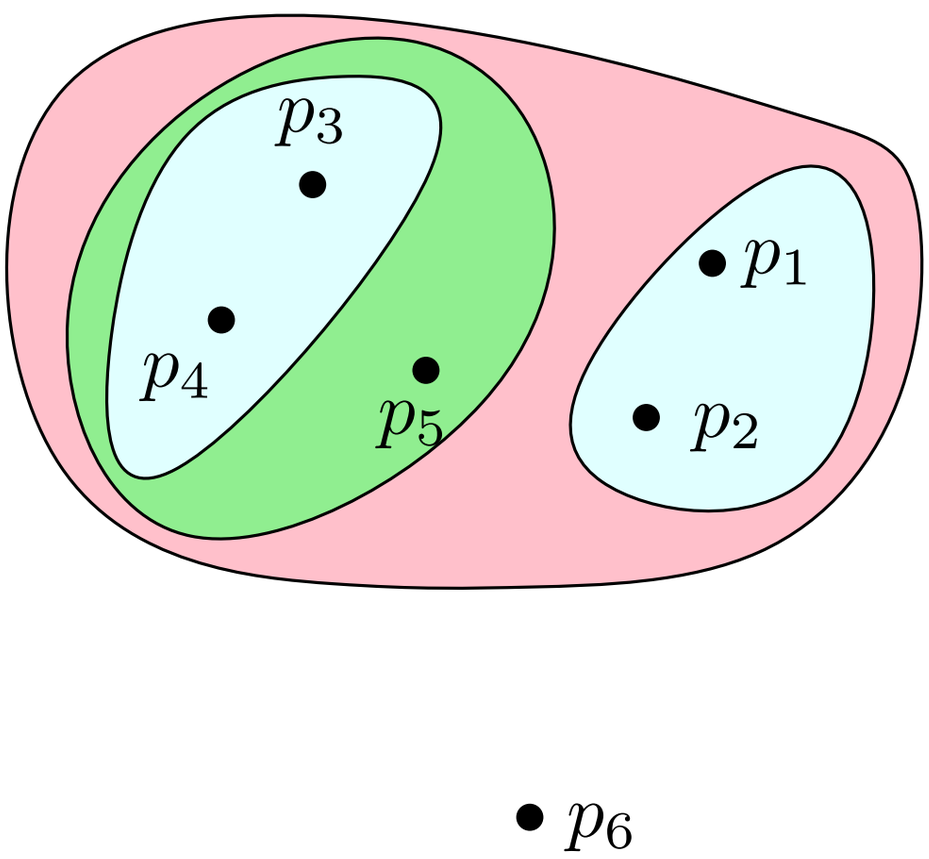


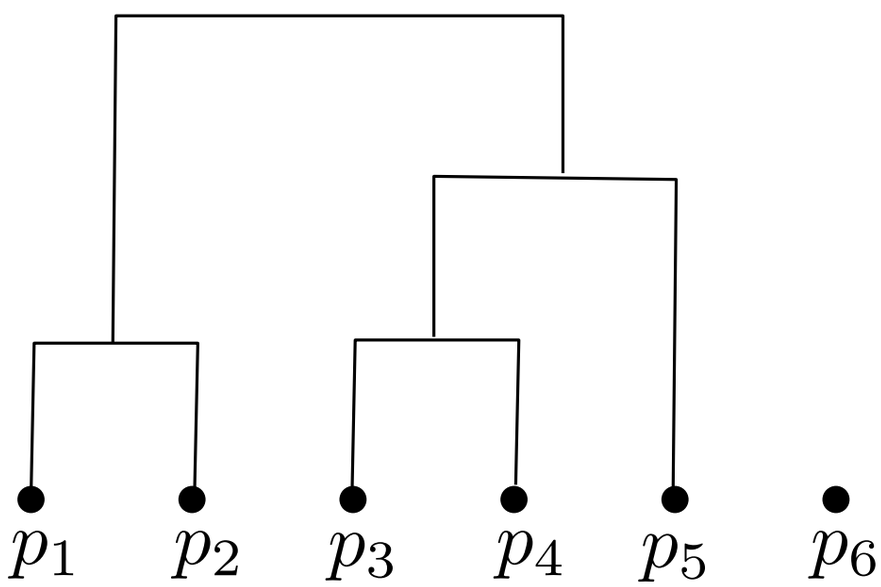
Hierarchical clustering algorithms

Goal: Build a hierarchy of clusters (nested family of partitions).

Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).



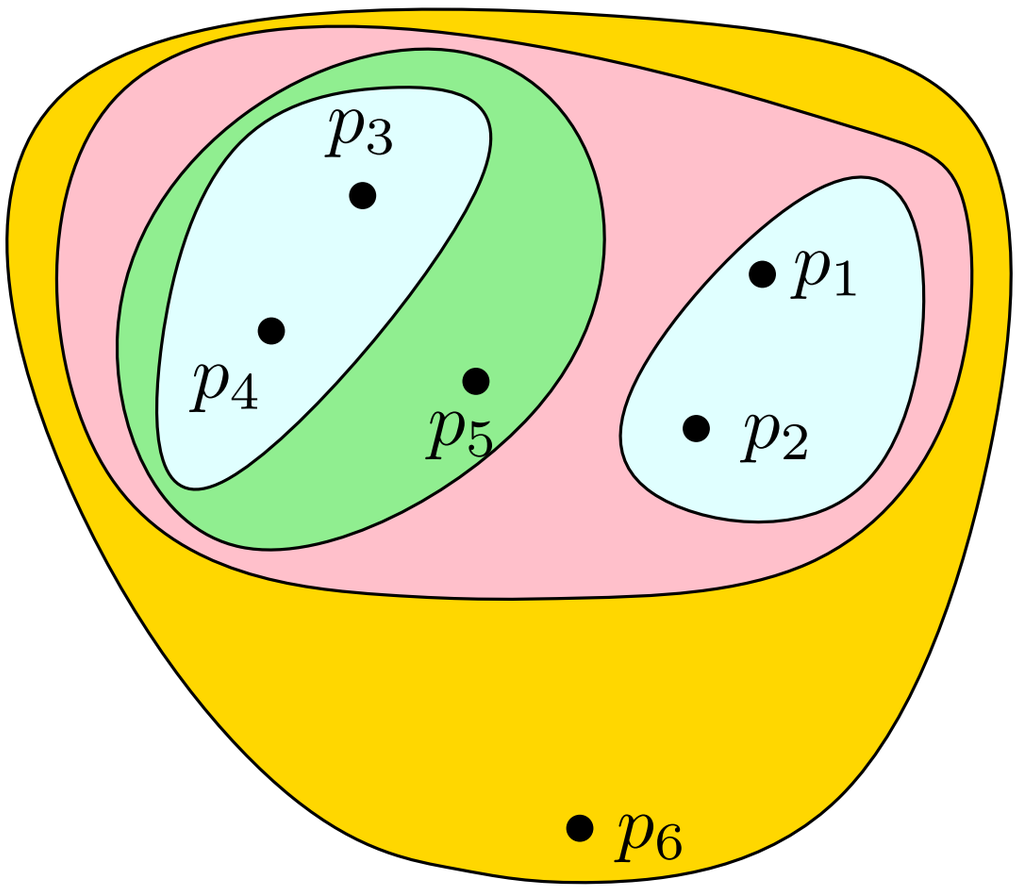


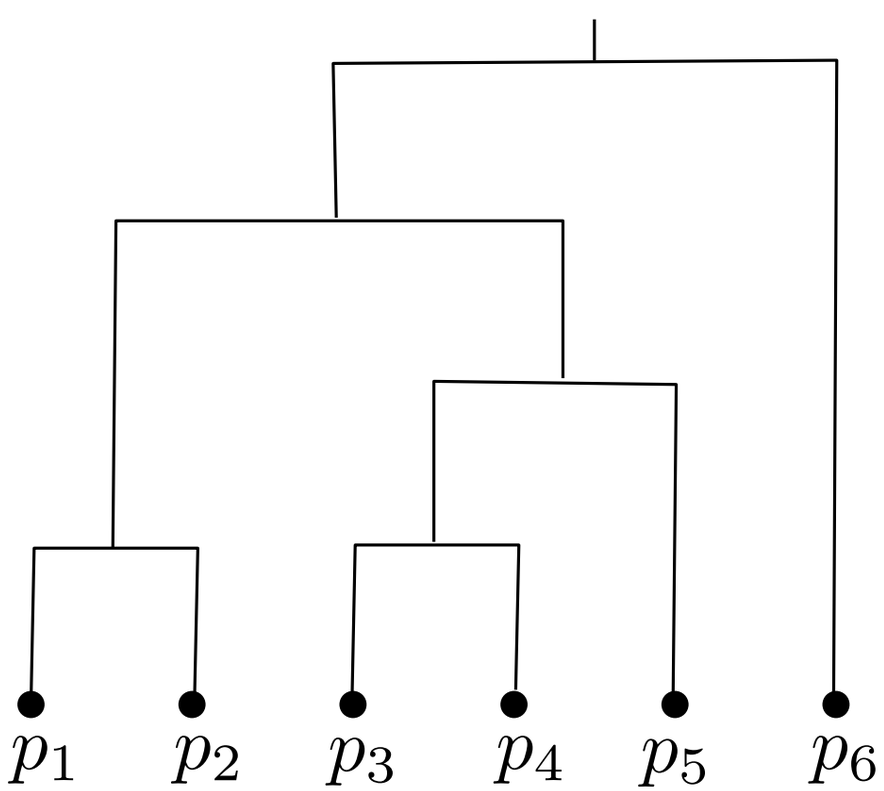
Hierarchical clustering algorithms

Goal: Build a hierarchy of clusters (nested family of partitions).

Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).



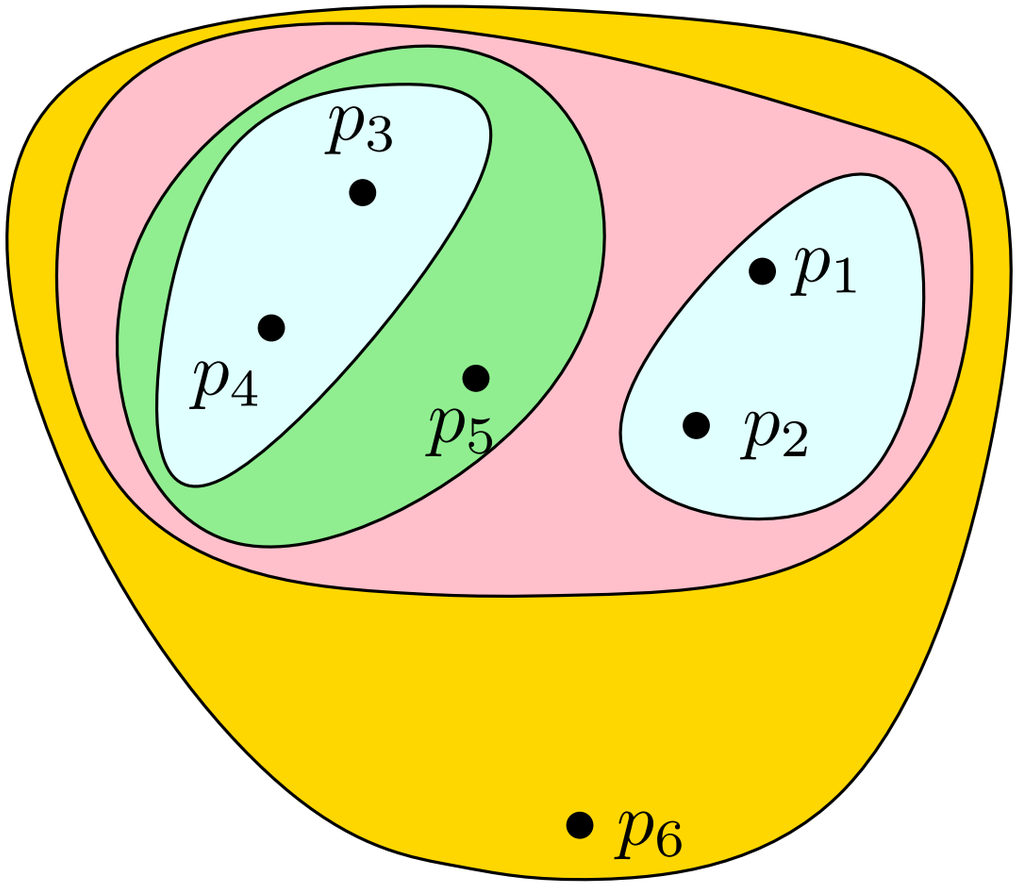


Hierarchical clustering algorithms

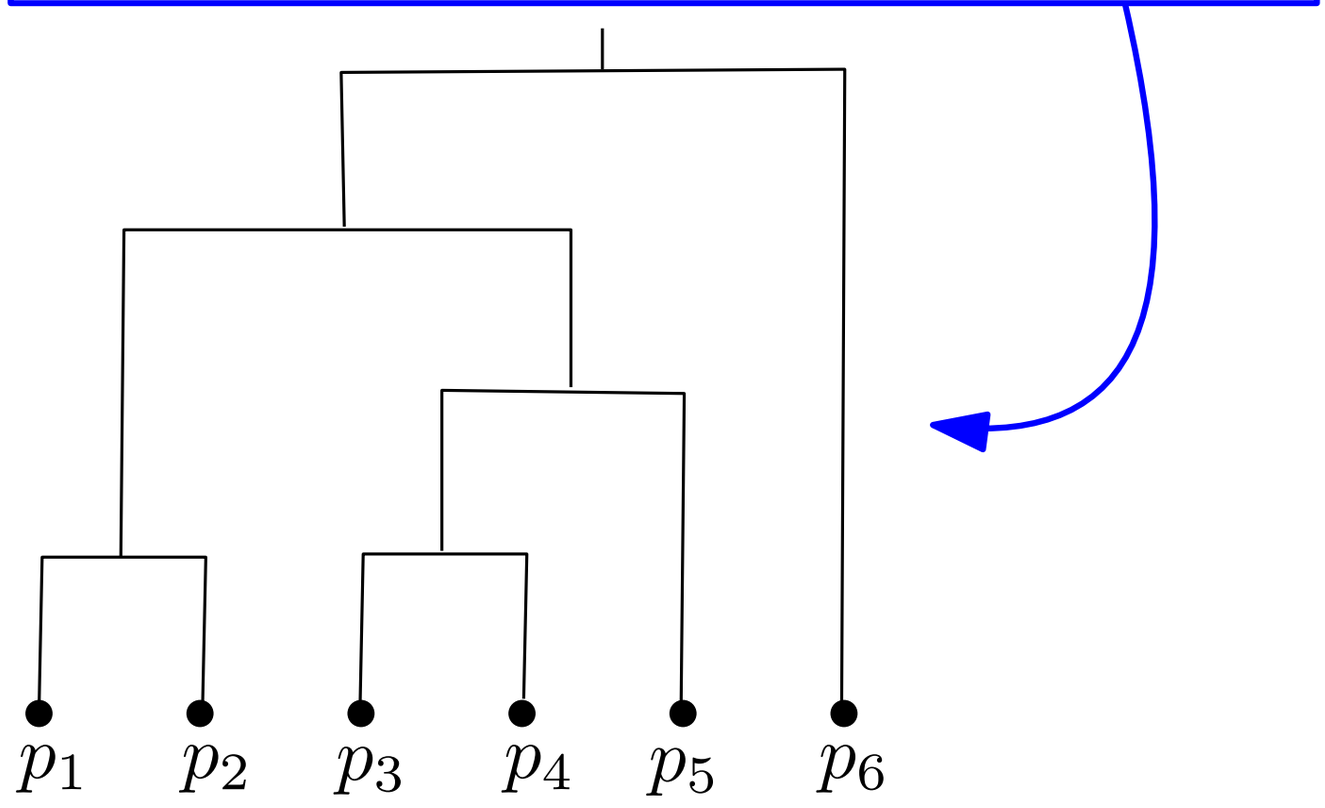
Goal: Build a hierarchy of clusters (nested family of partitions).

Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).



Dendogram, i.e., a tree such that:
- each leaf node is a singleton,
- each node represents a cluster,
- the root node contains the whole data,
- each internal node has two daughters, corresponding to the clusters that were merged to obtain it.



Hierarchical clustering algorithms

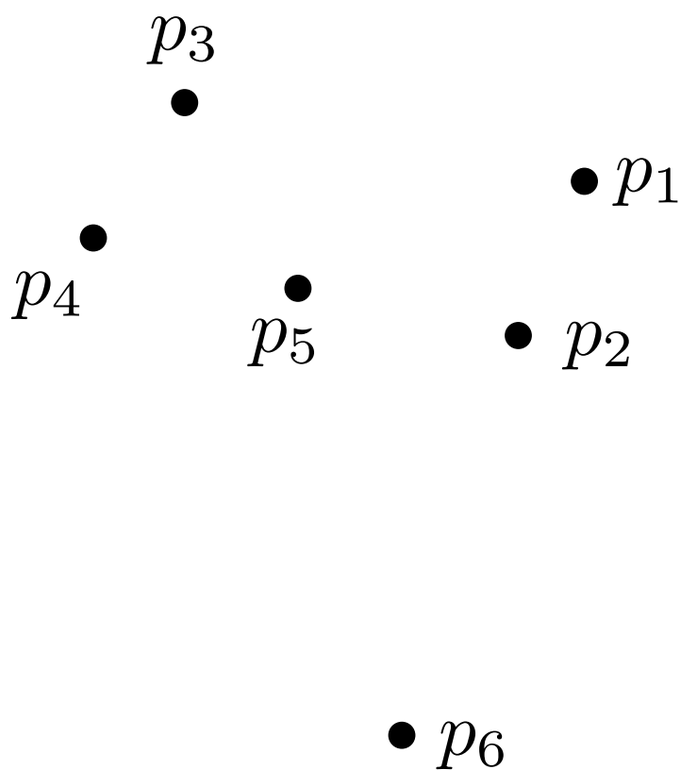
Goal: Build a hierarchy of clusters (nested family of partitions).

Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).

Dividing (top-down)

Start with a single global cluster and recursively split each cluster until reaching a stopping criterion.



Hierarchical clustering algorithms

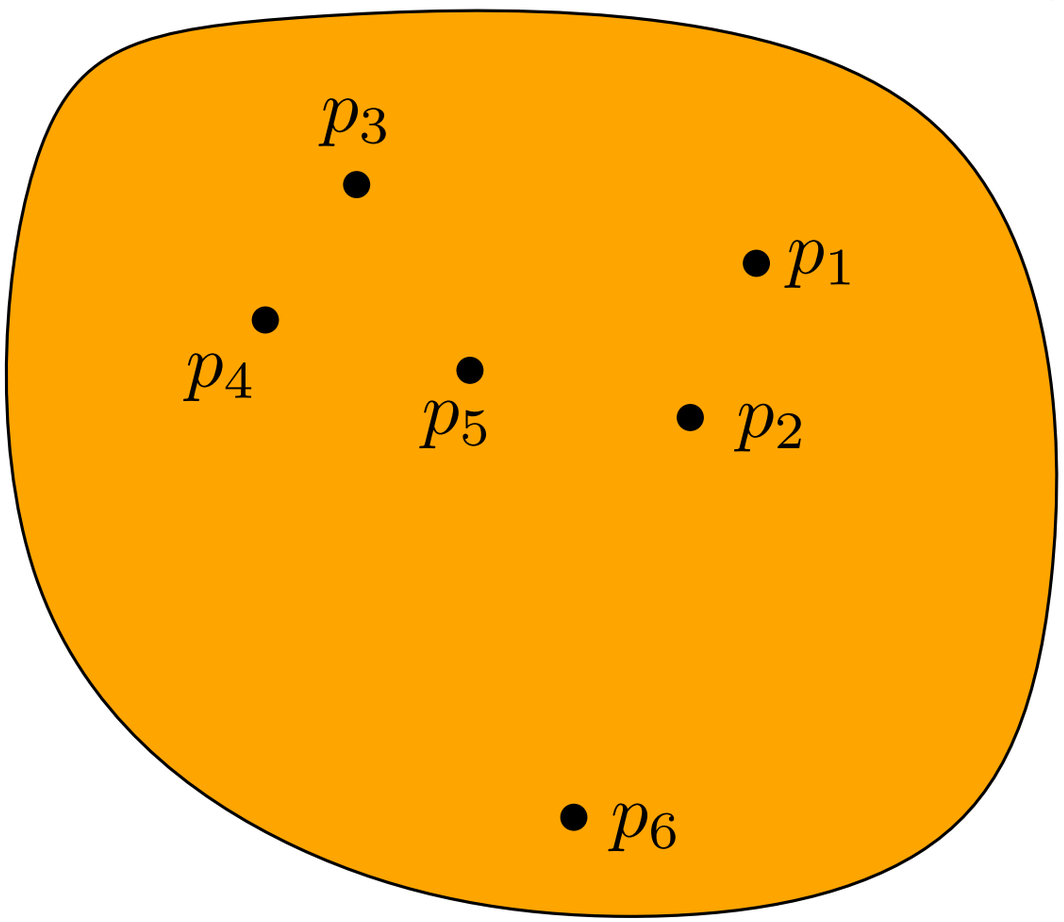
Goal: Build a hierarchy of clusters (nested family of partitions).

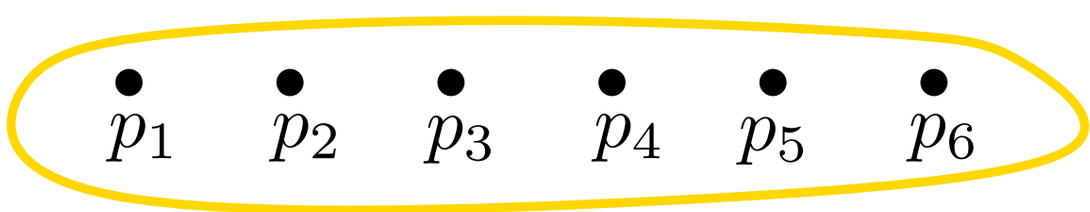
Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).

Dividing (top-down)

Start with a single global cluster and recursively split each cluster until reaching a stopping criterion.





Hierarchical clustering algorithms

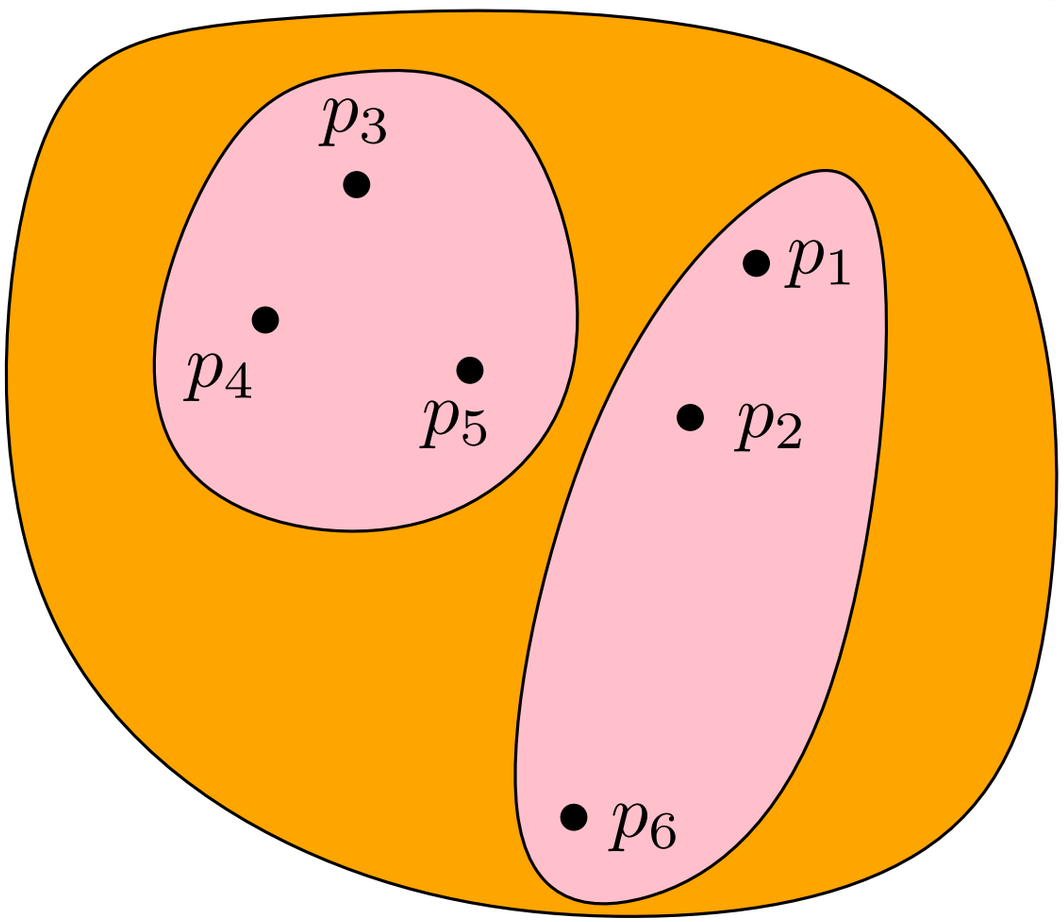
Goal: Build a hierarchy of clusters (nested family of partitions).

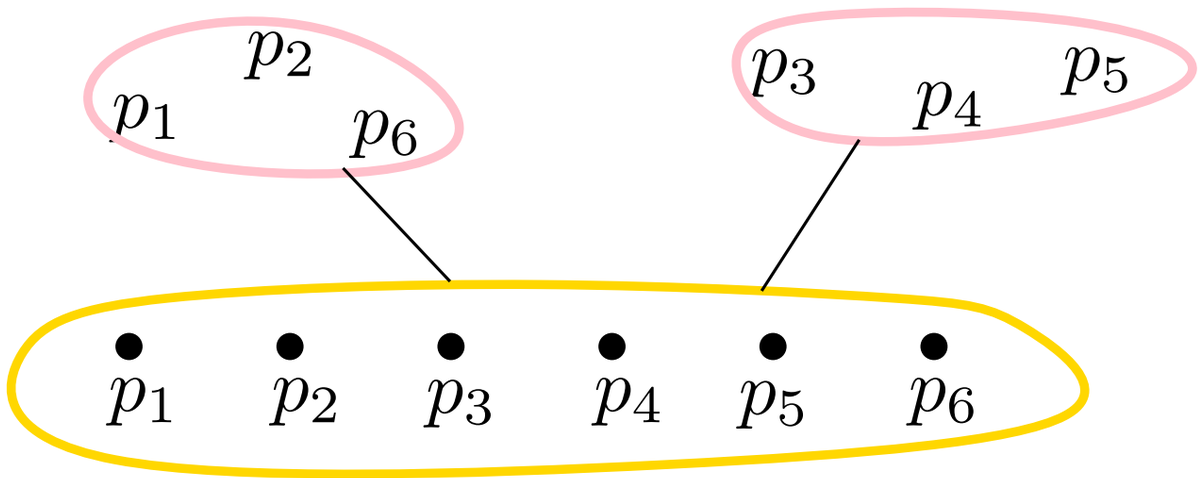
Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).

Dividing (top-down)

Start with a single global cluster and recursively split each cluster until reaching a stopping criterion.





Hierarchical clustering algorithms

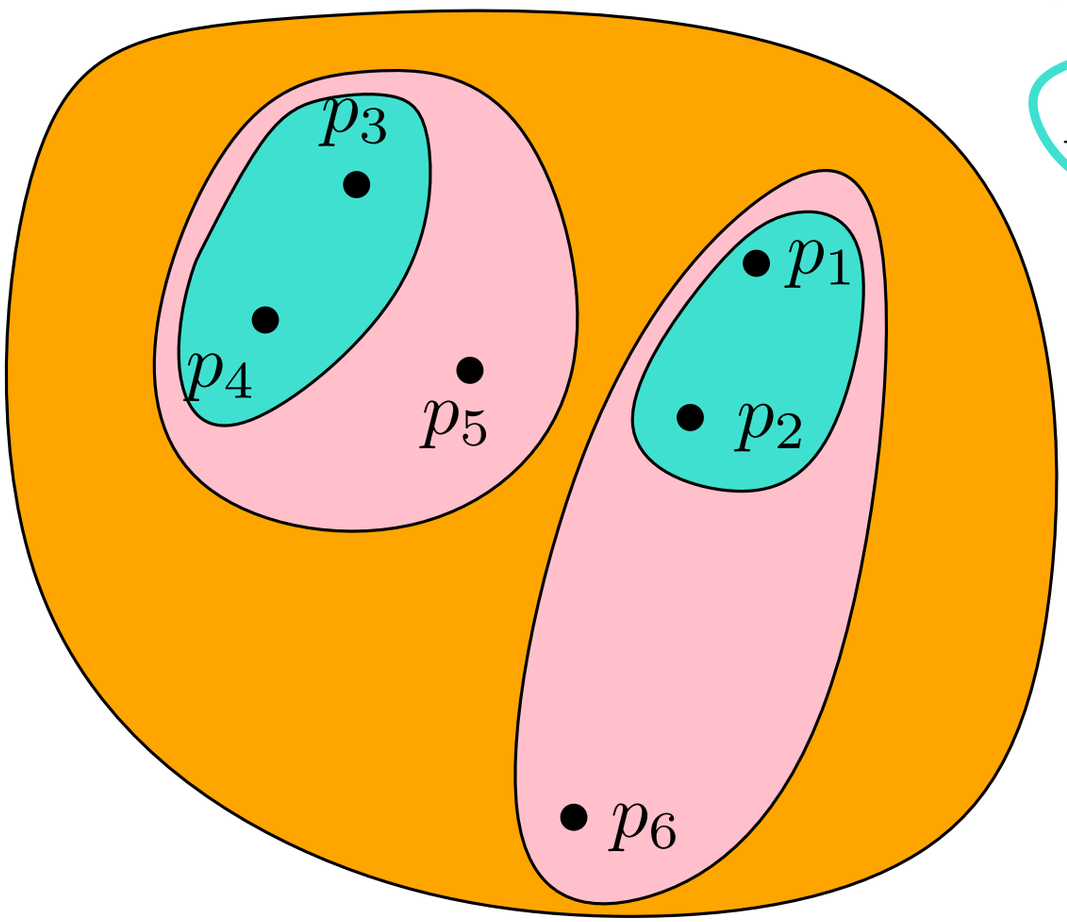
Goal: Build a hierarchy of clusters (nested family of partitions).

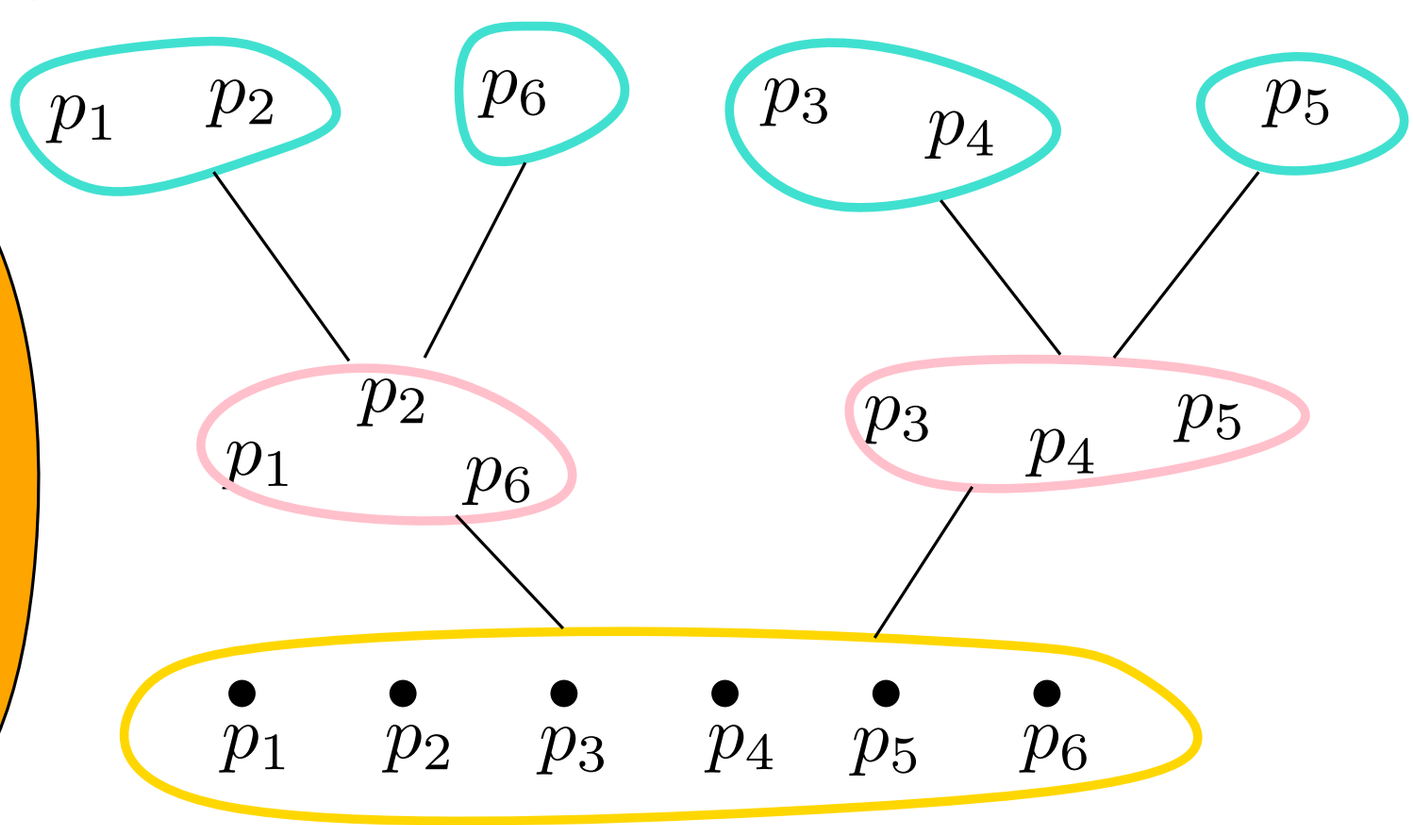
Agglomerative (bottom-up)

Start with single point cluster and recursively merge the most similar clusters to one parent cluster until reaching a stopping criterion (e.g., max distance or cluster number).

Dividing (top-down)

Start with a single global cluster and recursively split each cluster until reaching a stopping criterion.





Single linkage clustering

Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Single linkage clustering

Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).

Output: the resulting dendrogram.

Single linkage clustering

Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

sup: complete linkage

$\frac{1}{|C| \cdot |C'|} \sum$: average linkage

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).

Output: the resulting dendrogram.

Single linkage clustering

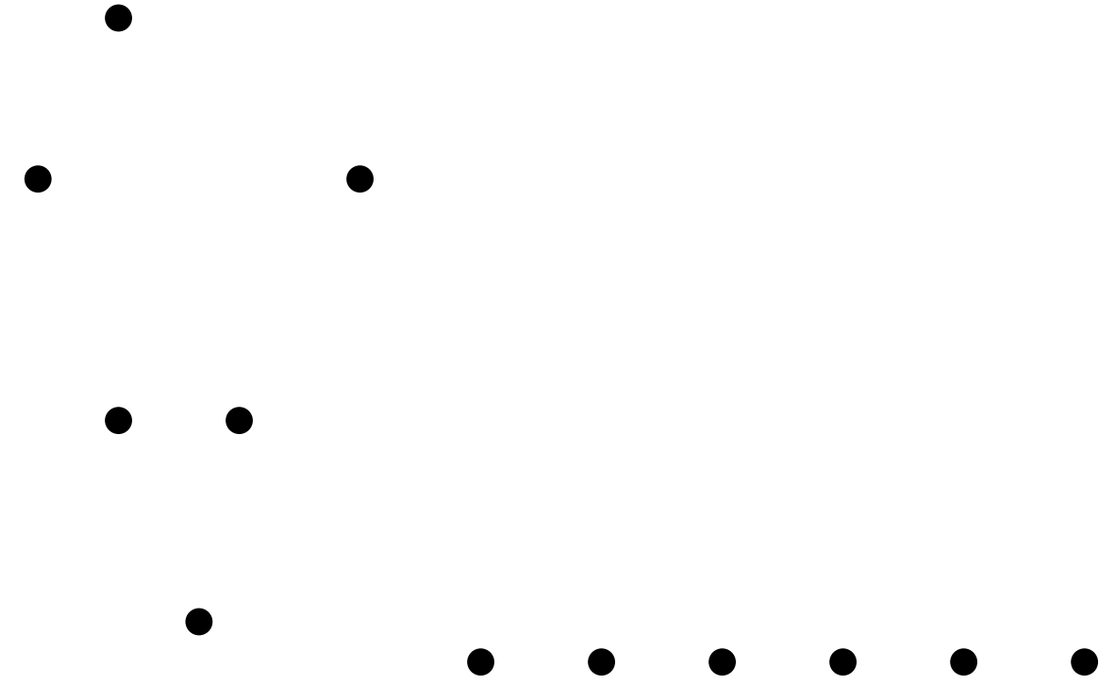
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).



Output: the resulting dendrogram.

Single linkage clustering

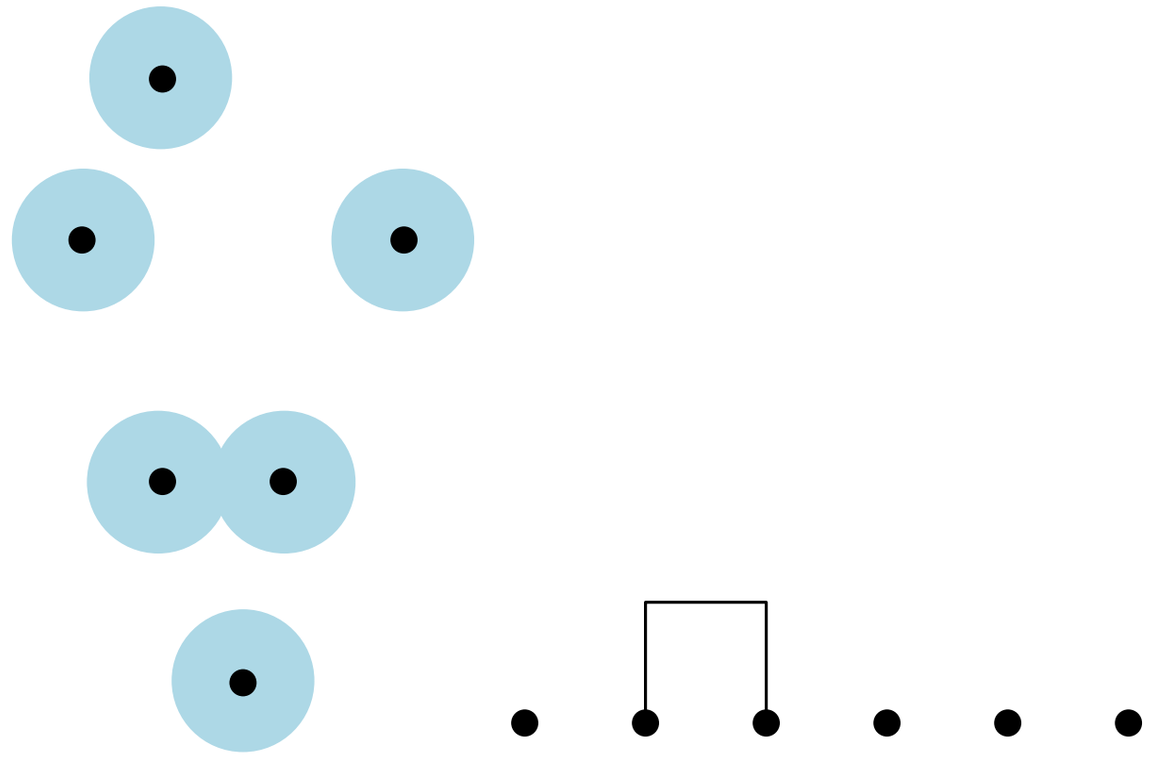
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).



Output: the resulting dendrogram.

Single linkage clustering

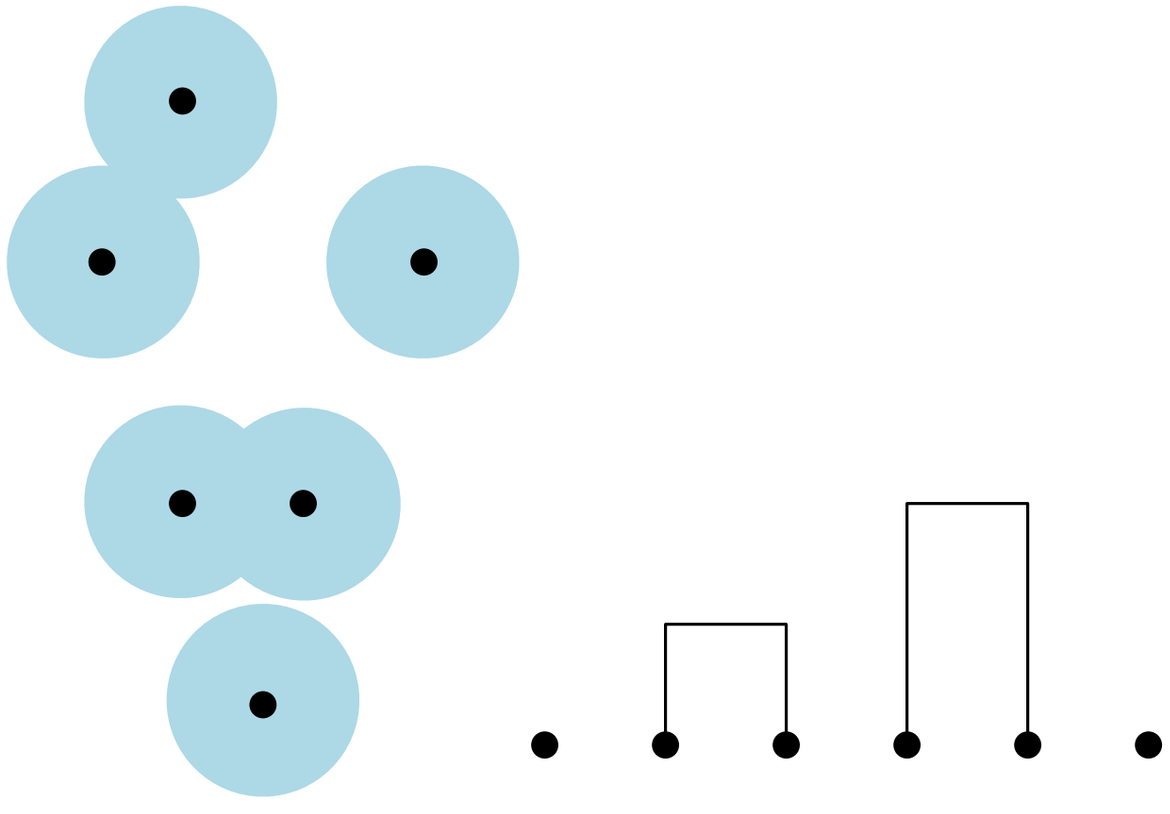
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).



Output: the resulting dendrogram.

Single linkage clustering

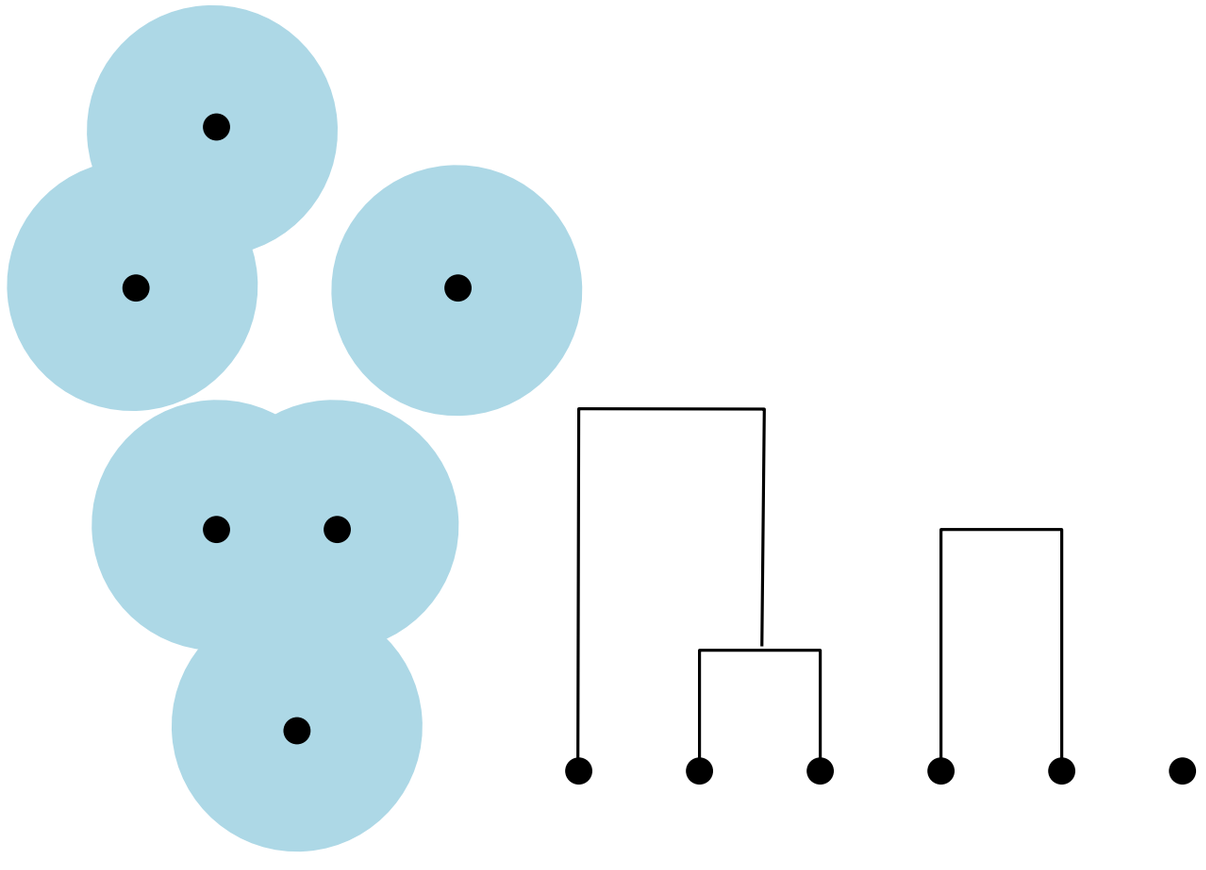
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).



Output: the resulting dendrogram.

Single linkage clustering

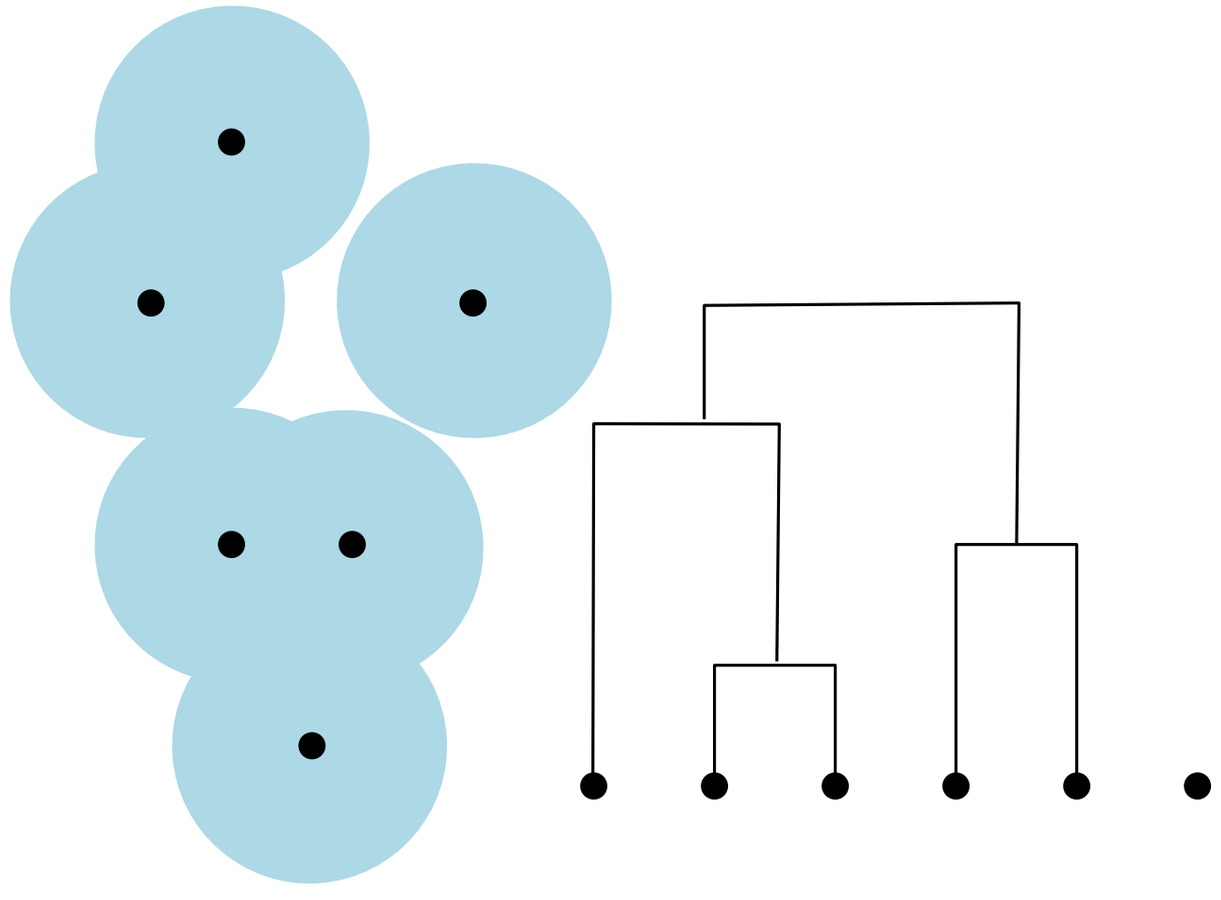
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).



Output: the resulting dendrogram.

Single linkage clustering

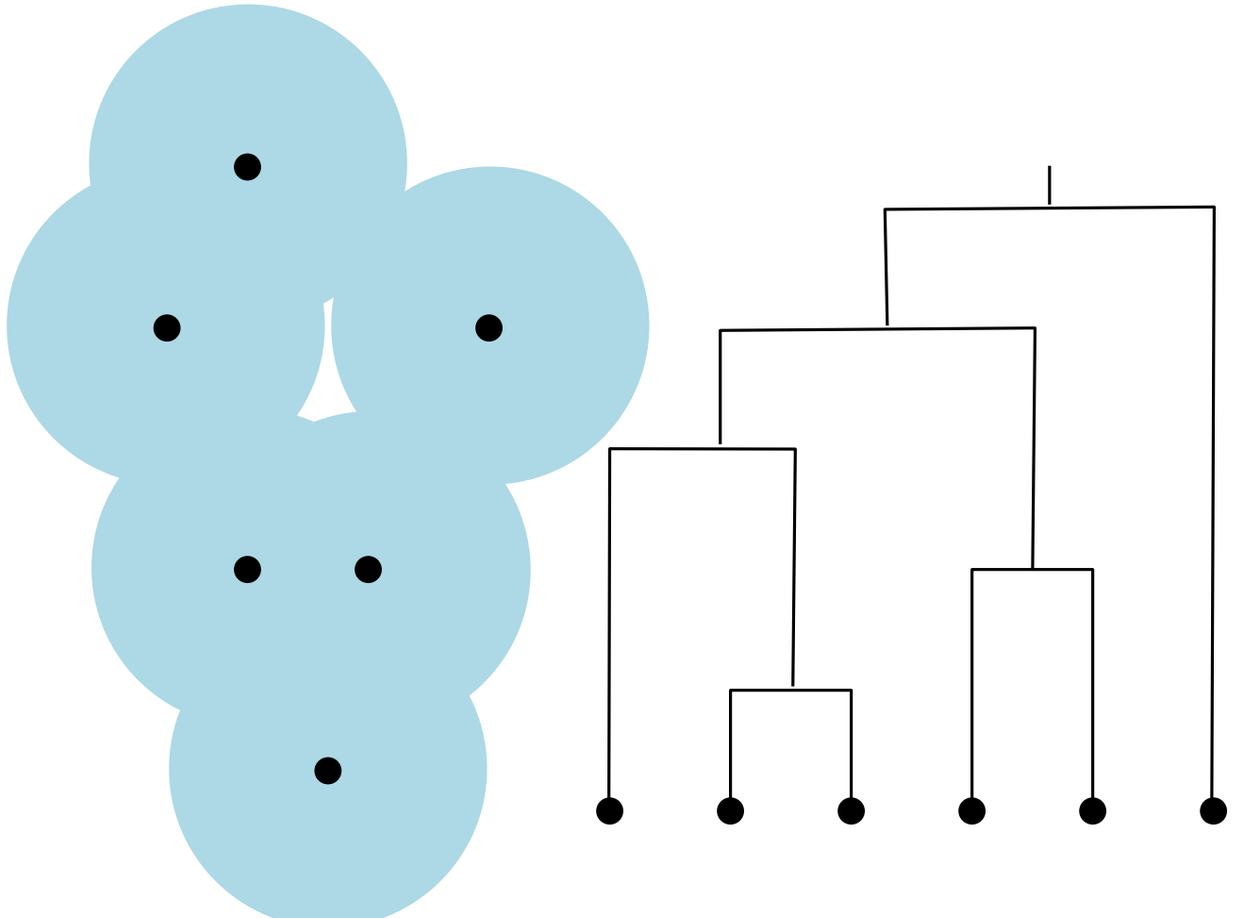
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

Given two clusters $C, C' \subseteq X_n$ let $d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$.

Agglomerative (bottom-up)

1. Start with a clustering where each x_i is a cluster.

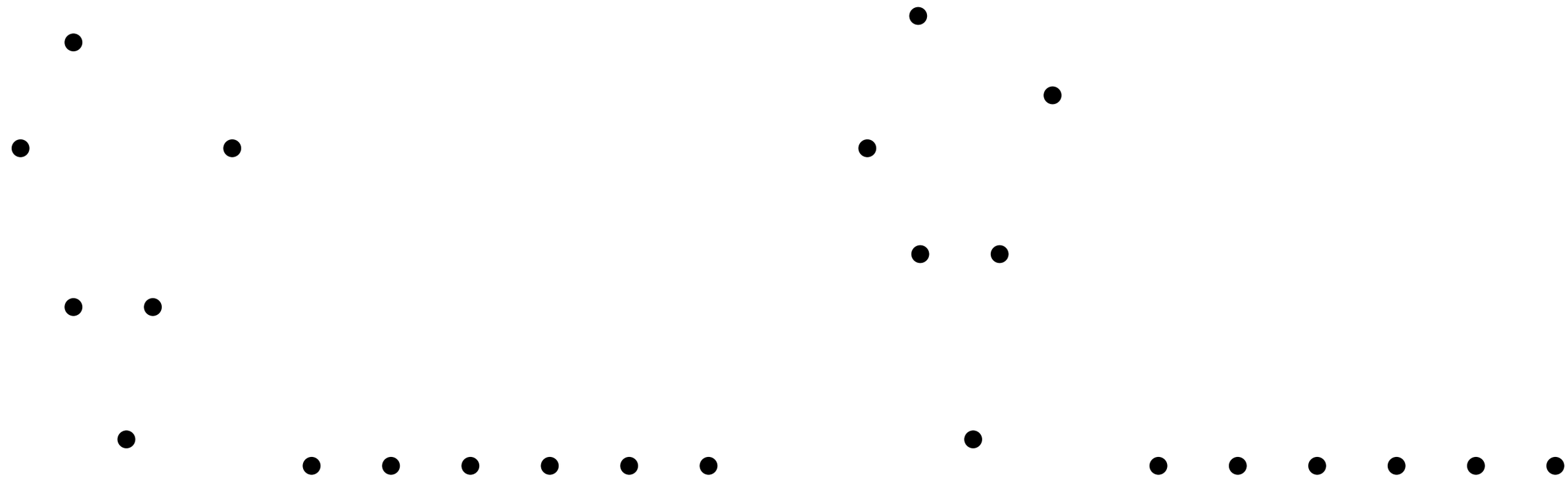
2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).



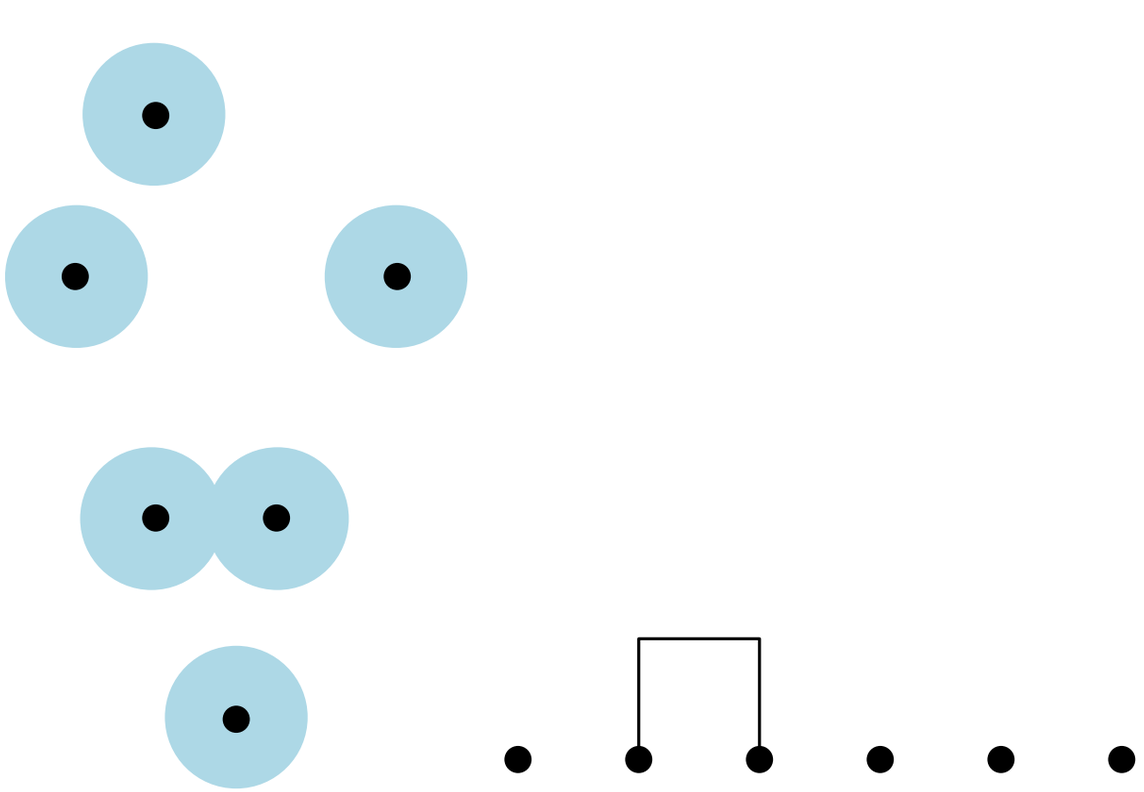
Output: the resulting dendrogram.

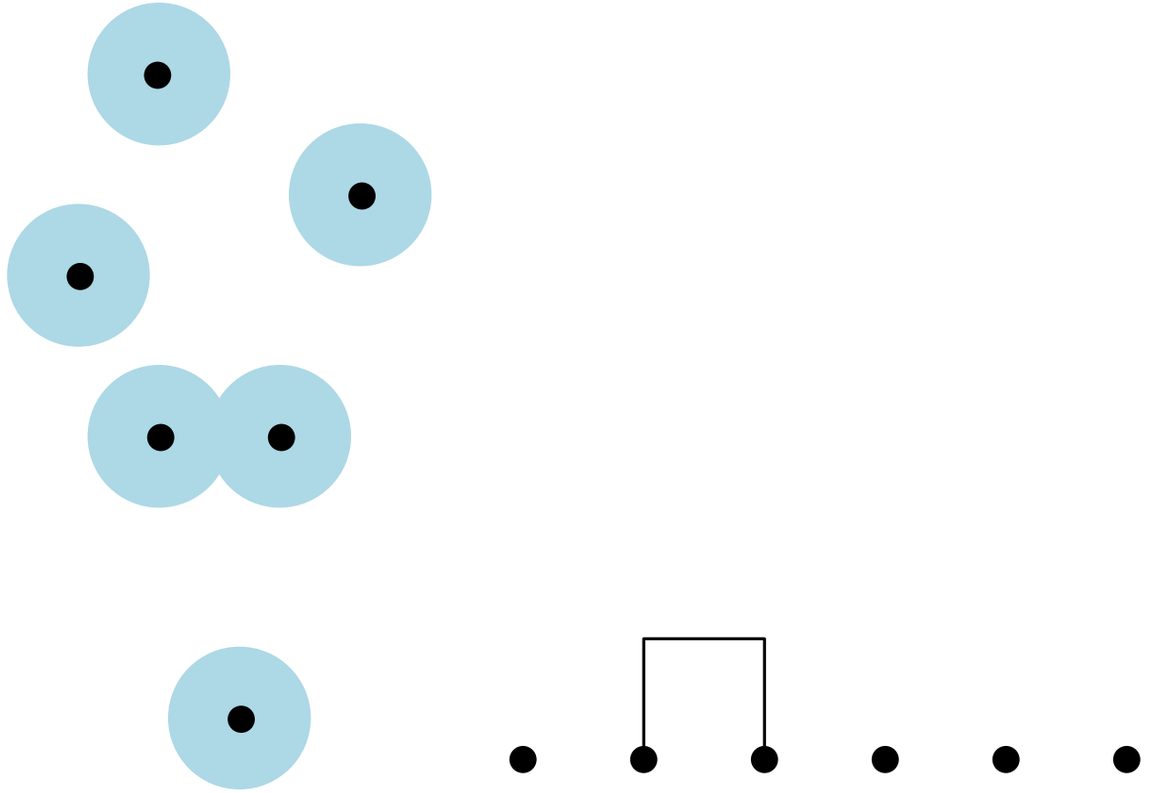
The (in)stability of dendrograms

The (in)stability of dendrograms

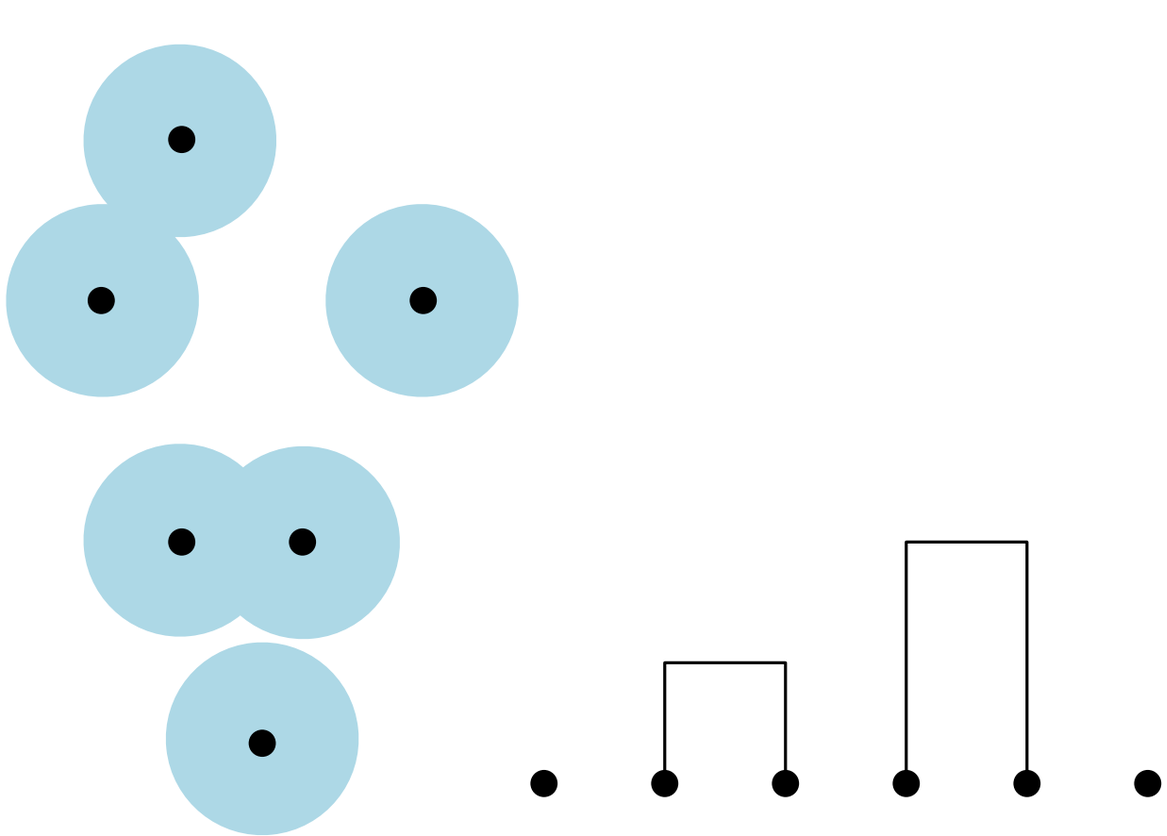


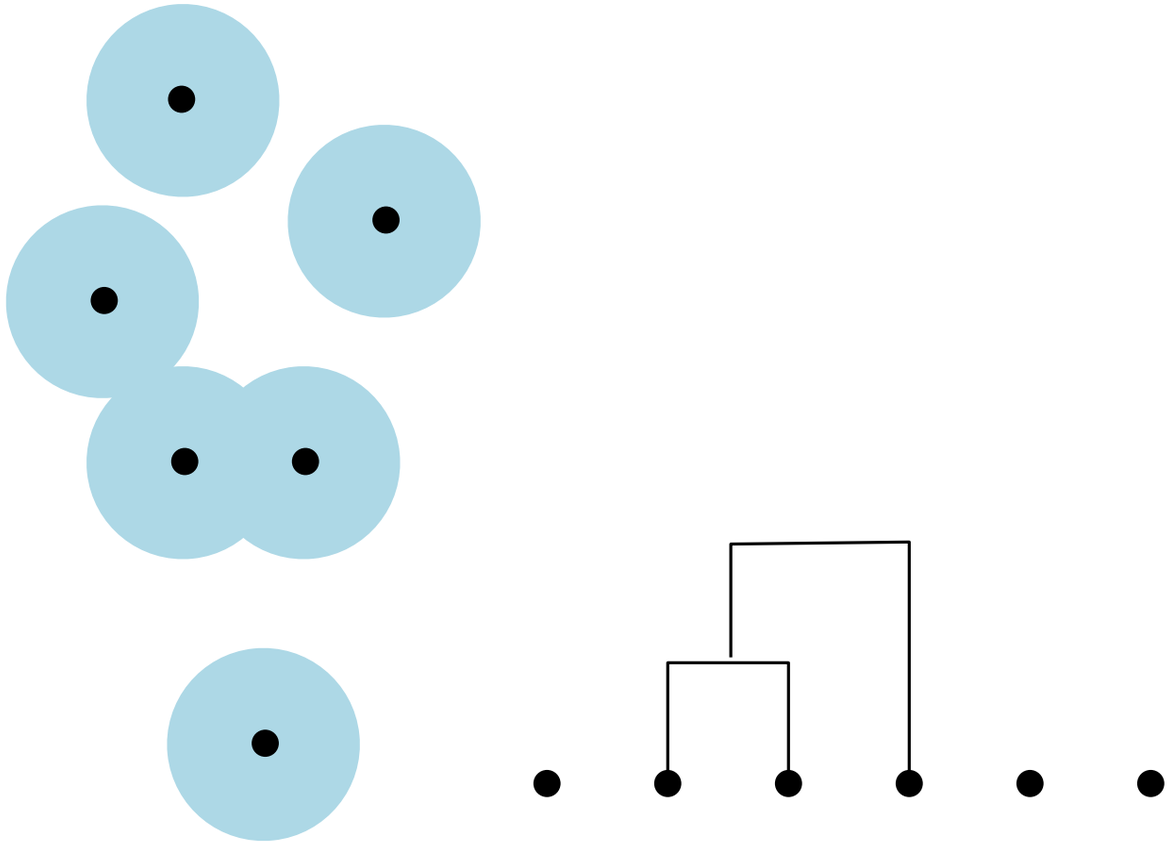
The (in)stability of dendrograms



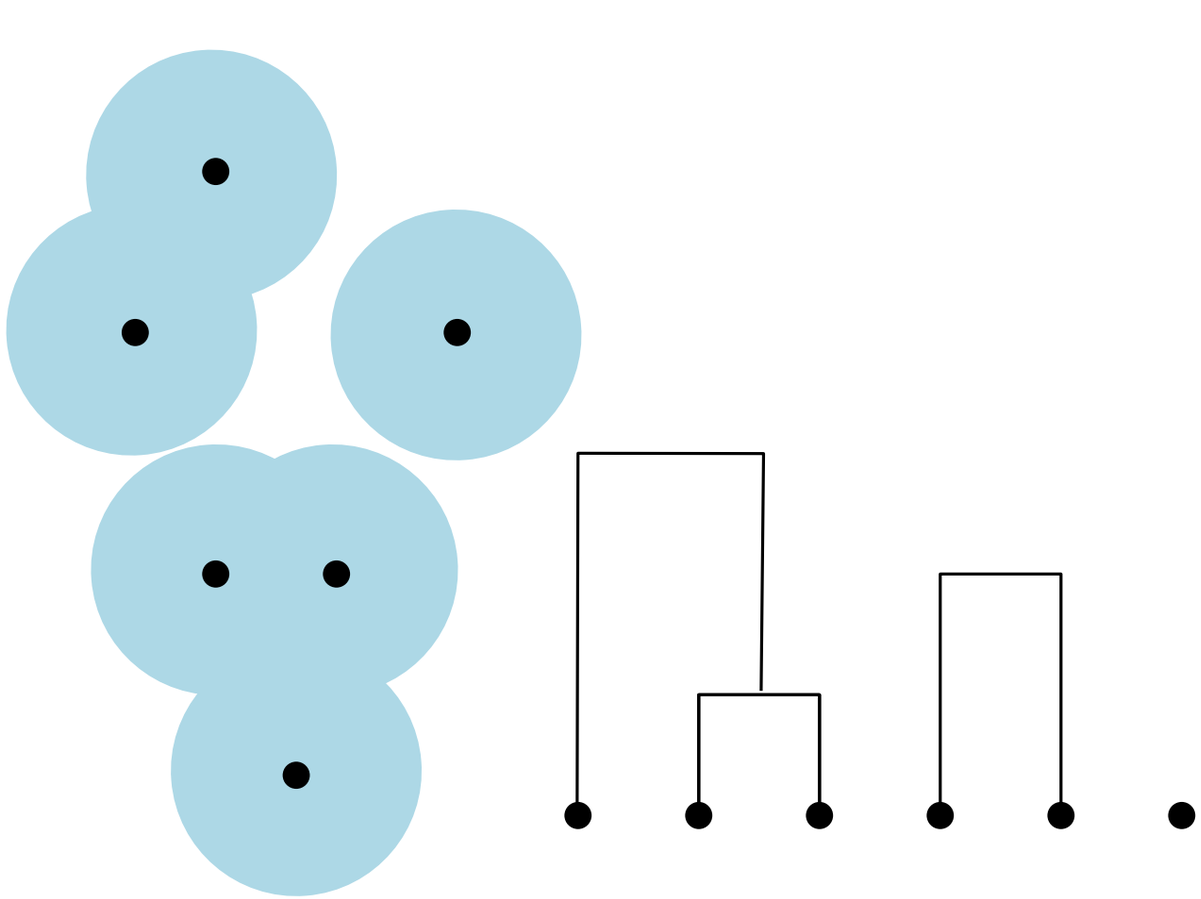


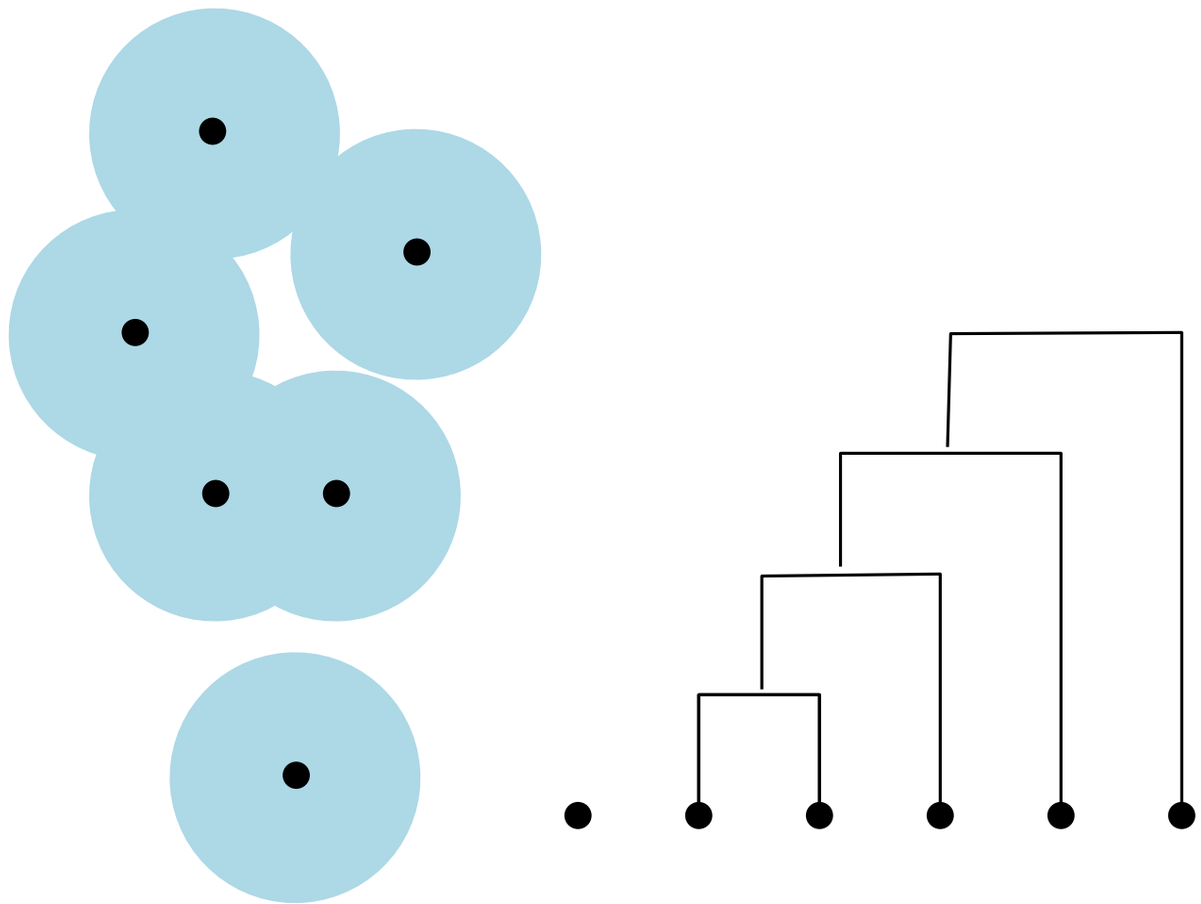
The (in)stability of dendrograms



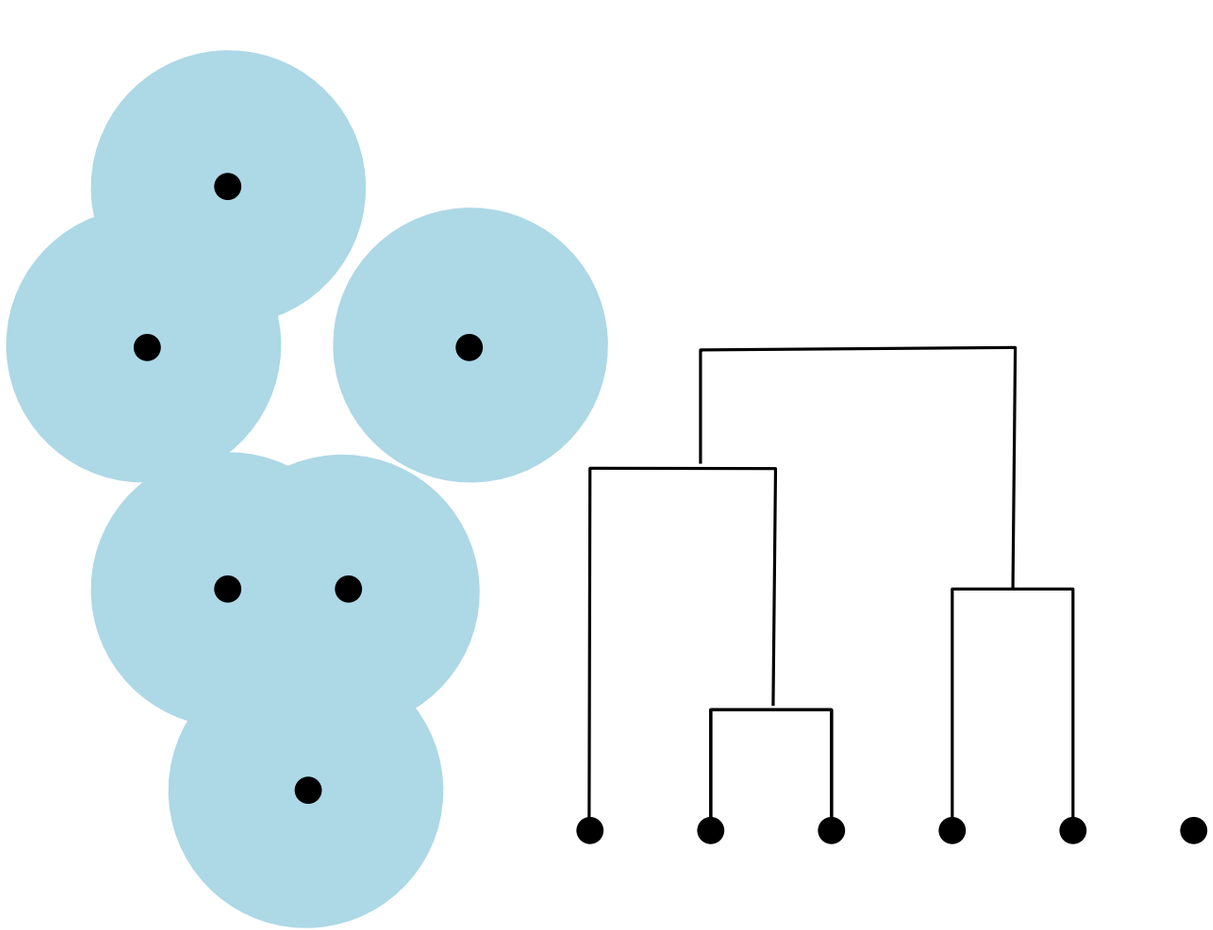


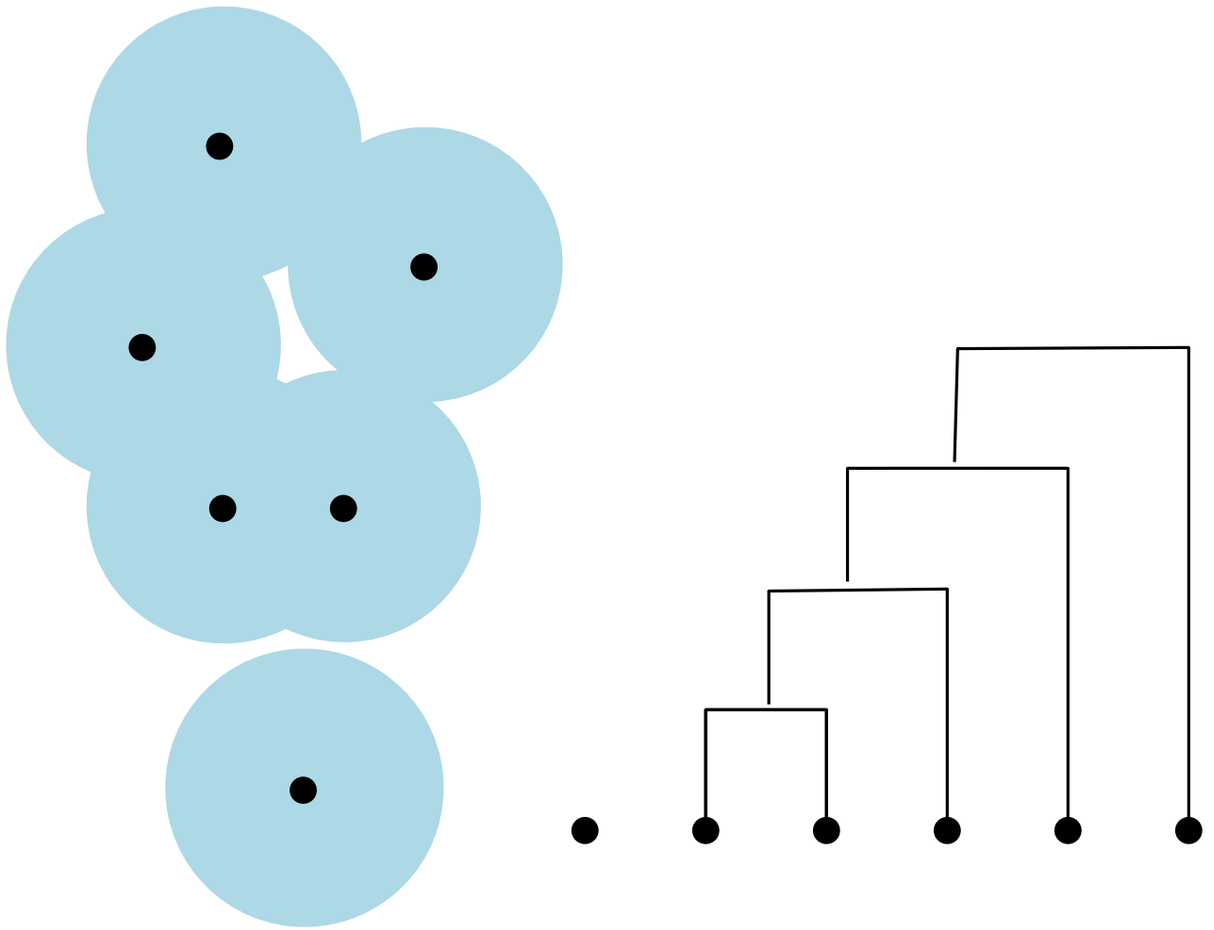
The (in)stability of dendrograms



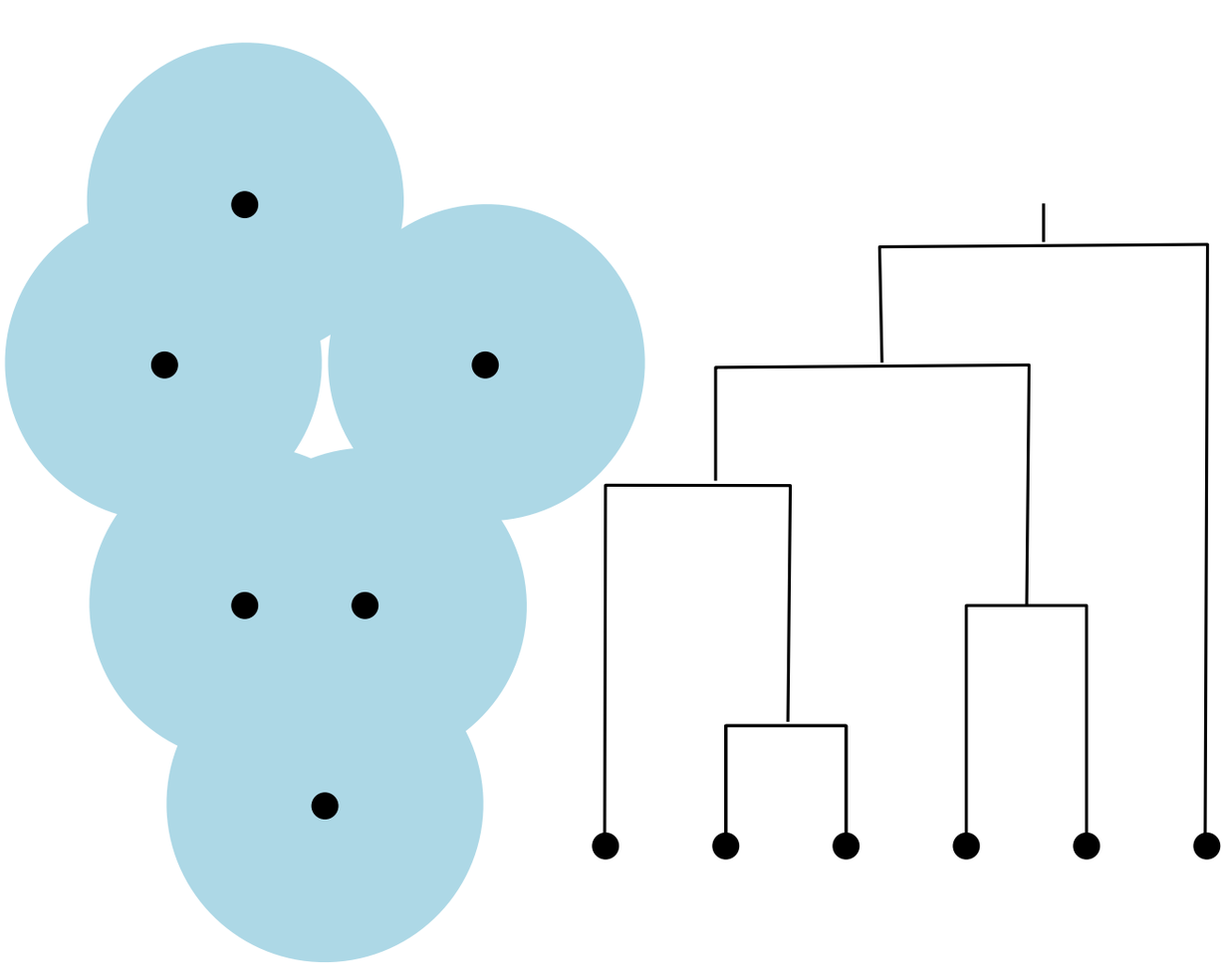


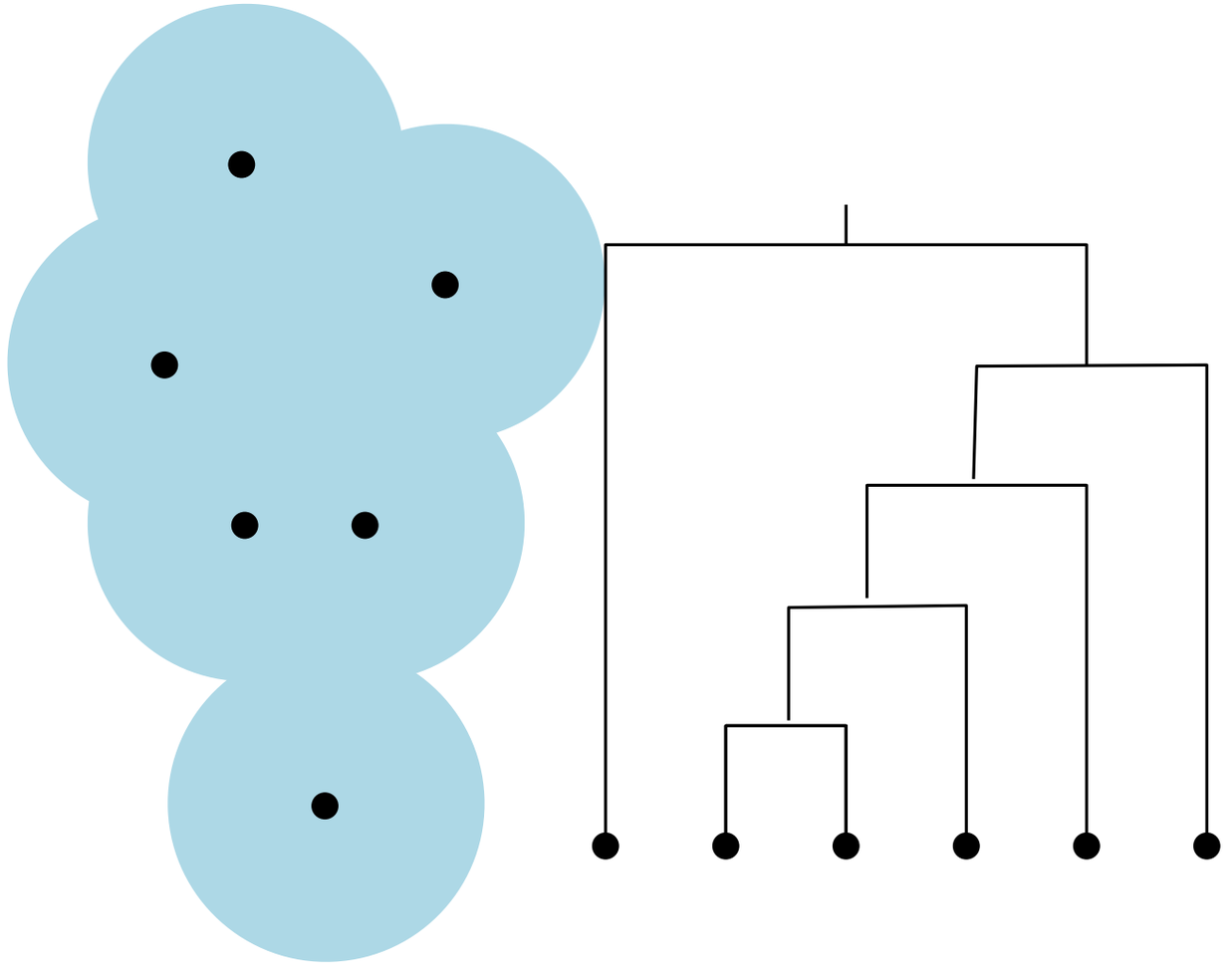
The (in)stability of dendrograms





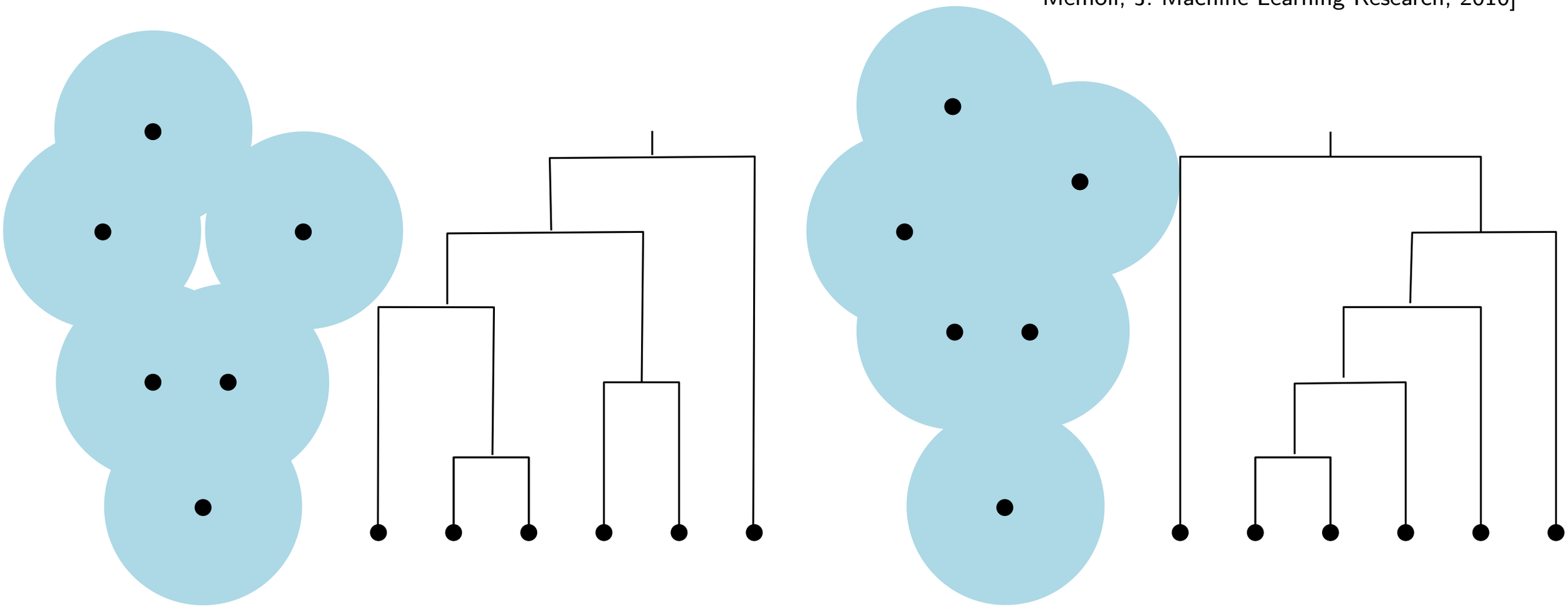
The (in)stability of dendrograms





The (in)stability of dendrograms

[*Characterization, Stability and Convergence of Hierarchical Clustering Methods*, Carlsson, Mémoli, J. Machine Learning Research, 2010]



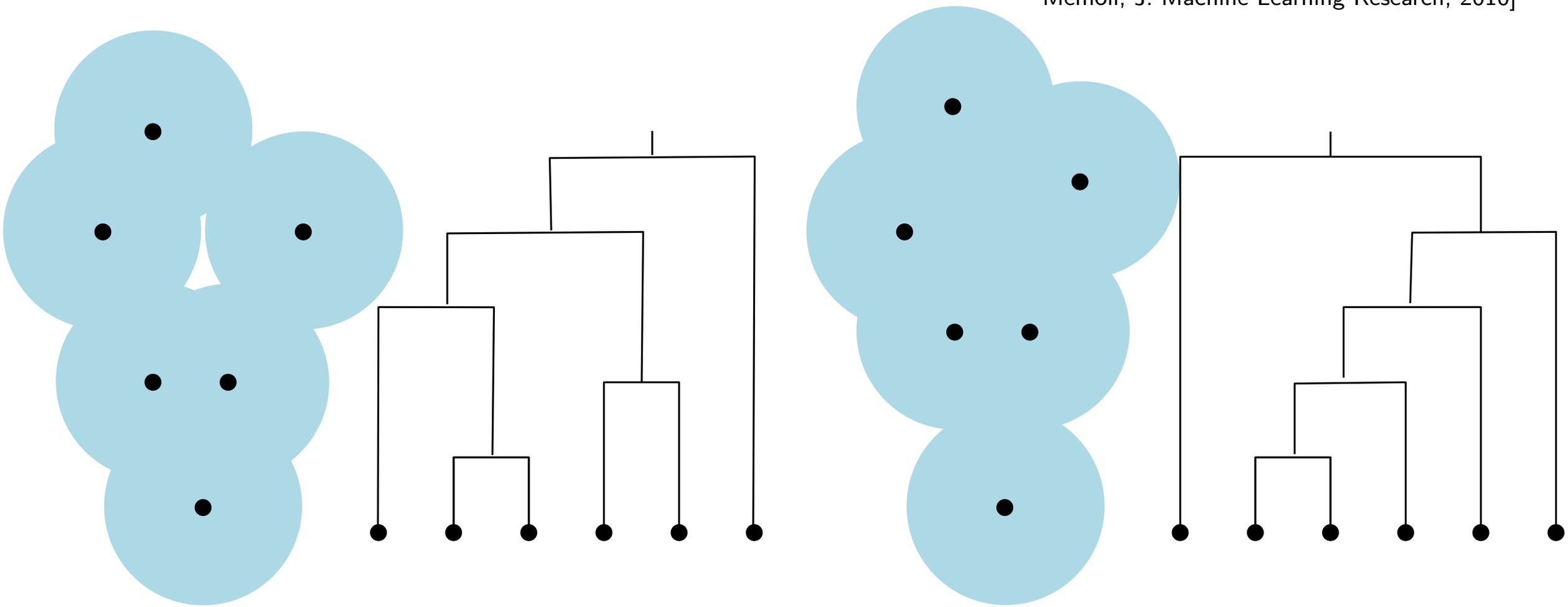
$d_{\mathcal{D}}(x, x') :=$ height of lowest common ancestor of x, x' in dendrogram $\mathcal{D}$.

Thm: $d_{GH}((X, d_{\mathcal{D}_X}), (Y, d_{\mathcal{D}_Y})) \leq d_{GH}((X, d_X), (Y, d_Y))$.

ultrametric!

The (in)stability of dendrograms

[*Characterization, Stability and Convergence of Hierarchical Clustering Methods*, Carlsson, Mémoli, J. Machine Learning Research, 2010]

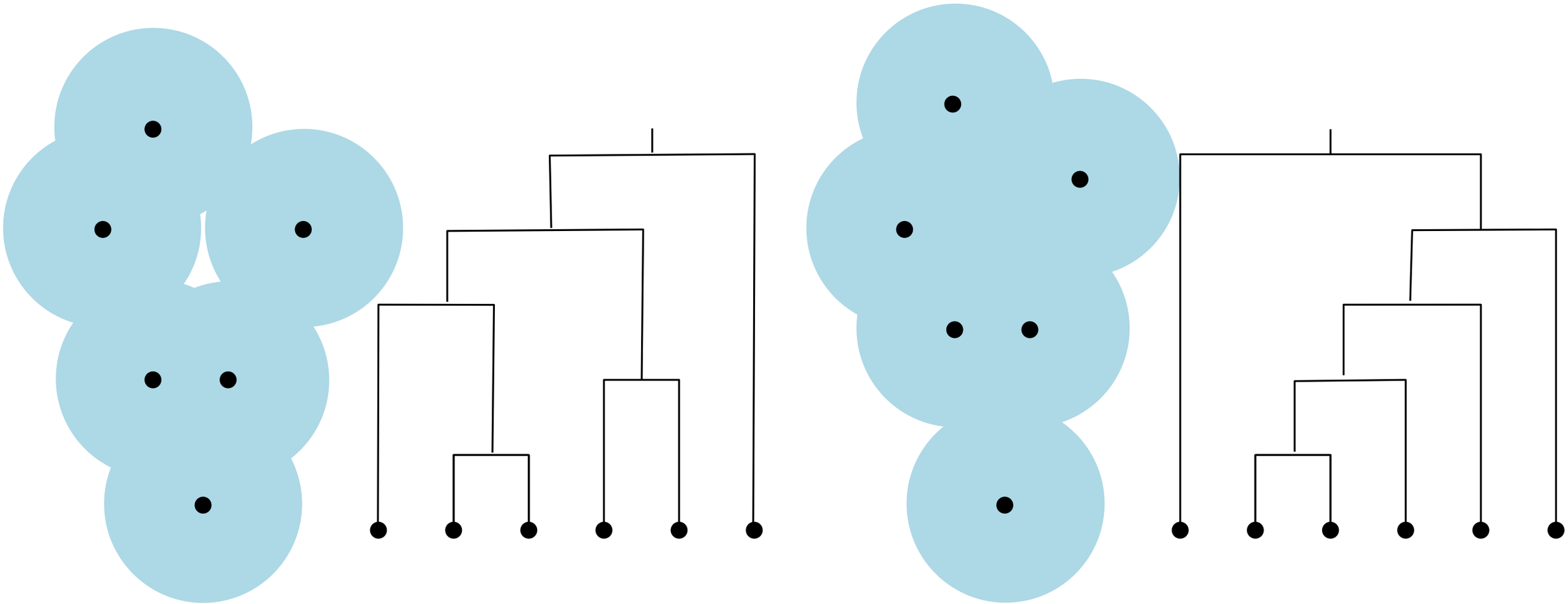


$d_{\mathcal{D}}(x, x') :=$ height of lowest common ancestor of x, x' in dendrogram $\mathcal{D}$.

Thm: $d_{GH}((X, d_{\mathcal{D}_X}), (Y, d_{\mathcal{D}_Y})) \leq d_{GH}((X, d_X), (Y, d_Y))$. ultrametric!

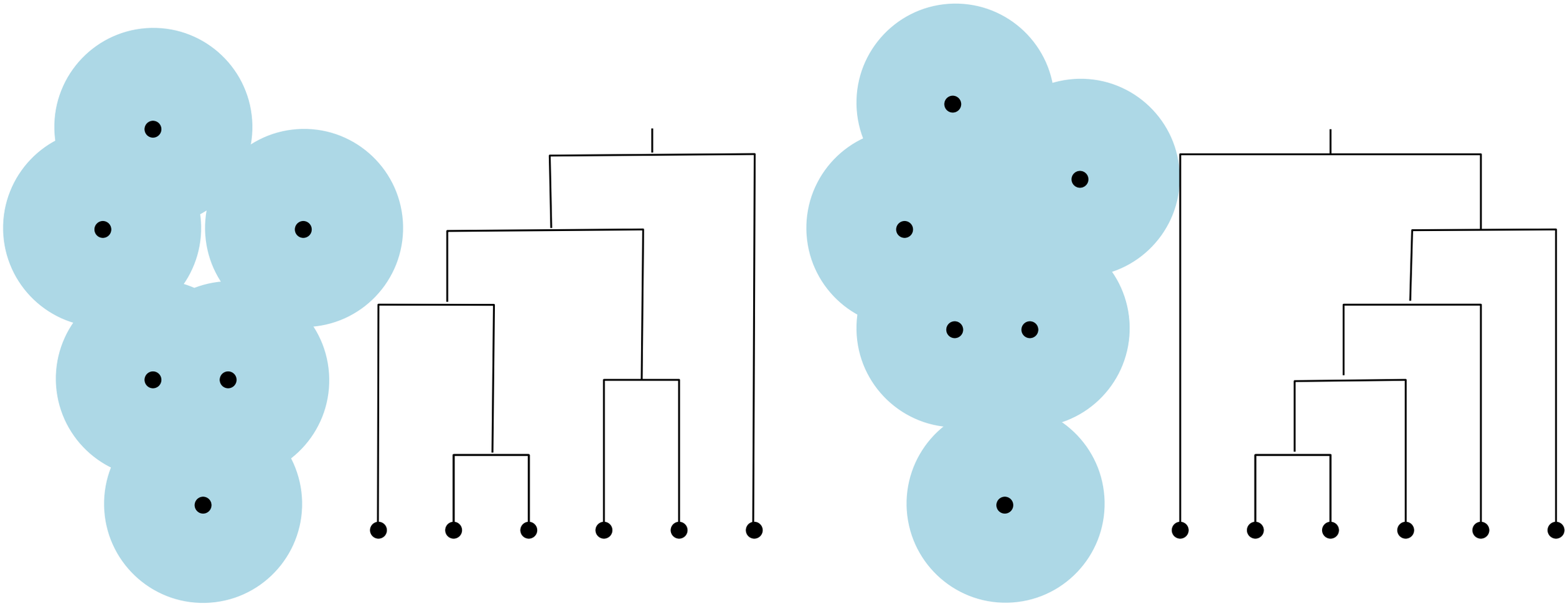
This is actually not true for complete and average clustering.

The (in)stability of dendrograms



Small perturbations on the input data can induce wide changes in the structure of the output dendrograms. However, the merging times (height of dendrogram nodes) remain stable.

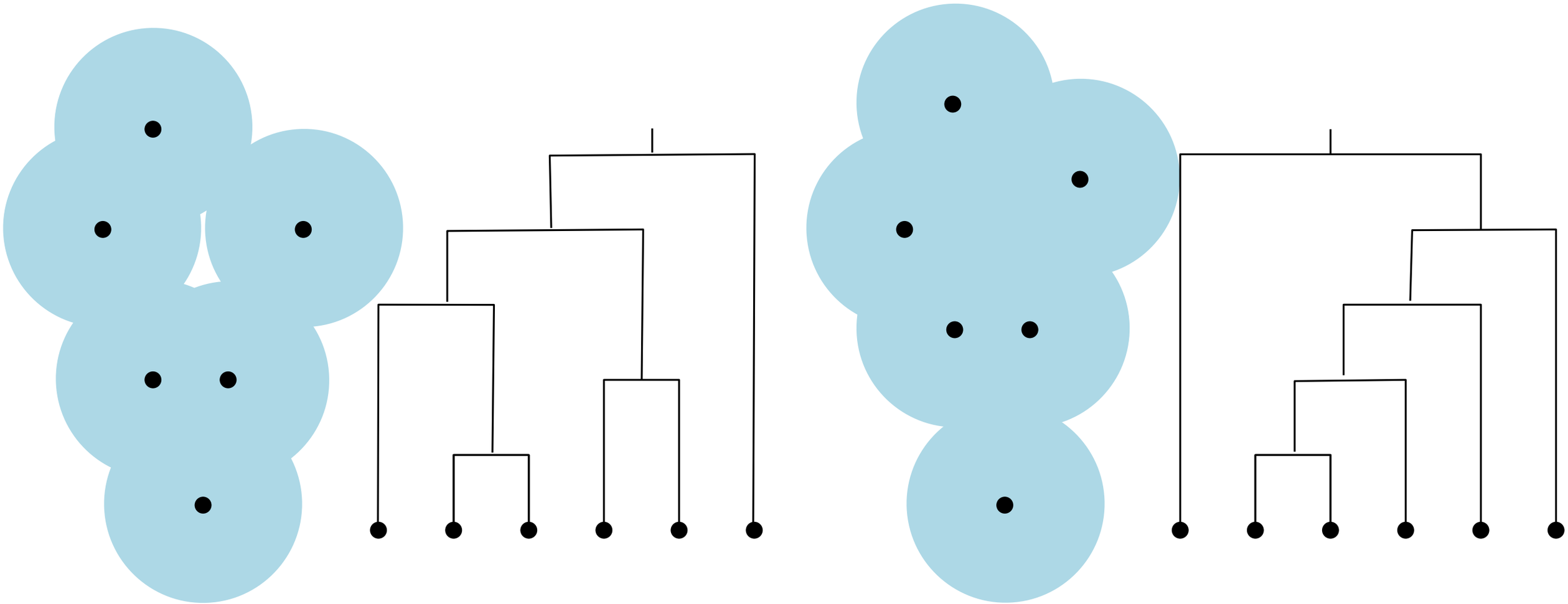
The (in)stability of dendrograms



Small perturbations on the input data can induce wide changes in the structure of the output dendrograms. However, the merging times (height of dendrogram nodes) remain stable.

Moreover, single linkage clustering keeps track of the evolution of the connected components of the distance function to the data (for Euclidean data).

The (in)stability of dendrograms



Small perturbations on the input data can induce wide changes in the structure of the output dendrograms. However, the merging times (height of dendrogram nodes) remain stable.

Moreover, single linkage clustering keeps track of the evolution of the connected components of the distance function to the data (for Euclidean data).

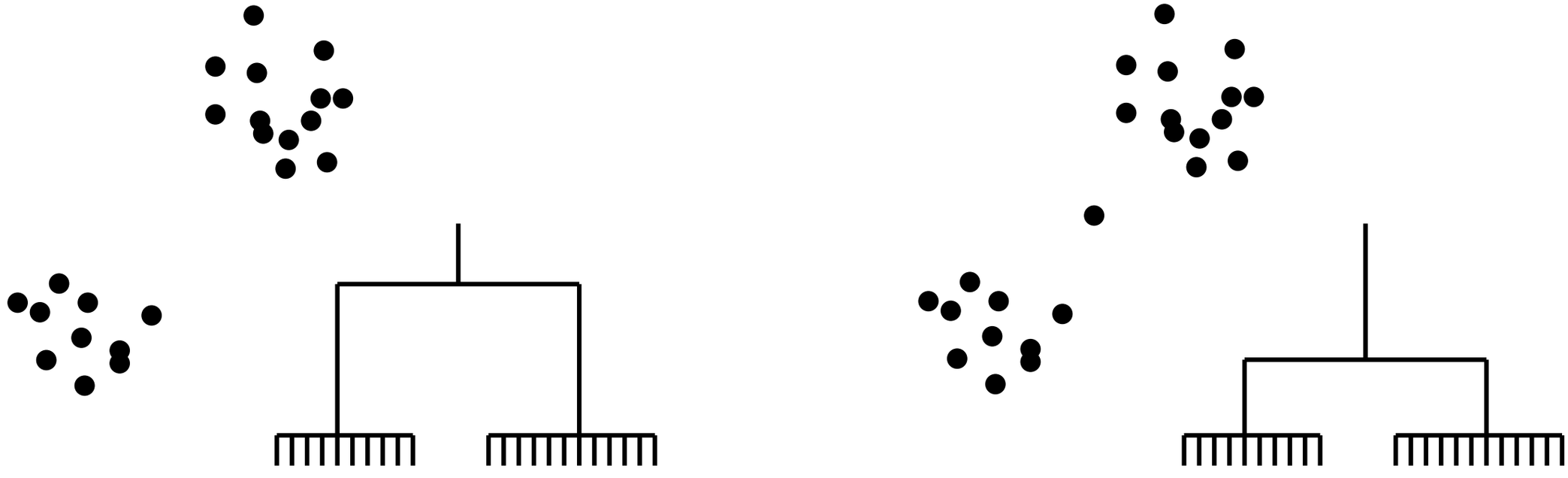
$\rightarrow$ 0-dimensional persistent homology provides a stable output!

The (in)stability of dendrograms

However, building a hierarchy based on spatial proximity is still not a great idea when there are outliers, since there is no stability of merging times anymore.

$\rightarrow$ 0-dimensional persistent homology provides a stable output!

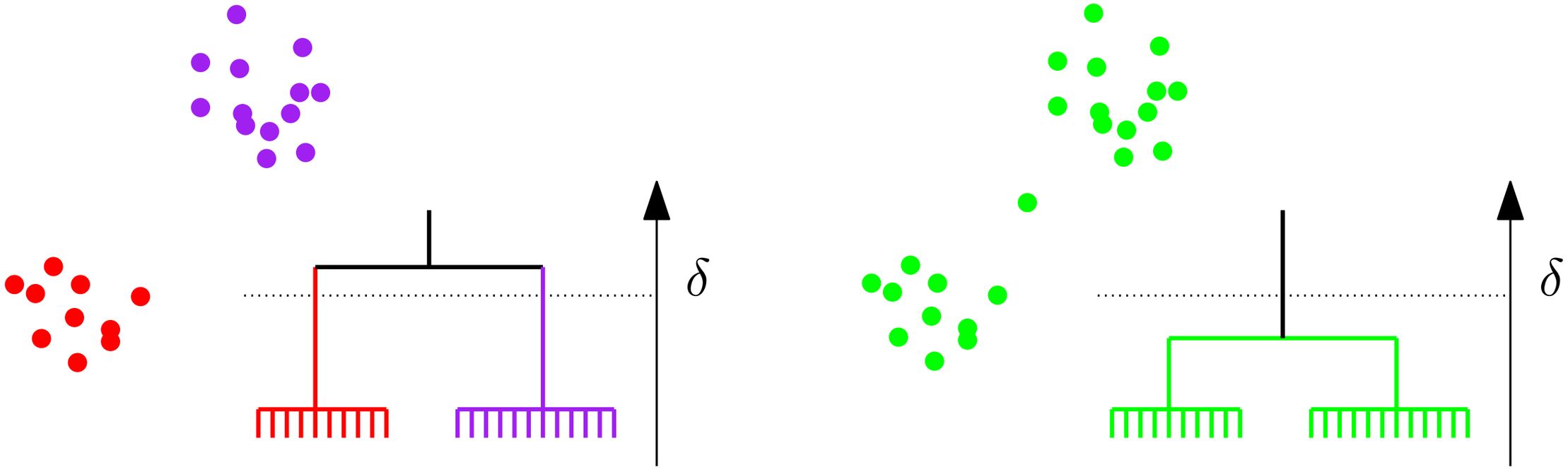
The (in)stability of dendrograms



However, building a hierarchy based on spatial proximity is still not a great idea when there are outliers, since there is no stability of merging times anymore.

$\rightarrow$ 0-dimensional persistent homology provides a stable output!

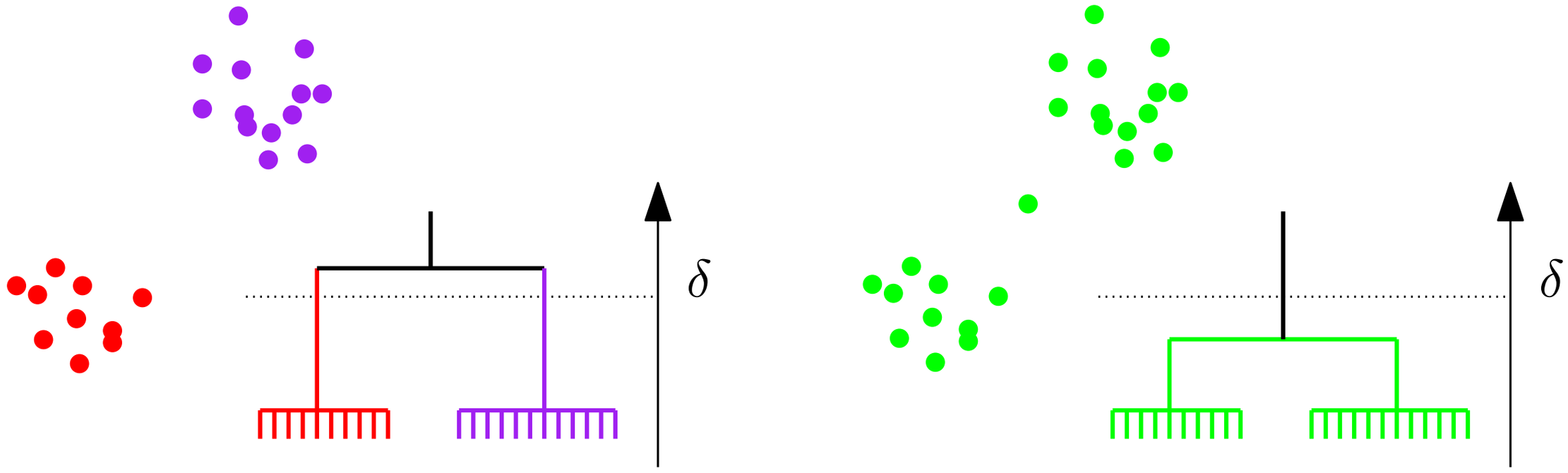
The (in)stability of dendrograms



However, building a hierarchy based on spatial proximity is still not a great idea when there are outliers, since there is no stability of merging times anymore.

$\rightarrow$ 0-dimensional persistent homology provides a stable output!

The (in)stability of dendrograms

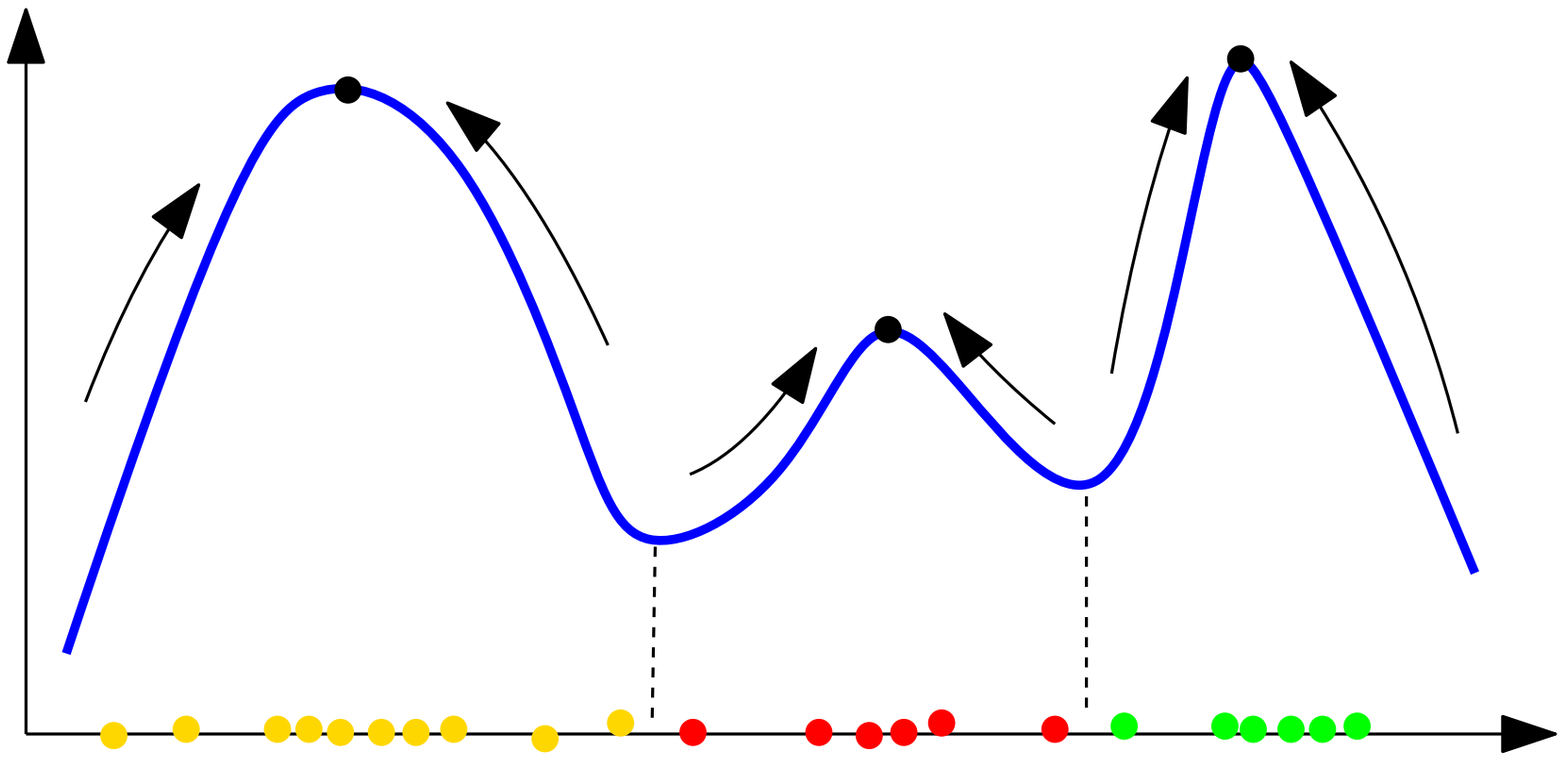


However, building a hierarchy based on spatial proximity is still not a great idea when there are outliers, since there is no stability of merging times anymore.

Another way to build a hierarchy is with the sublevel sets of a density function. Using density for clustering is at the core of mode-seeking algorithms.

$\rightarrow$ 0-dimensional persistent homology provides a stable output!

Mode seeking clustering



In mode seeking, data points are sampled according to some (unknown) probability density, and clusters are given with its basins of attraction.

Two approaches:

- **Iterative**, such as, e.g., Mean Shift. [*Mean shift: a robust approach toward feature space analysis*, Comaniciu et al., IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002]
- **Graph-based**, such as, e.g., [*A Graph-Theoretic Approach to Nonparametric Cluster Analysis*, Koontz et al., IEEE Trans. on Computers, 1976].

Mean Shift (2002)

Mean Shift (2002)

1. Pick random guess $x \in X$.

Mean Shift (2002)

1. Pick random guess $x \in X$.
2. Compute $$M(x) = \frac{\sum_{x_i \in N(x)} K(x,x_i) \cdot x_i}{\sum_{x_i \in N(x)} K(x,x_i)},$$

where $N(x)$ is a neighborhood of x, and K is a kernel, e.g., Gaussian kernel $K(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right)$.

Mean Shift (2002)

1. Pick random guess $x \in X$.
2. Compute $$M(x) = \frac{\sum_{x_i \in N(x)} K(x,x_i) \cdot x_i}{\sum_{x_i \in N(x)} K(x,x_i)},$$

where $N(x)$ is a neighborhood of x, and K is a kernel, e.g., Gaussian kernel $K(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right)$.

3. Update $x \leftarrow M(x)$.

Mean Shift (2002)

1. Pick random guess $x \in X$.
2. Compute $$M(x) = \frac{\sum_{x_i \in N(x)} K(x,x_i) \cdot x_i}{\sum_{x_i \in N(x)} K(x,x_i)},$$

where $N(x)$ is a neighborhood of x, and K is a kernel, e.g., Gaussian kernel $K(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right)$.

3. Update $x \leftarrow M(x)$.

Do that for many random guesses, postprocess and merge similar centroids, and use the distances to the centroids to decide clusters.

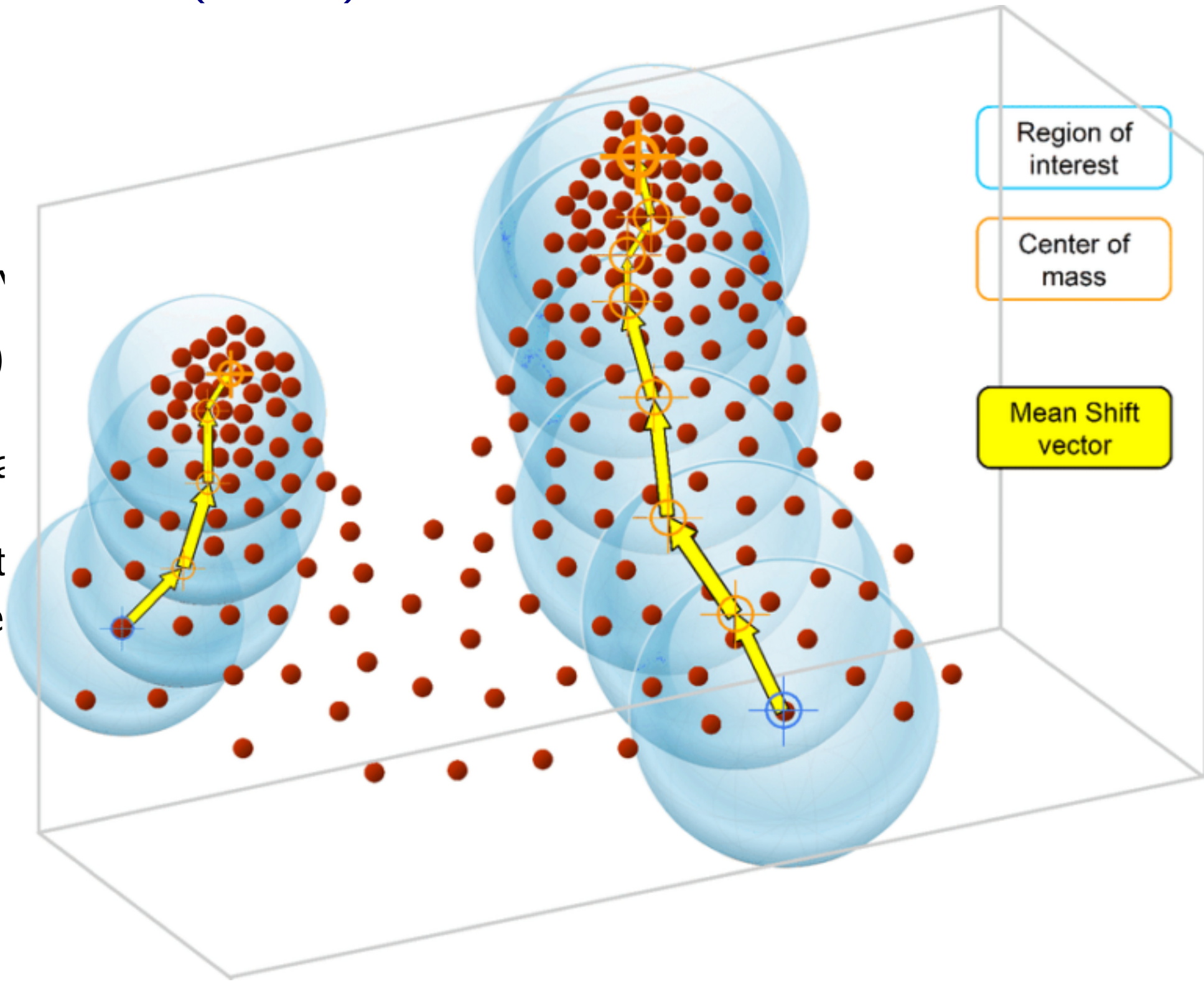
Mean Shift (2002)

1. Pick
2. Com

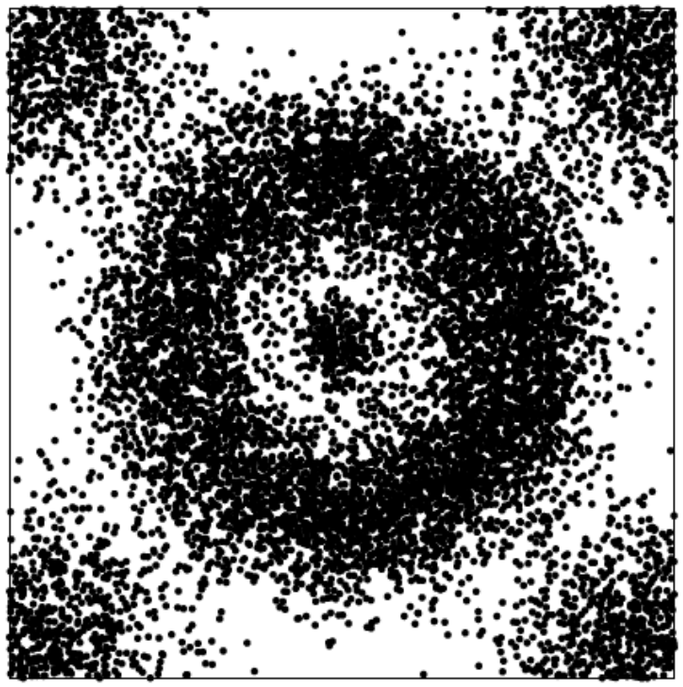
where M n kernel $K(x, y)$

3. Upda

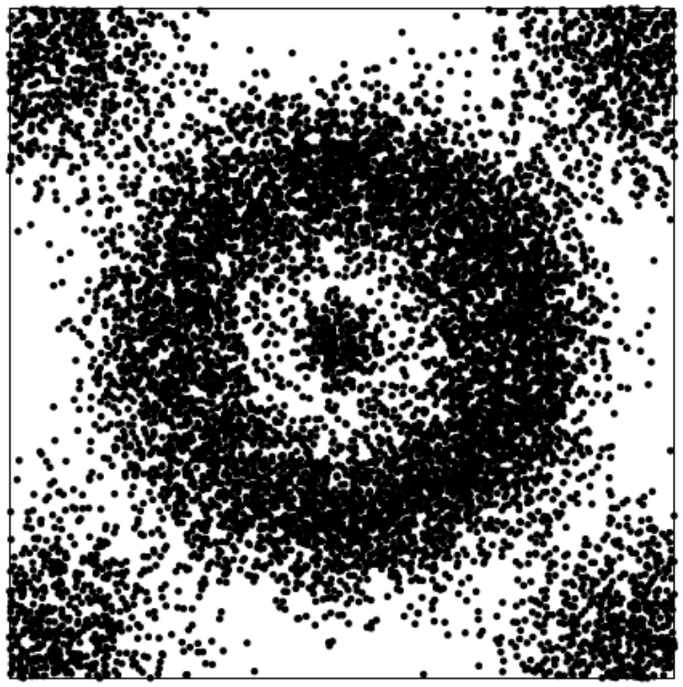
Do that ntroids, and use



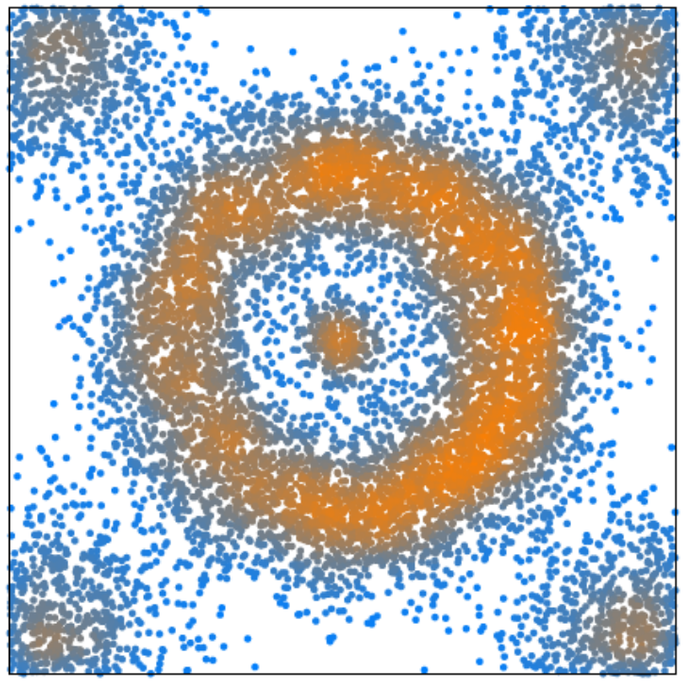
The Koonz, Narendra and Fukunaga algorithm (1976)



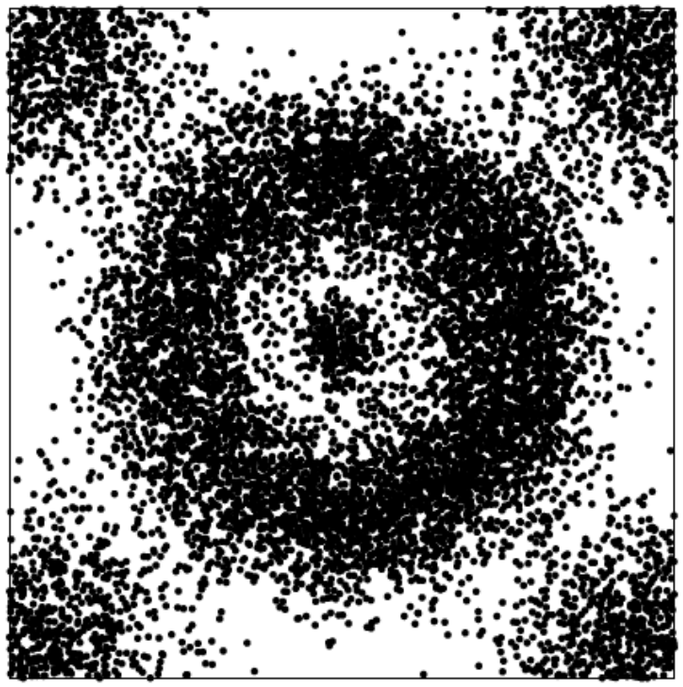
The Koonz, Narendra and Fukunaga algorithm (1976)



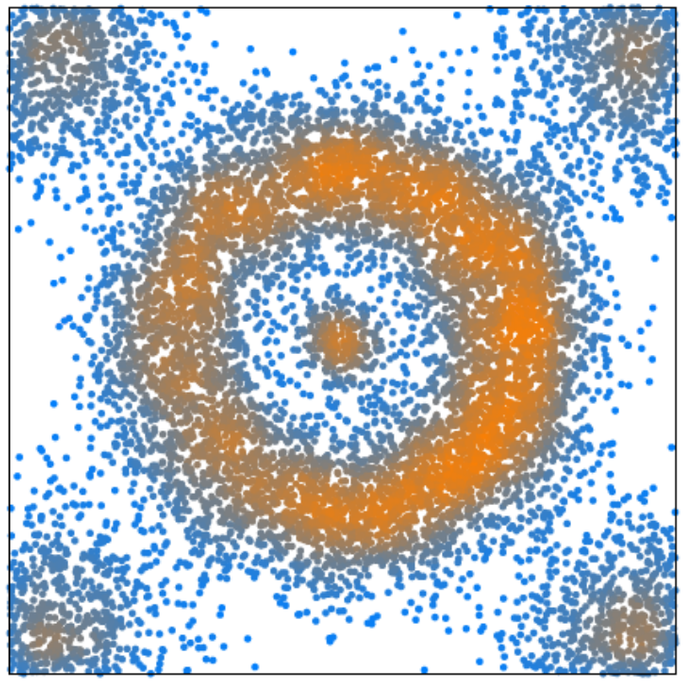
Density estimation



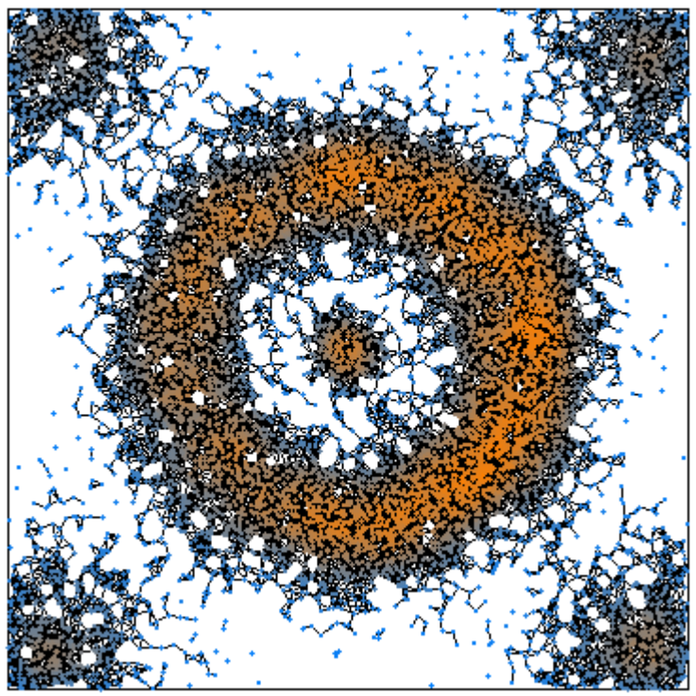
The Koonz, Narendra and Fukunaga algorithm (1976)



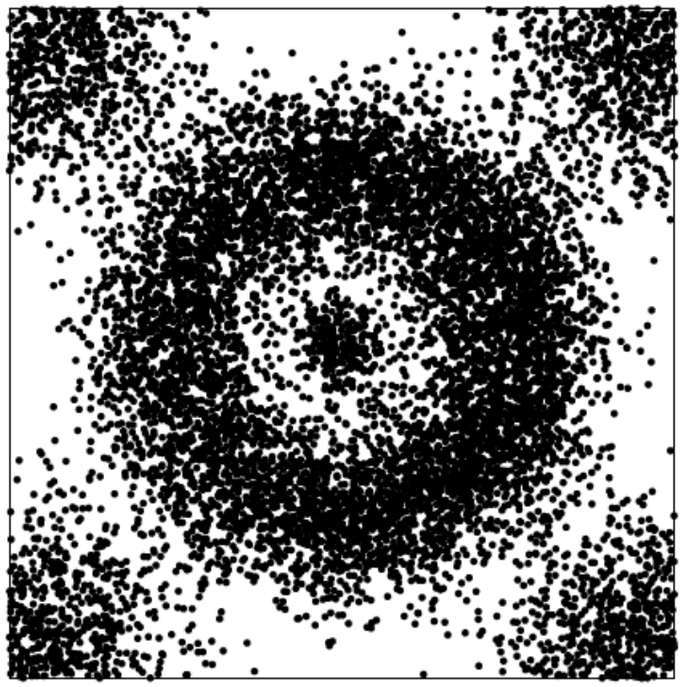
Density estimation



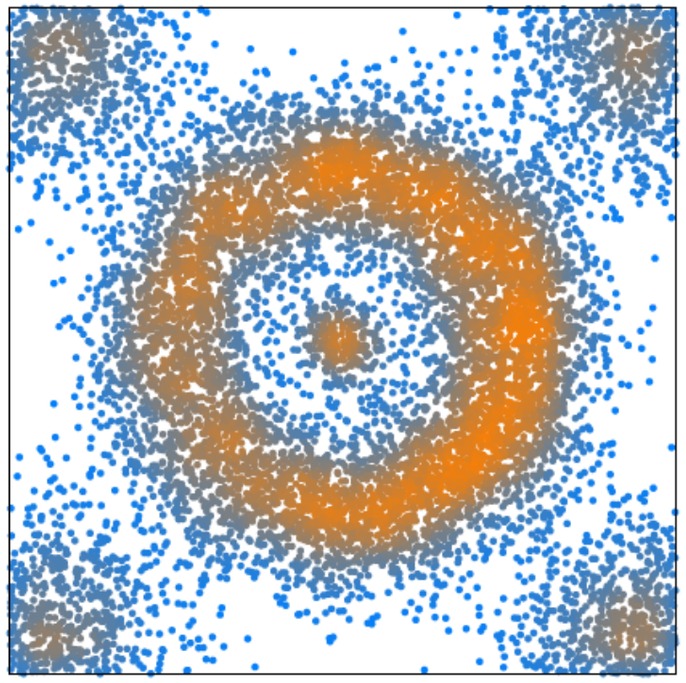
Neighborhood graph



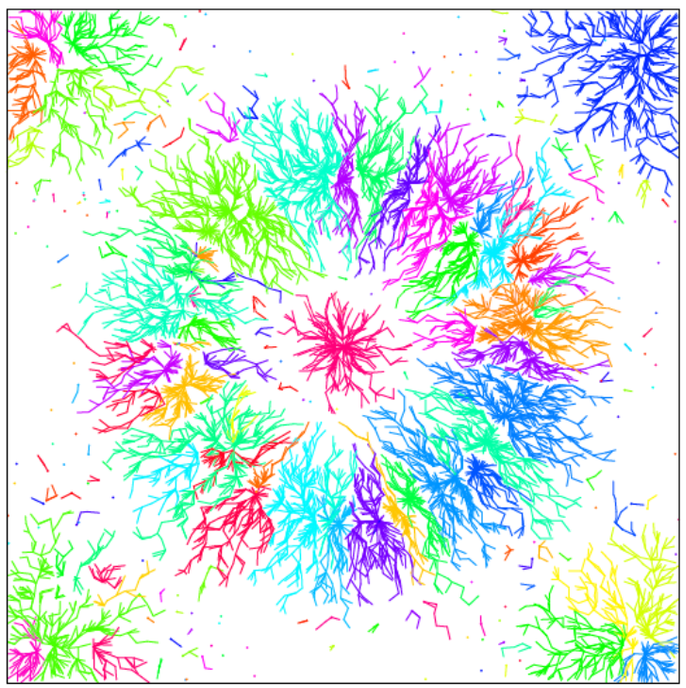
The Koonz, Narendra and Fukunaga algorithm (1976)



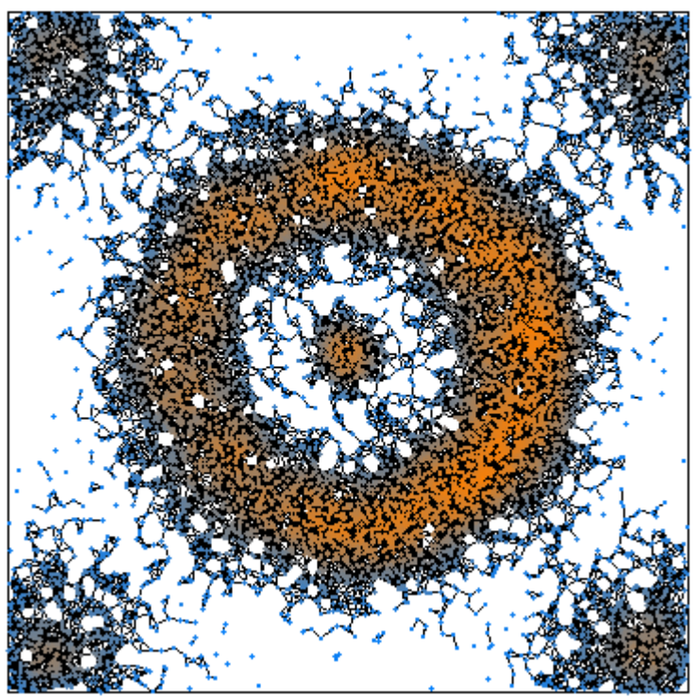
Density estimation



Neighborhood graph



Discrete approximation of the gradient; for each vertex v, a gradient edge is selected among the edges adjacent to v.



The Koonz, Narendra and Fukunaga algorithm (1976)

The algorithm:

Input: A neighborhood graph G with n vertices (the data points) and an n-dimensional vector $\hat{f}$ (density estimate).

Sort the vertex indices $\{1, 2, \ldots, n\}$ in decreasing order: $\hat{f}(1) \geq \cdots \geq \hat{f}(n)$.

Initialize a union-find data structure $\mathcal{U}$ and two lists g, r of length n.

`for` $i \in \{1, \ldots, n\}$:

  Let $\mathcal{N}$ be the set of neighbors of i in G that have indices lower than i

  `if` $\mathcal{N} = \varnothing$:

    Create a new entry e in $\mathcal{U}$ and attach vertex i to it: $\mathcal{U}$.`MakeSet`(i)

    $r[e] \leftarrow i$ *($r[e]$ stores the root vertex associated with the entry e)*

  `else`:

    $g[i] \leftarrow \operatorname{argmax}\{\hat{f}(j) : j \in \mathcal{N}\}$ *($g[i]$ stores the approximate gradient at vertex i)*

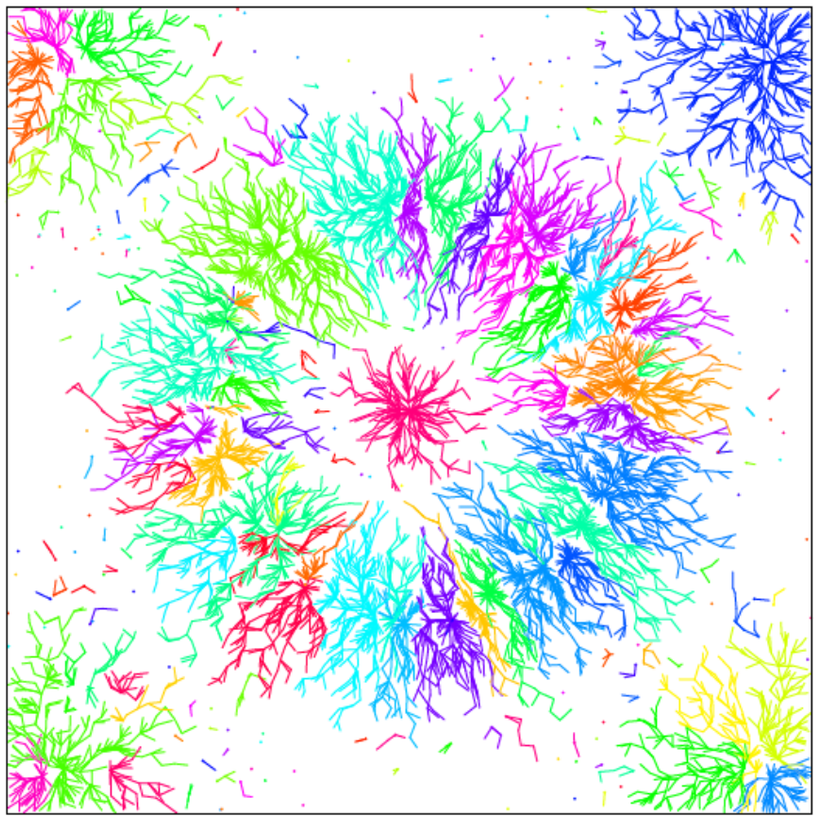
    $e_i \leftarrow \mathcal{U}.\texttt{Find}(g[i])$

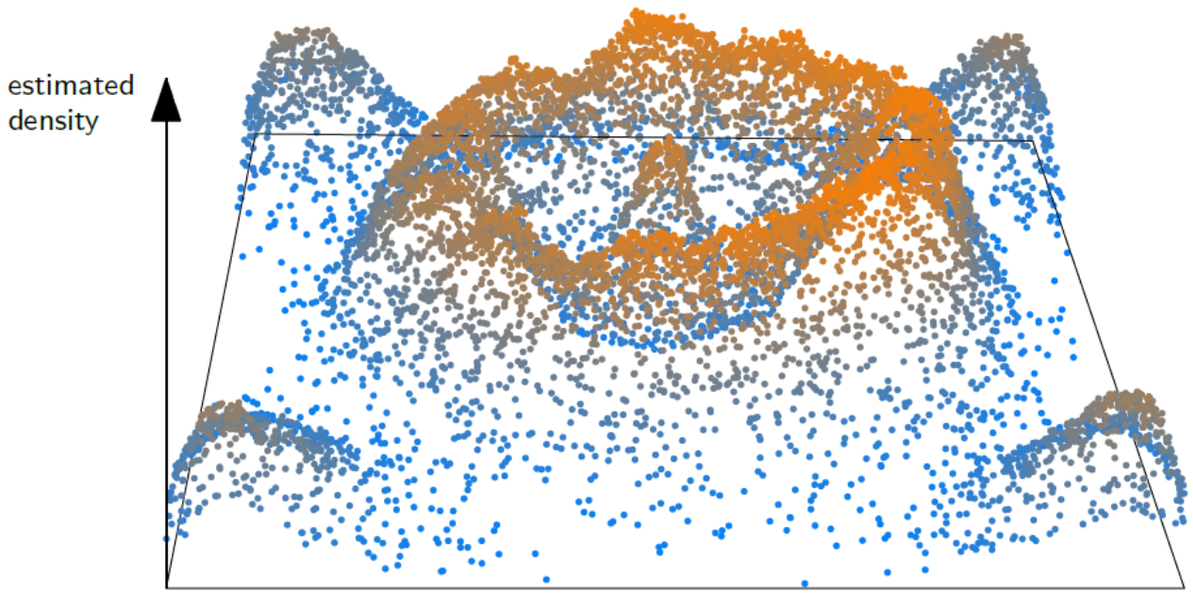
    Attach vertex i to the entry e_i: $\mathcal{U}.\texttt{Union}(i, e_i)$

Output: The collection of entries e in $\mathcal{U}$.

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

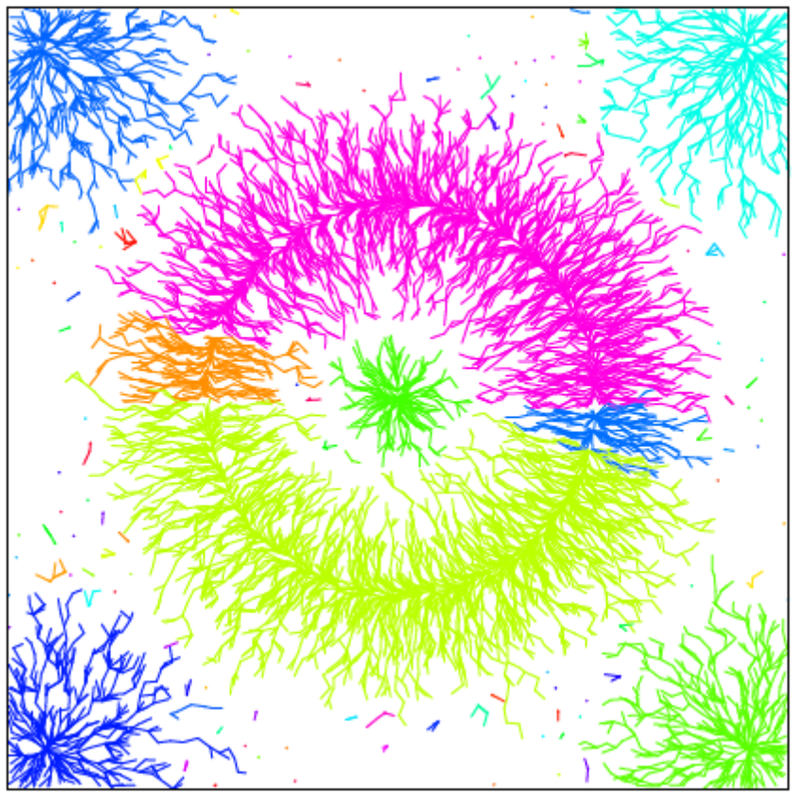


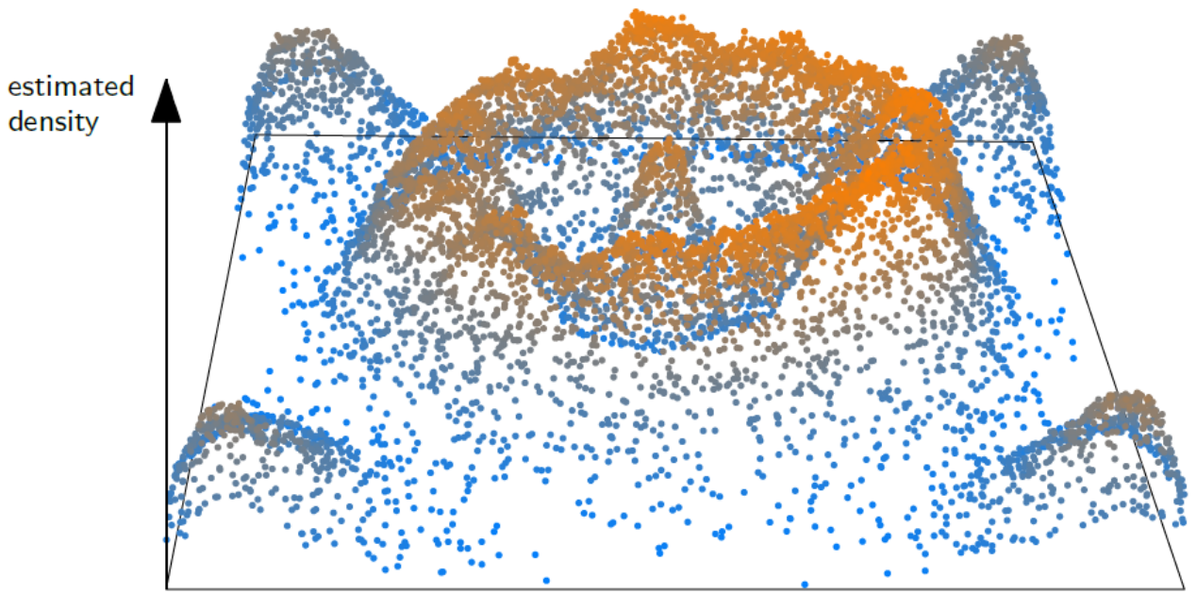


One has as many clusters as local maxima of the density estimate, which are very sensitive to noise and outliers.

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:





One has as many clusters as local maxima of the density estimate, which are very sensitive to noise and outliers.

The choice of the neighborhood graph (k-nearest neighbors, triangulations, etc) may result in wide changes in the output.

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

One has as many clusters as local maxima of the density estimate, which are very sensitive to noise and outliers.

The choice of the neighborhood graph (k-nearest neighbors, triangulations, etc) may result in wide changes in the output.

Approaches to overcome these issues:

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

One has as many clusters as local maxima of the density estimate, which are very sensitive to noise and outliers.

The choice of the neighborhood graph (k-nearest neighbors, triangulations, etc) may result in wide changes in the output.

Approaches to overcome these issues:

One can smooth out the density estimate, but smoothing is usually data-driven and hard to tune.

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

One has as many clusters as local maxima of the density estimate, which are very sensitive to noise and outliers.

The choice of the neighborhood graph (k-nearest neighbors, triangulations, etc) may result in wide changes in the output.

Approaches to overcome these issues:

One can smooth out the density estimate, but smoothing is usually data-driven and hard to tune.

Build a hierarchy of clusters with 0-dimensional persistent homology!

I. ToMATo algorithm

1. Introduction to hierarchical and mode-seeking clustering

2. **ToMATo algorithm and guarantees**

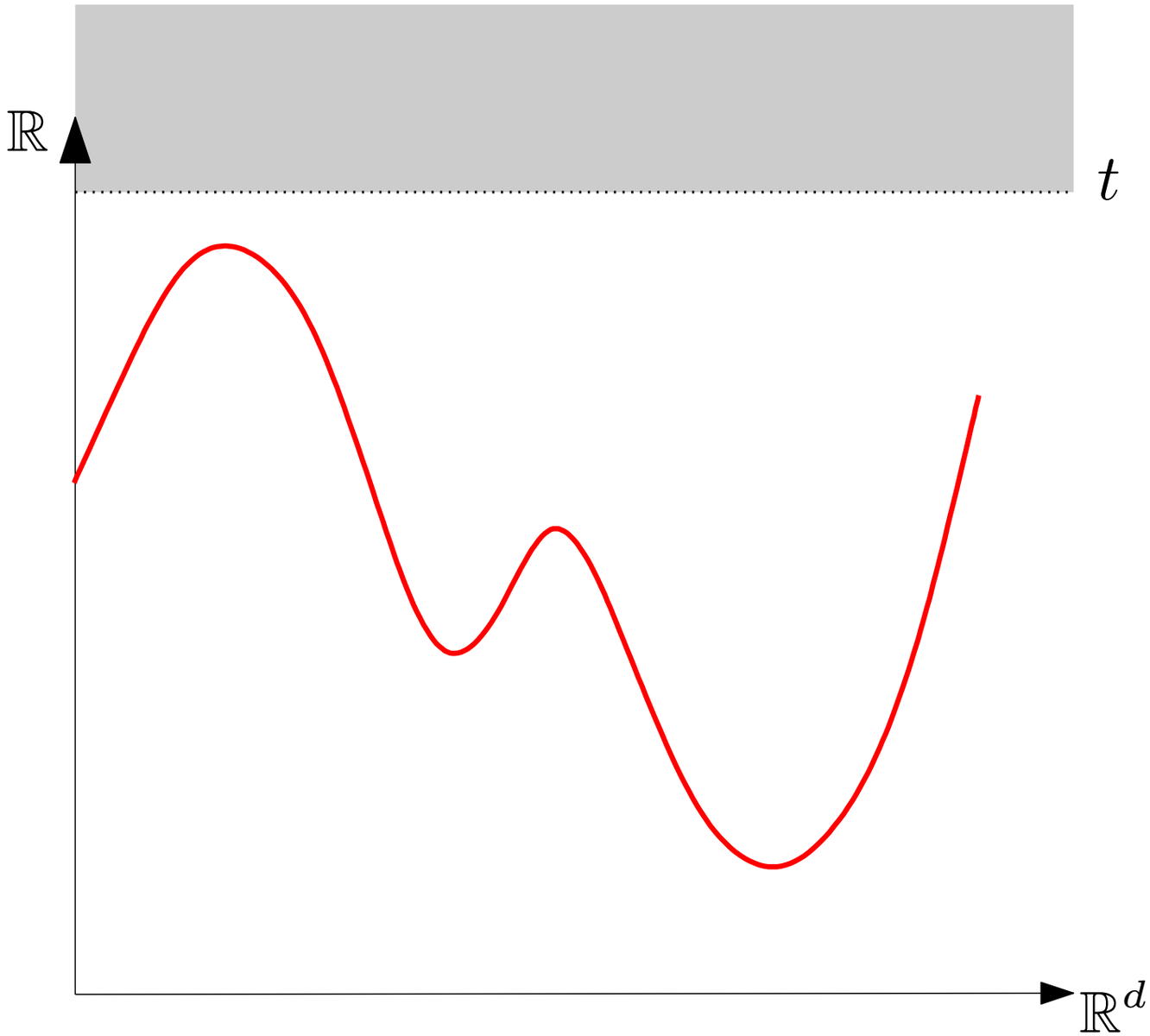
II. Mapper algorithm

1. Reeb spaces and Mappers

2. Confidence intervals

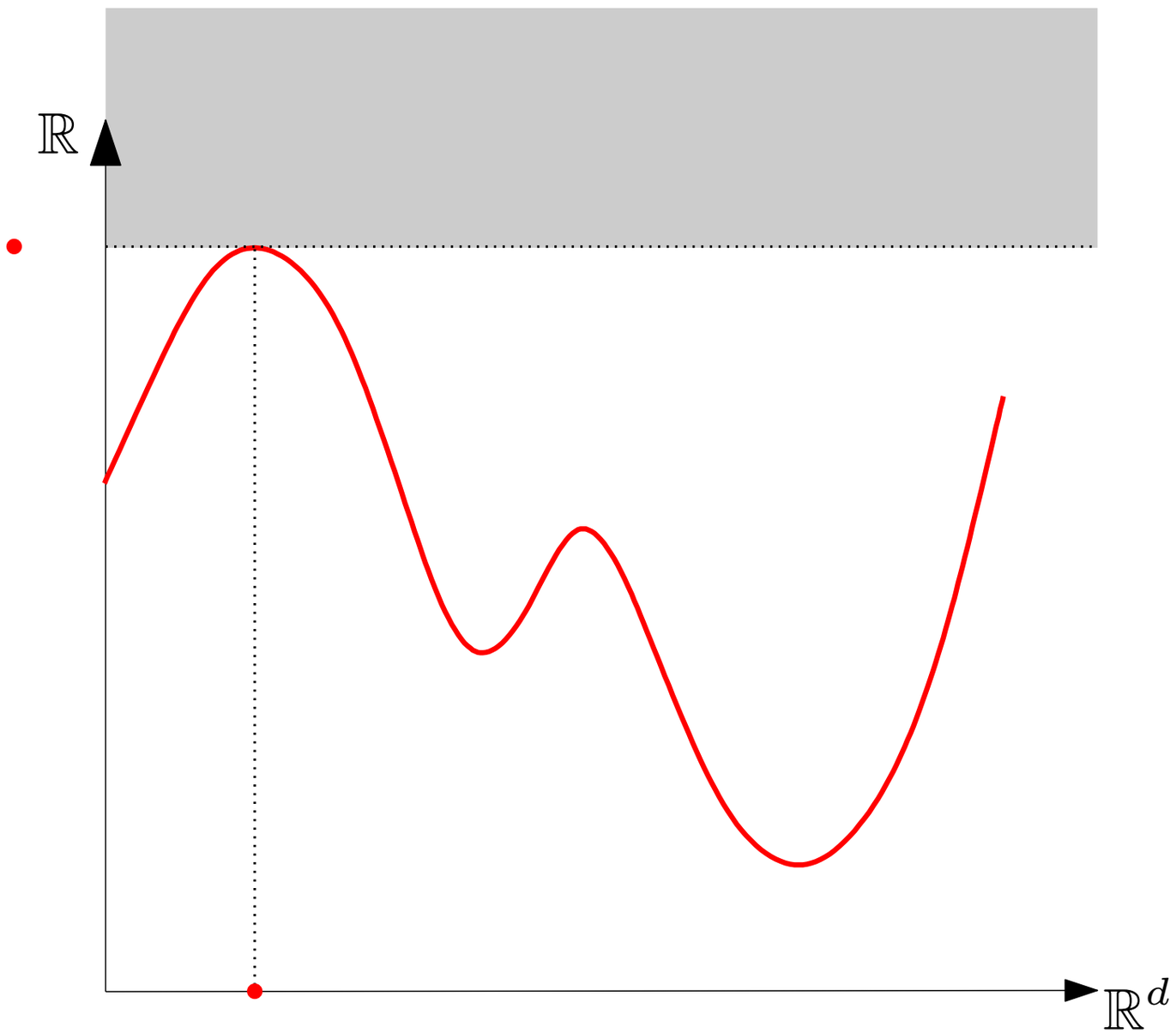
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



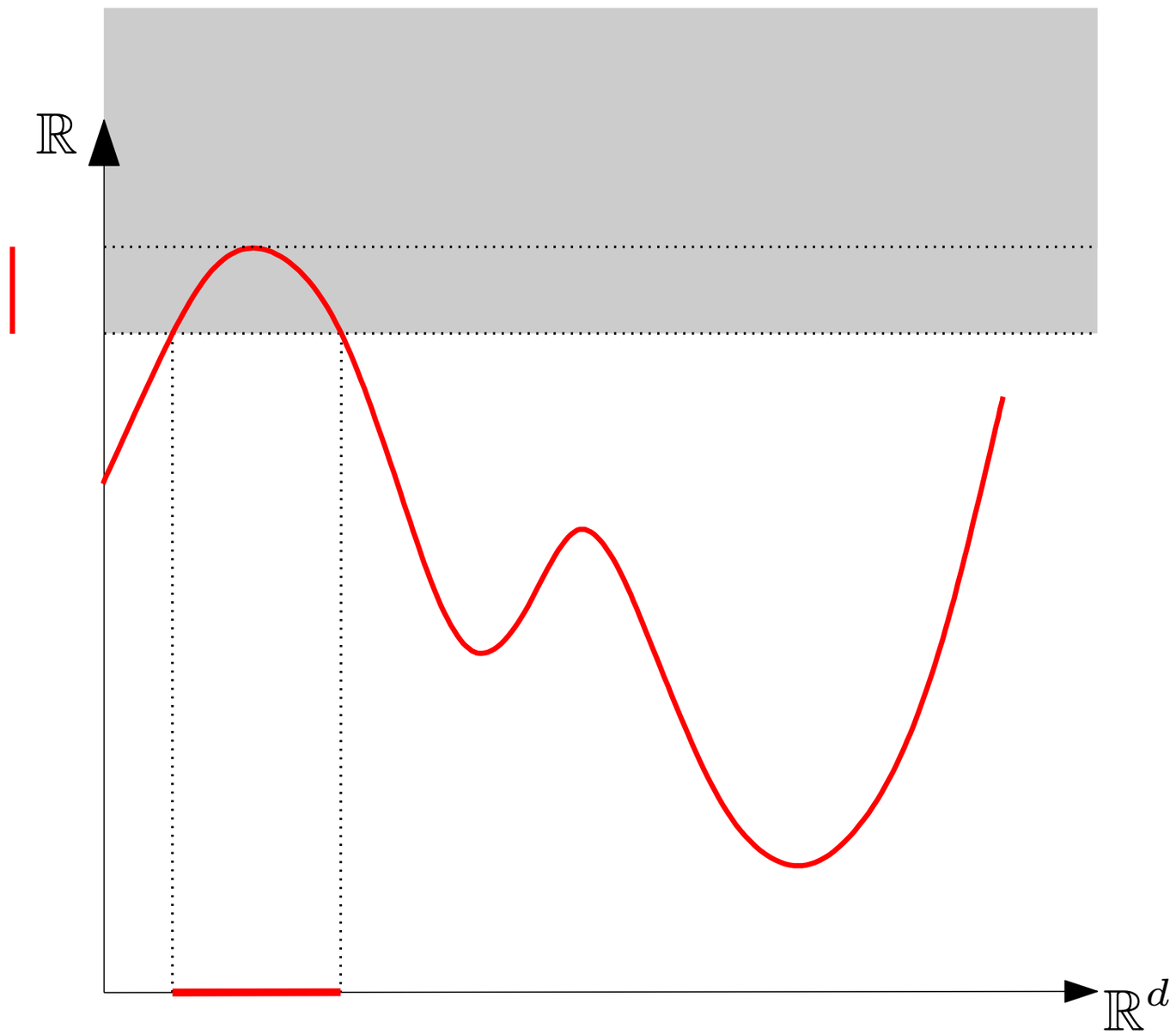
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



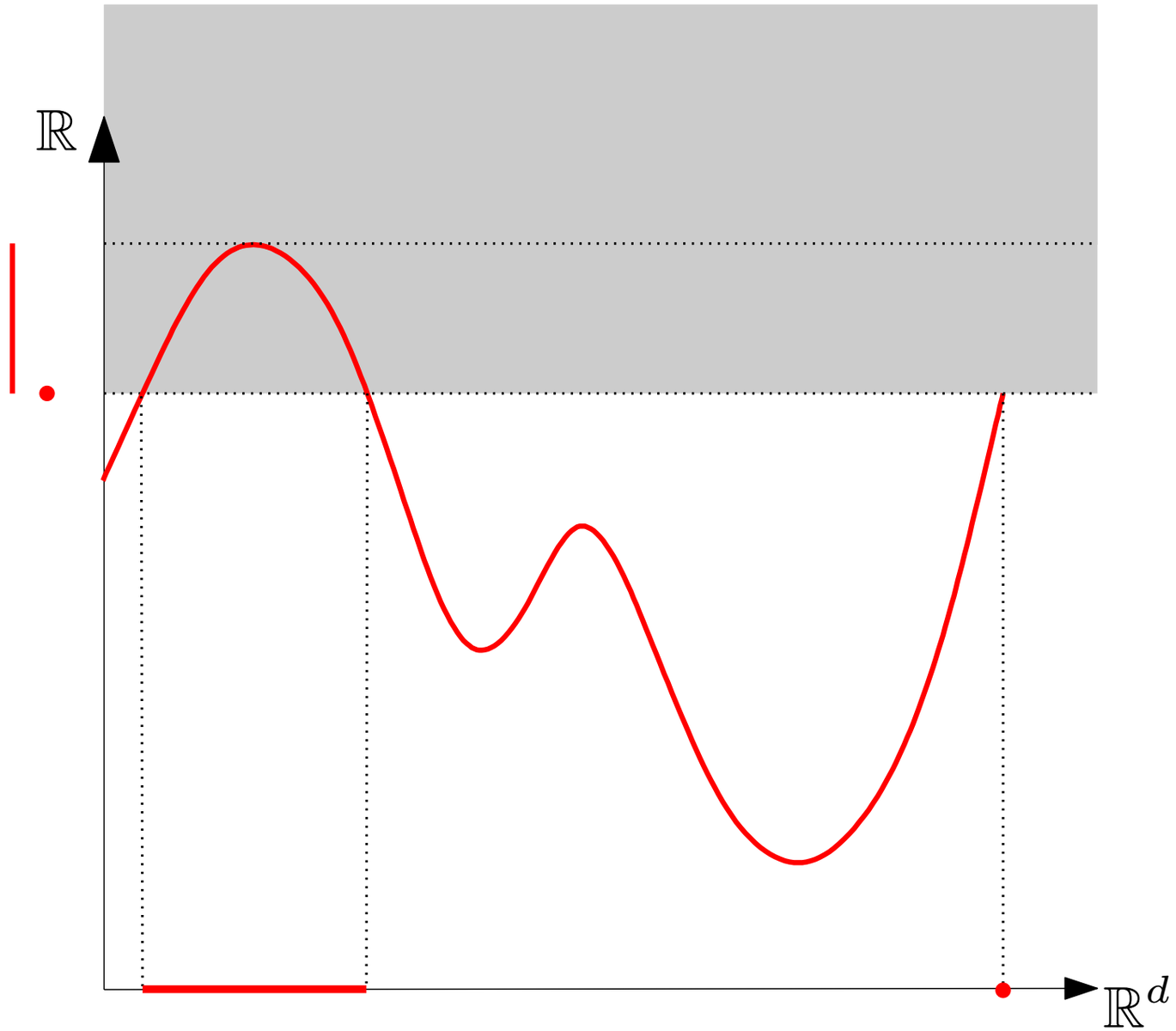
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



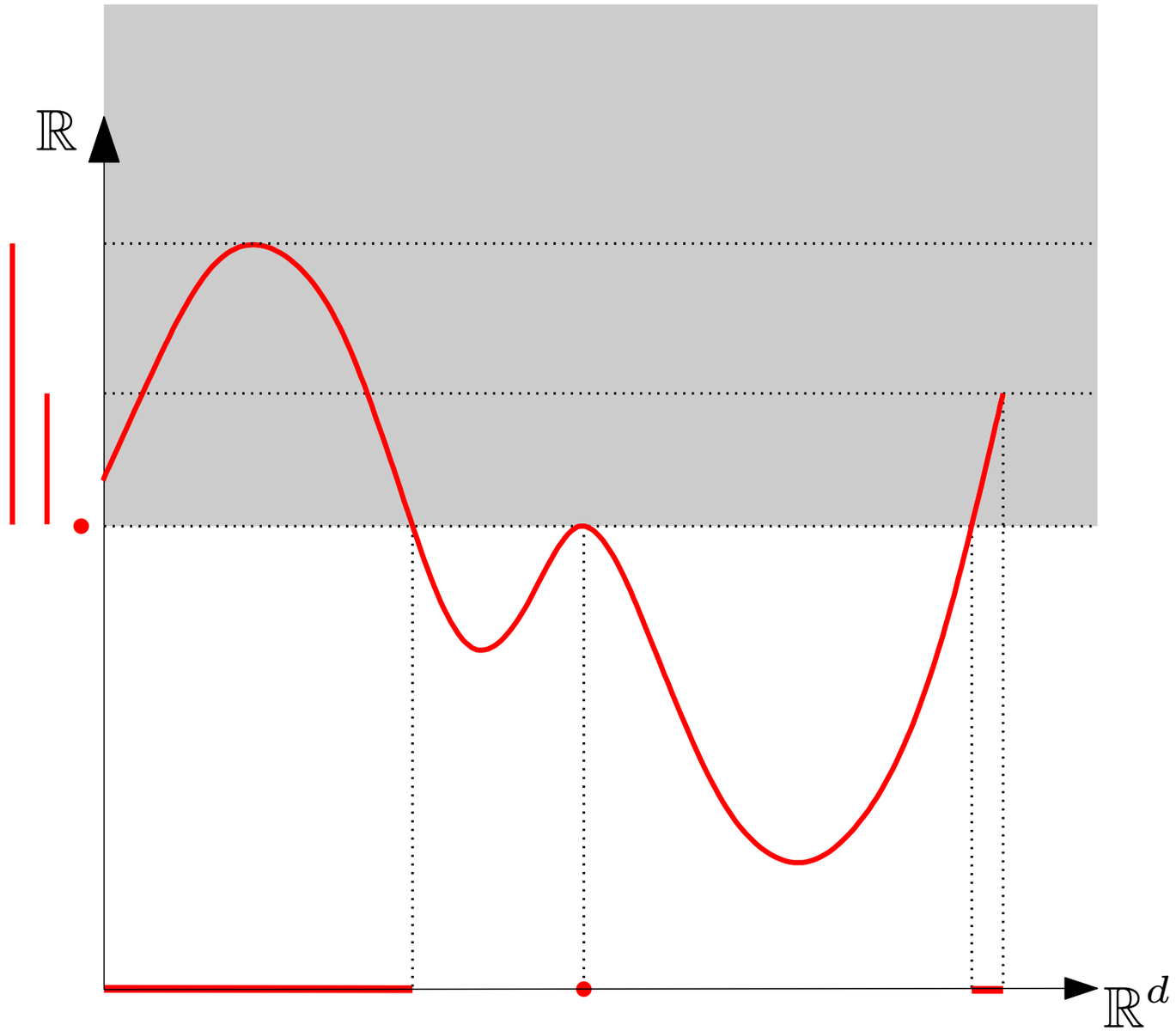
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



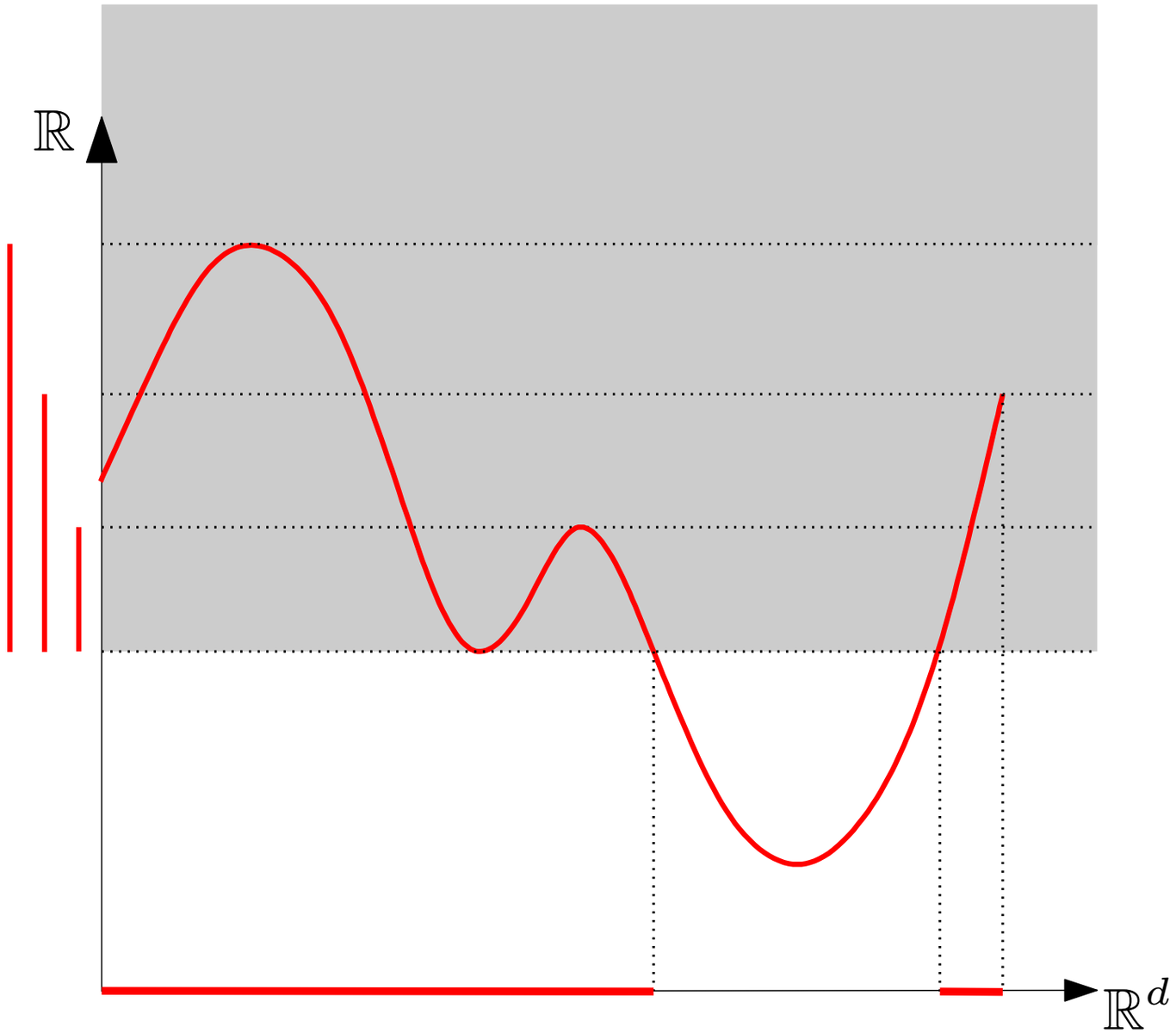
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



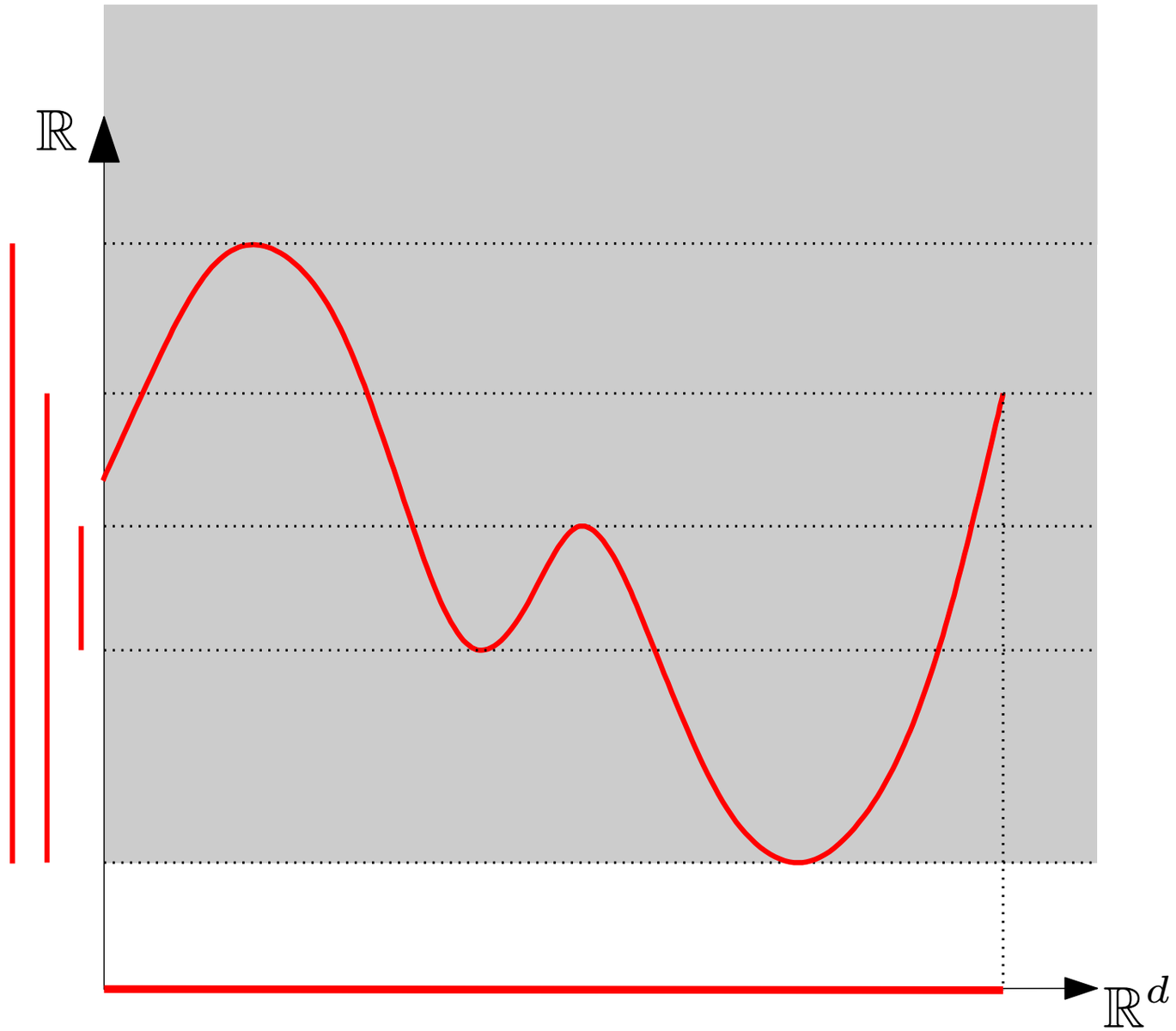
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



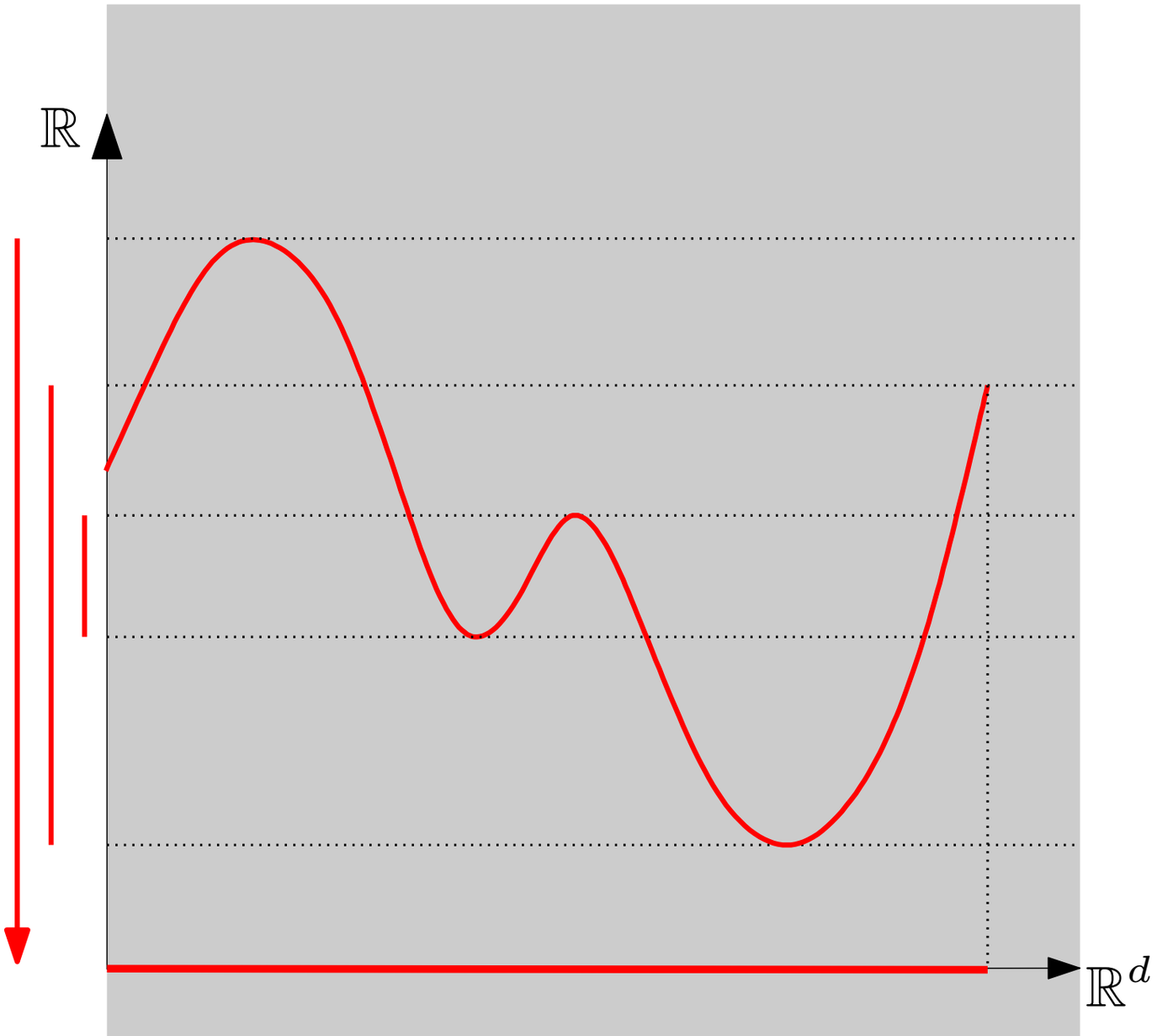
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



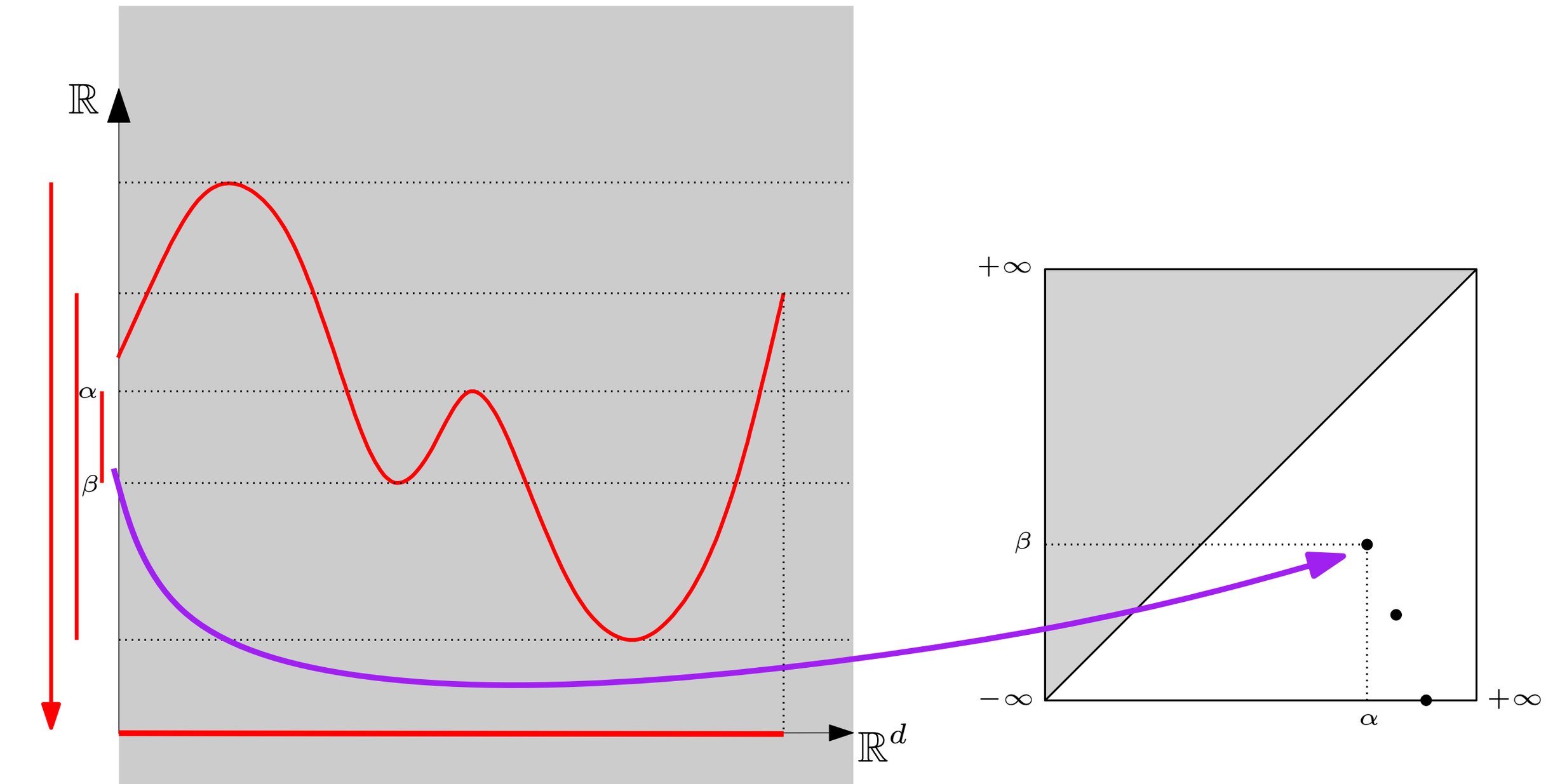
Reminder: 0-dimensional PH of density

Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.



Reminder: 0-dimensional PH of density

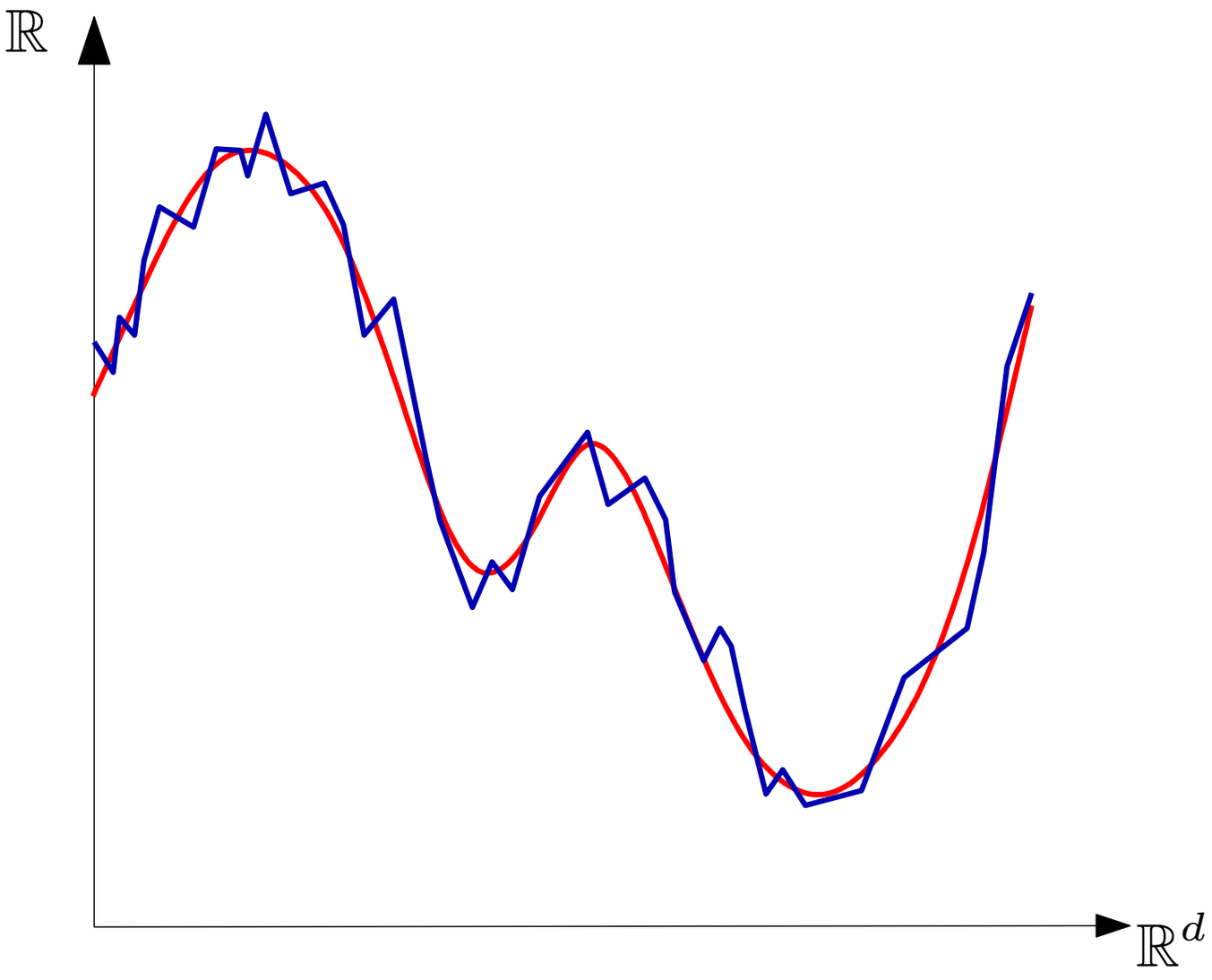
Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.

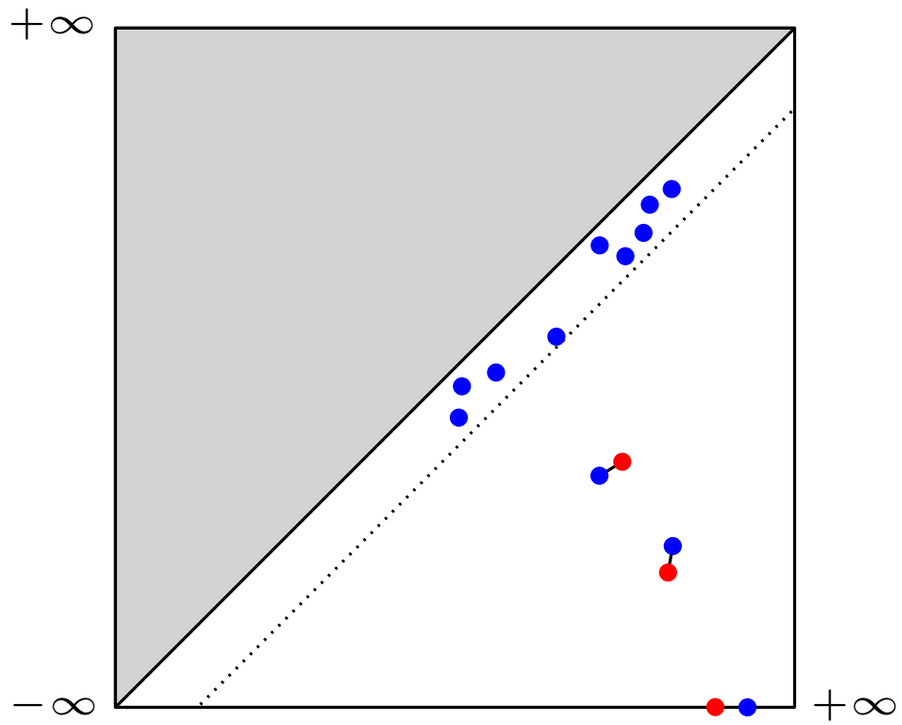


Reminder: 0-dimensional PH of density

Moreover, the stability theorem ensures that, given an underlying true density f, and an estimator $\hat{f}$ ot it, one has:

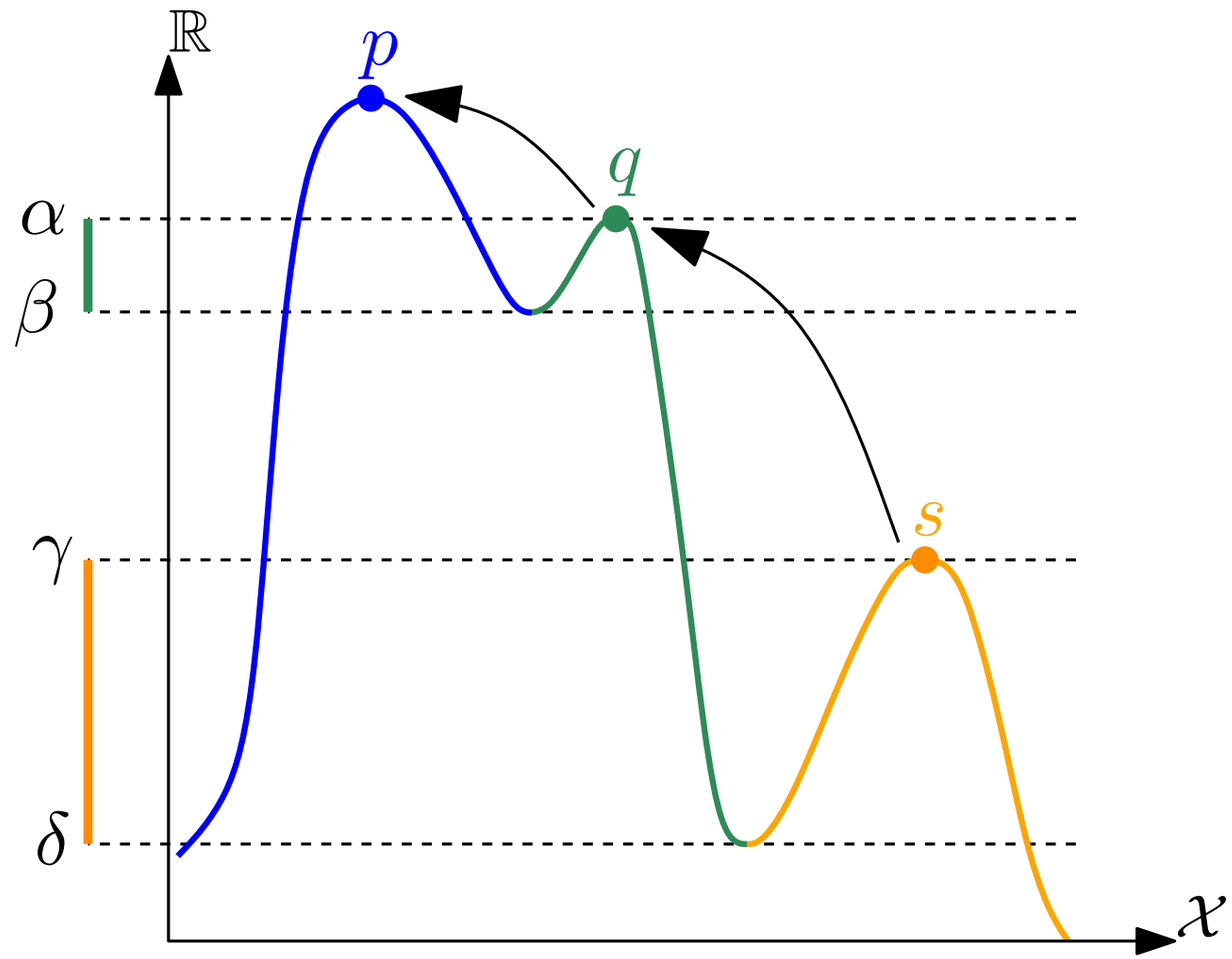
$$d_b(D_f, D_{\hat{f}}) \leq \|f - \hat{f}\|_\infty.$$





Building a hierarchy of cluster with 0-dimensional PH

In addition to being stable, 0-dimensional PH also remembers the connected components that were merged together during the filtration process and builds a hierarchy out of this information.

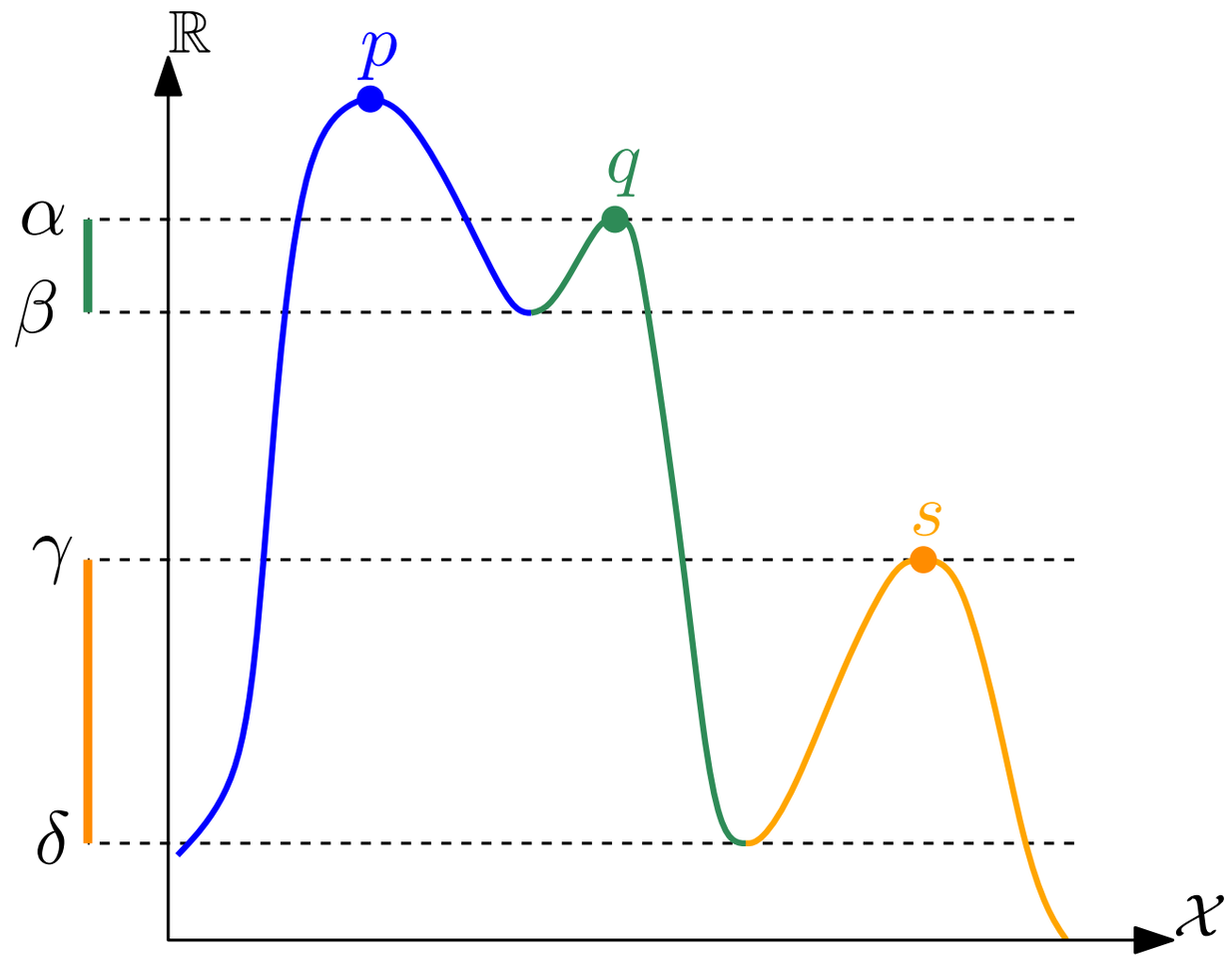


Building a hierarchy of cluster with 0-dimensional PH

In addition to being stable, 0-dimensional PH also remembers the connected components that were merged together during the filtration process and builds a hierarchy out of this information.

This means that, given a fixed threshold $\tau \geq 0$, one can even retrieve the clusters associated to all the bars of length (or prominence) $> \tau$!

$$0 \leq \tau \leq \alpha - \beta$$

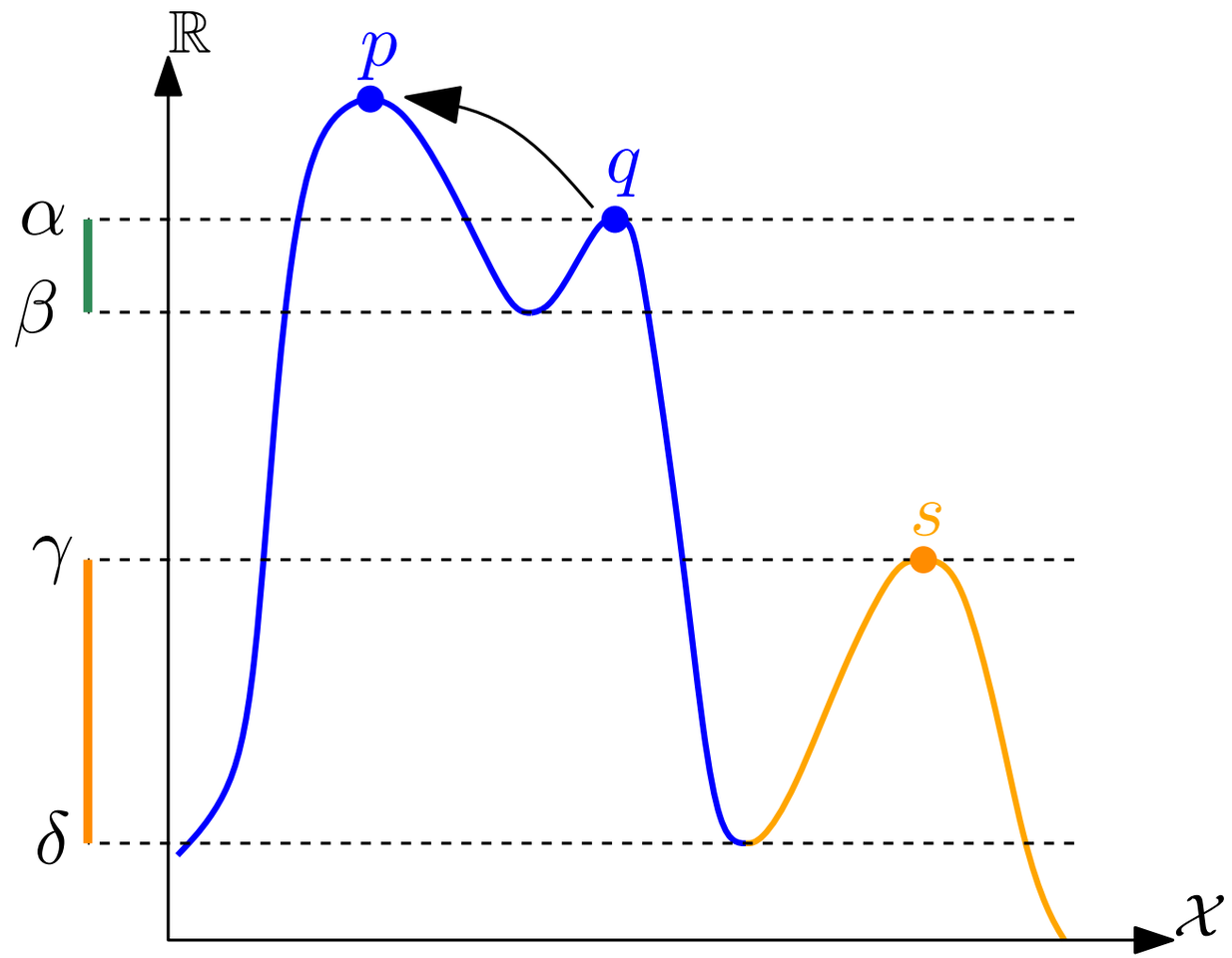


Building a hierarchy of cluster with 0-dimensional PH

In addition to being stable, 0-dimensional PH also remembers the connected components that were merged together during the filtration process and builds a hierarchy out of this information.

This means that, given a fixed threshold $\tau \geq 0$, one can even retrieve the clusters associated to all the bars of length (or prominence) $> \tau$!

$$\alpha - \beta < \tau \leq \gamma - \delta$$

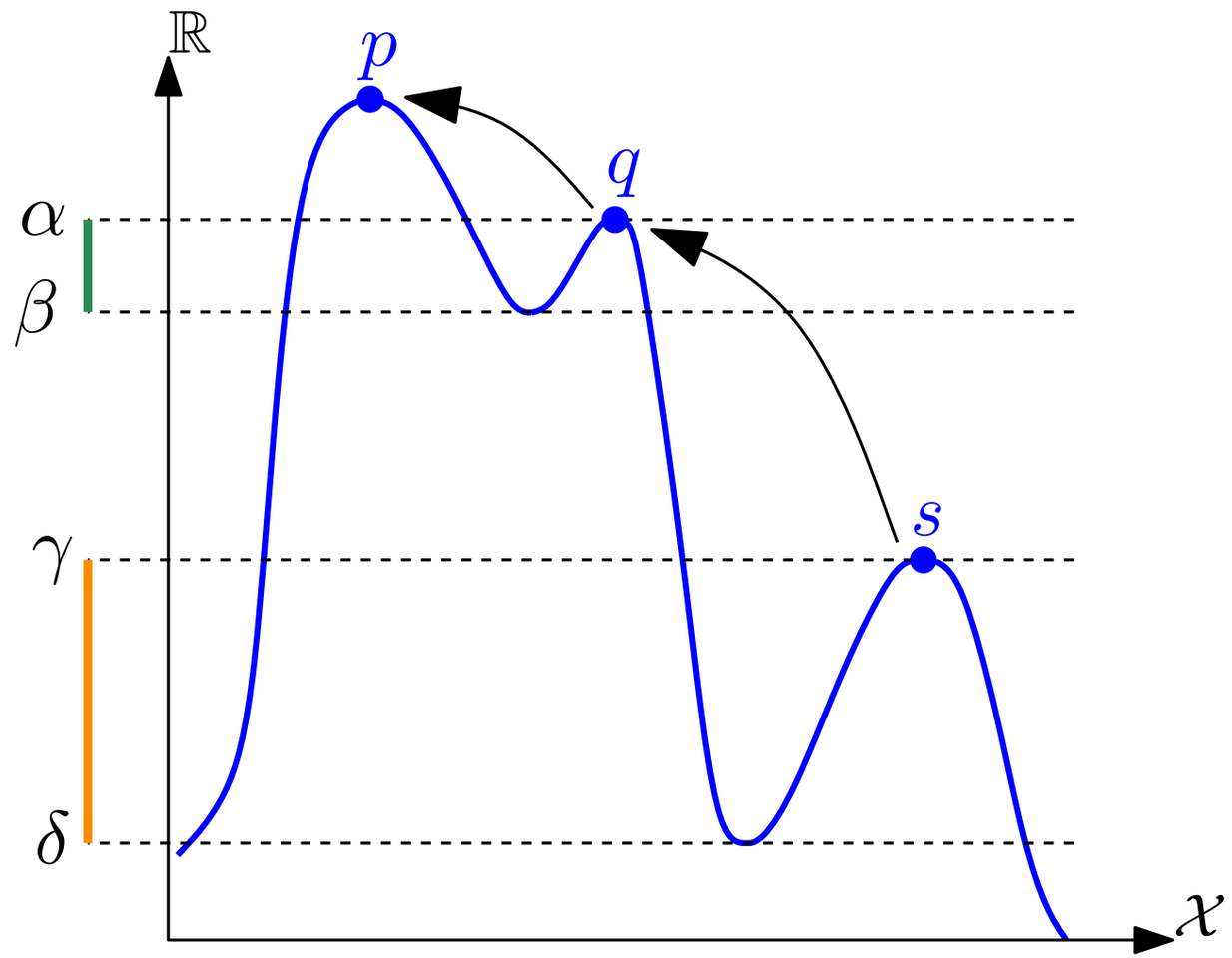


Building a hierarchy of cluster with 0-dimensional PH

In addition to being stable, 0-dimensional PH also remembers the connected components that were merged together during the filtration process and builds a hierarchy out of this information.

This means that, given a fixed threshold $\tau \geq 0$, one can even retrieve the clusters associated to all the bars of length (or prominence) $> \tau$!

$$\gamma - \delta < \tau \leq +\infty$$

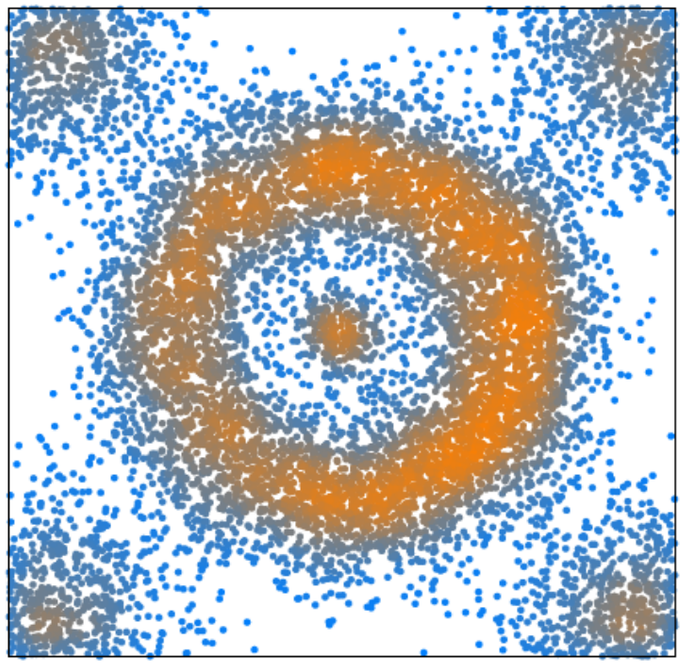


ToMATo: Topological Mode Analysis Tool

[*Persistence-Based Clustering in Riemannian Manifolds*, Chazal, Oudot, Skraba, Guibas, J. ACM, 2013]

1. Define an order on the point cloud with a density estimator $\hat{f}$.

   (sort data points by **decreasing** estimated density values)



ToMATo: Topological Mode Analysis Tool

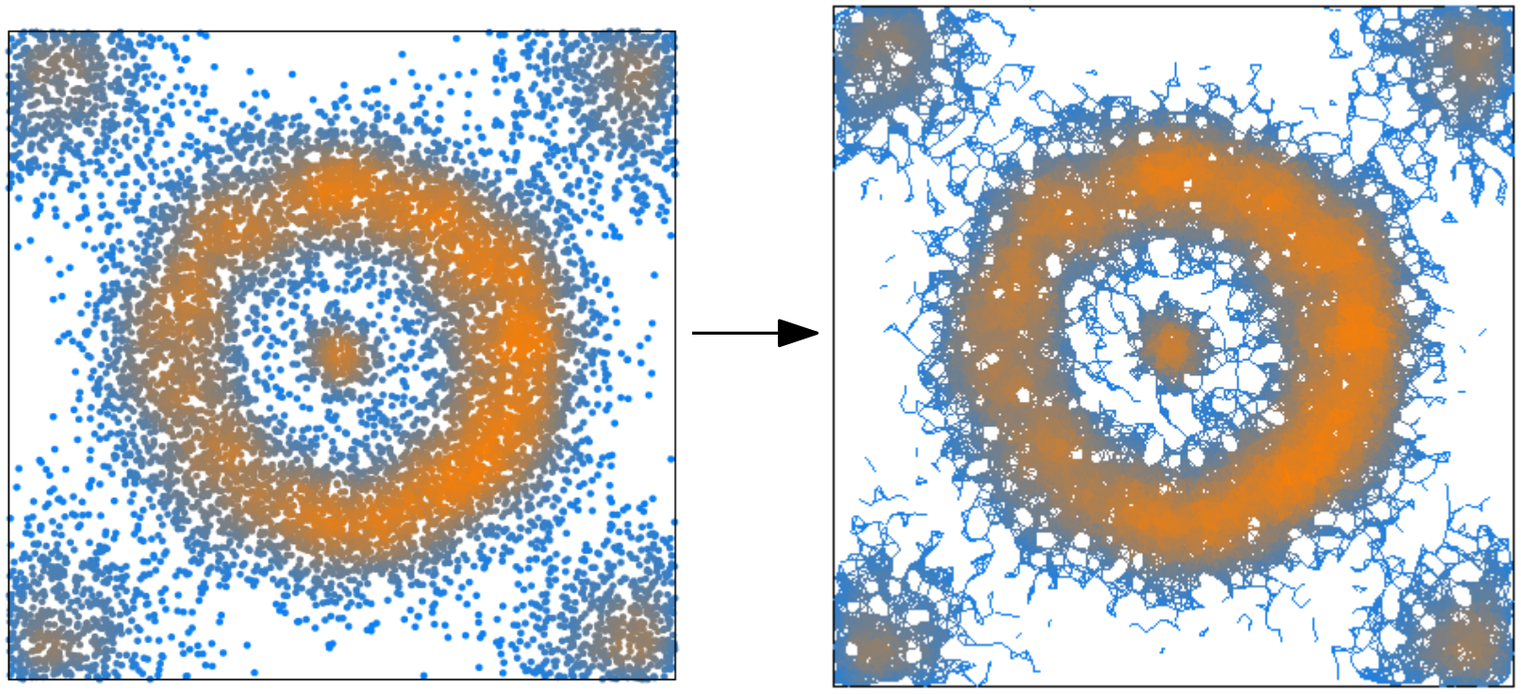
[*Persistence-Based Clustering in Riemannian Manifolds*, Chazal, Oudot, Skraba, Guibas, J. ACM, 2013]

1. Define an order on the point cloud with a density estimator $\hat{f}$.

   (sort data points by **decreasing** estimated density values)

2. Extend order to the graph edges (i.e., compute the *upper-star filtration*).

   ($\hat{f}([u,v]) = \min\{\hat{f}(u),\ \hat{f}(v)\}$)



ToMATo: Topological Mode Analysis Tool

[*Persistence-Based Clustering in Riemannian Manifolds*, Chazal, Oudot, Skraba, Guibas, J. ACM, 2013]

1. Define an order on the point cloud with a density estimator $\hat{f}$.

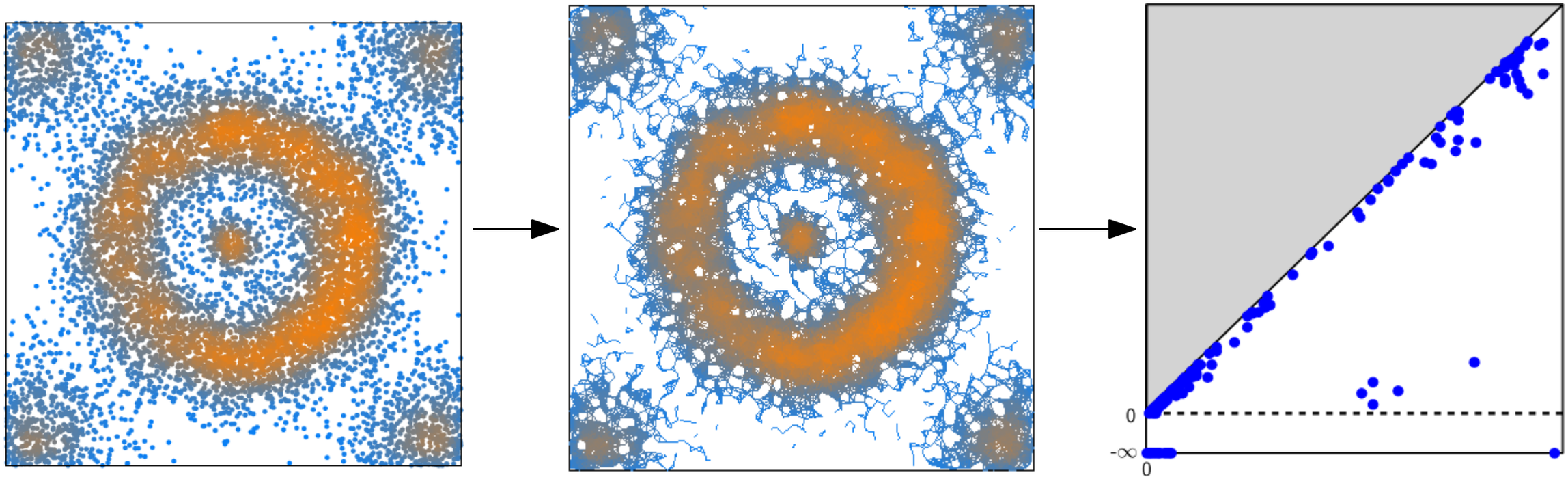
   (sort data points by **decreasing** estimated density values)

2. Extend order to the graph edges (i.e., compute the *upper-star filtration*).

   ($\hat{f}([u,v]) = \min\{\hat{f}(u),\ \hat{f}(v)\}$)

3. Compute the 0-dimensional persistence diagram of this filtration.

   (apply 0-dimensional persistence algorithm $\rightarrow$ union-find data structure)



ToMATo: Topological Mode Analysis Tool

[*Persistence-Based Clustering in Riemannian Manifolds*, Chazal, Oudot, Skraba, Guibas, J. ACM, 2013]

Given a neighborhood graph with n vertices and m edges:

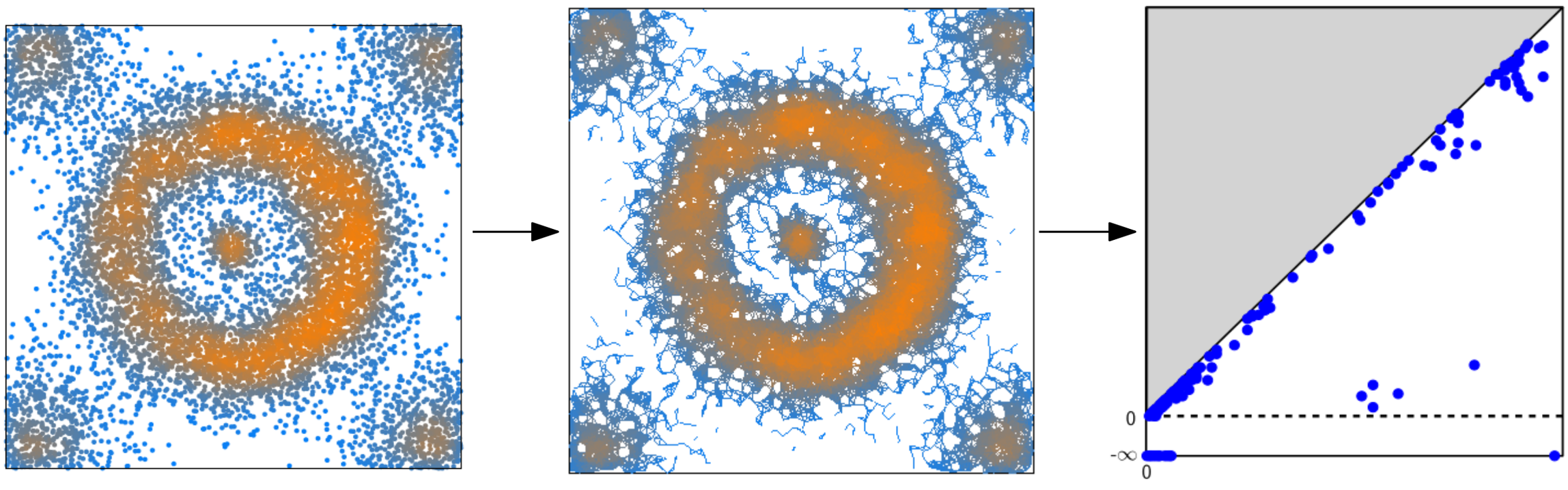
1. the algorithm sorts the vertices by decreasing density values,

2. and then makes a single pass through the vertex set, merging clusters on the fly using a union-find data structure.

$\rightarrow$ Running time: $O(n \log n + (n+m)\alpha(n))$

$\rightarrow$ Space complexity: $O(n+m)$

$\rightarrow$ Main memory usage: $O(n)$



Estimating the correct number of clusters

1. Define an order on the point cloud with a density estimator $\hat{f}$.

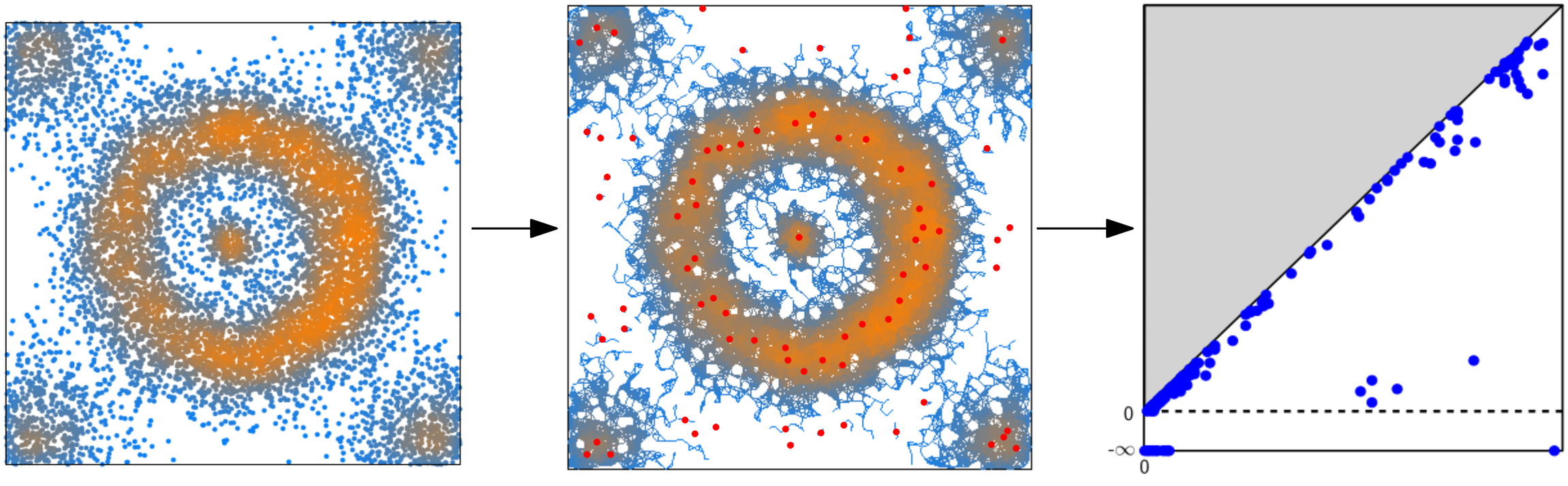
   (sort data points by **decreasing** estimated density values)

2. Extend order to the graph edges (i.e., compute the *upper-star filtration*).

   ($\hat{f}([u,v]) = \min\{\hat{f}(u),\ \hat{f}(v)\}$)

3. Compute the 0-dimensional persistence diagram of this filtration.

   (apply 0-dimensional persistence algorithm $\rightarrow$ union-find data structure)



Estimating the correct number of clusters

1. Define an order on the point cloud with a density estimator $\hat{f}$.

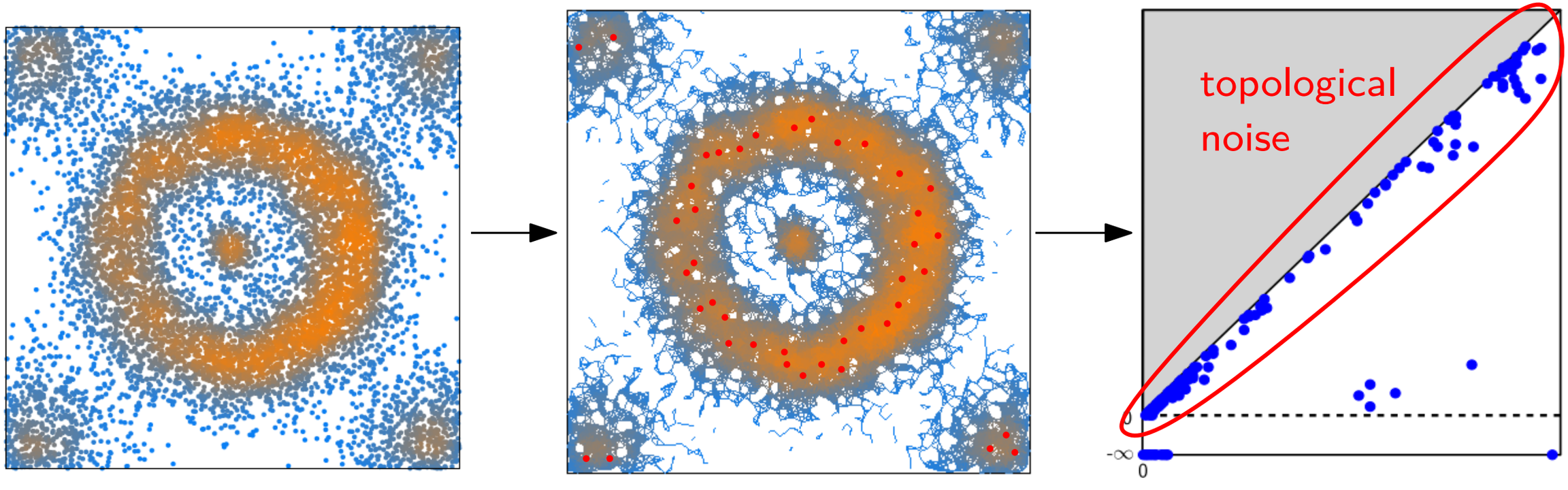
   (sort data points by **decreasing** estimated density values)

2. Extend order to the graph edges (i.e., compute the *upper-star filtration*).

   ($\hat{f}([u,v]) = \min\{\hat{f}(u),\ \hat{f}(v)\}$)

3. Compute the 0-dimensional persistence diagram of this filtration.

   (apply 0-dimensional persistence algorithm $\rightarrow$ union-find data structure)



Estimating the correct number of clusters

1. Define an order on the point cloud with a density estimator $\hat{f}$.

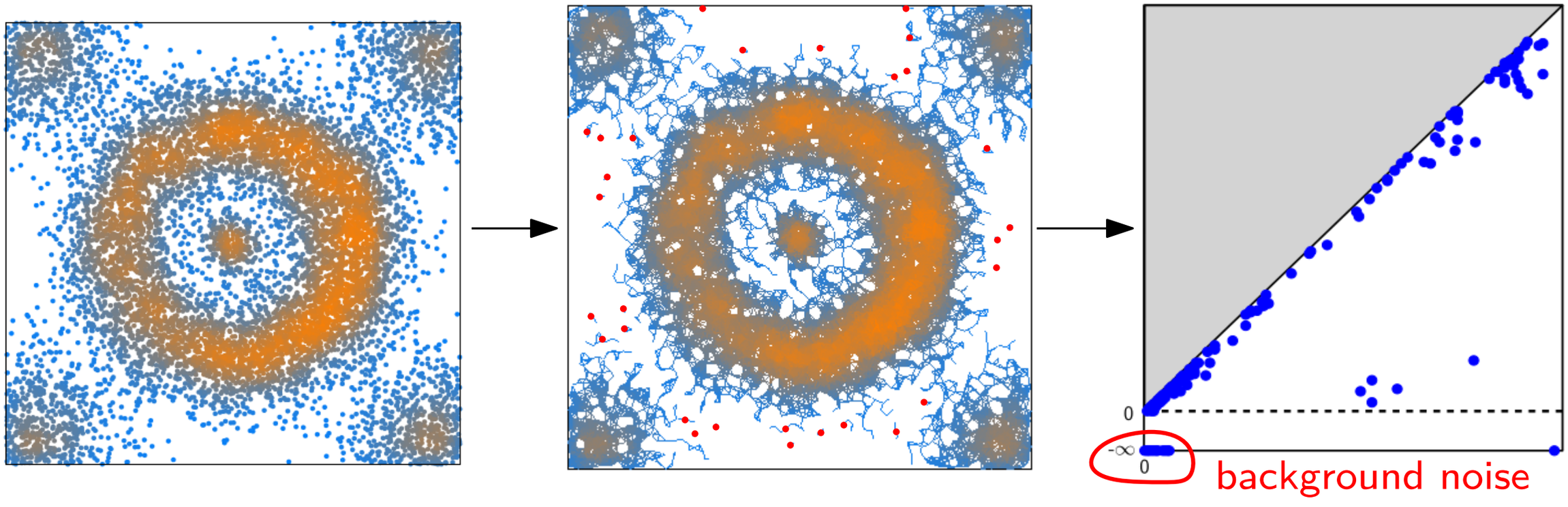
   (sort data points by **decreasing** estimated density values)

2. Extend order to the graph edges (i.e., compute the *upper-star filtration*).

   ($\hat{f}([u,v]) = \min\{\hat{f}(u),\ \hat{f}(v)\}$)

3. Compute the 0-dimensional persistence diagram of this filtration.

   (apply 0-dimensional persistence algorithm $\rightarrow$ union-find data structure)



Estimating the correct number of clusters

1. Define an order on the point cloud with a density estimator $\hat{f}$.

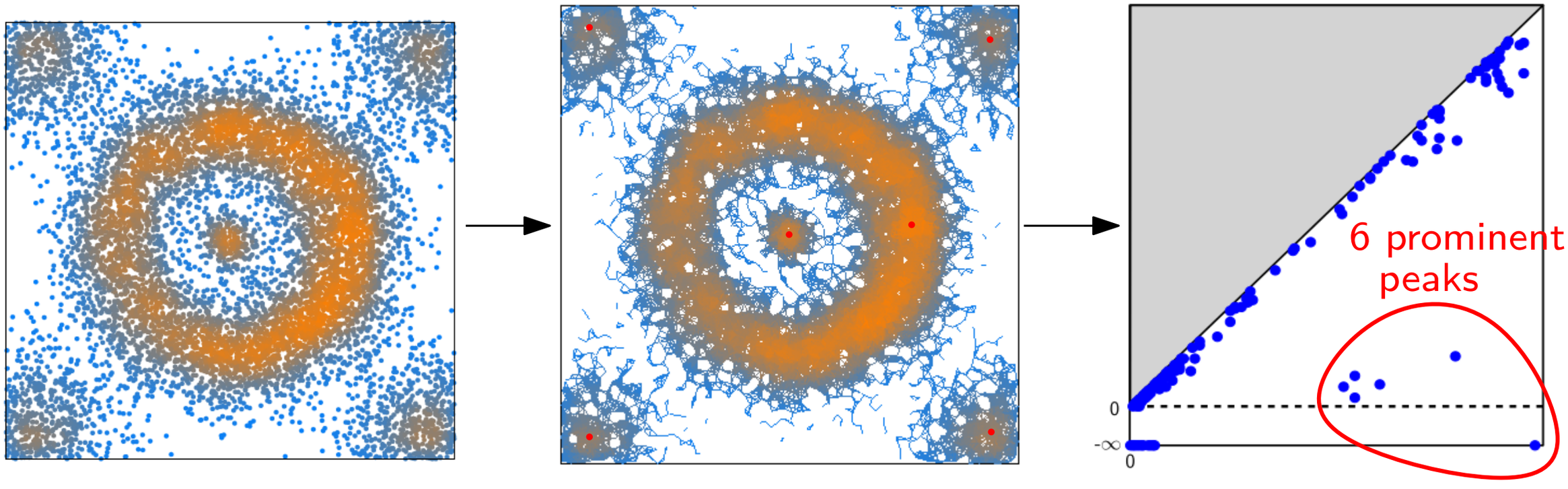
   (sort data points by **decreasing** estimated density values)

2. Extend order to the graph edges (i.e., compute the *upper-star filtration*).

   ($\hat{f}([u,v]) = \min\{\hat{f}(u),\ \hat{f}(v)\}$)

3. Compute the 0-dimensional persistence diagram of this filtration.

   (apply 0-dimensional persistence algorithm $\rightarrow$ union-find data structure)



Estimating the correct number of clusters

1. Define an order on the point cloud with a density estimator $\hat{f}$.

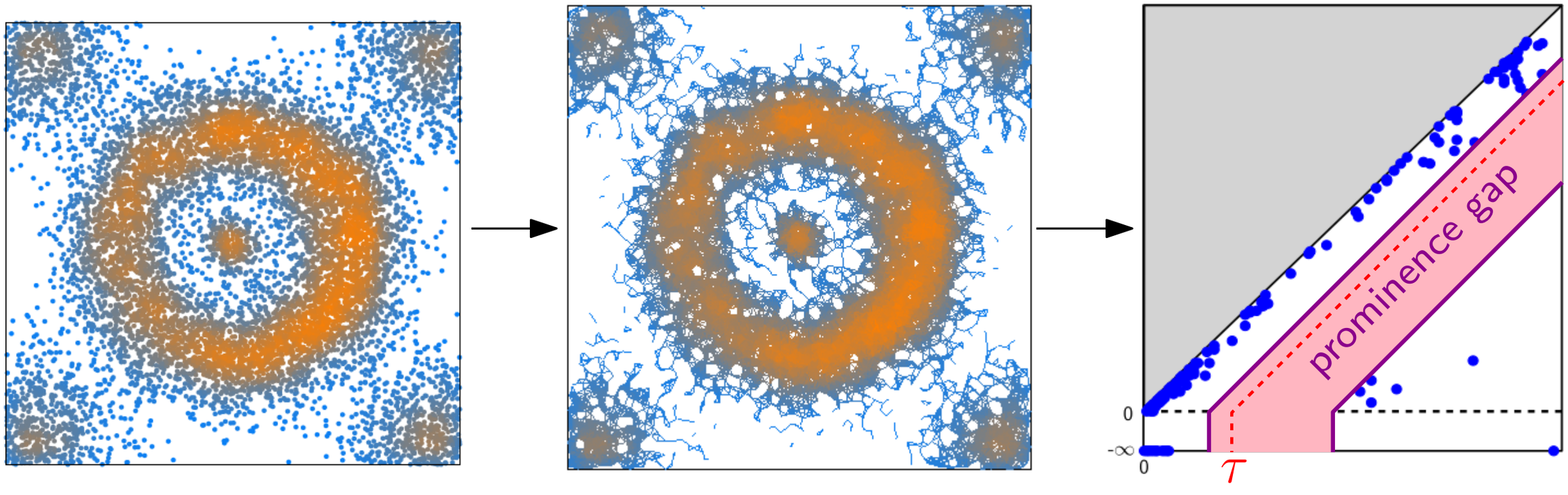
   (sort data points by **decreasing** estimated density values)

2. Extend order to the graph edges (i.e., compute the *upper-star filtration*).

   ($\hat{f}([u,v]) = \min\{\hat{f}(u),\ \hat{f}(v)\}$)

3. Compute the 0-dimensional persistence diagram of this filtration.

   (apply 0-dimensional persistence algorithm $\rightarrow$ union-find data structure)



Estimating the correct number of clusters

Hypotheses:

- $f : \mathbb{R}^d \to \mathbb{R}$ a c-Lipschitz probability density function,
- $P \subset \mathbb{R}^d$ a finite set of n points sampled i.i.d. according to f,
- $\hat{f} : P \to \mathbb{R}$ a density estimator s.t. $\eta := \max_{p \in P} |\hat{f}(p) - f(p)| < \Pi/5$,
- $G = (P, E)$ the δ-neighborhood graph for some positive $\delta < \frac{\Pi - 5\eta}{5c}$.

Note: Π is the prominence of the least prominent peak of f

Estimating the correct number of clusters

Hypotheses:

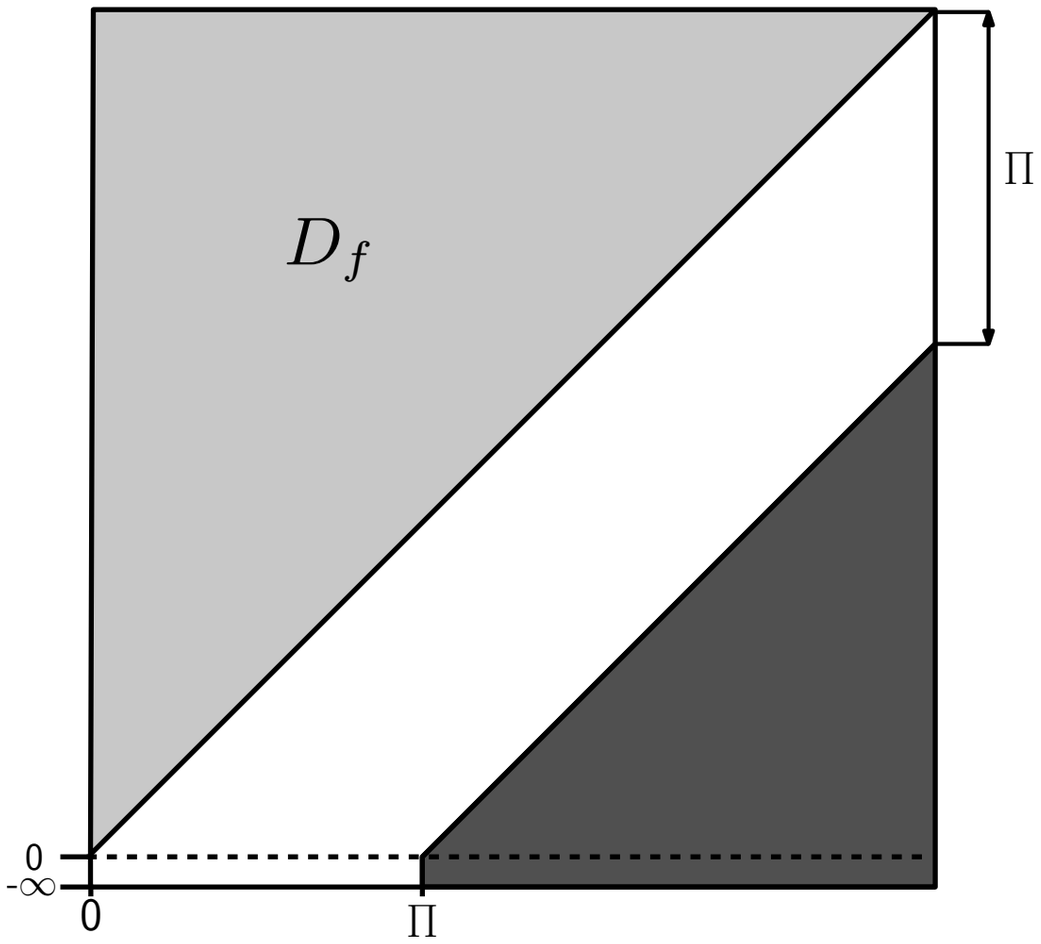
- $f : \mathbb{R}^d \to \mathbb{R}$ a c-Lipschitz probability density function,
- $P \subset \mathbb{R}^d$ a finite set of n points sampled i.i.d. according to f,
- $\hat{f} : P \to \mathbb{R}$ a density estimator s.t. $\eta := \max_{p \in P} |\hat{f}(p) - f(p)| < \Pi/5$,
- $G = (P, E)$ the δ-neighborhood graph for some positive $\delta < \frac{\Pi - 5\eta}{5c}$.

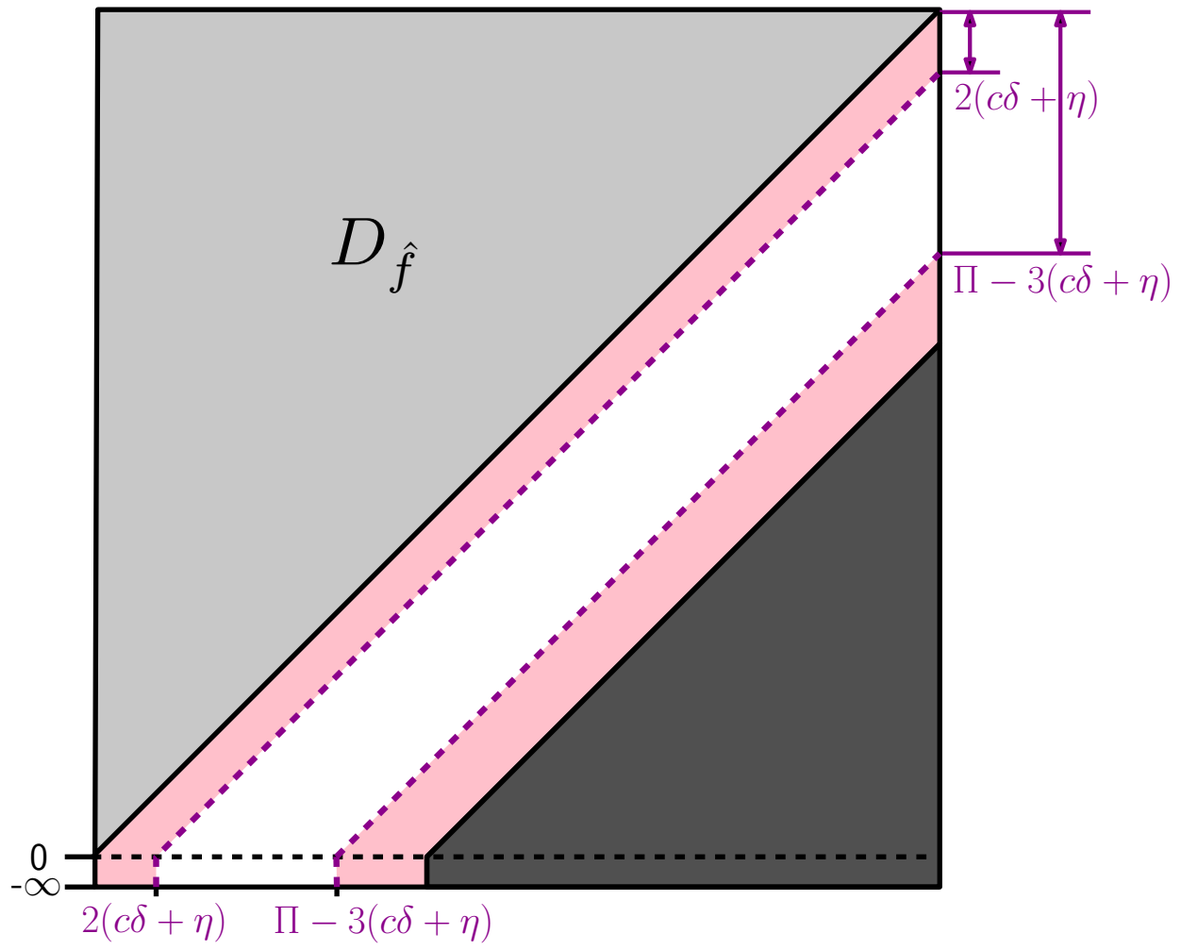
Note: Π is the prominence of the least prominent peak of f

Thm: For any choice of τ such that $2(c\delta + \eta) < \tau < \Pi - 3(c\delta + \eta)$, the number of clusters computed by the algorithm is equal to the number of peaks of f with probability at least $1 - e^{-\Omega(n)}$. *(the Ω notation hides factors depending on c, δ)*

Proof: Skipped. The main ingredient is the stability theorem.

Estimating the correct number of clusters





Thm: For any choice of τ such that $2(c\delta+\eta) < \tau < \Pi - 3(c\delta+\eta)$, the number of clusters computed by the algorithm is equal to the number of peaks of f with probability at least $1 - e^{-\Omega(n)}$. *(the Ω notation hides factors depending on c, δ)*

Proof: Skipped. The main ingredient is the stability theorem.

Pseudo-code

Input: A graph G with n vertices, an n-dimensional vector $\hat{f}$, and $\tau \geq 0$.

Sort the vertex indices $\{1, 2, \ldots, n\}$ in decreasing order: $\hat{f}(1) \geq \cdots \geq \hat{f}(n)$.

Initialize a union-find data structure $\mathcal{U}$ and two lists g, r of length n.

`for` $i \in \{1, \ldots, n\}$:

    Let $\mathcal{N}$ be the set of neighbors of i in G that have indices lower than i

    `if` $\mathcal{N} = \varnothing$:

        Create a new entry e in $\mathcal{U}$ and attach vertex i to it: $\mathcal{U}$.`MakeSet`(i)

        $r[e] \leftarrow i$ *($r[e]$ stores the root vertex associated with the entry e)*

    `else`:

        $g[i] \leftarrow \operatorname{argmax}\{\hat{f}(j) : j \in \mathcal{N}\}$ *($g[i]$ stores the approximate gradient at vertex i)*

        $e_i \leftarrow \mathcal{U}$.`Find`$(g[i])$

        Attach vertex i to the entry e_i: $\mathcal{U}$.`Union`(i, e_i)

        `for` $j \in \mathcal{N}$:

            $e \leftarrow \mathcal{U}$.`Find`(j)

            `if` $e \neq e_i$ and $\min\{\hat{f}(r[e]),\ \hat{f}(r[e_i])\} < \hat{f}(i) + \tau$:

                $\mathcal{U}$.`Union`$(e,\ e_i)$

                $r[e \cup e_i] \leftarrow \operatorname{argmax}\{\hat{f}(r[e]),\ \hat{f}(r[e_i])\}$

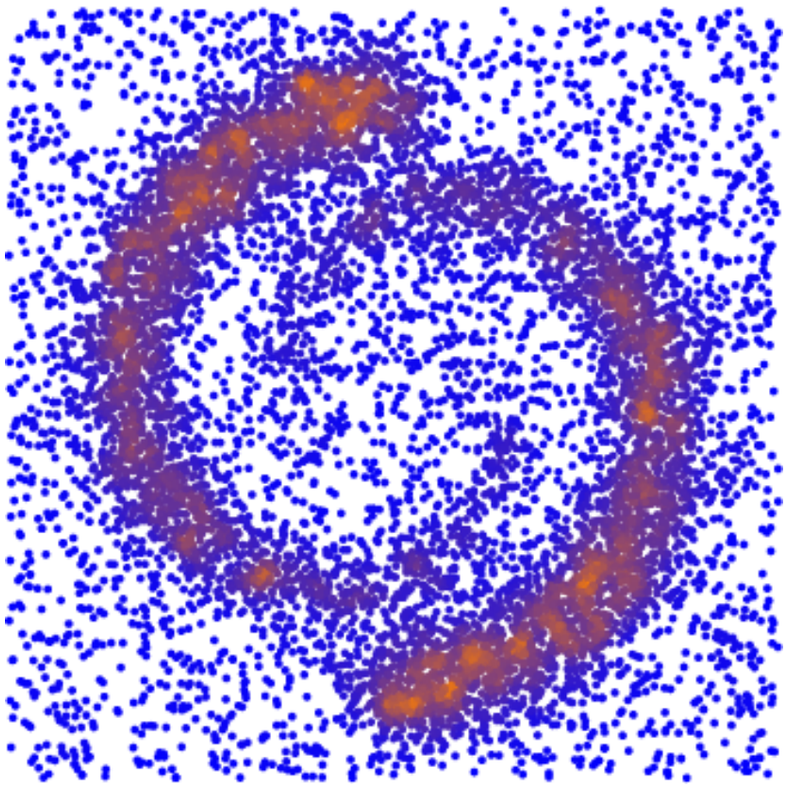
                $e_i \leftarrow e \cup e_i$

cluster merges with persistence

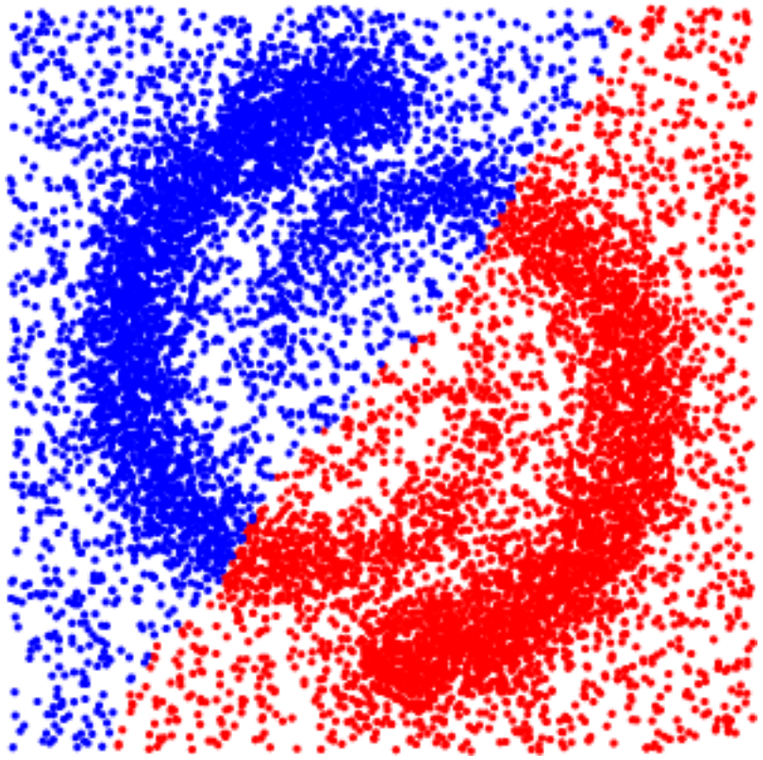
Output: the collection of entries e of $\mathcal{U}$ such that $\hat{f}(r(e)) \geq \tau$.

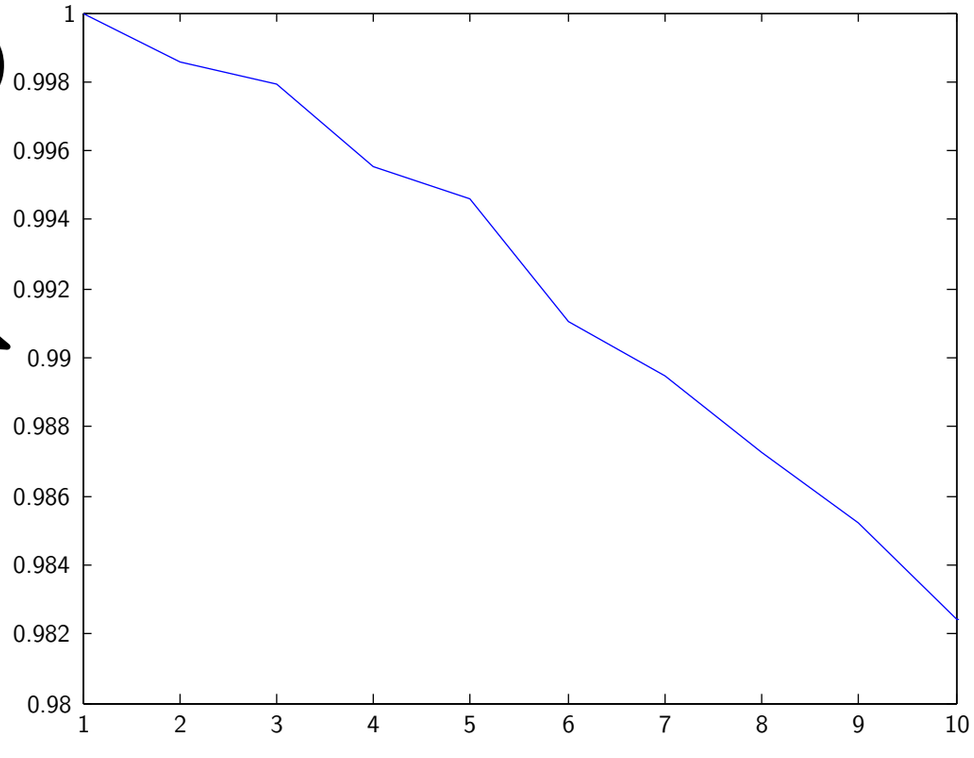
Experimental results

Synthetic Data



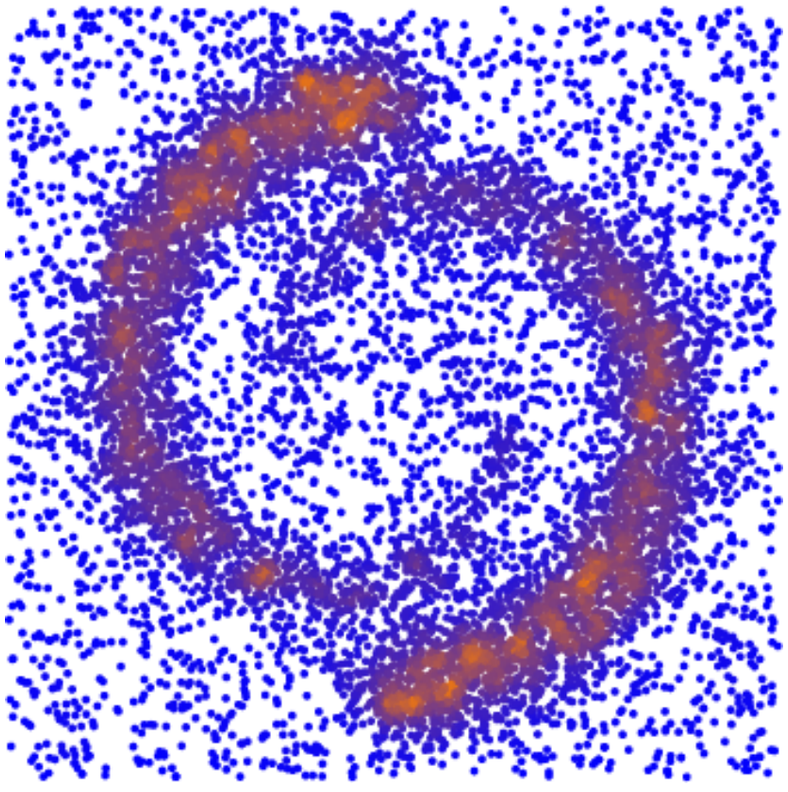
Spectral clustering
(k-means in eigenspace)





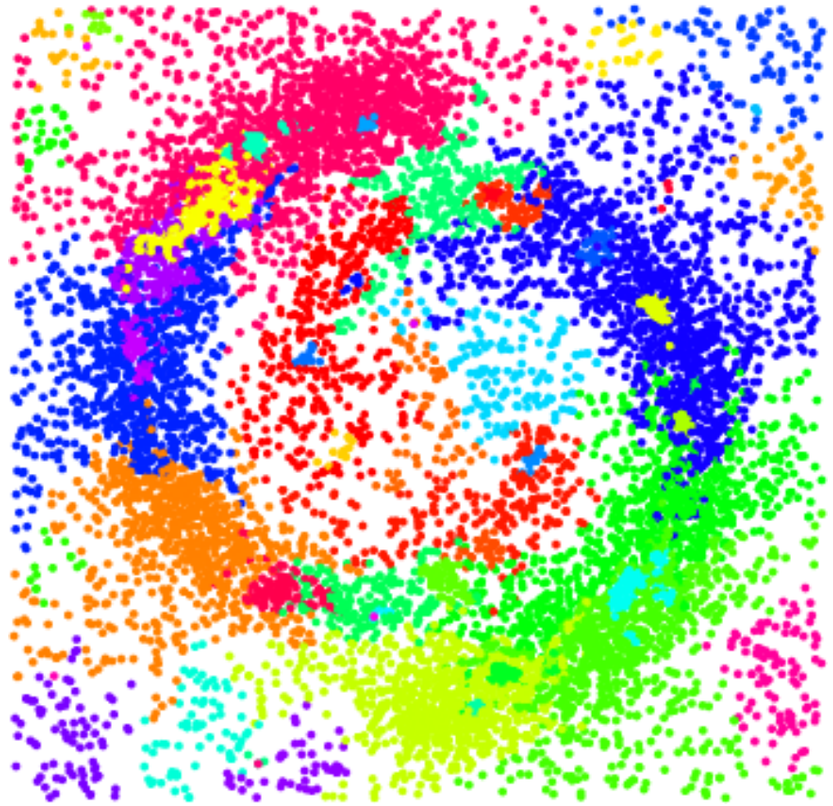
Experimental results

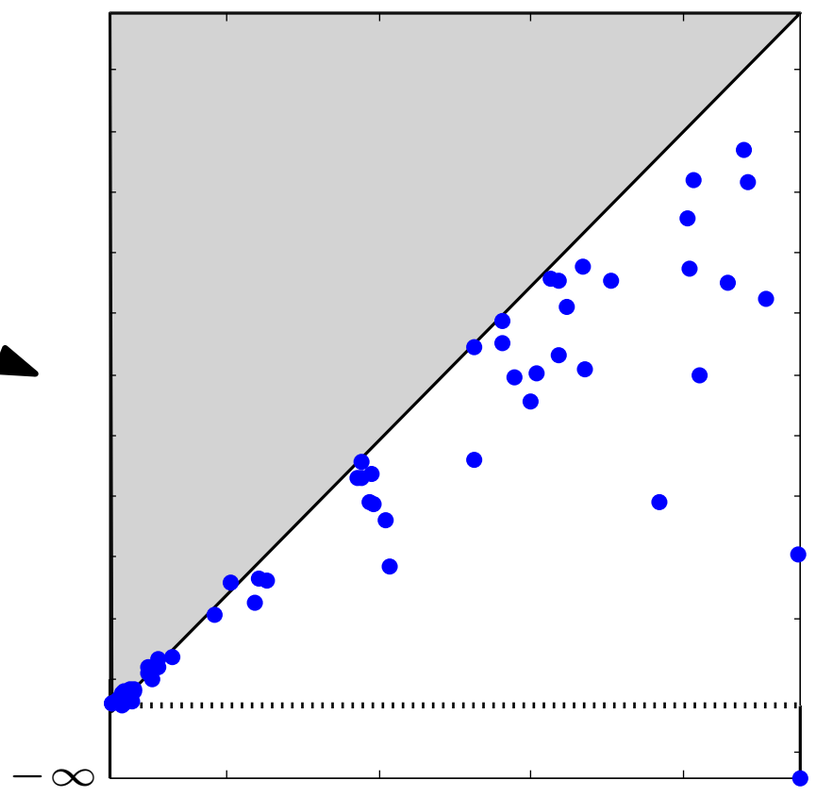
Synthetic Data



$\tau = 0$

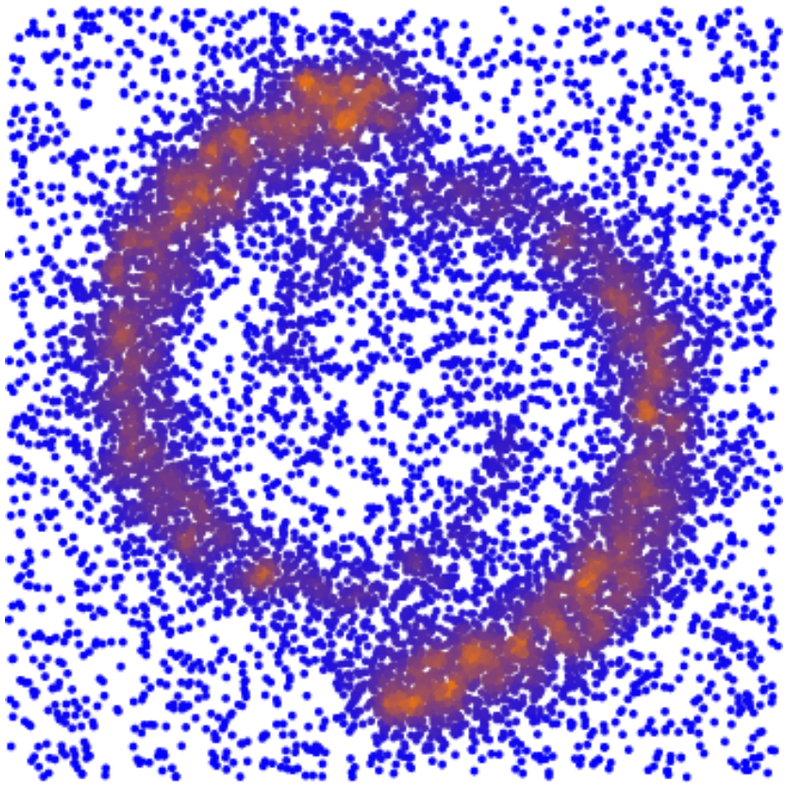
ToMATo



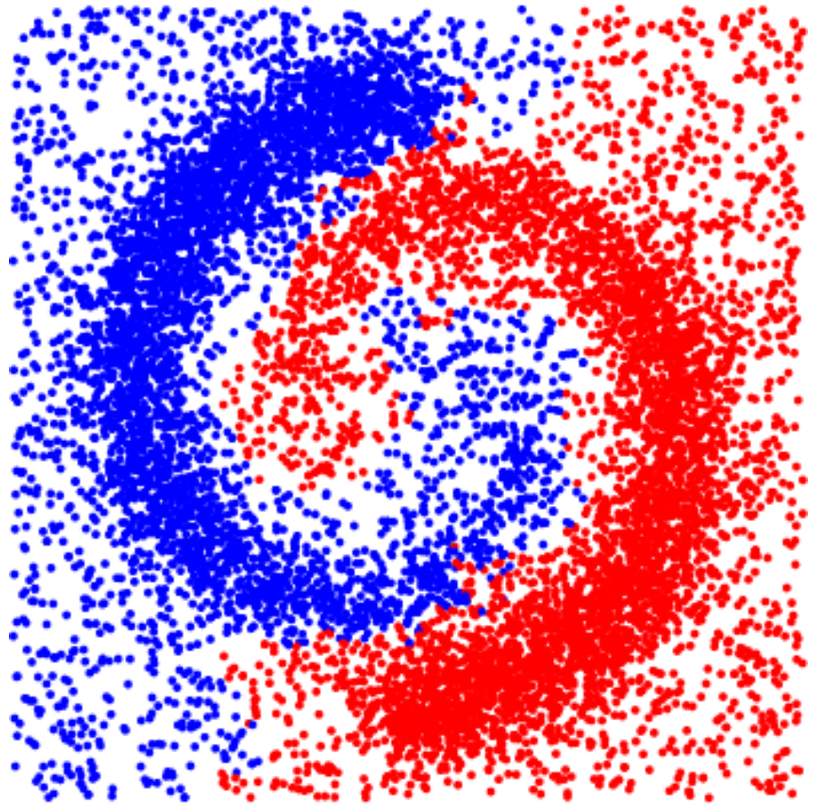


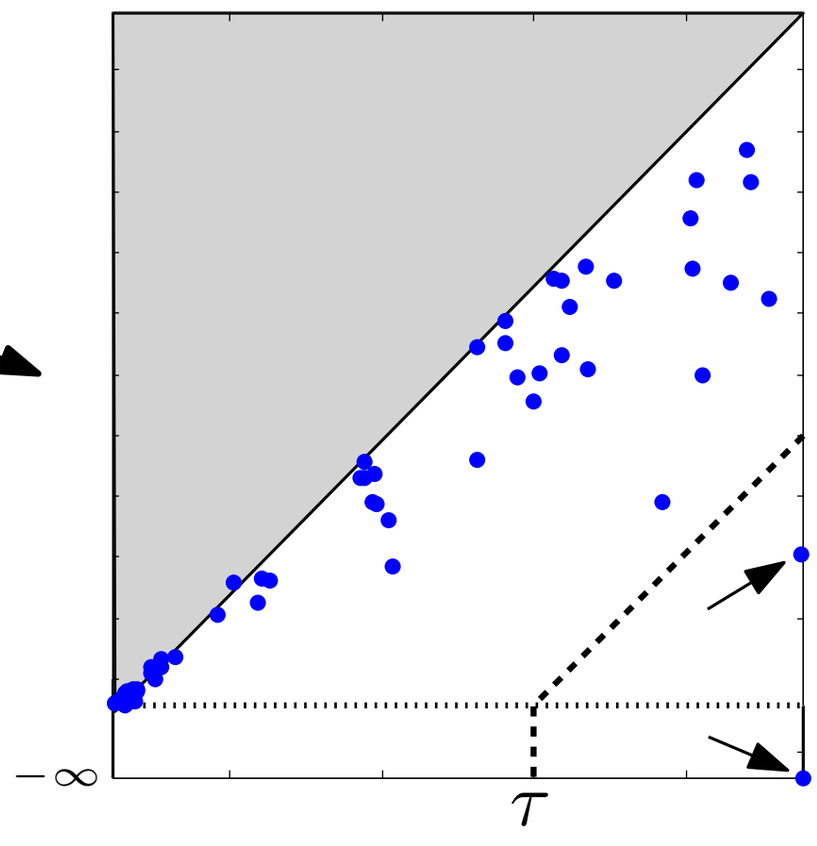
Experimental results

Synthetic Data



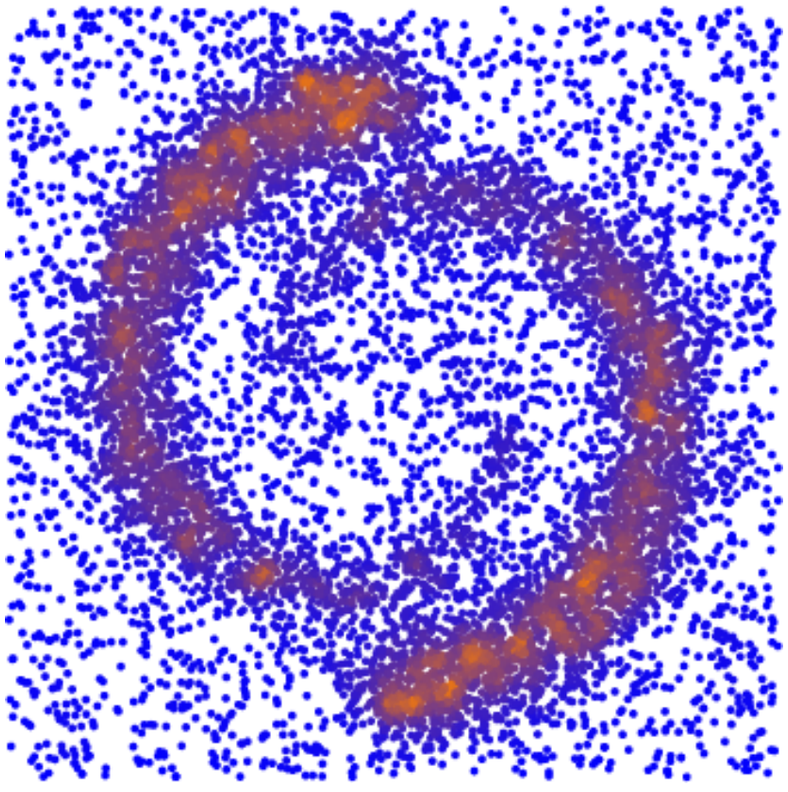
ToMATo

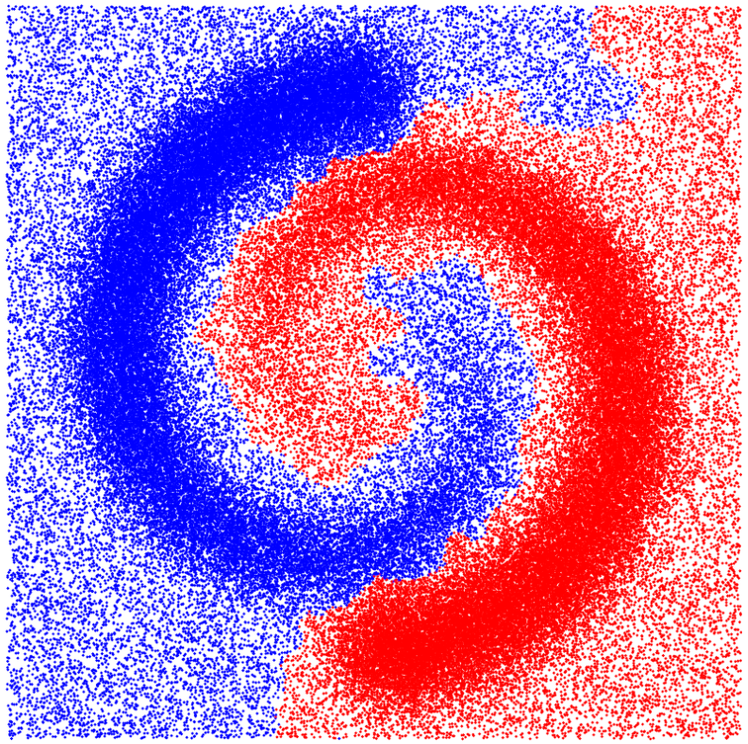


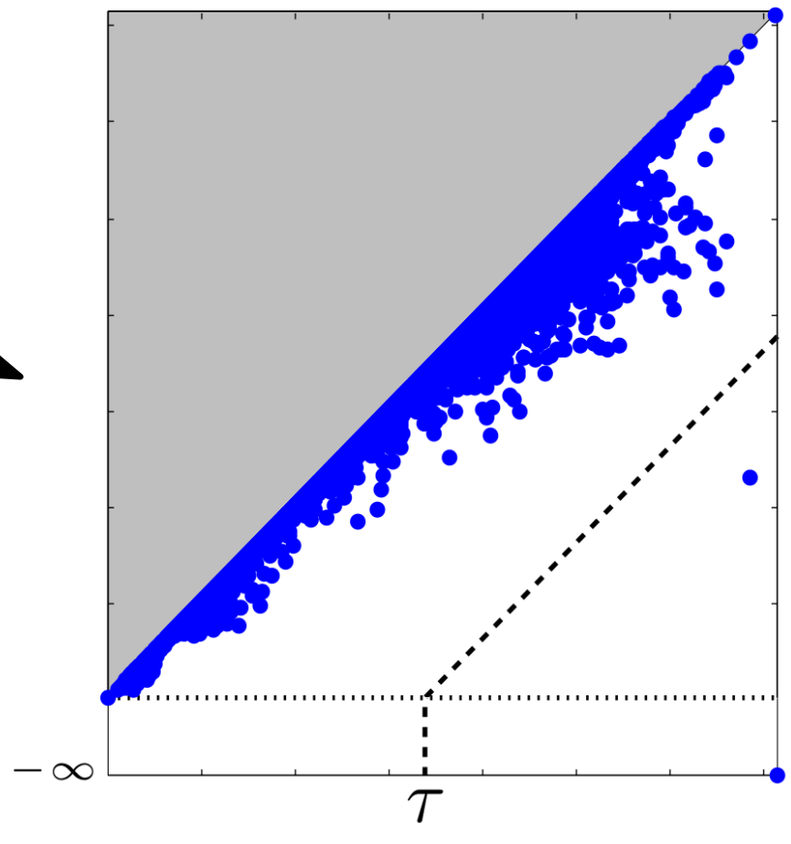


Experimental results

Synthetic Data





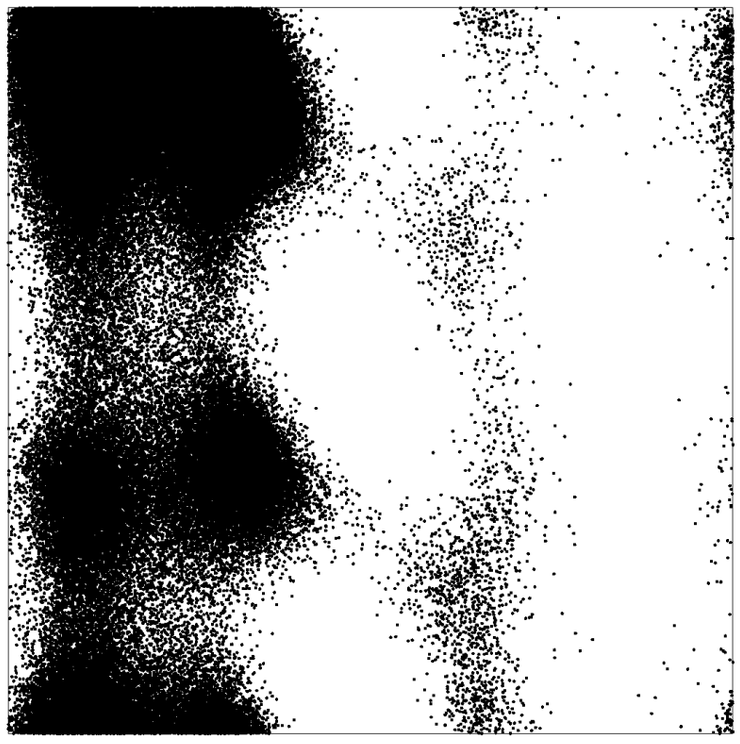


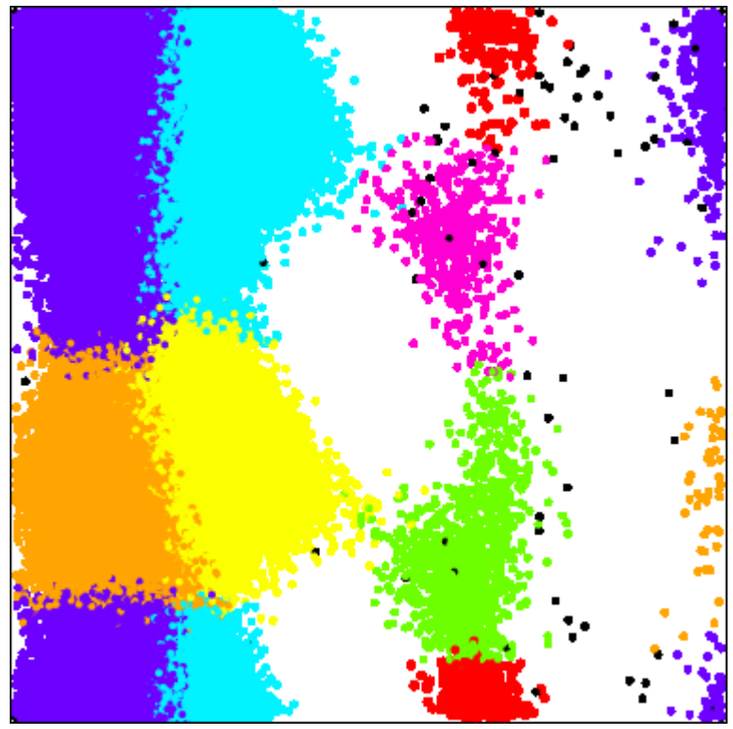
Experimental results

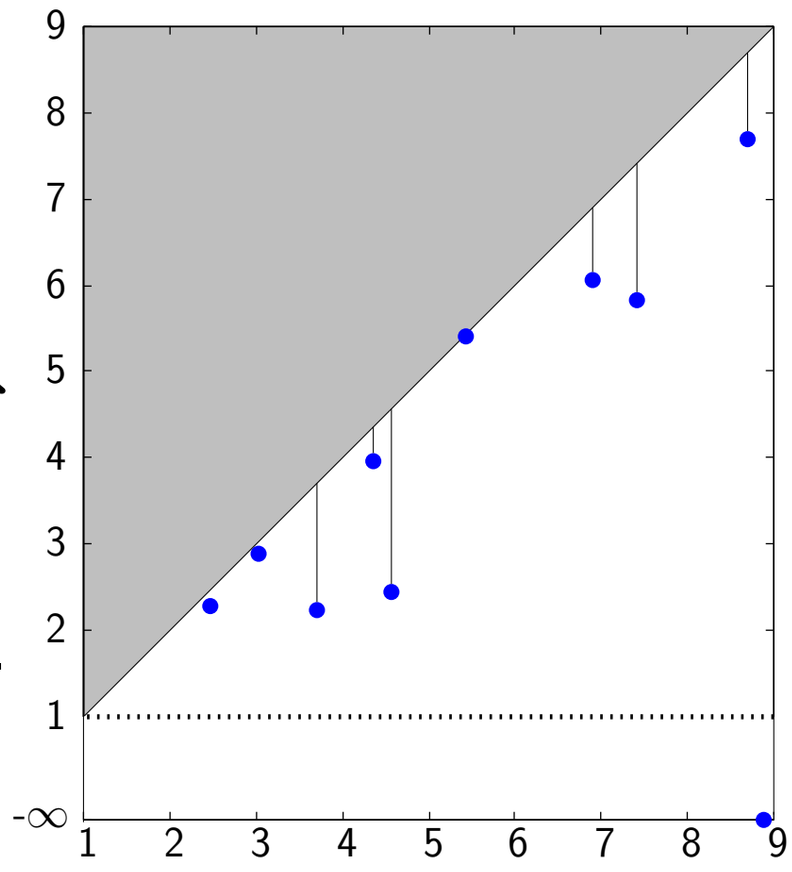
Biological Data

Alanine-Dipeptide conformations ($\mathbb{R}^{21}$)

with RMSD distance (non-Euclidean).







Common belief: 6 metastable states.

PD shows anywhere between 4 and 7 clusters.

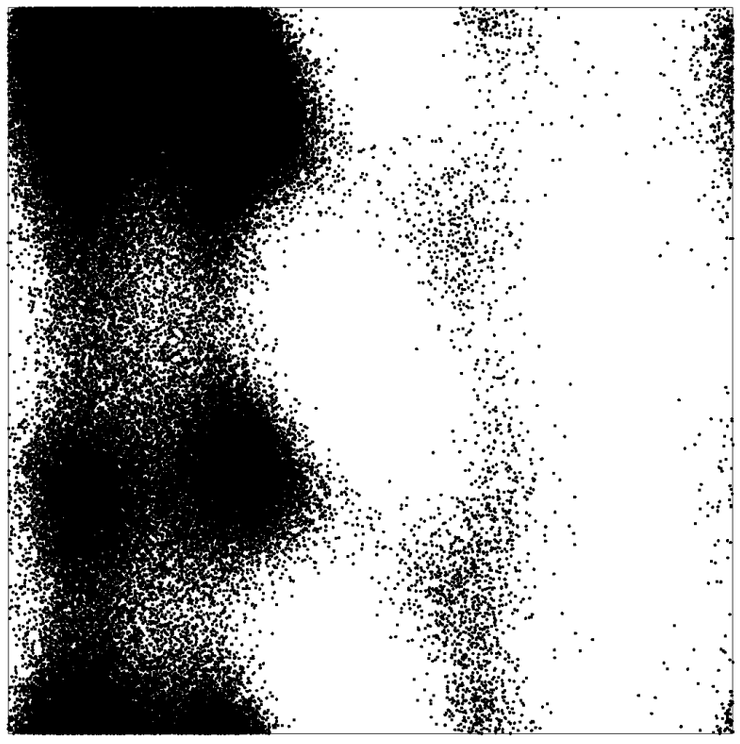
Experimental results

[*Topological methods for exploring low-density states in biomolecular folding pathways*, Yao, Sun, Huang, Bowman, Singh, Lesnick, Guibas, Pande, Carlsson, J. Chem. Phys., 2009]

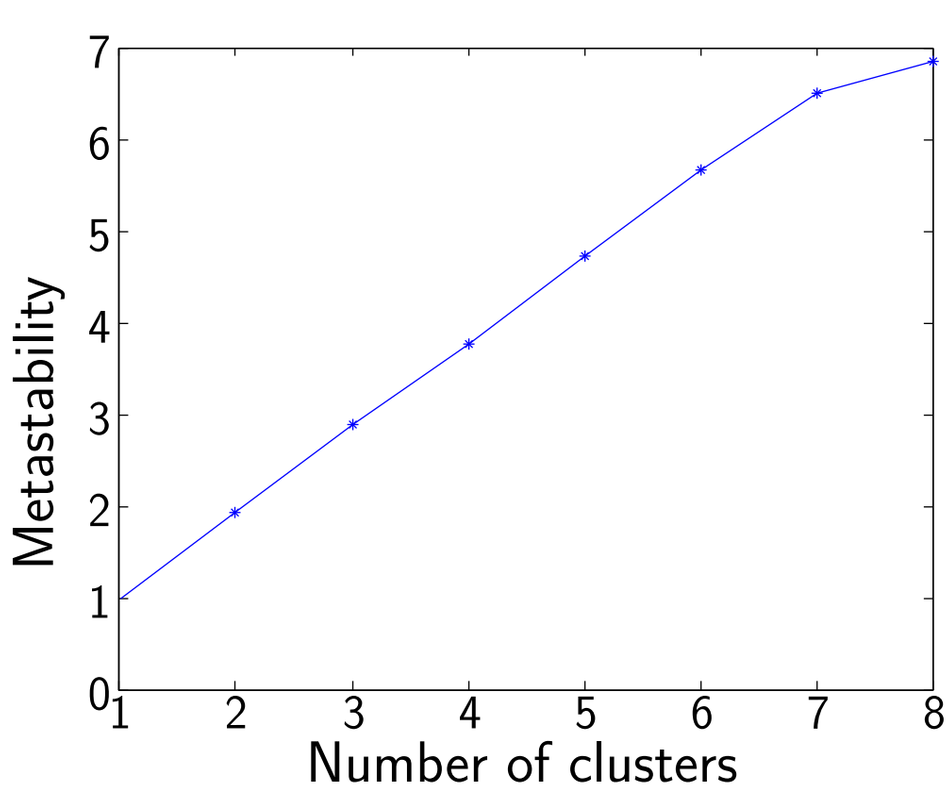
Biological Data

Alanine-Dipeptide conformations ($\mathbb{R}^{21}$)

with RMSD distance (non-Euclidean).



Rank	Prominence	Metastability
1	$+\infty$	0.99982
2	3827	1.91865
3	1334	2.8813
4	557	3.76217
5	85	4.73838
6	32	5.65553
7	26	6.50757
8	7.2	6.8193
9	3.0	-
10	2.2	-



Common belief: 6 metastable states.

PD shows anywhere between 4 and 7 clusters.

Measures of metastability confirm this insight.

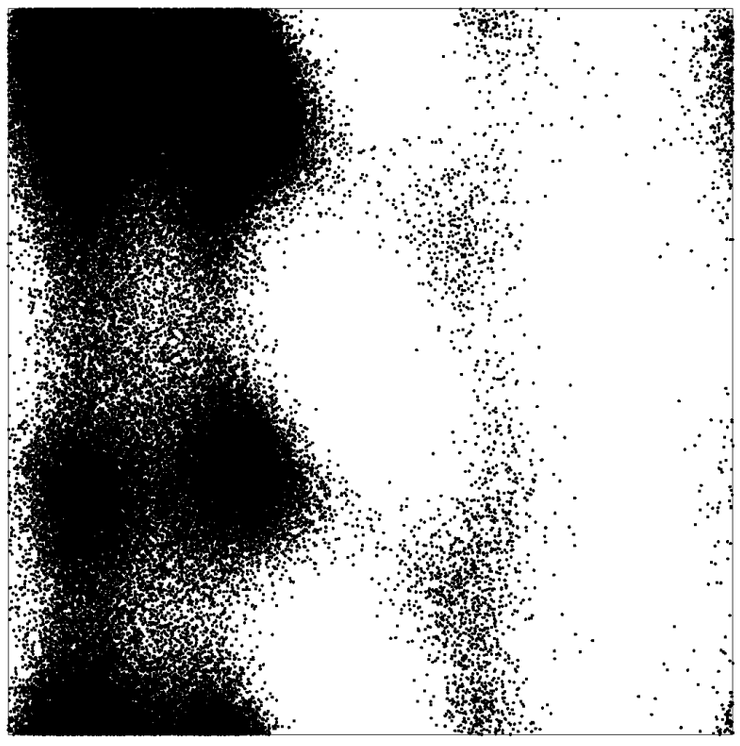
Experimental results

[*Topological methods for exploring low-density states in biomolecular folding pathways*, Yao, Sun, Huang, Bowman, Singh, Lesnick, Guibas, Pande, Carlsson, J. Chem. Phys., 2009]

Biological Data

Alanine-Dipeptide conformations ($\mathbb{R}^{21}$)

with RMSD distance (non-Euclidean).



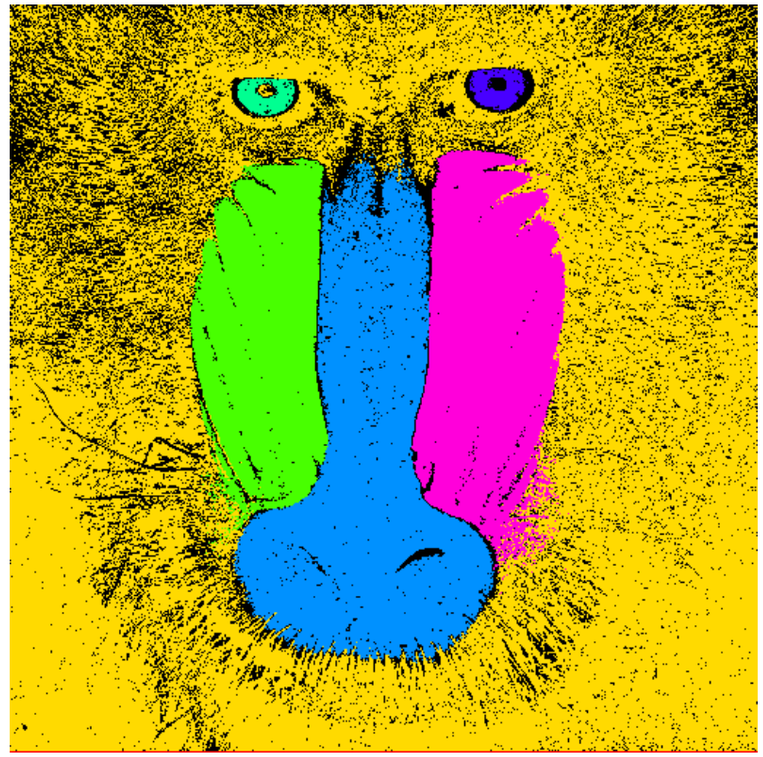
Note: Spectral Clustering takes a week of tweaking, while ToMATo runs out-of-the-box in a few minutes.

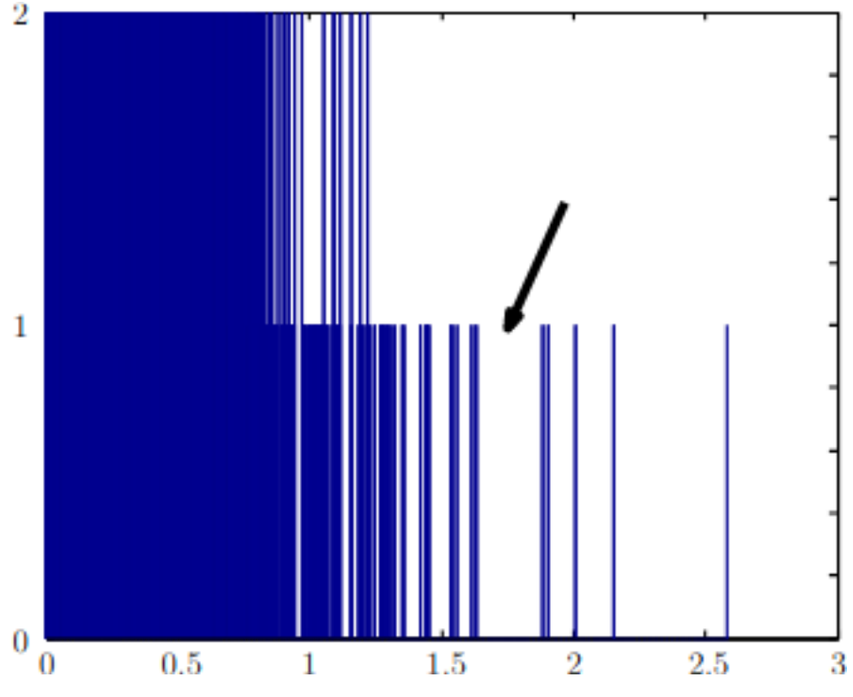
Experimental results

Image Segmentation

Density is estimated in 3D color space.

Neighborhood graph is built in image domain.



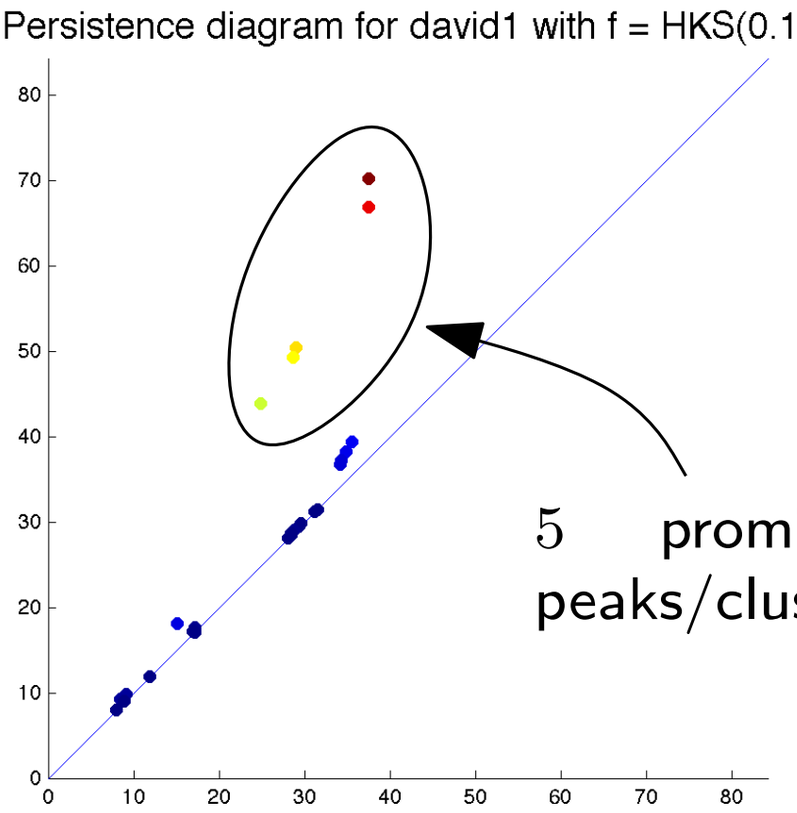


Distribution of prominences does not usually show a clear unique gap.

Still, relationship between choice of τ and number of obtained clusters remains explicit.

Application to non-rigid shape segmentation

[*Persistence-Based Segmentation of Deformable Shapes*, Skraba, Ovsjanikov, Chazal, Guibas, Proc. CVPR 2010]

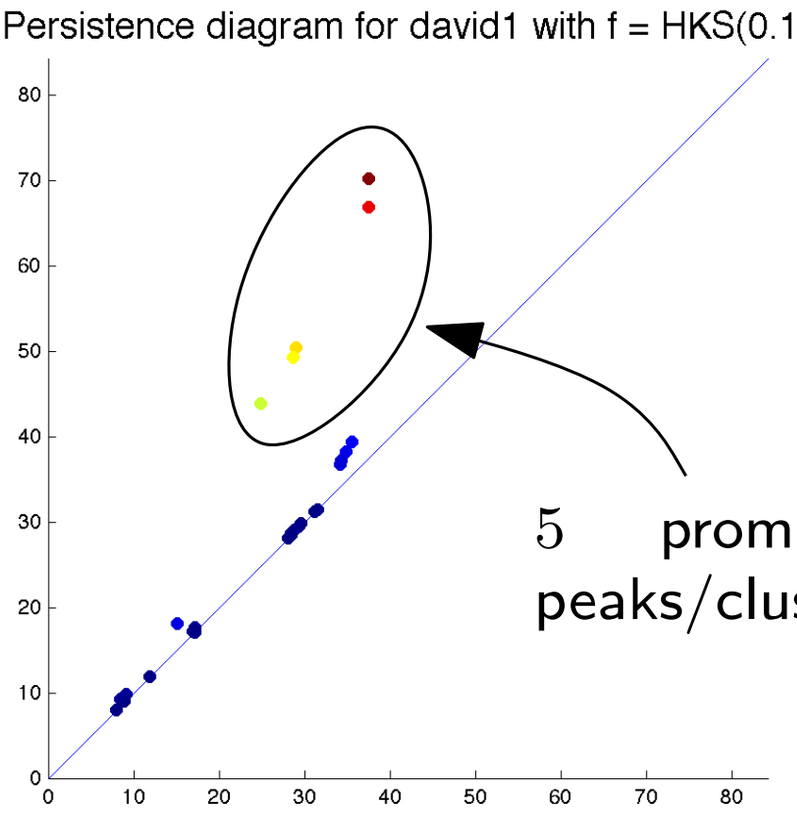


X : a 3D shape

f = HKS function on X

Application to non-rigid shape segmentation

[*Persistence-Based Segmentation of Deformable Shapes*, Skraba, Ovsjanikov, Chazal, Guibas, Proc. CVPR 2010]

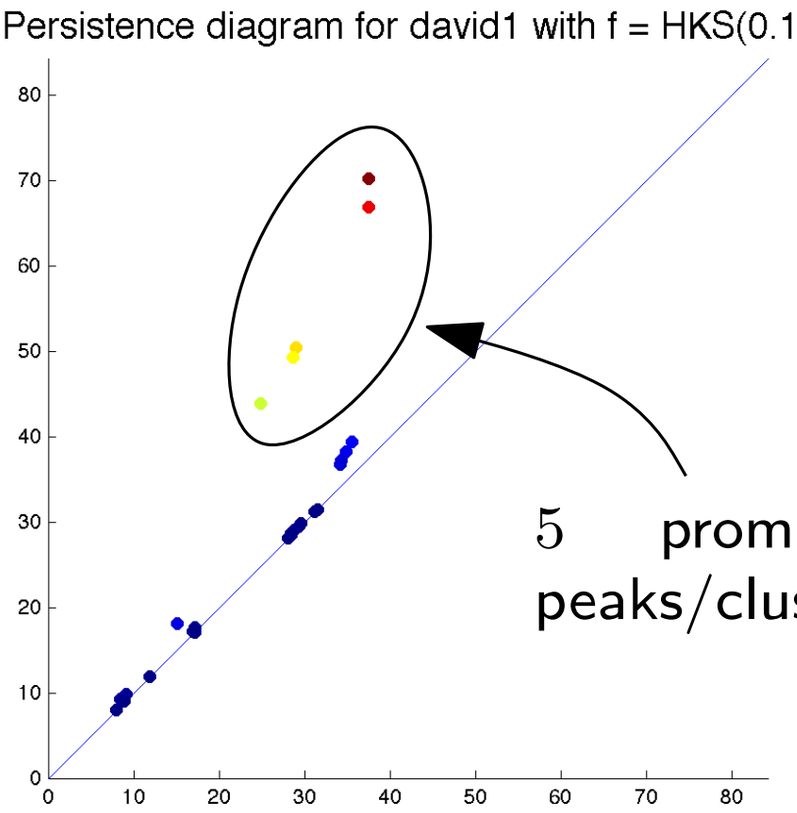


X : a 3D shape
f = HKS function on X

Problem: cluster boundaries are unstable, which gives dirty segments.

Application to non-rigid shape segmentation

[*Persistence-Based Segmentation of Deformable Shapes*, Skraba, Ovsjanikov, Chazal, Guibas, Proc. CVPR 2010]



X : a 3D shape
f = HKS function on X

Problem: cluster boundaries are unstable, which gives dirty segments.

See practical session! :-)

I. ToMATo algorithm

1. Introduction to hierarchical and mode-seeking clustering
2. ToMATo algorithm and guarantees

II. Mapper algorithm

1. Reeb spaces and Mappers
2. Confidence intervals

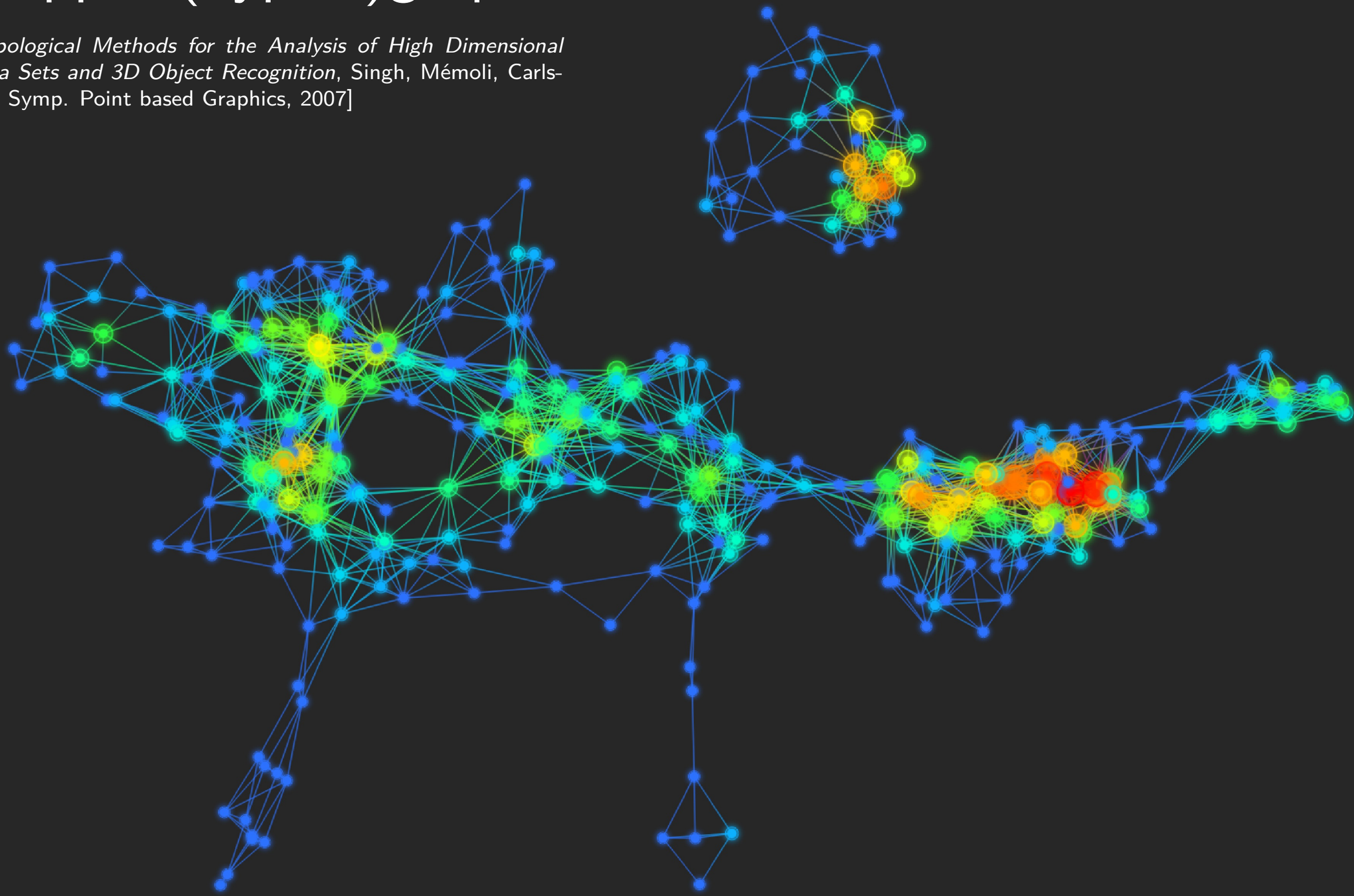
[*Structure and Stability of the One-Dimensional Mapper*, C., Oudot, Foundations of Computational Mathematics, 2018]

[*Statistical Analysis and Parameter Selection for Mapper*, C., Michel, Oudot, Journal of Machine Learning Research, 2018]

[*Statistical analysis of Mapper for stochastic and multivariate filters*, C., Michel, Journal of Applied and Computational Topology, 2022]

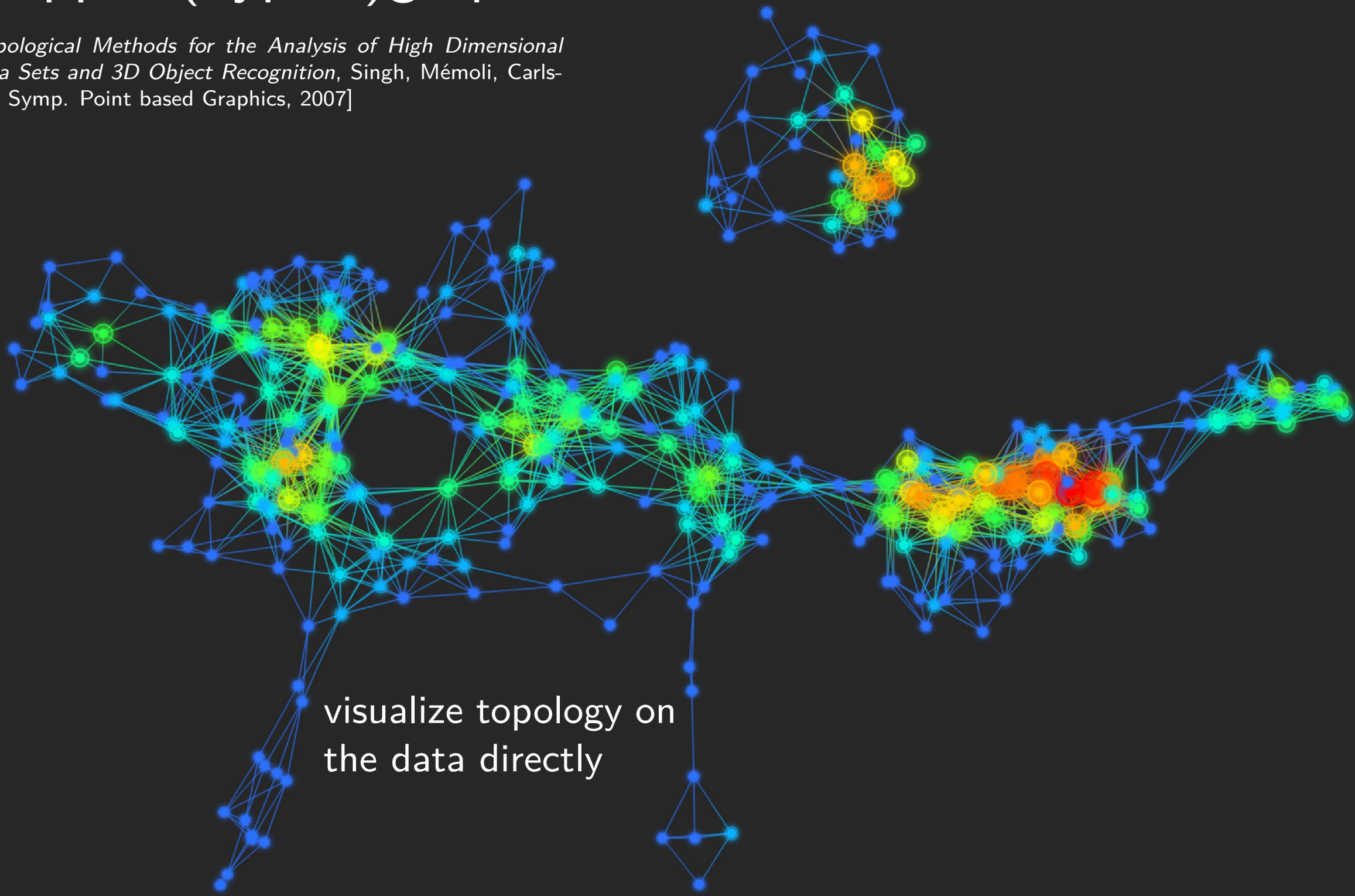
Mapper (hyper-)graphs

[*Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition*, Singh, Mémoli, Carlsson, Symp. Point based Graphics, 2007]



Mapper (hyper-)graphs

[*Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition*, Singh, Mémoli, Carlsson, Symp. Point based Graphics, 2007]



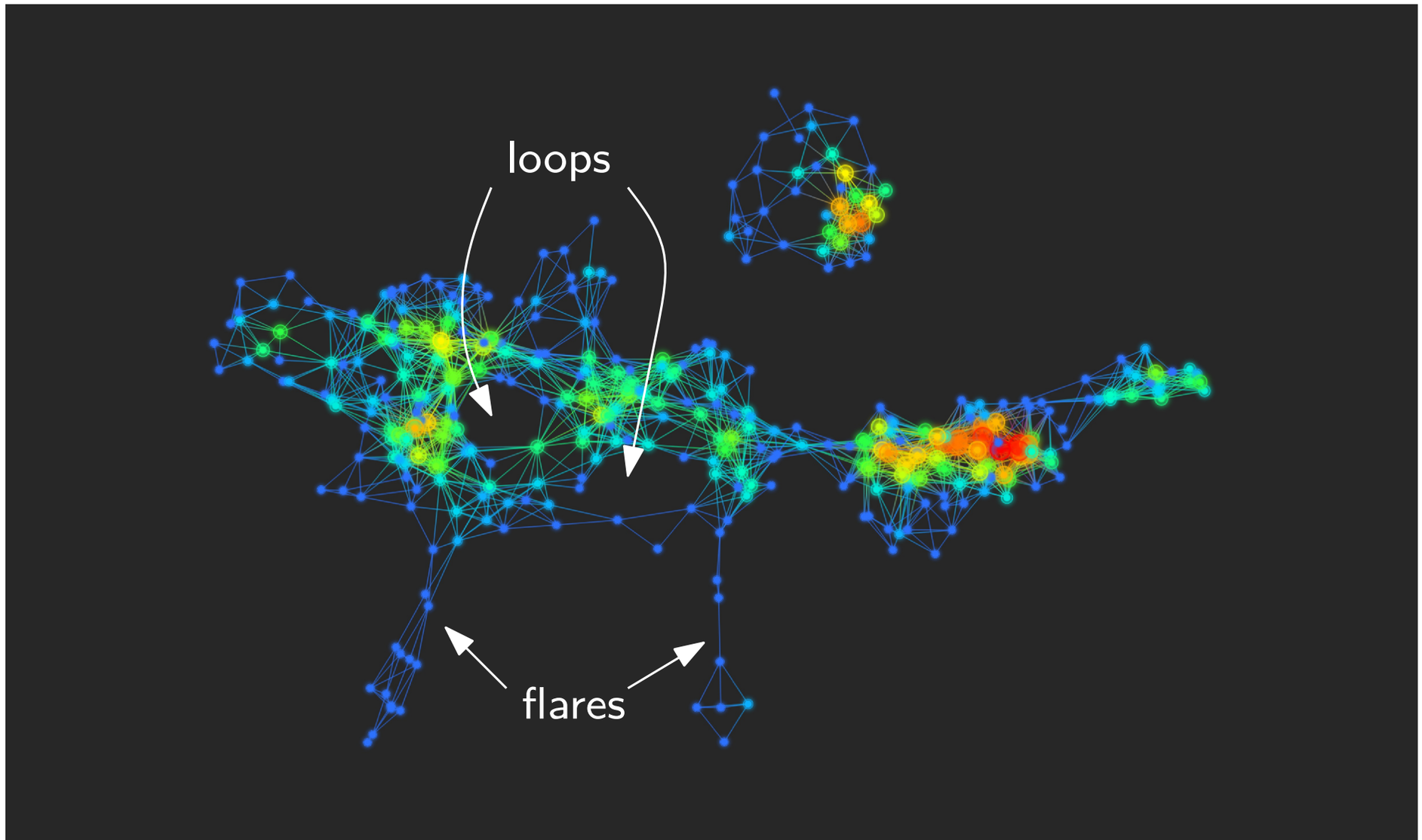
visualize topology on
the data directly

Mapper in applications

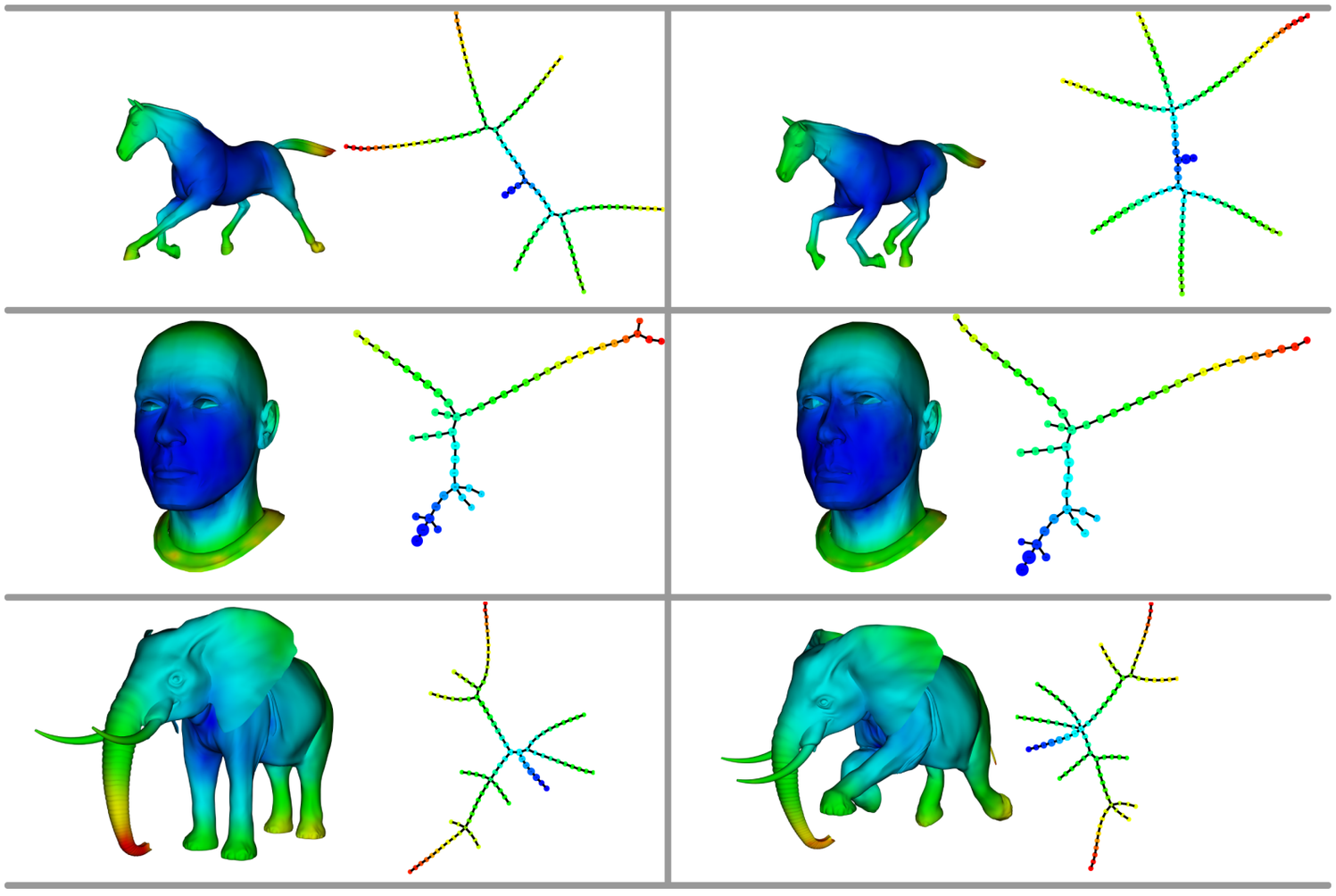
Two types of applications:

→ clustering
→ feature selection

principle: identify statistically relevant sub-populations through patterns (flares, loops)

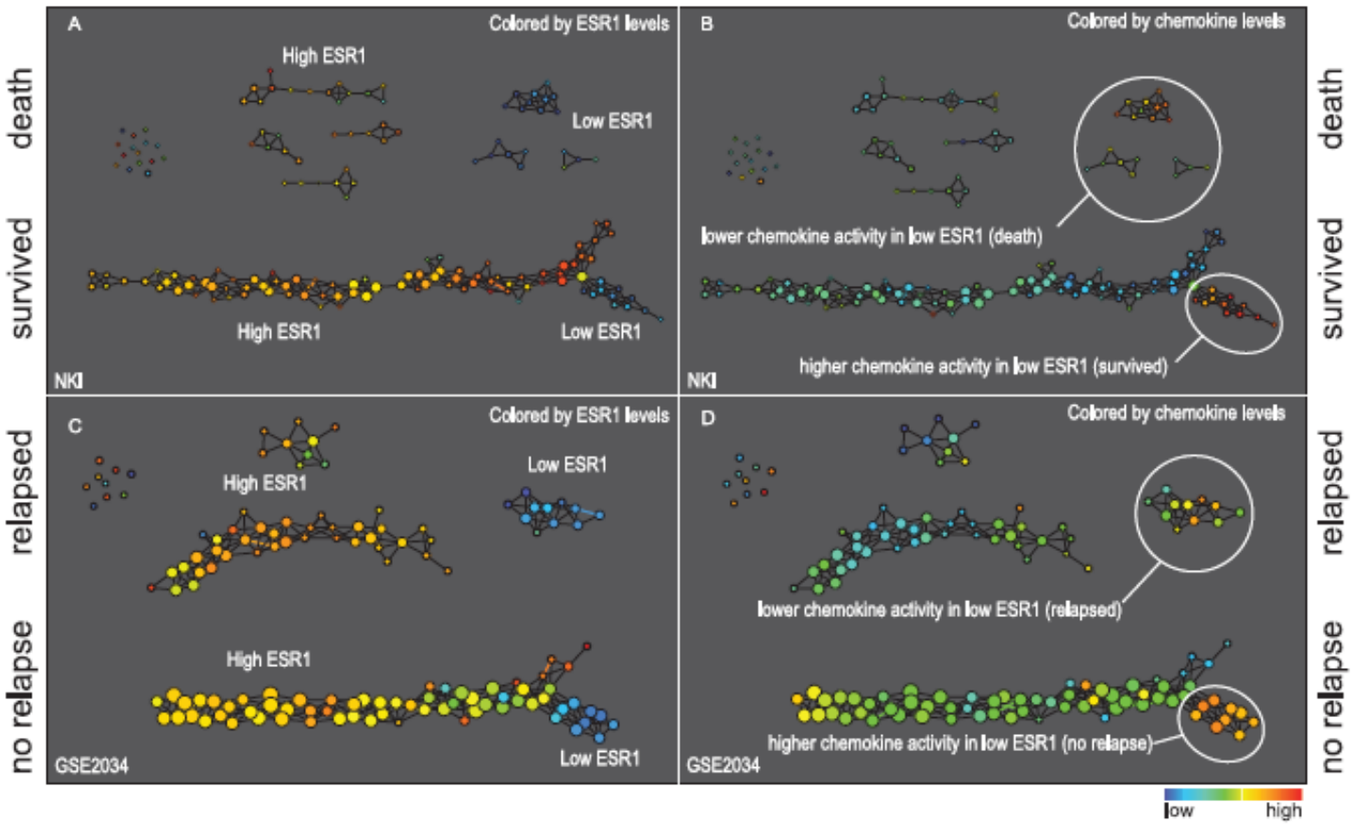


Mapper in applications



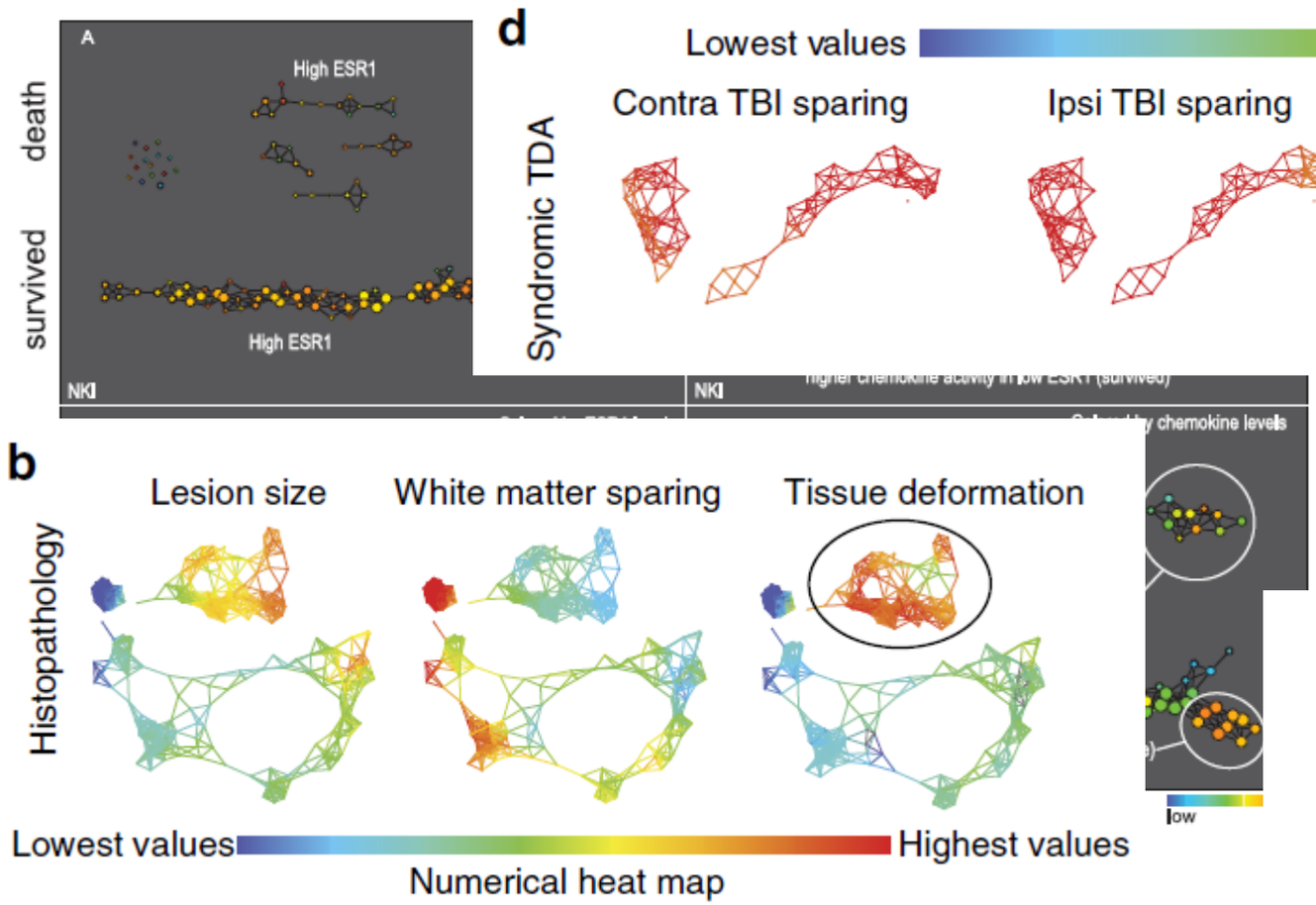
3d shapes classification

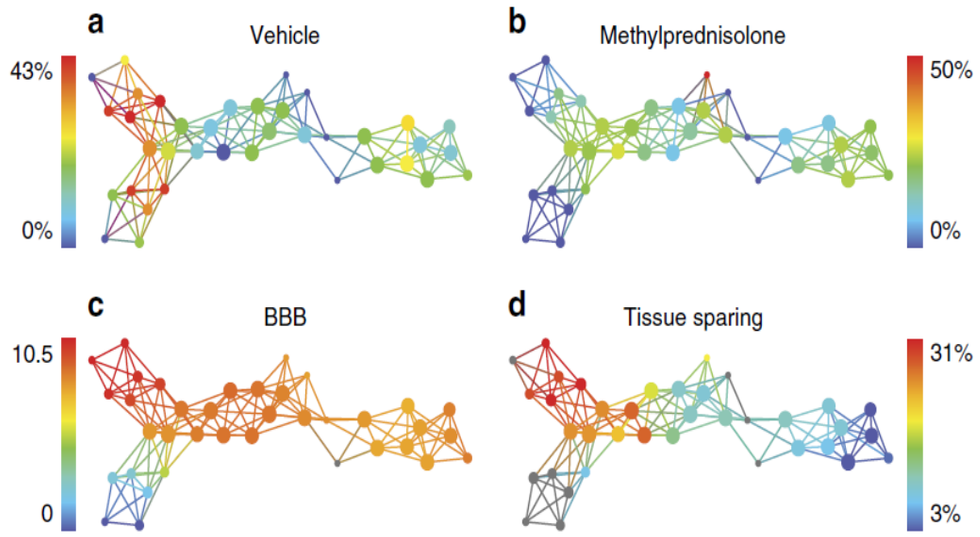
Mapper in applications



breast cancer subtype identification

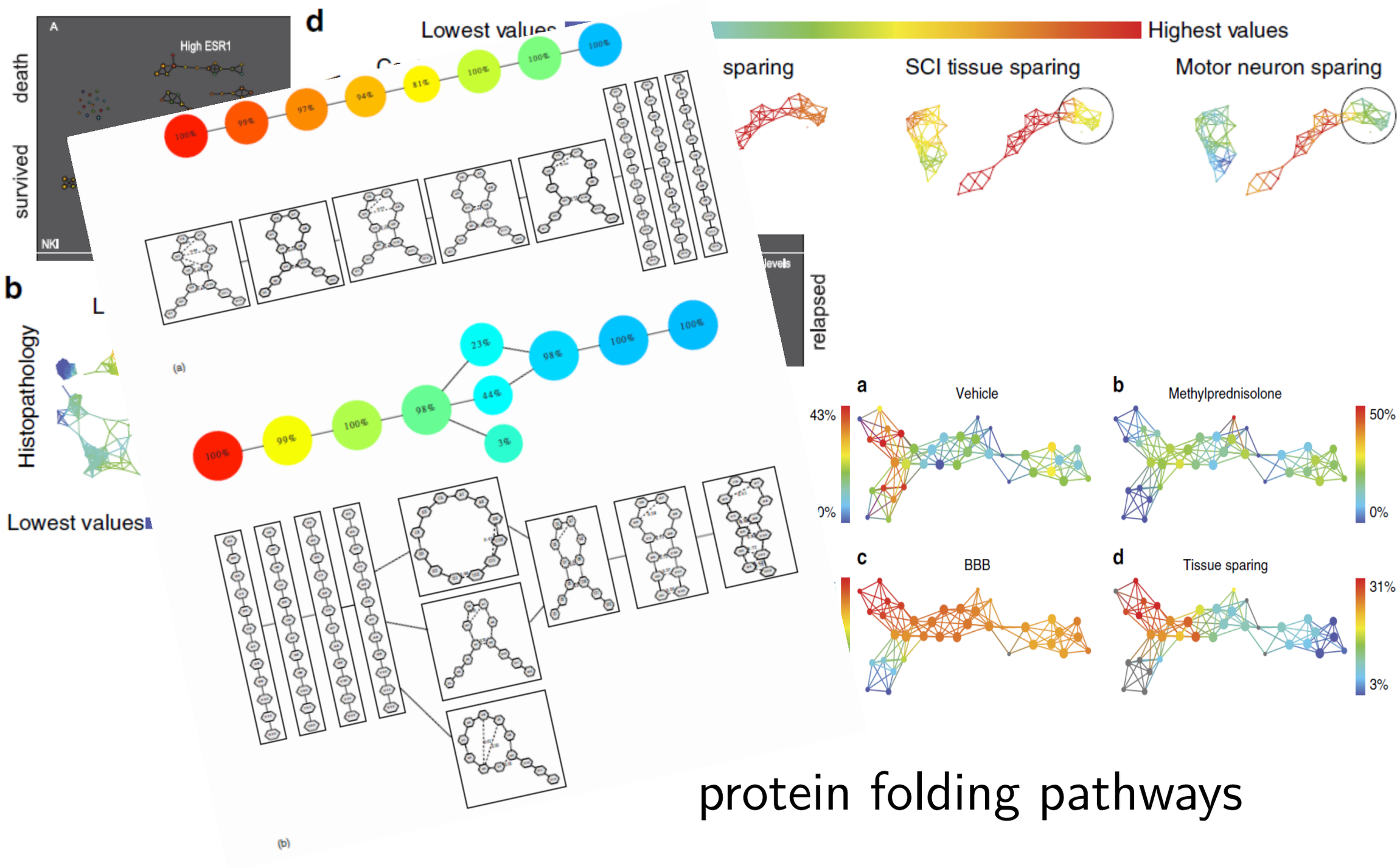
Mapper in applications





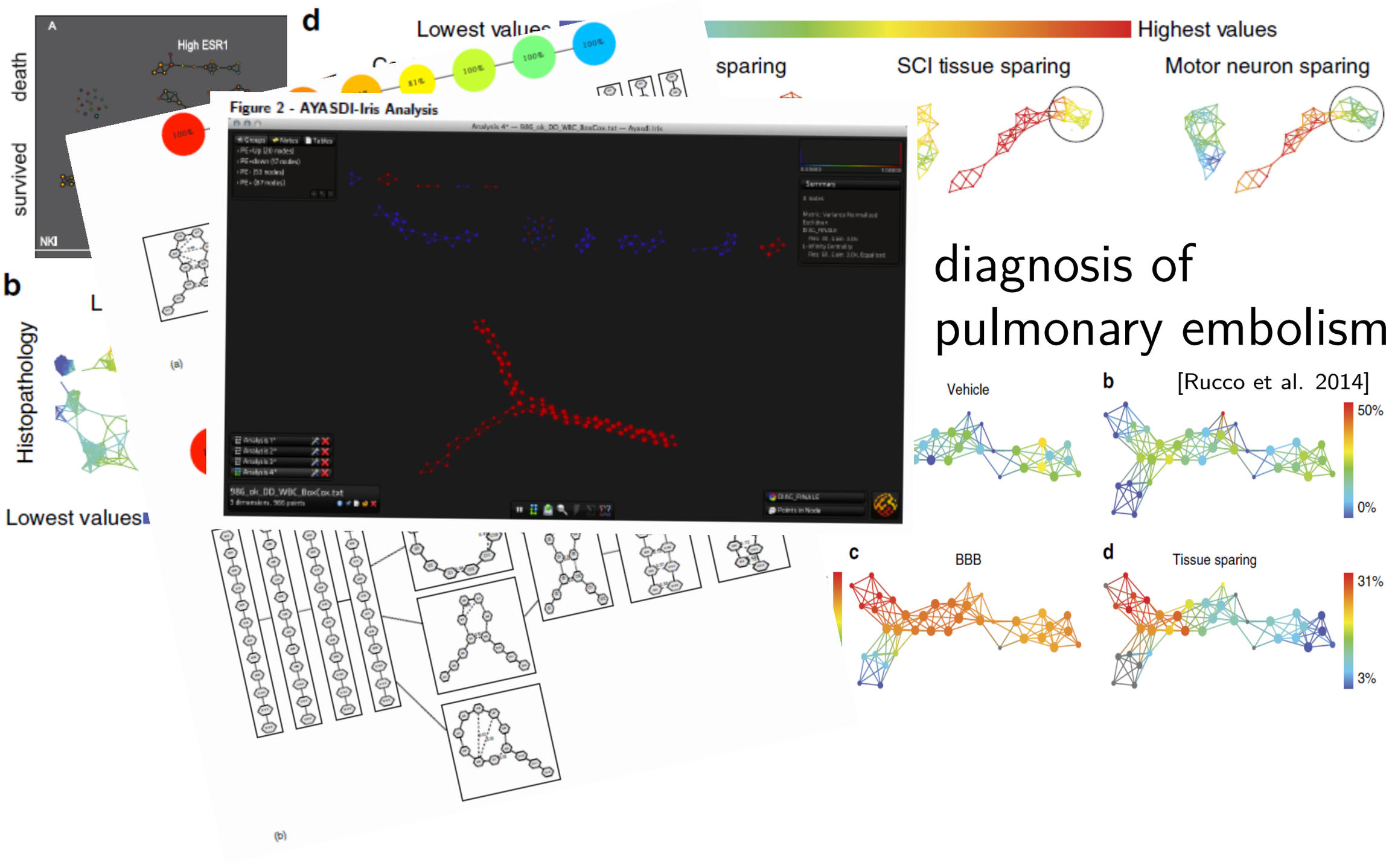
recovery from spinal cord injuries

Mapper in applications



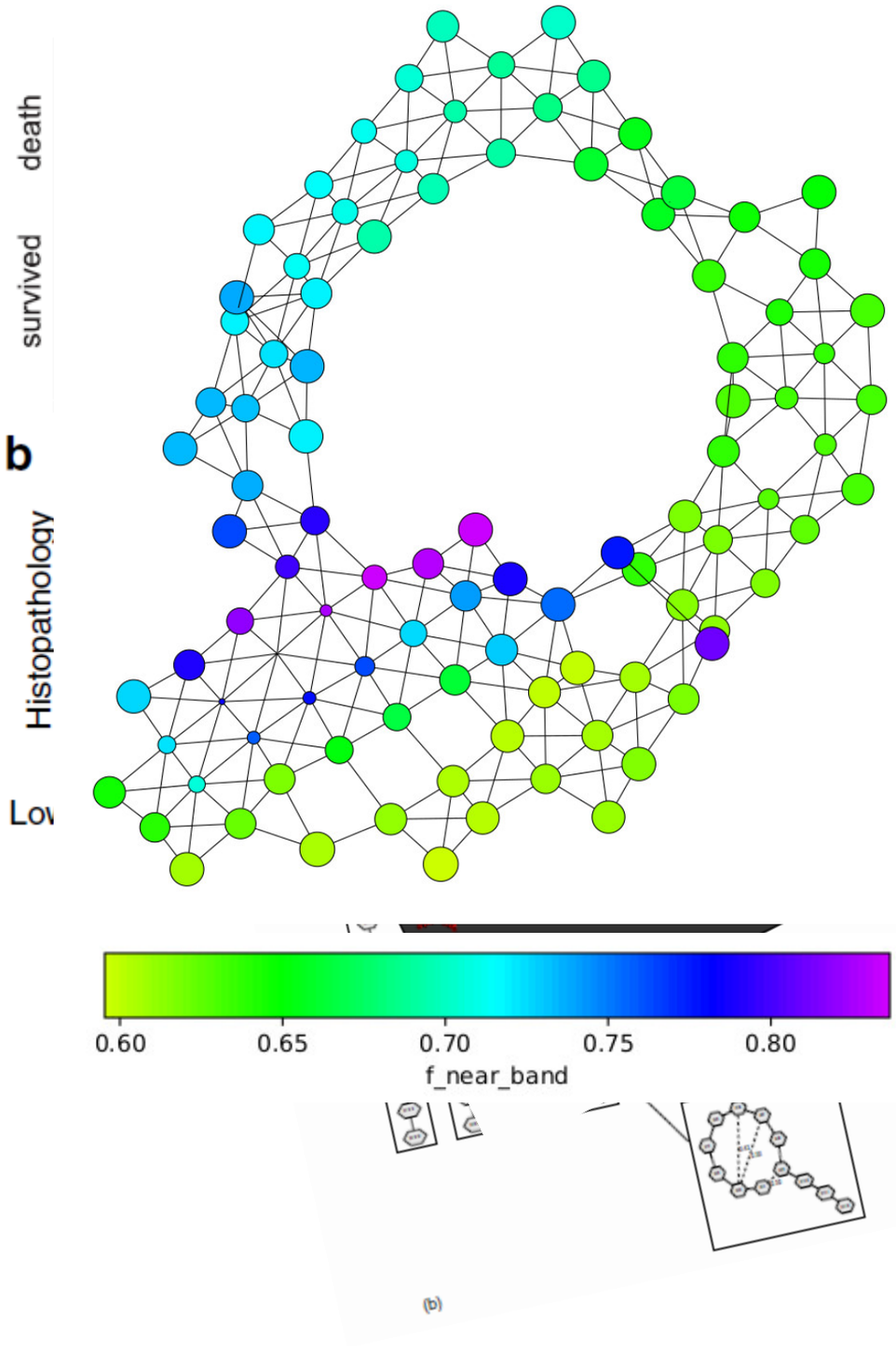
protein folding pathways

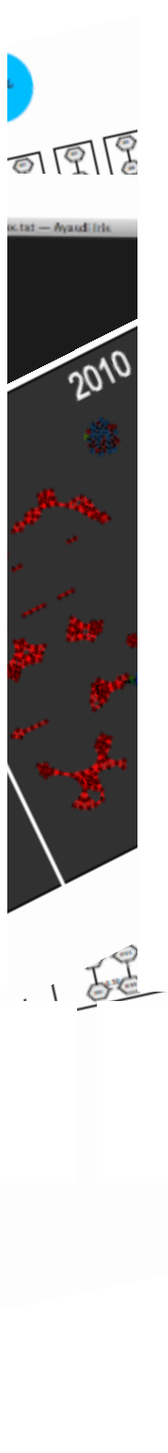
Mapper in applications

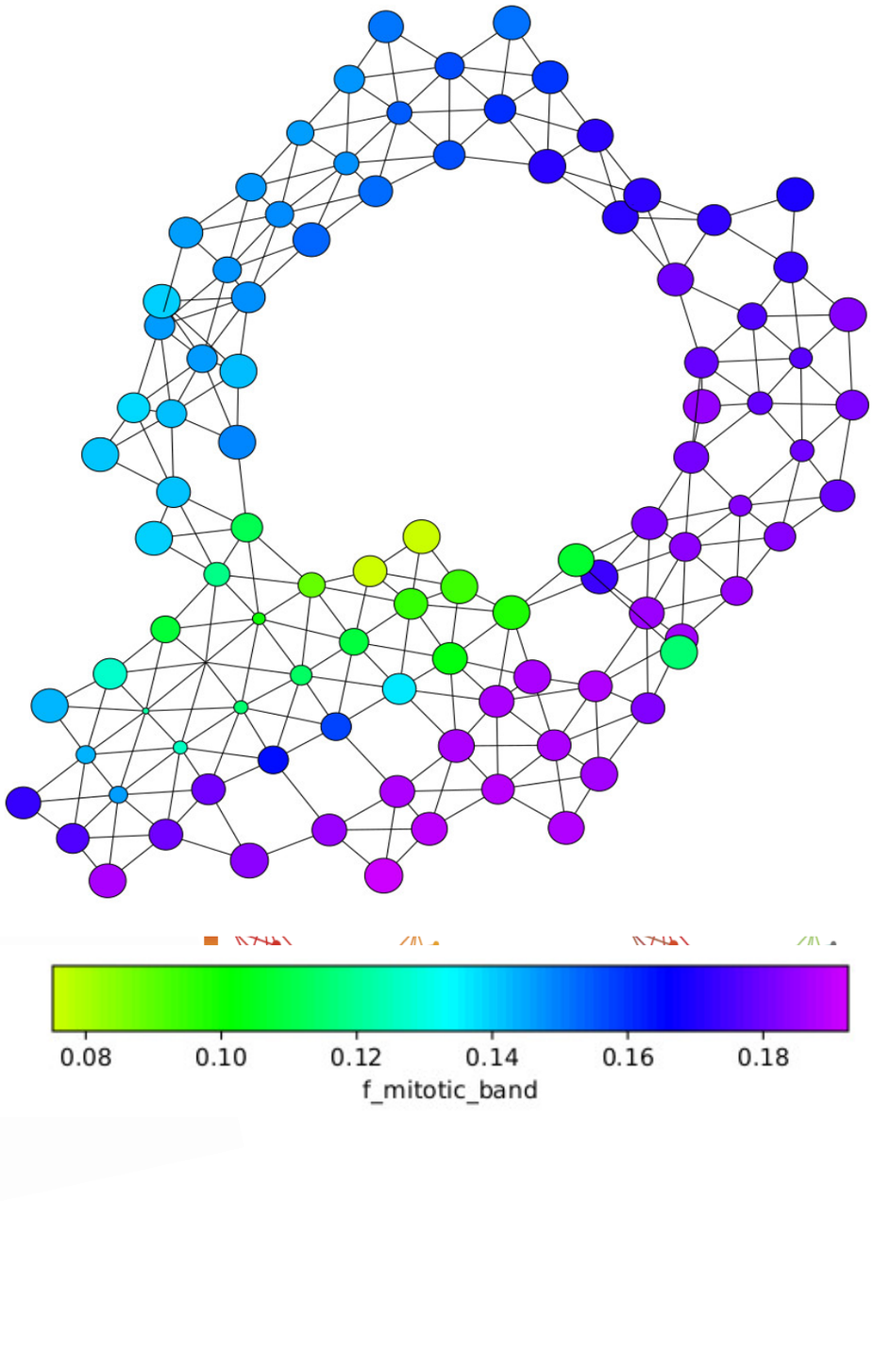


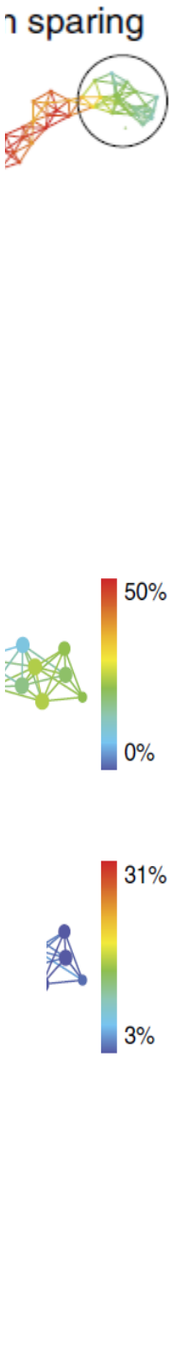
Mapper in applications

Formal identification of cell cycle

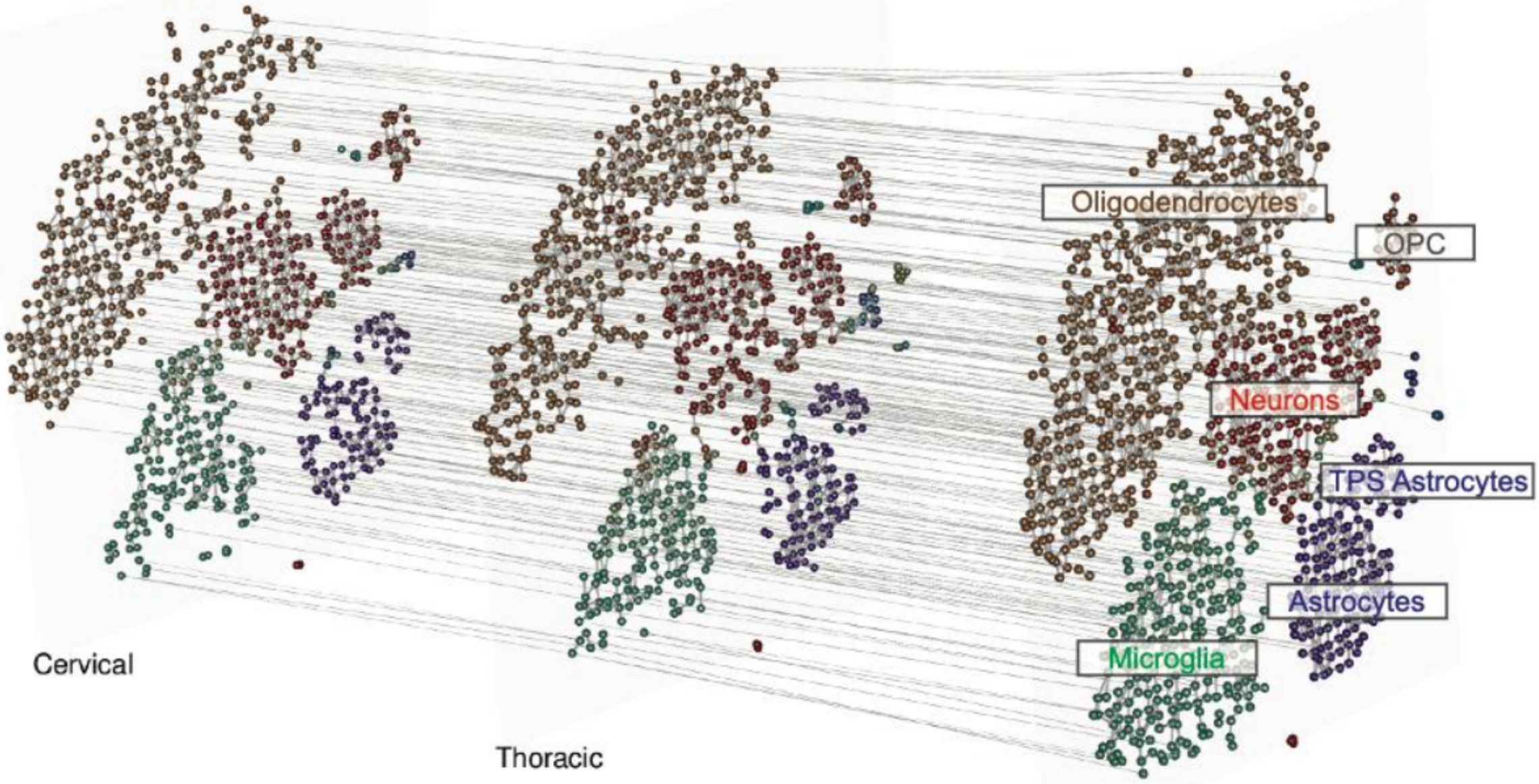






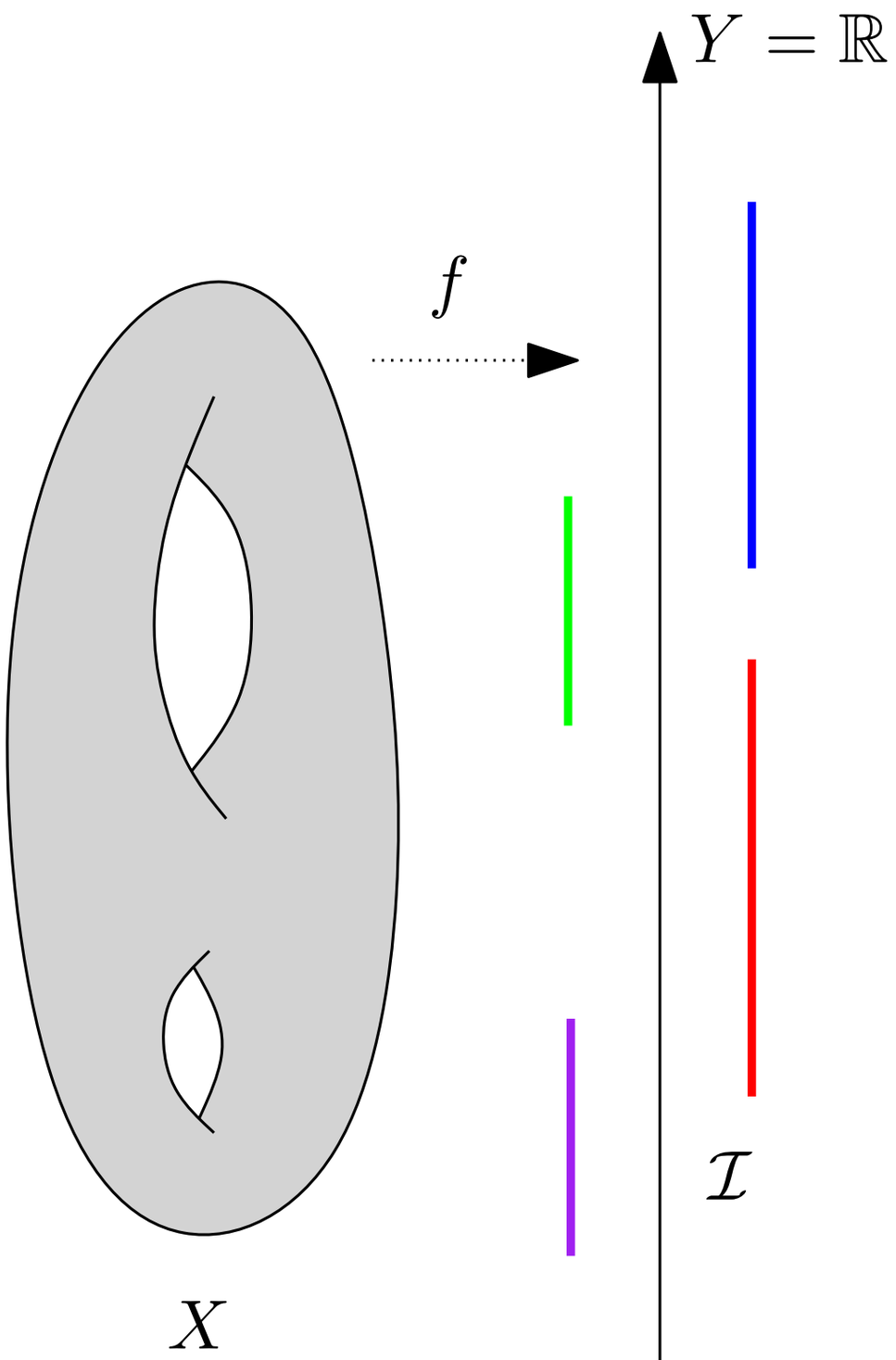


Mapper in applications

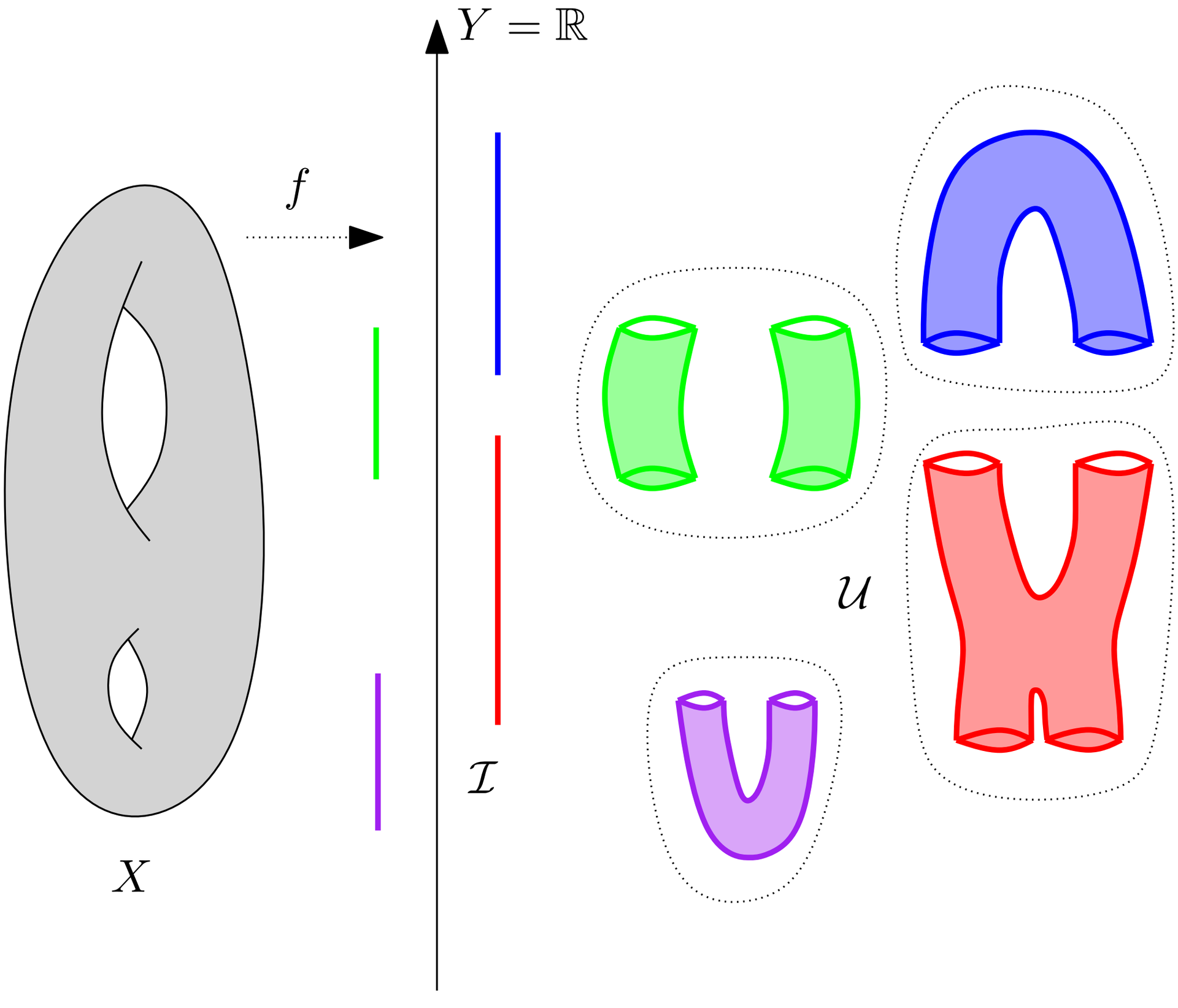


Genomic analysis of spinal cord

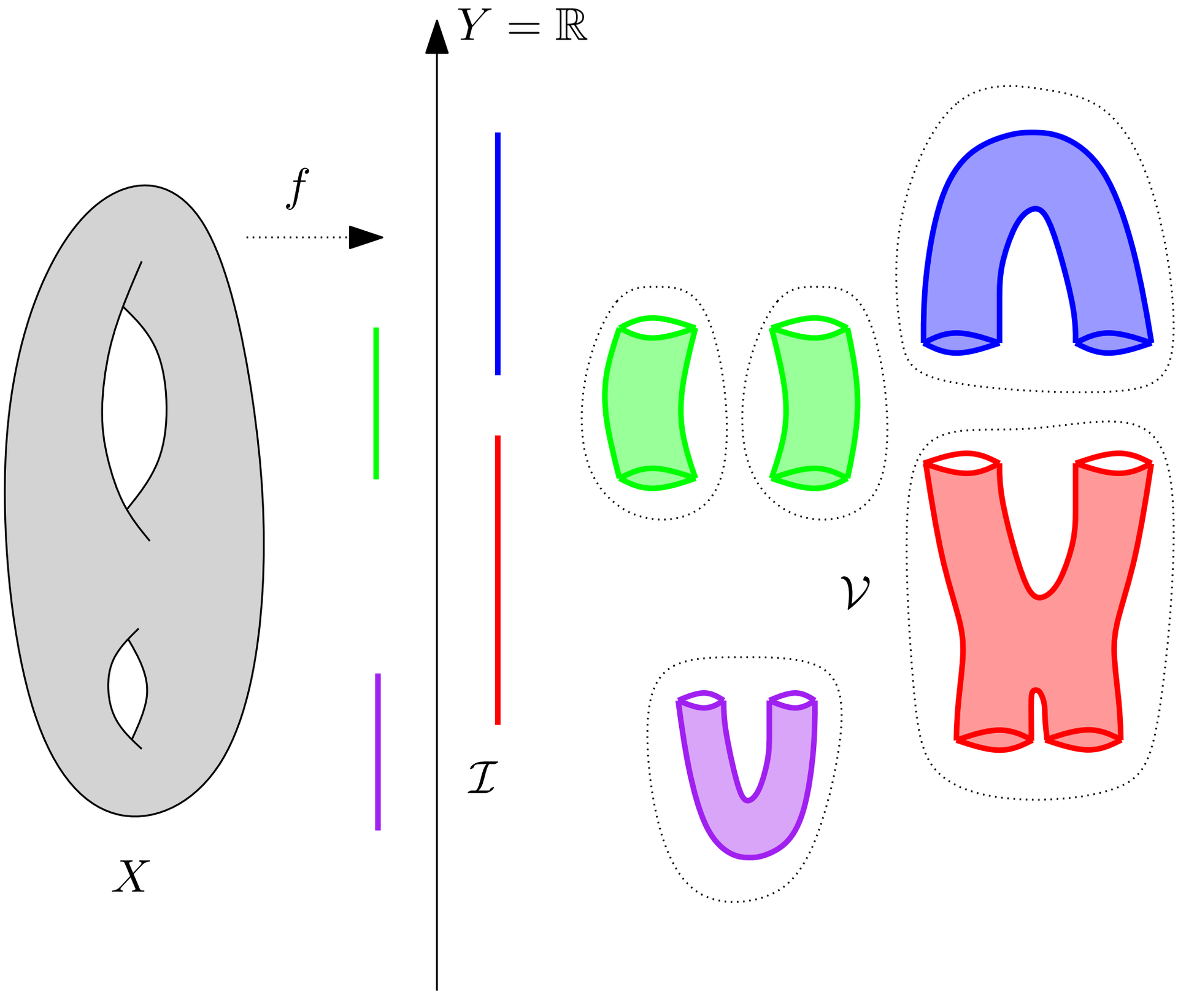
Mapper in the continuous setting



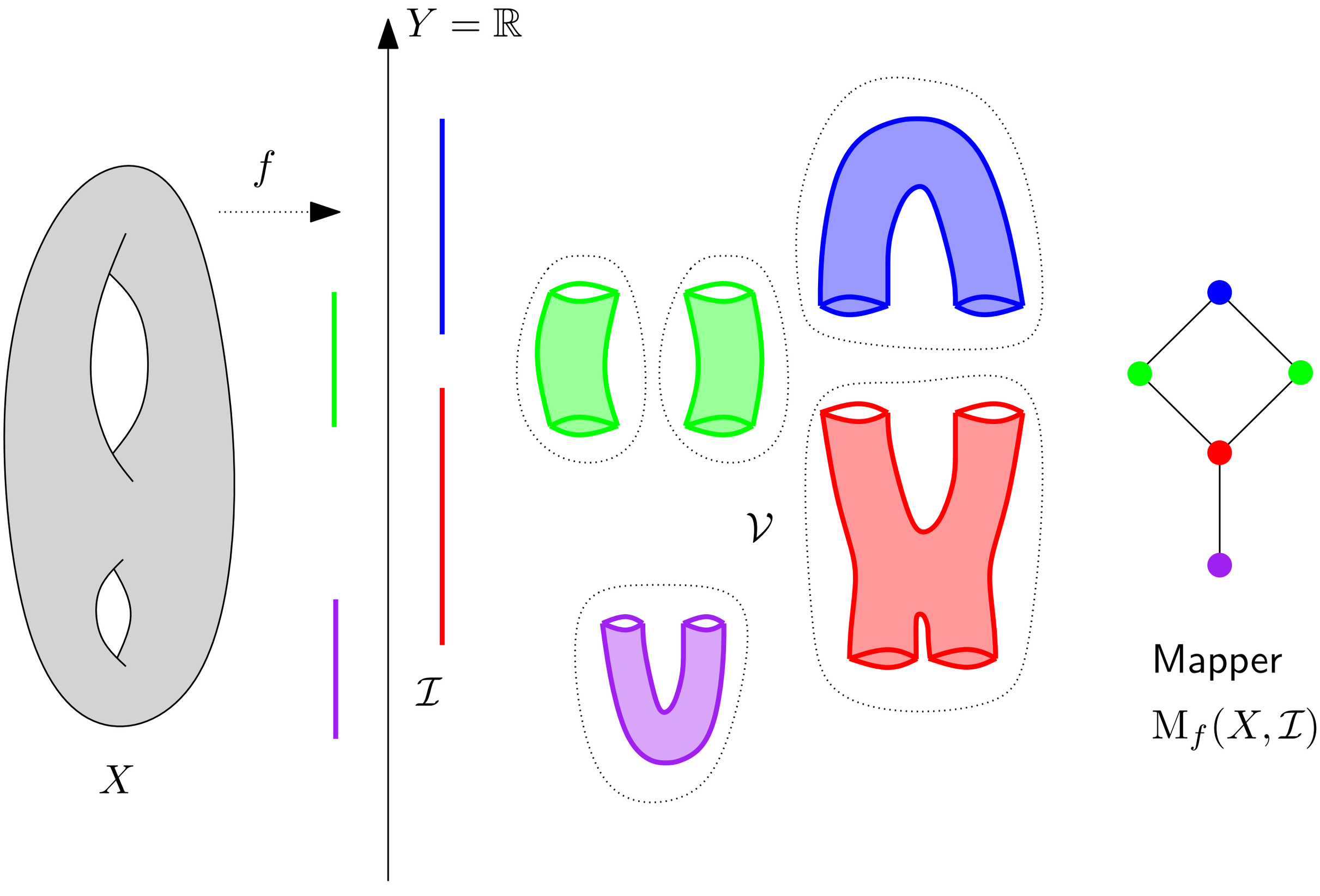
Mapper in the continuous setting



Mapper in the continuous setting



Mapper in the continuous setting



Mapper in the continuous setting

Inputs:

- topological space X
- continuous function $f : X \to Y$
- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals: $\mathrm{im}(f) \subseteq \bigcup_{I \in \mathcal{I}} I$

Method:

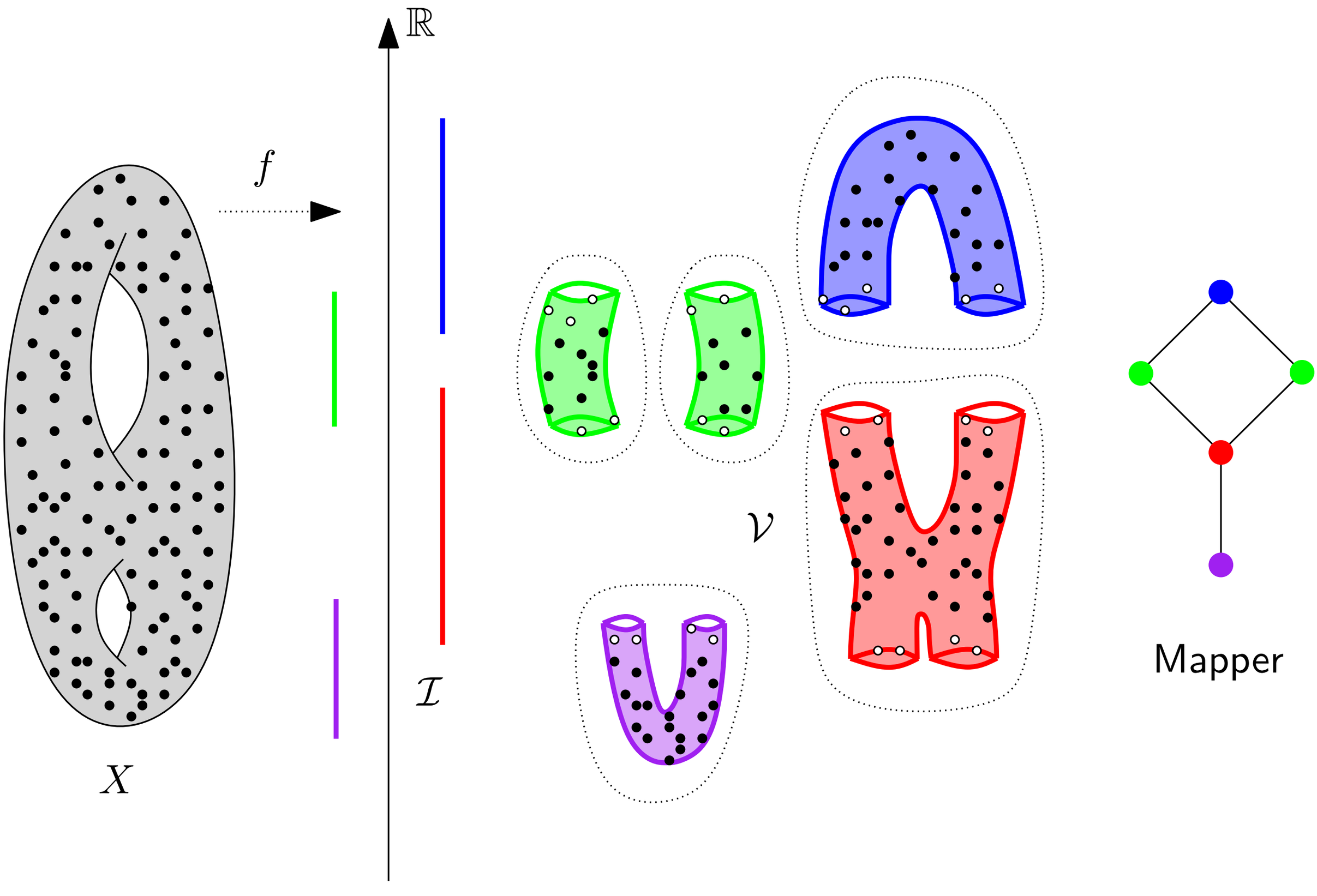
1. Compute *pullback cover* $\mathcal{U}$ of X: $\mathcal{U} = \{f^{-1}(I)\}_{I \in \mathcal{I}}$

2. Refine $\mathcal{U}$ by separating each of its elements into its various connected components in $X \to$ connected cover $\mathcal{V}$

3. The Mapper is the *nerve* of $\mathcal{V}$:

   - 1 vertex per element $V \in \mathcal{V}$
   - 1 edge per intersection $V \cap V' \neq \emptyset$, $V, V' \in \mathcal{V}$
   - 1 k-simplex per $(k+1)$-fold intersection $\bigcap_{i=0}^{k} V_i \neq \emptyset$, $V_0, \cdots, V_k \in \mathcal{V}$

Mapper in practice



Mapper in practice

Inputs:

- point cloud $P \subseteq X$ with metric d_P
- continuous function $f : X \to Y$
- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals: $\mathrm{im}(f) \subseteq \bigcup_{I \in \mathcal{I}} I$

Method:

1. Compute *pullback cover* $\mathcal{U}$ of P: $\mathcal{U} = \{f^{-1}(I)\}_{I \in \mathcal{I}}$

2. Refine $\mathcal{U}$ by separating each of its elements into its various clusters, as identified by a clustering algorithm $\to$ connected cover $\mathcal{V}$

3. The Mapper is the *nerve* of $\mathcal{V}$:

   intersections are assessed by the presence of common data points

   - 1 vertex per element $V \in \mathcal{V}$
   - 1 edge per intersection $V \cap V' \neq \emptyset$, $V, V' \in \mathcal{V}$
   - 1 k-simplex per $(k+1)$-fold intersection $\bigcap_{i=0}^{k} V_i \neq \emptyset$, $V_0, \cdots, V_k \in \mathcal{V}$

Mapper in practice

Parameters:

- function $f : P \to \mathbb{R}$

- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals

- clustering algorithm $\mathcal{C}$

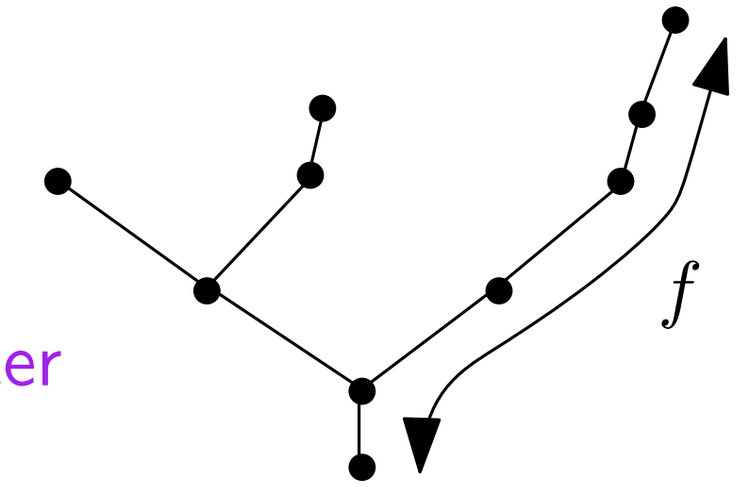
Mapper in practice

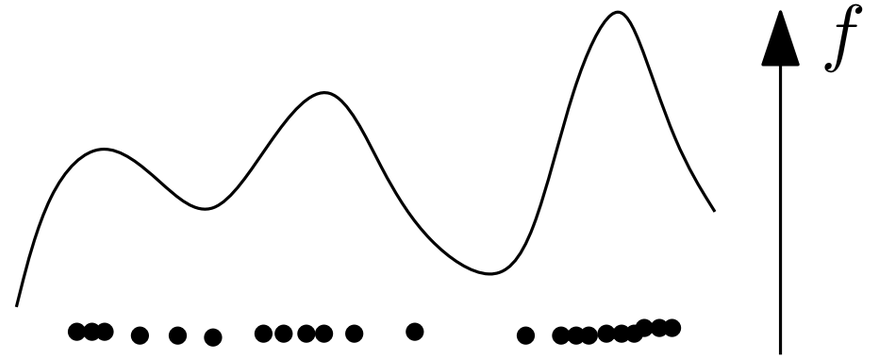
Parameters:

- function $f : P \to \mathbb{R}$ ← lens or filter

- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals

- clustering algorithm $\mathcal{C}$





Classical choices:

- density estimates
- centrality $f(x) = \sum_{y \in X} d(x, y)$
- eccentricity $f(x) = \max_{y \in X} d(x, y)$
- PCA coordinates
- Eigenfunctions of graph laplacians.
- Functions detecting outliers.
- Distance to a root point.
- Prior knowledge

Mapper in practice

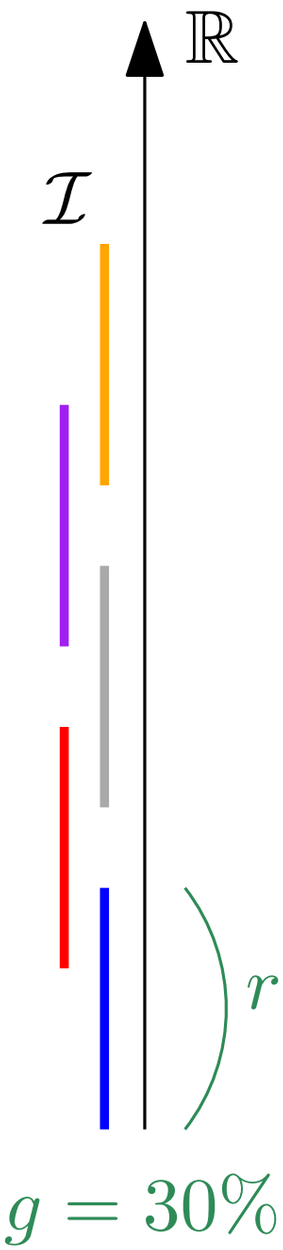
Parameters:

- function $f : P \to \mathbb{R}$
- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals
- clustering algorithm $\mathcal{C}$

range scale

Uniform cover:

- resolution / granularity: r (diameter of intervals)
- gain: g (percentage of overlap)



Mapper in practice

Parameters:

- function $f : P \to \mathbb{R}$

- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals

- clustering algorithm $\mathcal{C}$

range scale

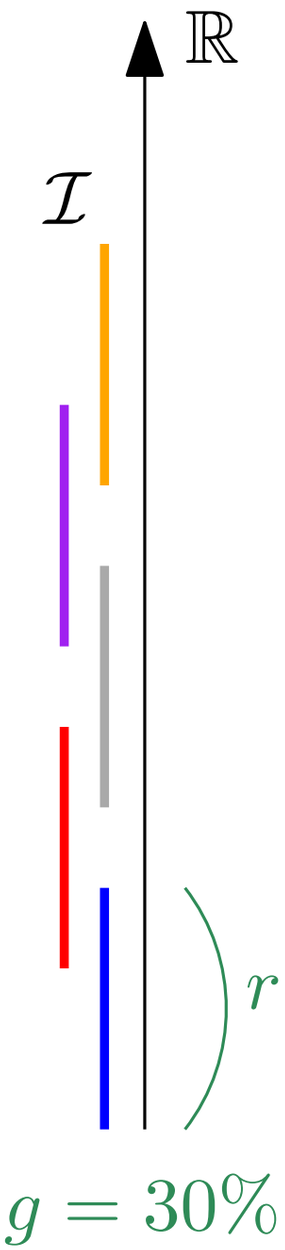
Uniform cover:

- resolution / granularity: r (diameter of intervals)
- gain: g (percentage of overlap)

Intuition:

- small $r \to$ finer resolution, more nodes.
- large $r \to$ rougher resolution, less nodes.

- small $g \to$ less connectivity, nerve dimension small.
- large $g \to$ more connectivity, nerve dimension large.



Mapper in practice

Parameters:

- function $f : P \to \mathbb{R}$

- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals

- clustering algorithm $\mathcal{C}$

Classical choices:

- any clustering algorithm works

- different clustering algorithms/parameters for each preimage

- for theoretical reasons, we prefer to work with

  hierarchical clustering with (predefined) neighborhood size δ

geometric scale

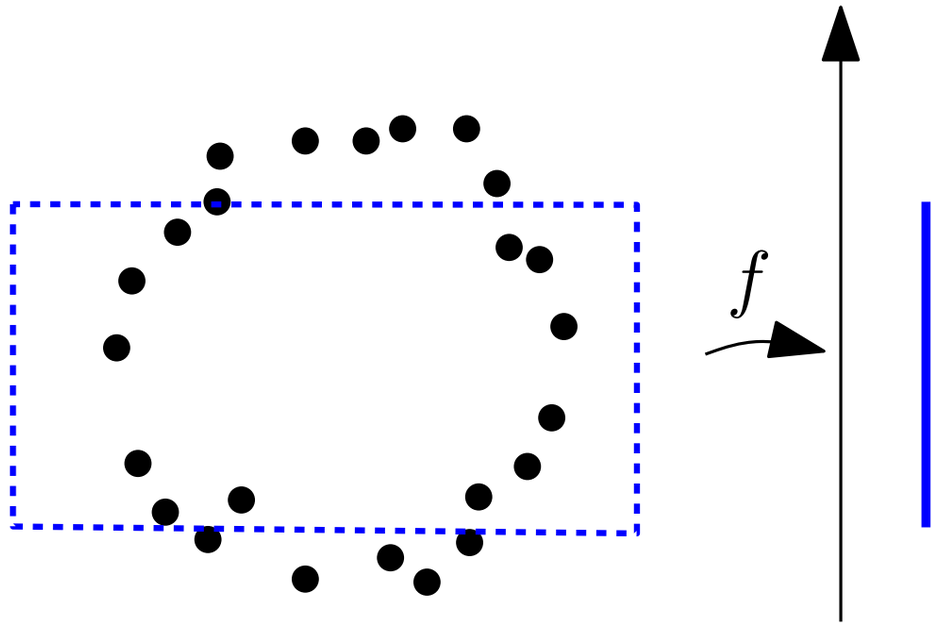
Mapper in practice

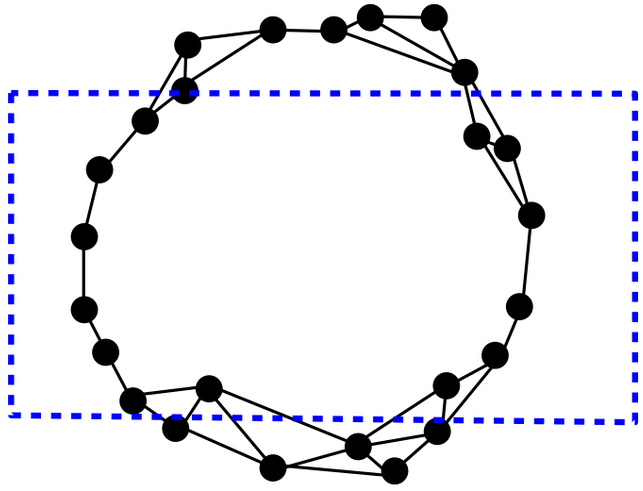
Parameters:

- function $f : P \to \mathbb{R}$

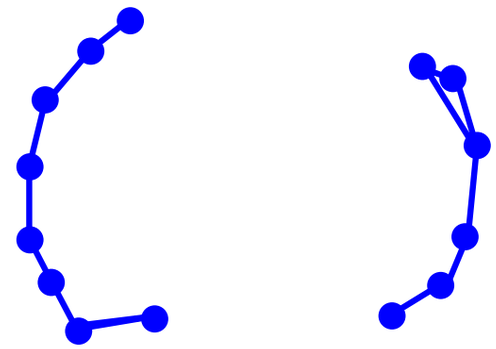
- cover $\mathcal{I}$ of $\mathrm{im}(f)$ by open intervals

- clustering algorithm $\mathcal{C}$



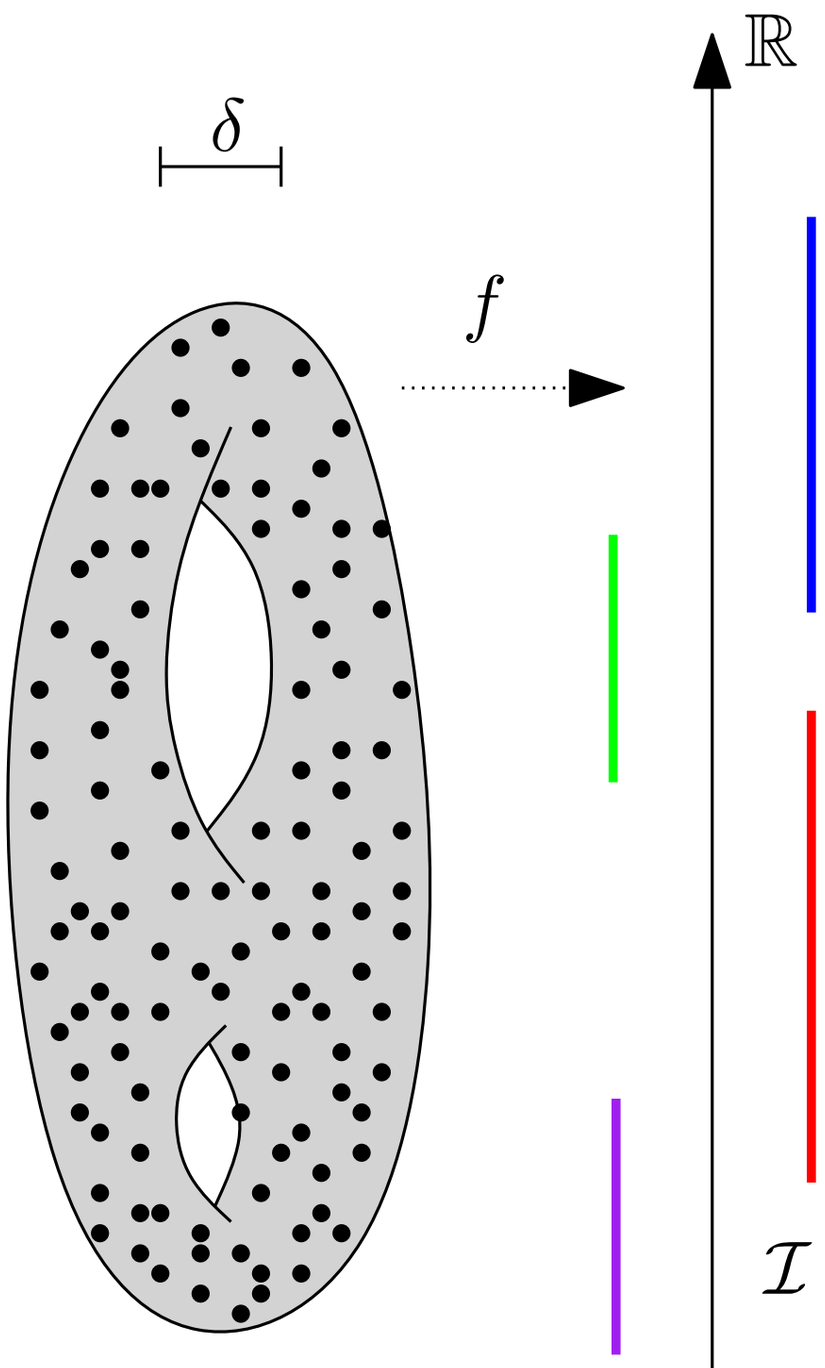


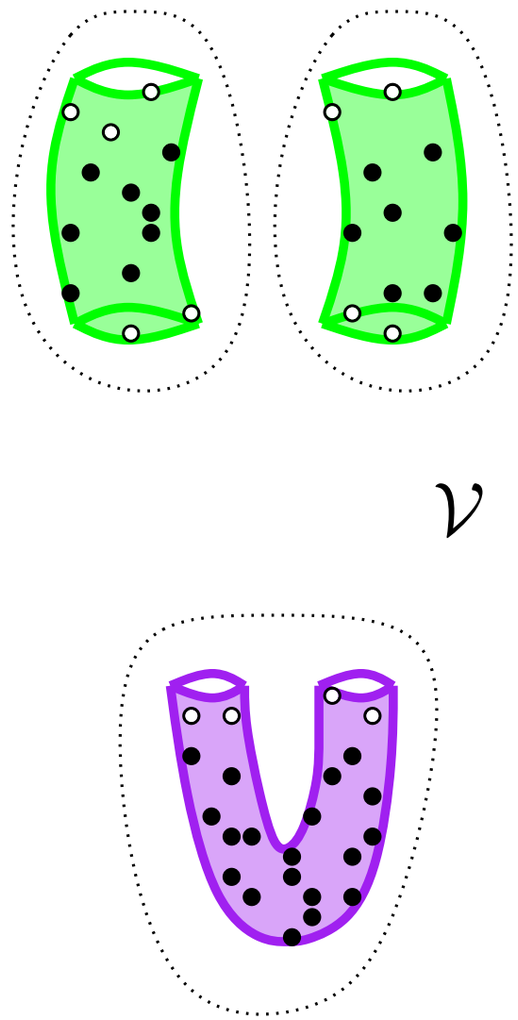
Build a neighboring graph (kNN,...)

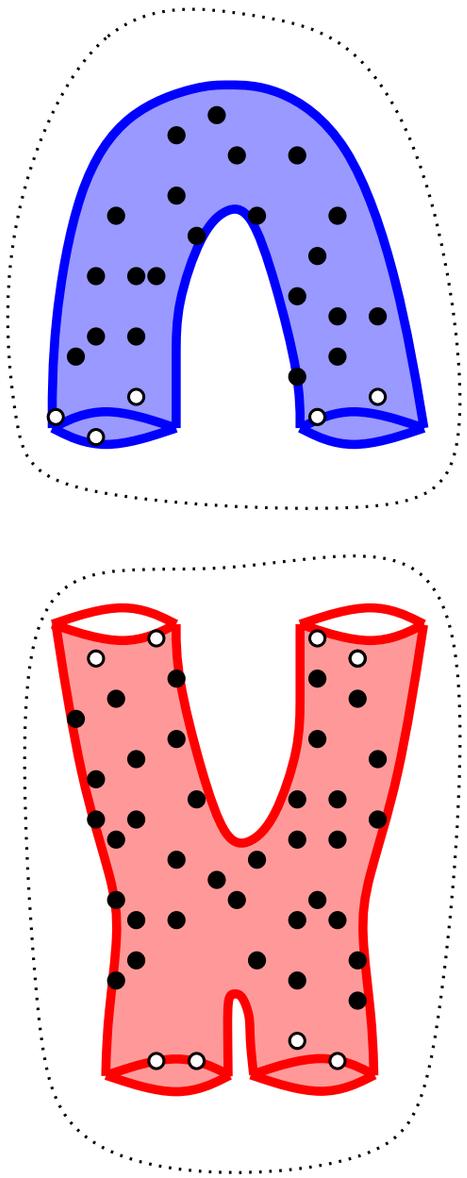


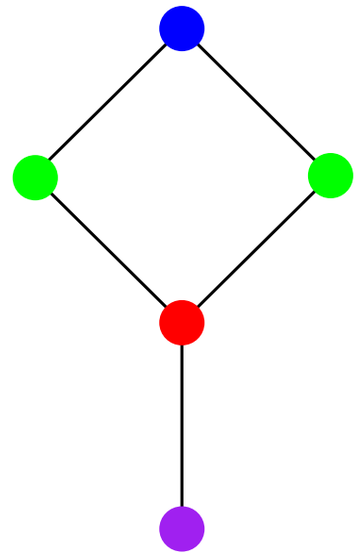
Take the connected components of the subgraph spanned by the vertices in the preimage $f^{-1}(U)$.

Mapper in practice









Mapper

$\mathrm{M}^{\bullet}_{f,\delta}(P,\mathcal{I})$

$G_\delta = \delta$-neighborhood graph

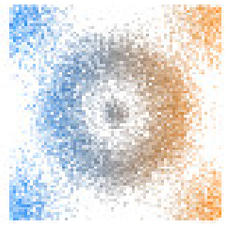
Choice of parameters

In practice, trial-and-error:

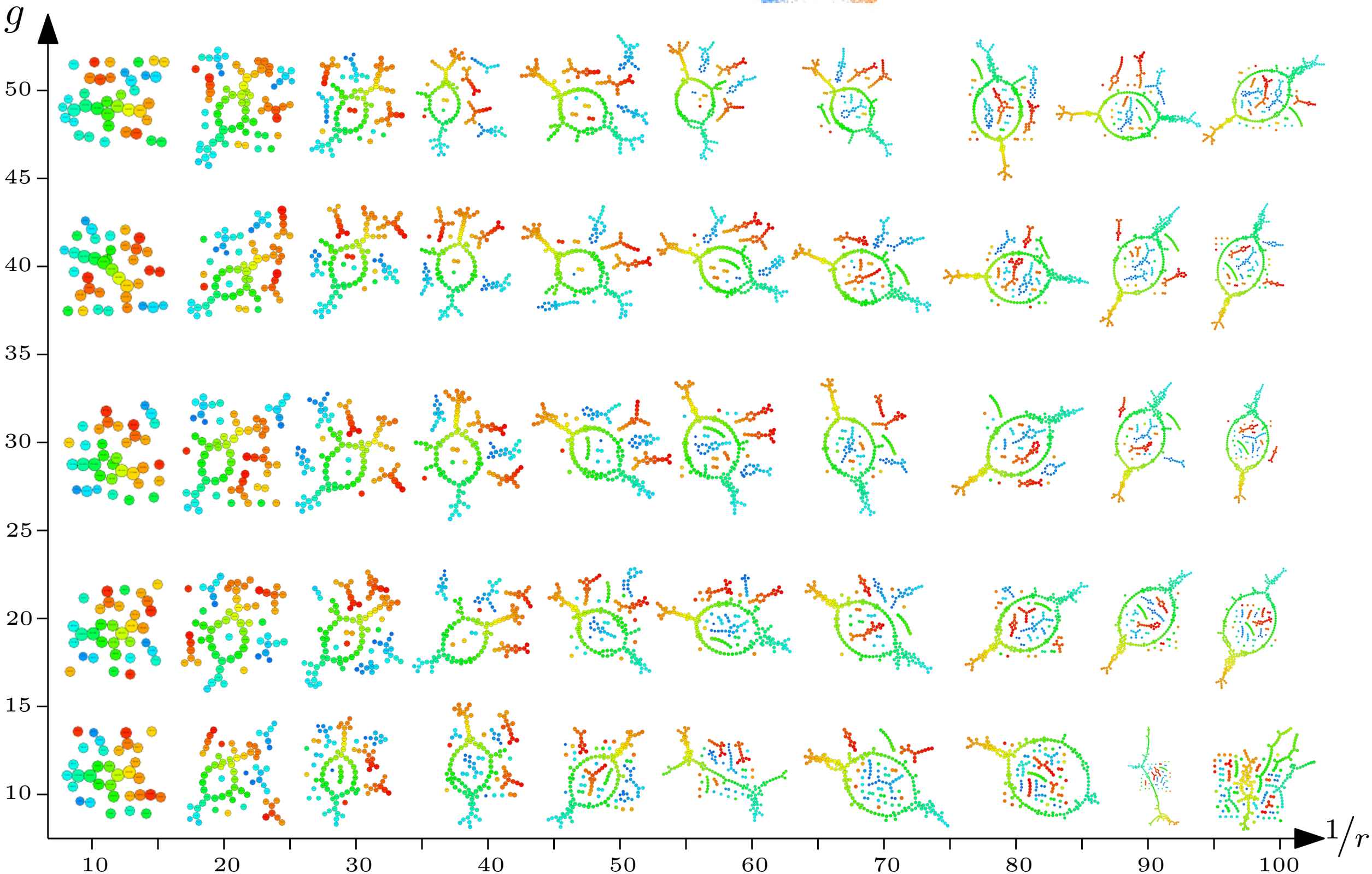
high-dimensional data sets[40,48]. This is performed automatically within the software, by deploying an ensemble machine learning algorithm that iterates through overlapping subject bins of different sizes that resample the metric space (with replacement), thereby using a combination of the metric location and similarity of subjects in the network topology. After performing millions of iterations, the algorithm returns the most stable, consensus vote for the resulting 'golden network' (Reeb graph), representing the multidimensional data shape[12,40].

[*Topological Data Analysis for Discovery in Preclinical Spinal Cord Injury and Traumatic Brain Injury*, Nielson et al., Nature, 2015]

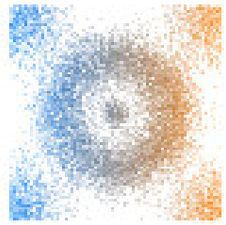
Choice of parameters



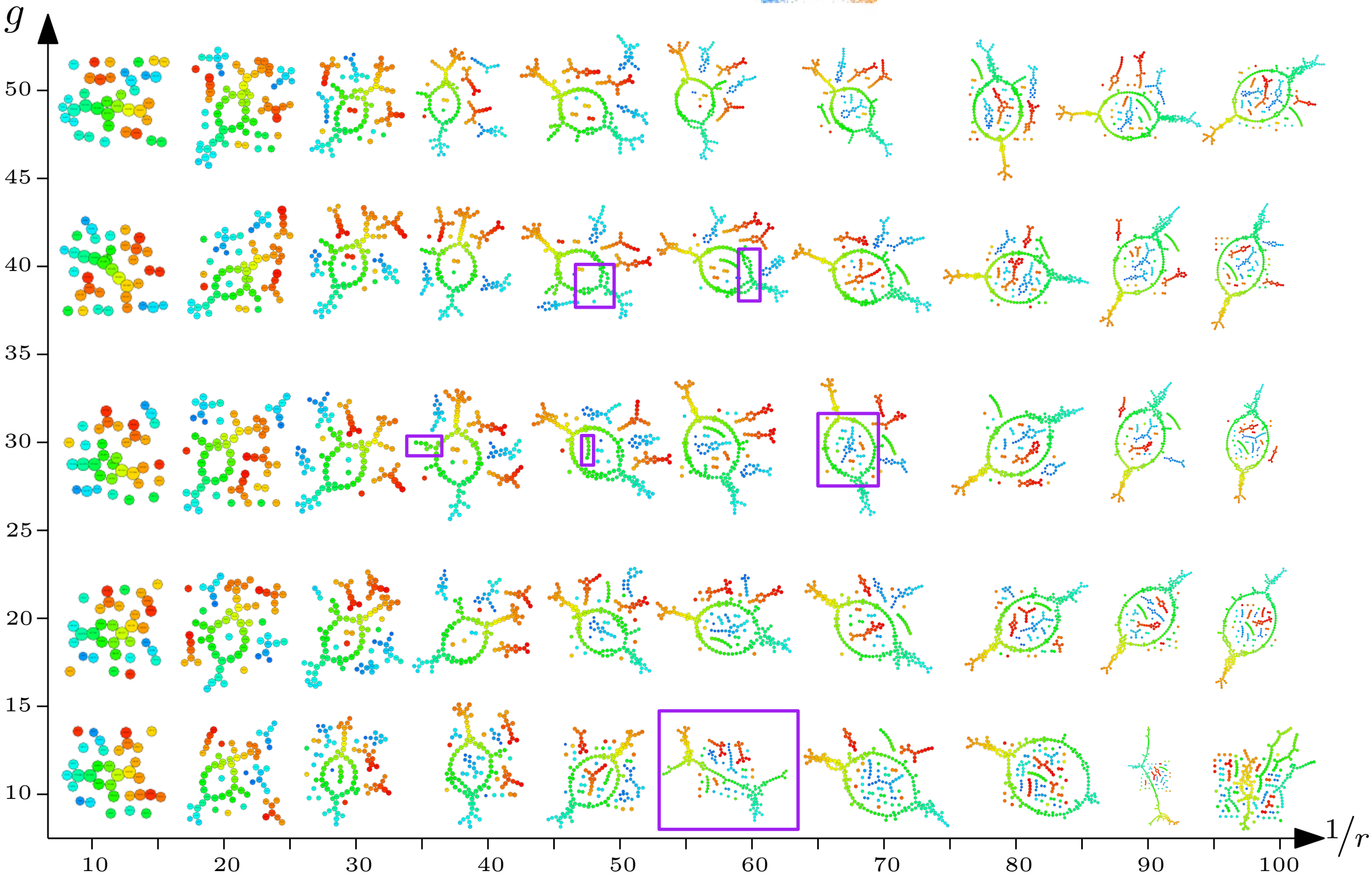
$f = f_x$, $\delta = 1\%$



Choice of parameters



$f = f_x$, $\delta = 1\%$



I. ToMATo algorithm

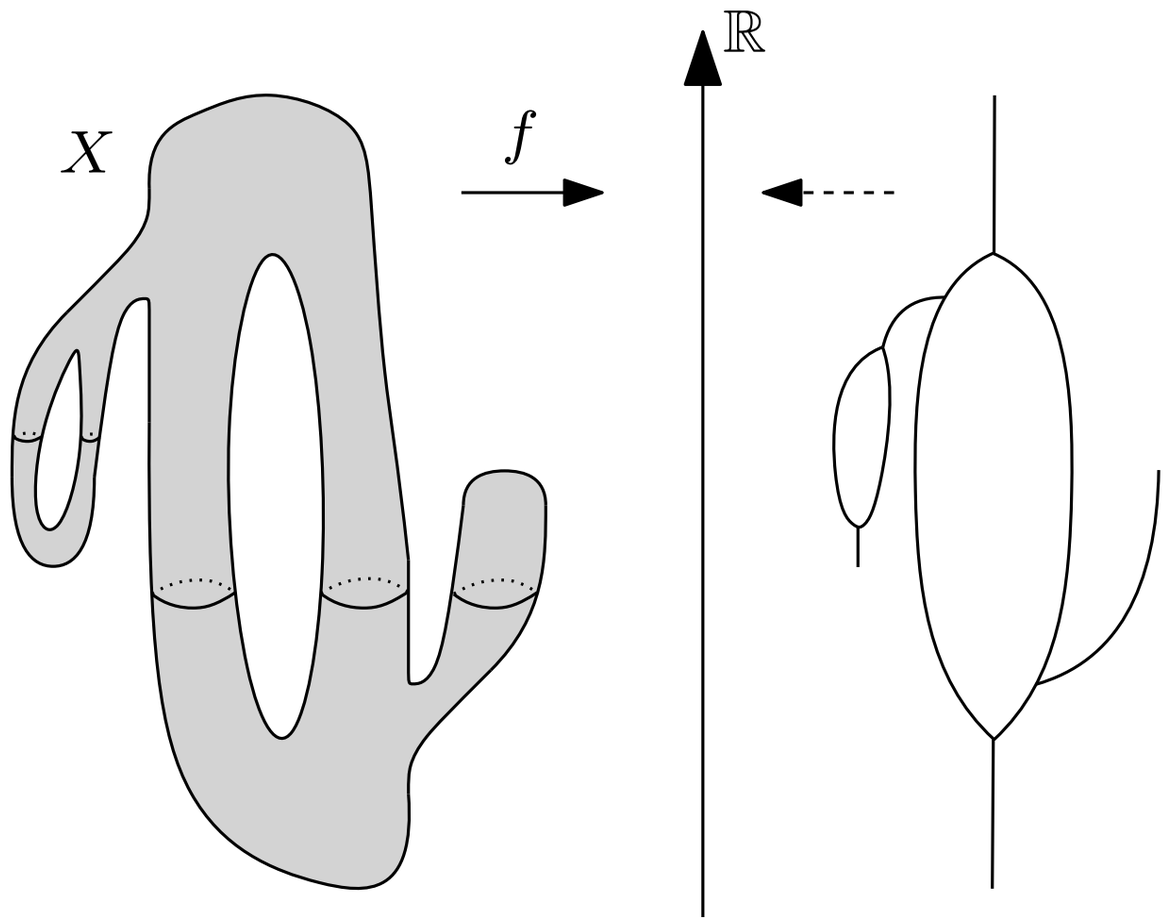
1. Introduction to hierarchical and mode-seeking clustering
2. ToMATo algorithm and guarantees

II. Mapper algorithm

1. Reeb spaces and Mappers
2. Confidence intervals

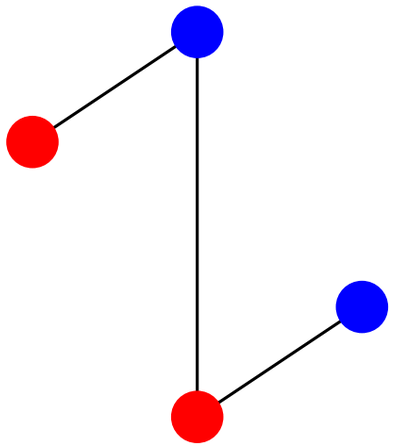
Reeb Graph

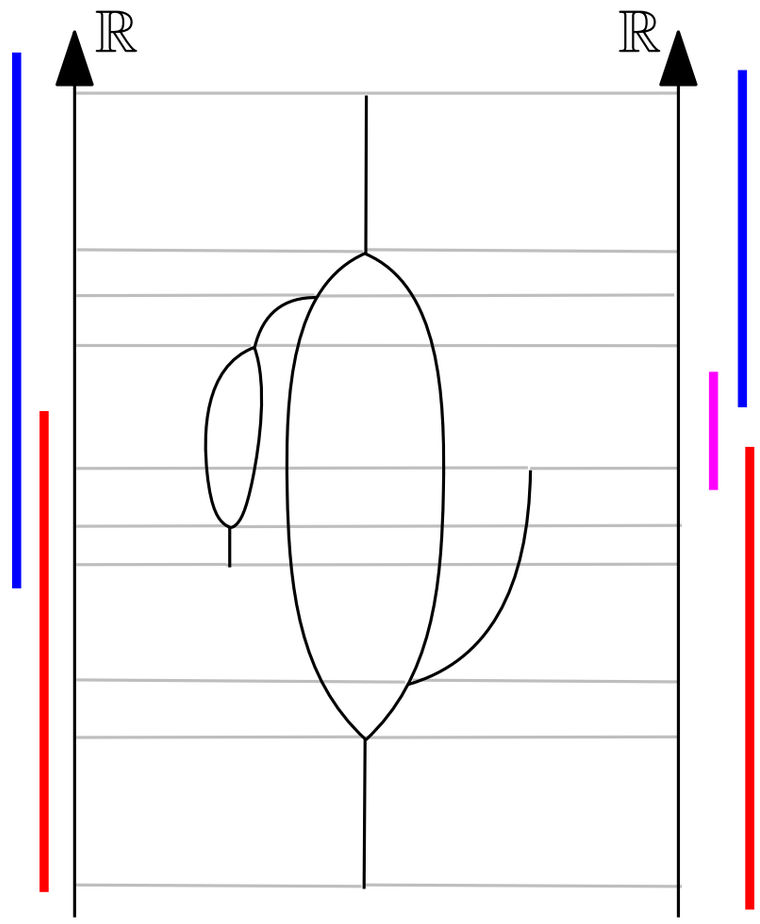
Reeb graph $\sim$ Mapper with extremely small resolution

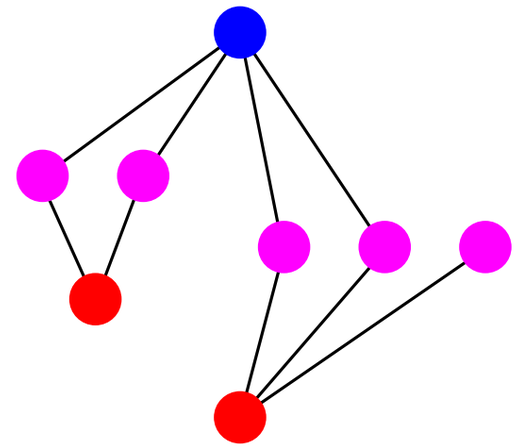


Reeb Graph

Mapper $\sim$ *pixelized* Reeb graph





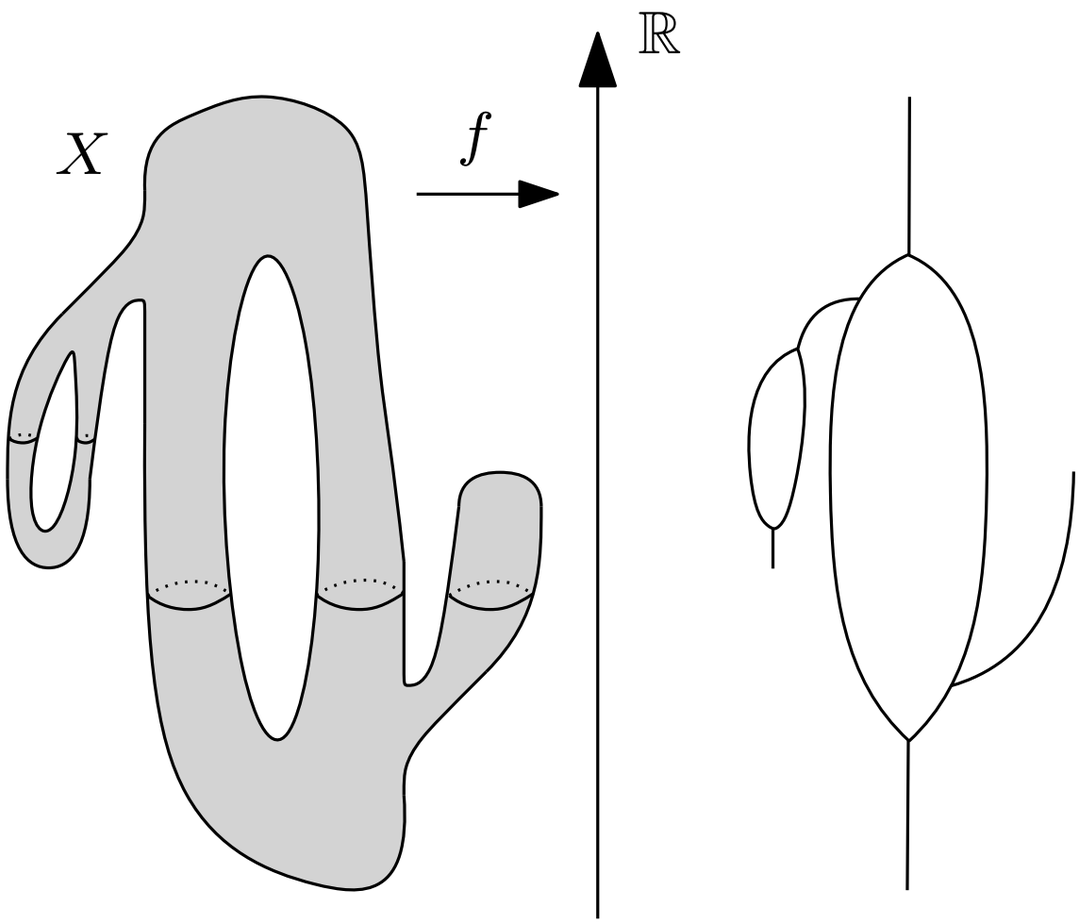


Reeb Graph

[*Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique*, Reeb, C. R. Acad. Sci. Paris, 1946]

$x \sim y \iff [\ f(x) = f(y)$ and x, y belong to same cc of $f^{-1}(\{f(x)\})\ ]$

Def: $\mathrm{R}_f(X) := X/\sim$

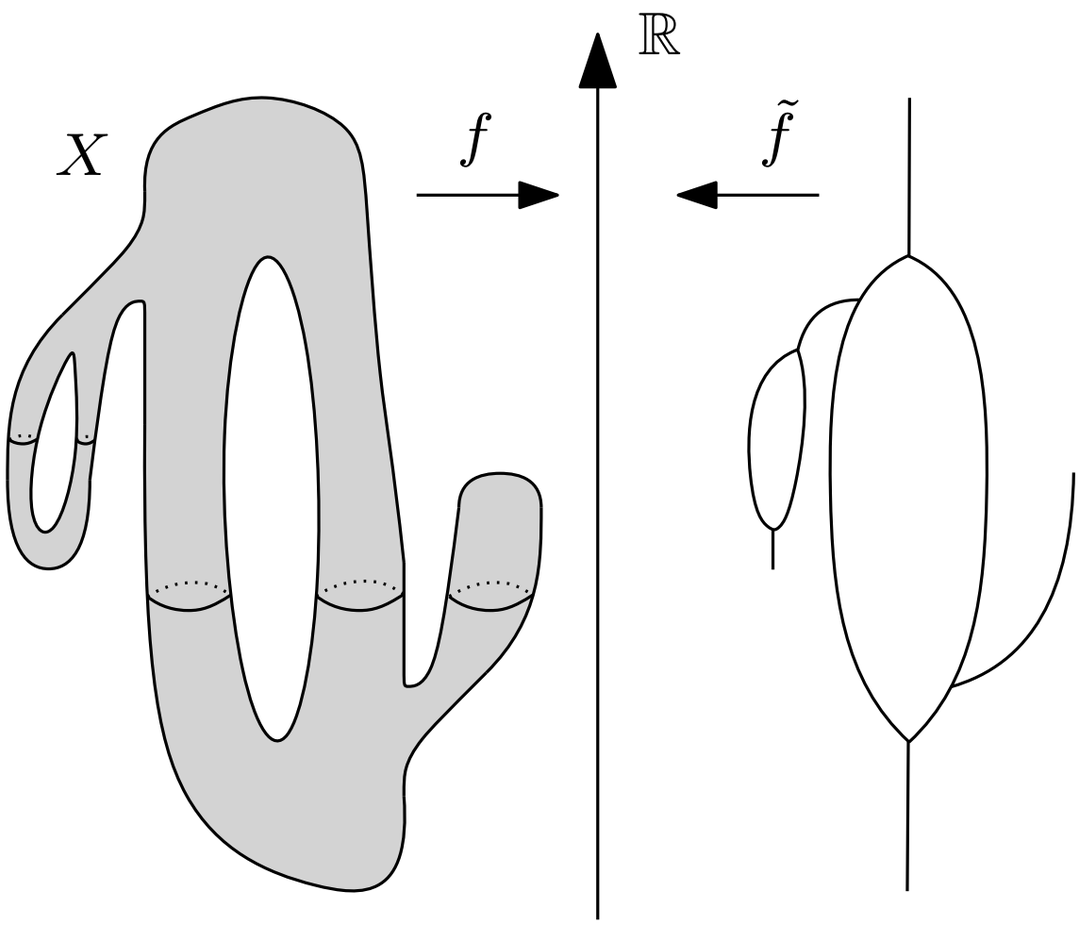


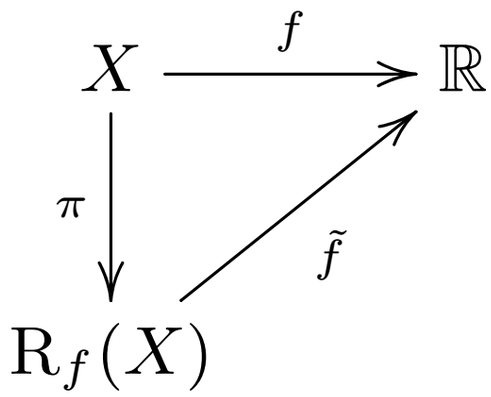
Reeb Graph

[*Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique*, Reeb, C. R. Acad. Sci. Paris, 1946]

$x \sim y \iff [\ f(x) = f(y)$ and x, y belong to same cc of $f^{-1}(\{f(x)\})\ ]$

Def: $\mathrm{R}_f(X) := X/\sim$



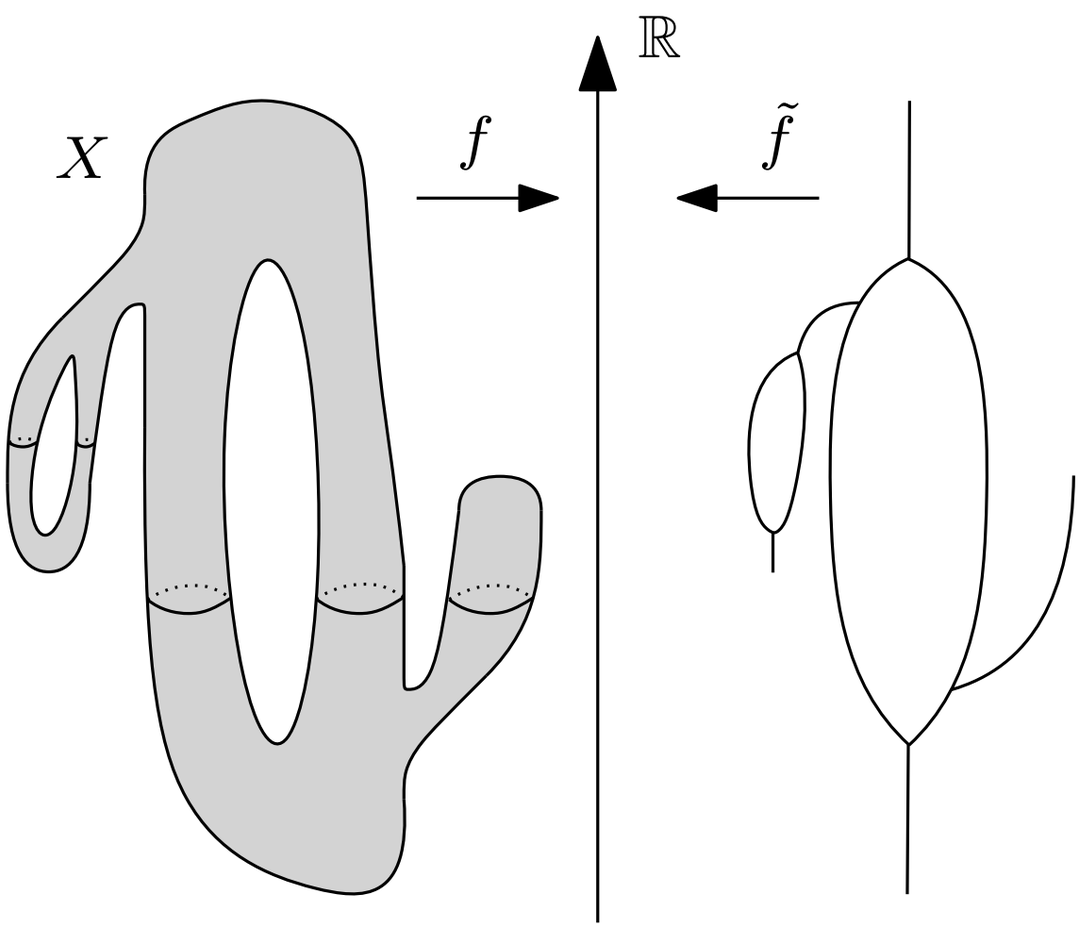


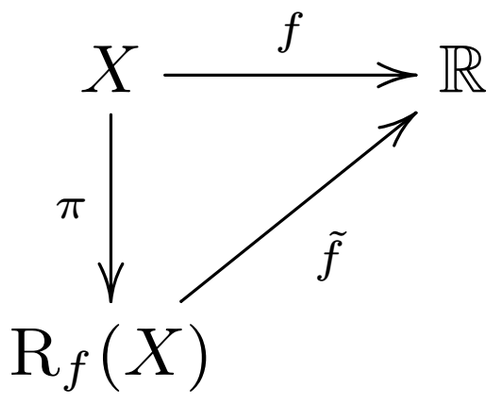
Reeb Graph

[*Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique*, Reeb, C. R. Acad. Sci. Paris, 1946]

$x \sim y \iff [\ f(x) = f(y)$ and x, y belong to same cc of $f^{-1}(\{f(x)\})\ ]$

Def: $\mathrm{R}_f(X) := X/\sim$





Prop: $\mathrm{R}_f(X)$ is a graph when (X, f) is Morse or of **Morse type**.

Prop: $H_*(\mathrm{R}_f(X)) = H_*(X)/\bar{H}_*(X)$.

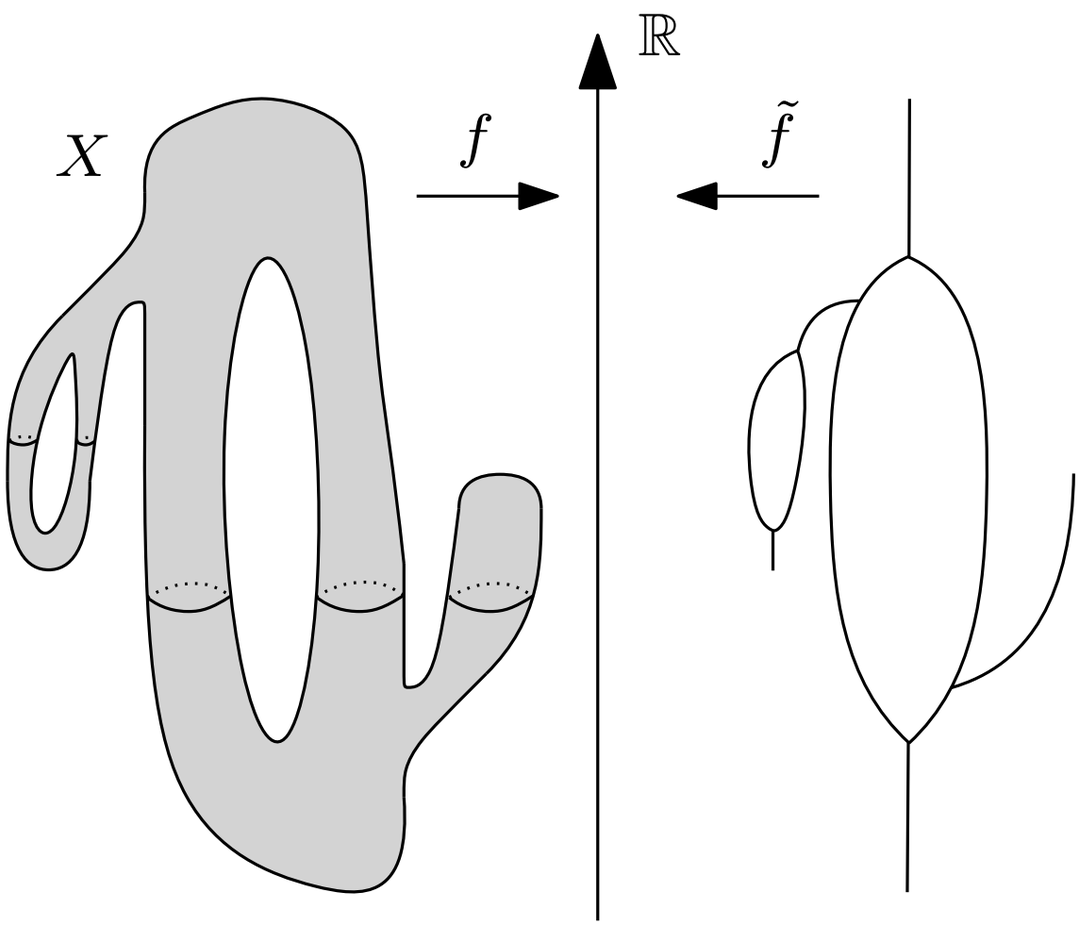
[*Reeb Graphs: Approximation and Persistence*, Dey, Wang, DCG, 2013]

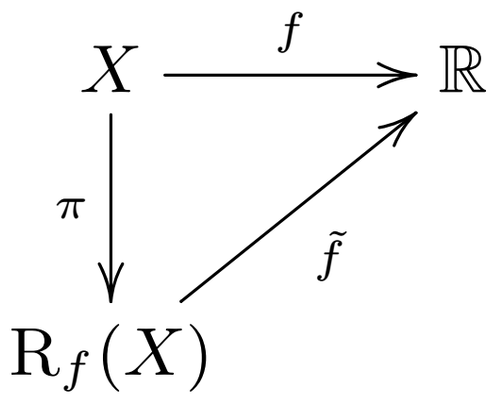
Reeb Graph

[*Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique*, Reeb, C. R. Acad. Sci. Paris, 1946]

$x \sim y \iff [\ f(x) = f(y)$ and x, y belong to same cc of $f^{-1}(\{f(x)\})\ ]$

Def: $\mathrm{R}_f(X) := X/\sim$





Prop: $\mathrm{R}_f(X)$ is a graph when (X, f) is Morse or of **Morse type**.

horizontal homology $\sim$ 'those homology classes that are included in a finite union of levelsets of f'

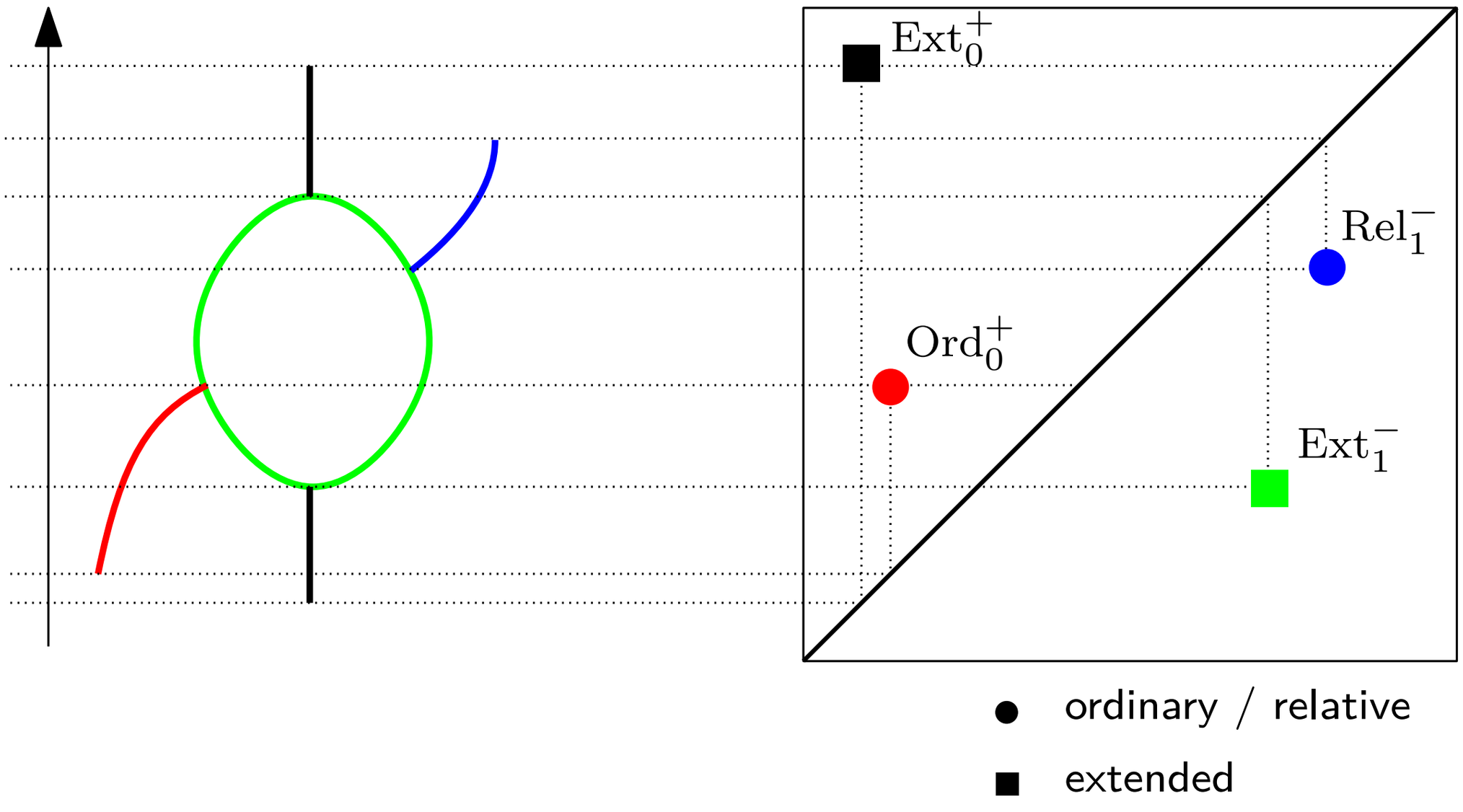
Prop: $H_*(\mathrm{R}_f(X)) = H_*(X)/\bar{H}_*(X)$.

[*Reeb Graphs: Approximation and Persistence*, Dey, Wang, DCG, 2013]

Graph Descriptor

Thm: $D_{\tilde{f}}$ provides a **bag-of-features** descriptor for $\mathrm{R}_f(X)$:

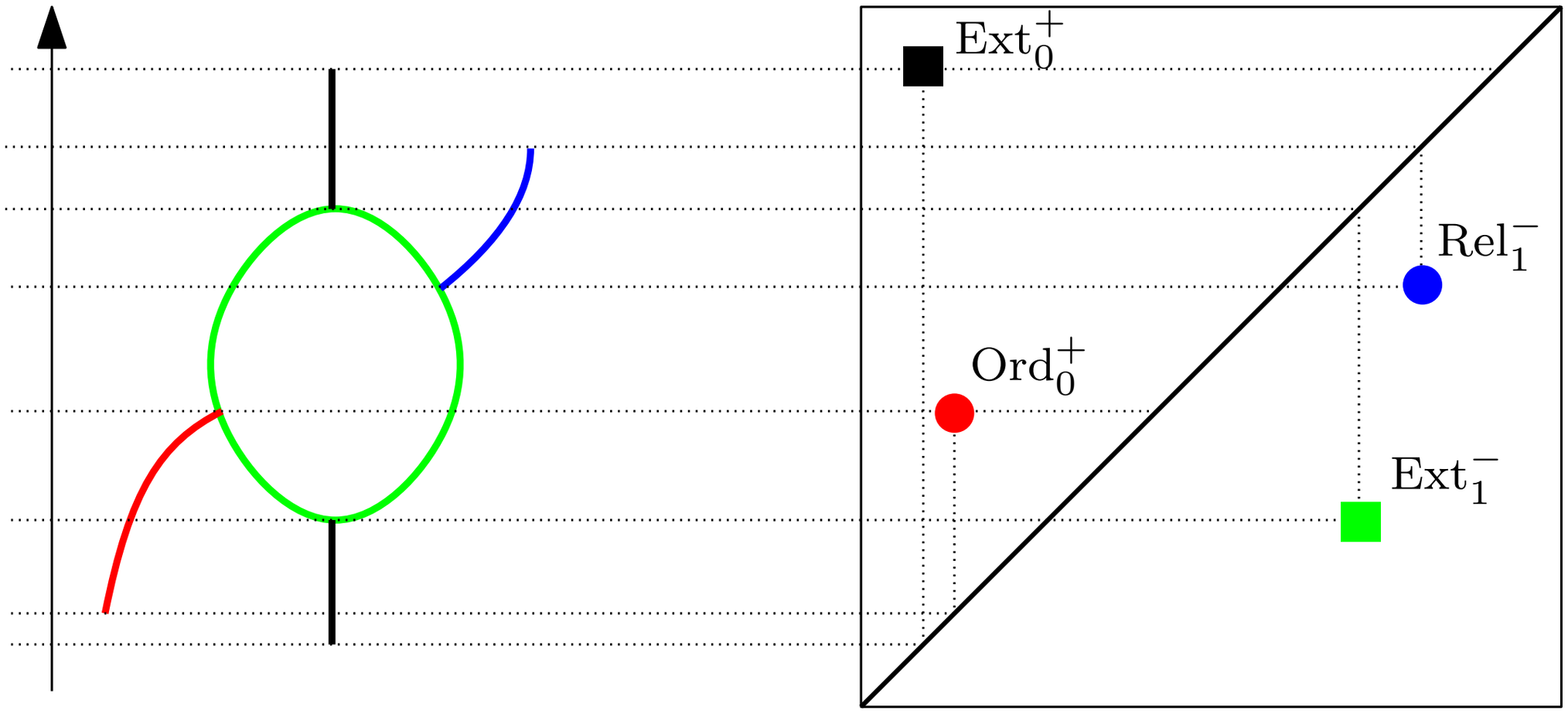
$\mathrm{Ord}_0 \tilde{f} \longleftrightarrow$ downward branches	$\mathrm{Ext}_0 \tilde{f} \longleftrightarrow$ trunks (cc)
$\mathrm{Rel}_1 \tilde{f} \longleftrightarrow$ upward branches	$\mathrm{Ext}_1 \tilde{f} \longleftrightarrow$ loops



Graph Descriptor

Thm: $D_{\tilde{f}}$ provides a **bag-of-features** descriptor for $\mathrm{R}_f(X)$:

$\mathrm{Ord}_0 \tilde{f} \longleftrightarrow$ downward branches	$\mathrm{Ext}_0 \tilde{f} \longleftrightarrow$ trunks (cc)
$\mathrm{Rel}_1 \tilde{f} \longleftrightarrow$ upward branches	$\mathrm{Ext}_1 \tilde{f} \longleftrightarrow$ loops



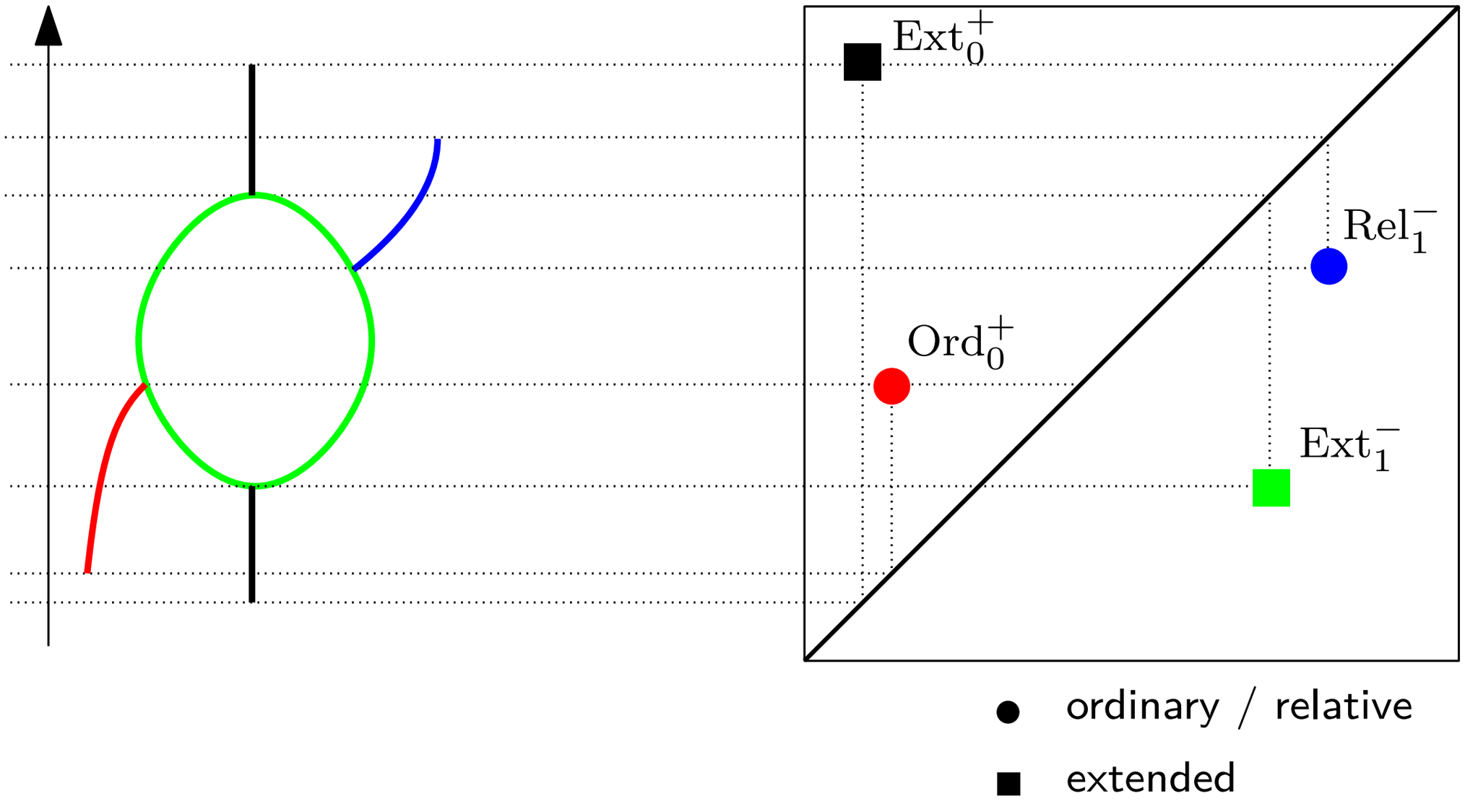
... and distance to diagonal measures the (in-)stability of each feature w.r.t. perturbations of (X, f)

● ordinary / relative

■ extended

Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.

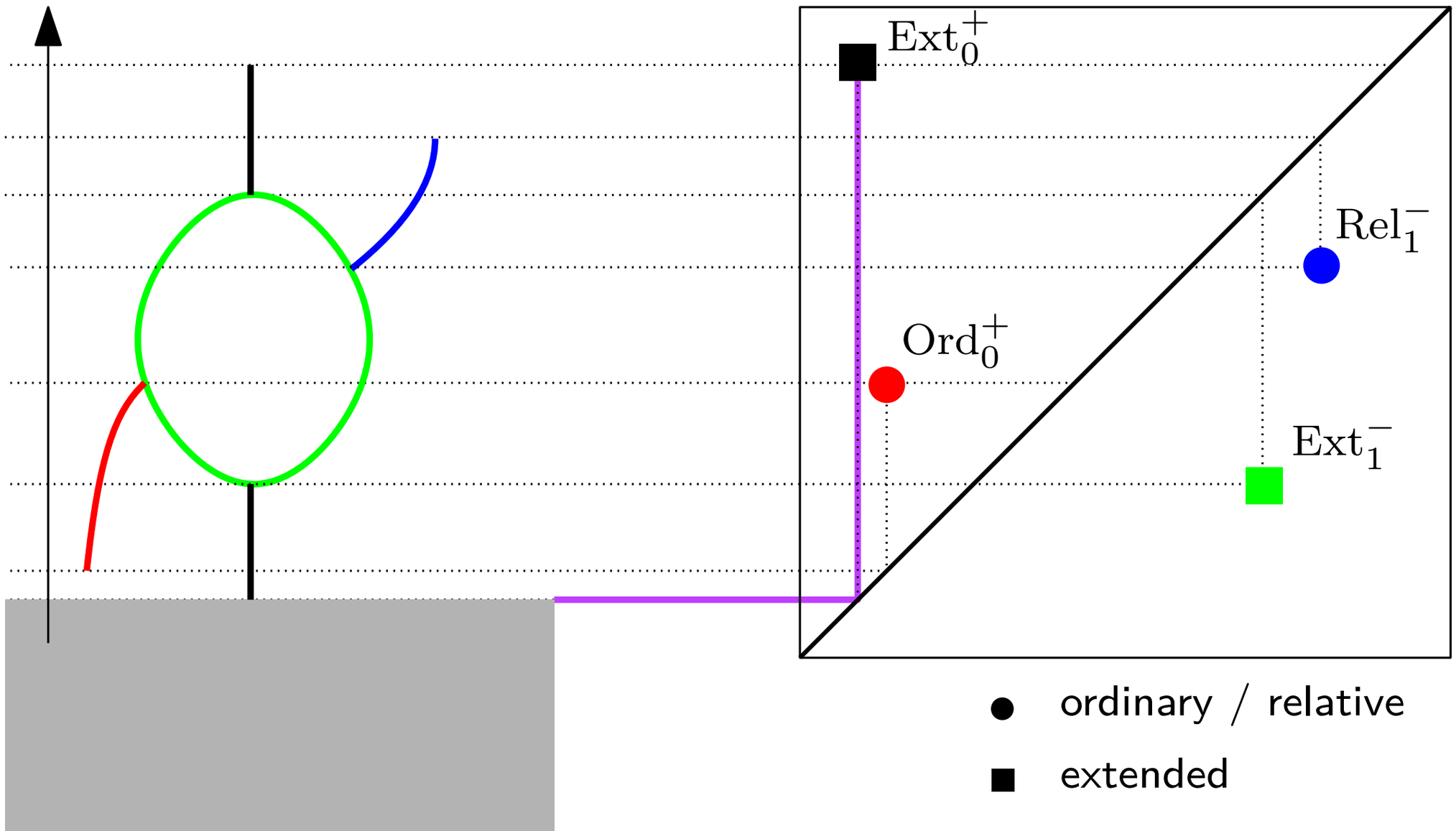


Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.

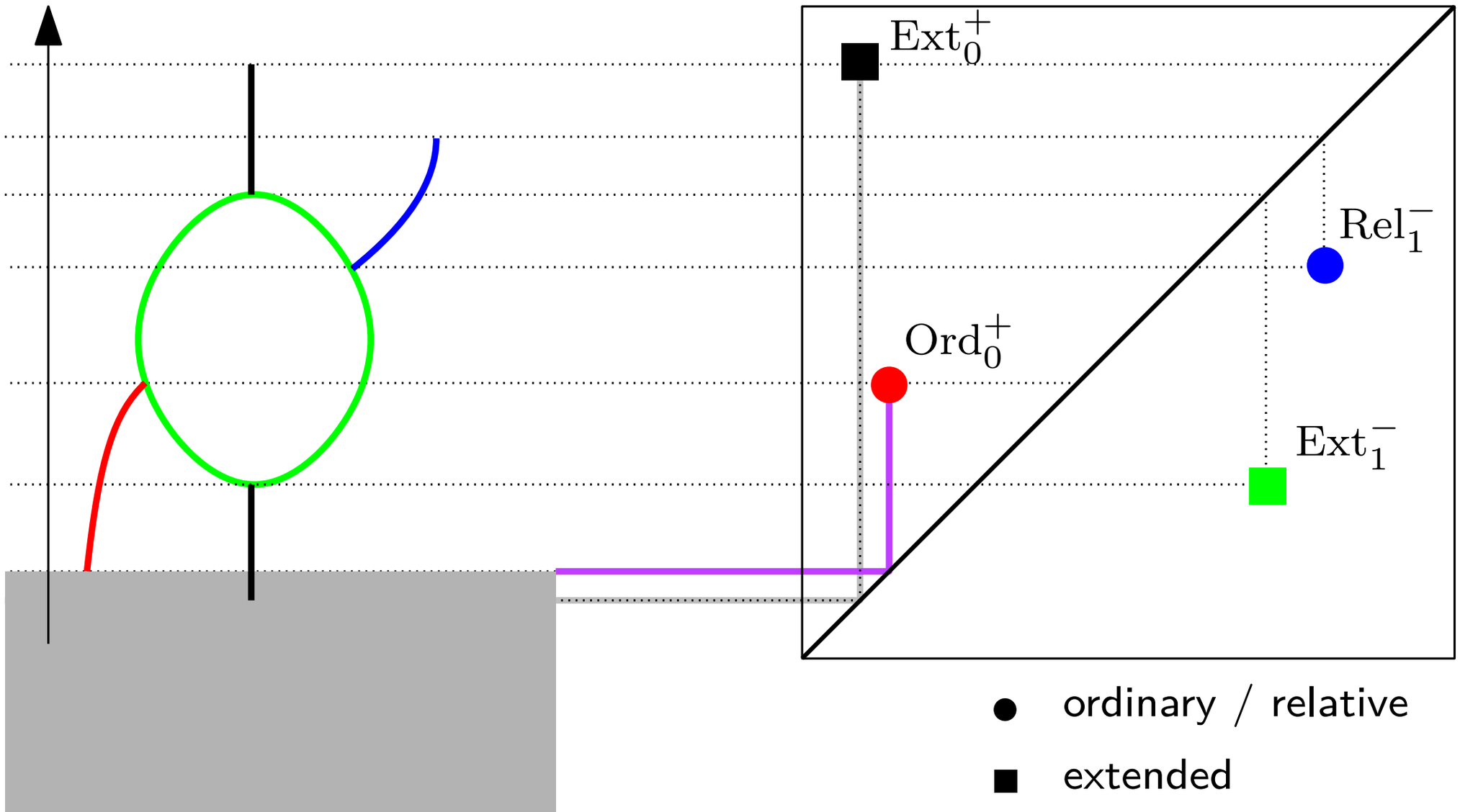
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



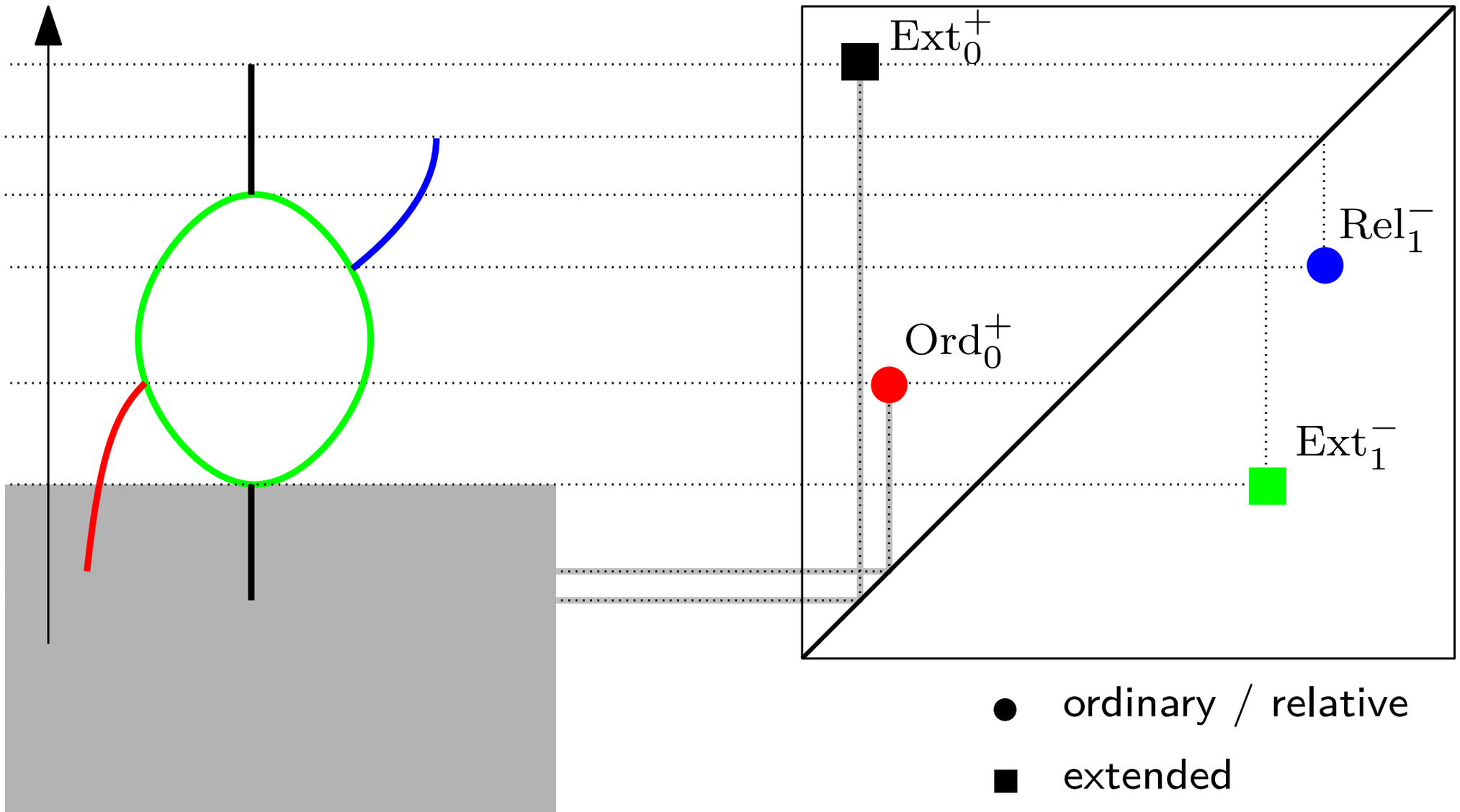
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



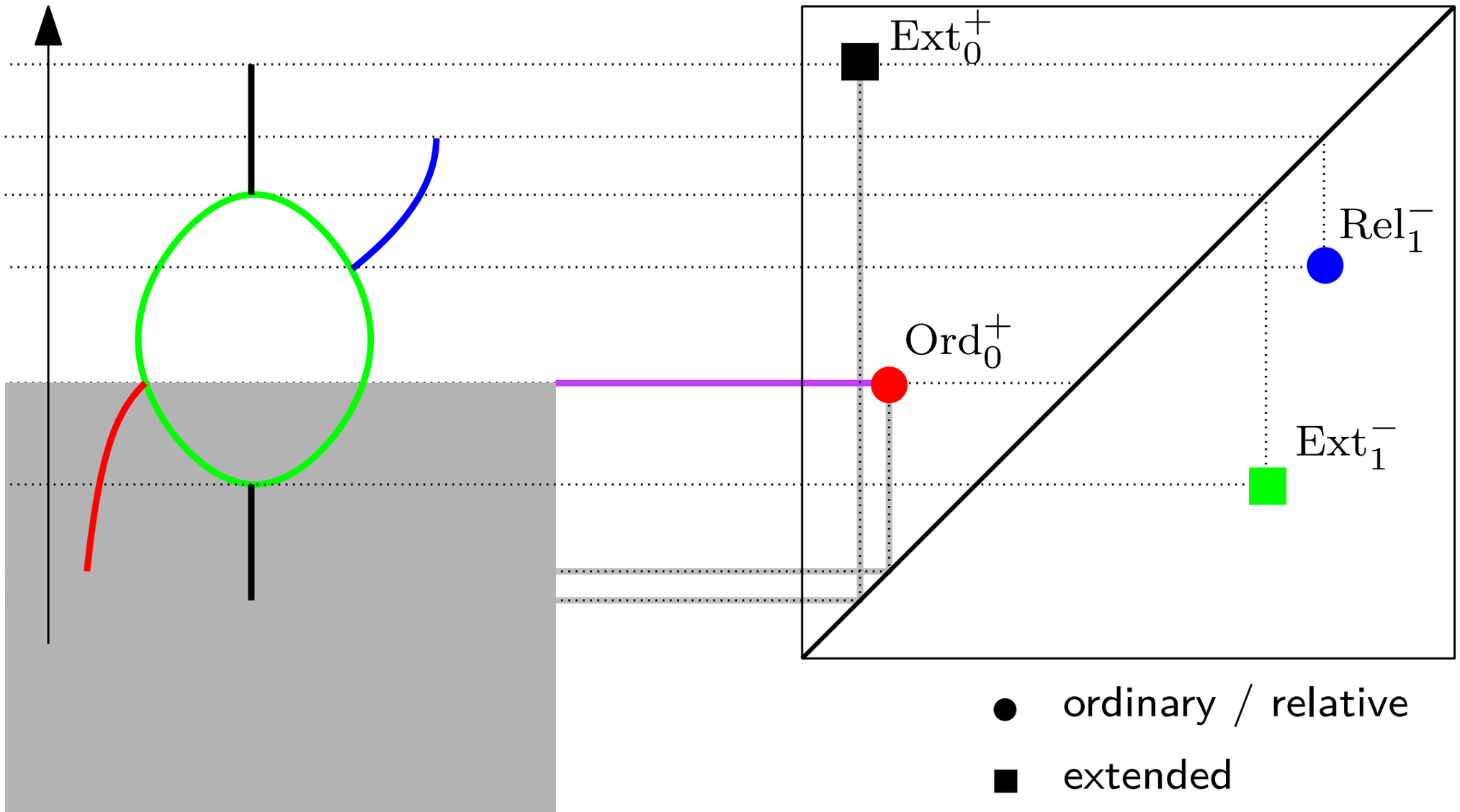
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



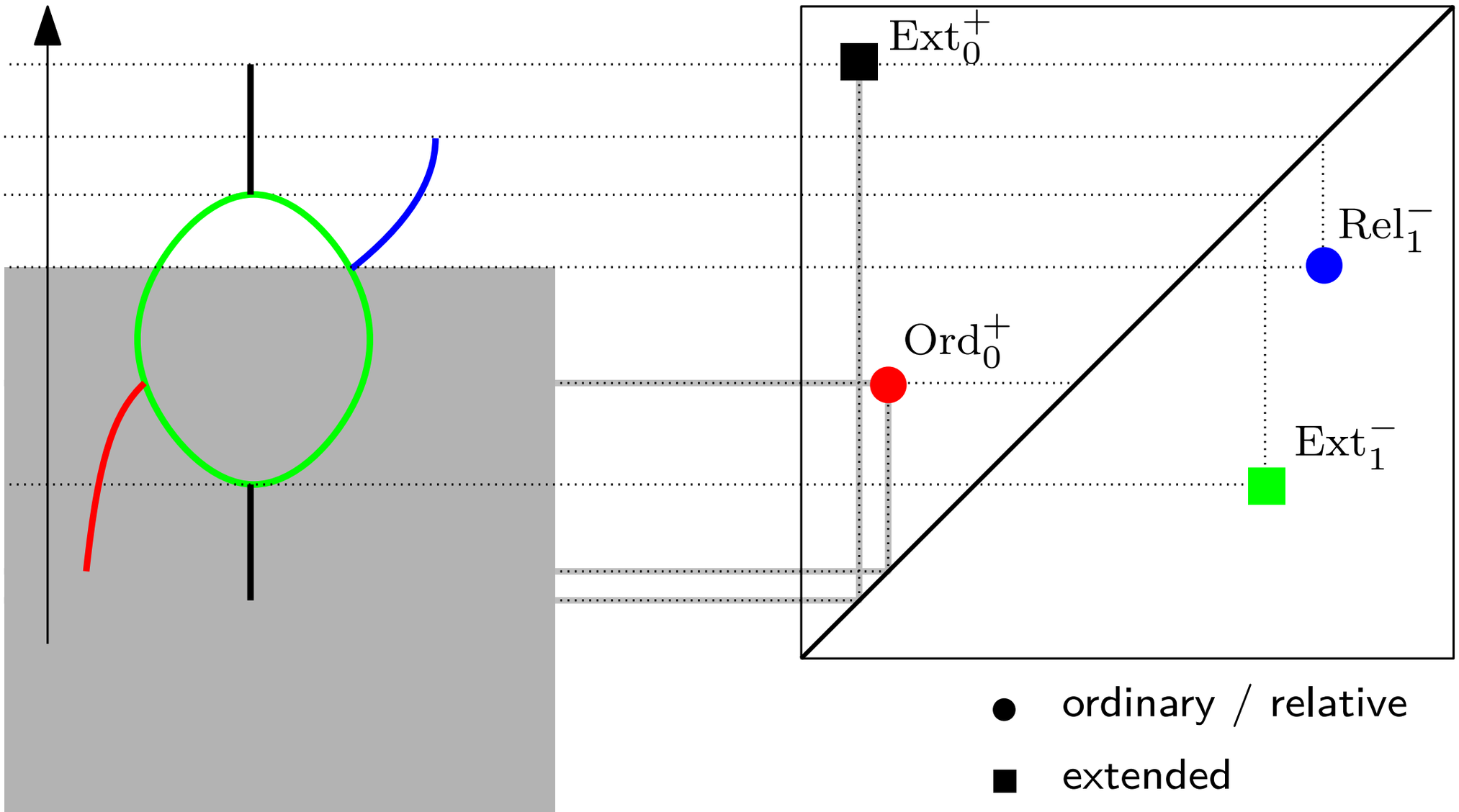
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



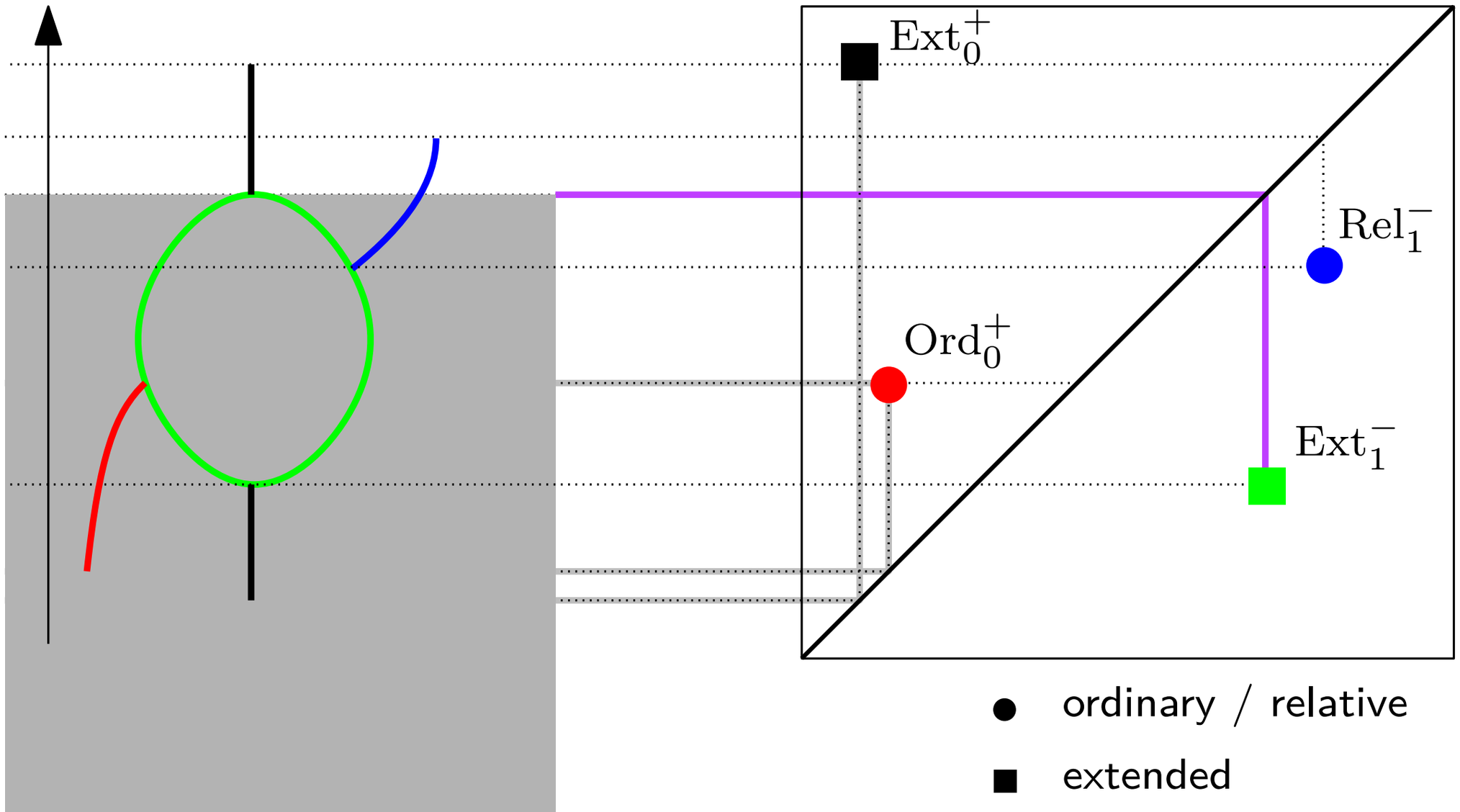
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



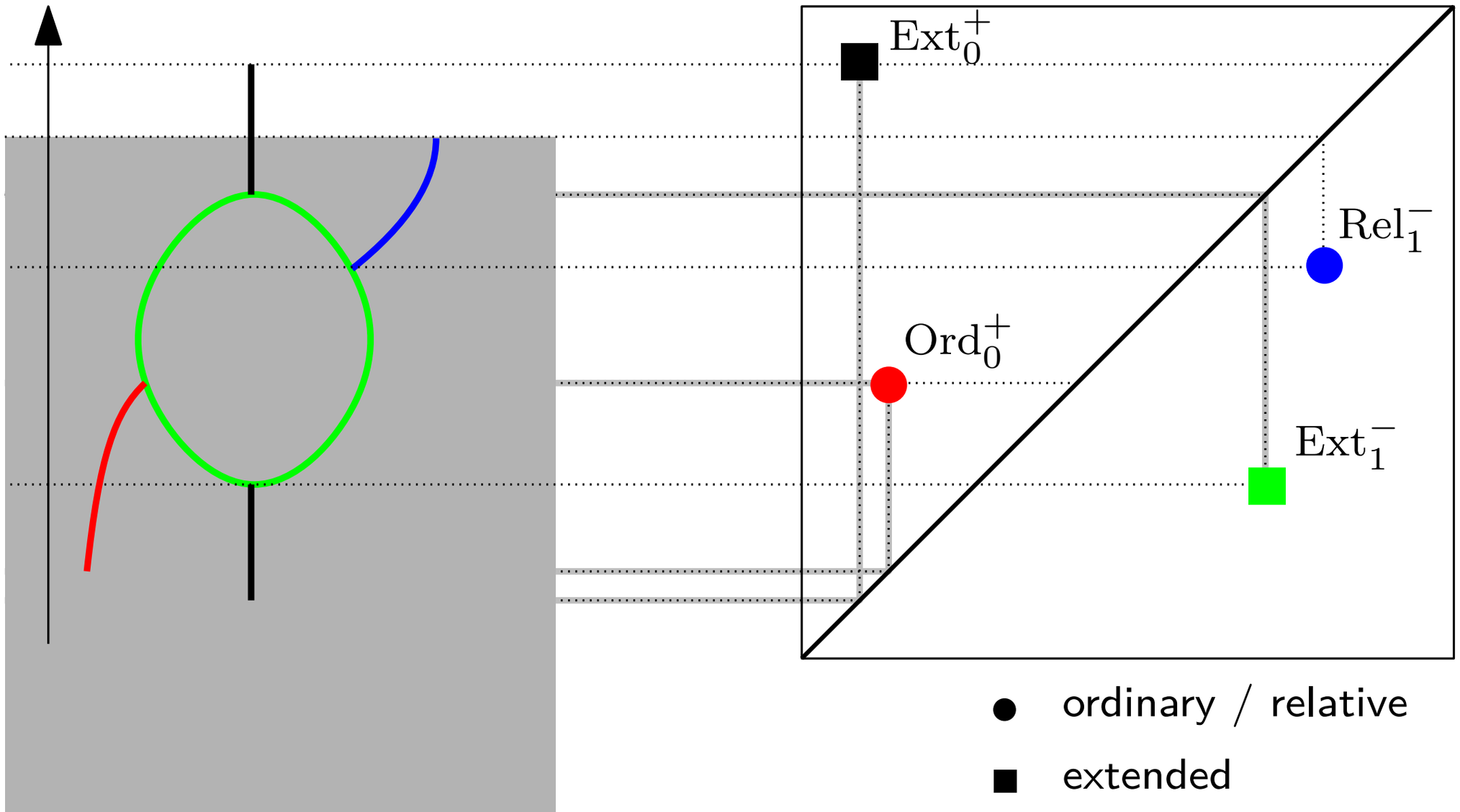
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



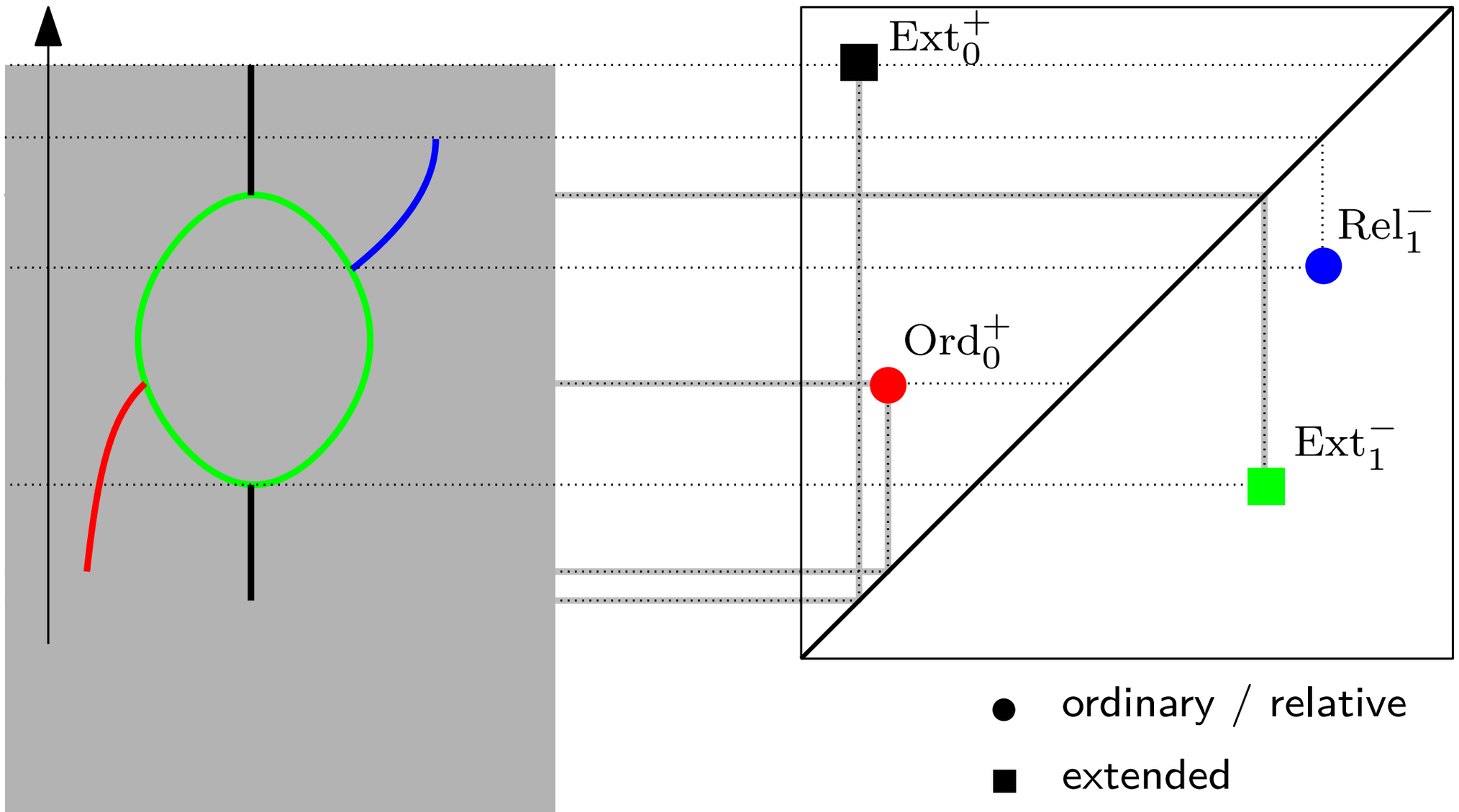
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



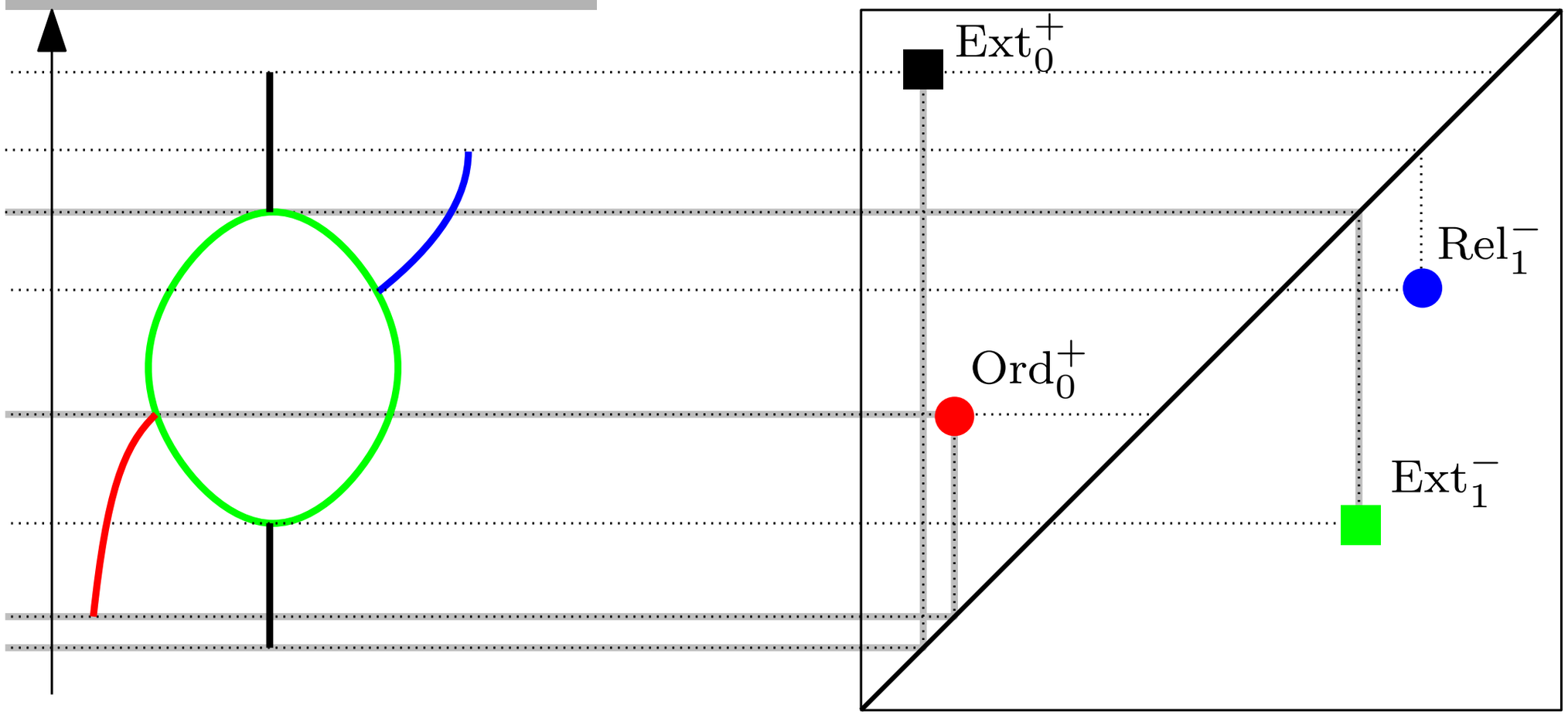
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



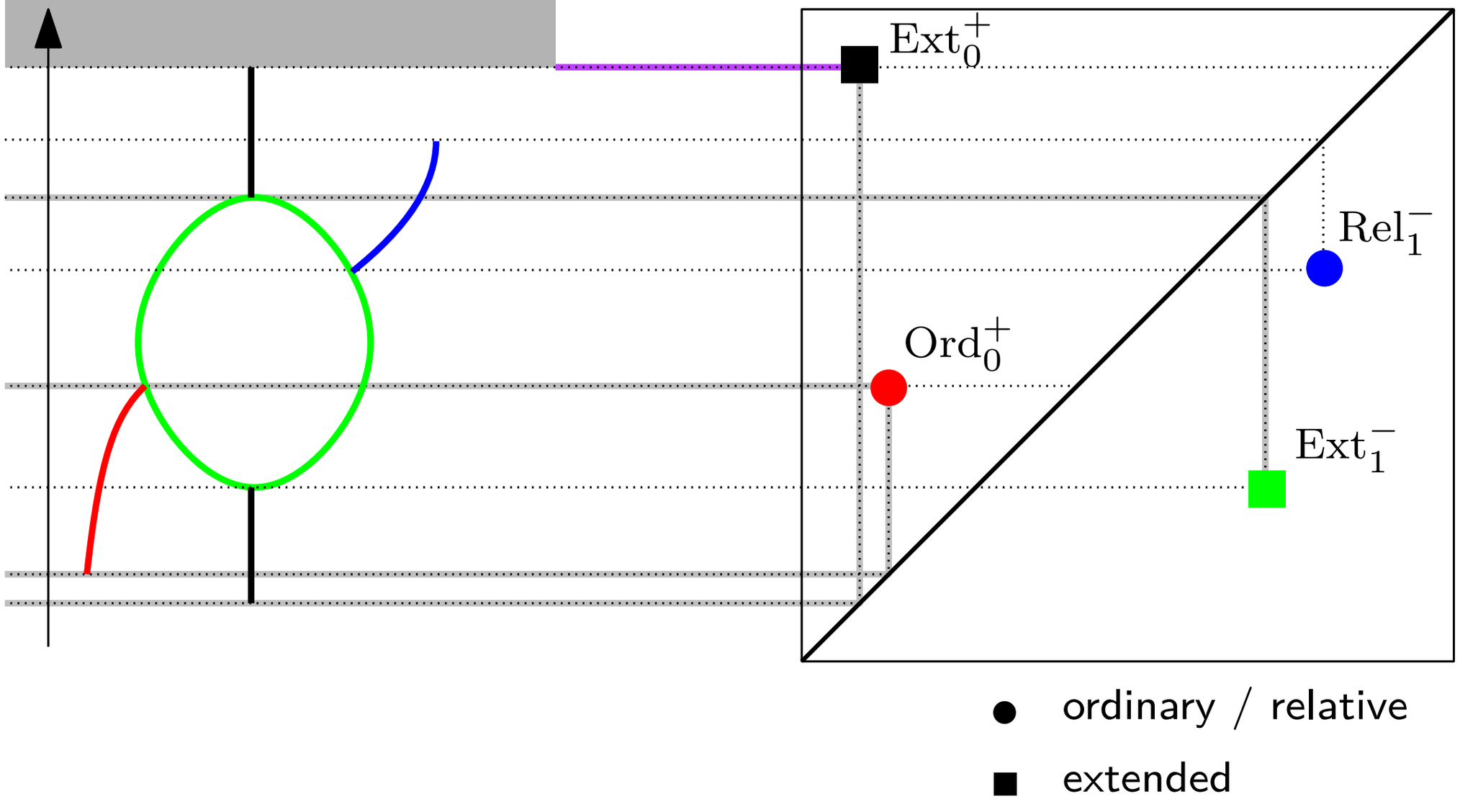
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



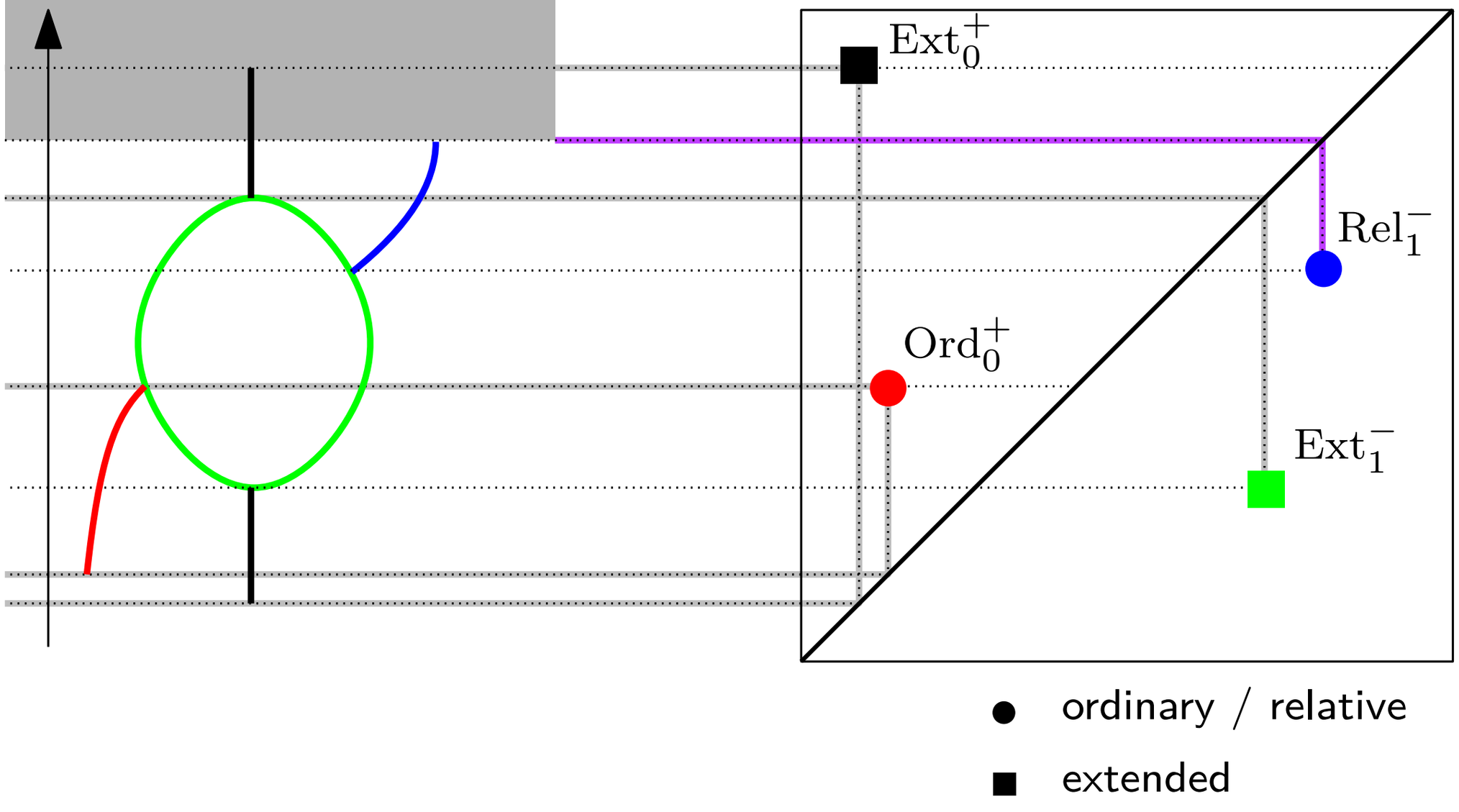
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



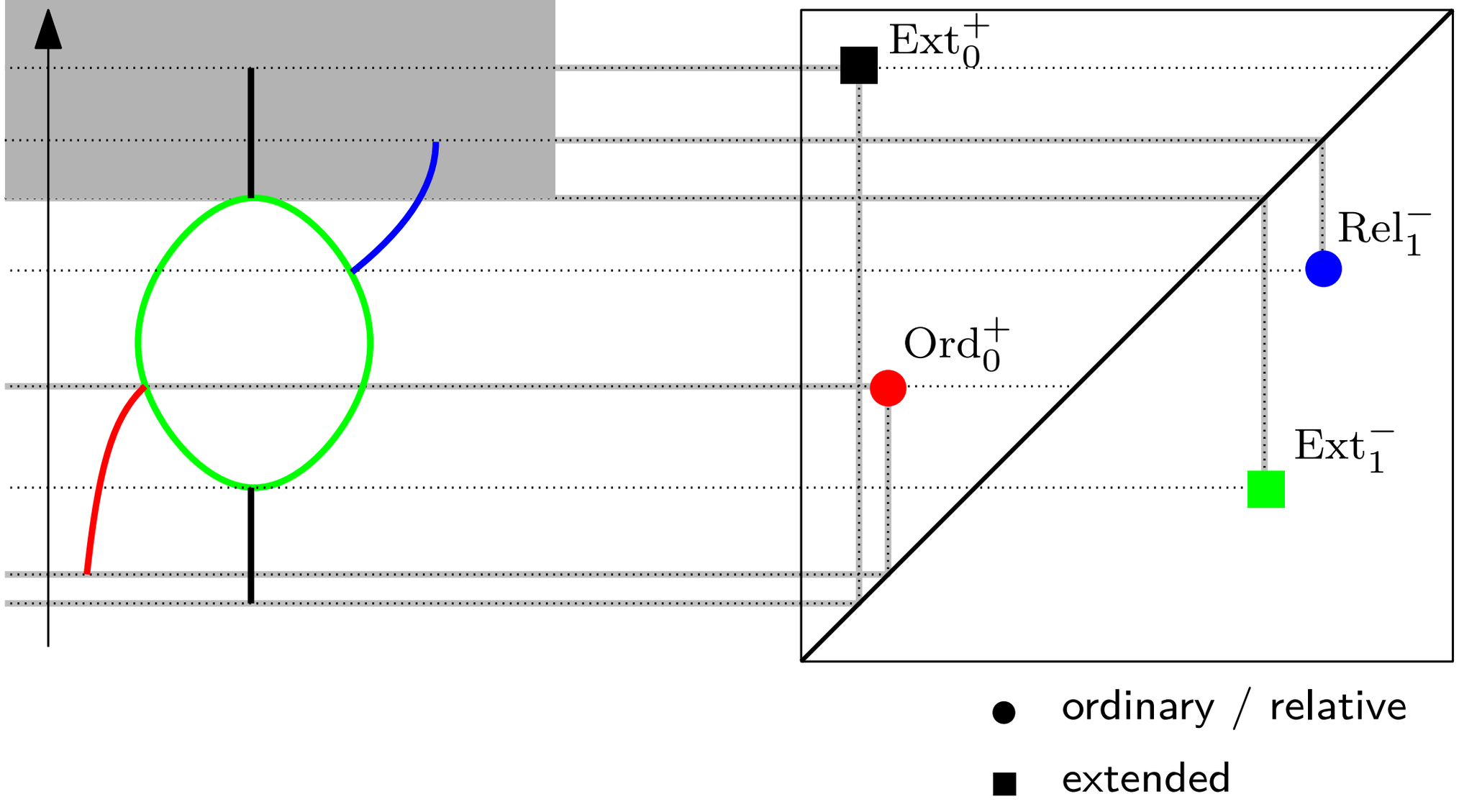
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



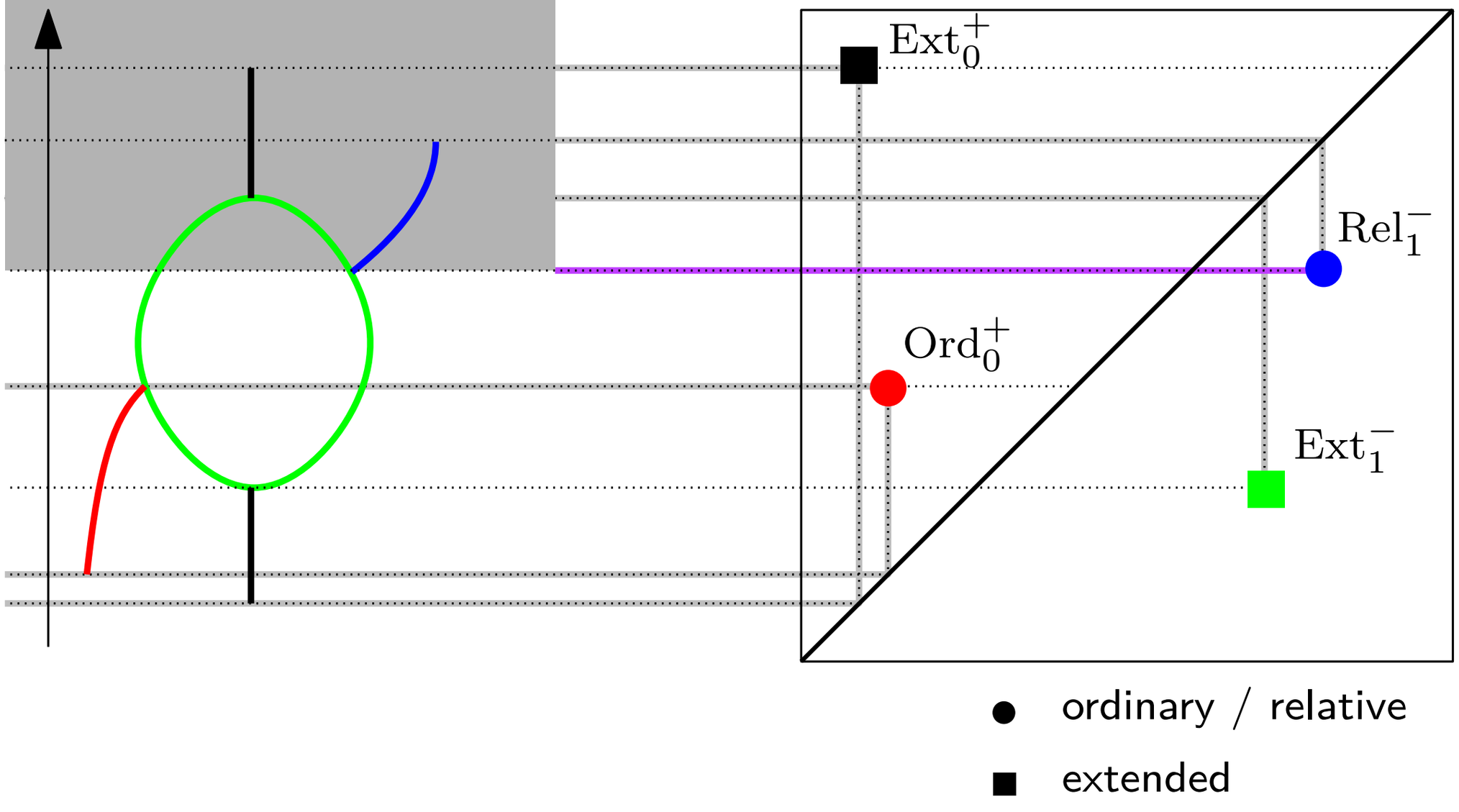
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



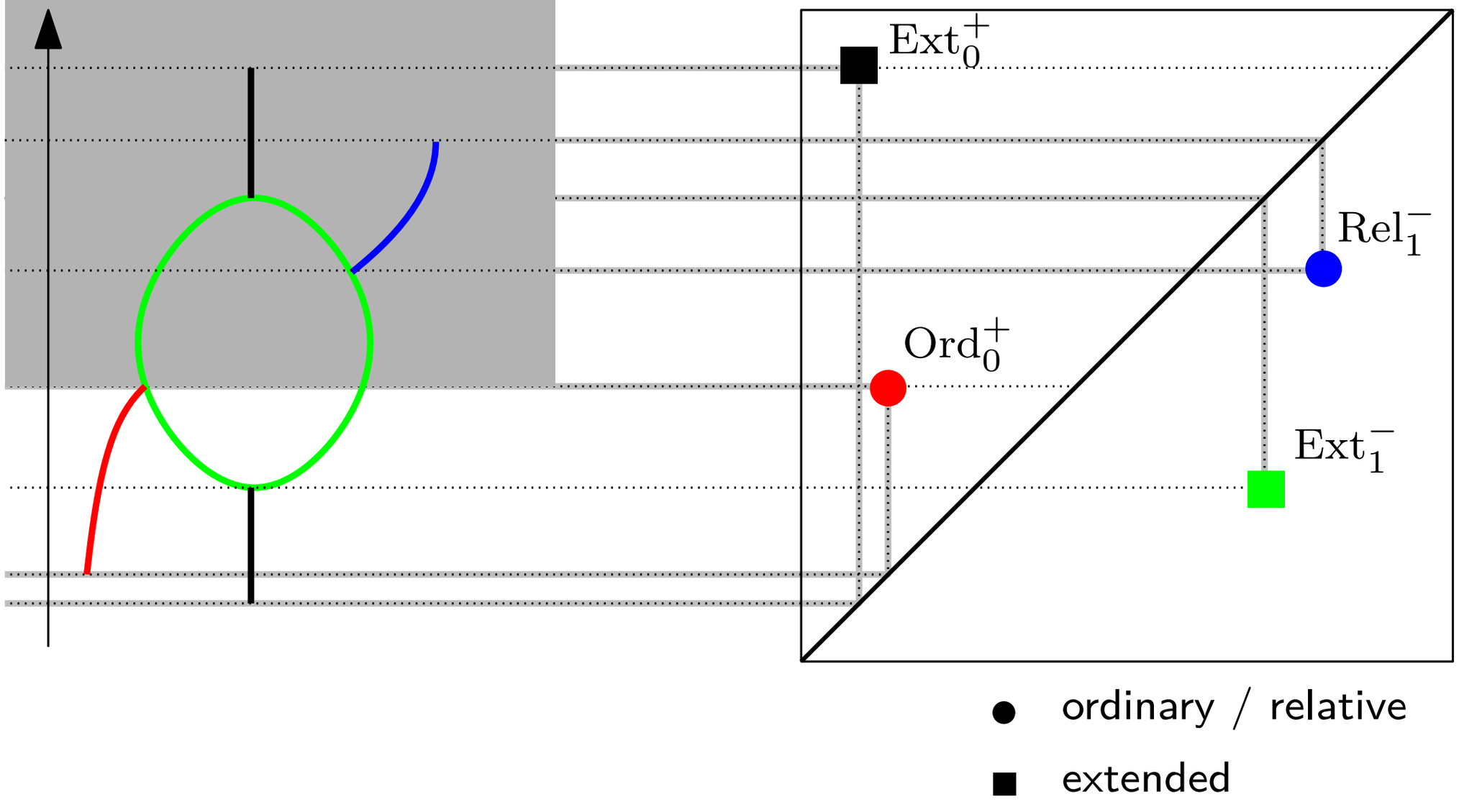
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



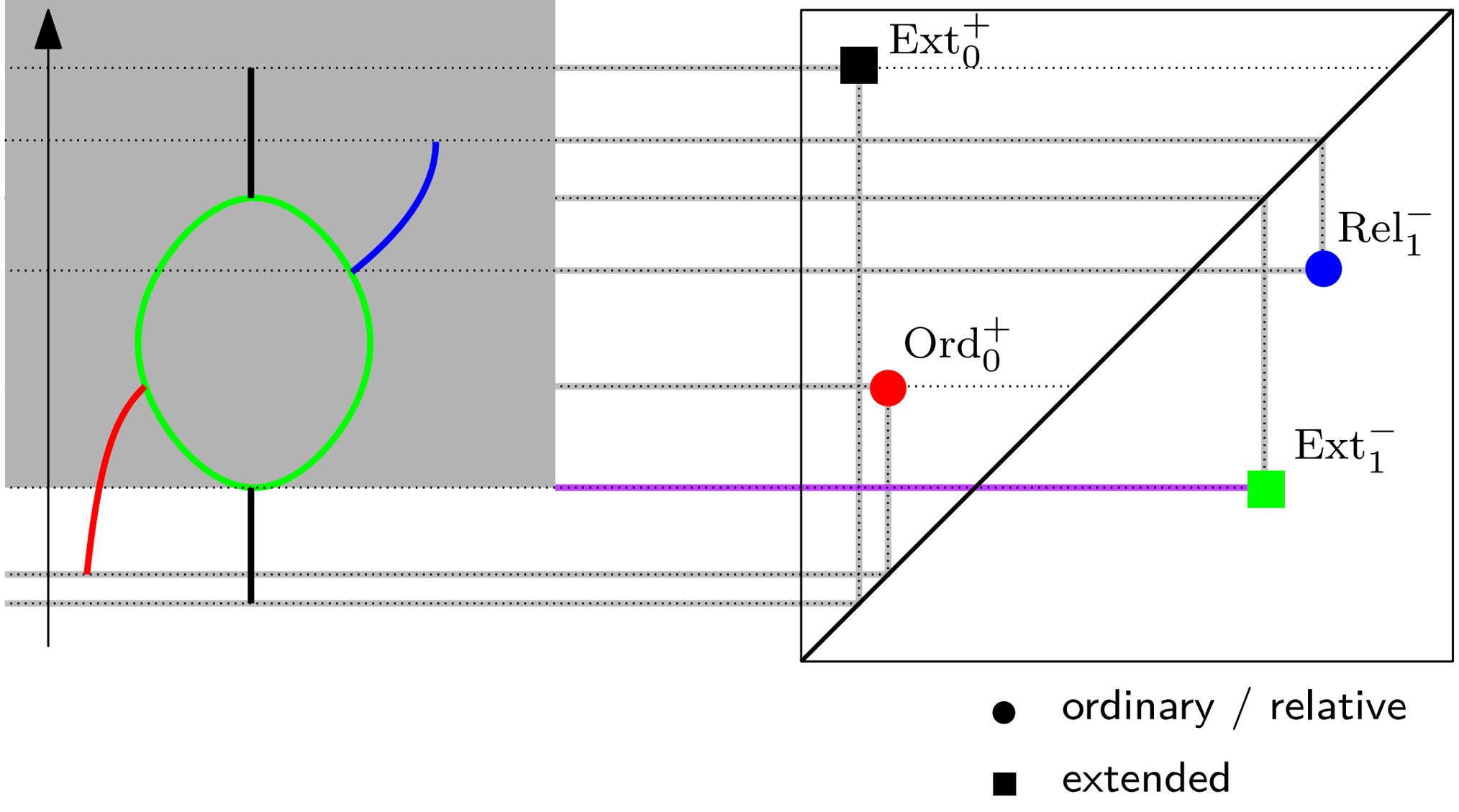
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



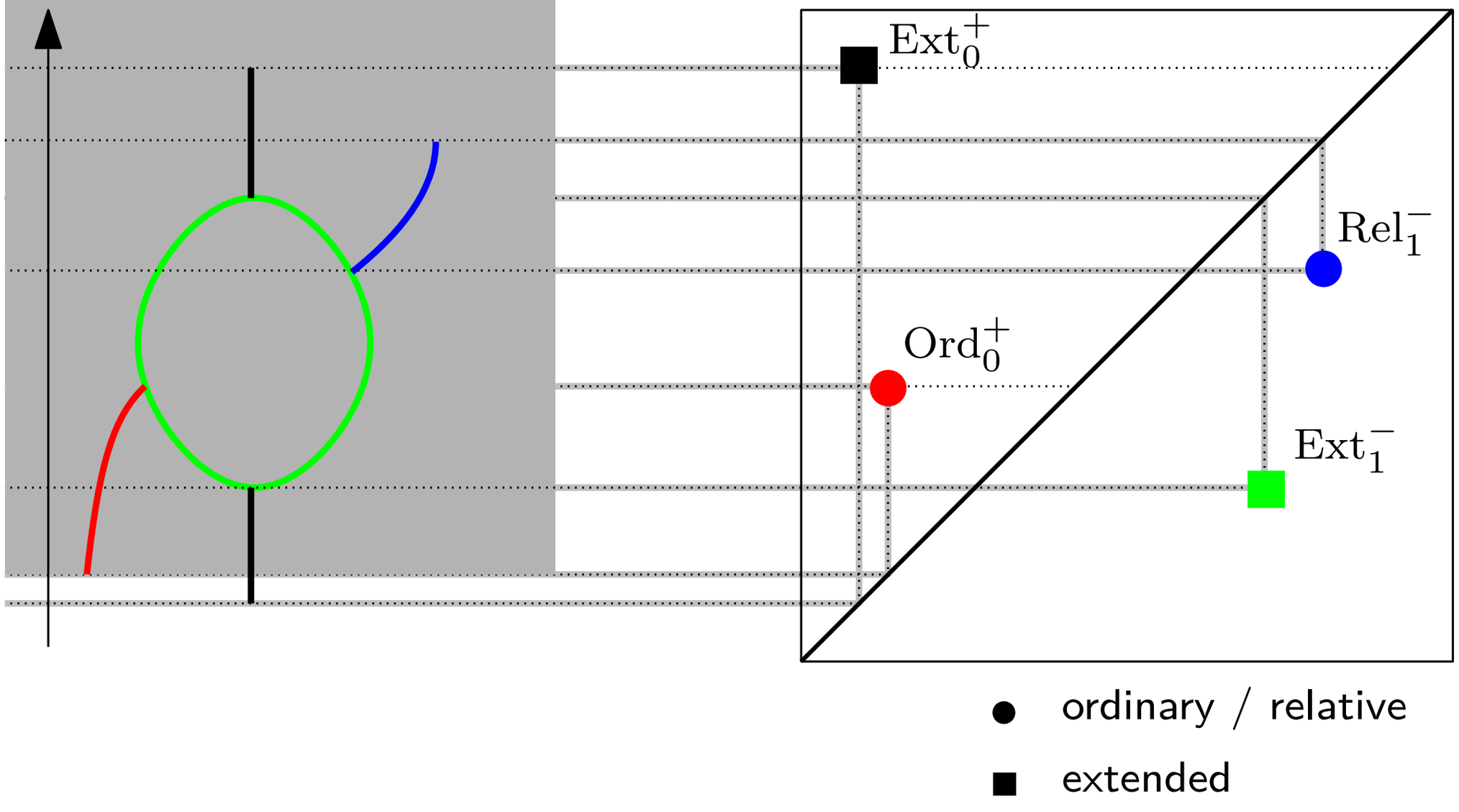
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



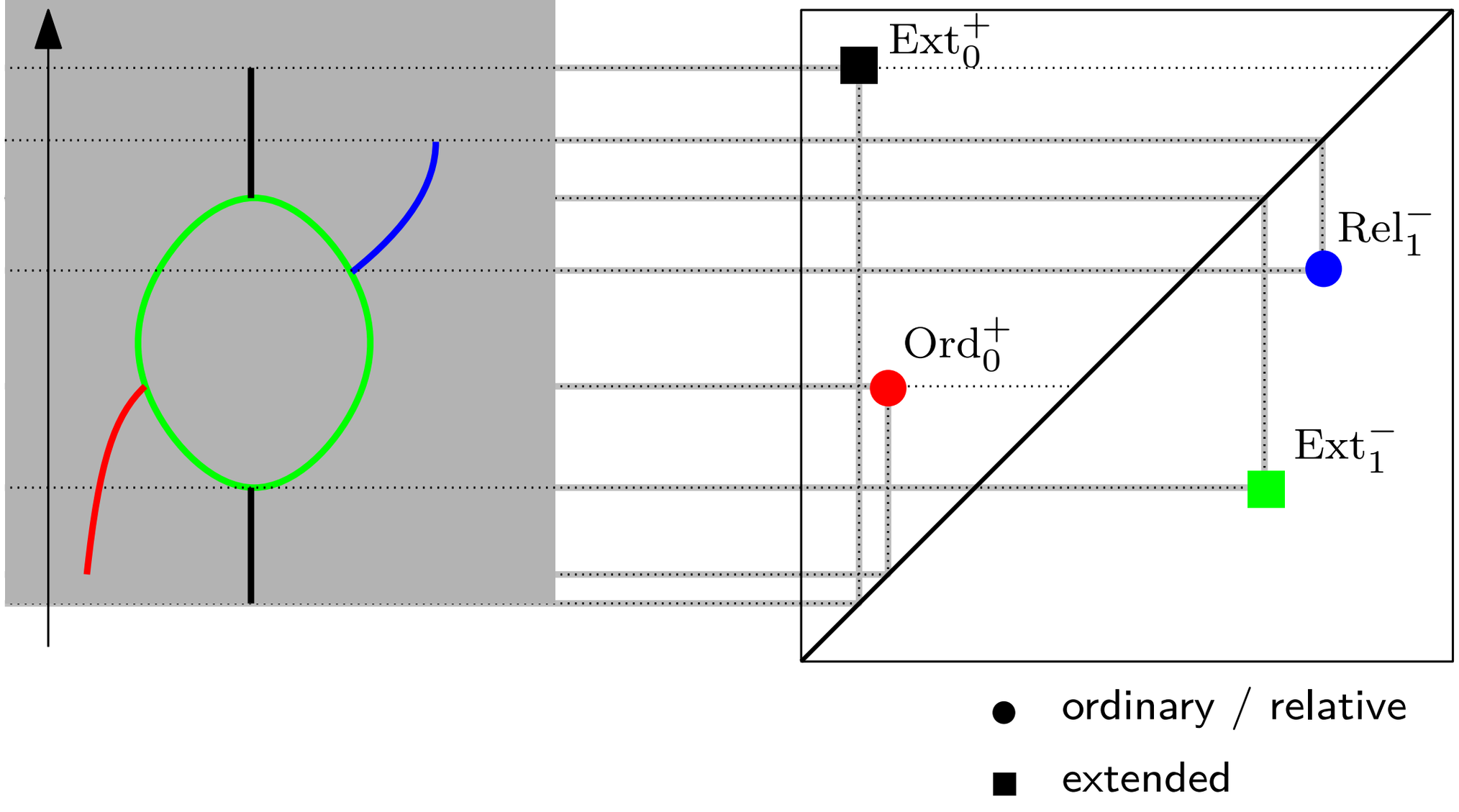
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



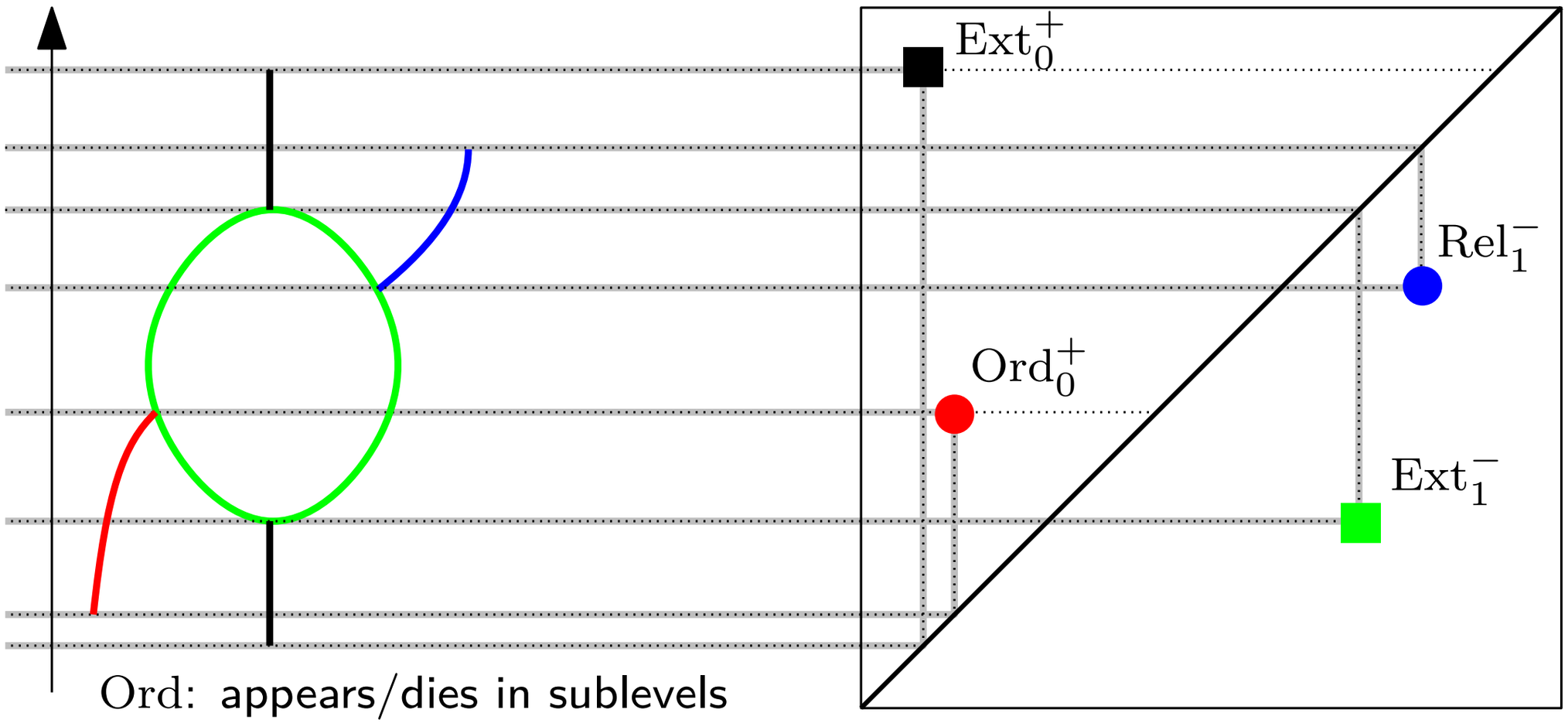
Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



Graph Descriptor

The construction of $D_{\tilde{f}}$ is based on **extended persistence**, which uses a family of *excursion sets* (sublevel then superlevel sets) of the Reeb graph.



Ord: appears/dies in sublevels

Rel: appears/dies in superlevels

Ext: appears in sublevels, dies in superlevels

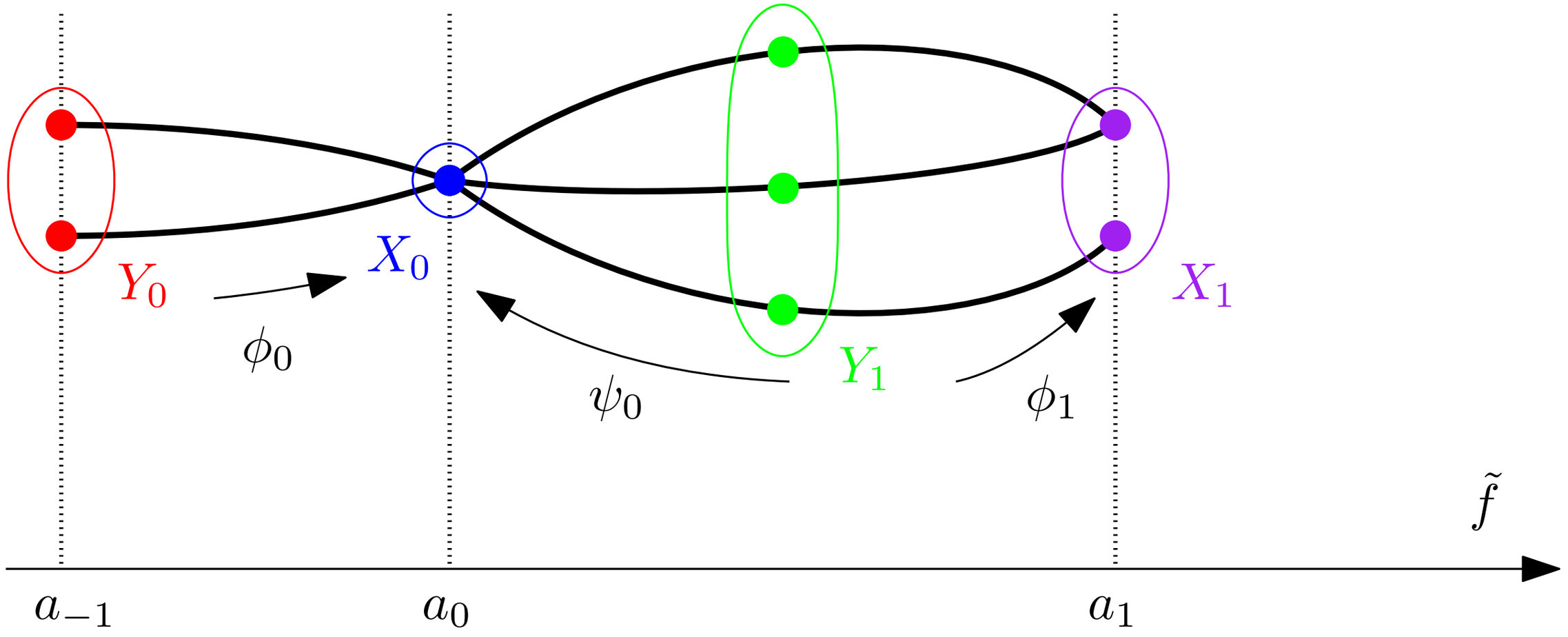
● ordinary / relative

■ extended

Extension to Mapper

Reeb graph is a *telescope* (stratified space):

$$Y_0 \times [a_{-1}, a_0] \cup_{\psi_{-1}} X_0 \times \{a_0\} \cup_{\phi_0} Y_1 \times [a_0, a_1] \cup_{\psi_0} X_1 \times \{a_1\} \cup_{\phi_1} ...$$



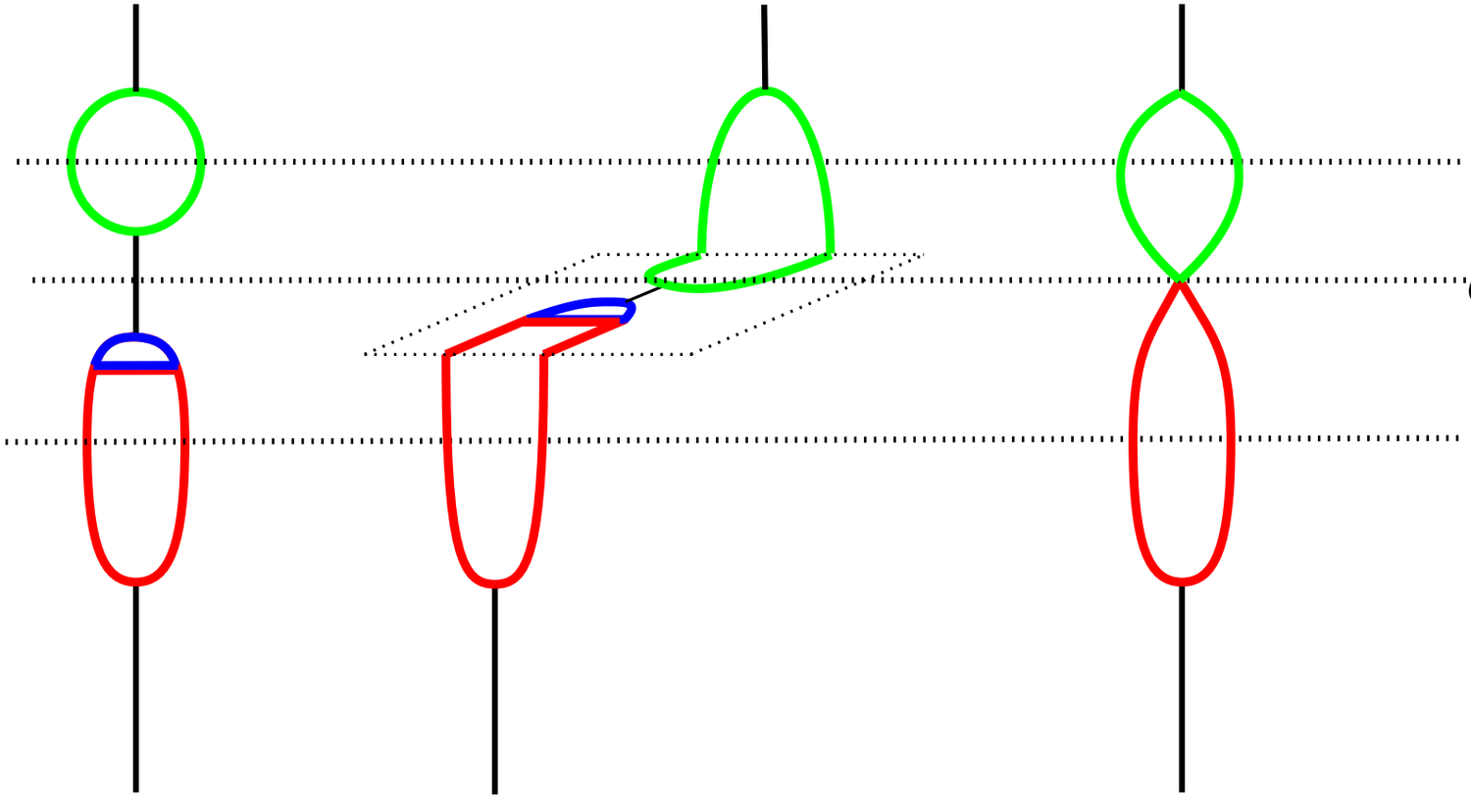
Idea: deform the Reeb graph so that it becomes the Mapper and track the corresponding changes in the persistence diagram $D_{\tilde{f}}$.

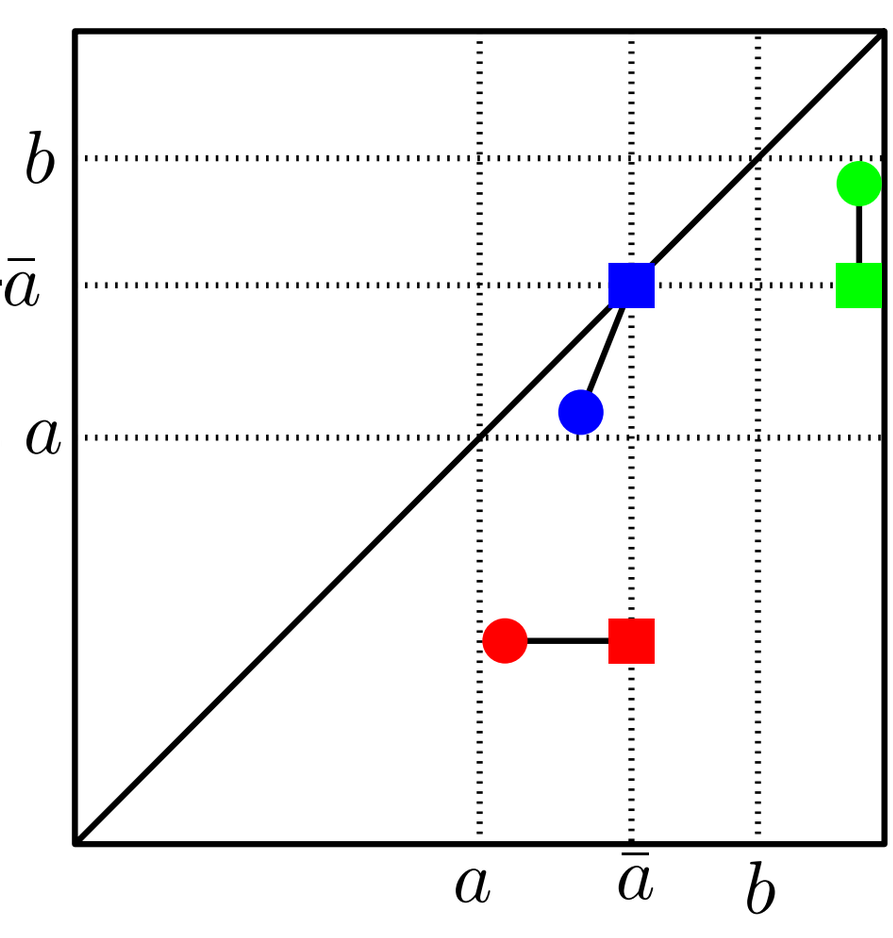
Operation 1: Merge $M_{a,b}$

$$(Y_{i-1} \times [a_{i-1}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \cup_{\phi_i} \ldots \cup_{\psi_{j-1}} (X_j \times \{a_j\}) \cup_{\phi_j} (Y_j \times [a_j, a_{j+1}])$$

$$\downarrow$$

$$(Y_{i-1} \times [a_{i-1}, \bar{a}]) \cup_{f_{i-1}} (\tilde{f}^{-1}([a, b]) \times \{\bar{a}\}) \cup_{g_j} (Y_j \times [\bar{a}, a_{j+1}])$$



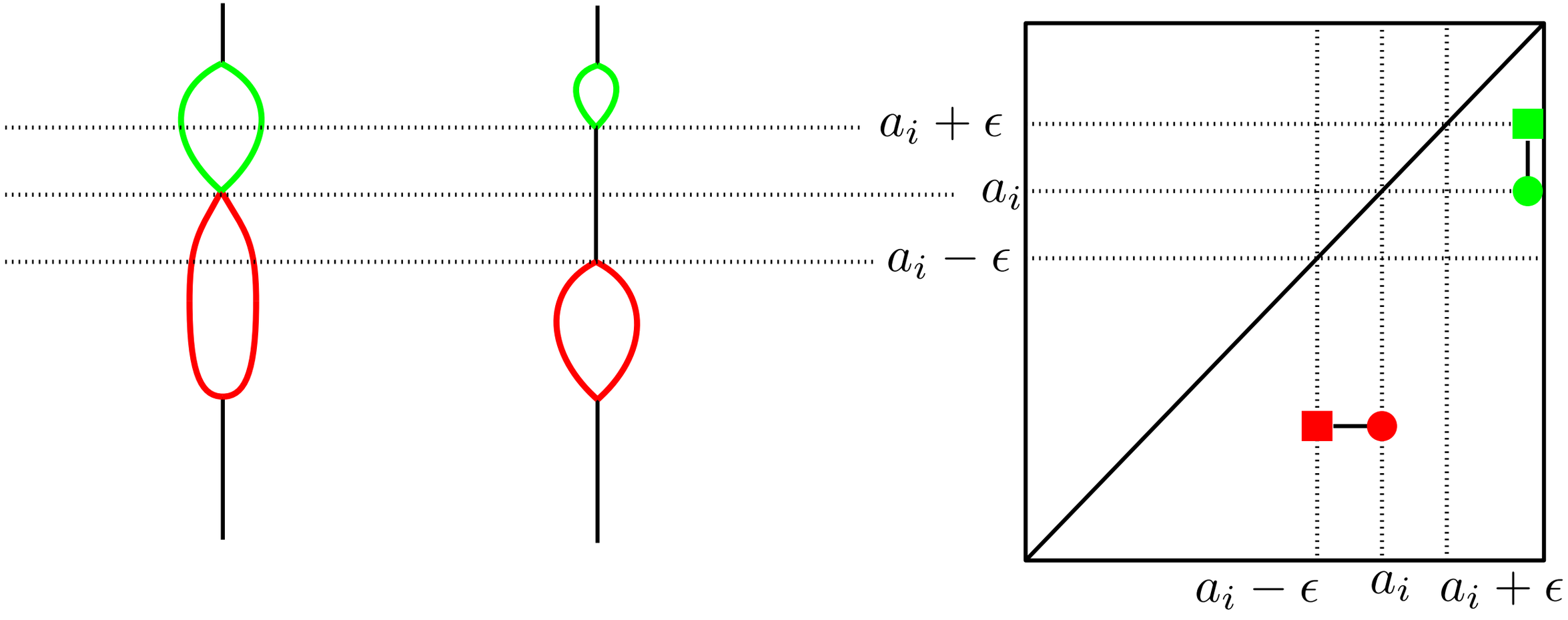


Operation 2: Split $Sp_{a_i,\epsilon}$

$$(Y_{i-1} \times [a_{i-1}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \cup_{\phi_i} (Y_i \times [a_i, a_{i+1}])$$

$$\downarrow$$

$$(Y_{i-1} \times [a_{i-1}, a_i - \epsilon]) \cup_{\psi_{i-1}^{a_i-\epsilon}} (X_i \times \{a_i - \epsilon\}) \cup_{\mathrm{id}} (X_i \times [a_i - \epsilon, a_i + \epsilon]) \cup_{\mathrm{id}} (X_i \times \{a_i + \epsilon\}) \cup_{\phi_i^{a_i+\epsilon}} (Y_i \times [a_i + \epsilon, a_{i+1}])$$

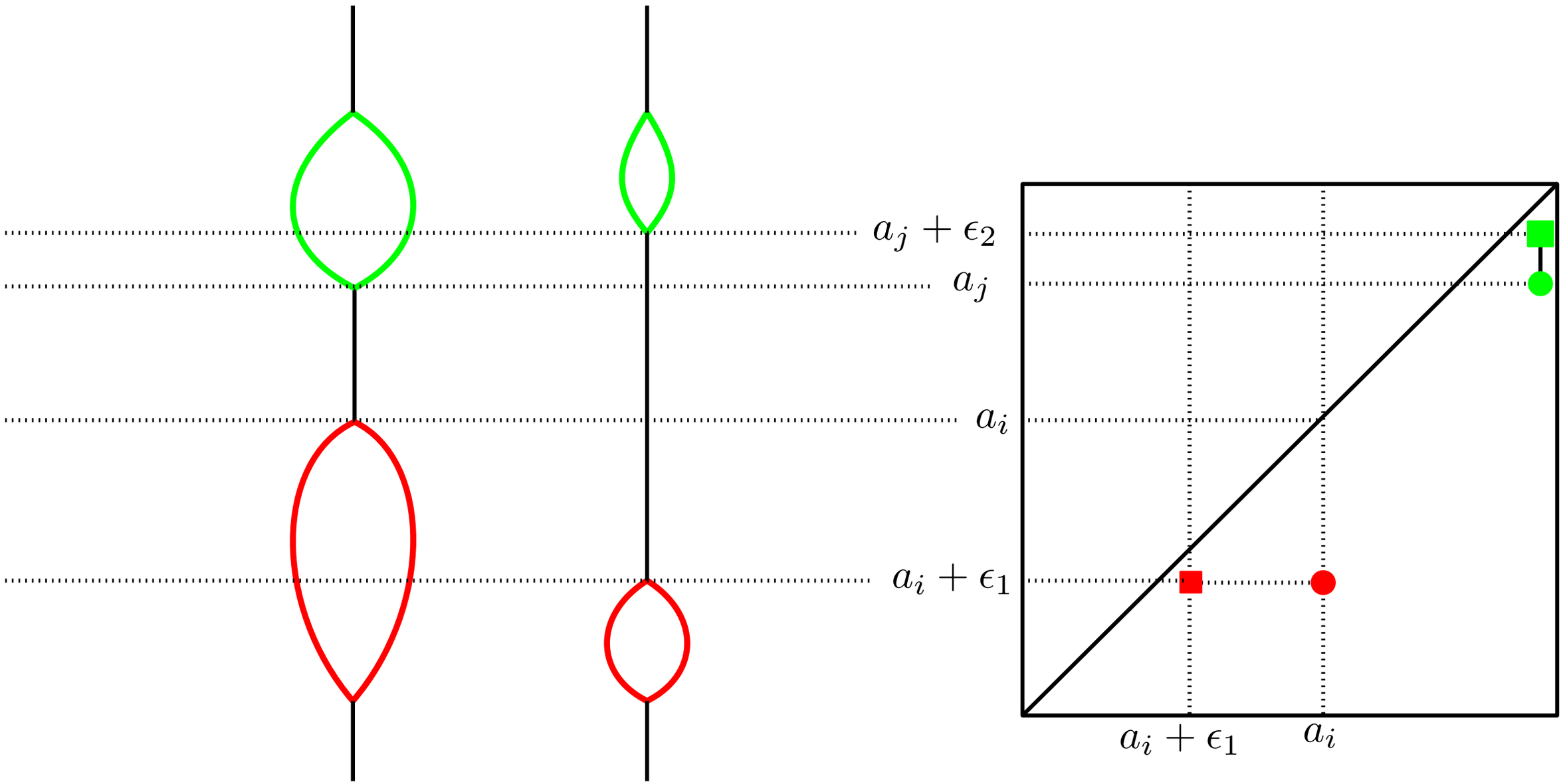


Operation 3: Shift $Sh_{a_i,\epsilon}$

$$(Y_{i-1} \times [a_{i-1}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \cup_{\phi_i} (Y_i \times [a_i, a_{i+1}])$$

$$\downarrow$$

$$(Y_{i-1} \times [a_{i-1}, a_i + \epsilon]) \cup_{\psi_{i-1}} (X_i \times \{a_i + \epsilon\}) \cup_{\phi_i} (Y_i \times [a_i + \epsilon, a_{i+1}])$$



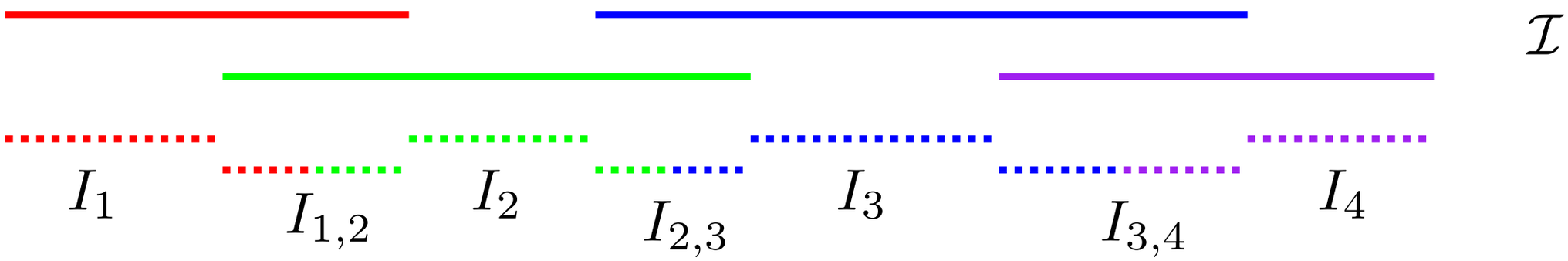
Formula Reeb graph $\longrightarrow$ Mapper

Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$

Formula Reeb graph $\rightarrow$ Mapper

Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$

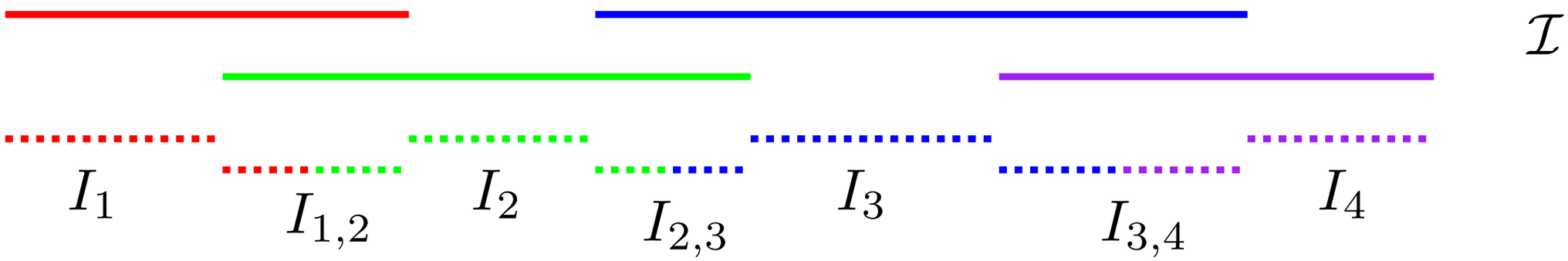
- $M_{\mathcal{I}}$ is the union of all M_{I_k} and $M_{I_{k,k+1}}$ for $I \in \mathcal{I}$



Formula Reeb graph $\rightarrow$ Mapper

Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$

- $M_{\mathcal{I}}$ is the union of all M_{I_k} and $M_{I_{k,k+1}}$ for $I \in \mathcal{I}$

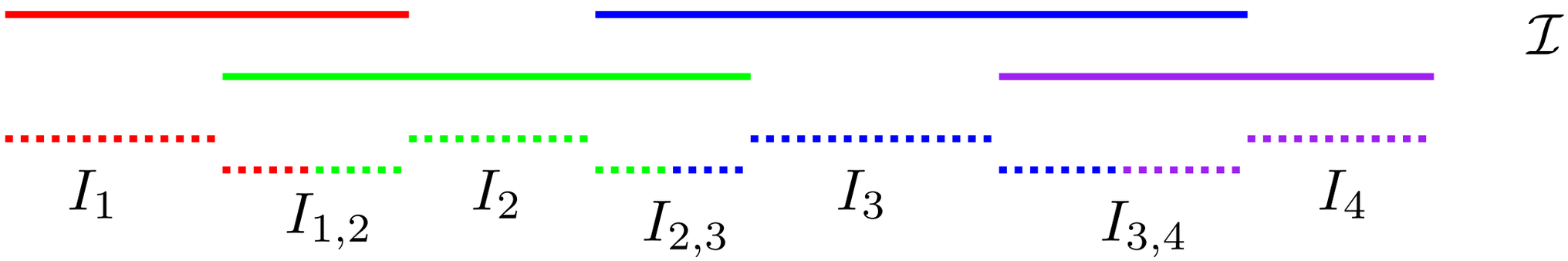


- $Sp_{\mathcal{I}}$ is the union of all $Sp_{\epsilon,\bar{a}}$ with ϵ small

Formula Reeb graph $\rightarrow$ Mapper

Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$

- $M_{\mathcal{I}}$ is the union of all M_{I_k} and $M_{I_{k,k+1}}$ for $I \in \mathcal{I}$

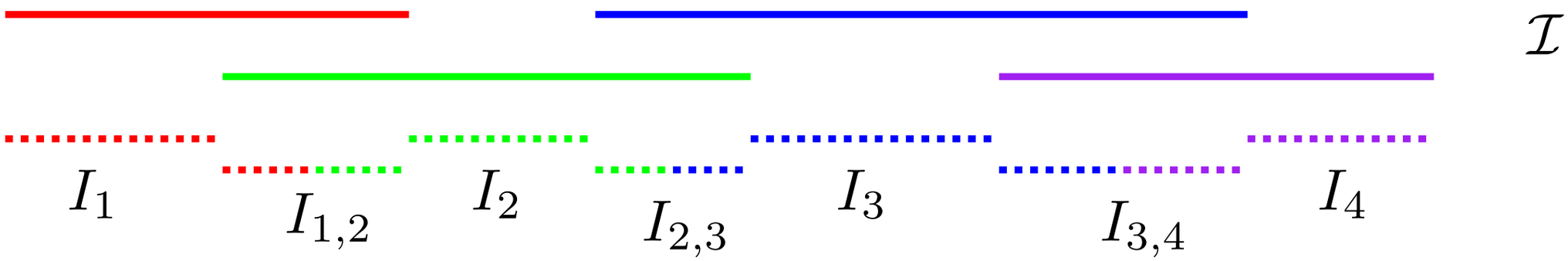


- $Sp_{\mathcal{I}}$ is the union of all $Sp_{\epsilon,\bar{a}}$ with ϵ small
- $Sh_{\mathcal{I}}$ is the union of all $Sh_{\epsilon_1,\bar{a}+\epsilon}$ and $Sh_{\epsilon_2,\bar{a}-\epsilon}$ with ϵ_1, ϵ_2 small

Formula Reeb graph $\rightarrow$ Mapper

Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$

- $M_{\mathcal{I}}$ is the union of all M_{I_k} and $M_{I_{k,k+1}}$ for $I \in \mathcal{I}$

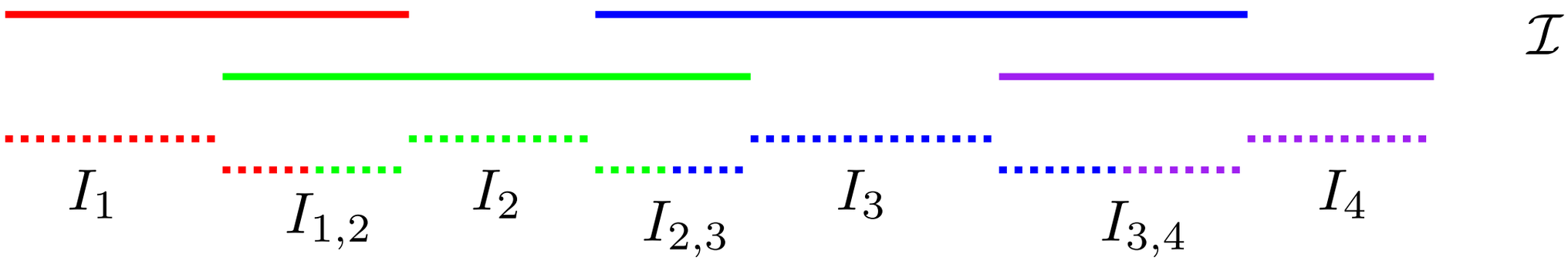


- $Sp_{\mathcal{I}}$ is the union of all $Sp_{\epsilon,\bar{a}}$ with ϵ small
- $Sh_{\mathcal{I}}$ is the union of all $Sh_{\epsilon_1,\bar{a}+\epsilon}$ and $Sh_{\epsilon_2,\bar{a}-\epsilon}$ with ϵ_1, ϵ_2 small
- $M'_{\mathcal{I}}$ is the union of all M_{I_k} for $I \in \mathcal{I}$

Formula Reeb graph $\rightarrow$ Mapper

Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$

- $M_{\mathcal{I}}$ is the union of all M_{I_k} and $M_{I_{k,k+1}}$ for $I \in \mathcal{I}$

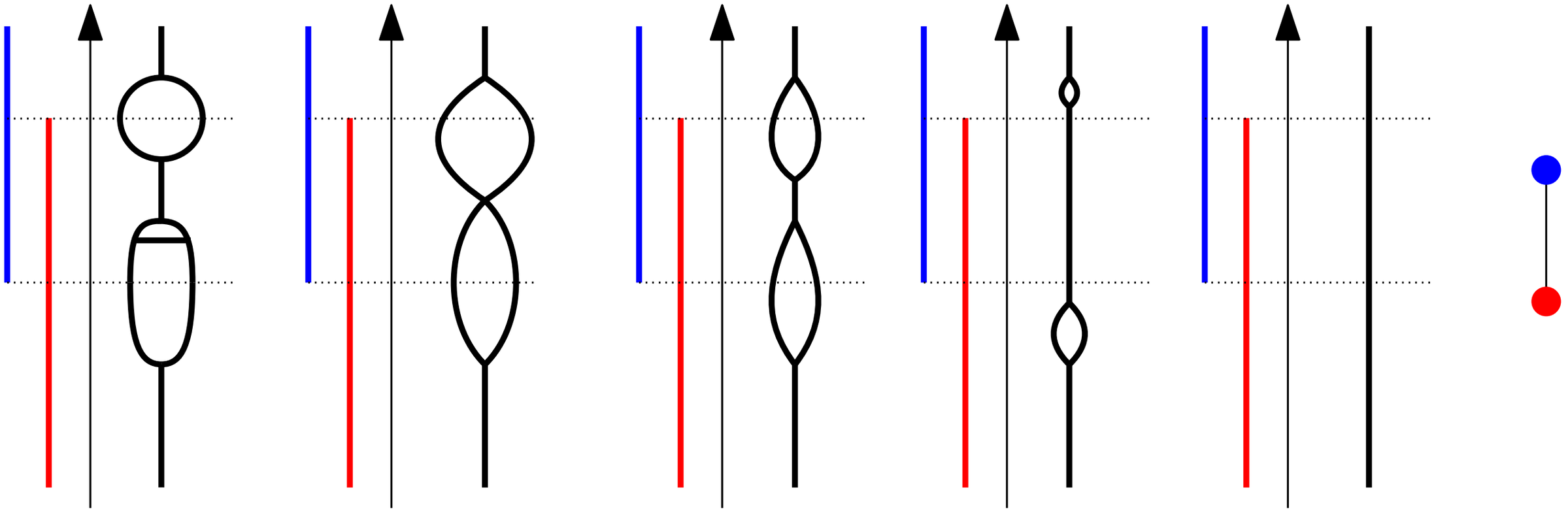


- $Sp_{\mathcal{I}}$ is the union of all $Sp_{\epsilon,\bar{a}}$ with ϵ small
- $Sh_{\mathcal{I}}$ is the union of all $Sh_{\epsilon_1,\bar{a}+\epsilon}$ and $Sh_{\epsilon_2,\bar{a}-\epsilon}$ with ϵ_1, ϵ_2 small
- $M'_{\mathcal{I}}$ is the union of all M_{I_k} for $I \in \mathcal{I}$

$$\mathrm{M}_f(X, \mathcal{I}) = M'_{\mathcal{I}} \circ Sh_{\mathcal{I}} \circ Sp_{\mathcal{I}} \circ M_{\mathcal{I}}(\mathrm{R}_f(X))$$

Formula Reeb graph $\rightarrow$ Mapper

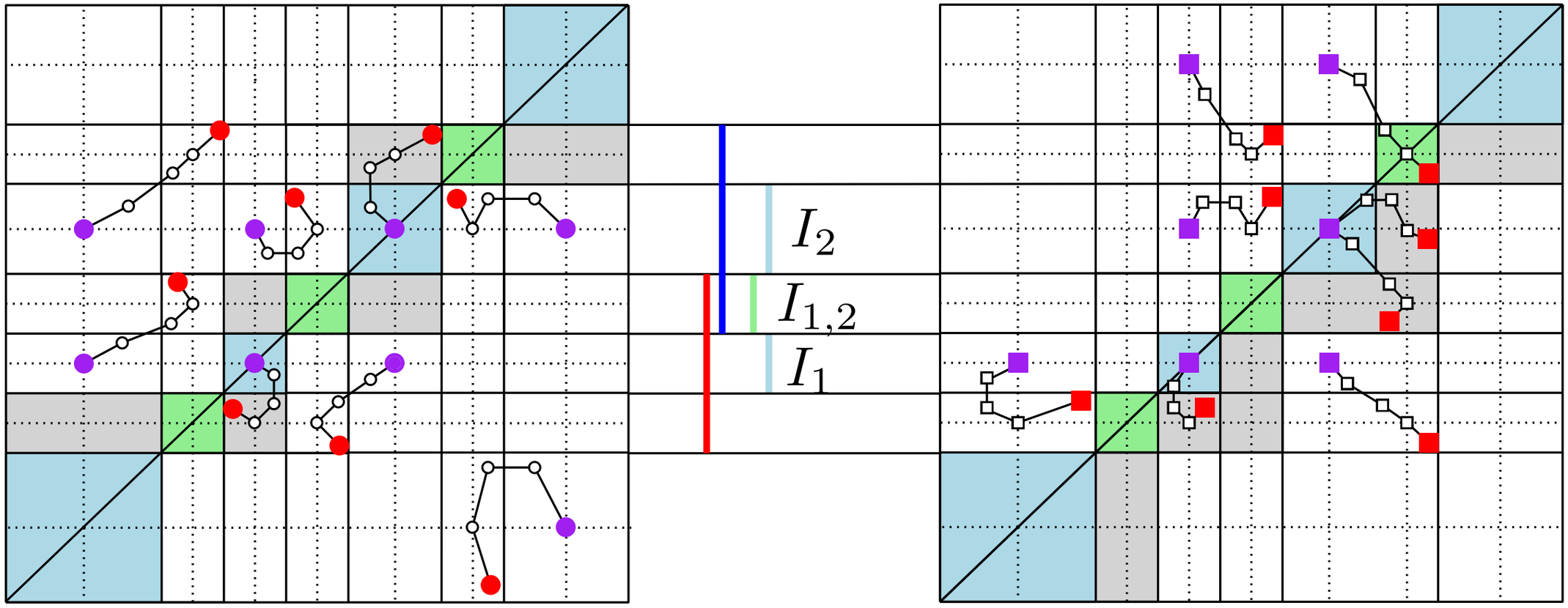
Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$



$$\mathrm{M}_f(X, \mathcal{I}) = M'_{\mathcal{I}} \circ Sh_{\mathcal{I}} \circ Sp_{\mathcal{I}} \circ M_{\mathcal{I}}(\mathrm{R}_f(X))$$

Formula Reeb graph $\rightarrow$ Mapper

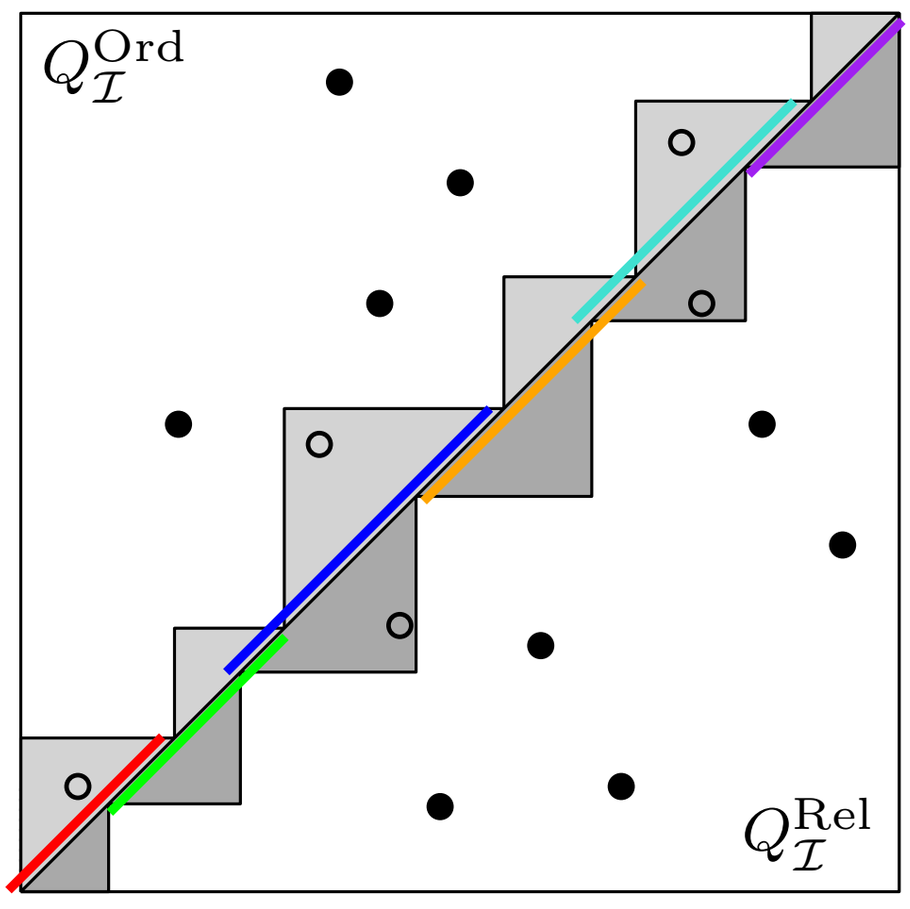
Let $\mathcal{I}$ be the cover of $\mathrm{im}(f)$

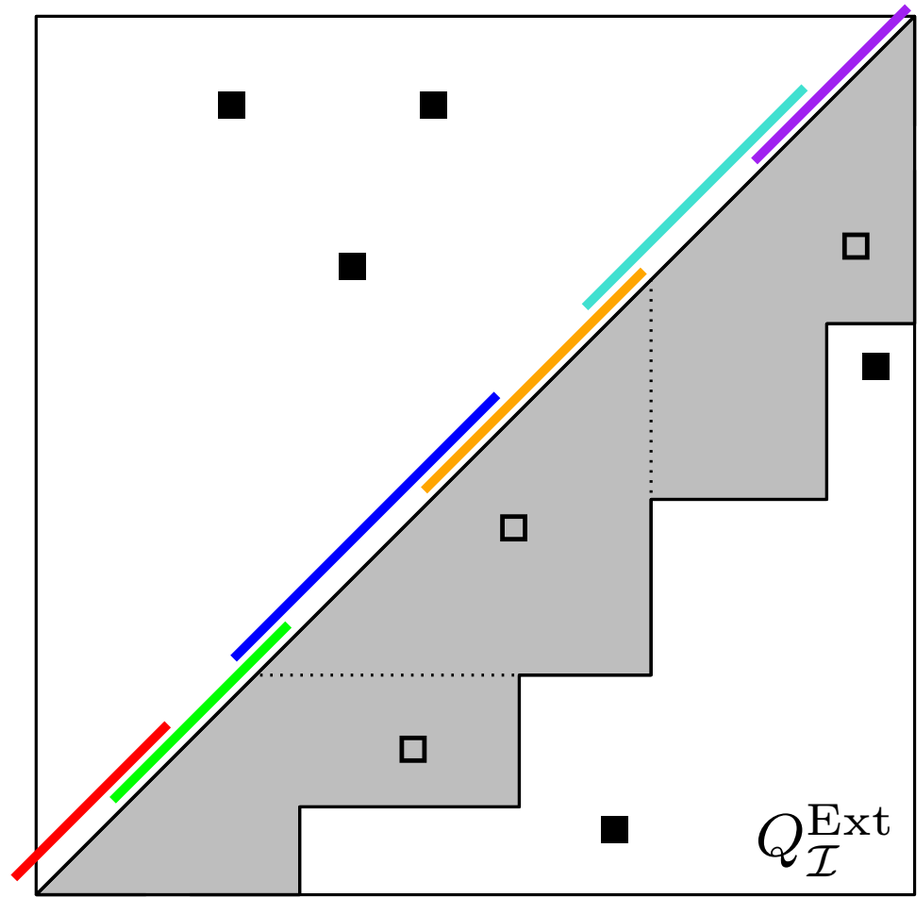


$$\mathrm{M}_f(X, \mathcal{I}) = M'_{\mathcal{I}} \circ Sh_{\mathcal{I}} \circ Sp_{\mathcal{I}} \circ M_{\mathcal{I}}(\mathrm{R}_f(X))$$

Descriptor for Mapper

Def: $D_{\tilde{f},\mathcal{I}} := \mathrm{Ord}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ord}} \cup \mathrm{Rel}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Rel}} \cup \mathrm{Ext}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ext}}$



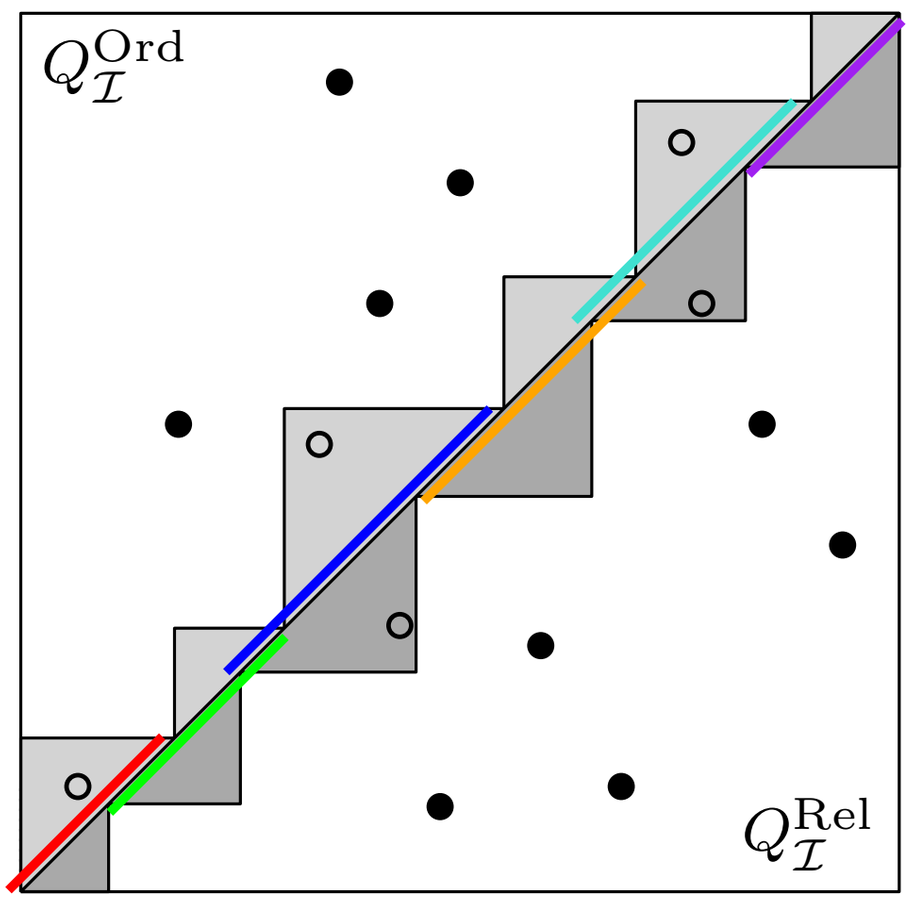


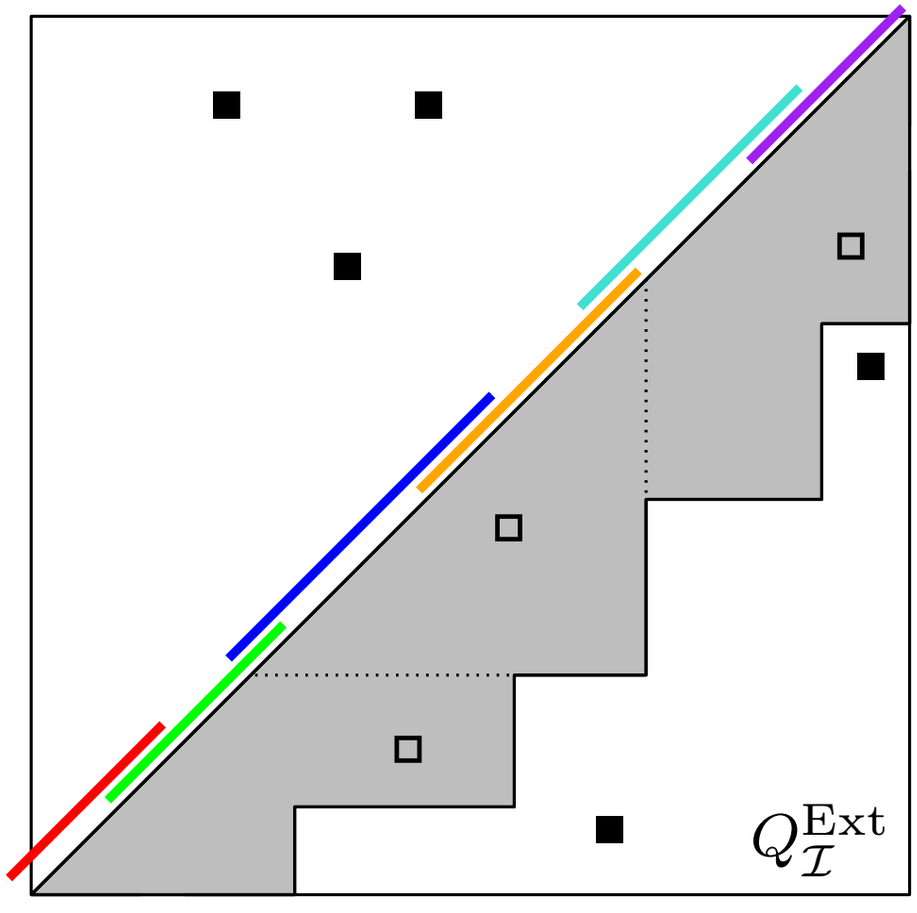
Descriptor for Mapper

Def: $D_{\tilde{f},\mathcal{I}} := \mathrm{Ord}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ord}} \cup \mathrm{Rel}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Rel}} \cup \mathrm{Ext}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ext}}$

Thm: $D_{\tilde{f},\mathcal{I}}$ provides a **bag-of-features** descriptor for $\mathrm{M}_f(X,\mathcal{I})$:

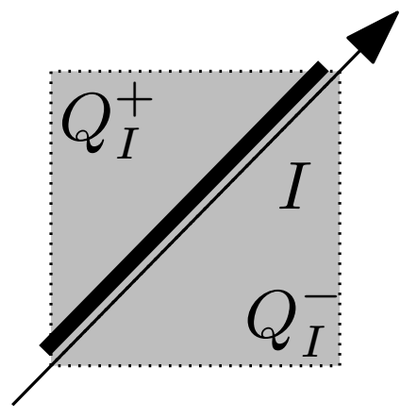
$\mathrm{Ord}_0 \longleftrightarrow$ downward branches	$\mathrm{Ext}_0 \longleftrightarrow$ trunks (cc)
$\mathrm{Rel}_1 \longleftrightarrow$ upward branches	$\mathrm{Ext}_1 \longleftrightarrow$ loops

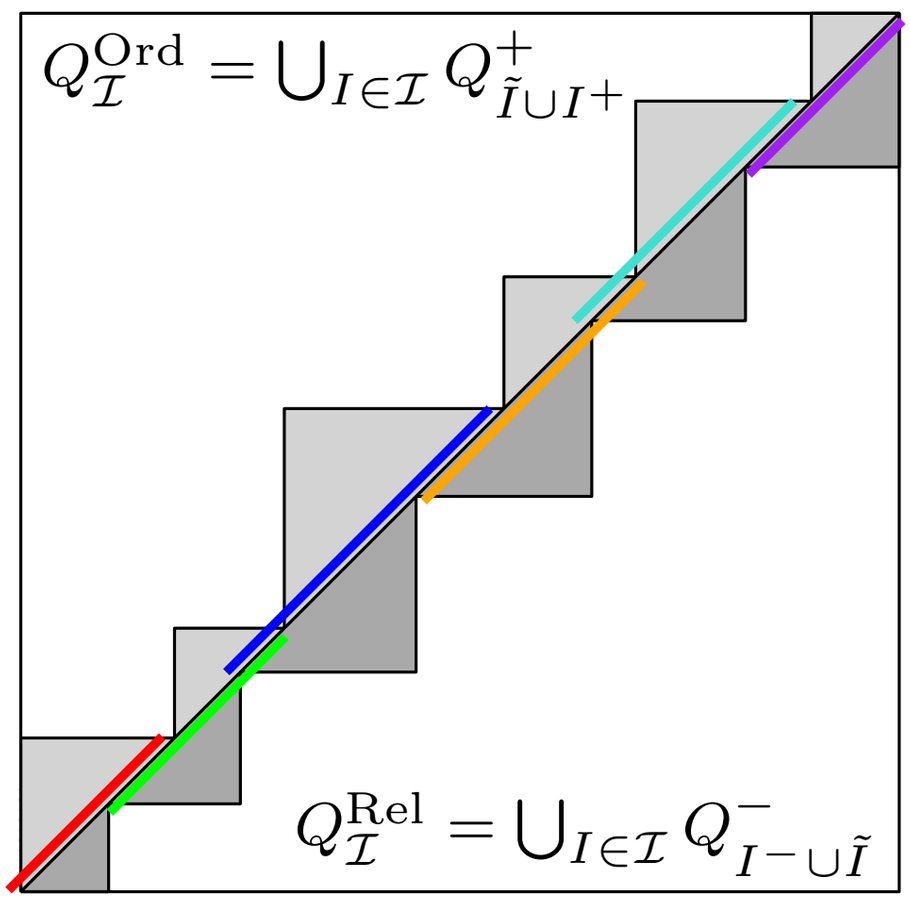


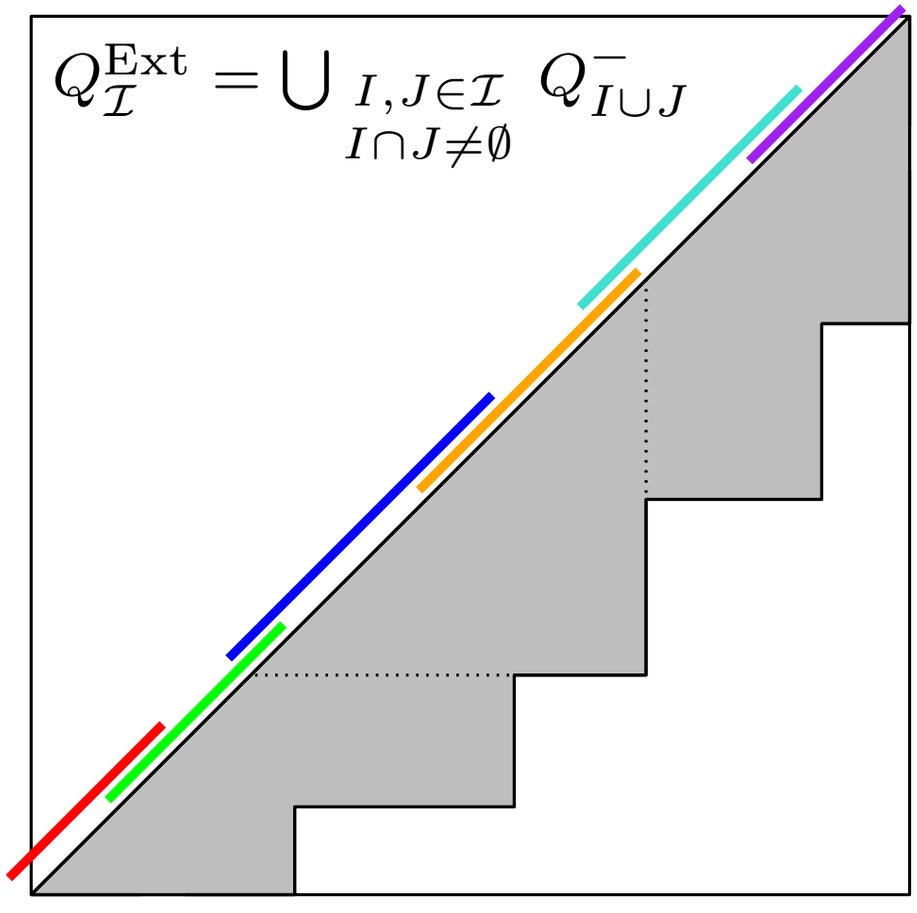


Descriptor for Mapper

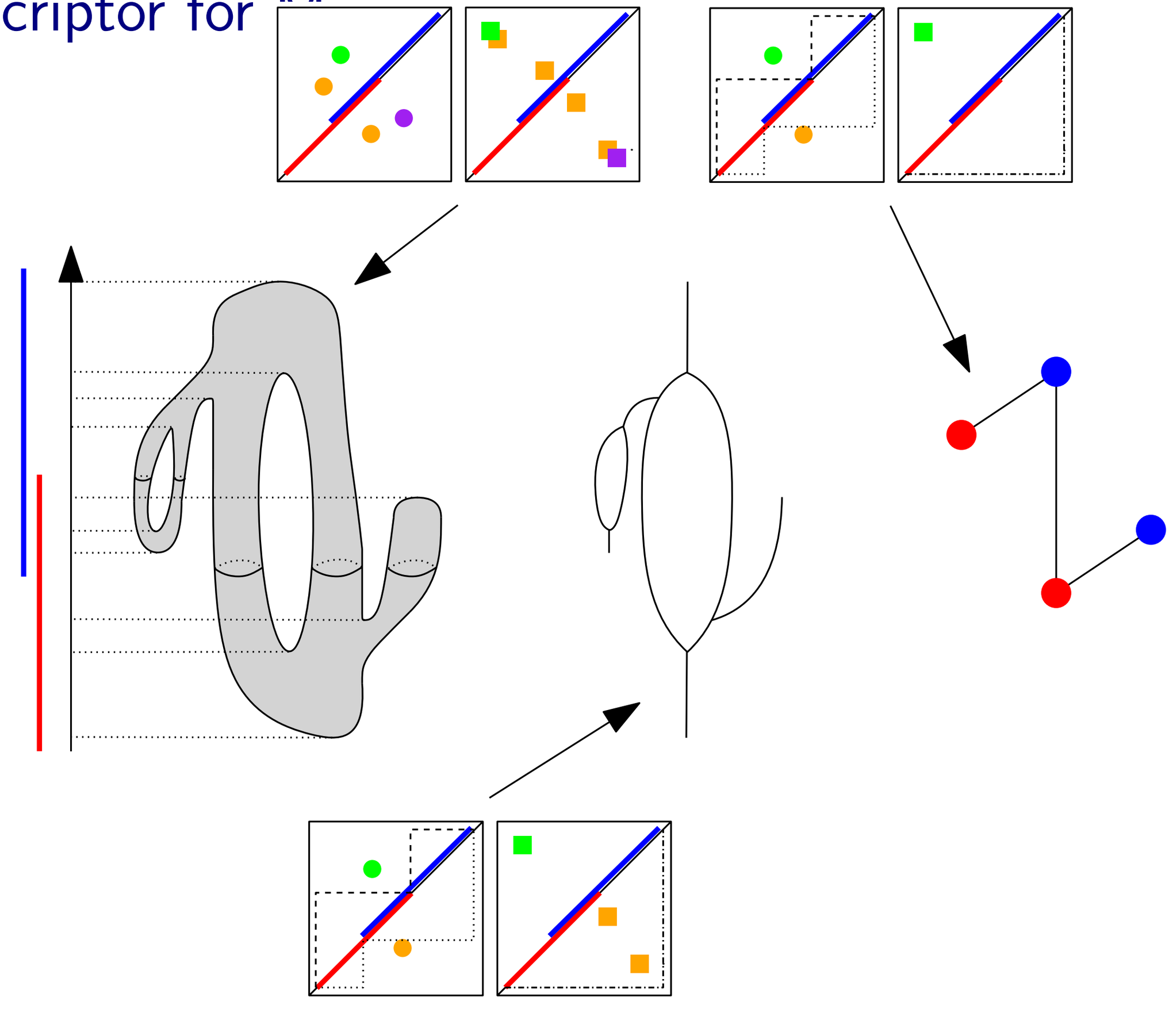
Let $\mathcal{I}$ minimal cover of $\mathrm{Im} f \subseteq \mathbb{R}$. For $I \in \mathcal{I}$, let $I = I^- \sqcup \tilde{I} \sqcup I^+$, and define the staircases $Q^{\cdot}_{\mathcal{I}}$ with:







Descriptor for M



Structure of Mapper

Def: $D_{\tilde{f},\mathcal{I}} := \mathrm{Ord}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ord}} \cup \mathrm{Rel}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Rel}} \cup \mathrm{Ext}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ext}}$

Thm: $D_{\tilde{f},\mathcal{I}}$ provides a **bag-of-features** descriptor for $\mathrm{M}_f(X,\mathcal{I})$:

$\mathrm{Ord}_0 \longleftrightarrow$ downward branches	$\mathrm{Ext}_0 \longleftrightarrow$ trunks (cc)
$\mathrm{Rel}_1 \longleftrightarrow$ upward branches	$\mathrm{Ext}_1 \longleftrightarrow$ loops

Cor: $D_{\tilde{f},\mathcal{I}} = D_{\tilde{f}}$ whenever the resolution r of $\mathcal{I}$ is smaller than the smallest distance from $D_{\tilde{f}} \setminus \Delta$ to the diagonal Δ.

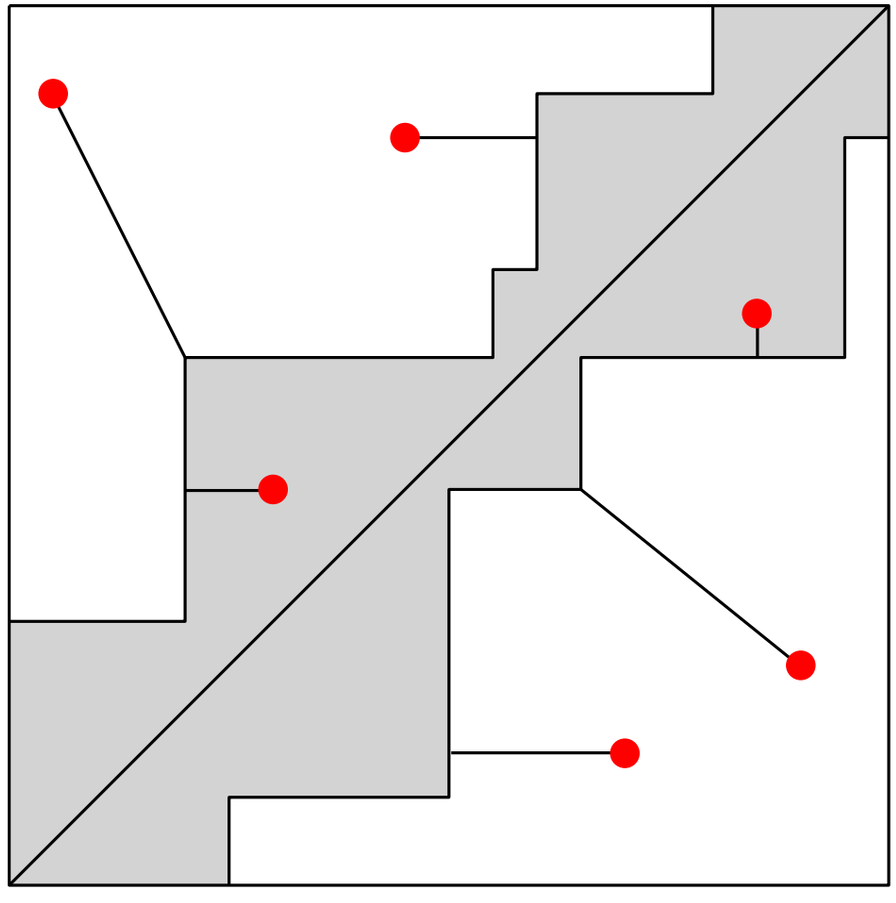
Stability of Mapper

Def: $D_{\tilde{f},\mathcal{I}} := \mathrm{Ord}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ord}} \cup \mathrm{Rel}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Rel}} \cup \mathrm{Ext}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ext}}$

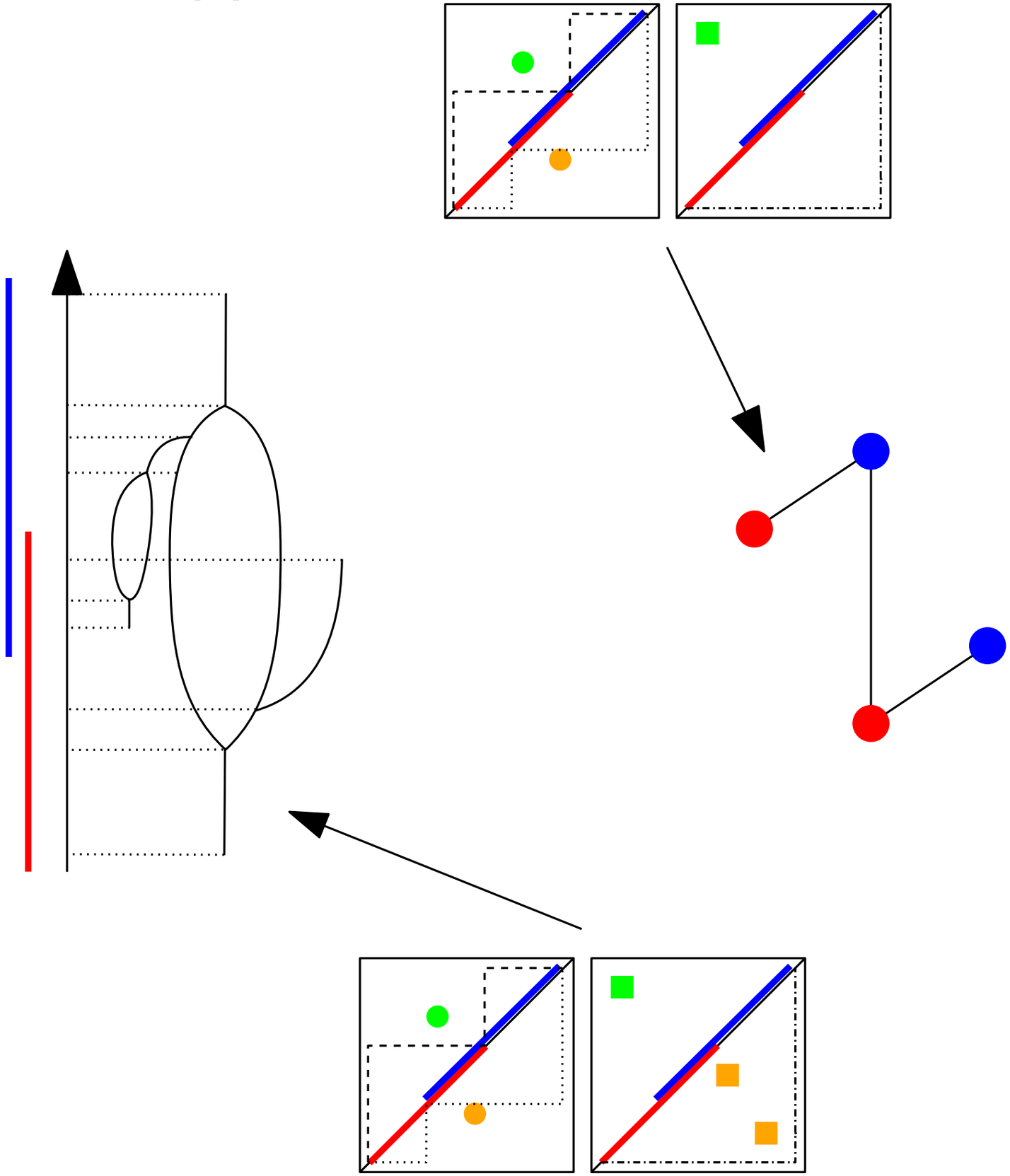
Thm: $D_{\tilde{f},\mathcal{I}}$ provides a **bag-of-features** descriptor for $\mathrm{M}_f(X,\mathcal{I})$:

$\mathrm{Ord}_0 \longleftrightarrow$ downward branches	$\mathrm{Ext}_0 \longleftrightarrow$ trunks (cc)
$\mathrm{Rel}_1 \longleftrightarrow$ upward branches	$\mathrm{Ext}_1 \longleftrightarrow$ loops

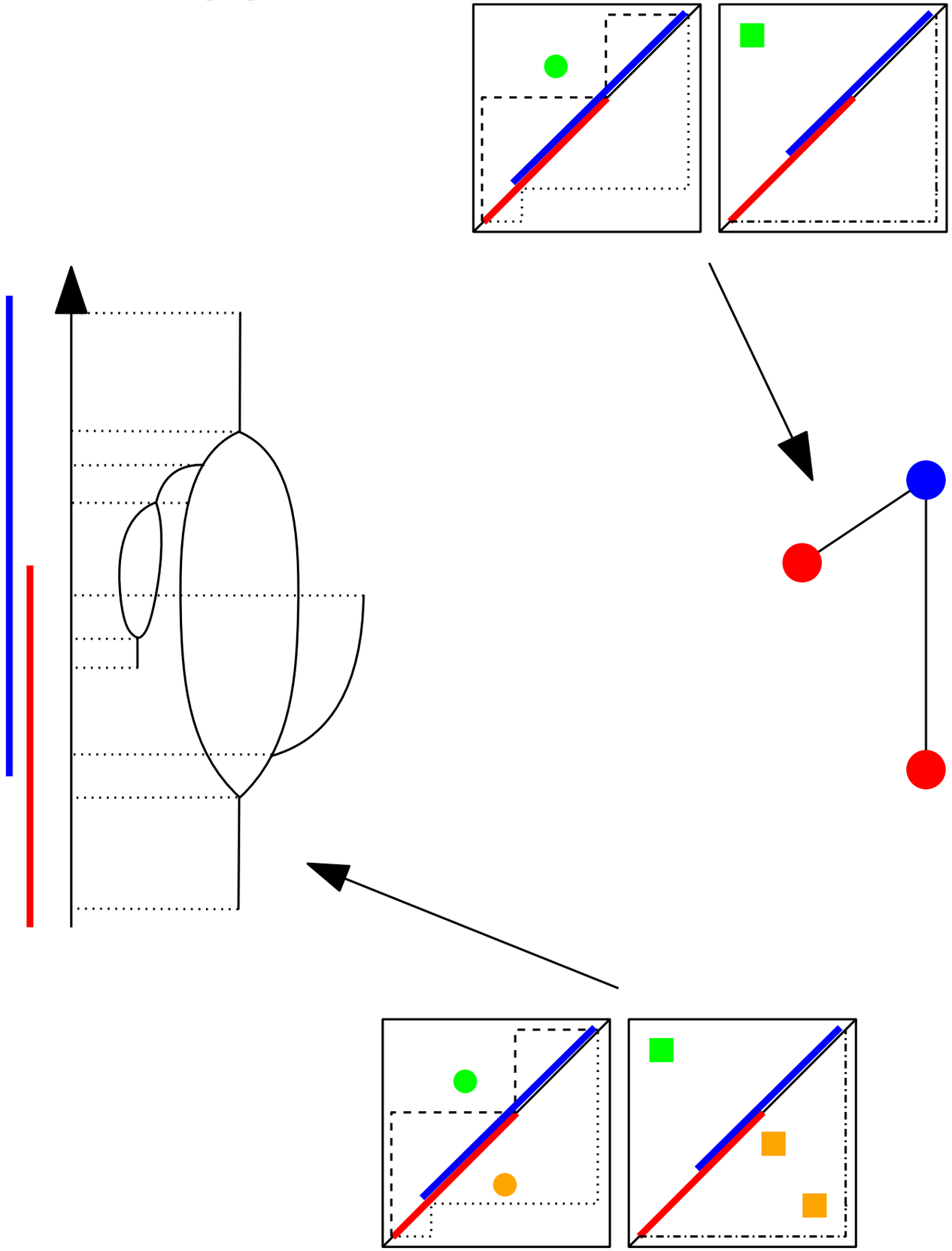
... and distance to staircase boundary measures (in-)stability of each feature w.r.t. perturbations of $(X, f, \mathcal{I})$



Stability of Mapper

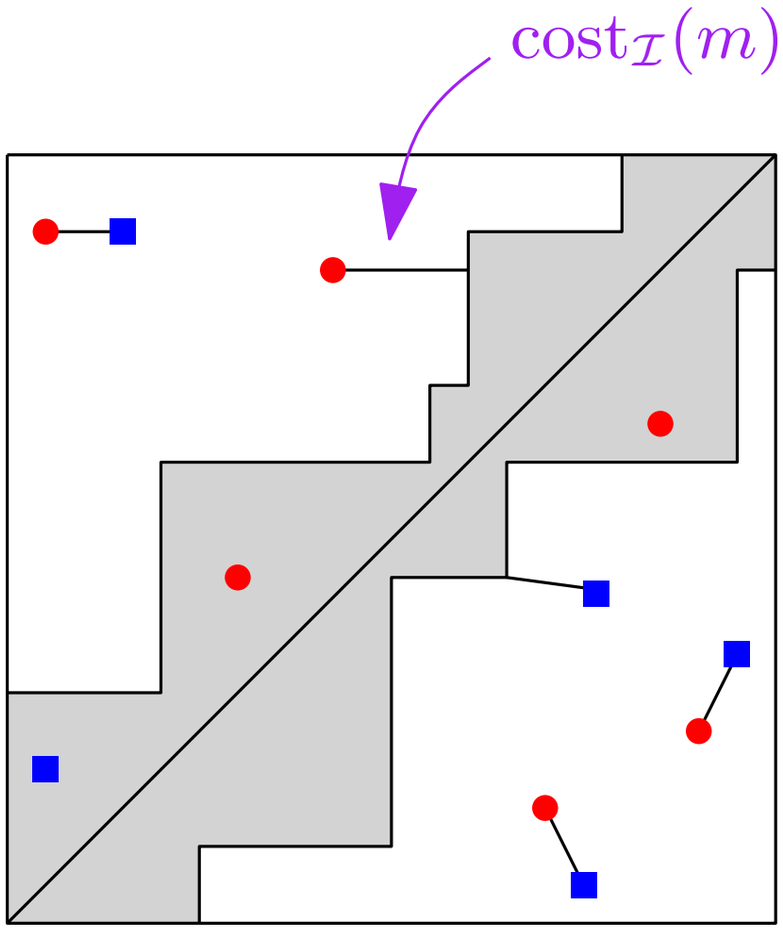


Stability of Mapper



Stability of Mapper

Def: $D_{\tilde{f},\mathcal{I}} := \mathrm{Ord}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ord}} \cup \mathrm{Rel}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Rel}} \cup \mathrm{Ext}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ext}}$



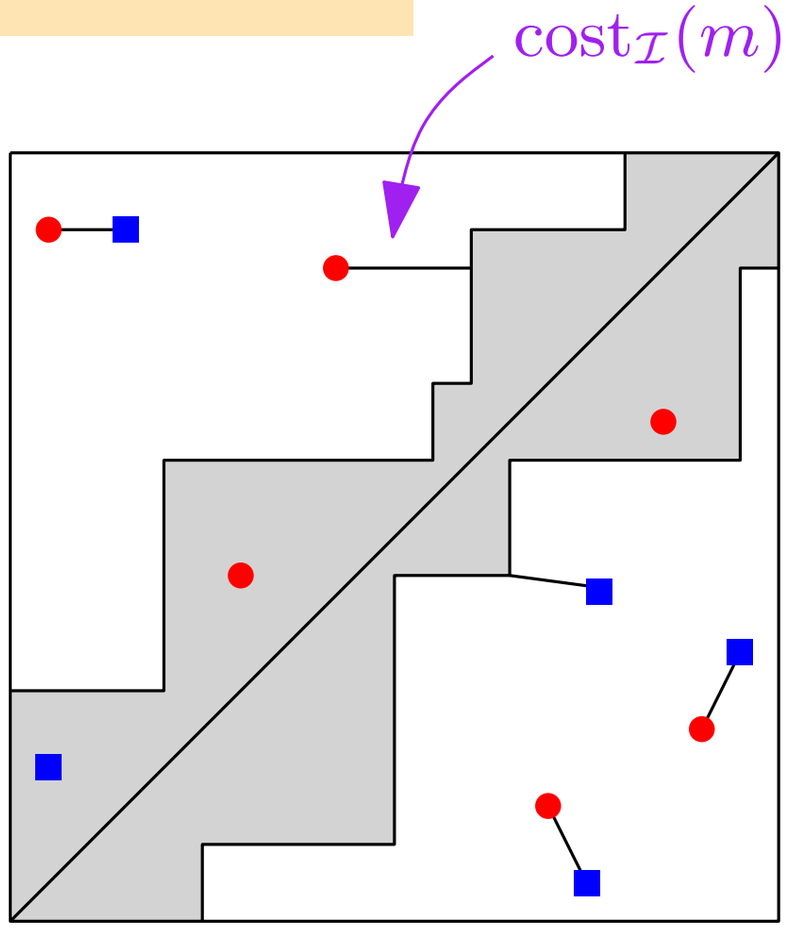
$m : D_{\tilde{f},\mathcal{I}} \longleftrightarrow D_{\tilde{f}',\mathcal{I}}$

Stability of Mapper

Def: $D_{\tilde{f},\mathcal{I}} := \mathrm{Ord}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ord}} \cup \mathrm{Rel}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Rel}} \cup \mathrm{Ext}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ext}}$

Thm: For any functions $f, f' : X \to \mathbb{R}$ of Morse type,

$$d_{\mathcal{I}}(D_{\tilde{f},\mathcal{I}},\ D_{\tilde{f}',\mathcal{I}}) \leq \|f - f'\|_\infty.$$



$m : D_{\tilde{f},\mathcal{I}} \longleftrightarrow D_{\tilde{f}',\mathcal{I}}$

Stability of Mapper

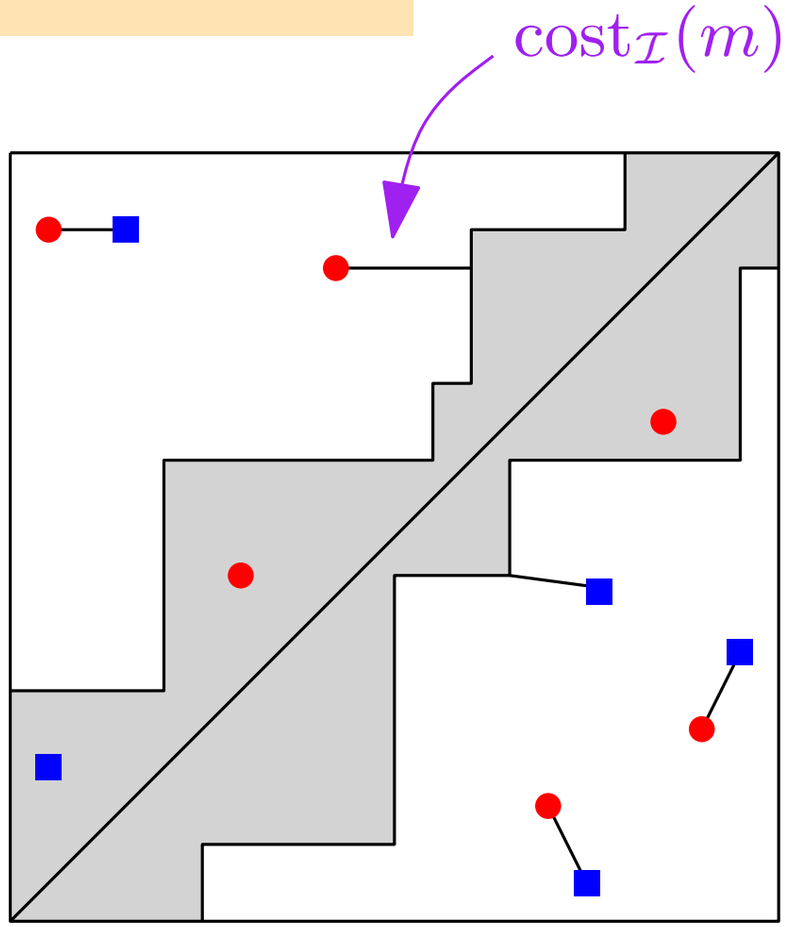
Def: $D_{\tilde{f},\mathcal{I}} := \mathrm{Ord}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ord}} \cup \mathrm{Rel}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Rel}} \cup \mathrm{Ext}\tilde{f} \setminus Q_{\mathcal{I}}^{\mathrm{Ext}}$

Thm: For any functions $f, f' : X \to \mathbb{R}$ of Morse type,

$$d_{\mathcal{I}}(D_{\tilde{f},\mathcal{I}},\ D_{\tilde{f}',\mathcal{I}}) \leq \|f - f'\|_\infty.$$

Extensions to:

- perturbations of X
- perturbations of $\mathcal{I}$



$m : D_{\tilde{f},\mathcal{I}} \longleftrightarrow D_{\tilde{f}',\mathcal{I}}$

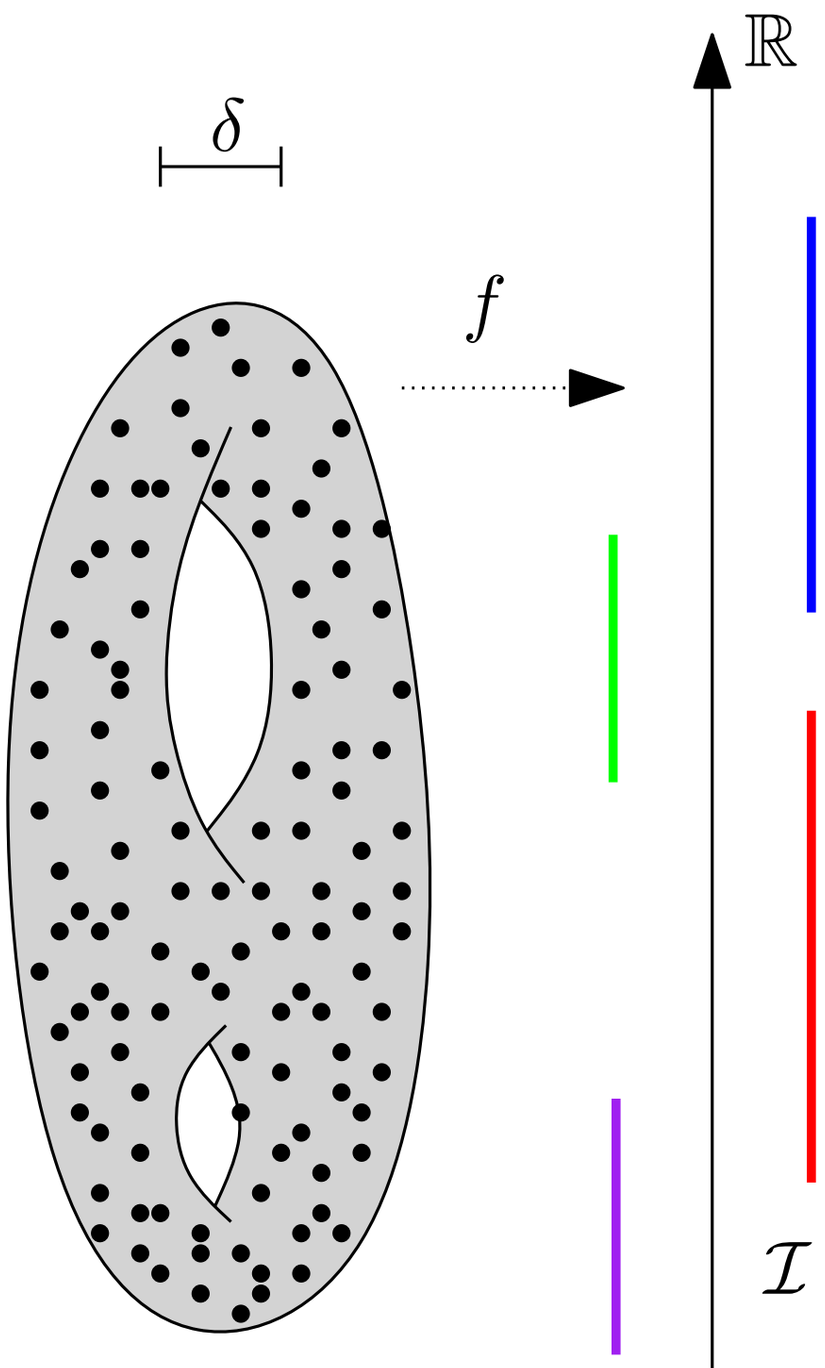
I. ToMATo algorithm

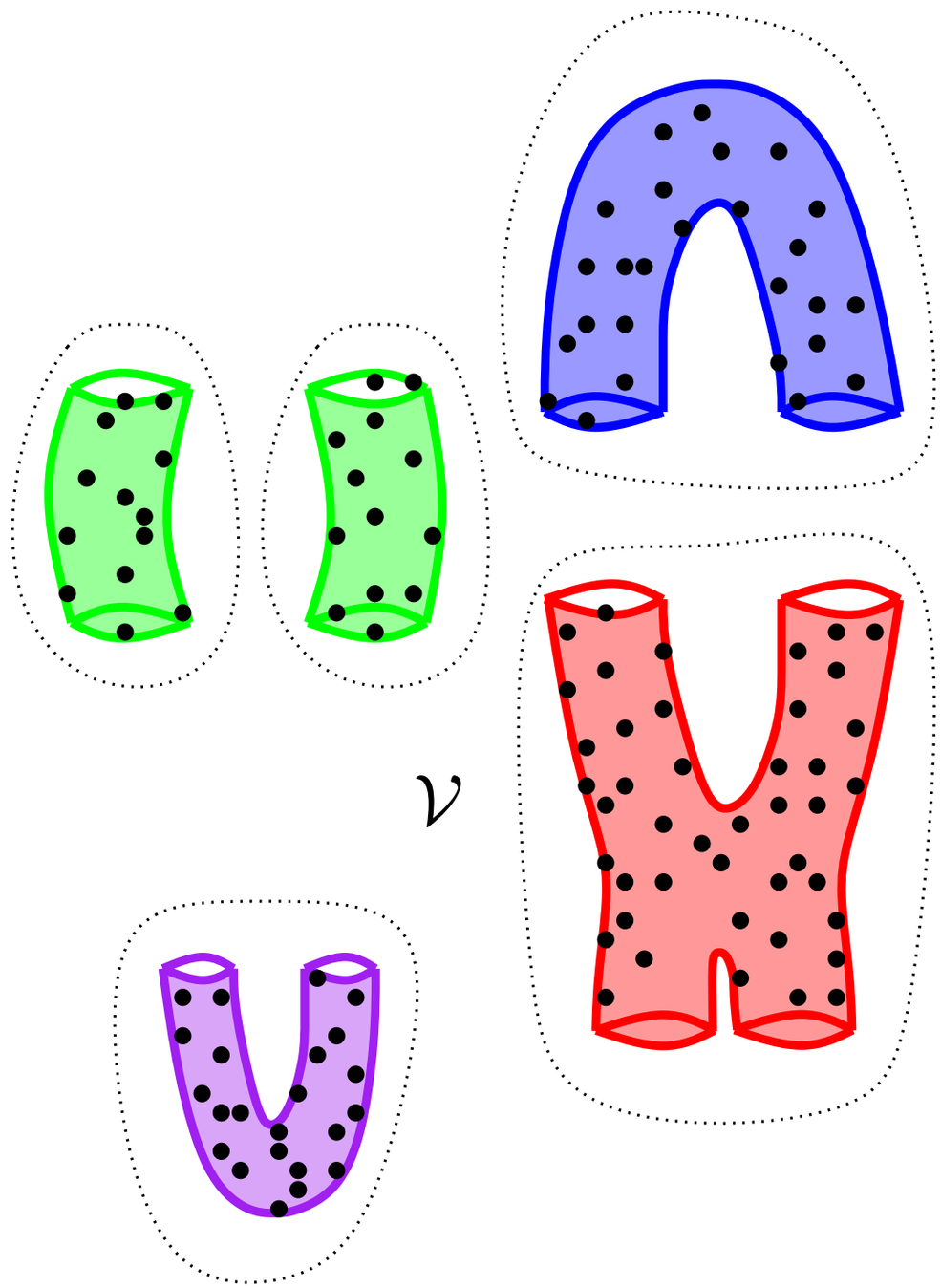
1. Introduction to hierarchical and mode-seeking clustering
2. ToMATo algorithm and guarantees

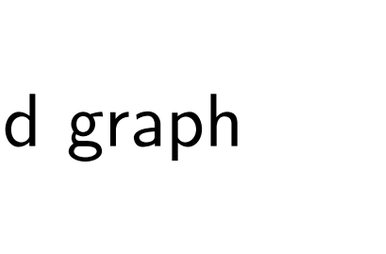
II. Mapper algorithm

1. Reeb spaces and Mappers
2. **Confidence intervals**

Mapper in practice





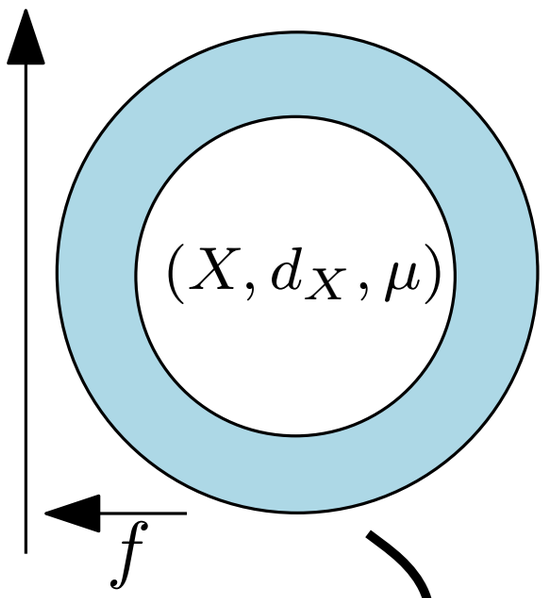


Mapper

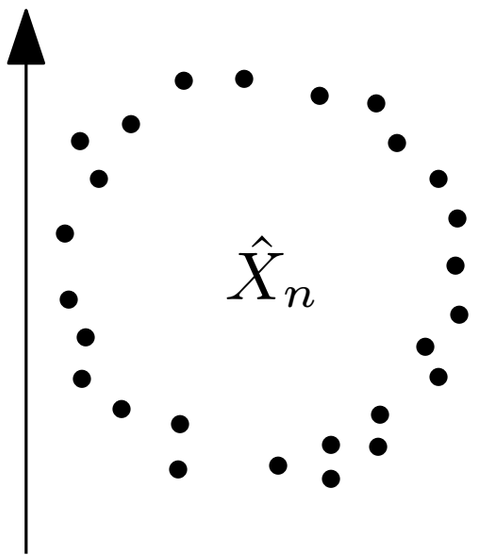
$\mathrm{M}^{\bullet}_{f,\delta}(\hat{X}_n, \mathcal{I})$

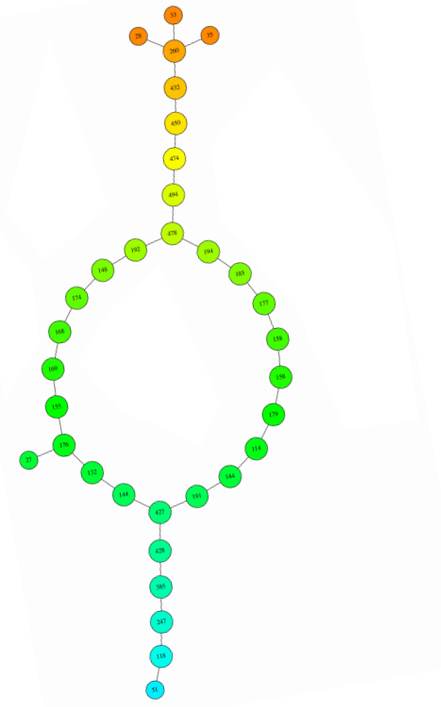
$G_\delta = \delta$-neighborhood graph

Statistics for Mapper

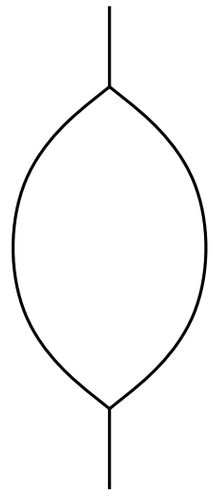


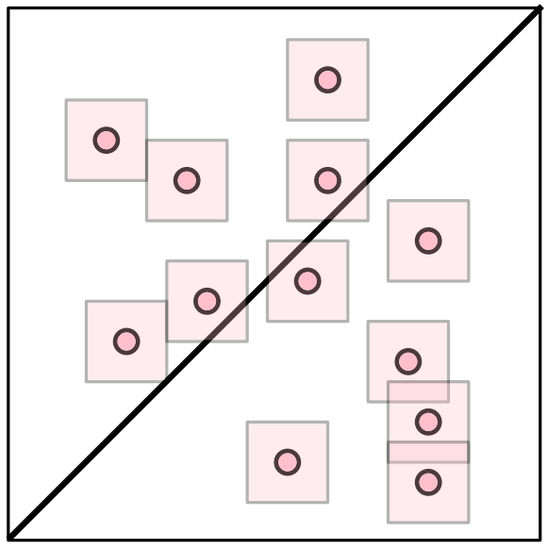
n points sampled
i.i.d. according to μ.
$+$ cover $\mathcal{I}$





$\mathrm{M}^{\bullet}_{f,\delta}(\hat{X}_n, \mathcal{I})$

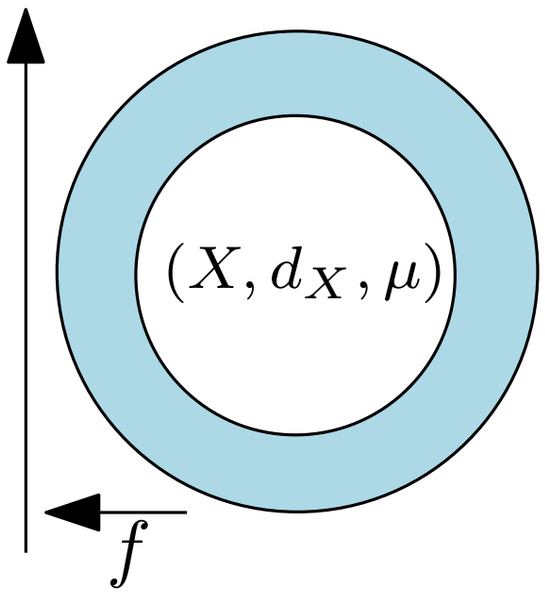




Questions:

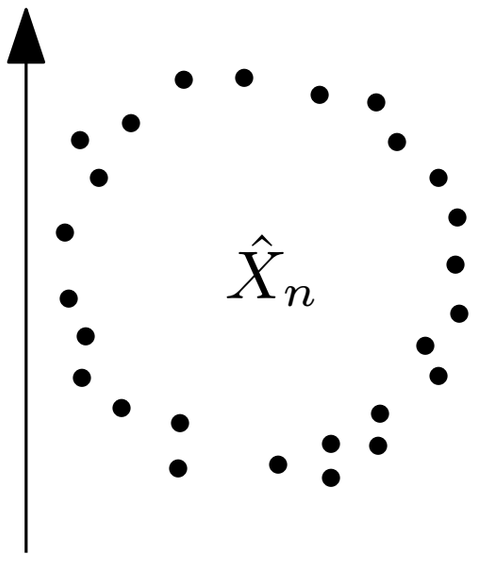
- Statistical properties of the estimator $\mathrm{M}^{\bullet}_{f,\delta}(\hat{X}_n, \mathcal{I})$?
- Convergence to the ground truth $\mathrm{R}_f(X)$ in d_b? Deviation bounds?

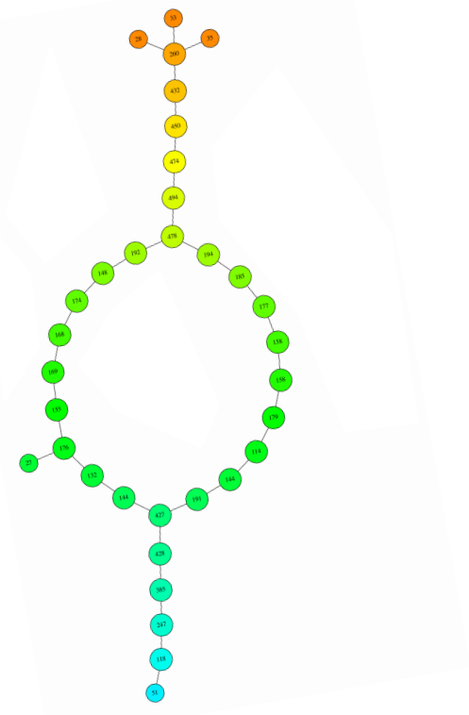
Statistics for Mapper



n points sampled i.i.d. according to μ.

$+$ cover $\mathcal{I}$





Let $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I})$ denote $\mathrm{M}_f(G_\delta, \mathcal{I})$

1. Link between $\mathrm{R}_f(X)$ and $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{V})$?

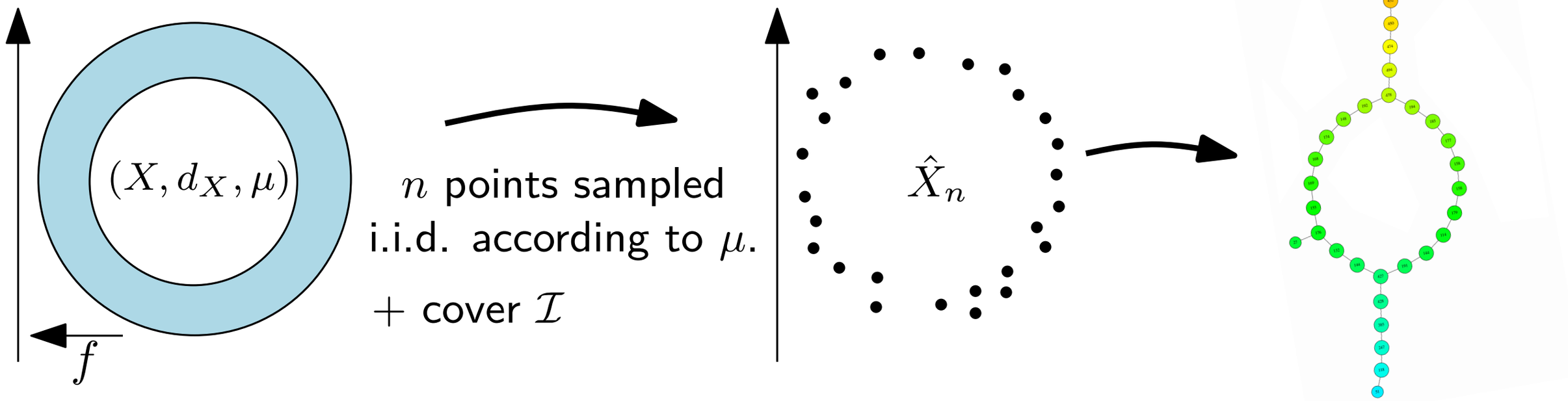
a. support $\to$ δ-neighborhood graph
$X \to G_\delta(\hat{X}_n)$

b. Reeb graph $\to$ Mapper

2. Link between $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I})$ and $\mathrm{M}^\bullet_{f,\delta}(\hat{X}_n, \mathcal{I})$?

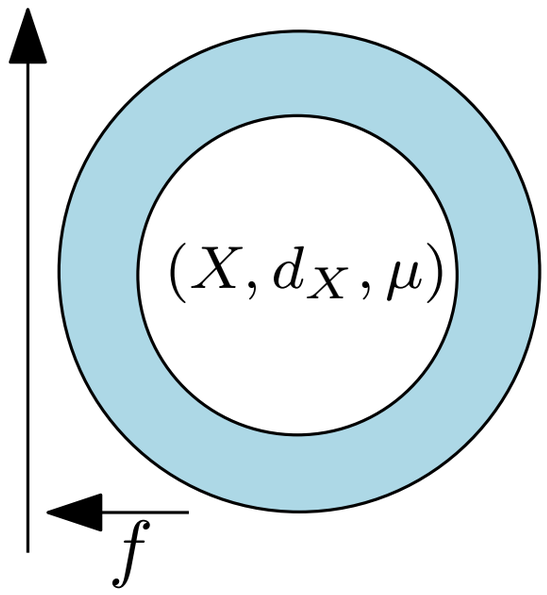
intersections given by metric graph $\to$ intersections given by points

Statistics for Mapper



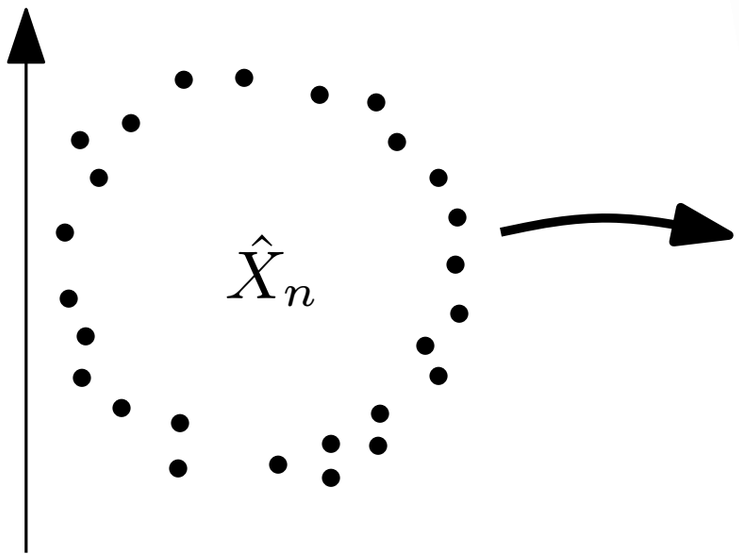
1. Link between $\mathrm{R}_f(X)$ and $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I})$?

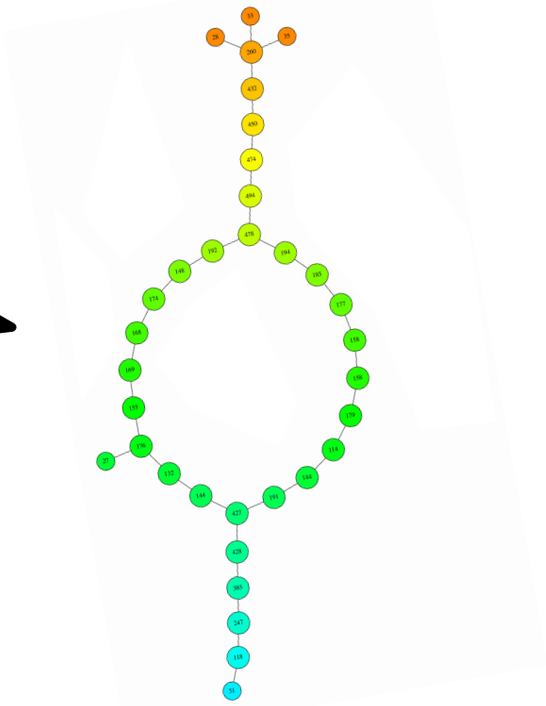
Statistics for Mapper



n points sampled i.i.d. according to μ.

$+$ cover $\mathcal{I}$





1. Link between $\mathrm{R}_f(X)$ and $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I})$?

   support $\to$ δ-neighborhood graph

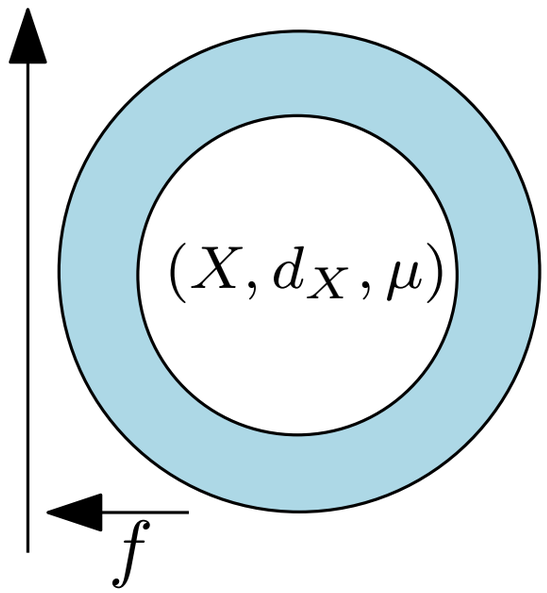
[*Reeb Graphs: Approximation and Persistence*, Dey, Wang, DCG, 2013]

Thm: If $4d_H(X, \hat{X}_n) \leq \delta \leq \min\left\{\frac{1}{4}\mathrm{rch}(X), \frac{1}{4}\rho(X)\right\}$, then one has:

$$d_b(D_{\tilde{f}}, D_{\tilde{f}_{\mathrm{PL}}}) \leq 2\omega(\delta)$$

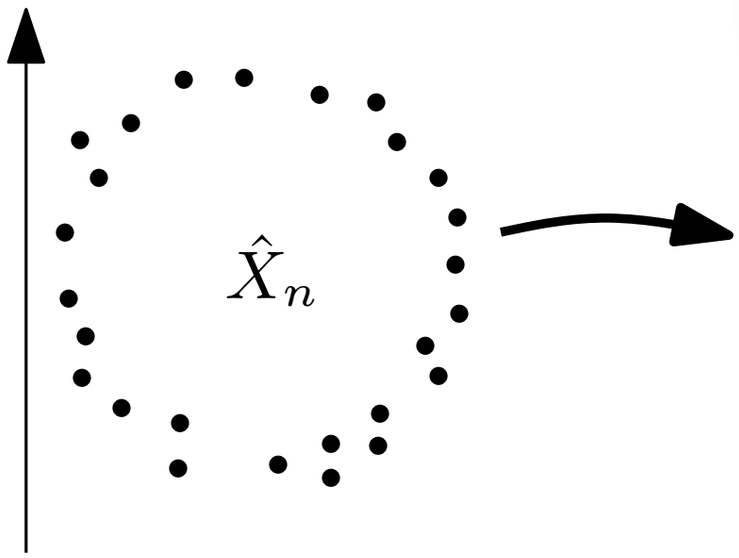
where $\tilde{f}_{\mathrm{PL}}$ is the piecewise-linear approximation of $\tilde{f}$ defined on $\mathrm{R}_f(G_\delta(\hat{X}_n))$.

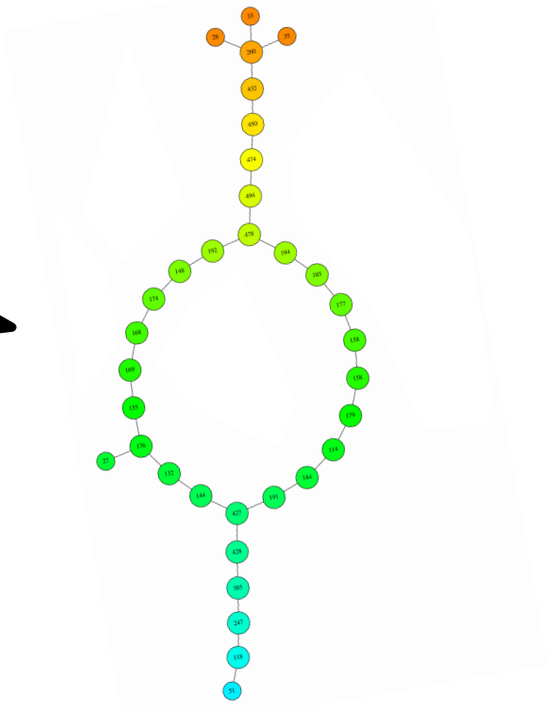
Statistics for Mapper



n points sampled i.i.d. according to μ.

$+$ cover $\mathcal{I}$





1. Link between $\mathrm{R}_f(X)$ and $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I})$?

support $\to$ δ-neighborhood graph

[*Reeb Graphs: Approximation and Persistence*, Dey, Wang, DCG, 2013]

Thm: If $4d_H(X, \hat{X}_n) \leq \delta \leq \min\left\{\frac{1}{4}\mathrm{rch}(X), \frac{1}{4}\rho(X)\right\}$, then one has:

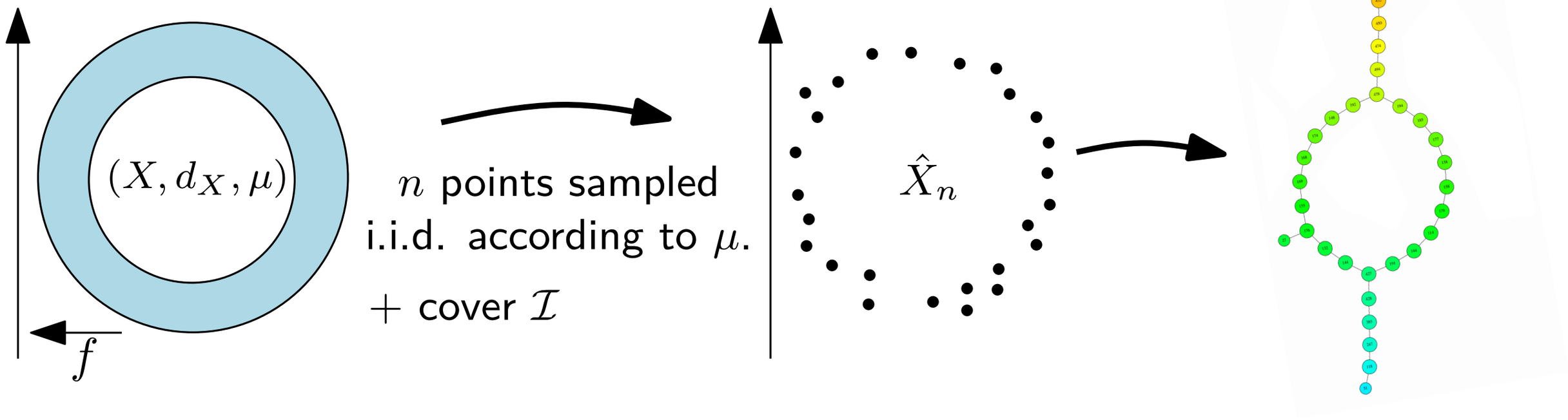
$$d_b(D_{\tilde{f}}, D_{\tilde{f}_{\mathrm{PL}}}) \leq 2\omega(\delta)$$

where $\tilde{f}_{\mathrm{PL}}$ is the piecewise-linear approximation of $\tilde{f}$ defined on $\mathrm{R}_f(G_\delta(\hat{X}_n))$.

Reeb graph $\to$ Mapper

Thm: $d_b(D_{\tilde{f}_{\mathrm{PL}}}, D_{\tilde{f}_{\mathrm{PL}},\mathcal{I}}) \leq r$.

Statistics for Mapper



1. Link between $\mathrm{R}_f(X)$ and $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I})$?

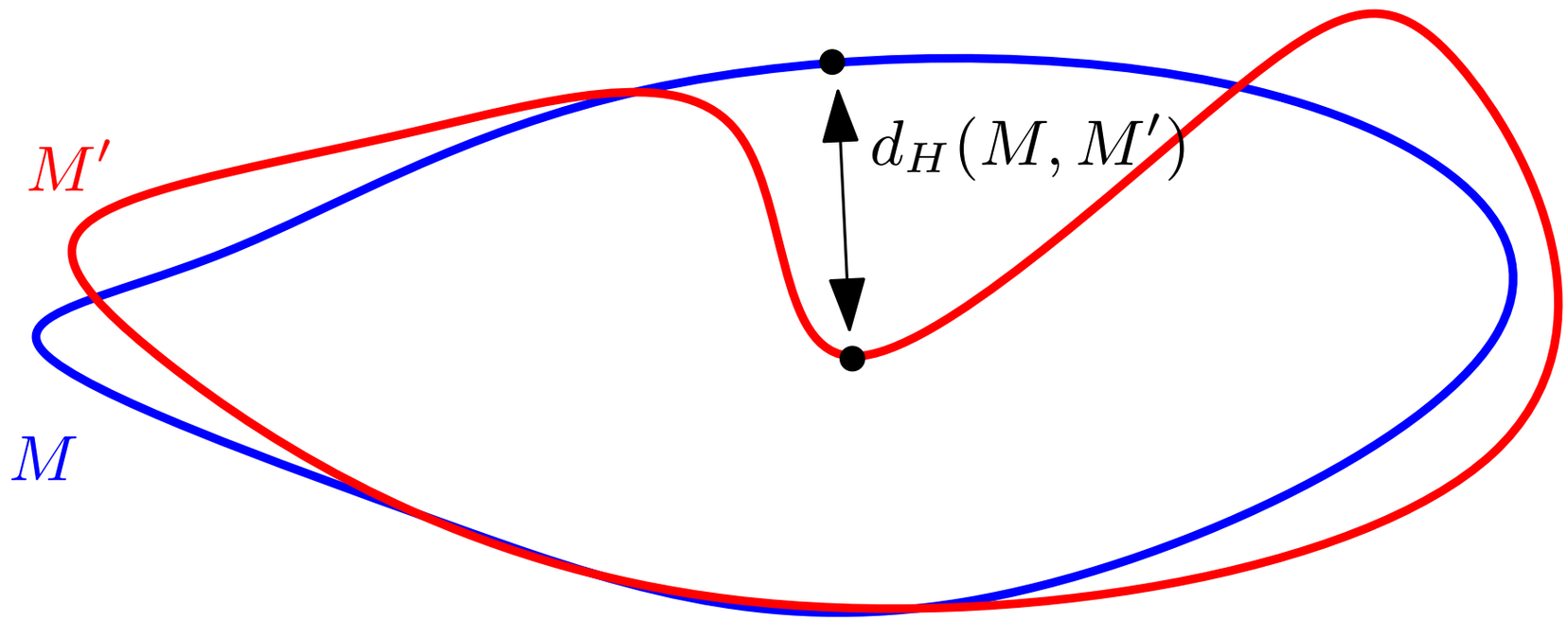
ω: modulus of continuity of f, defined as $\omega : \delta \mapsto \sup\{|f(x) - f(y)| \, : \, d(x,y) \leq \delta\}$

rch: reach of X.

ρ: radius of convexity of X: largest r s.t. geodesic balls of radius r are convex.

d_H: Hausdorff distance.

Statistics for Mapper



Def: The distance function to a compact $M \subset \mathbb{R}^d$, $d_M : \mathbb{R}^d \to \mathbb{R}_+$ is:

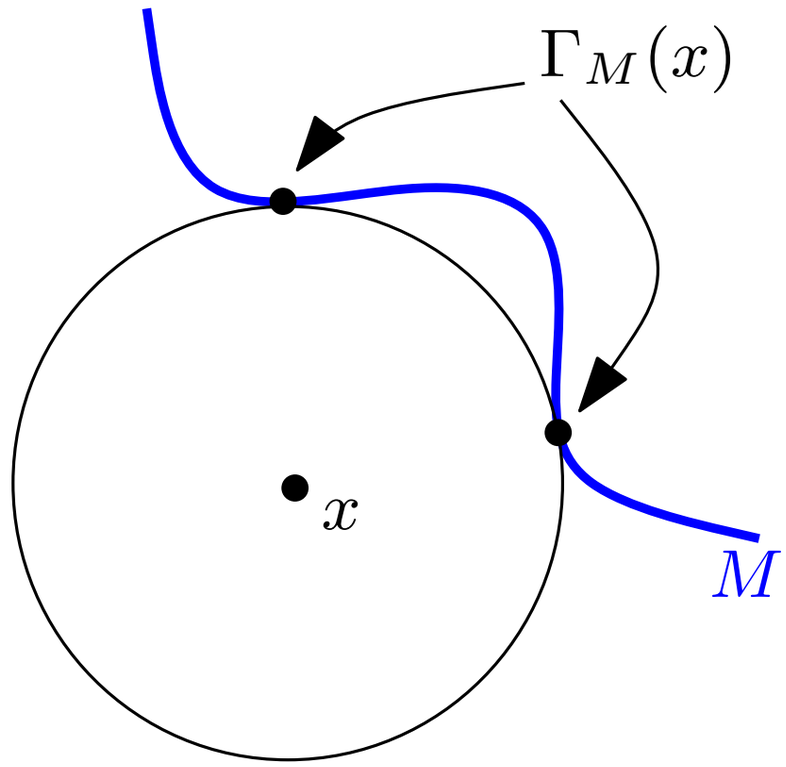
$$d_M(x) = \inf_{p \in M} \|x - p\|$$

Def: The Hausdorff distance between two compact sets $M, M' \subset \mathbb{R}^d$ is:

$$d_H(M, M') = \sup_{x \in \mathbb{R}^d} |d_M(x) - d_{M'}(x)|$$

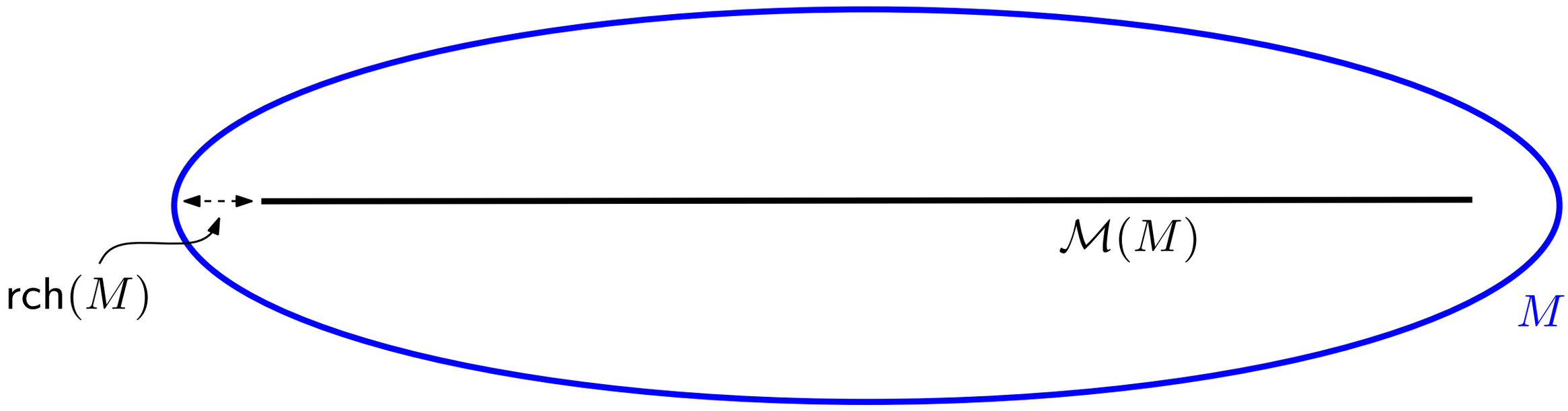
Statistics for Mapper

$$\Gamma_M(x) = \{y \in M : d_M(x) = \|x - y\|\}$$



Def: The medial axis of M is:

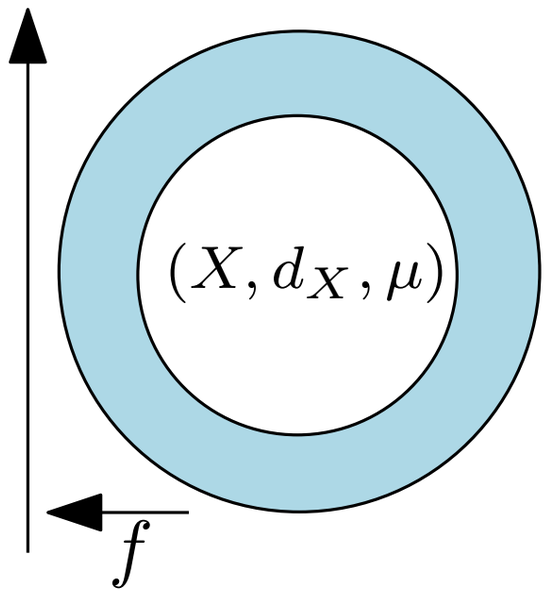
$$\mathcal{M}(M) = \{x \in \mathbb{R}^d : |\Gamma_M(x)| \geq 2\}$$



Def: The reach of M, rch(M) is the smallest distance from $\mathcal{M}(M)$ to M:

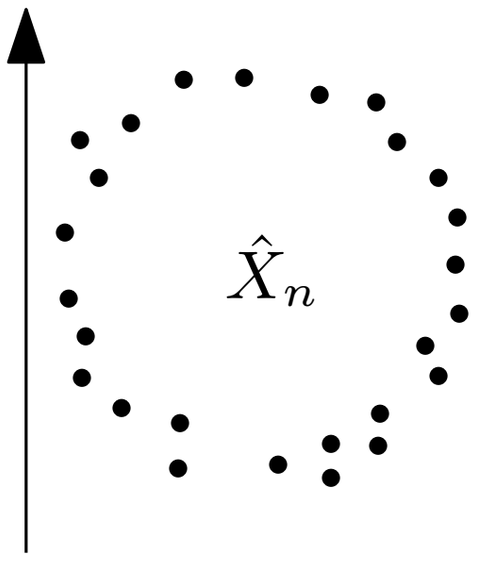
$$\text{rch}(M) = \inf_{y \in \mathcal{M}(M)} d_M(y)$$

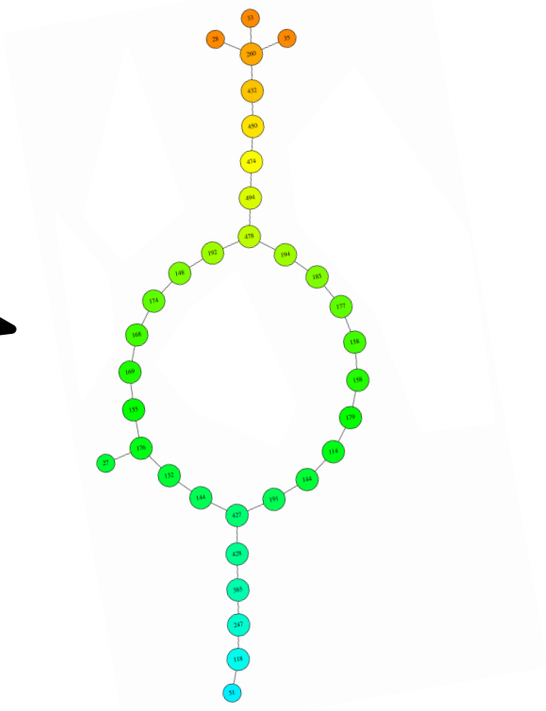
Statistics for Mapper



n points sampled i.i.d. according to μ.

$+$ cover $\mathcal{I}$





2. Link between $\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I})$ and $\mathrm{M}^{\bullet}_{f,\delta}(\hat{X}_n, \mathcal{I})$?

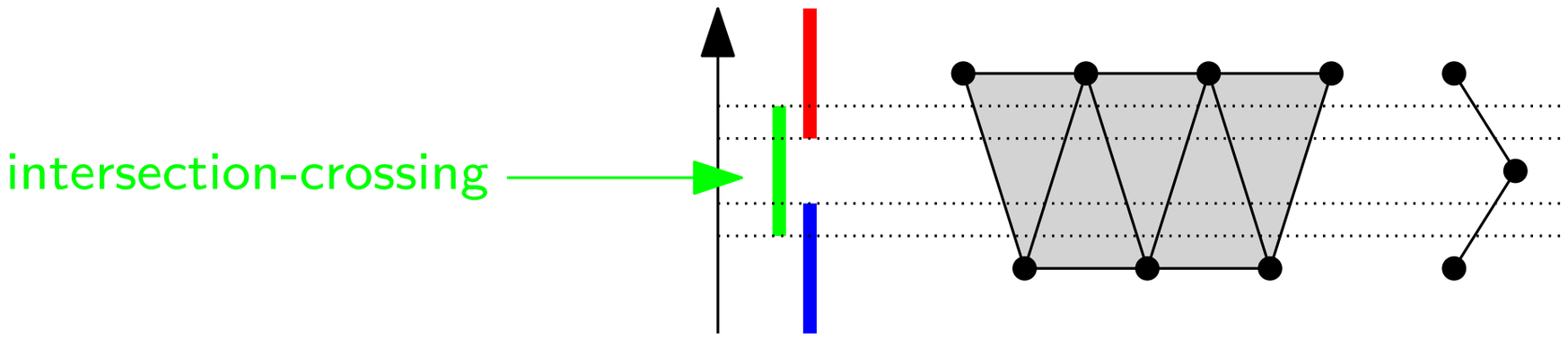
intersections given by metric graph $\rightarrow$ intersections given by points

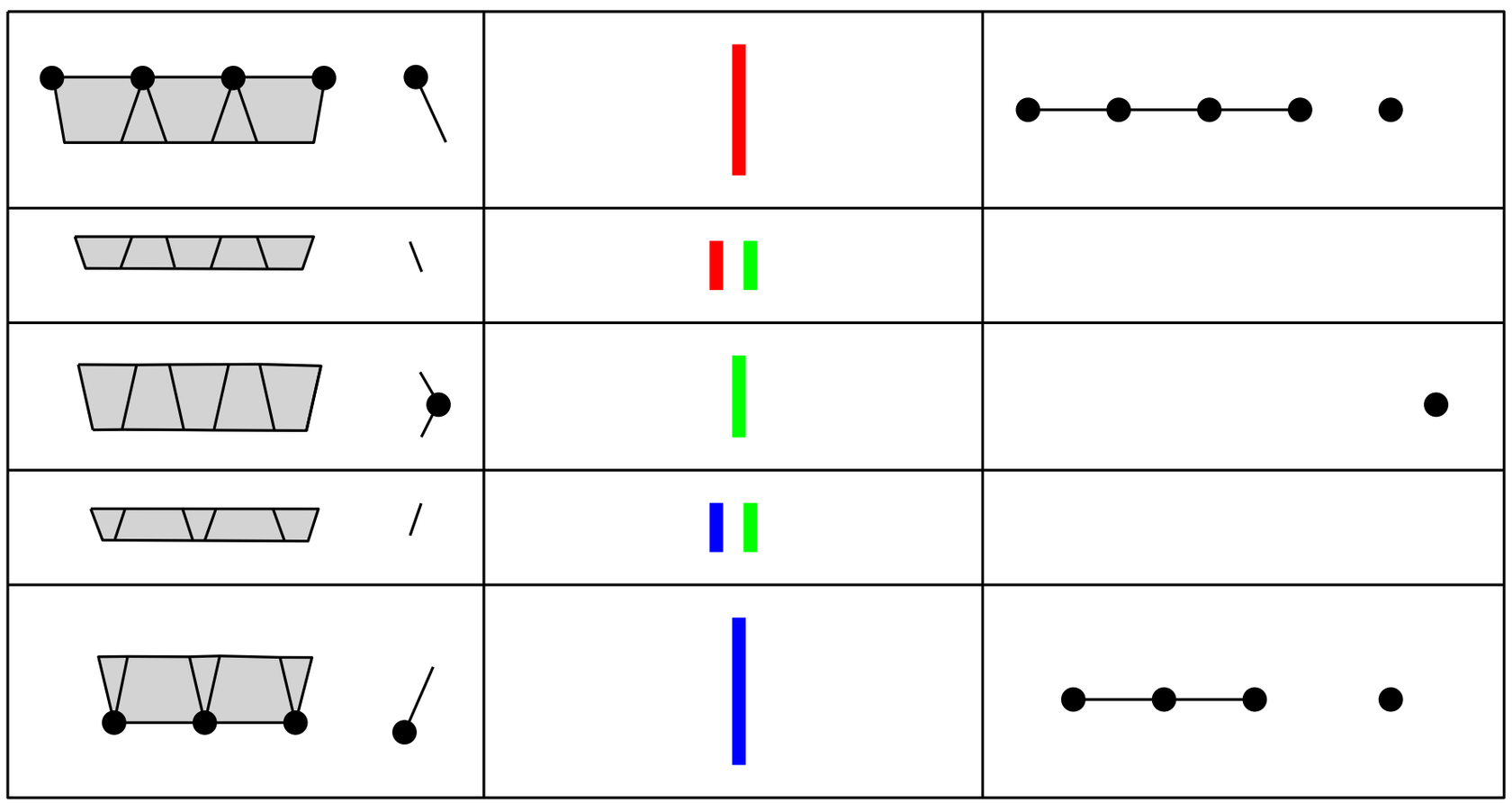
Thm: If there are no **intersection-crossing edges**, then

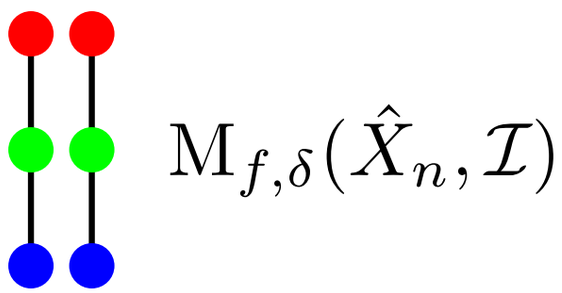
$$\mathrm{M}_{f,\delta}(\hat{X}_n, \mathcal{I}) \simeq \mathrm{M}^{\bullet}_{f,\delta}(\hat{X}_n, \mathcal{I})$$

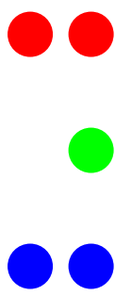
and thus we can define $D^{\bullet}_{\tilde{f},\mathcal{I}} := D_{\tilde{f},\mathcal{I}}$.

Statistics for Mapper

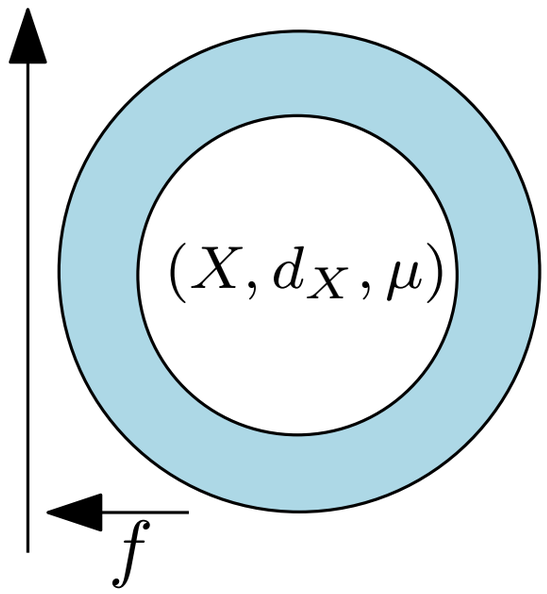






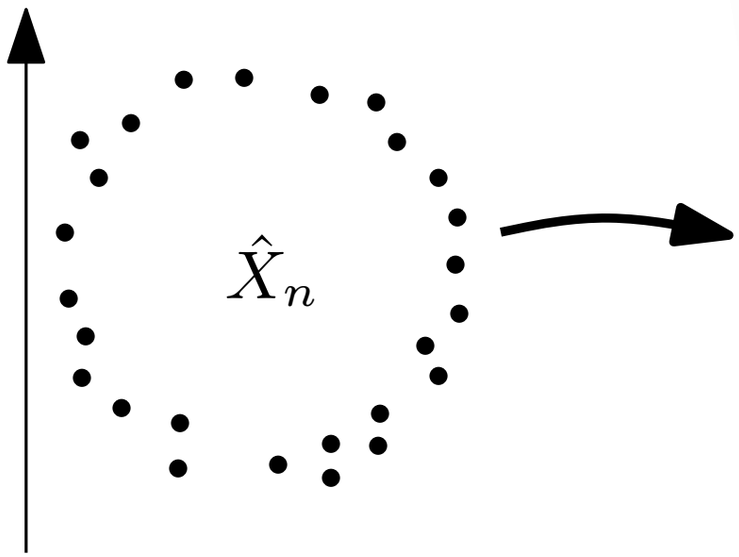


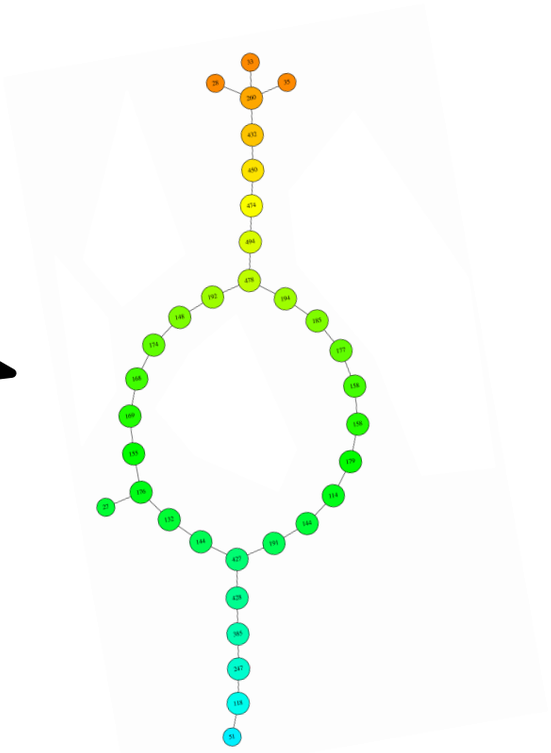
Statistics for Mapper



n points sampled i.i.d. according to μ.

$+$ cover $\mathcal{I}$





Gathering everything, it only remains to upper bound $d_H(X, \hat{X}_n)$ (which is random since $\hat{X}_n$ is random).

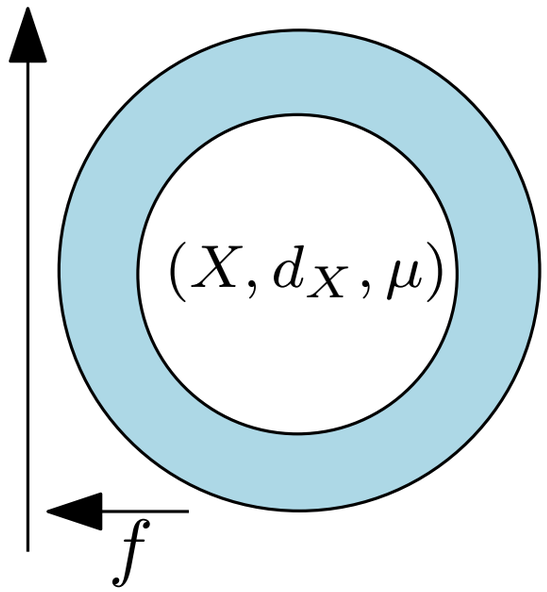
Hyp-Def: μ is called (a, b)-standard if:

$$\mu(B(x, r)) \geq \min\{1, ar^b\} \text{ for all } x \in X \text{ and } r > 0.$$

Then it is known that one has asymptotically almost surely:

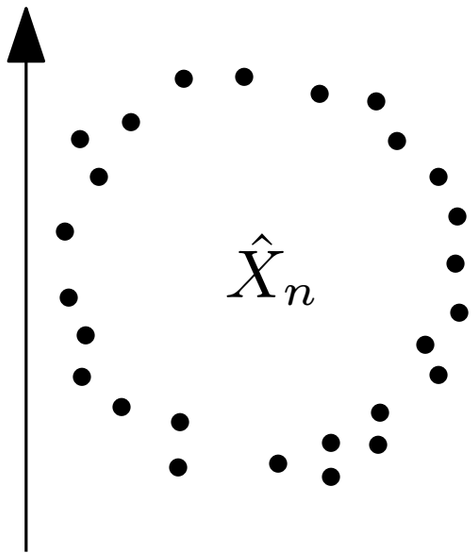
$$d_H(X, \hat{X}_n) \leq C(a, b) \left(\frac{\log n}{n}\right)^{1/b}$$

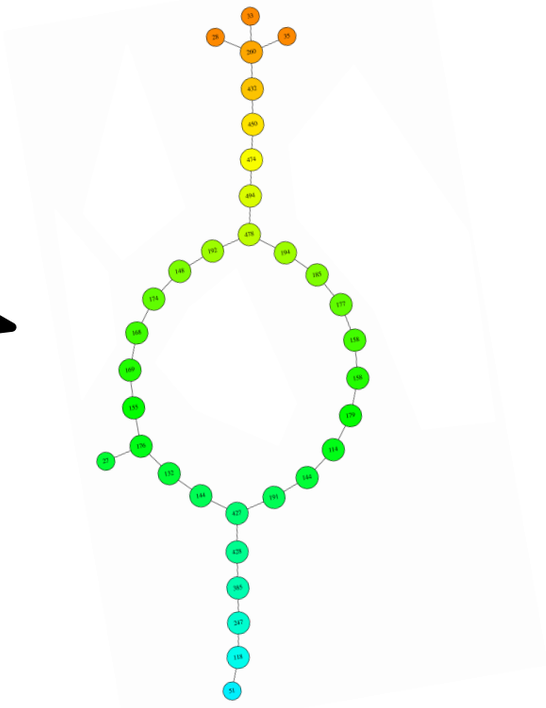
Statistics for Mapper



n points sampled i.i.d. according to μ.

$+$ cover $\mathcal{I}$





Thm: If μ is (a, b)-standard and f is c-Lipschitz then for:

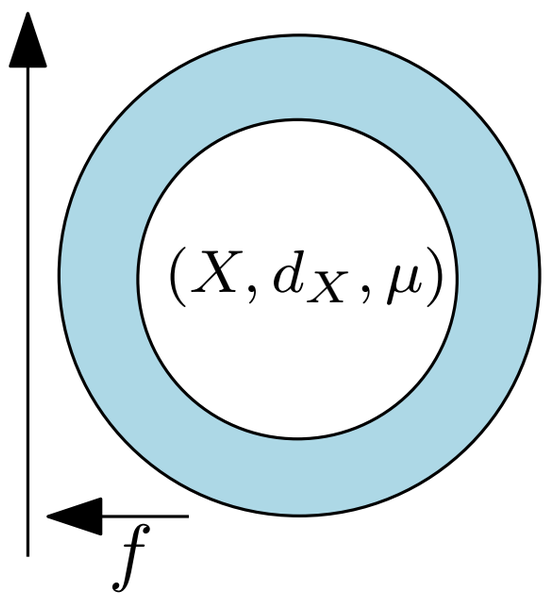
$$\delta_n = 4\left(\frac{2\log n}{an}\right)^{1/b},\ g_n \in \left(\tfrac{1}{3},\ \tfrac{1}{2}\right),\ r_n = \frac{c\delta_n}{g_n}, \qquad \text{one has } \forall \varepsilon > 0$$

$$\sup_{\mu \in \mathcal{P}} \mathbb{E}\left[d_b\left(D^{\bullet}_{\tilde{f}_{\mathrm{PL}}, \mathcal{I}_n},\ D_{\tilde{f}}\right)\right] \leq C\ \left(\frac{\log n}{n}\right)^{1/b},$$

where C depends only on a, b, c.

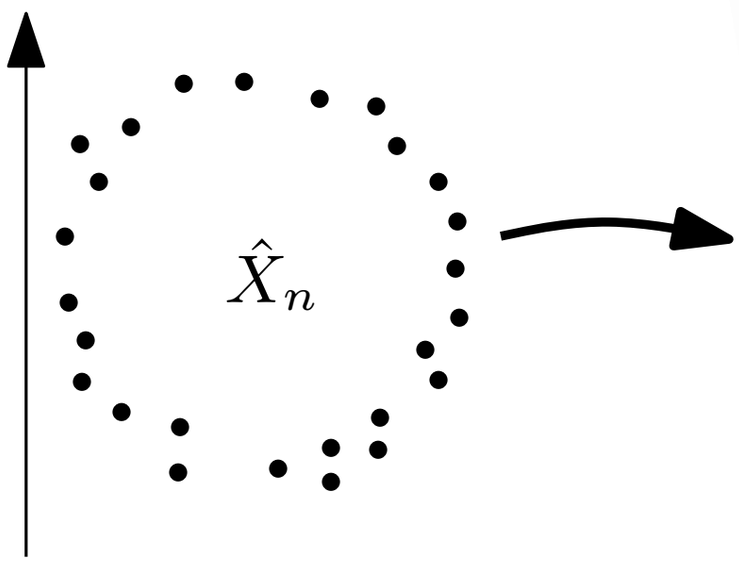
More generally: $r_n = \omega(\delta_n)/g_n$

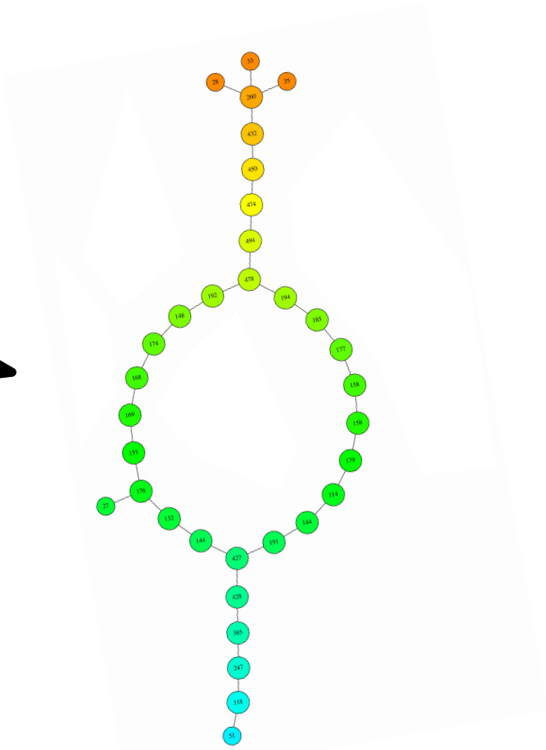
Statistics for Mapper



n points sampled
i.i.d. according to μ.

$+$ cover $\mathcal{I}$





Moreover, the estimator $D^{\bullet}_{\tilde{f}_{\mathrm{PL}},\mathcal{I}_n}$ is **minimax optimal** (up to a $\log n$ factor) on the space $\mathcal{P}$ of (a,b)-standard probability measures on X.

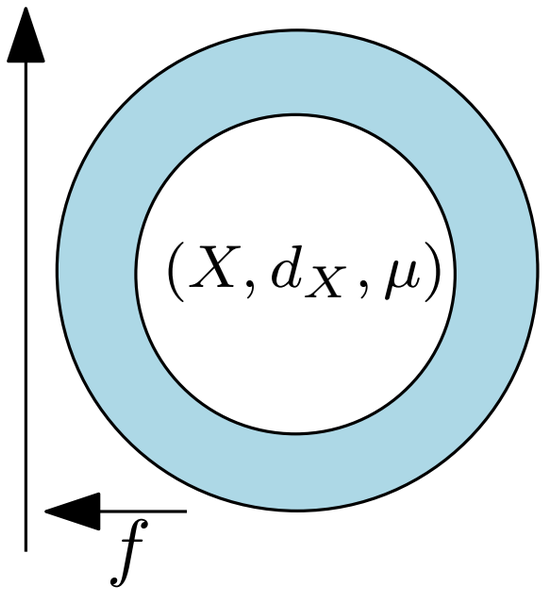
Thm: For any estimator $\widehat{\mathrm{R}}$, one has:

$$\sup_{\mu\in\mathcal{P}} \mathbb{E}\left[d_b\left(D_{\widehat{\mathrm{R}}},\ D_{\tilde{f}}\right)\right] \geq C\ \left(\frac{1}{n}\right)^{1/b},$$

where C depends only on a, b.

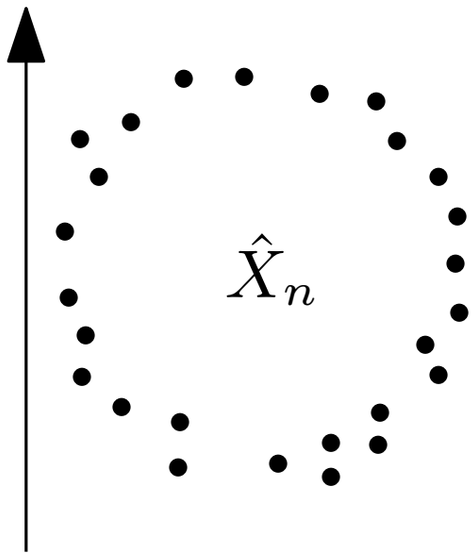
Proof: Consequence of Le Cam's lemma.

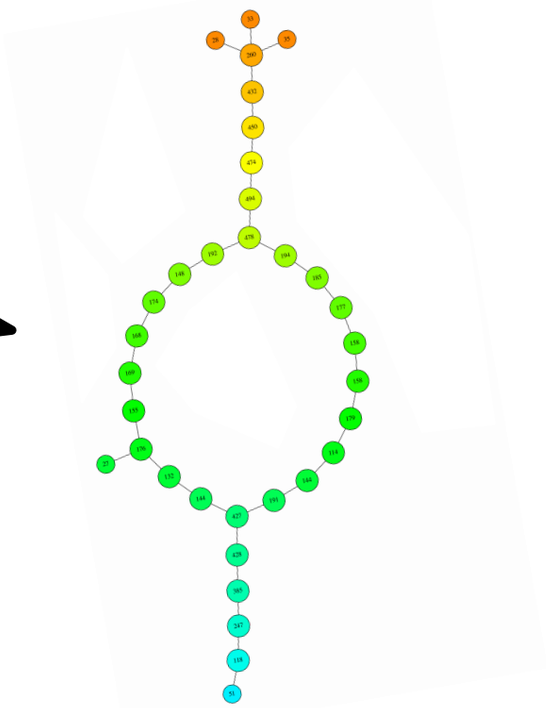
Statistics for Mapper



n points sampled i.i.d. according to μ.

$+$ cover $\mathcal{I}$





Thm: If μ is (a, b)-standard and f is c-Lipschitz then for:

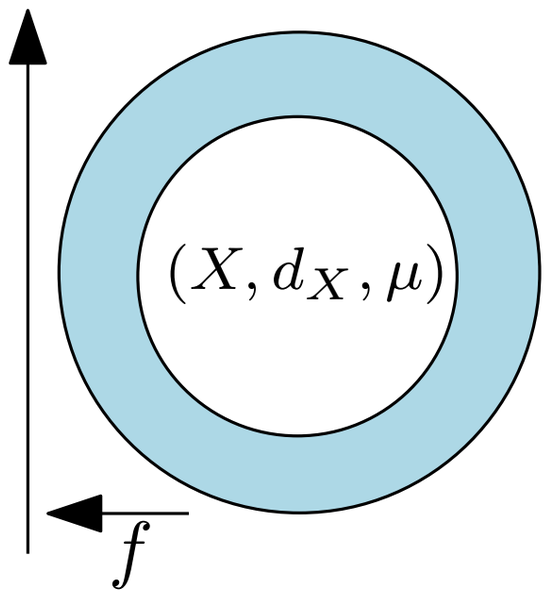
$$\delta_n = 4\left(\frac{2\log n}{an}\right)^{1/b},\ g_n \in \left(\tfrac{1}{3}, \tfrac{1}{2}\right),\ r_n = \frac{c\delta_n}{g_n}, \quad \text{one has } \forall \varepsilon > 0$$

$$\sup_{\mu \in \mathcal{P}} \mathbb{E}\left[d_b\left(D^{\bullet}_{\tilde{f}_{\mathrm{PL}}, \mathcal{I}_n},\ D_{\tilde{f}}\right)\right] \leq C\ \left(\frac{\log n}{n}\right)^{1/b},$$

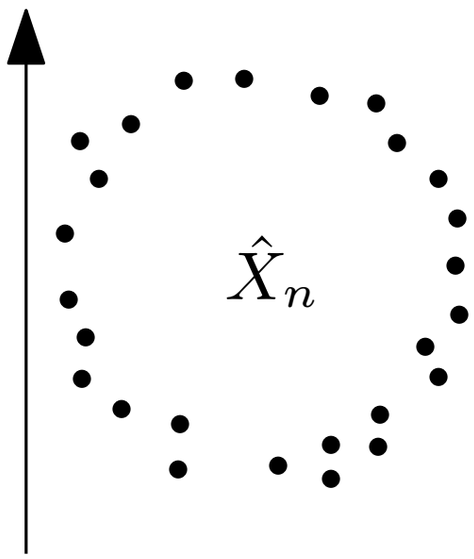
where C depends only on a, b, c.

More generally: $r_n = \omega(\delta_n)/g_n$

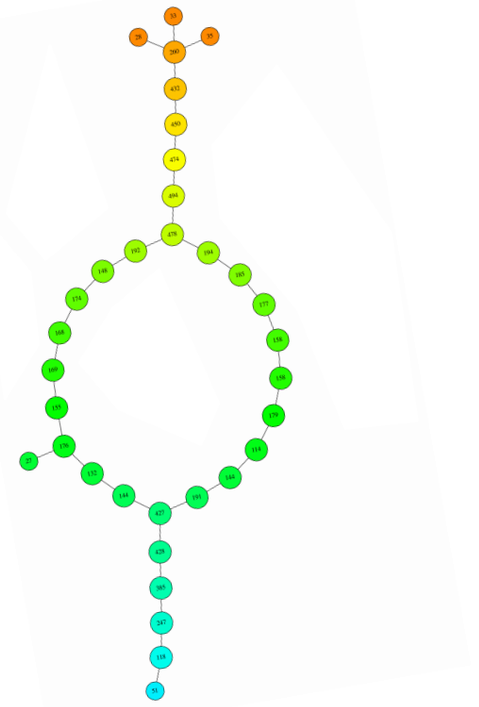
Statistics for Mapper



n points sampled
i.i.d. according to μ

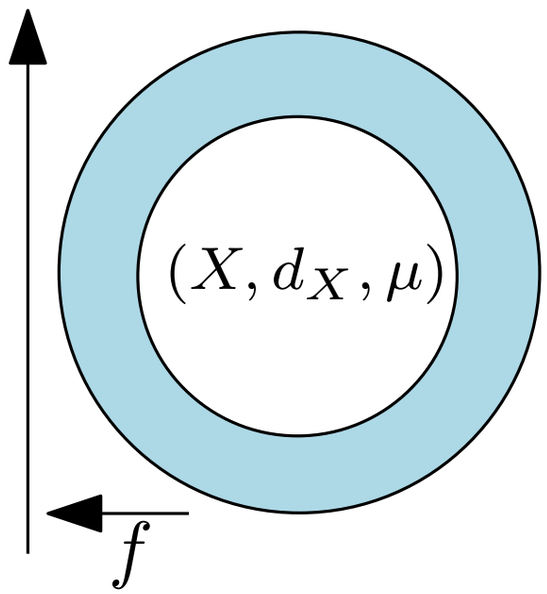


δ_n
$\mathcal{I}_n(g_n, r_n)$

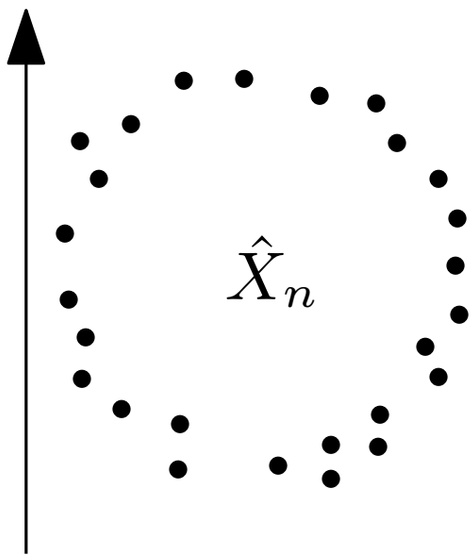


Fortunately, one can use subsampling to tune δ_n: let $\beta > 0$, $s(n) = \frac{n}{\log(n)^{1+\beta}}$ and define $\delta_n := d_H(\hat{X}_n^{s(n)}, \hat{X}_n)$, where $\hat{X}_n^{s(n)}$ is a subset of $\hat{X}_n$ of size $s(n)$.

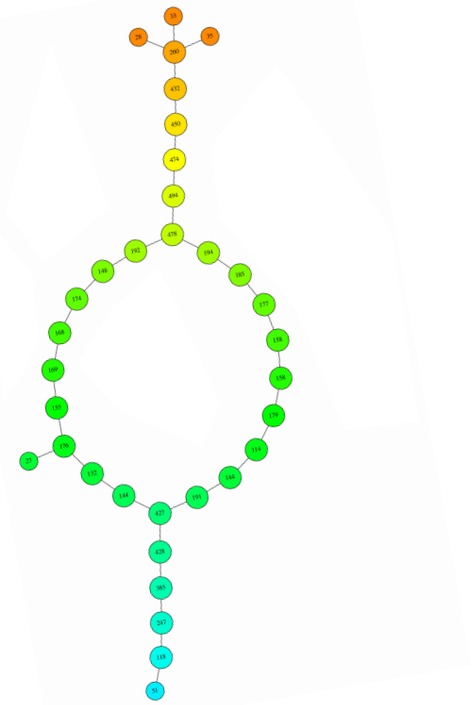
Statistics for Mapper



n points sampled
i.i.d. according to μ



δ_n
$\mathcal{I}_n(g_n, r_n)$



Fortunately, one can use subsampling to tune δ_n: let $\beta > 0$, $s(n) = \frac{n}{\log(n)^{1+\beta}}$ and define $\delta_n := d_H(\hat{X}_n^{s(n)}, \hat{X}_n)$, where $\hat{X}_n^{s(n)}$ is a subset of $\hat{X}_n$ of size $s(n)$.

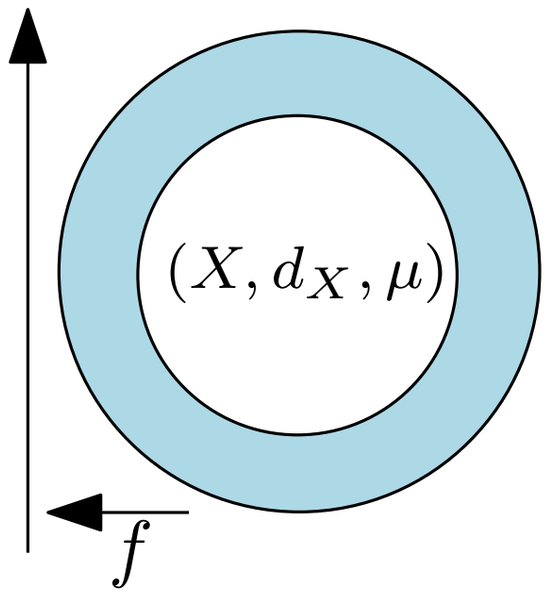
Thm: If μ is (a, b)-standard and f is c-Lipschitz, then for:

$$\delta_n = d_H(\hat{X}_n^{s(n)}, \hat{X}_n),\ g_n \in \left(\tfrac{1}{3}, \tfrac{1}{2}\right),\ r_n = \frac{c\delta_n}{g_n}, \quad \text{one has } \forall \varepsilon > 0$$

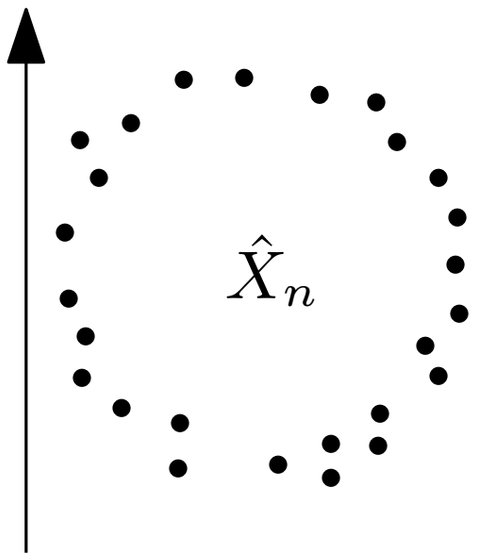
$$\sup_{\mu \in \mathcal{P}} \mathbb{E}\left[d_b\left(D^{\bullet}_{\tilde{f}_{\mathrm{PL}}, \mathcal{I}_n}, D_{\tilde{f}}\right)\right] \leq C\left(\frac{\log(n)^{2+\beta}}{n}\right)^{1/b},$$

where C depends only on a, b, c.

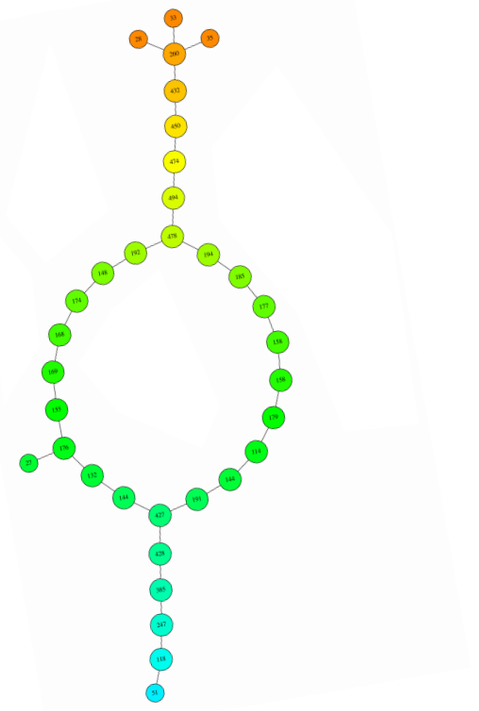
Statistics for Mapper



n points sampled
i.i.d. according to μ



δ_n
$\mathcal{I}_n(g_n, r_n)$



Ex : PCA filter

Π_1 : orthonormal projection onto first principal direction of covariance operator.

$\widehat{\Pi}_1$: orthonormal projection onto first principal direction of empirical covariance operator.

$$\mathbb{E}\left[d_b\left(D^{\bullet}_{\widehat{\Pi}_{1,\mathrm{PL}},\mathcal{I}_n}, D_{\tilde{\Pi}_1}\right)\right] \lesssim \left(\frac{\log(n)^{2+\beta}}{n}\right)^{1/b} \vee \frac{1}{\sqrt{n}}$$

[*PCA-Kernel Estimation*, Biau, Mas, *Statistics & Risk Modeling with Applications in Finance and Insurance*, 2012]

Get confidence region with $\mathbb{E}\left[d(\cdot,\cdot)\right] = \int_\alpha \mathbb{P}(d(\cdot,\cdot) \geq \alpha)\mathrm{d}\alpha$.

Multivariate case: filter-based pseudometric

[*Topological Analysis of Nerves, Reeb Spaces, Mappers, and Multiscale Mappers*, Dey, Mémoli, Wang, SoCG, 2017]

Def: The *filter-based pseudometric* $d_f : M \times M \to \mathbb{R}$ is defined as

$$d_f(x, x') = \inf_{\gamma \in \Gamma(x,x')} \operatorname{diam}_Y(f \circ \gamma),$$

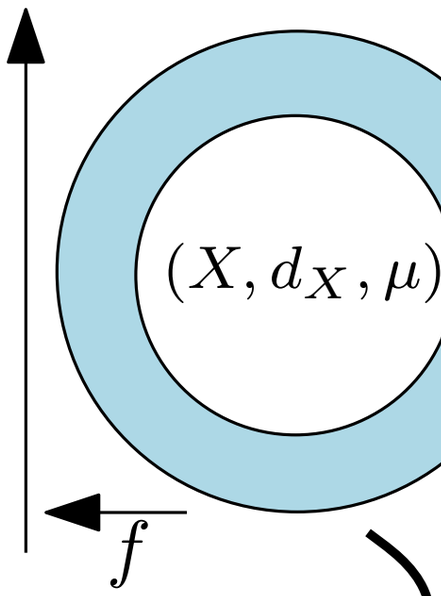
where $\Gamma(x, x')$ denotes the set of all continuous paths $\gamma : [0, 1] \to M$ such that $\gamma(0) = x$ and $\gamma(1) = x'$, and diam_Y denotes the *diameter* of a subset of Y.

Def: The *Gromov-Hausdorff* metric d_{GH} between $(M, d_f), (M', d_{f'})$ is defined as

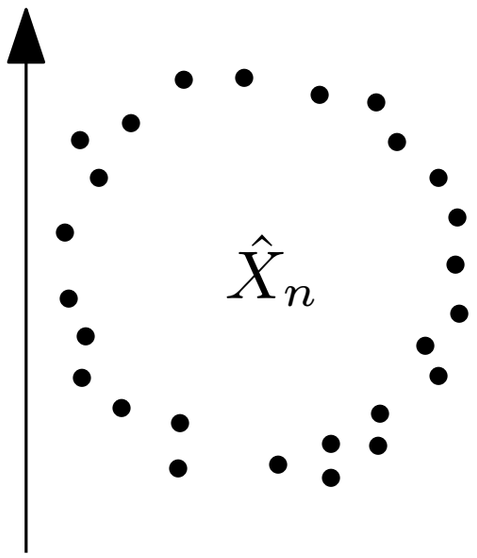
$$d_{GH}(M, M') = \frac{1}{2} \inf_C \sup_{(x,x'),(y,y') \in C} |d_f(x, y) - d_{f'}(x', y')|,$$

where C denotes the set of all correspondences between M and M' (subsets of $M \times M'$ s.t. projections onto M and M' are surjective).

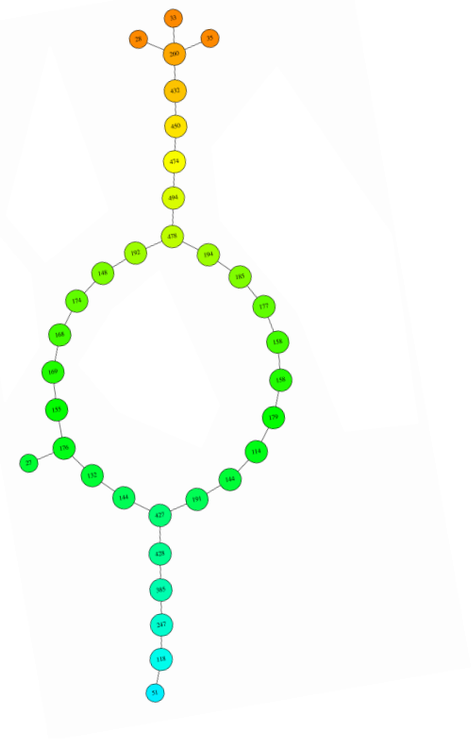
Statistics for multivariate Mapper

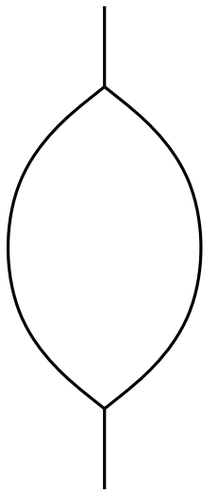


n points sampled
i.i.d. according to μ



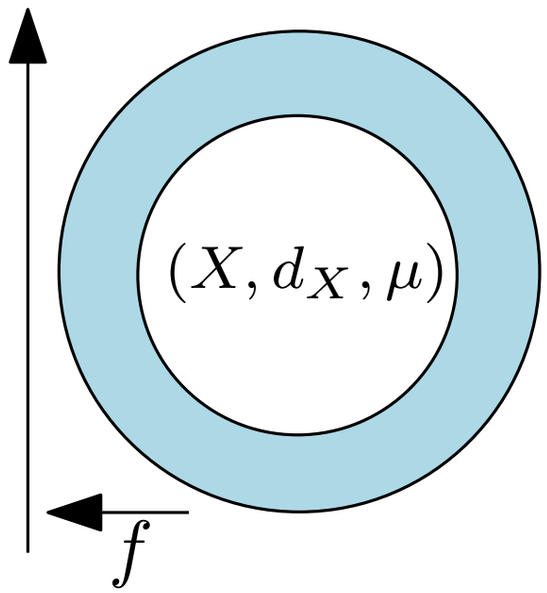
δ_n
$\mathcal{I}$



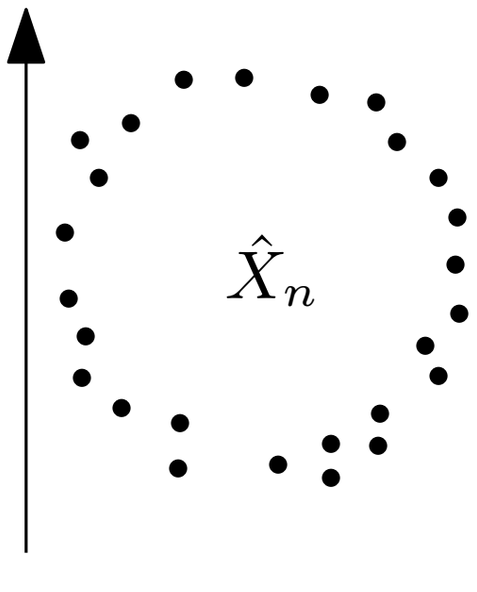


$$\mathbb{E}\left[d_{GH}(\mathrm{M}_f, \mathrm{R}_f) \leq ?\right] \geq 0.95$$

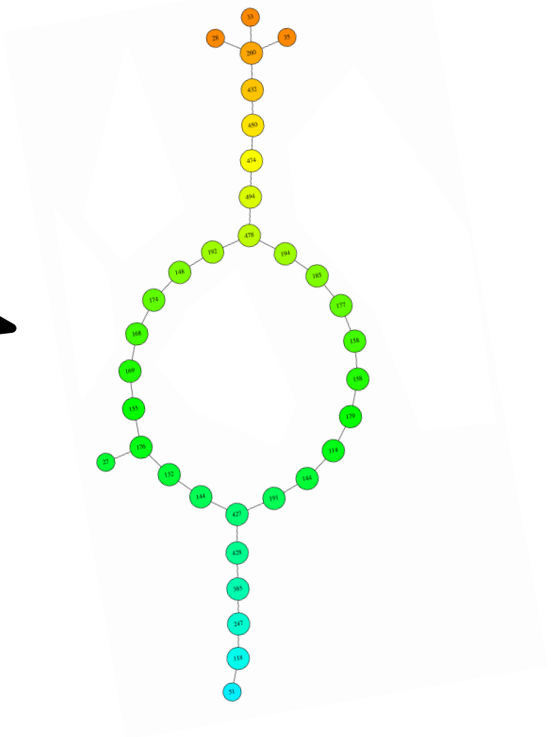
Statistics for multivariate Mapper



n points sampled
i.i.d. according to μ



δ_n
$\mathcal{I}$



Thm: If μ and $f\#\mu$ are (a, b)-standard, then for δ_n as before, one has:

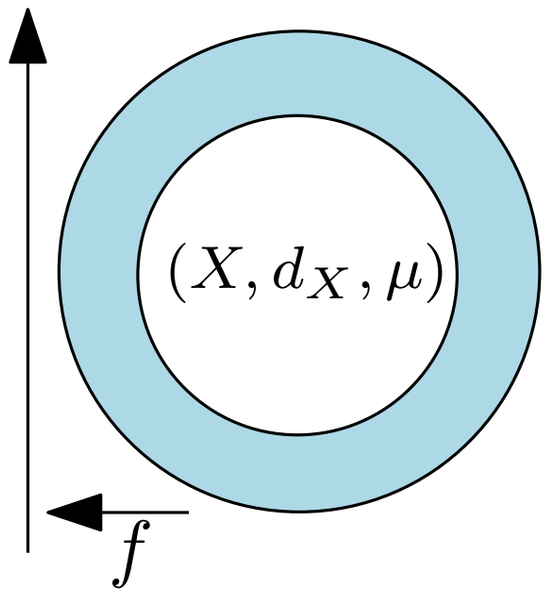
$$\mathbb{E}\left[d_{GH}\left(\mathrm{M}^{\bullet}_{f,\delta_n}(\hat{X}_n, \mathcal{I}), \mathrm{R}_f(X)\right)\right] \leq 5 \cdot \mathbb{E}\left[\mathrm{res}(\mathcal{I})\right] + C\omega\left(\frac{\log(n)^{i}}{n}\right.$$

where C depends only on a, b, and res denotes the *resolution* of the cover $\mathcal{I}$, i.e., the diameter of its elements.

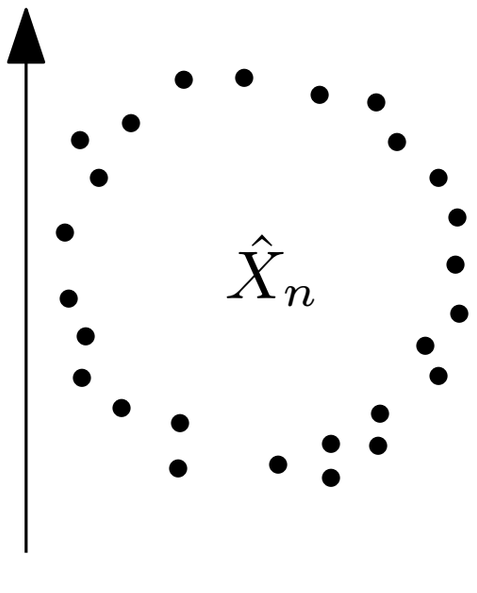
Moreover, using covers with hypercubes or K-means, or quantized Distance-to-Measure allows to bound $\mathbb{E}\left[\mathrm{res}(\mathcal{I})\right]$.

[*A k-points-based distance for robust geometric inference*, Brecheteau, Levrard, *Bernouilli*, 2020]

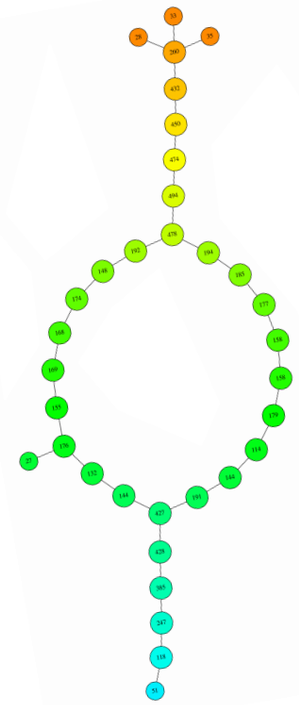
Statistics for multivariate Mapper



n points sampled
i.i.d. according to μ



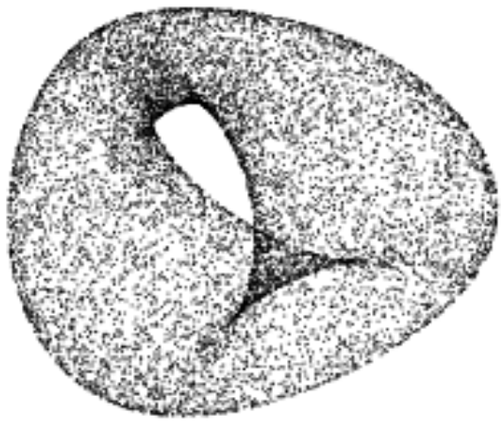
δ_n
$\mathcal{I}$

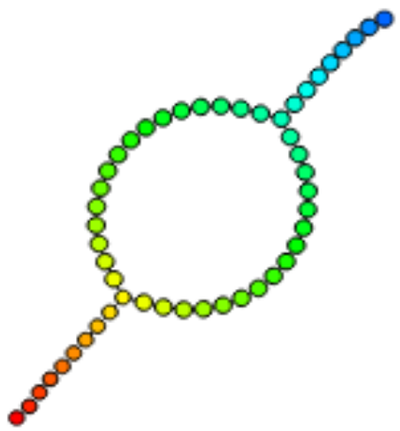


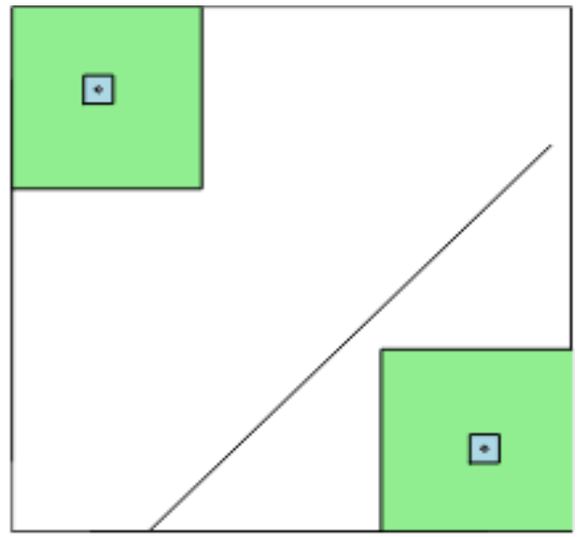
Thm: If $w(u) \leq cu^\gamma$ for some $c > 0, \gamma \in (0, 1)$, and for a cover $\mathcal{I}$ given by thickening a k-means partition in $\mathbb{R}^D$:

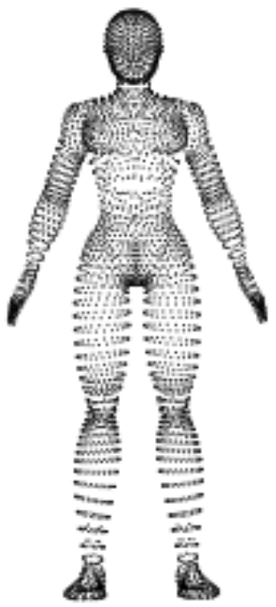
$$\mathbb{E}\left[\mathrm{res}(\mathcal{I})\right] \leq k^{-(2\gamma^2)/(2\gamma b+b^2)} + \left(\frac{kD}{n}\right)^{\gamma/(2b+4\gamma)}$$

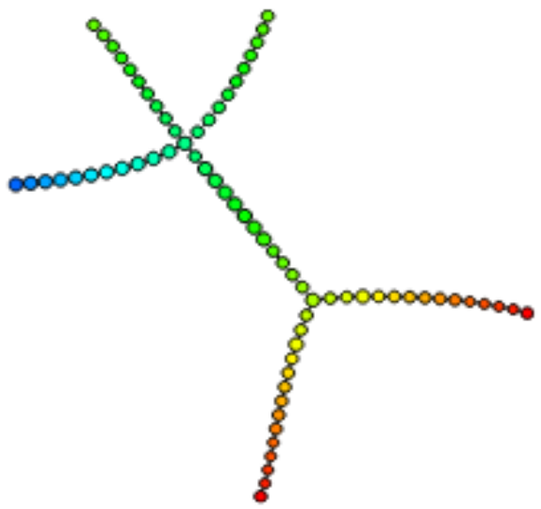
Experiments 85% confidence intervals

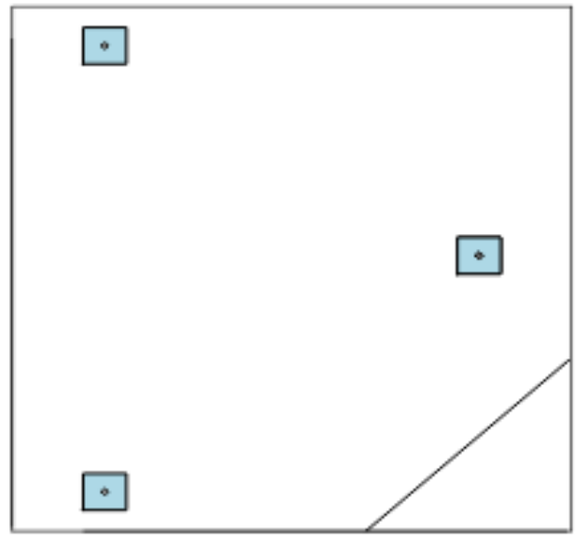


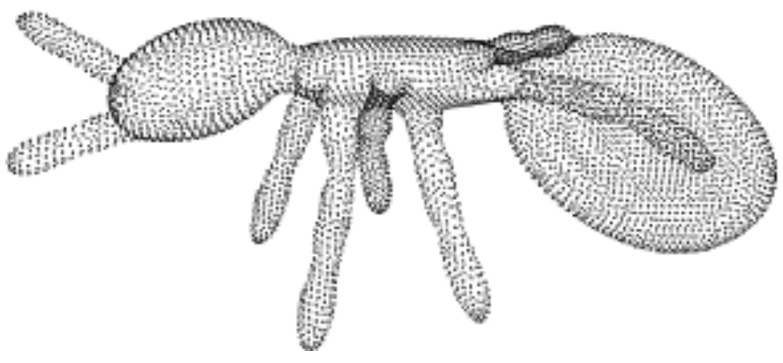


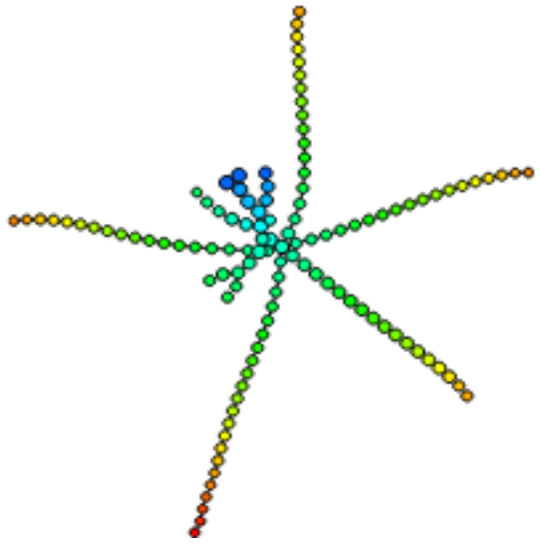


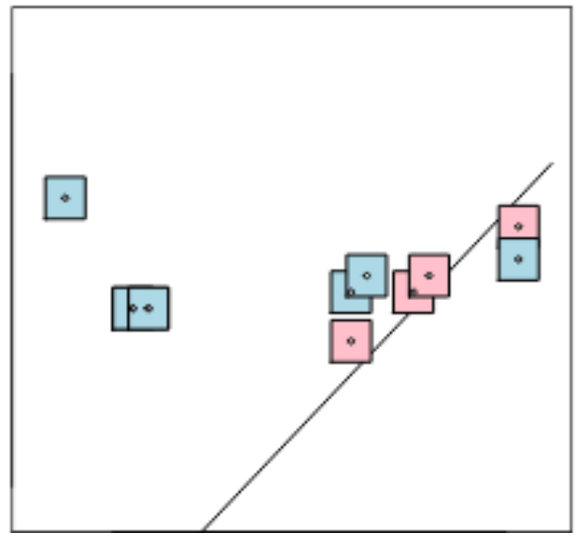




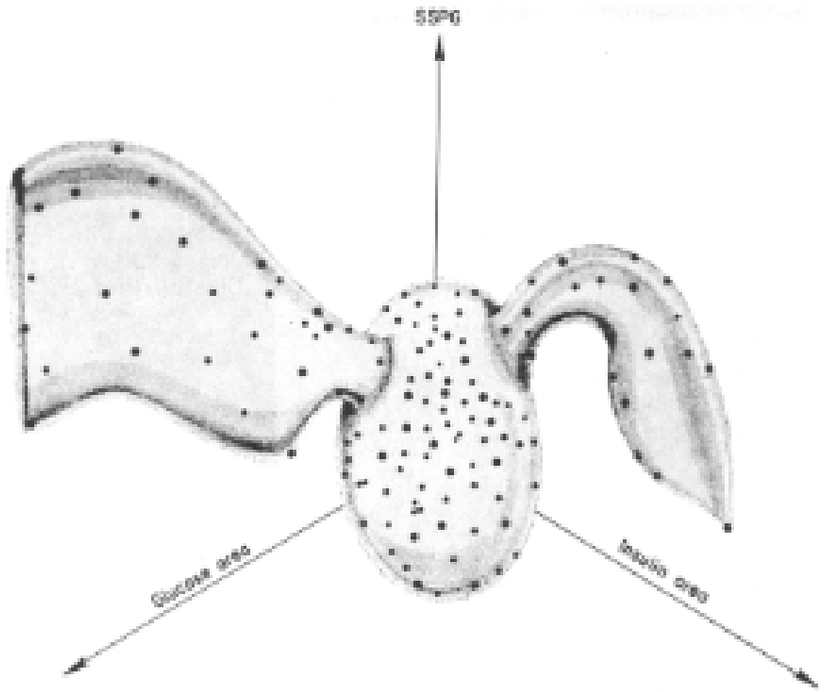


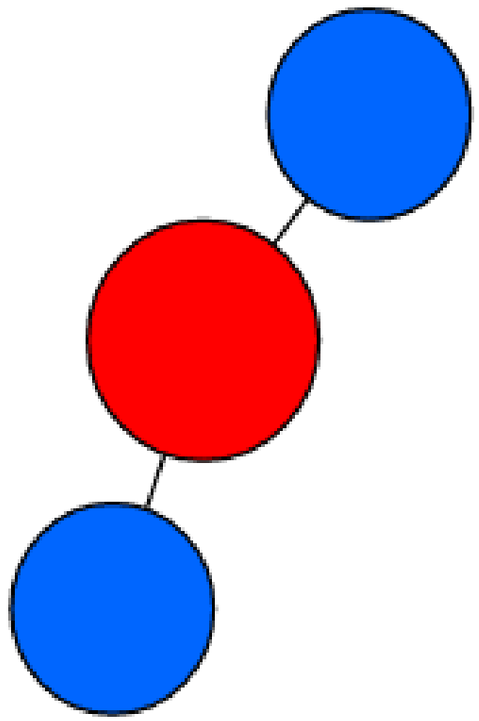


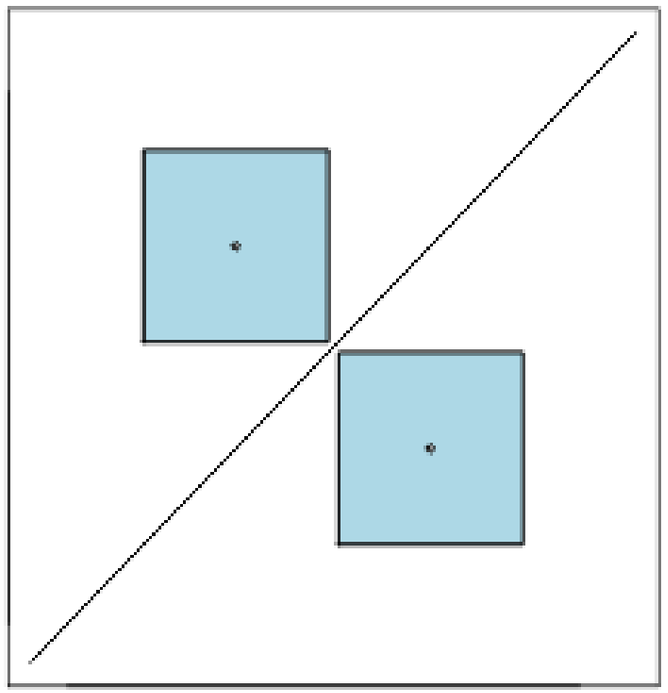


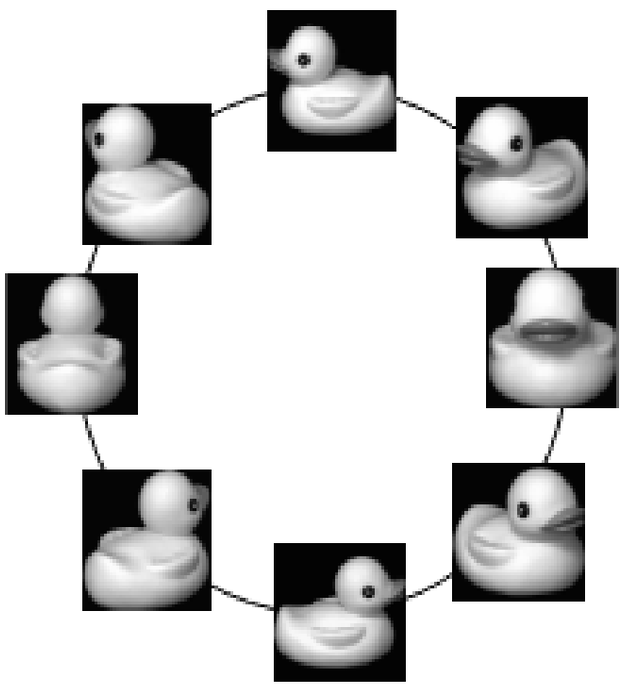


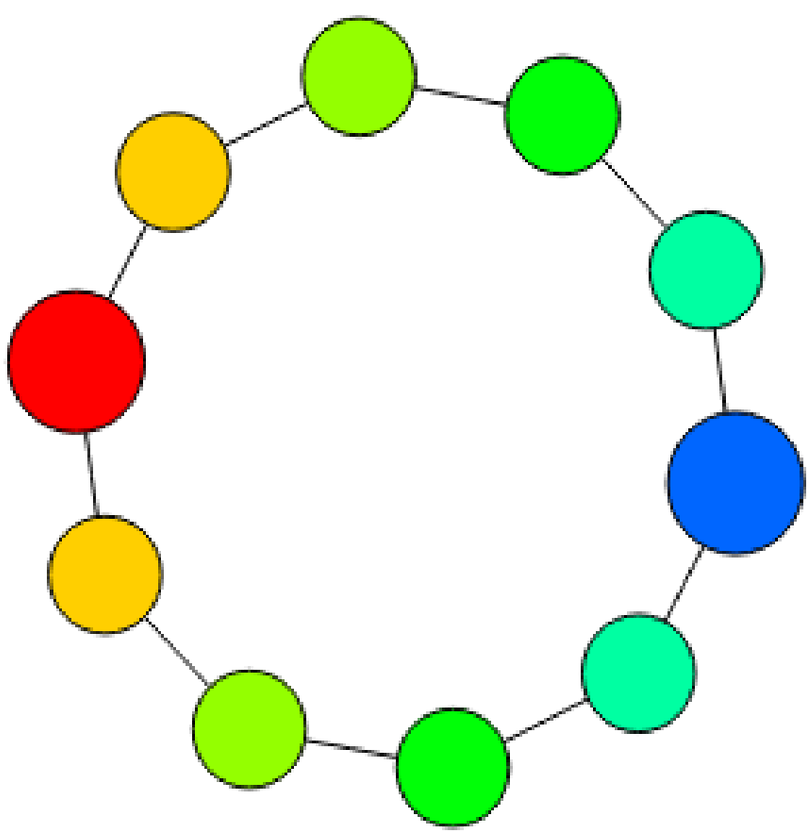
Experiments 85% confidence intervals

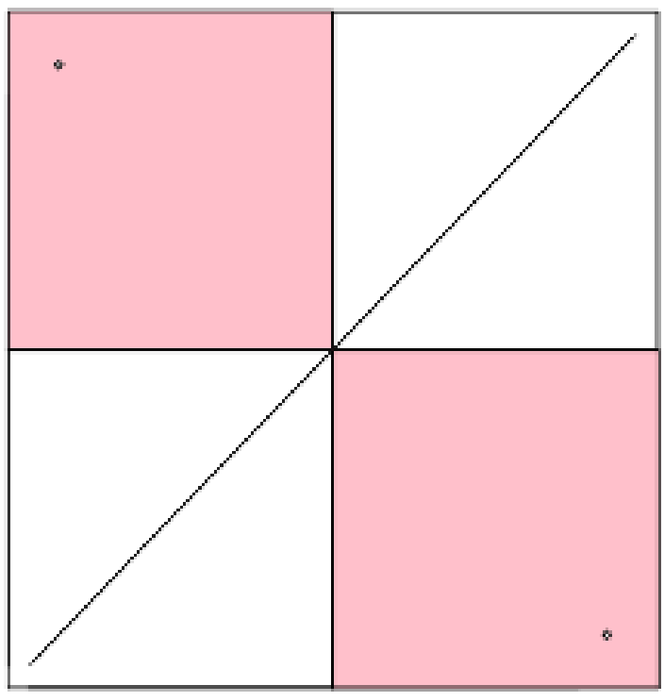




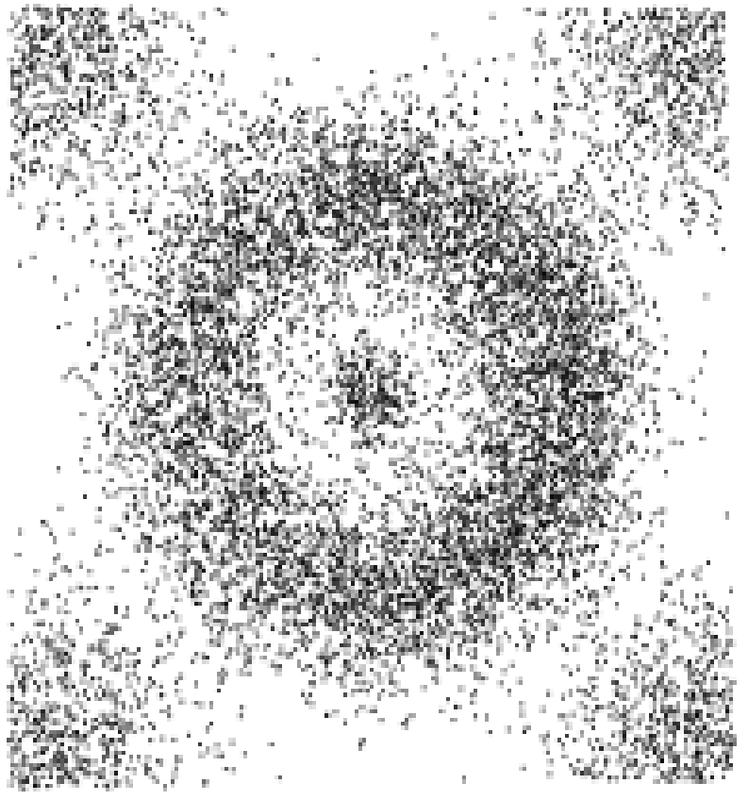


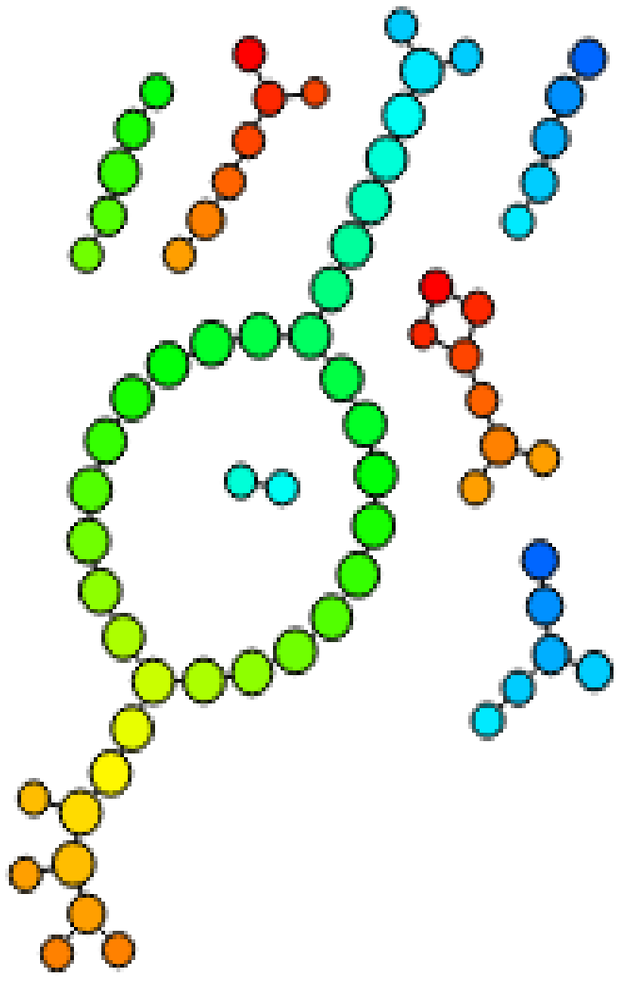


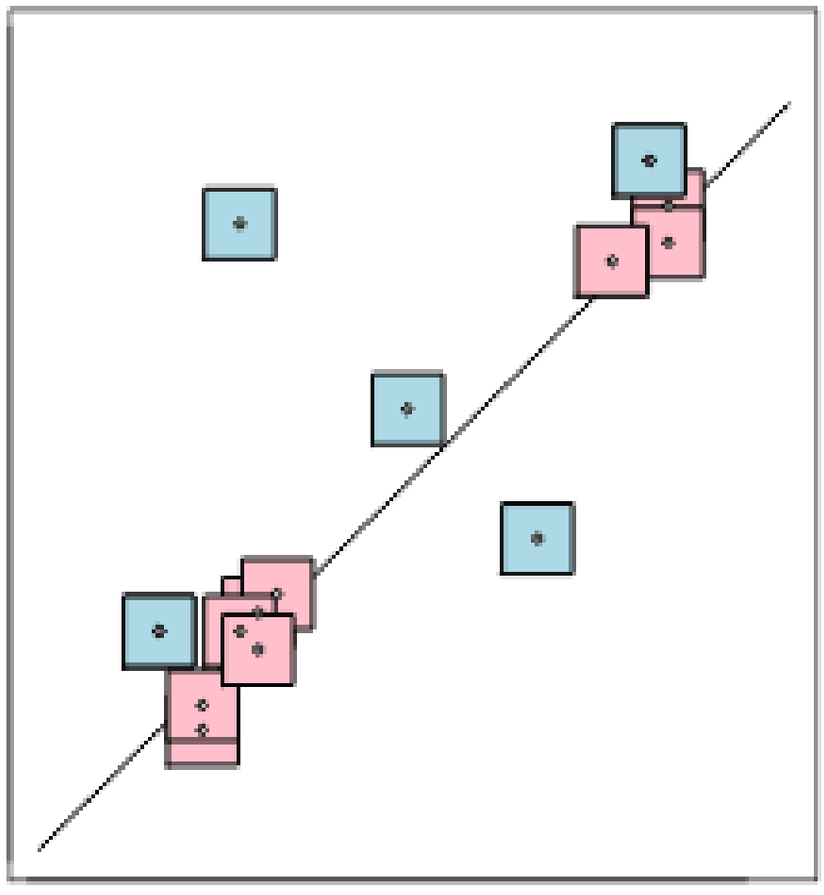




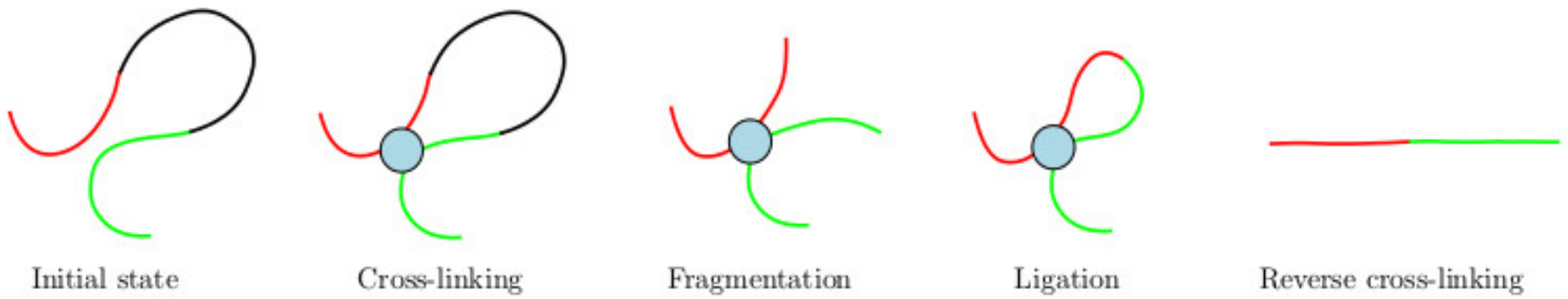
Experiments 85% confidence intervals

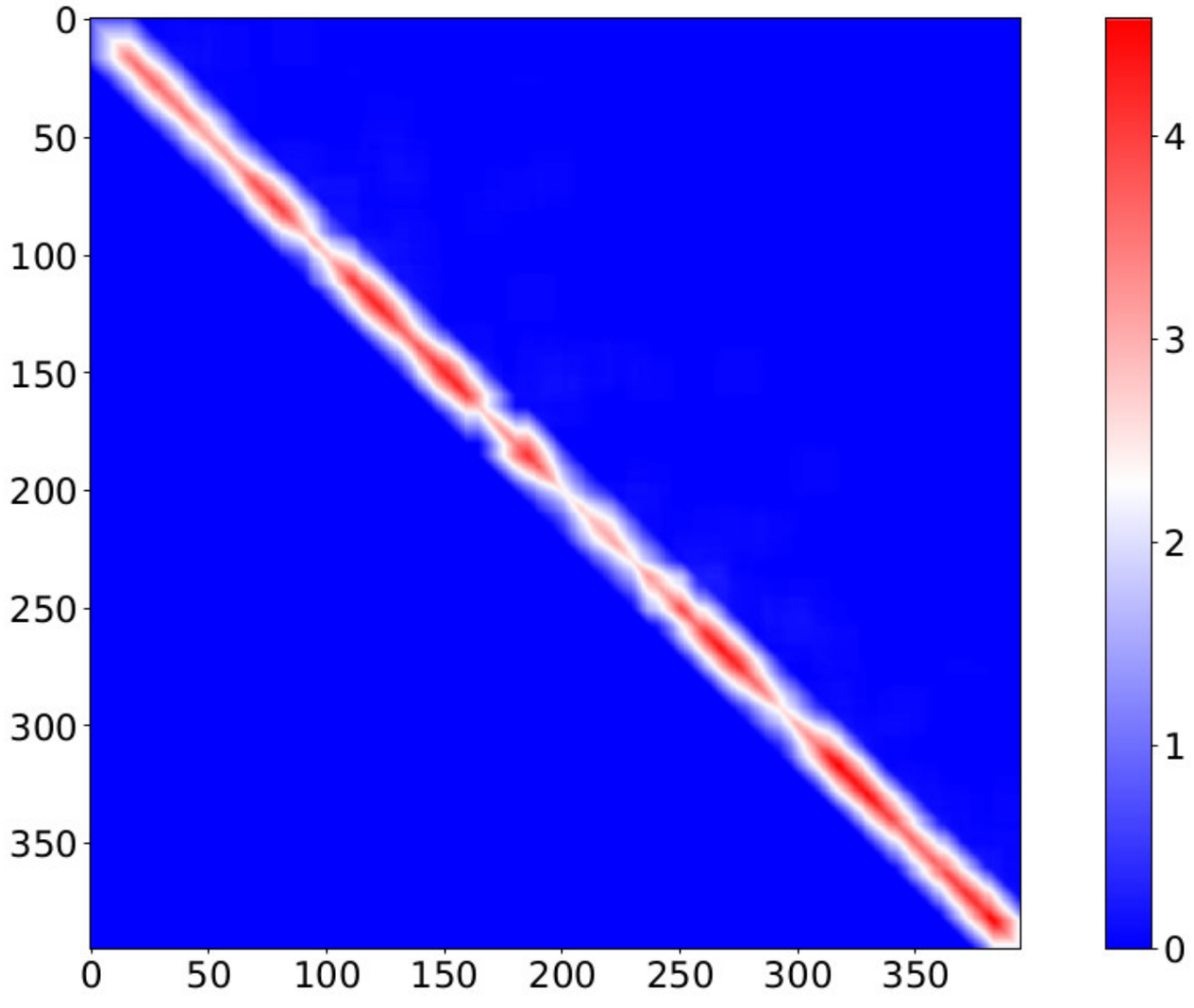




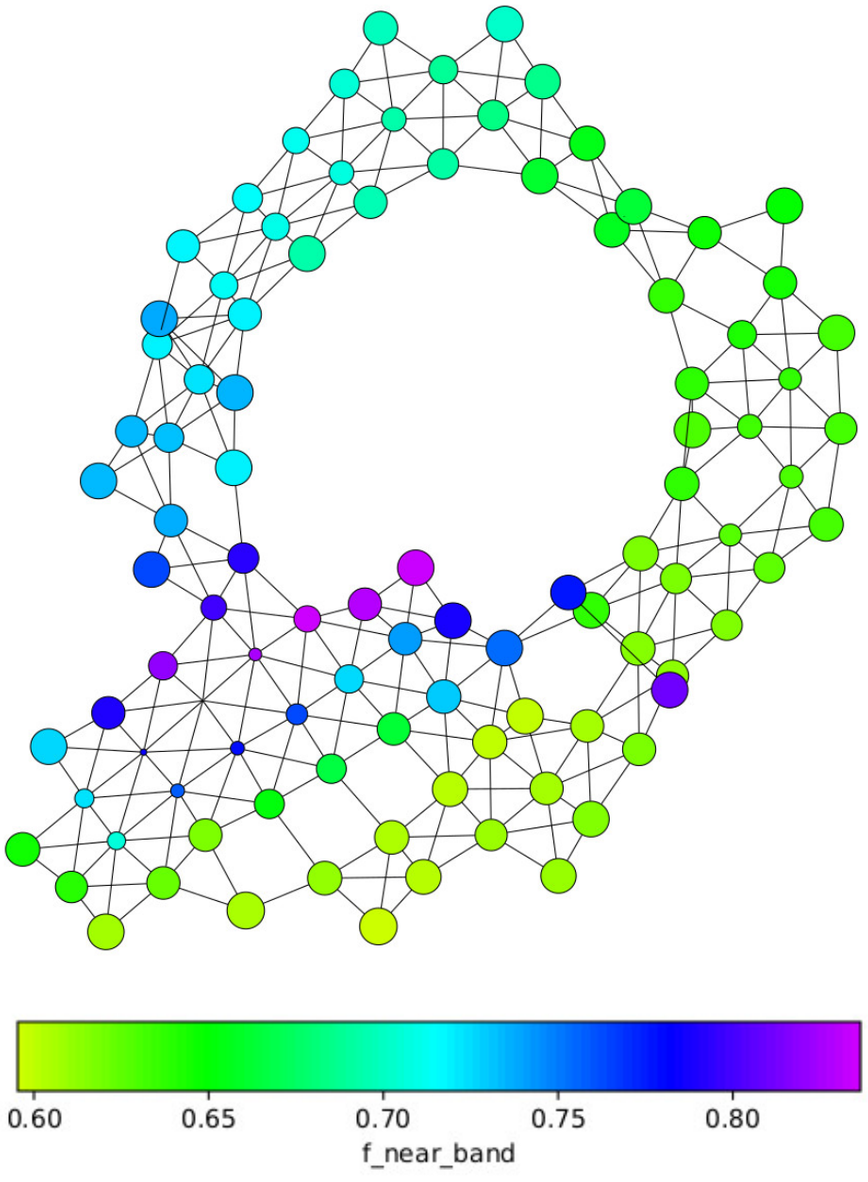


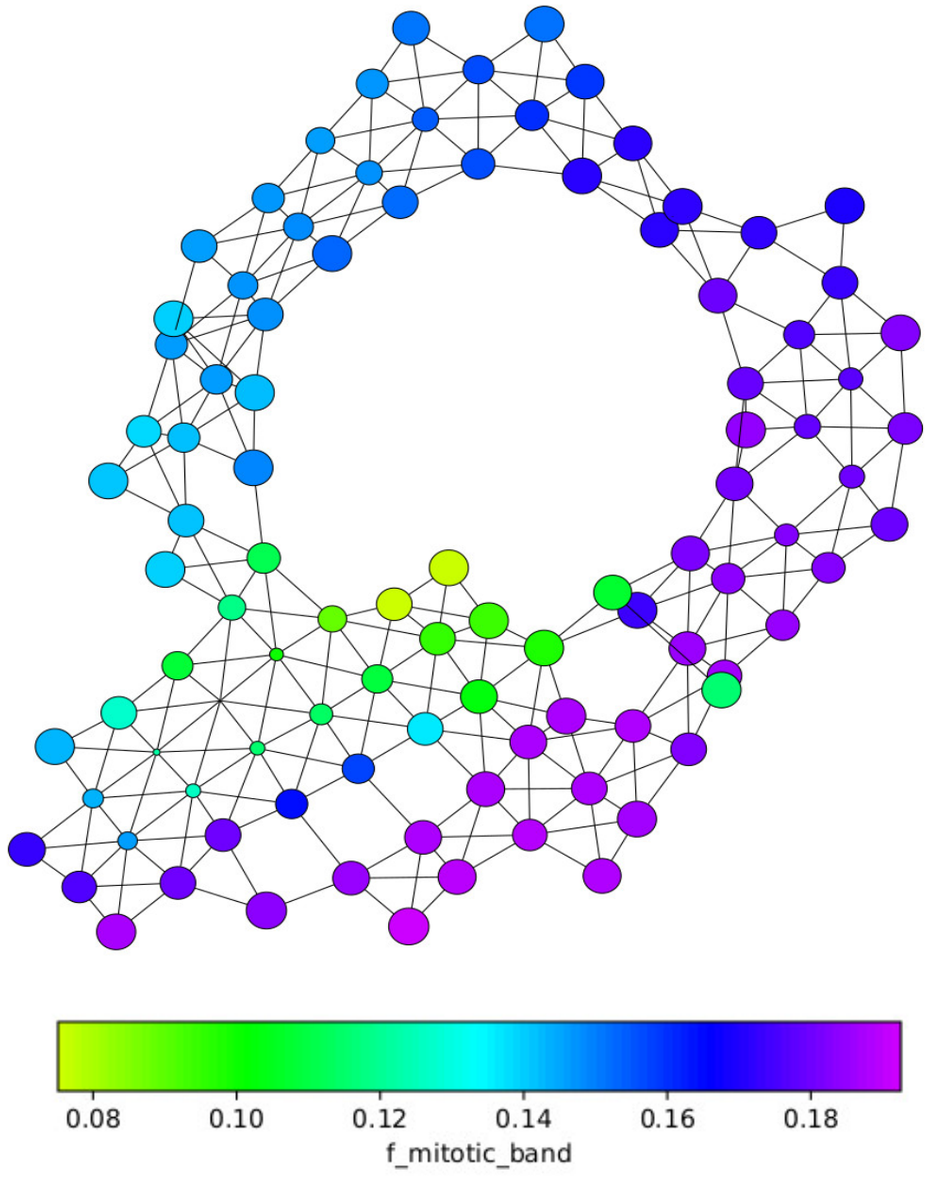
Experiments Chromosome conformation capture





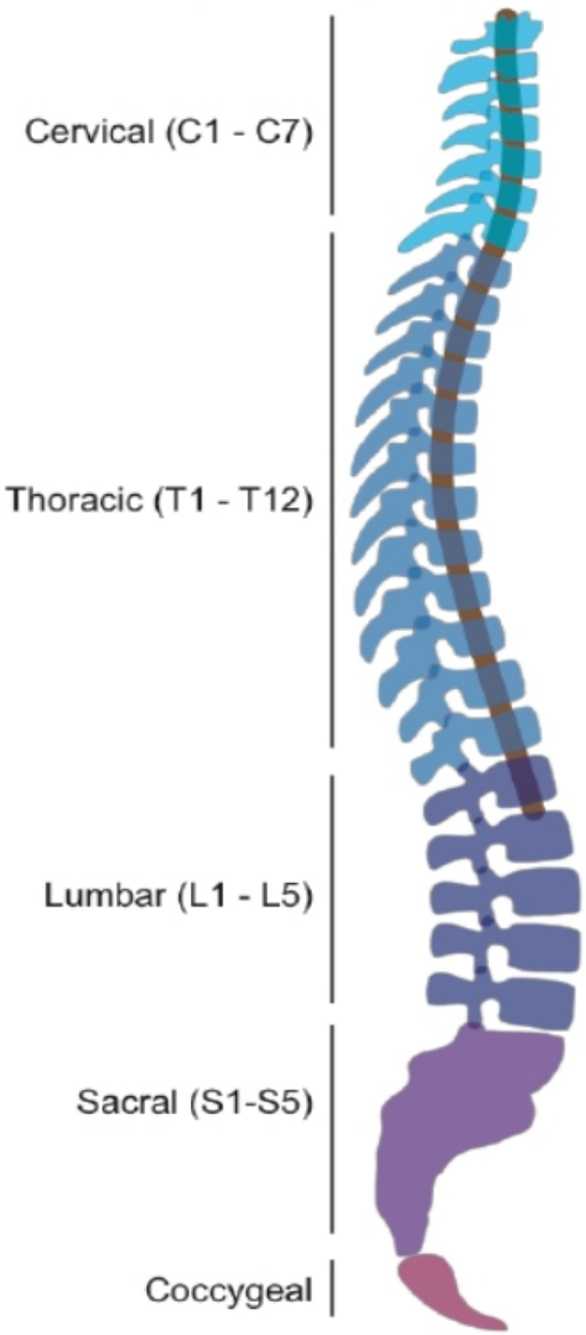
Experiments Chromosome conformation capture

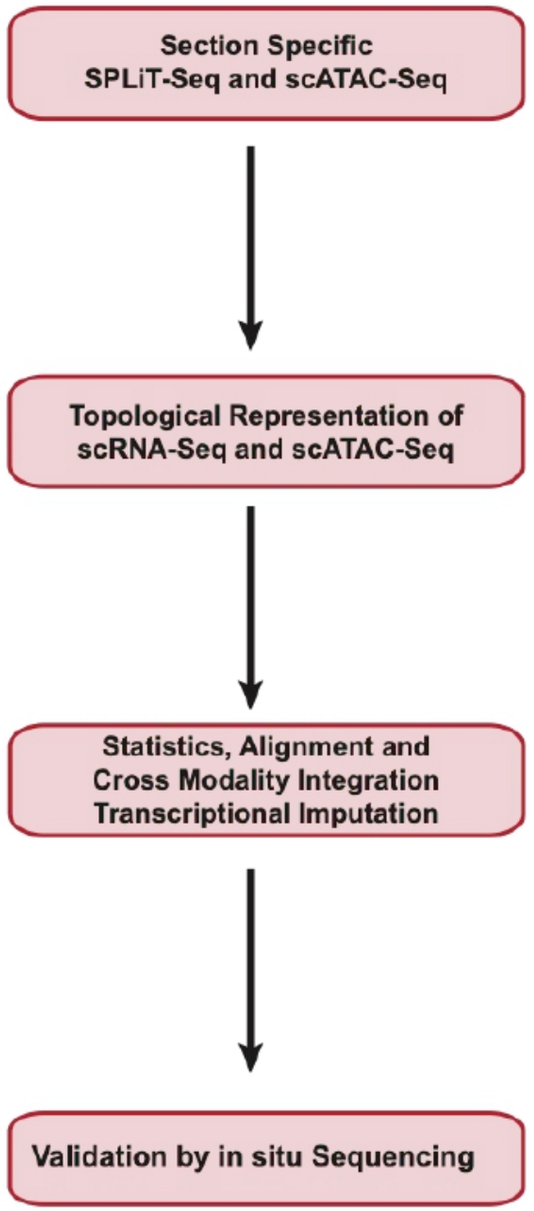


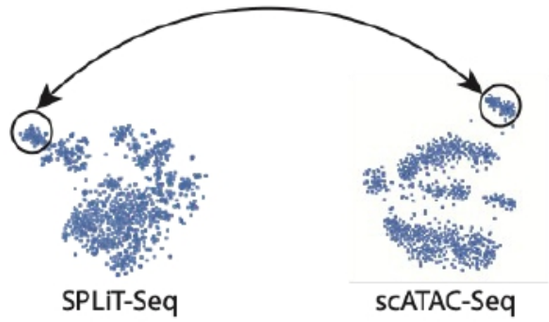


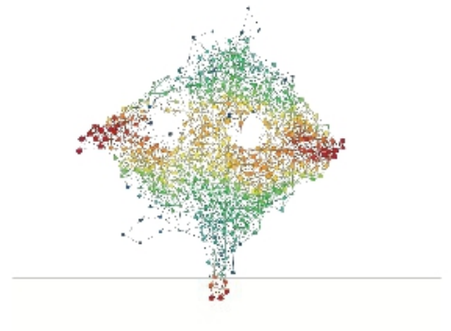
Formal identification of cell cycle with 95% confidence

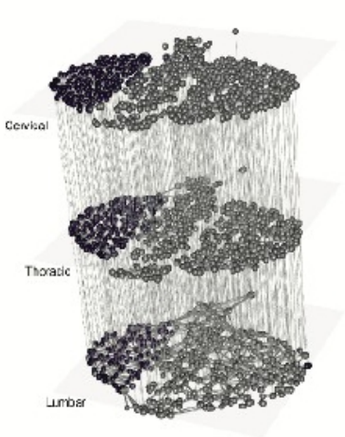
Experiments Spinal cord data

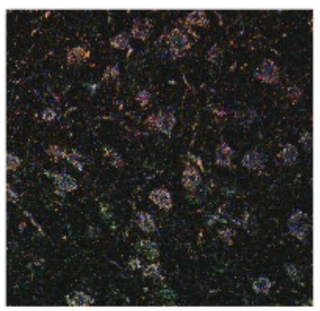




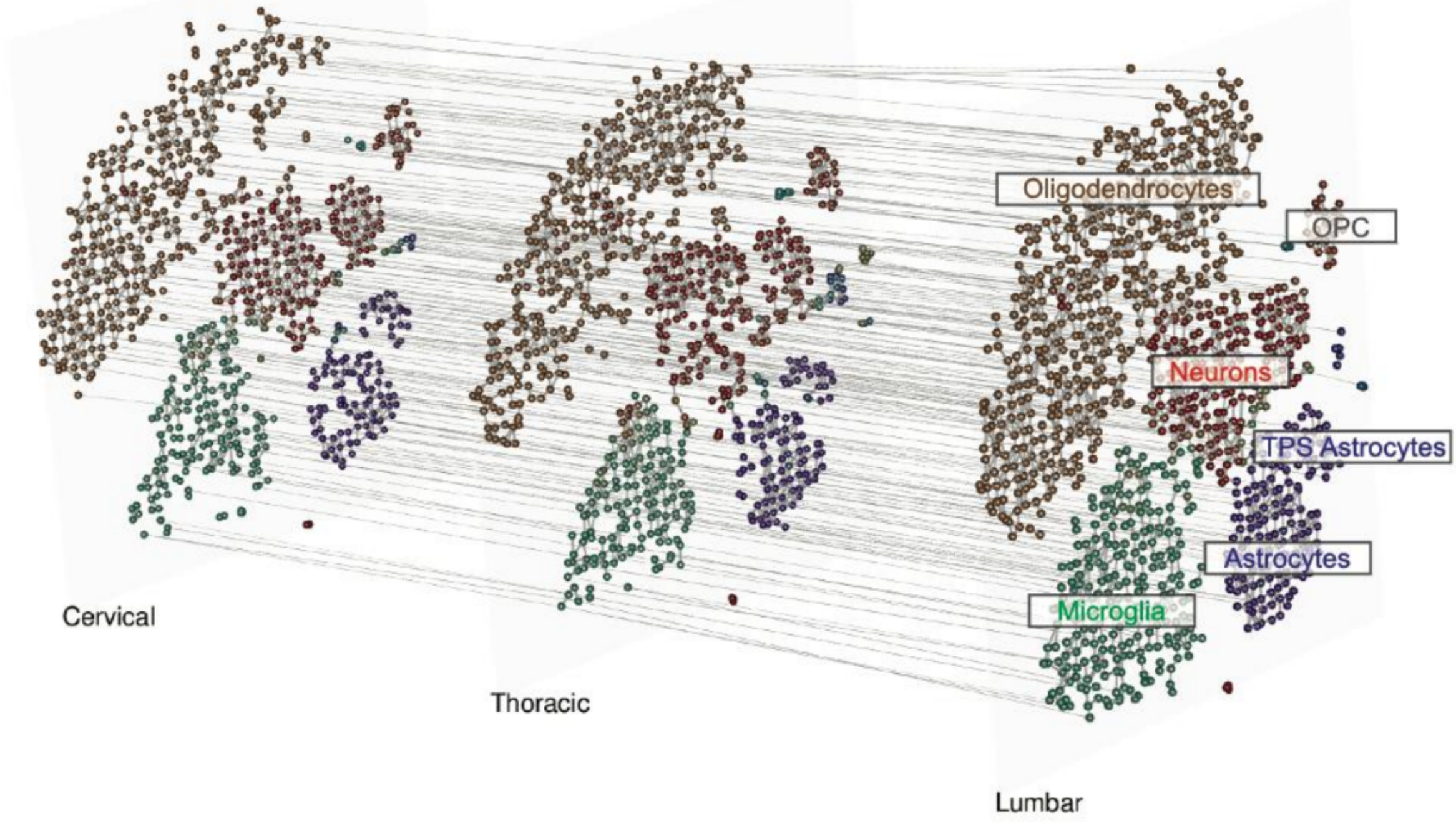




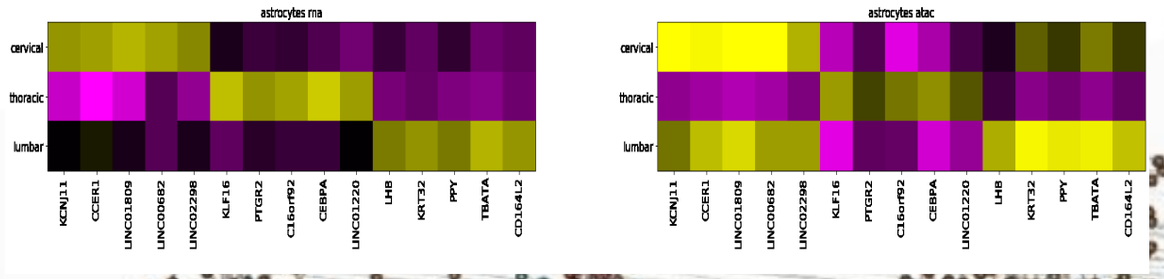


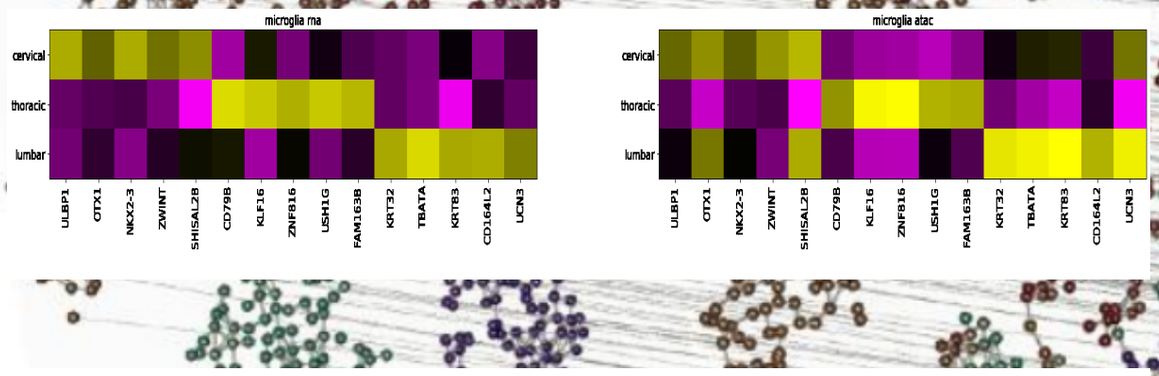


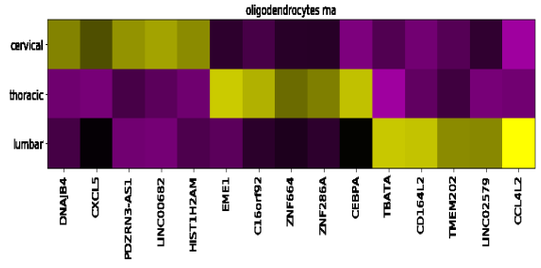
Experiments Spinal cord data

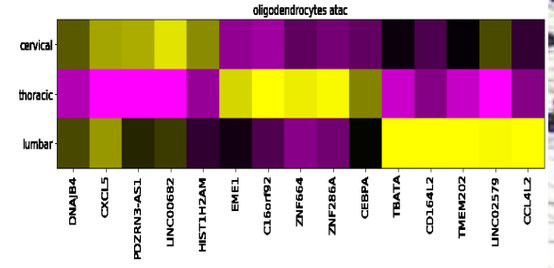


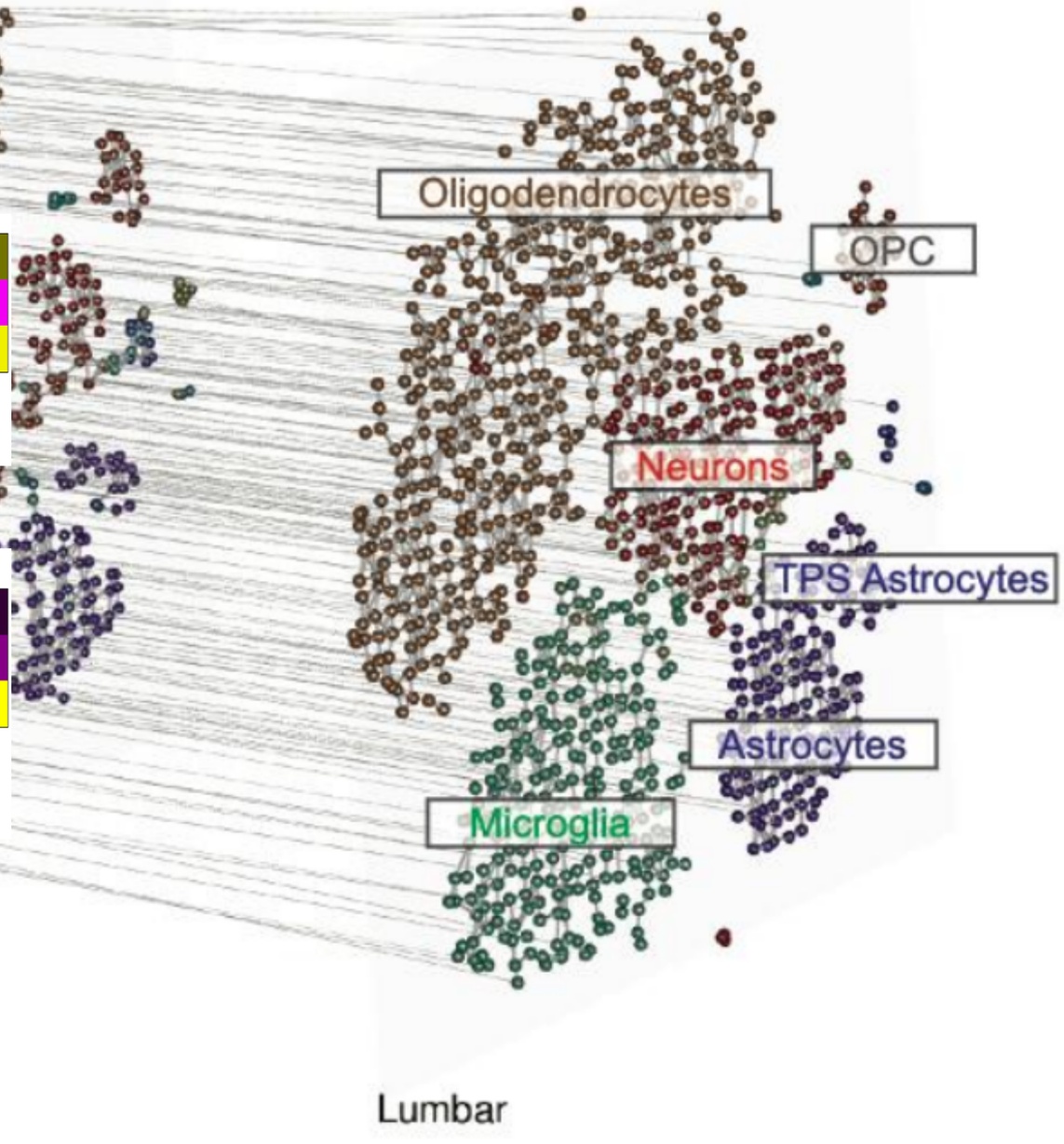
Experiments Spinal cord data







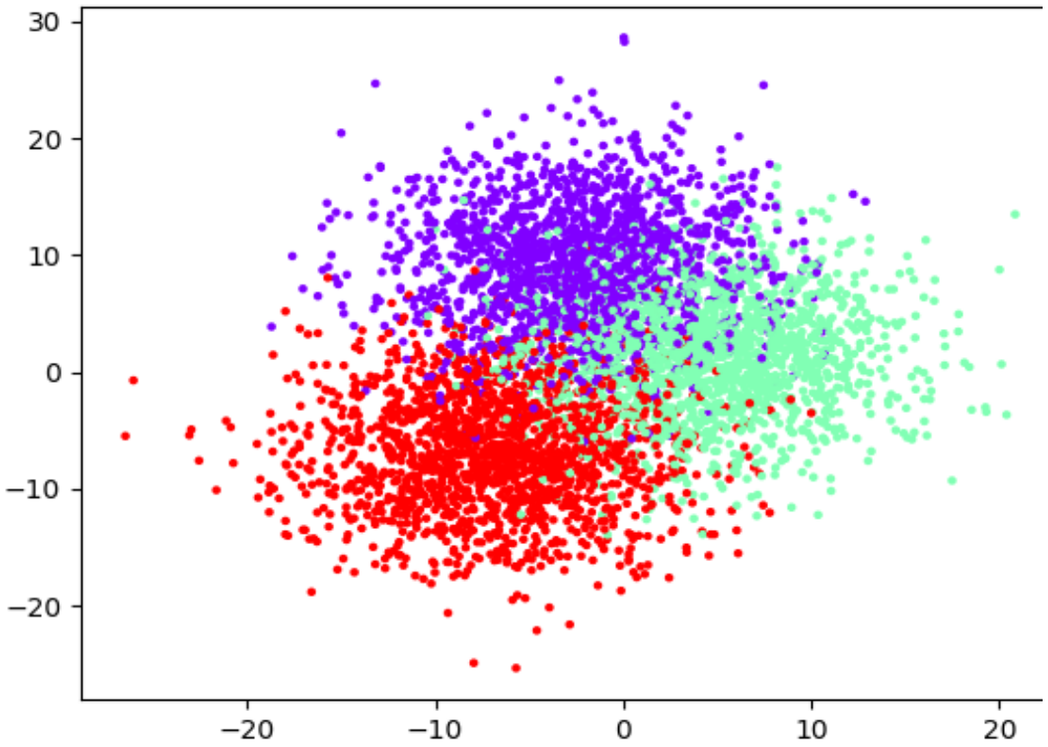


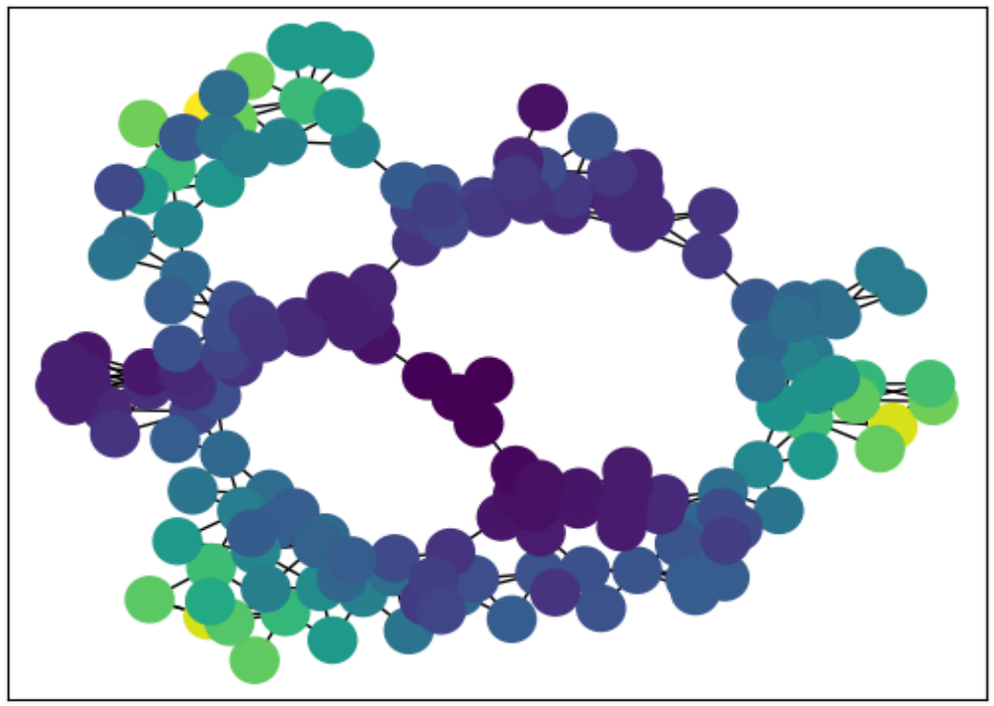


Experiments Machine learning classifier

Machine Learning Classifier Monitoring

Filter = confidence of Random Forest classifier (in $\mathbb{R}^3$)

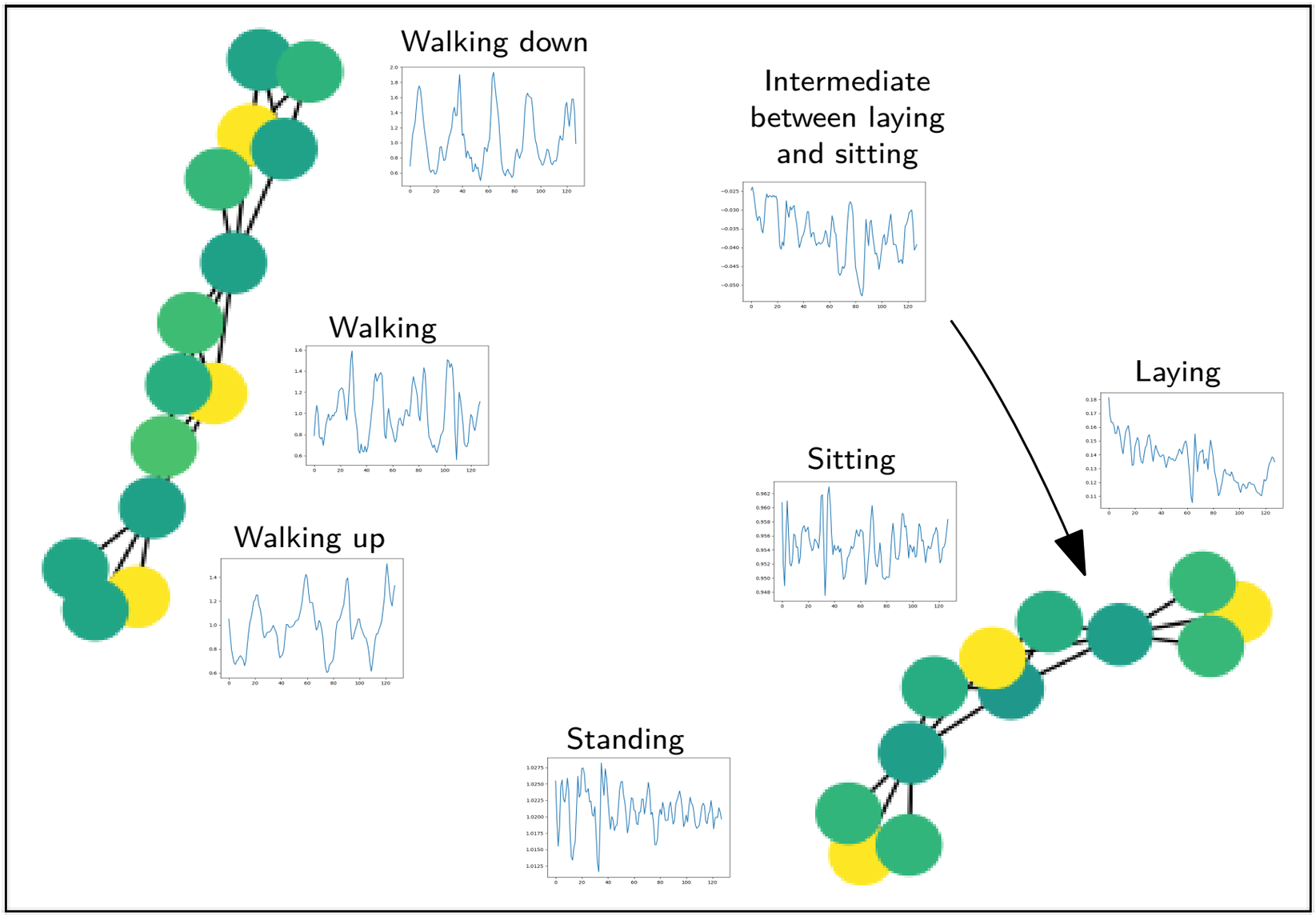




Experiments Machine learning classifier

Machine Learning Classifier Monitoring

Filter = confidence of Random Forest classifier (in $\mathbb{R}^6$)



Other works

Another line of work is about the *interleaving distance* between Mappers and Reeb spaces *seen as cosheaves* $\mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$.

[*Convergence between categorical representations of Reeb space and Mapper*, Munch, Wang, SoCG, 2016]

Prop: For $f : X \to \mathbb{R}^d$, $d_I(\mathcal{C}(\mathrm{R}_f(X)), \mathcal{C}(\mathrm{M}_f(X, \mathcal{I}))) \leq \mathrm{res}(\mathcal{I})$.

[*Probabilistic convergence and stability of random Mapper graphs*, Brown et al., JACT, 2020]

Prop: For $f : X \to \mathbb{R}$,

$$\lim_{n \to +\infty} \ \mathbb{P}\left(d_I(\mathcal{C}(\mathrm{R}_f(X)), \mathcal{C}(\mathrm{M}_f(\hat{X}_n, \mathcal{I}))) \leq \mathrm{res}(\mathcal{I}) \right) = 1.$$

I. ToMATo algorithm

1. Introduction to hierarchical and mode-seeking clustering

2. ToMATo algorithm and guarantees

II. Mapper algorithm

1. Reeb spaces and Mappers

2. Confidence intervals

Thanks! Questions?