Topological Machine Learning (II): Guiding ML models

1. Hierarchical and Mode Seeking Clustering
2. Topology-based Clustering
3. Topology-based Optimization

Topological Machine Learning (II): Guiding ML models

1. Hierarchical and Mode Seeking Clustering
2. Topology-based Clustering
3. Topology-based Optimization

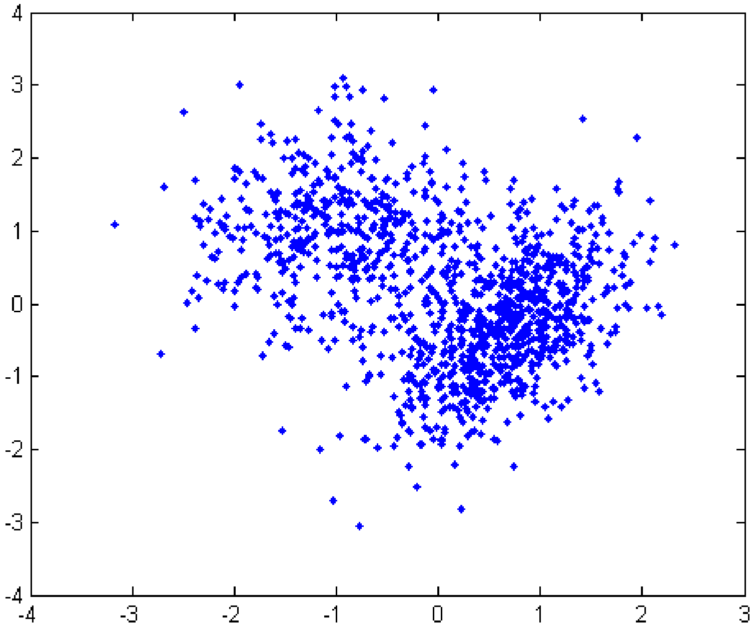
General clustering

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.

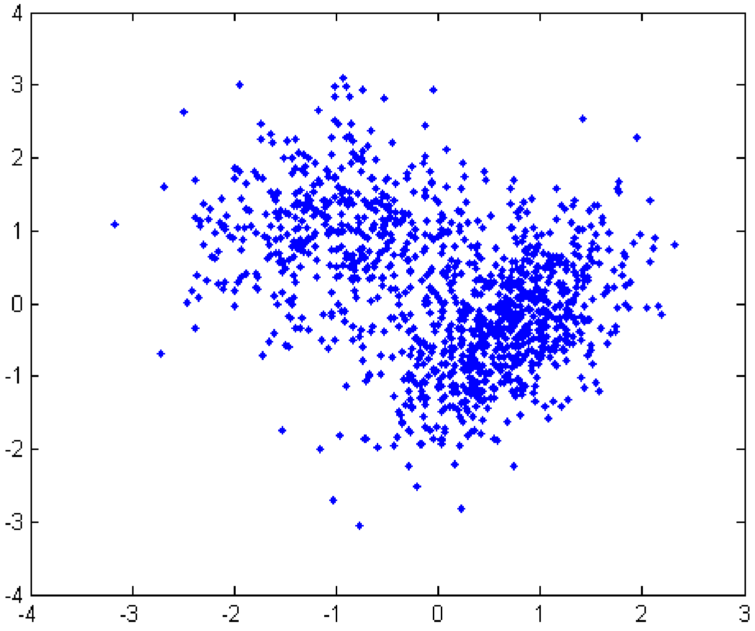
Input: a finite set of observations: point cloud embedded in an Euclidean space (i.e., with well-defined coordinates) or a more general metric space (pairwise distance or similarity) matrix.

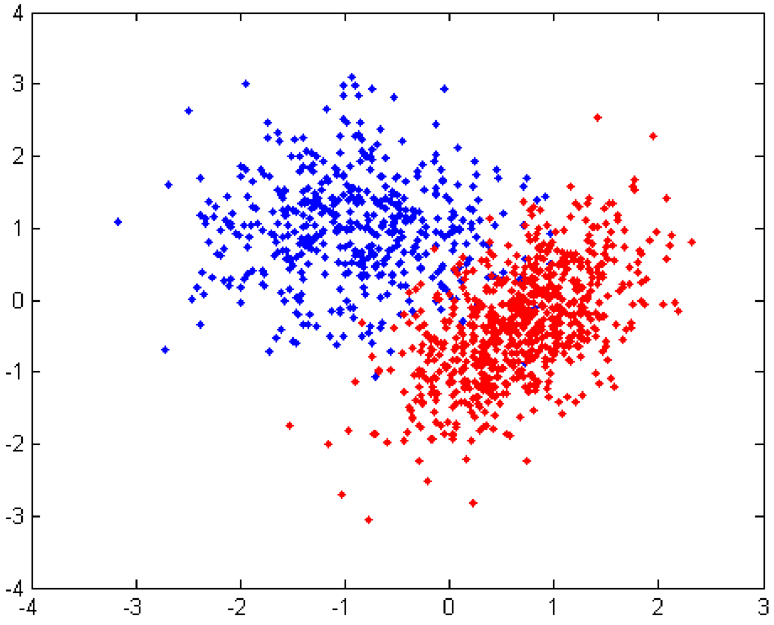


General clustering

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...

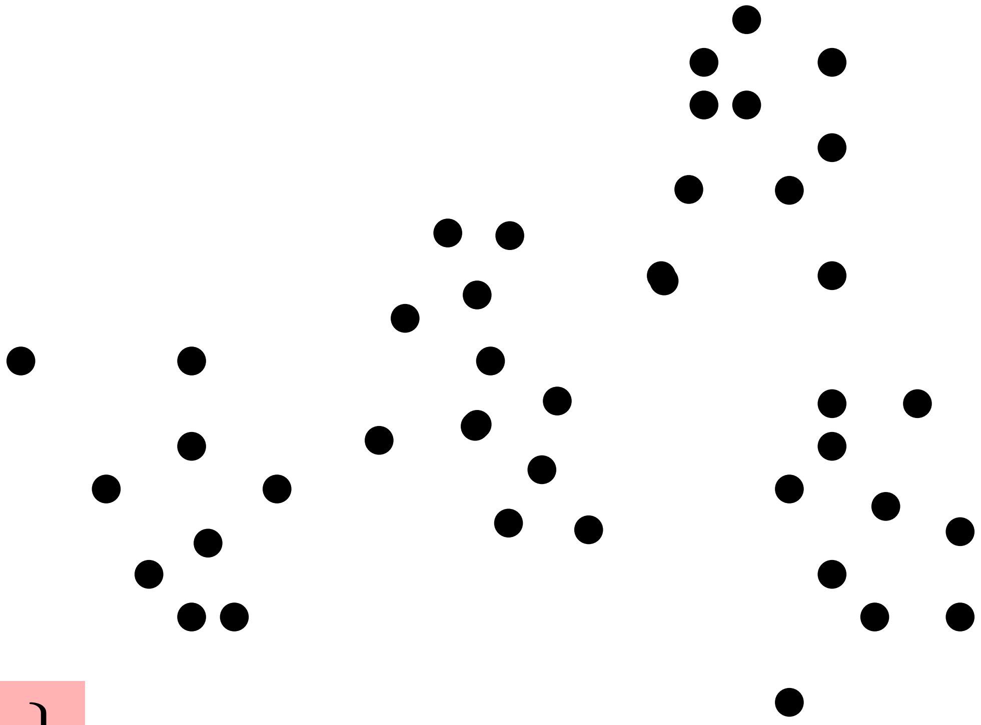
We will see a few standard algorithms and how they can be improved with (0-dimensional) persistent homology.

The k-means algorithm

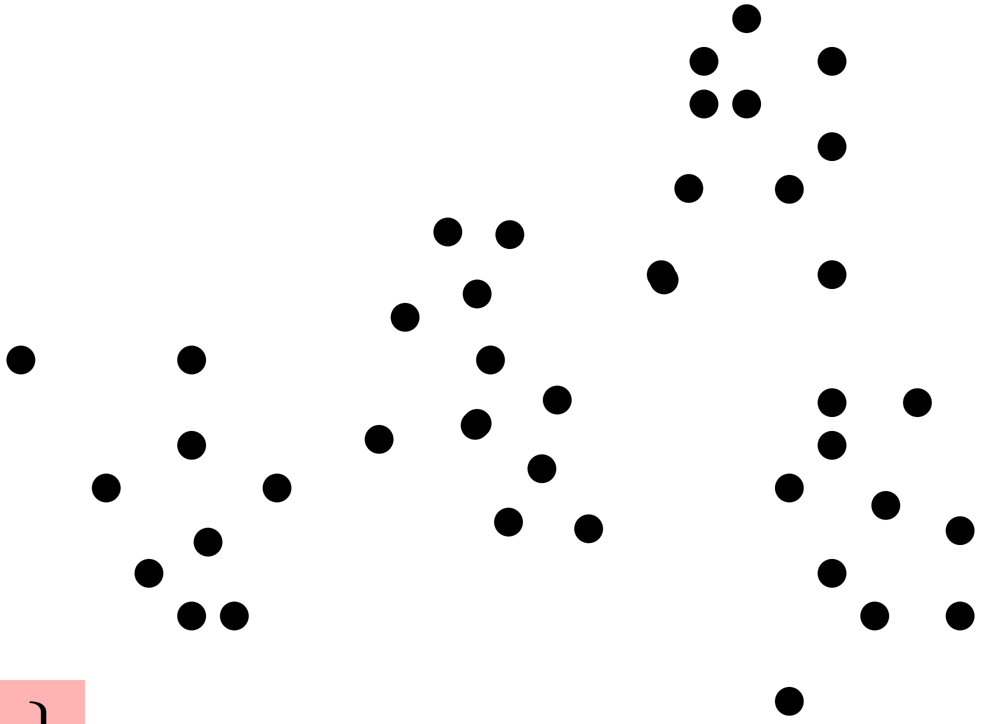
Input: A (large) set of n points X and an integer $k < n$.

Goal: Find a set of k points $L = \{y_1, \dots, y_k\}$ 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 $k < n$.

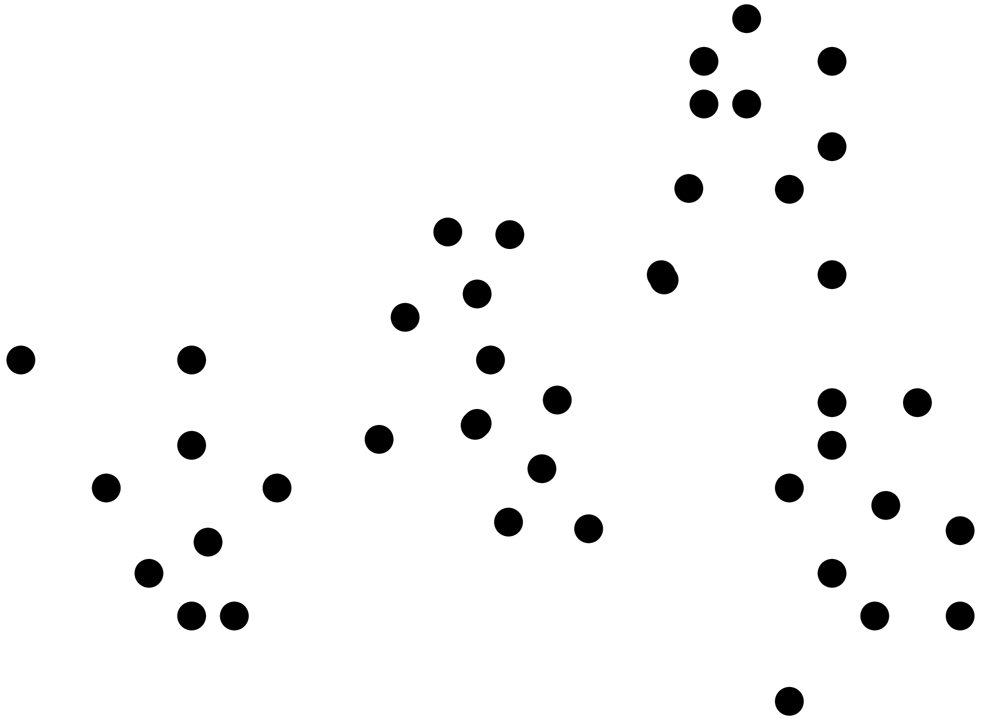
Goal: Find a set of k points $L = \{y_1, \ldots, y_k\}$ that minimizes

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

This is a NP hard problem!

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

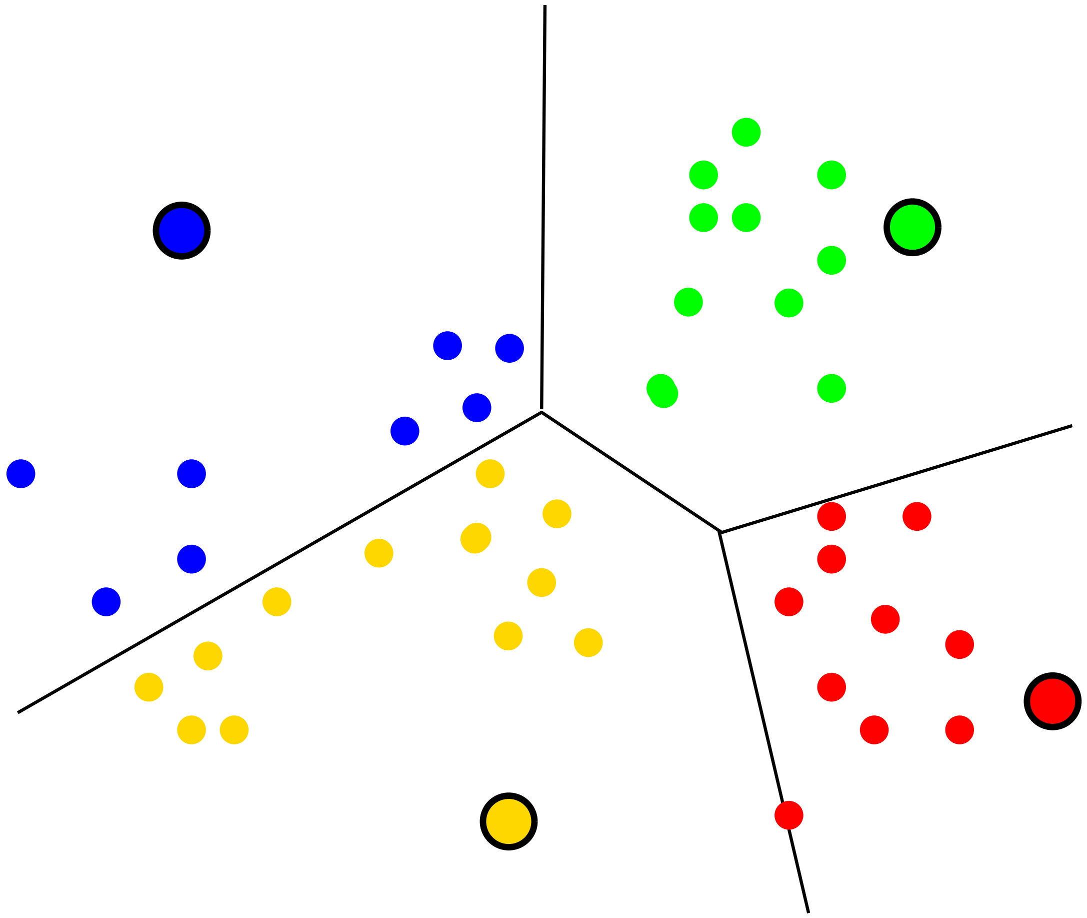
The k-means algorithm



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 $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$
```

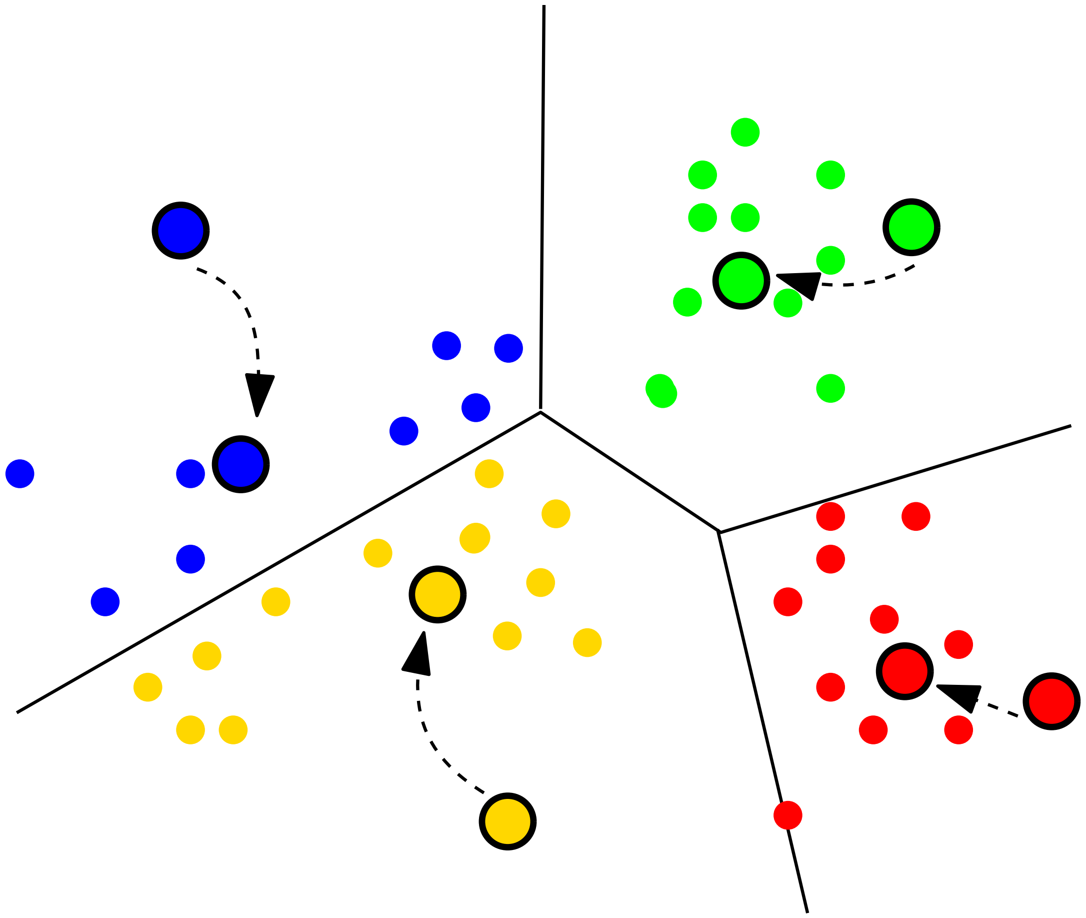
The k-means algorithm



Lloyd's algorithm

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

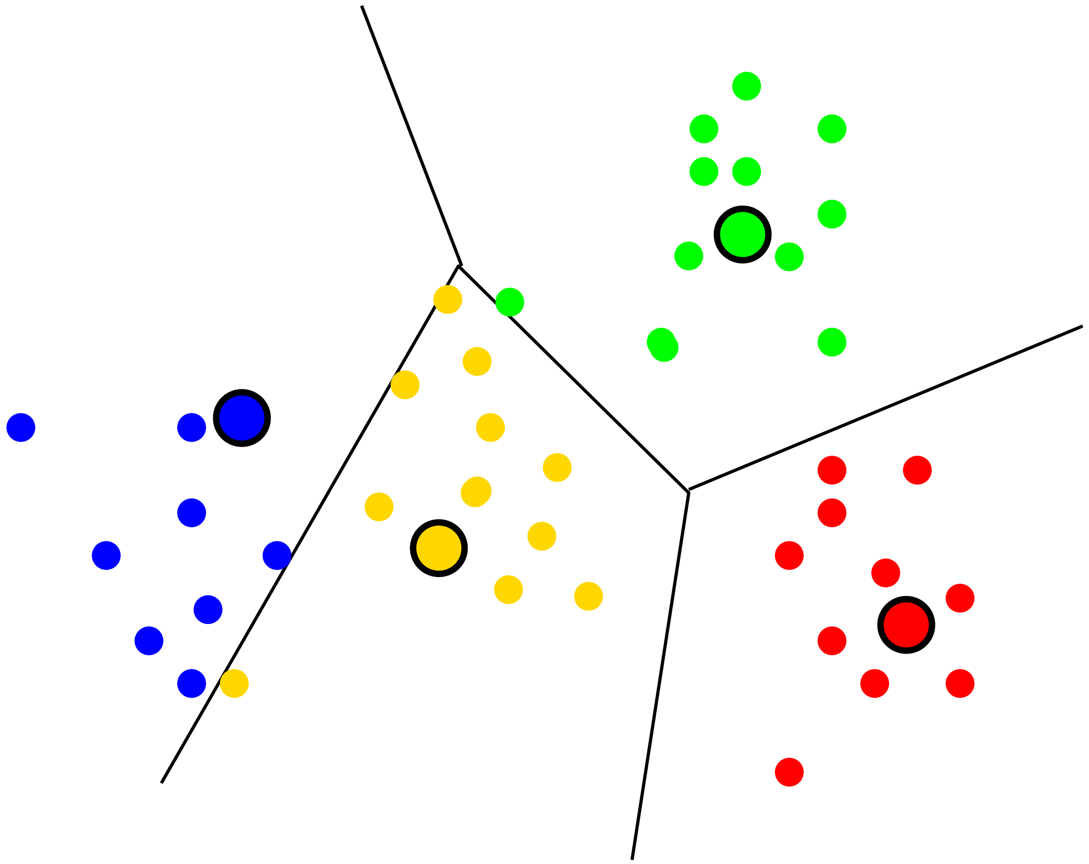
The k-means algorithm



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 $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$
```

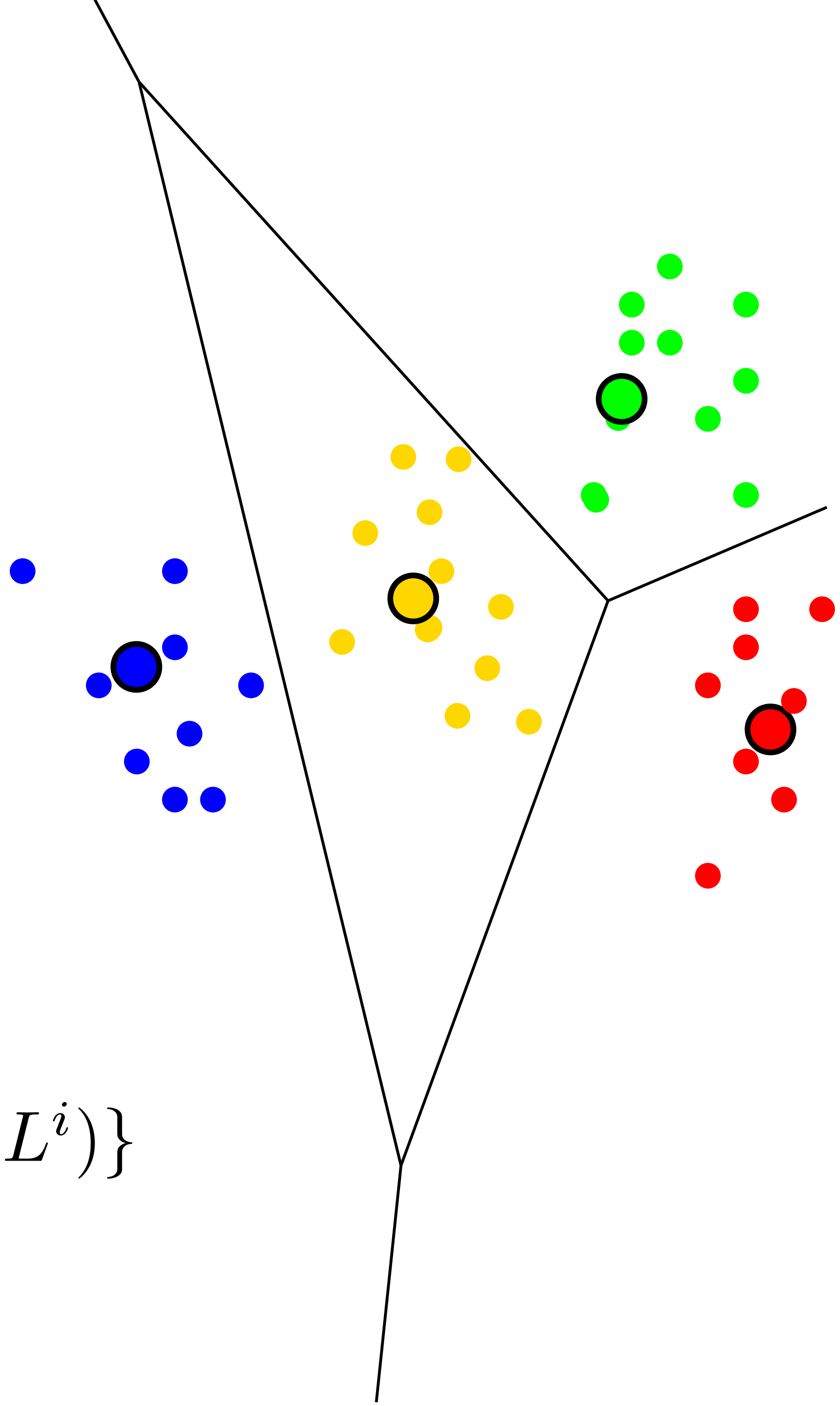
The k-means algorithm



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 $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$
```

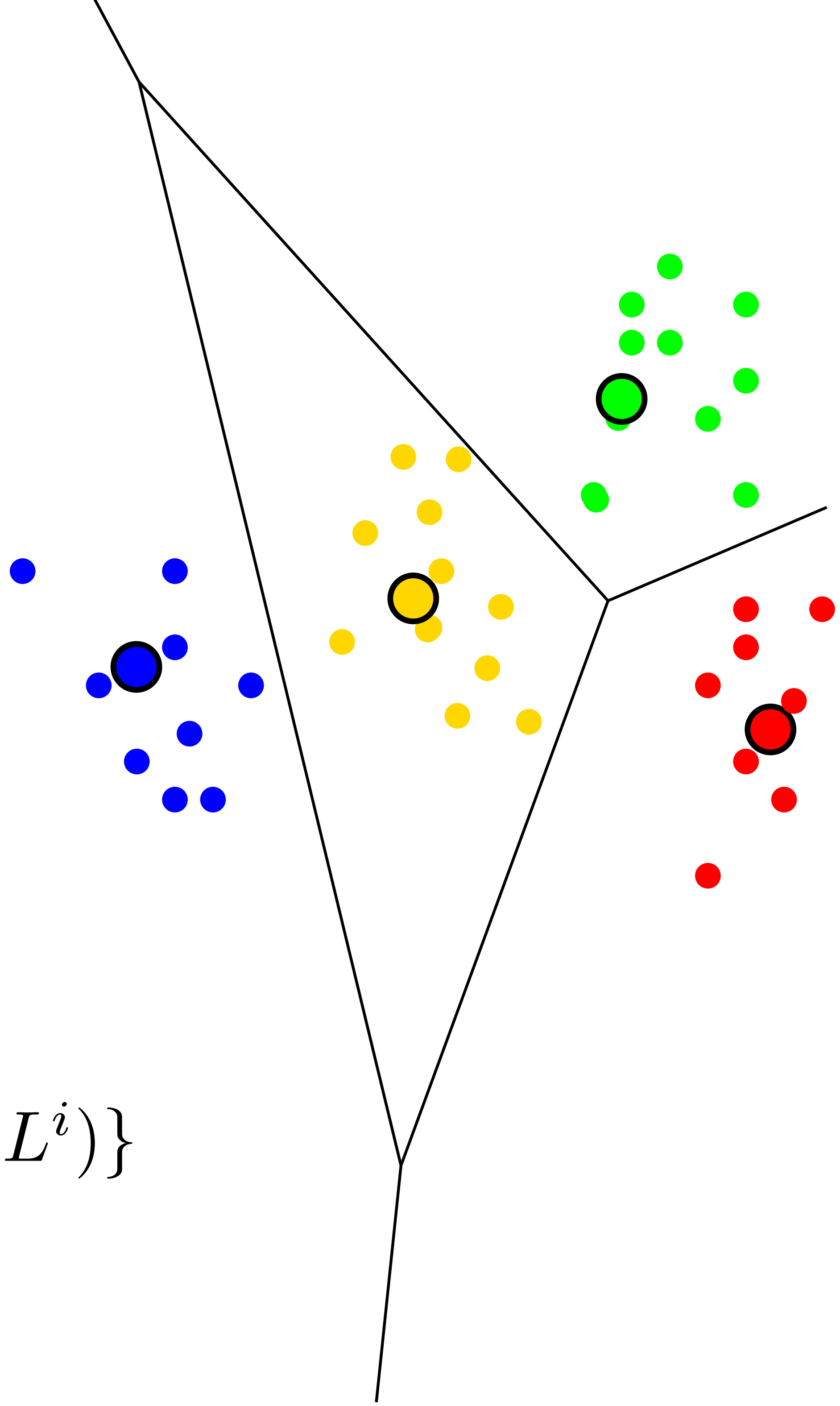
The k-means algorithm



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 $j \in \{1, \ldots, k\}$:
        $y_j^{i+1} \leftarrow \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$
    $i \leftarrow i + 1$
```

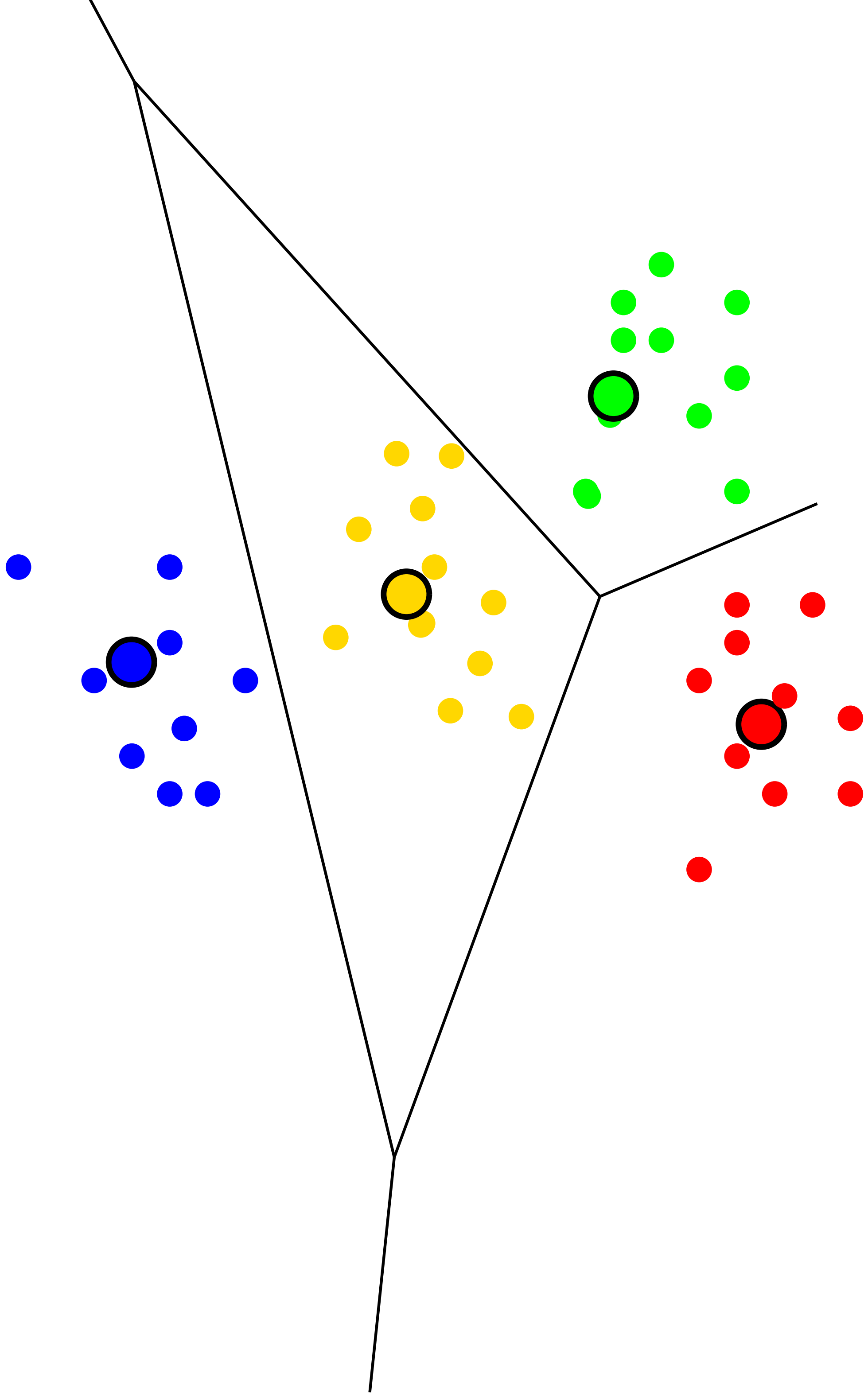
The k-means algorithm



Lloyd's algorithm

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

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

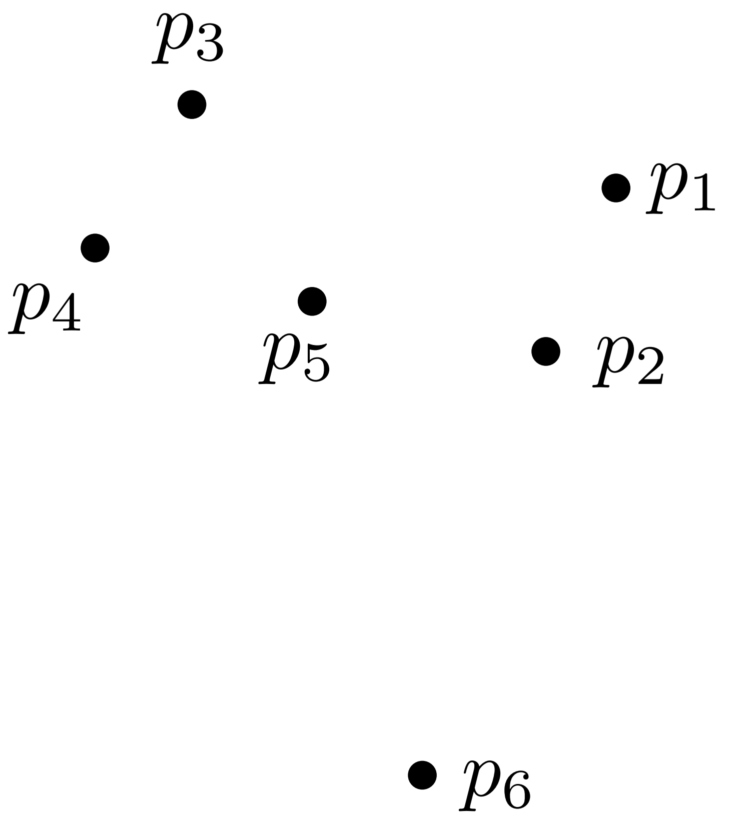
Goal: Build a hierarchy of clusters (nested family of partitions).

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).



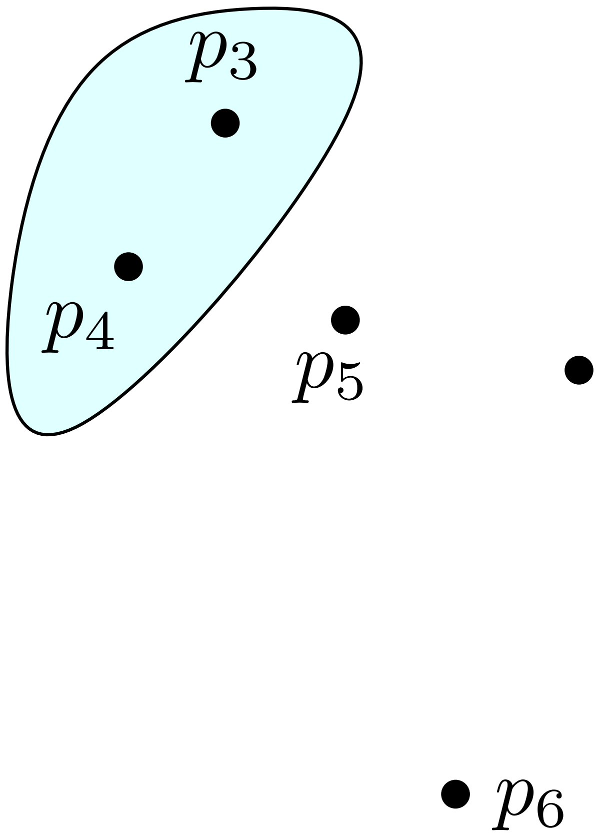
p_1
p_2
p_3
p_4
p_5
p_6

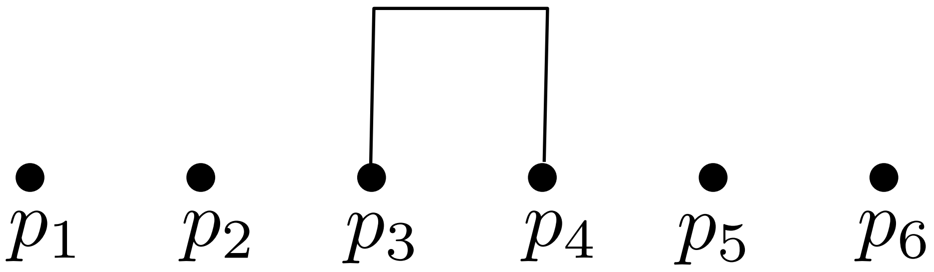
Hierarchical clustering algorithms

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).



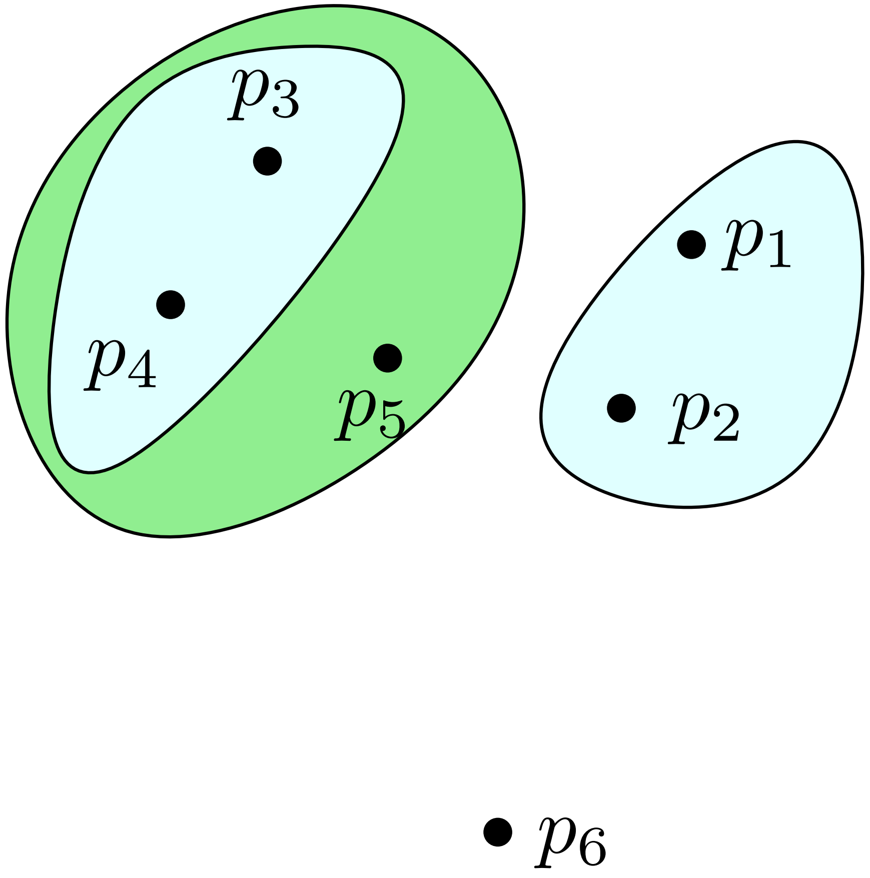


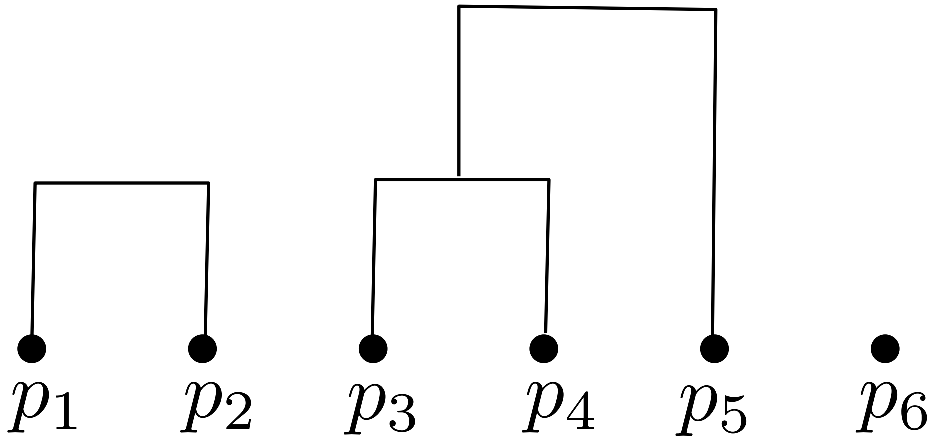
Hierarchical clustering algorithms

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).



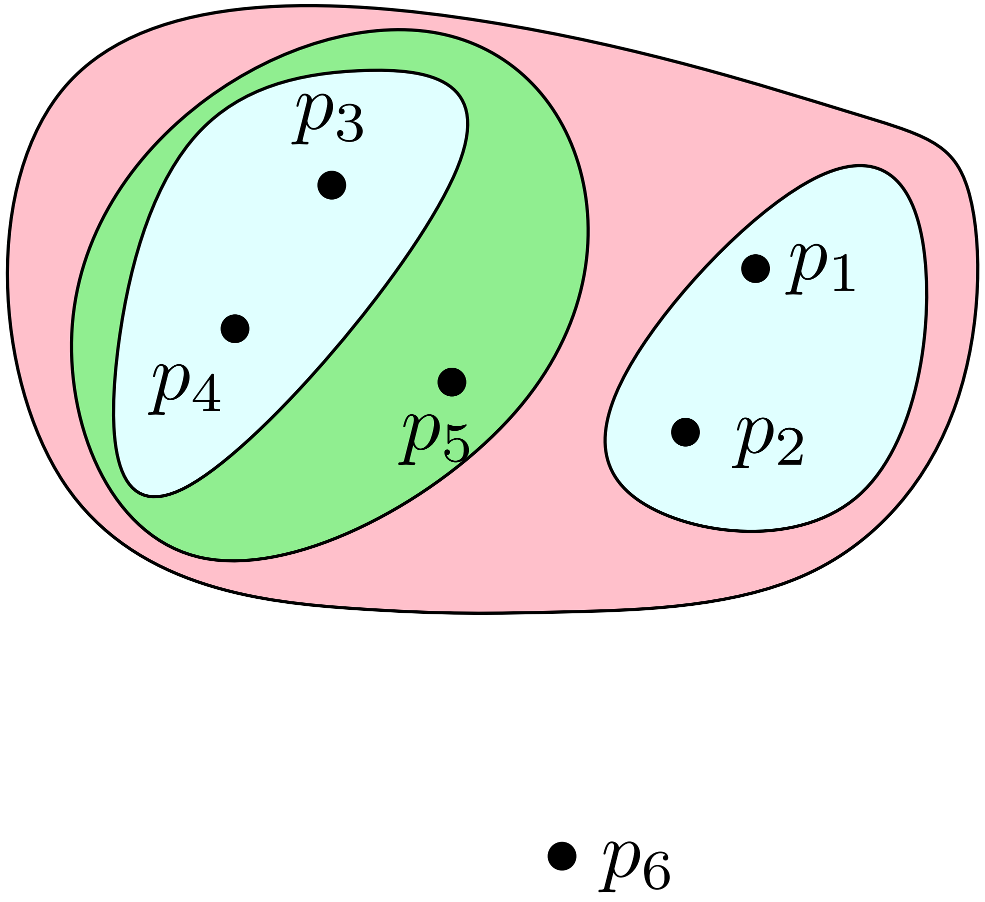


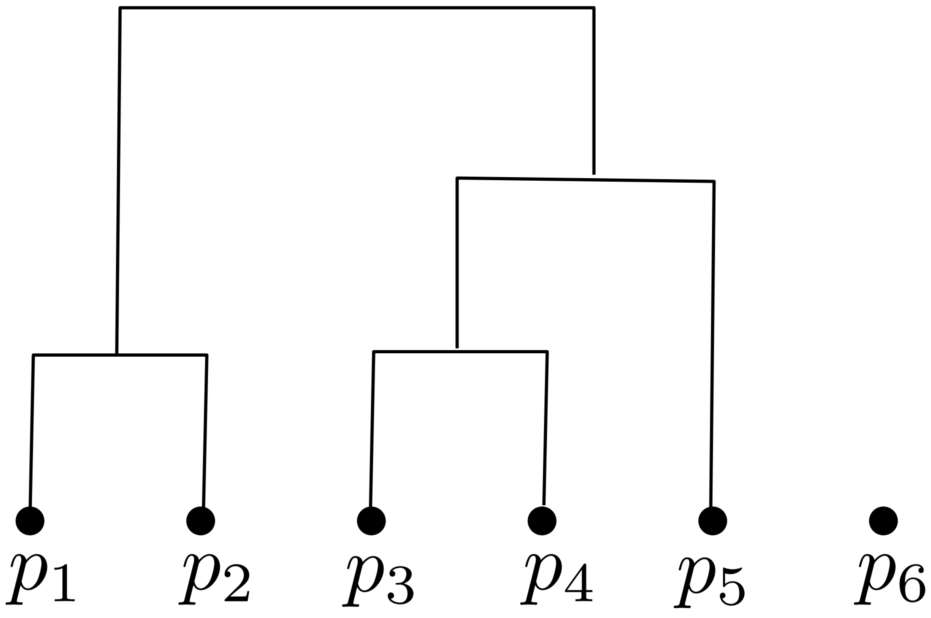
Hierarchical clustering algorithms

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).



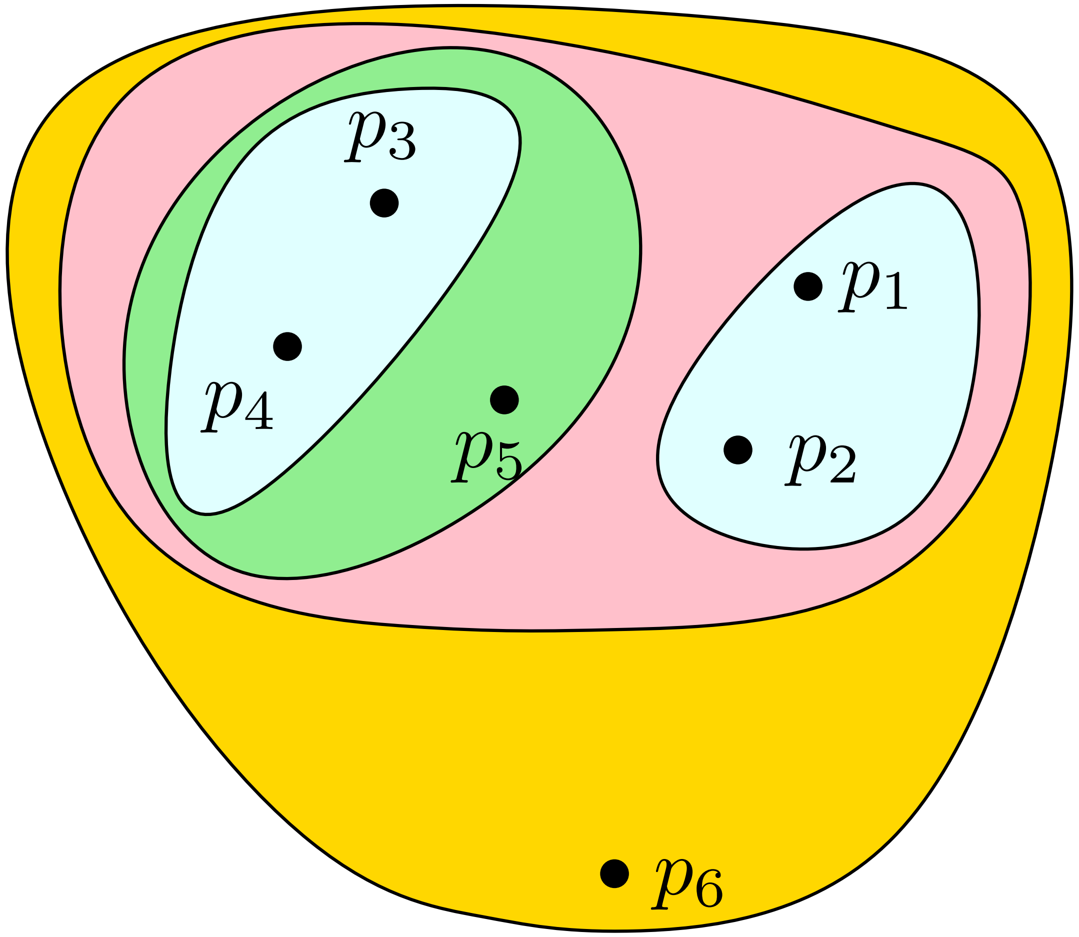


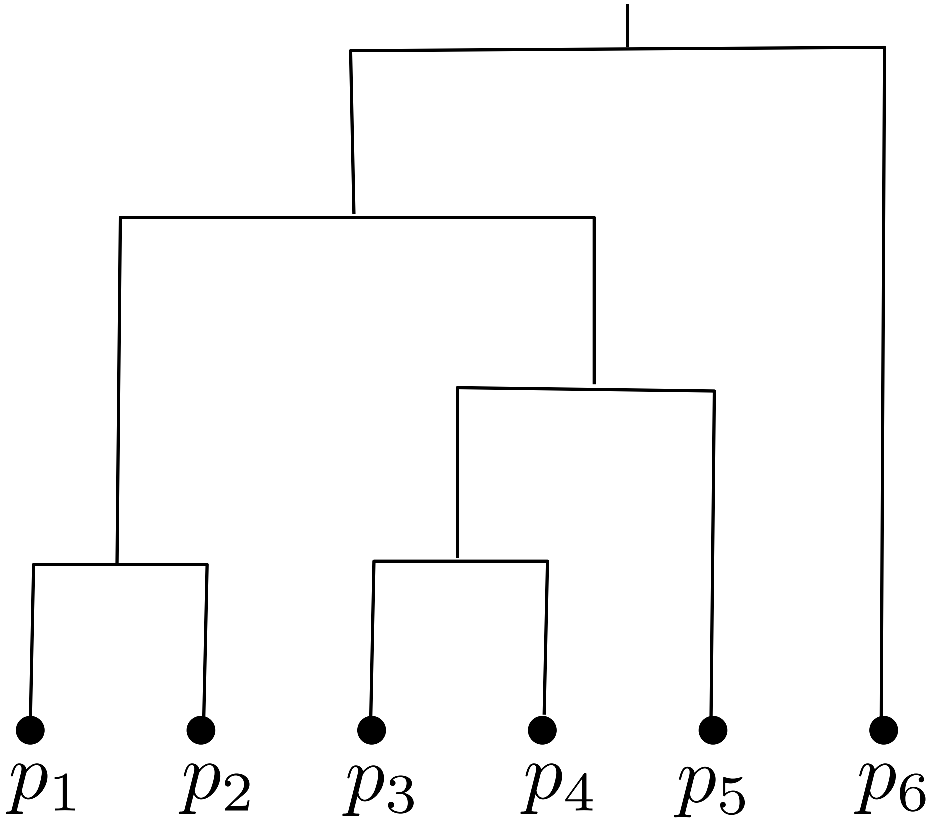
Hierarchical clustering algorithms

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).





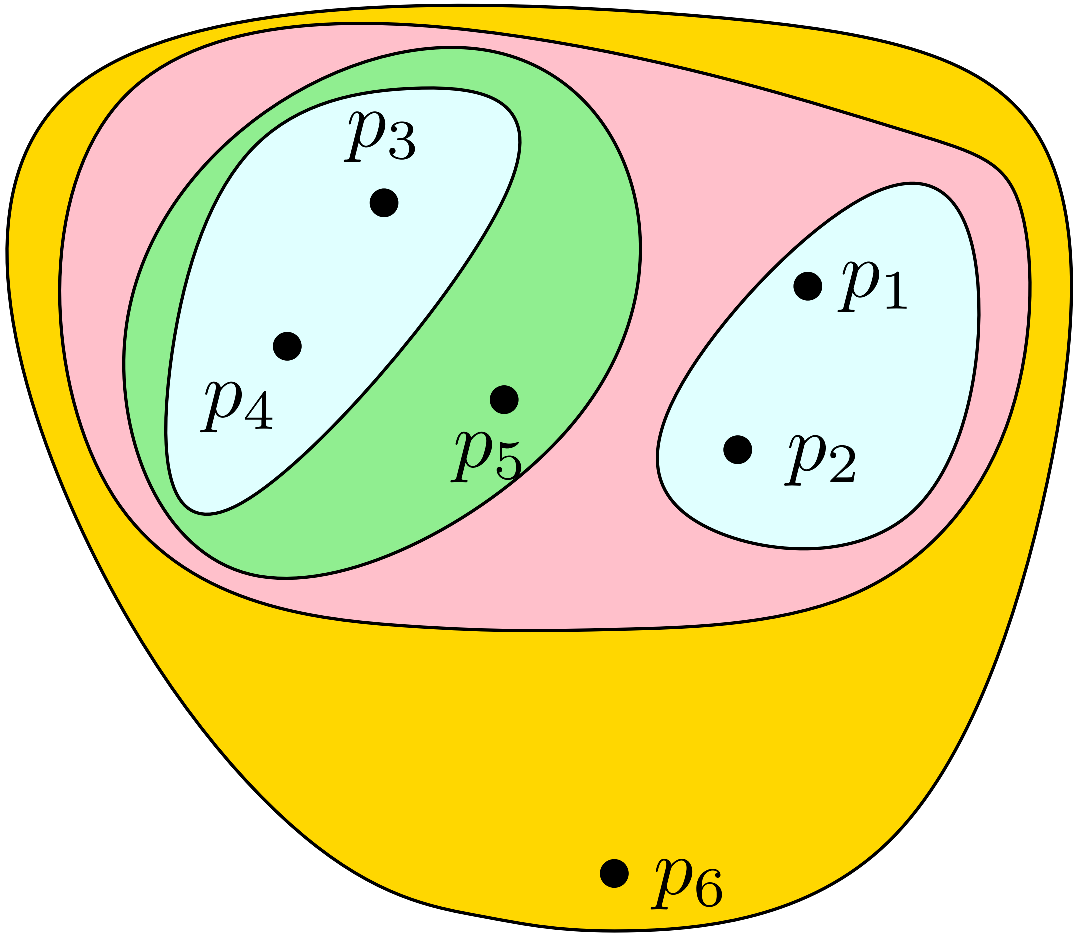
Hierarchical clustering algorithms

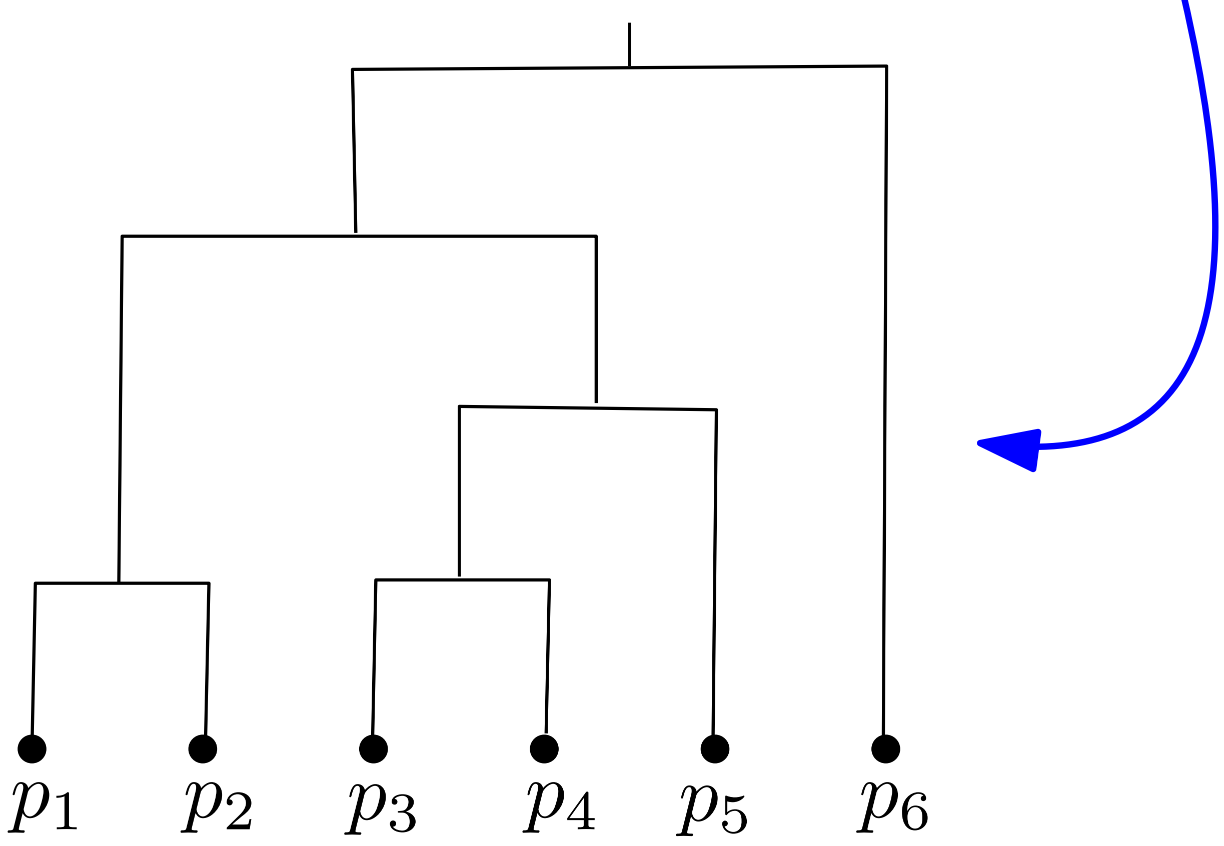
Goal: Build a hierarchy of clusters (nested family of partitions).

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.



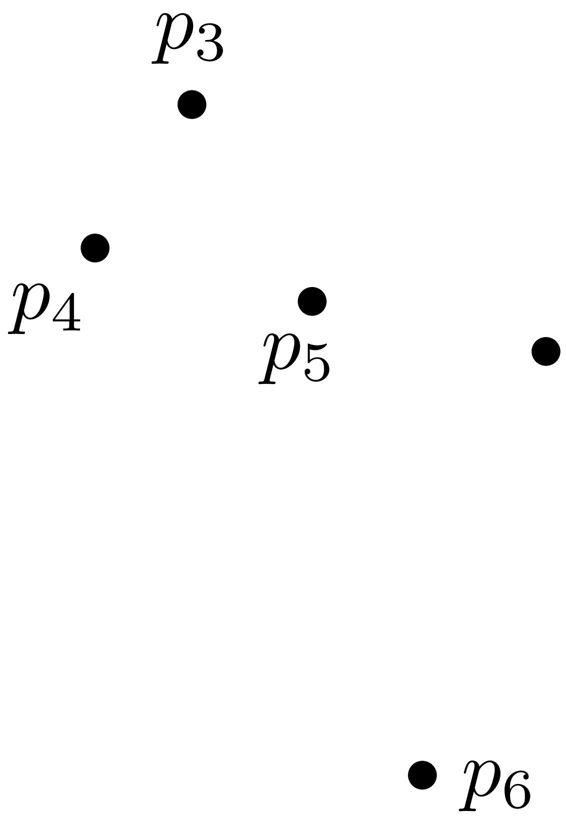


Hierarchical clustering algorithms

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

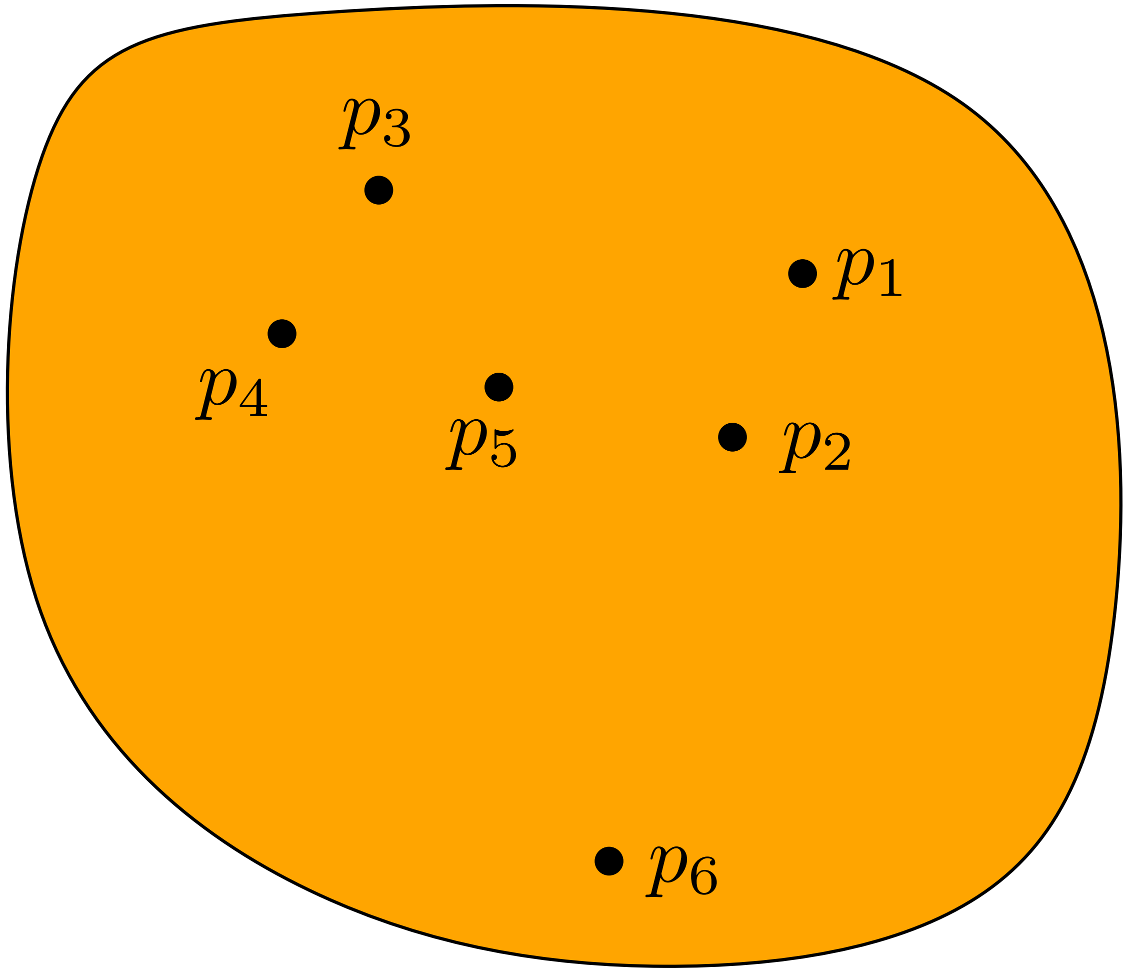
Goal: Build a hierarchy of clusters (nested family of partitions).

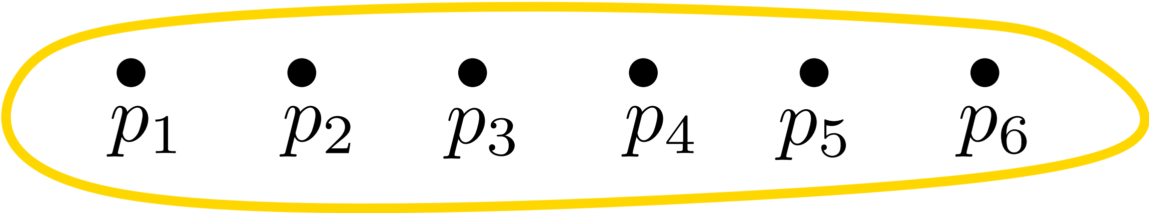
Agglomerative (bottom-up)

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.



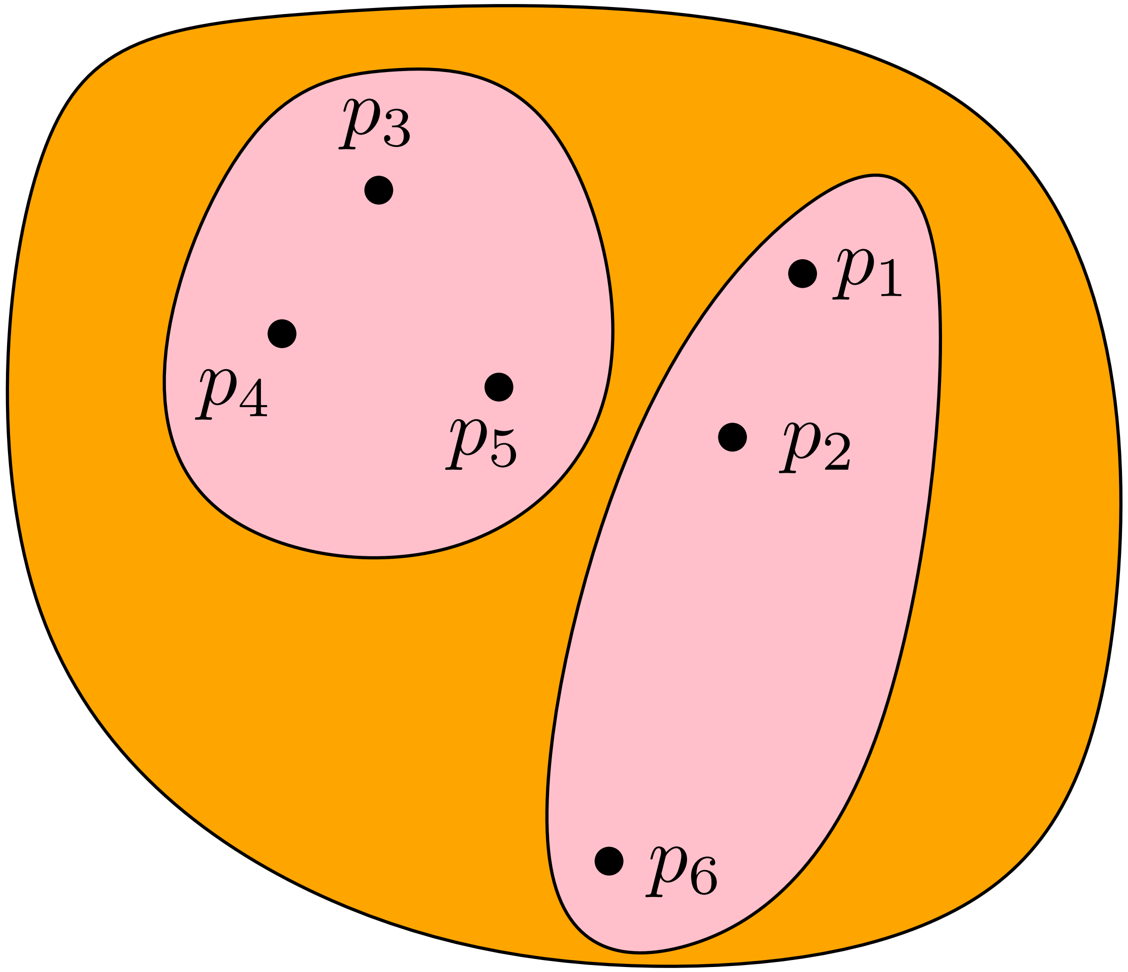


Hierarchical clustering algorithms

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

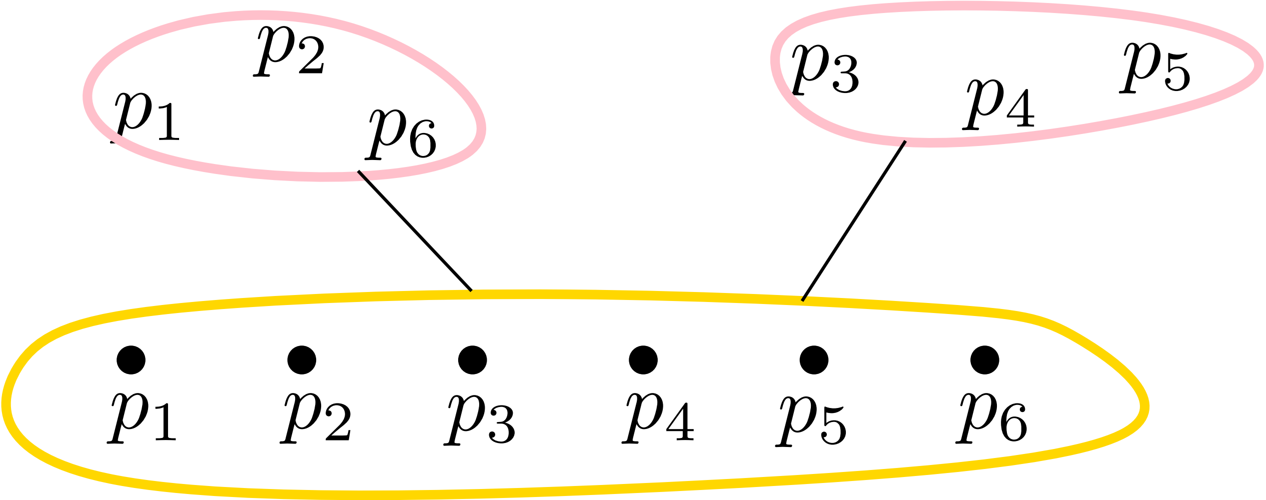
Agglomerative (bottom-up)

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

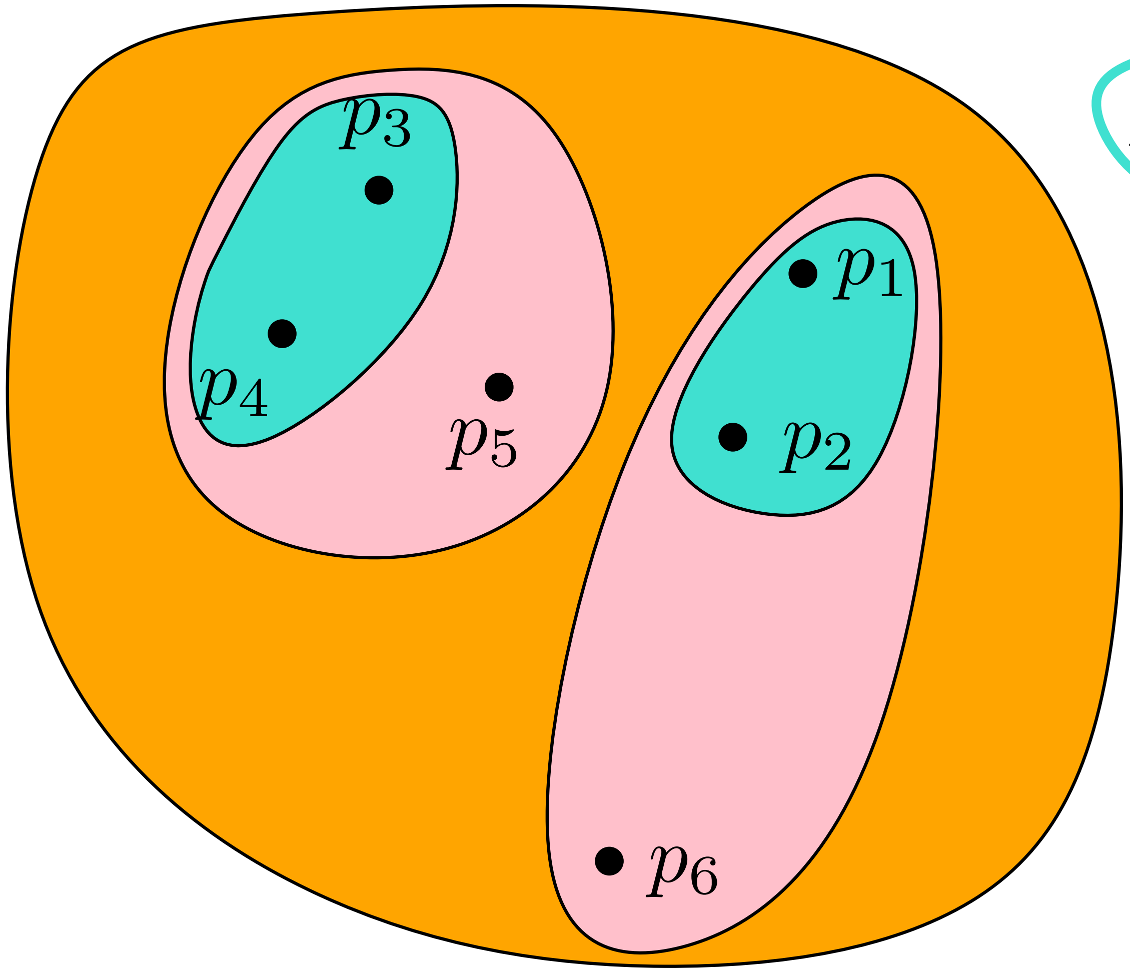
Goal: Build a hierarchy of clusters (nested family of partitions).

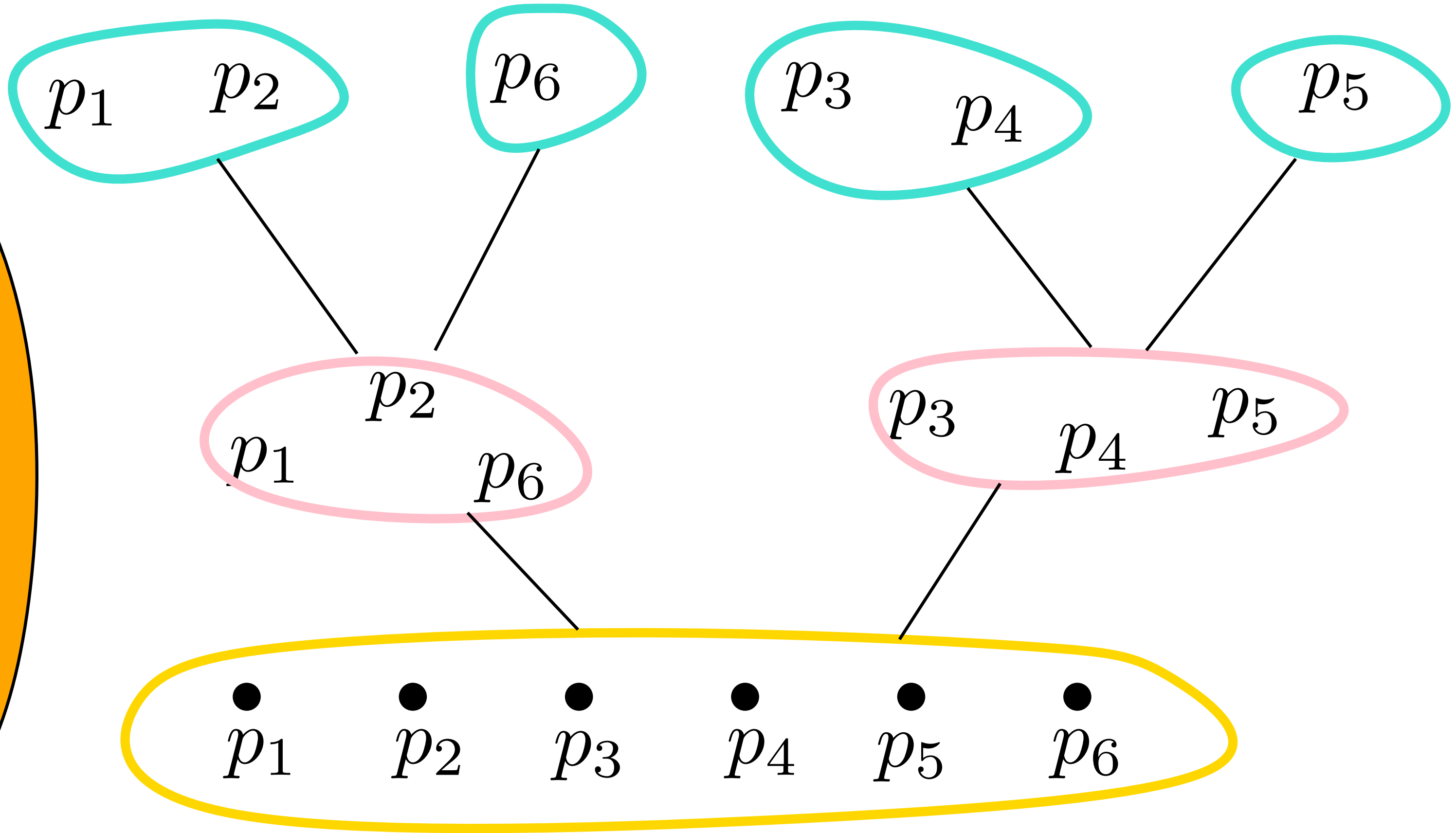
Agglomerative (bottom-up)

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 $X_n = \{x_1, \dots, x_n\}$ 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

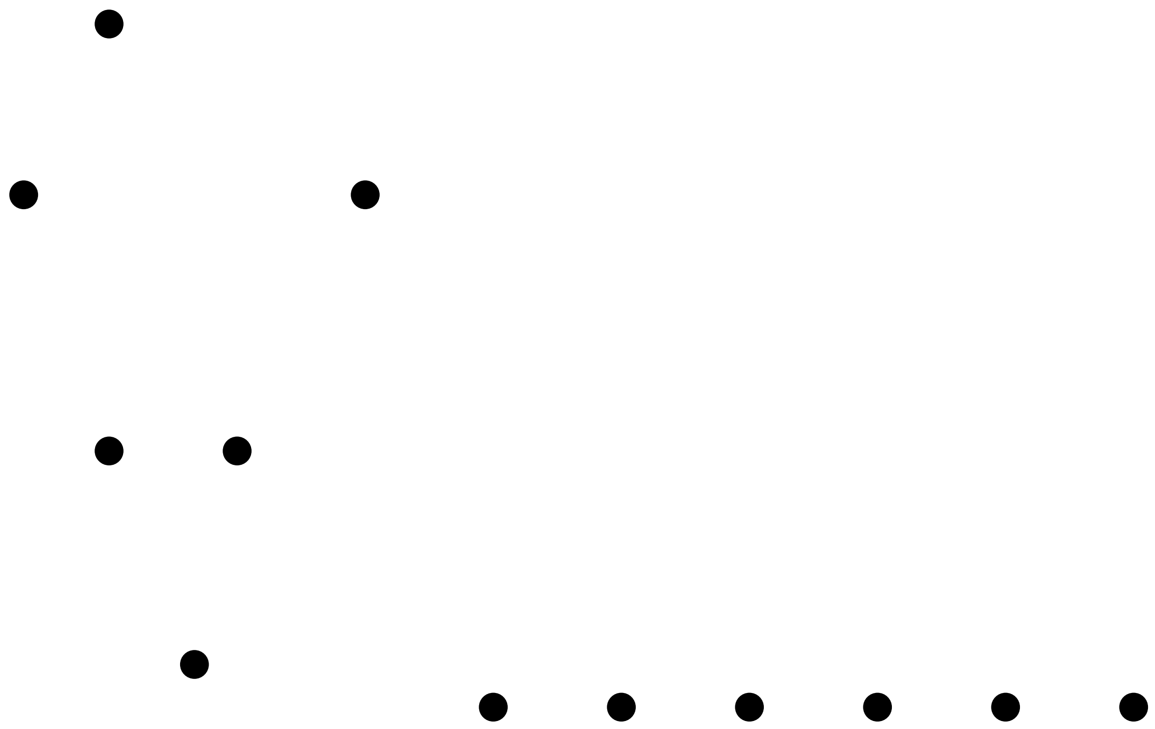
Input: A set $X_n = \{x_1, \dots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

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

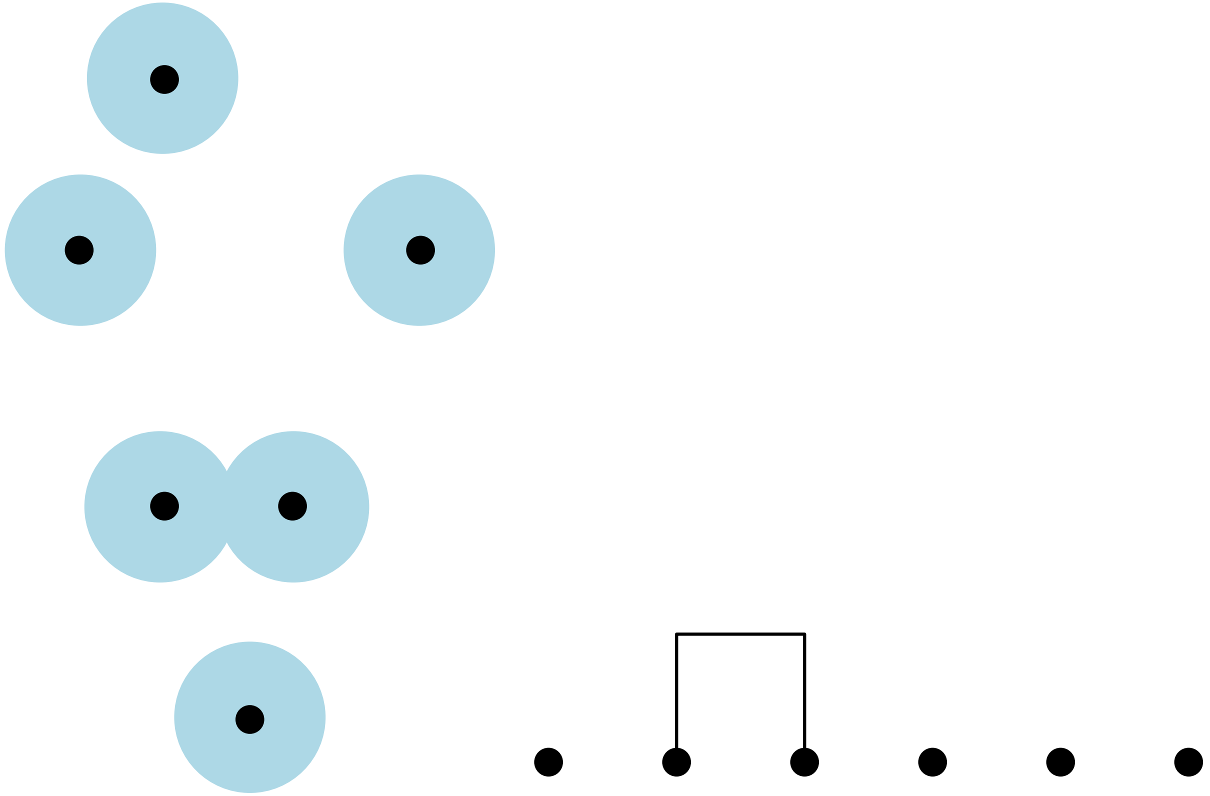
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

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

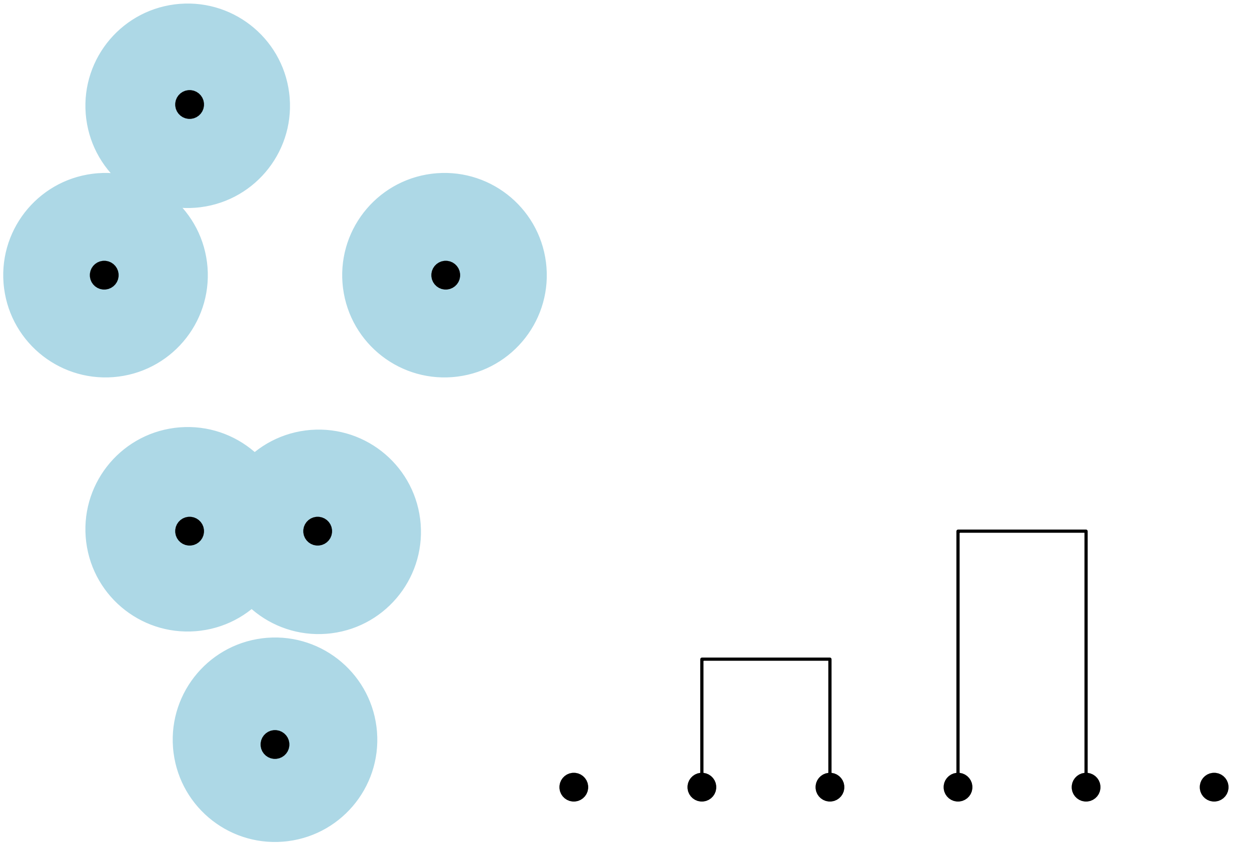
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

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

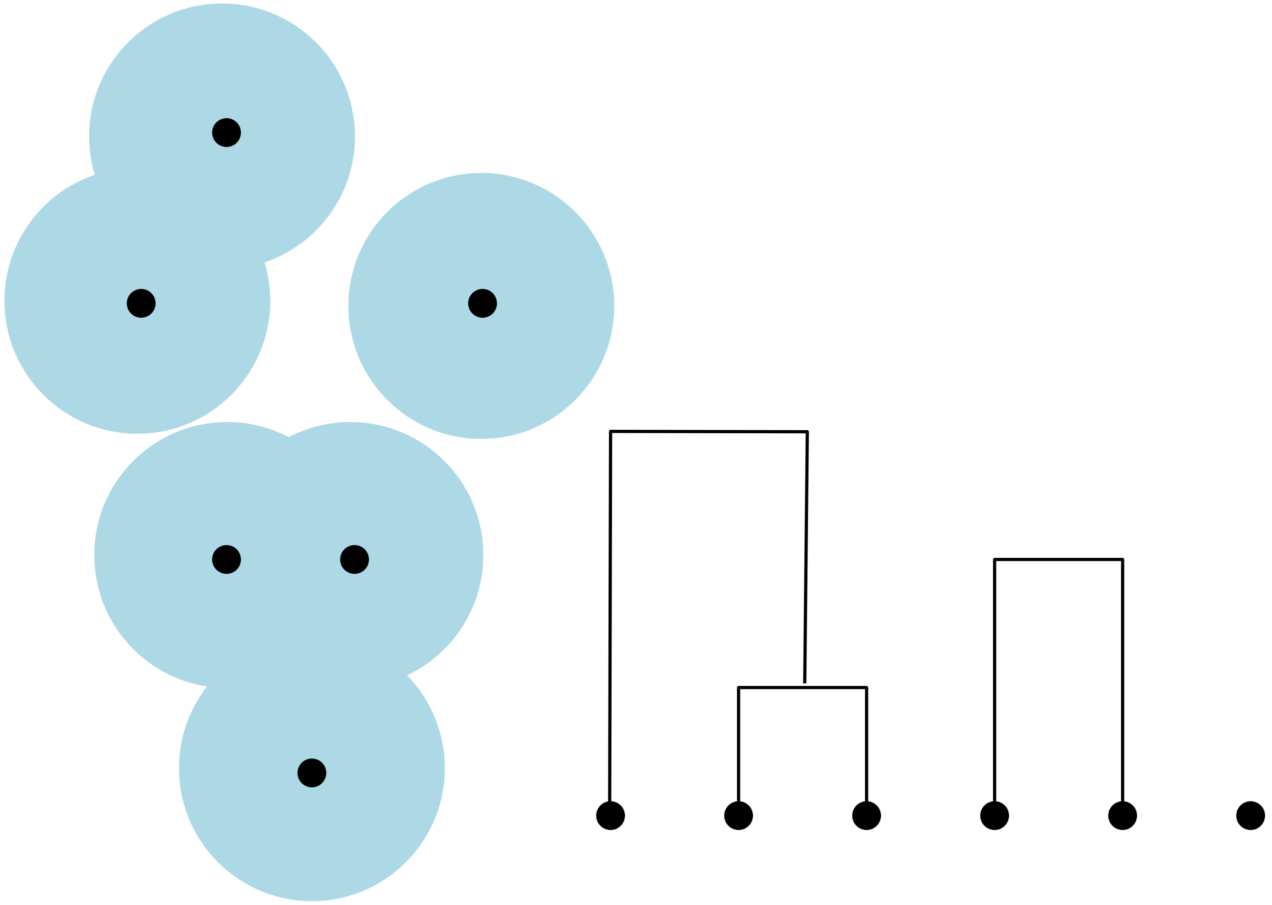
Input: A set $X_n = \{x_1, \dots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

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

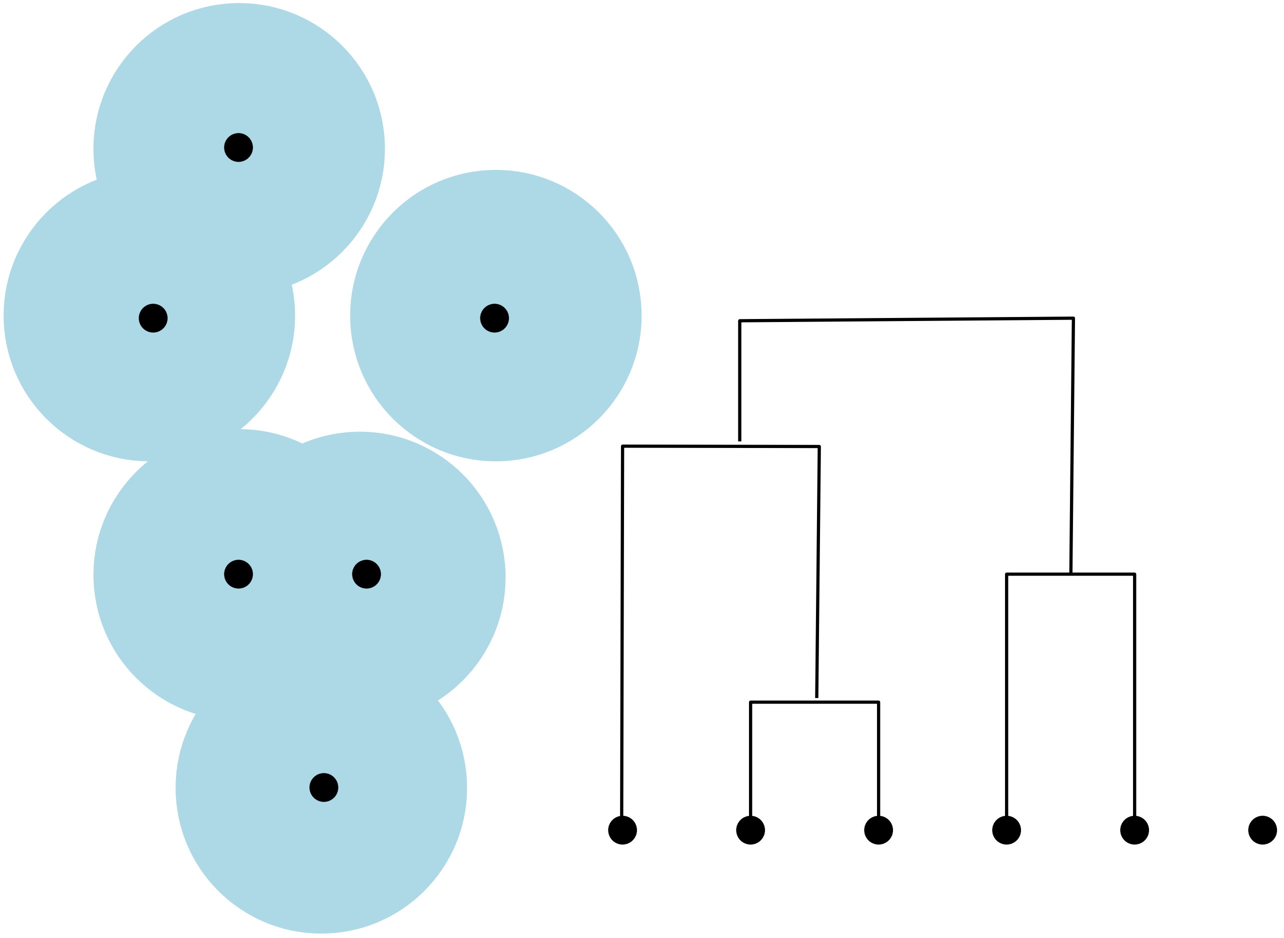
Input: A set $X_n = \{x_1, \ldots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

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

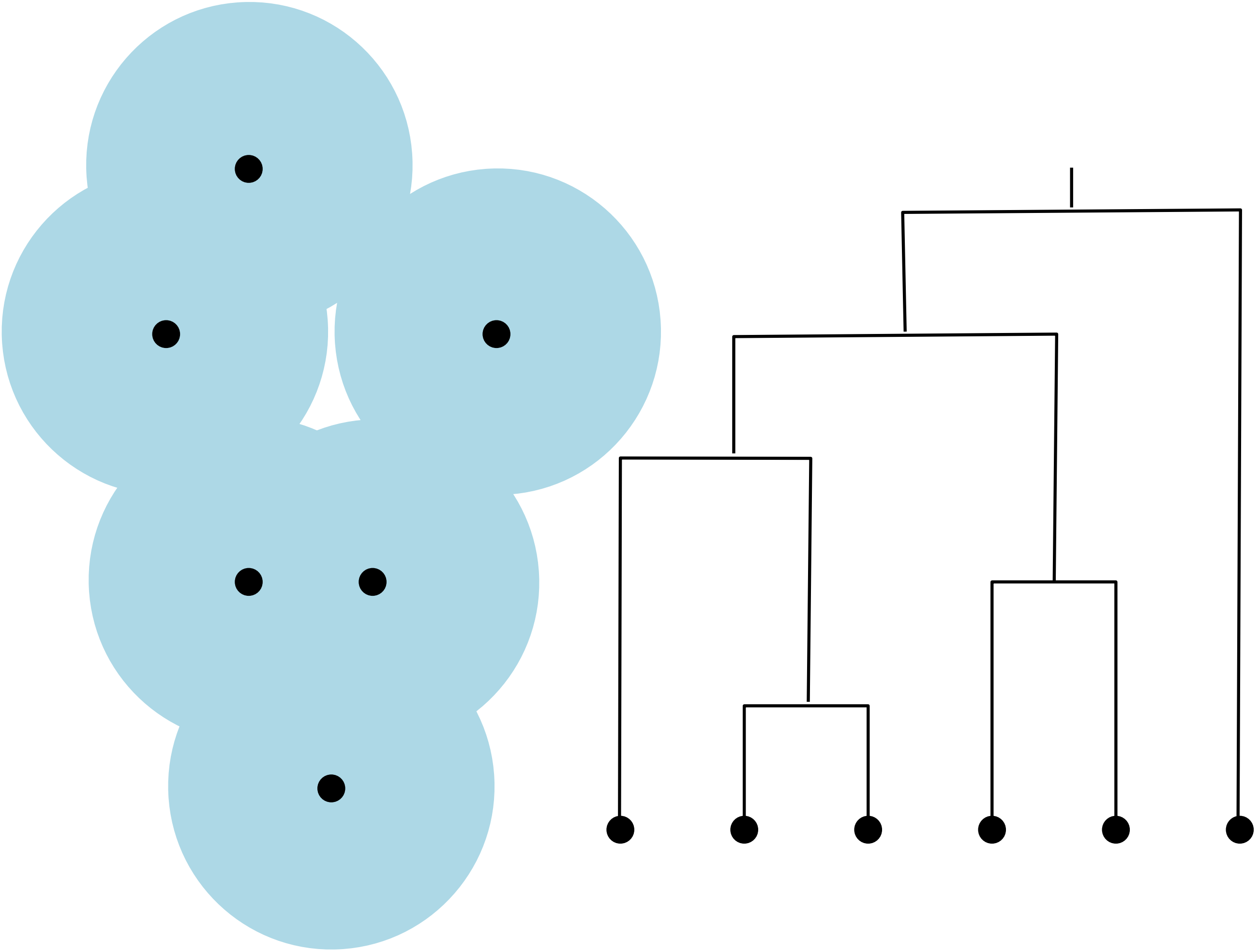
Input: A set $X_n = \{x_1, \dots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $((d_{i,j}))_{i,j}$).

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.

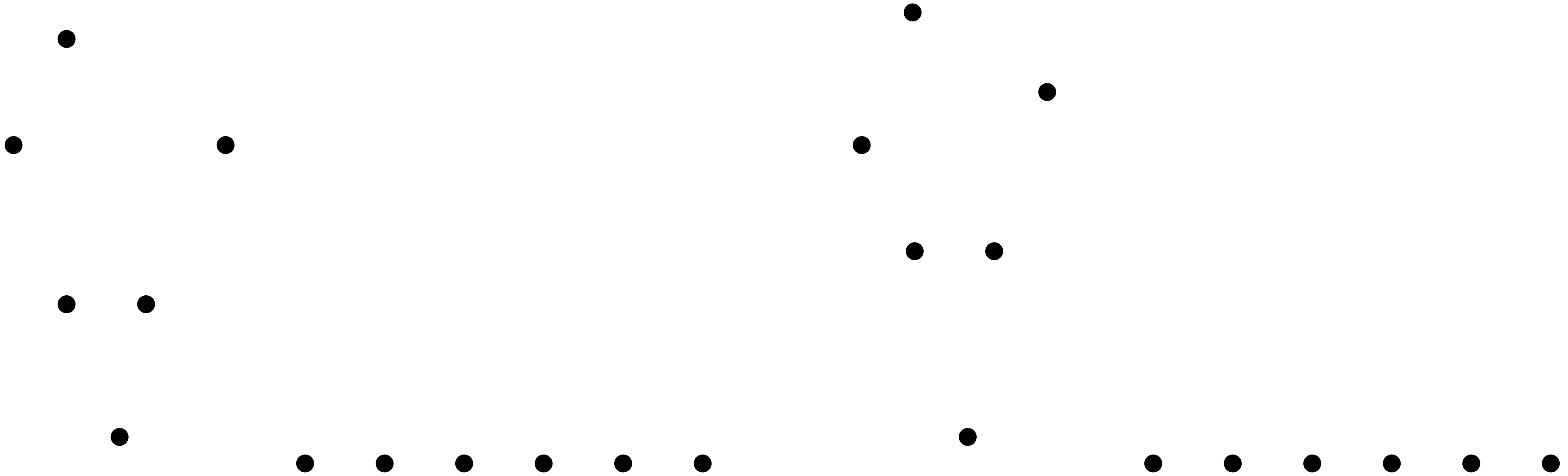
2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).



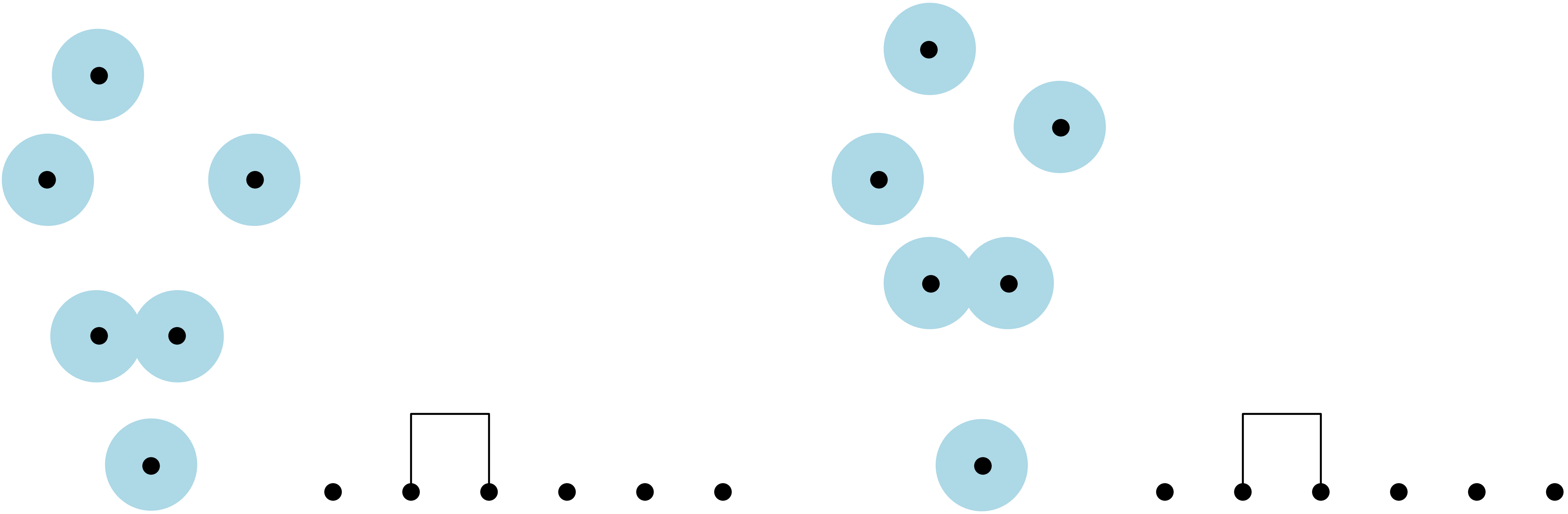
Output: the resulting dendrogram.

The (in)stability of dendrograms

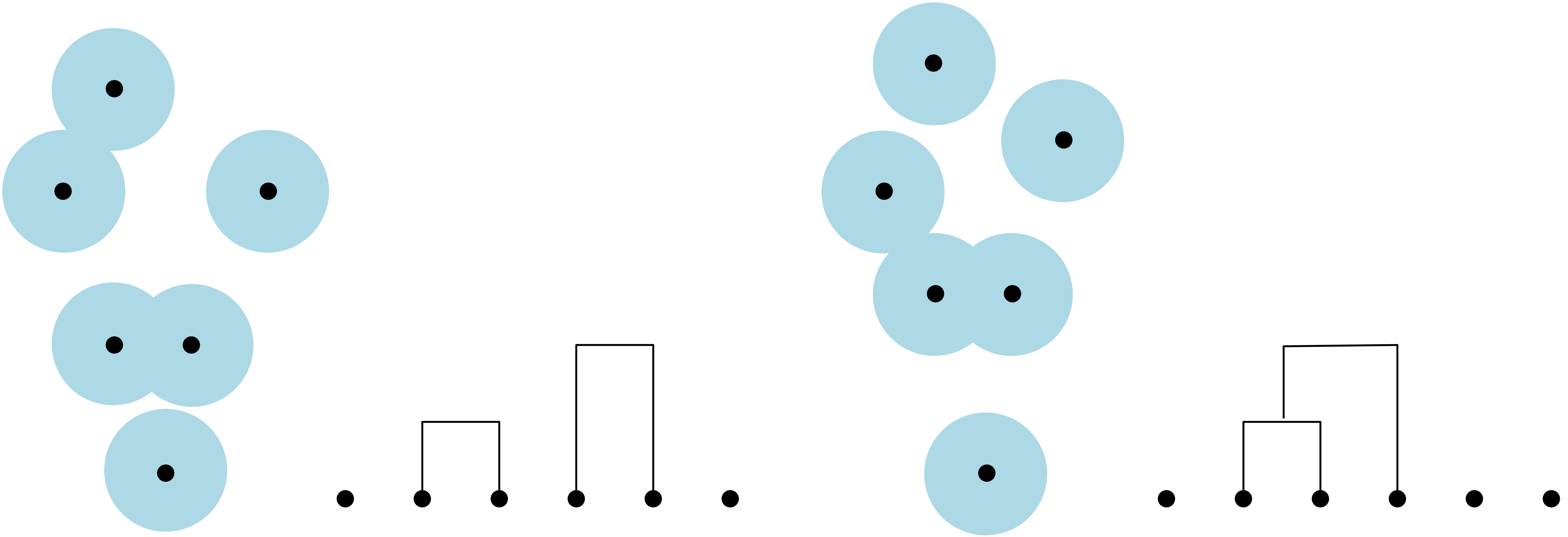
The (in)stability of dendrograms



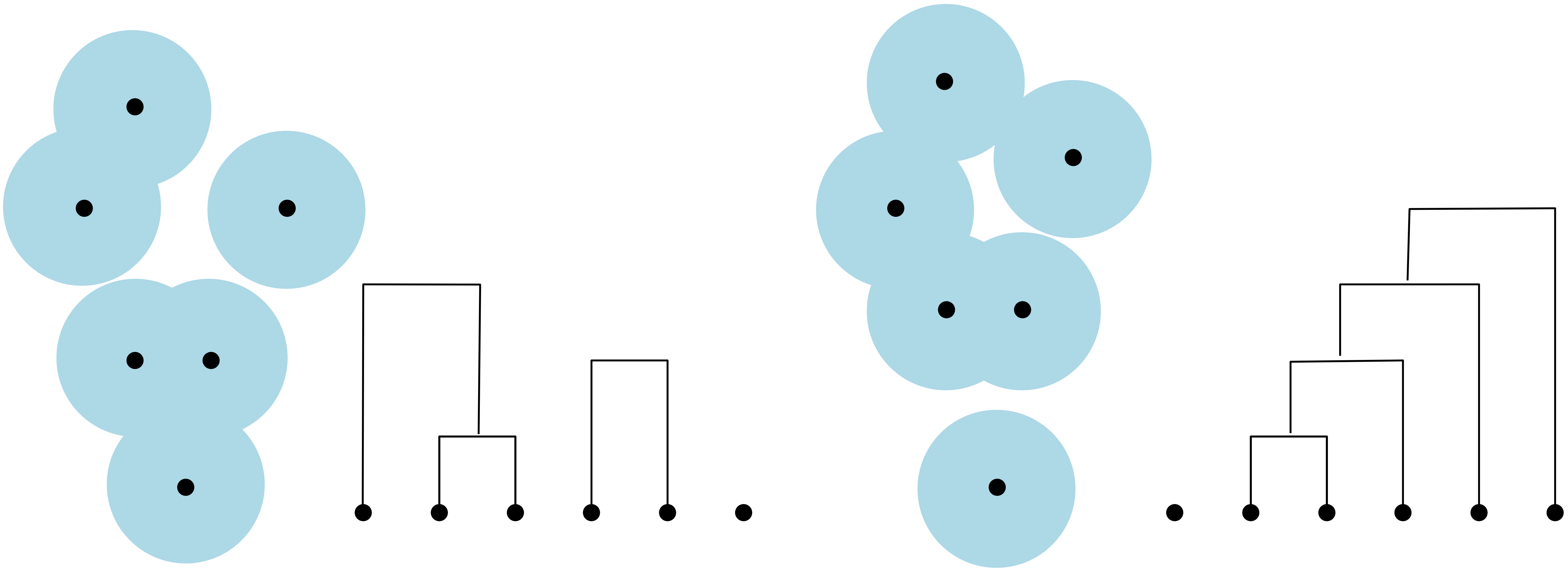
The (in)stability of dendrograms



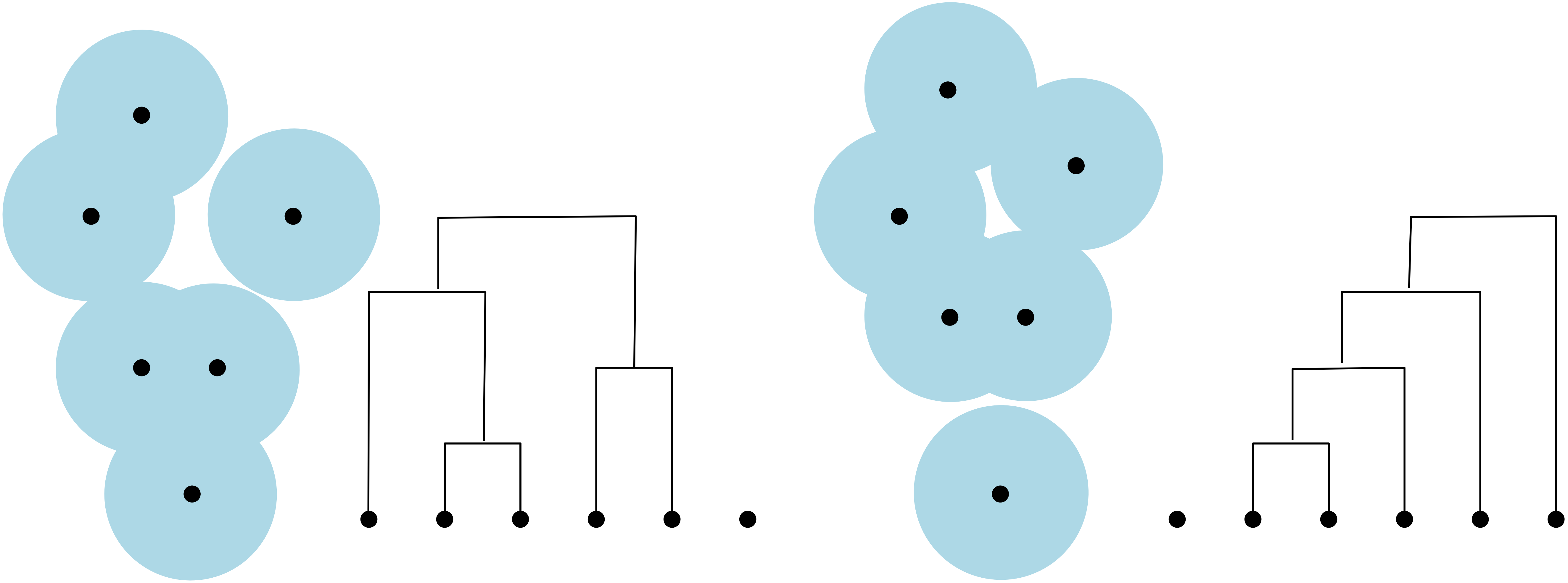
The (in)stability of dendrograms



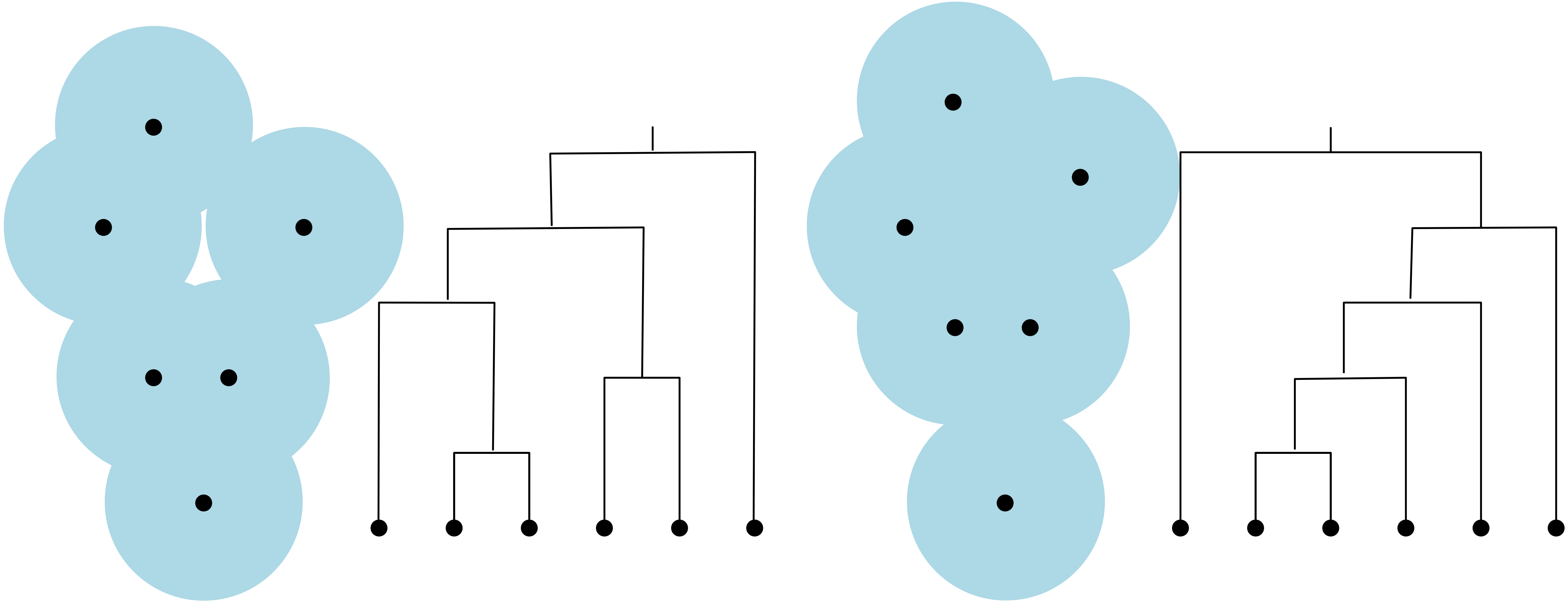
The (in)stability of dendrograms



The (in)stability of dendrograms

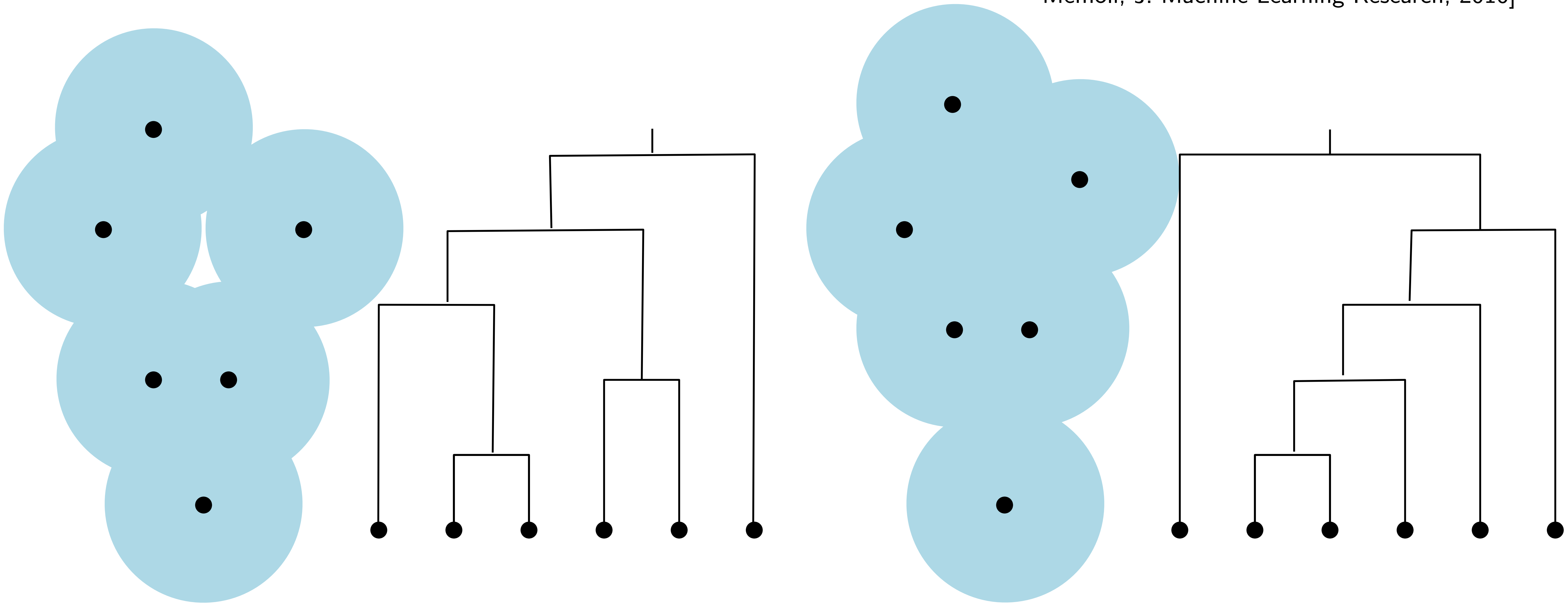


The (in)stability of dendrograms



The (in)stability of dendrograms

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



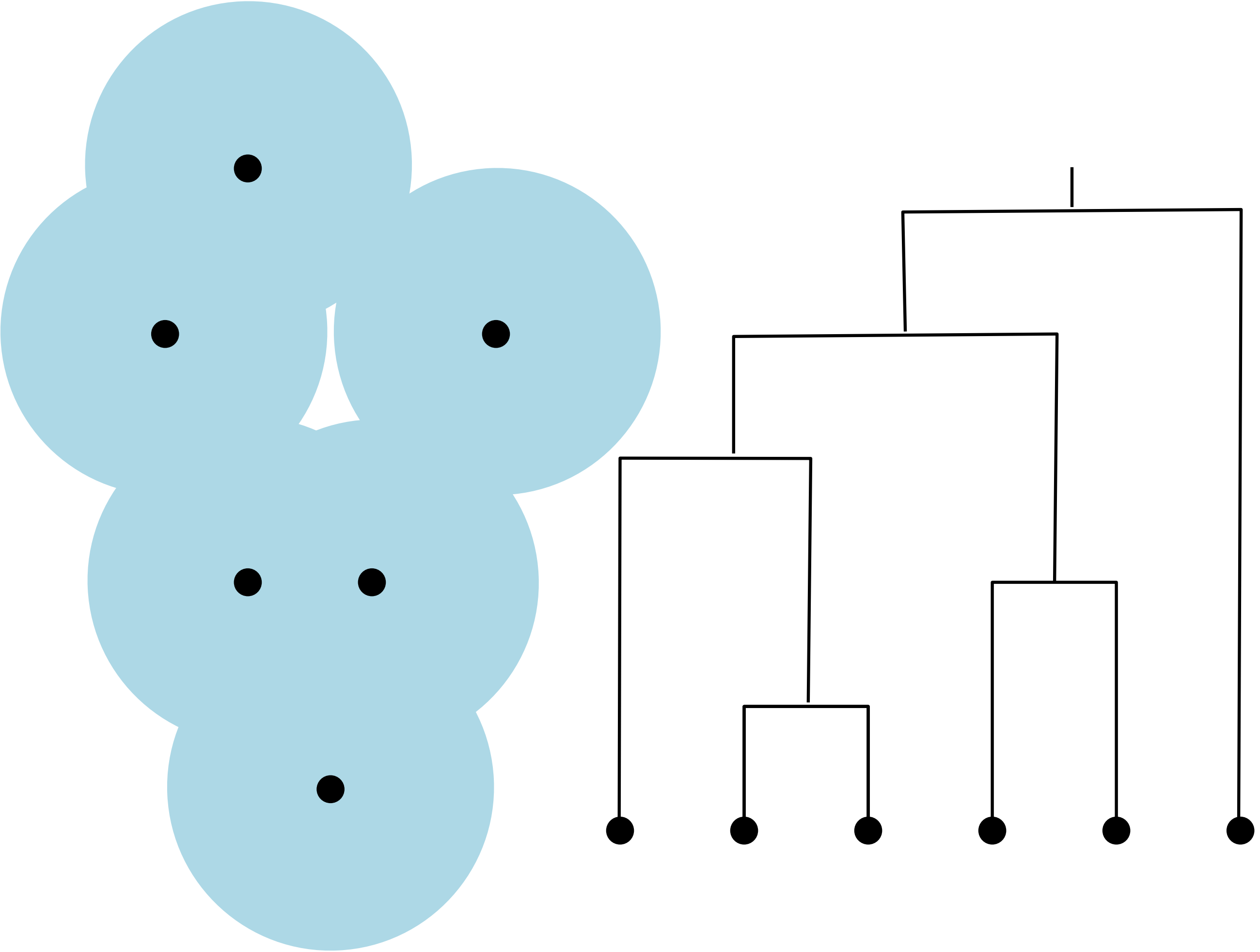
$d_{\mathcal{D}}(x, x') :=$ height of lowest common ancestor of x, x' in dendrogram $\mathcal{D}$.

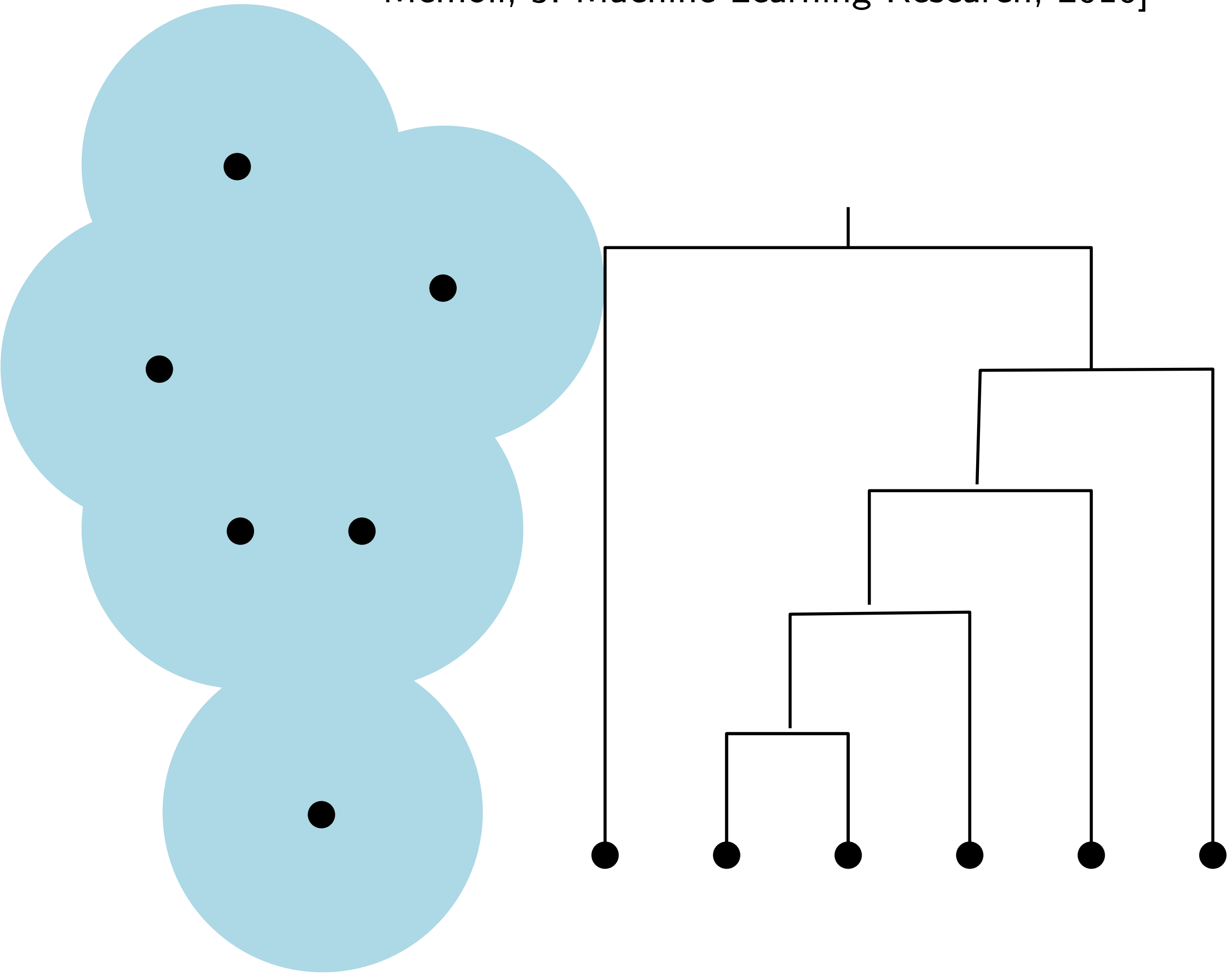
Thm: $d_{GH}((X, d_{\mathcal{D}_X}), (Y, d_{\mathcal{D}_Y})) \leq d_{GH}((X, d_X), (Y, d_Y))$.

ultrametric!

The (in)stability of dendrograms

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





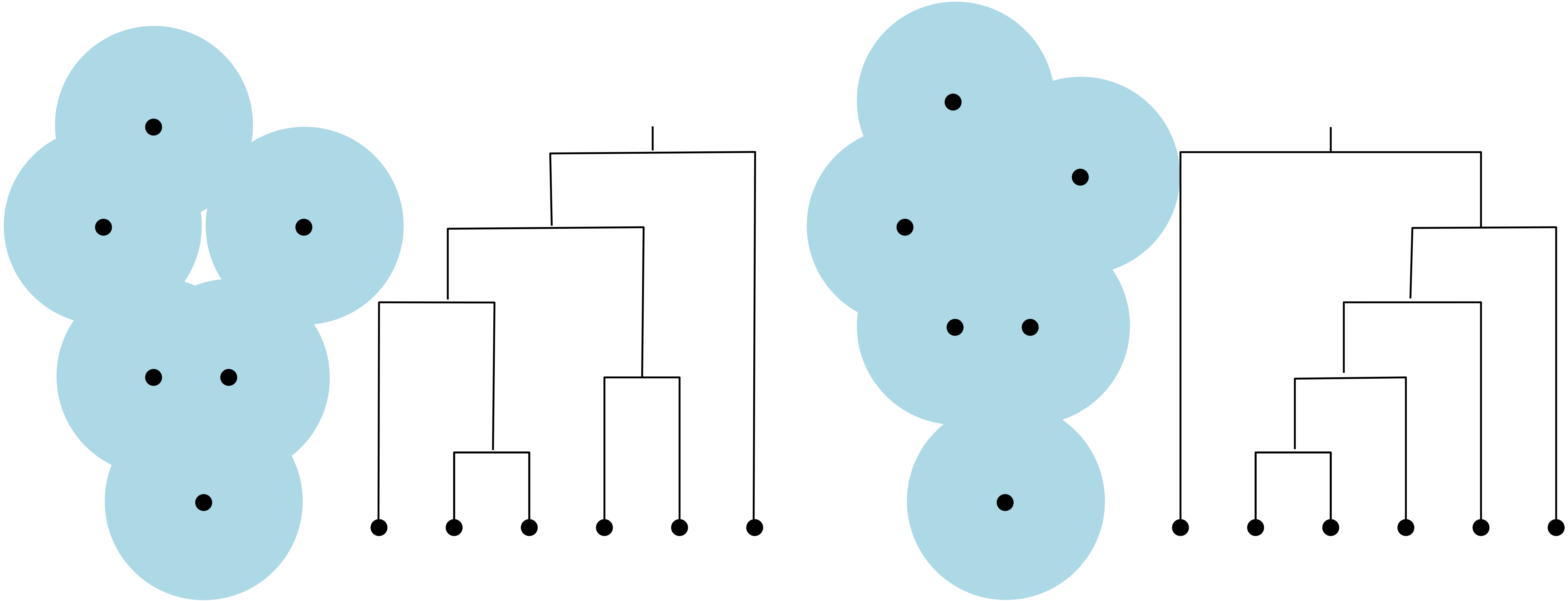
$d_{\mathcal{D}}(x, x') :=$ height of lowest common ancestor of x, x' in dendrogram $\mathcal{D}$.

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

ultrametric!

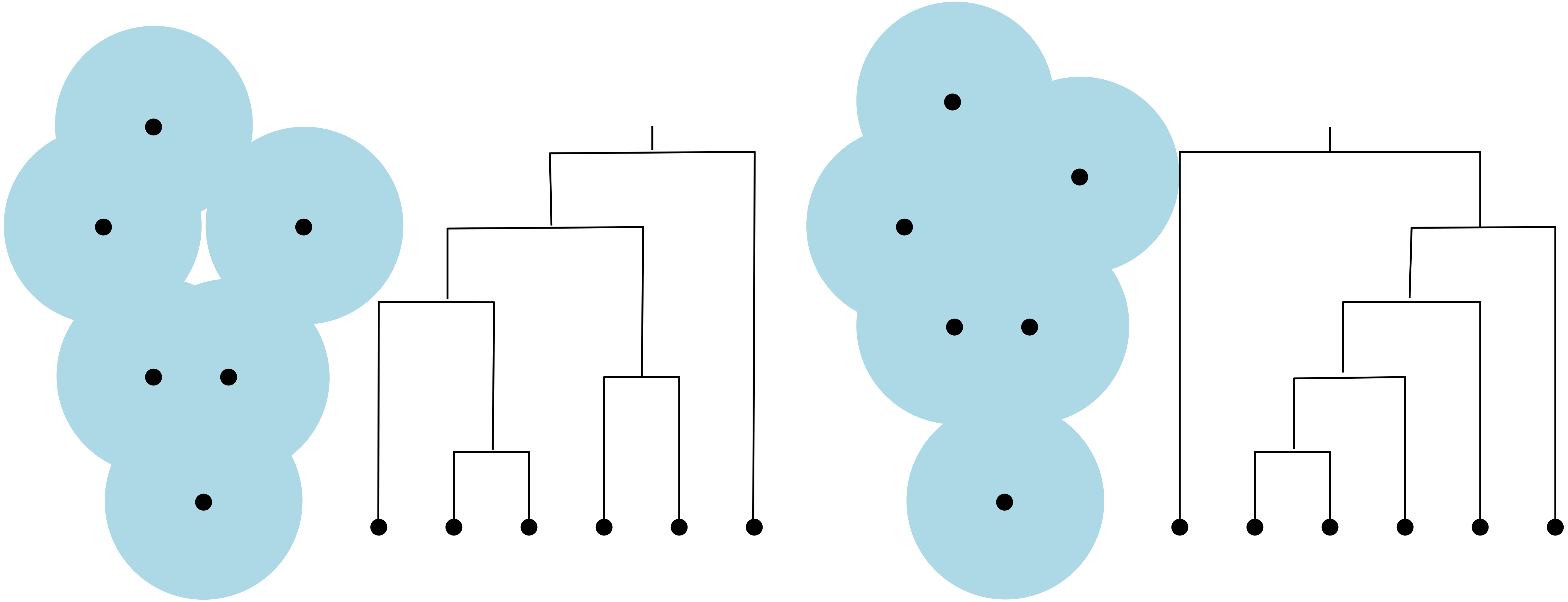
This is actually not true for complete and average clustering.

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.

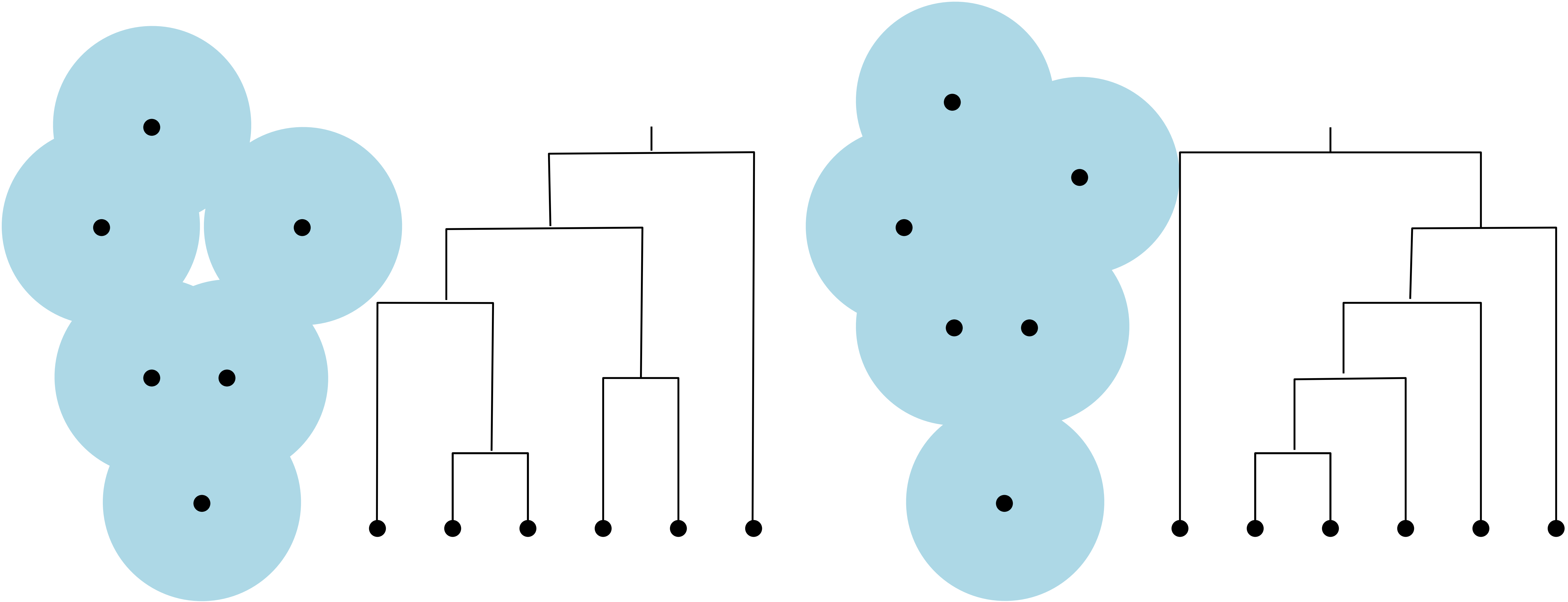
The (in)stability of dendrograms



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).

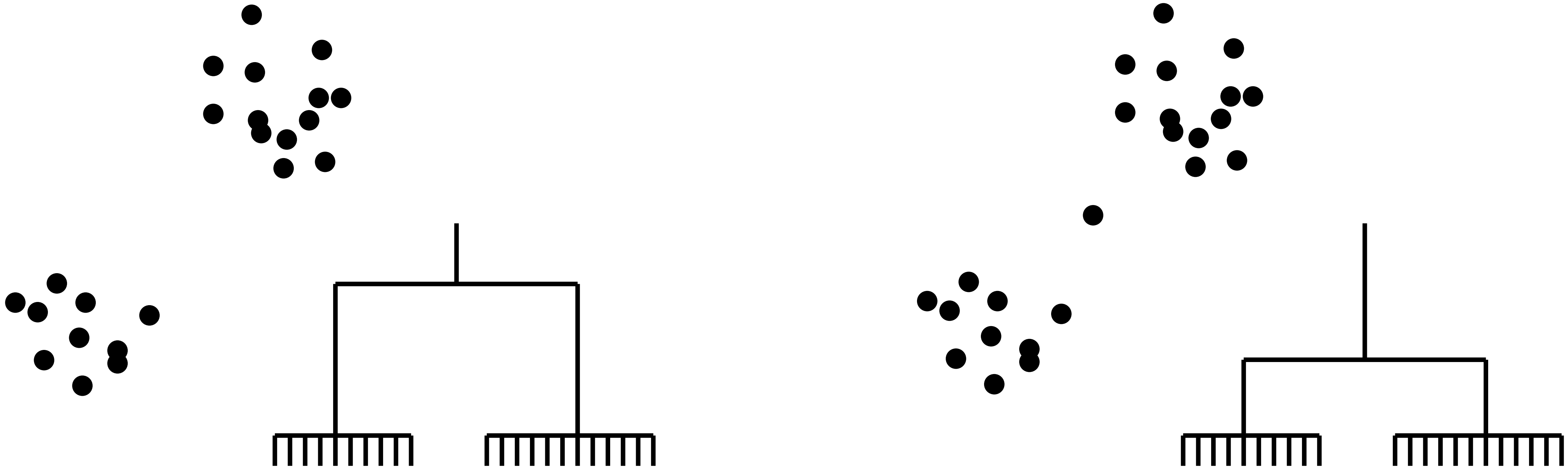
→ 0-dimensional persistent homology provides a stable output!

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.

→ 0-dimensional persistent homology provides a stable output!

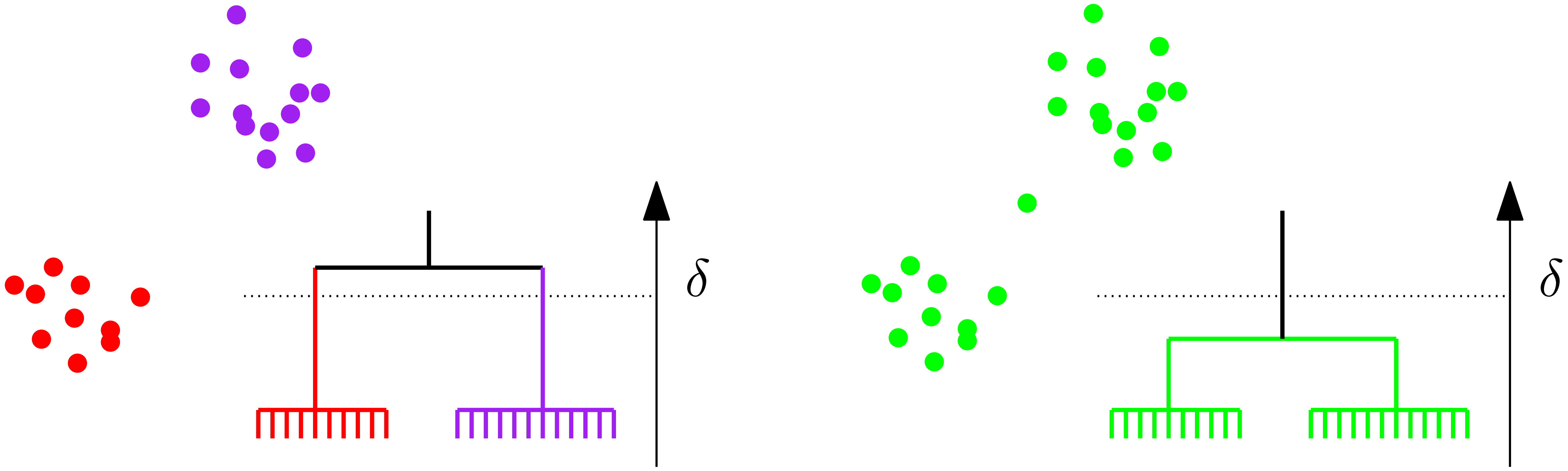
The (in)stability of dendrograms



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.

→ 0-dimensional persistent homology provides a stable output!

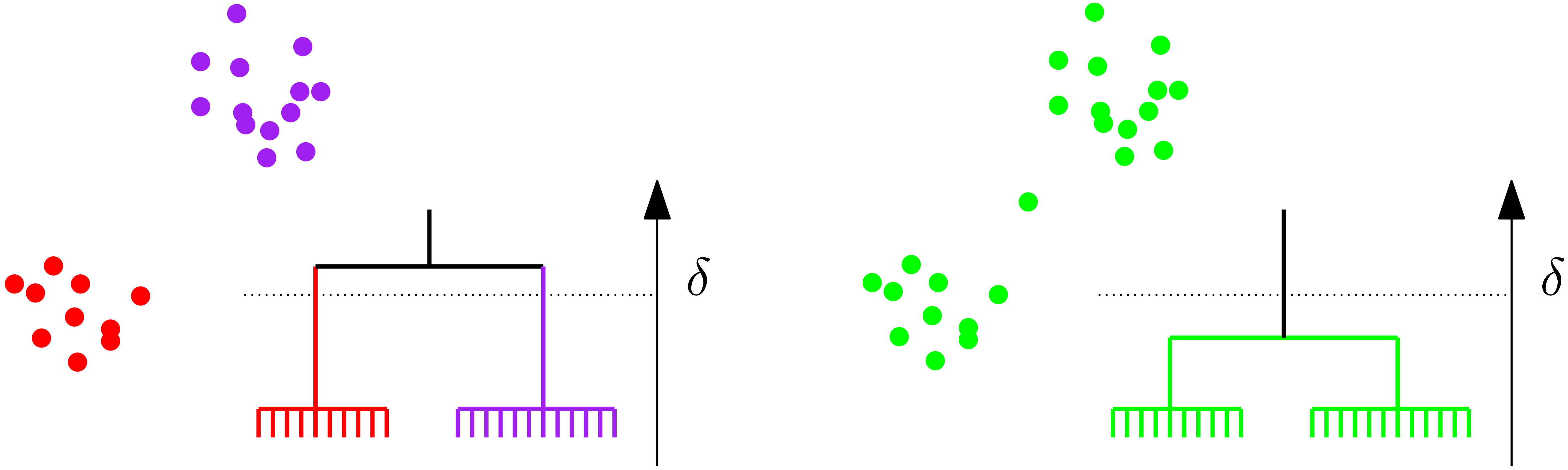
The (in)stability of dendrograms



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.

→ 0-dimensional persistent homology provides a stable output!

The (in)stability of dendrograms

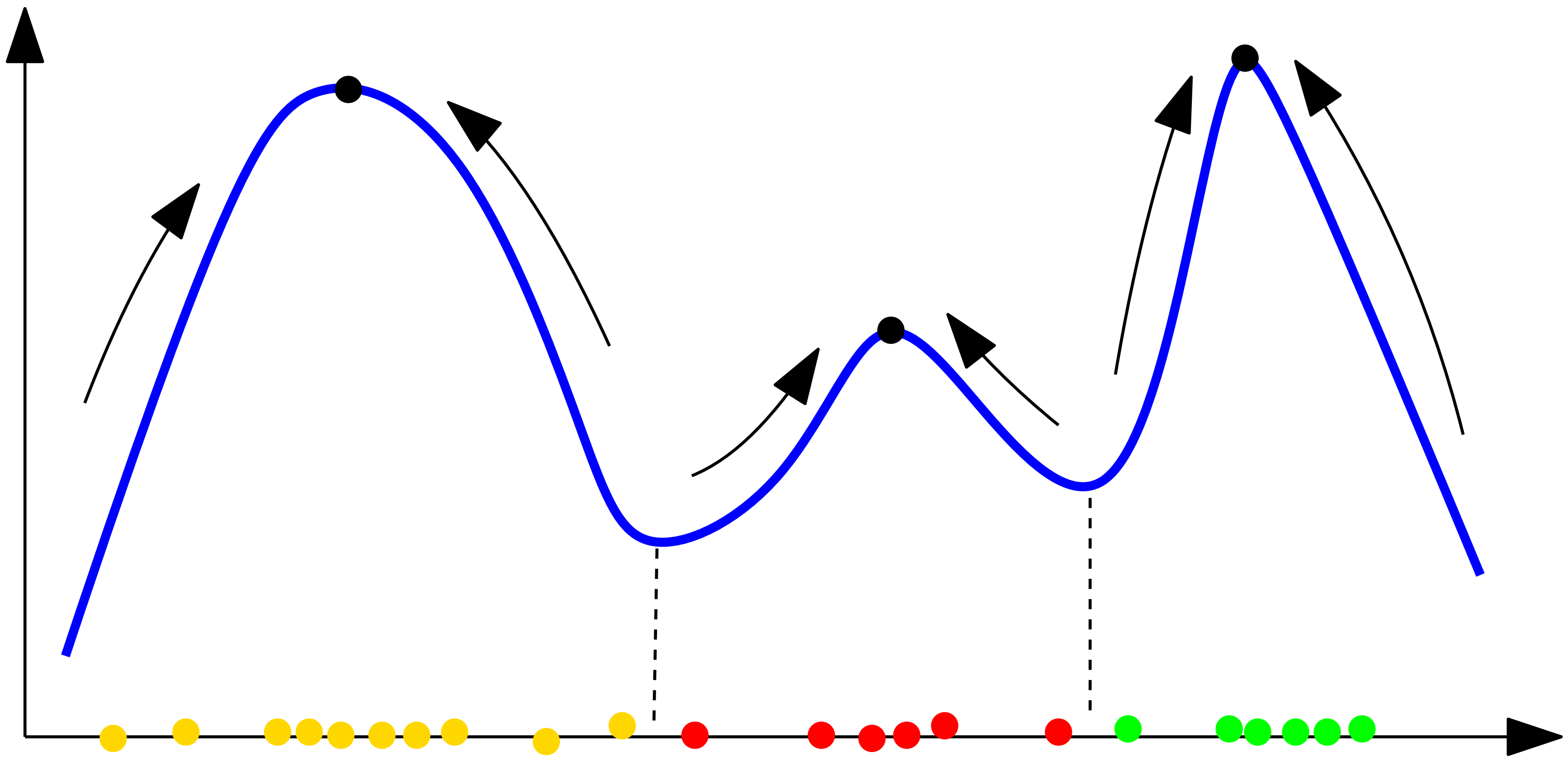


However, building a hierarchy based on spatial proximity is still not a great idea when there are outliers, since there is no stability of merging times anymore.

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.

→ 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.

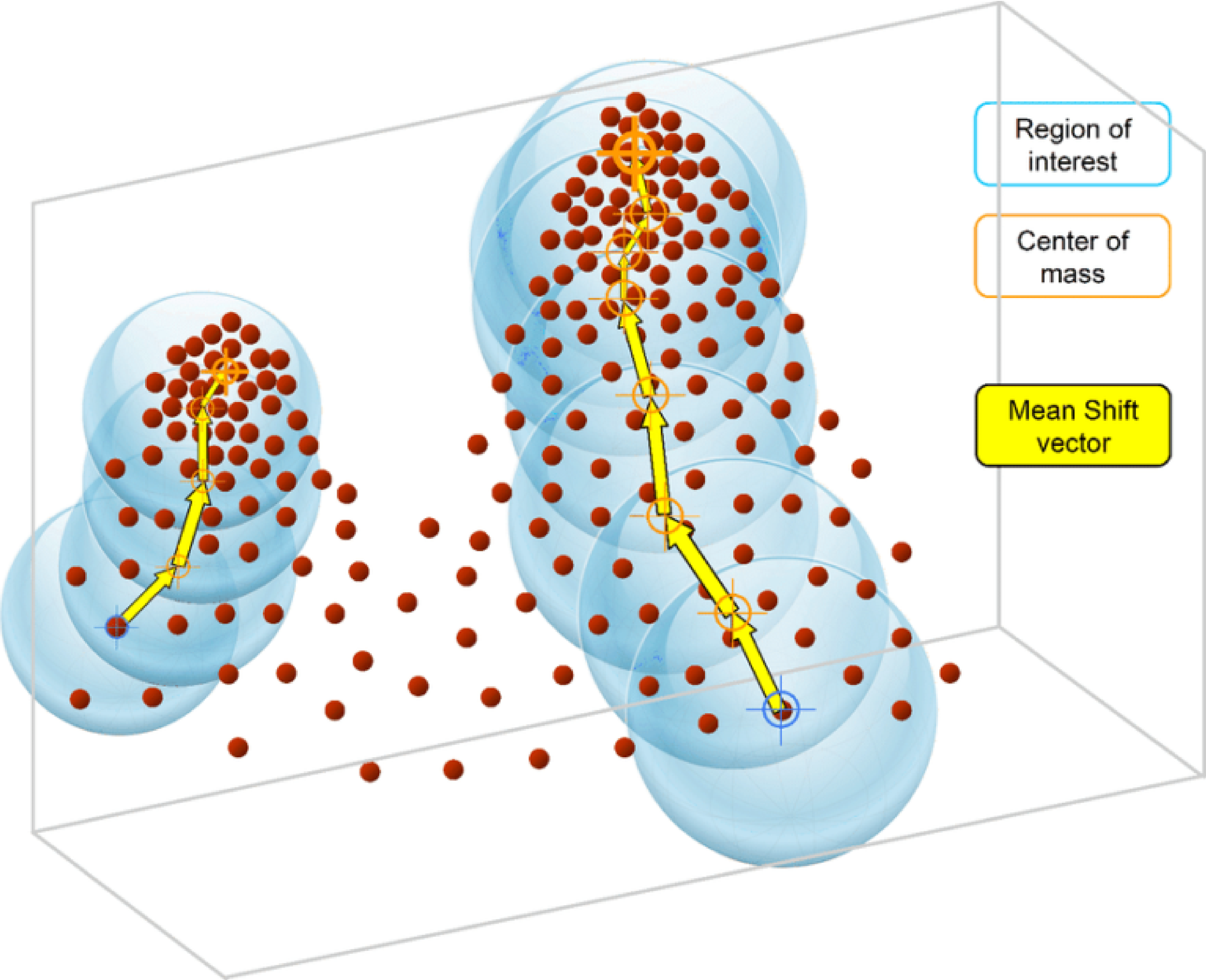
Mean Shift (2002)

1. Pick
2. Com

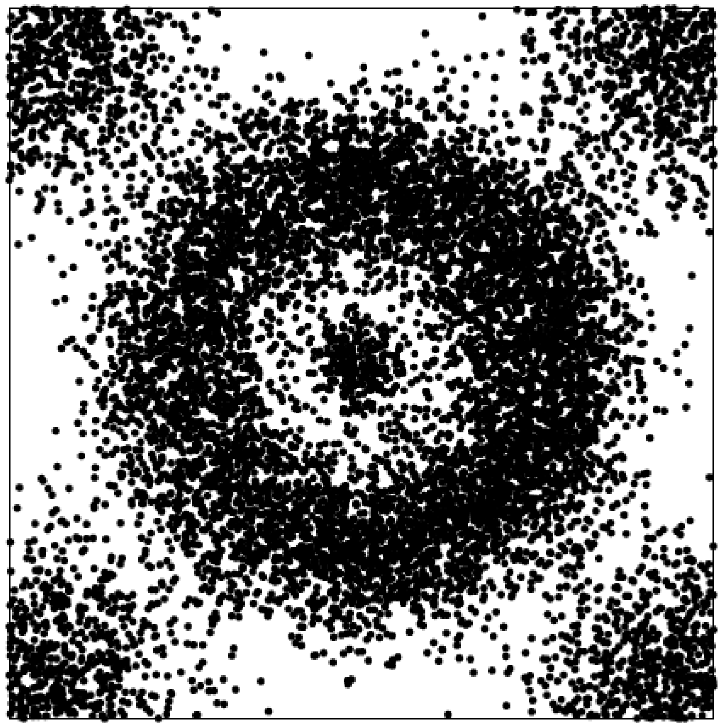
where N n kernel
$K(x, y)$

3. Upda

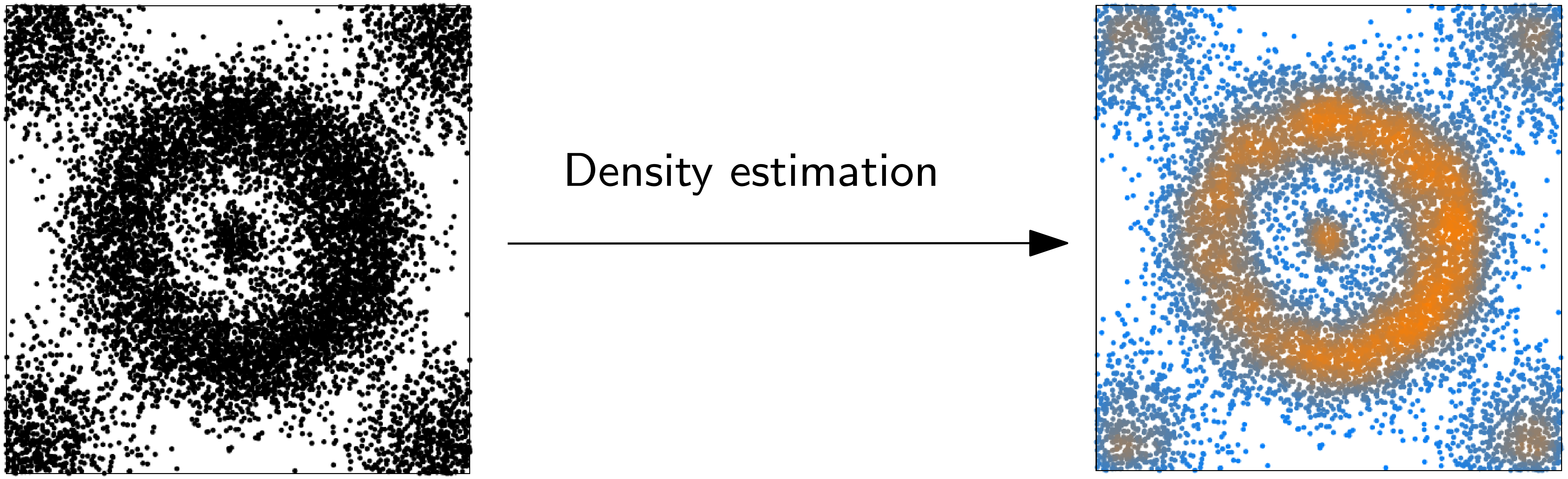
Do that ntroids,
and use



The Koonz, Narendra and Fukunaga algorithm (1976)

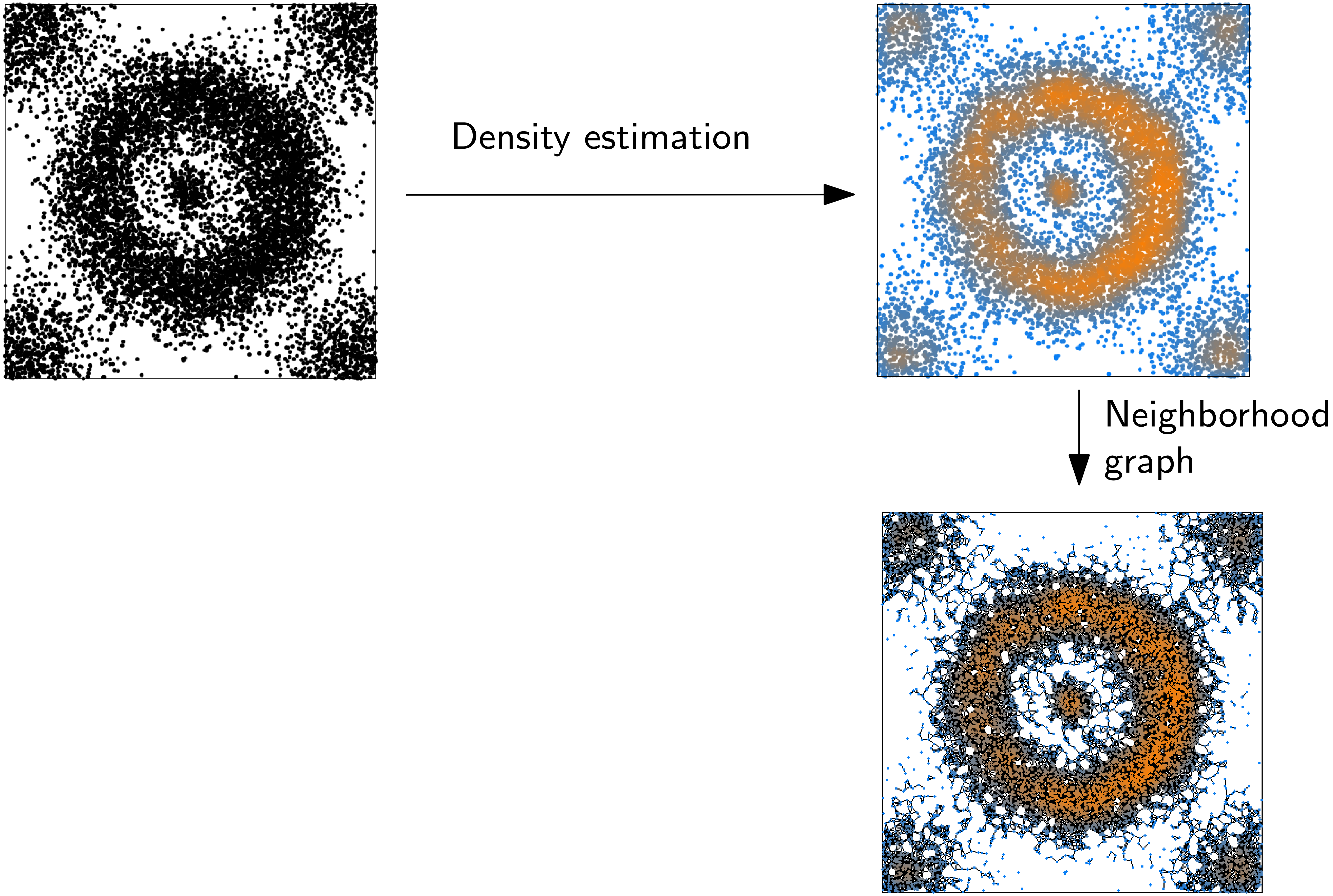


The Koonz, Narendra and Fukunaga algorithm (1976)



Density estimation

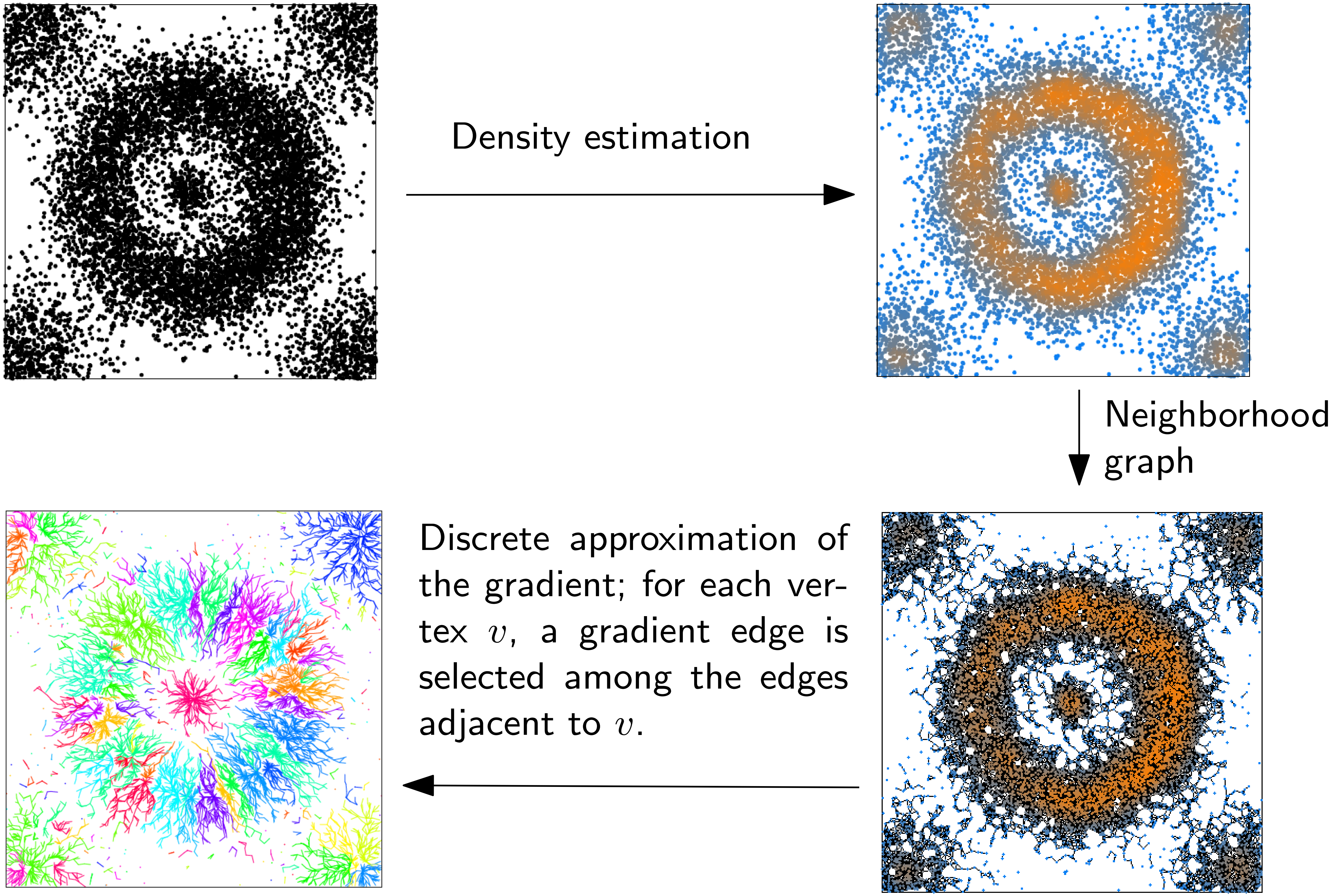
The Koonz, Narendra and Fukunaga algorithm (1976)



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:

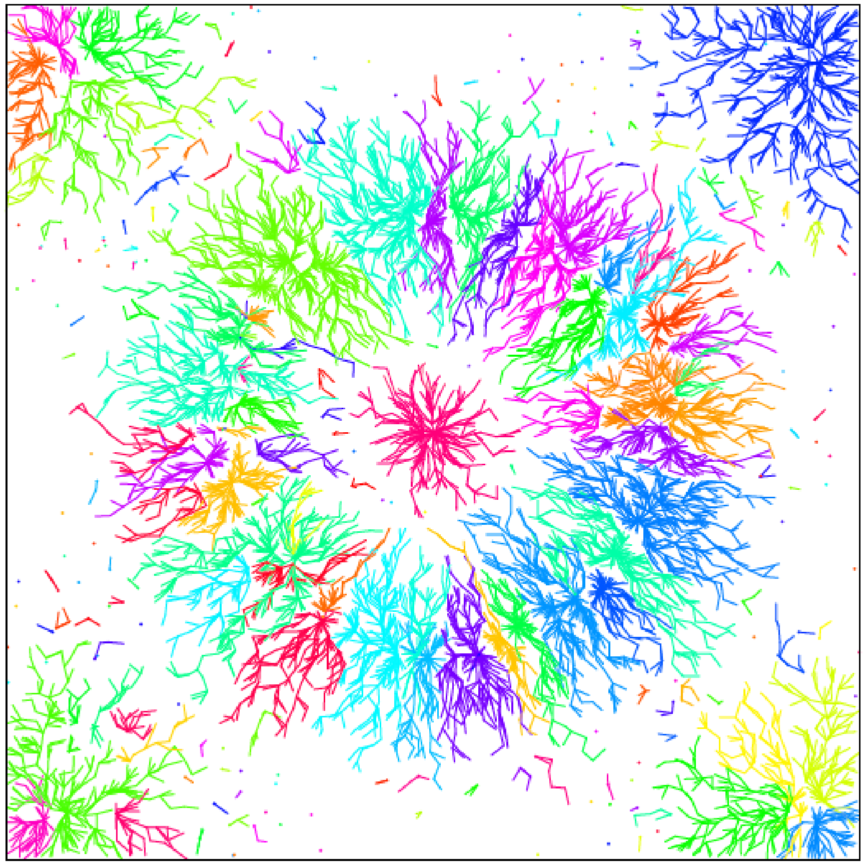
Input: A neighborhood graph G with n vertices (the data points) and an n-dimensional vector $\hat{f}$ (density estimate).

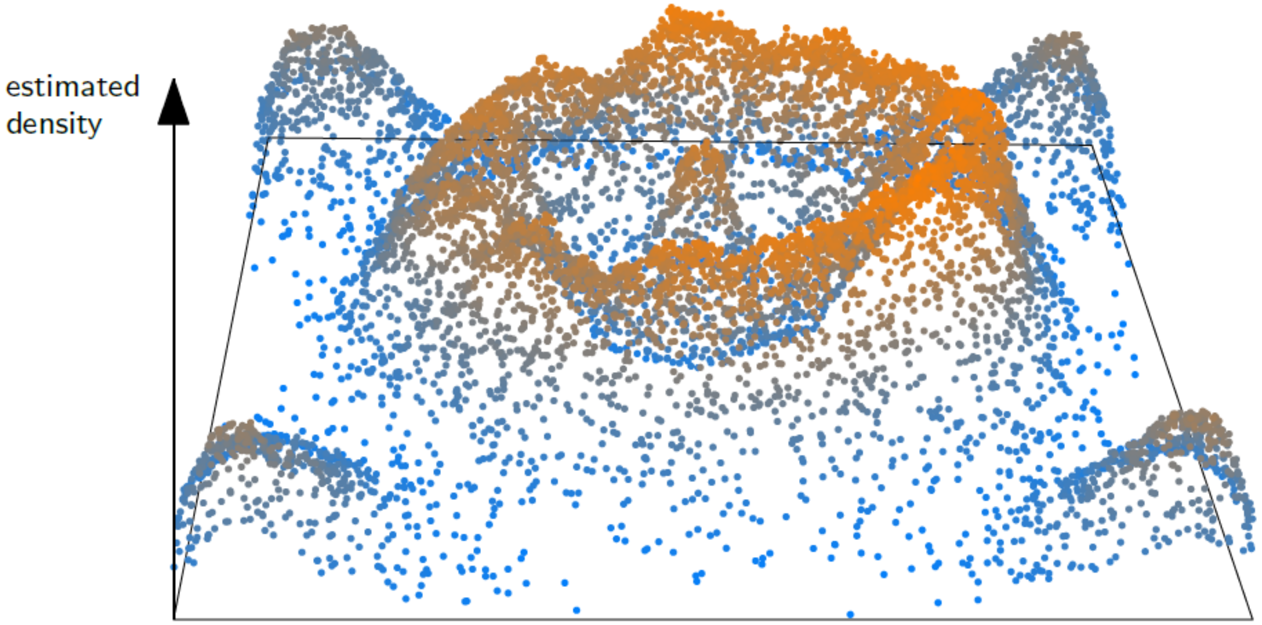
```
Sort the vertex indices $\{1, 2, \ldots, n\}$ in decreasing order: $\hat{f}(1) \geq \cdots \geq \hat{f}(n)$.
Initialize a union-find data structure $\mathcal{U}$ and two lists $g, r$ of length $n$.

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:
        $g[i] \leftarrow \mathrm{argmax}\{\hat{f}(j) : j \in \mathcal{N}\}$ ($g[i]$ stores the approximate gradient at vertex $i$)
        $e_i \leftarrow \mathcal{U}.\mathtt{Find}(g[i])$
        Attach vertex $i$ to the entry $e_i$: $\mathcal{U}.\mathtt{Union}(i, e_i)$
```

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

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

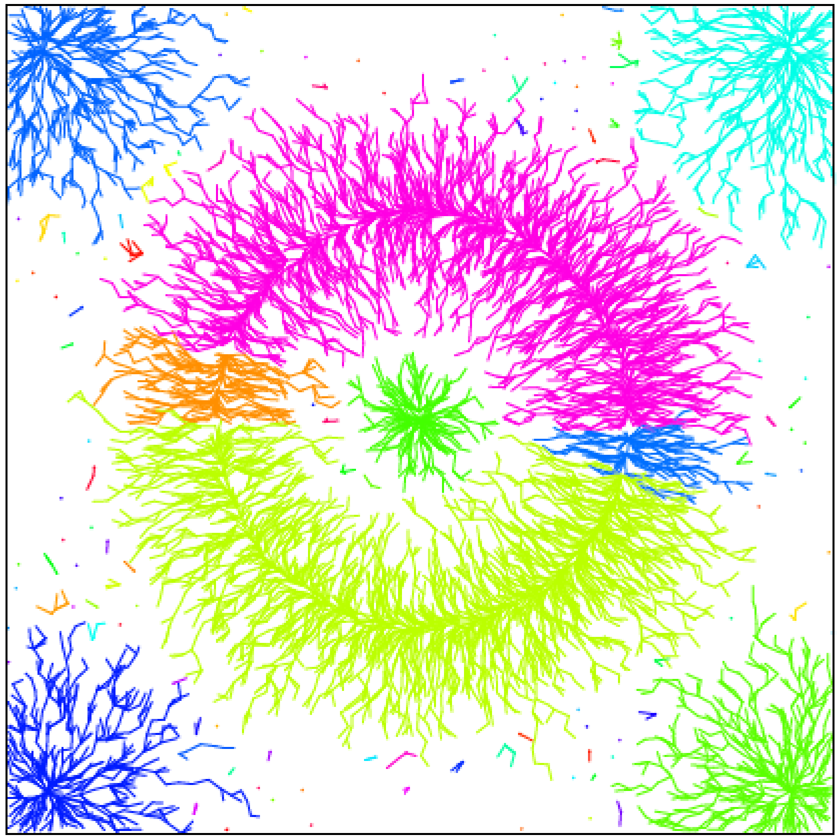


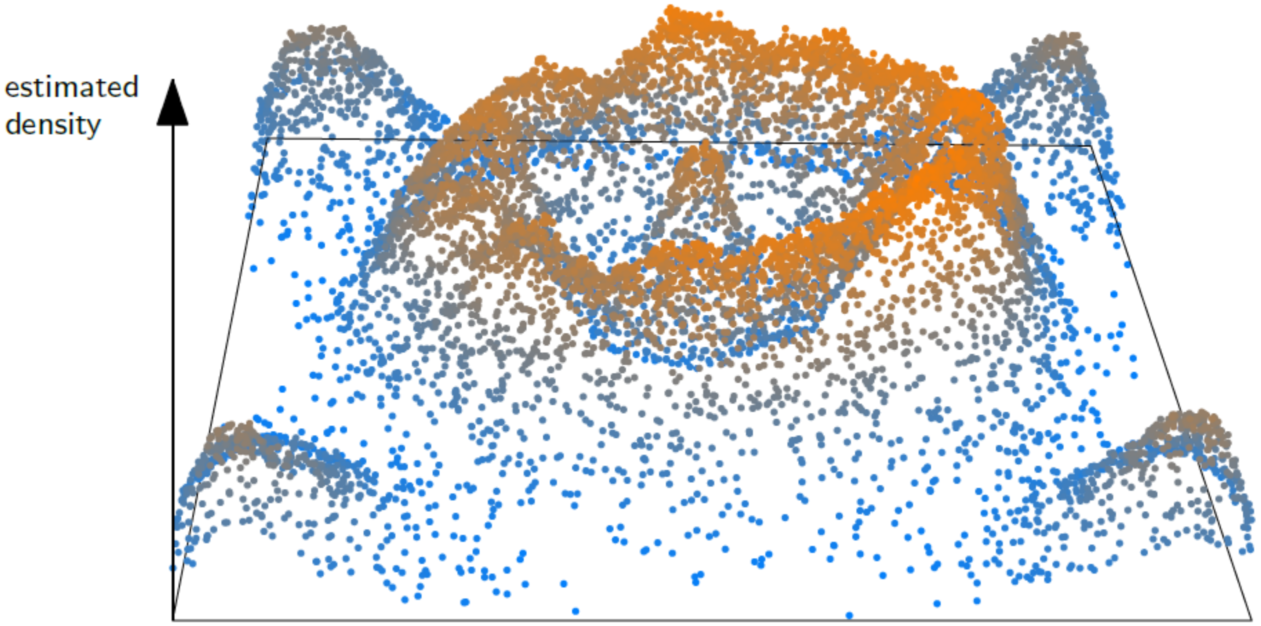


One has as many clusters as local maxima of the density estimate, which are very sensitive to noise and outliers.

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.

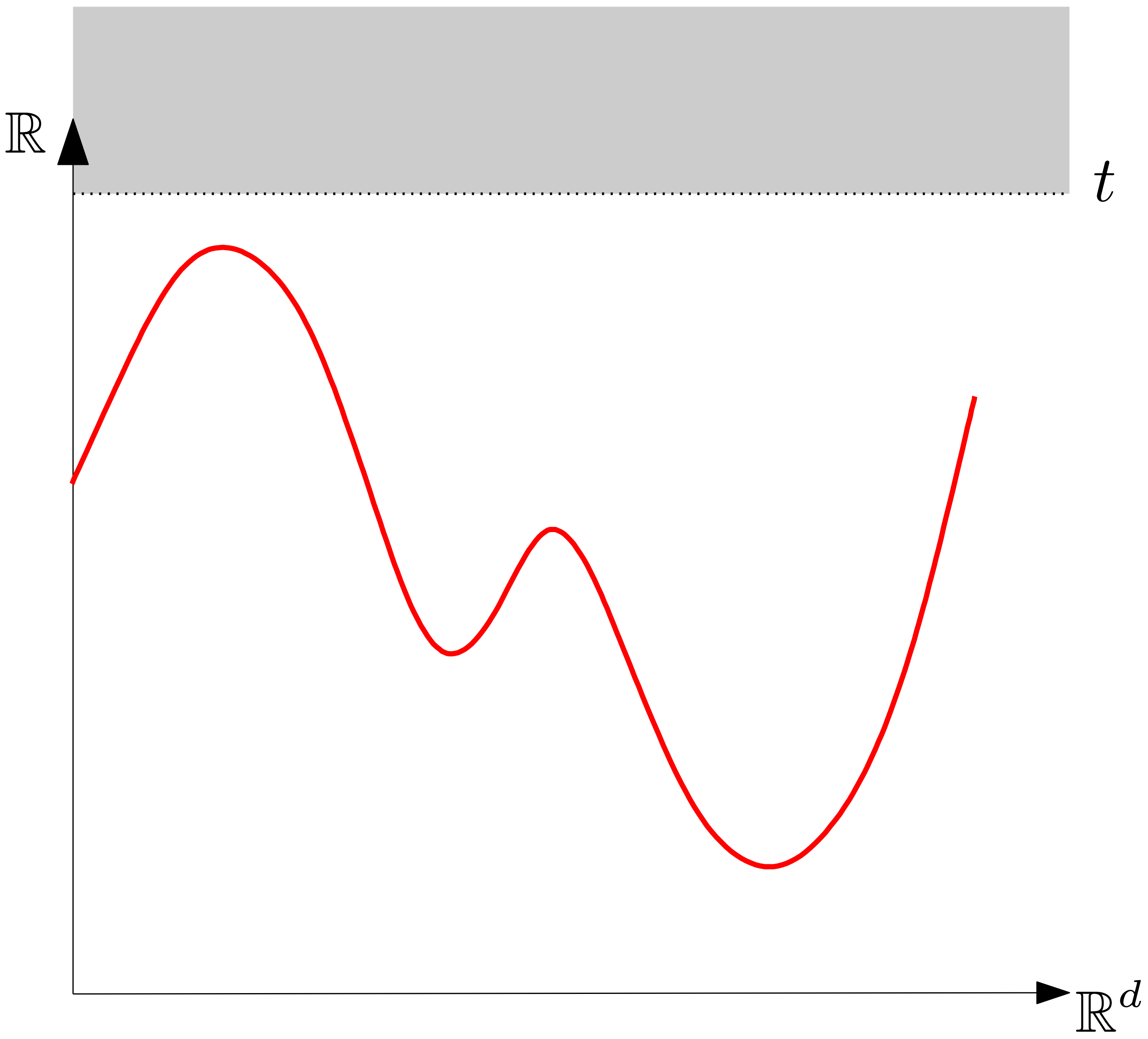
Build a hierarchy of clusters with 0-dimensional persistent homology!

Topological Machine Learning (II): Guiding ML models

1. Hierarchical and Mode Seeking Clustering
2. Topology-based Clustering
3. Topology-based Optimization

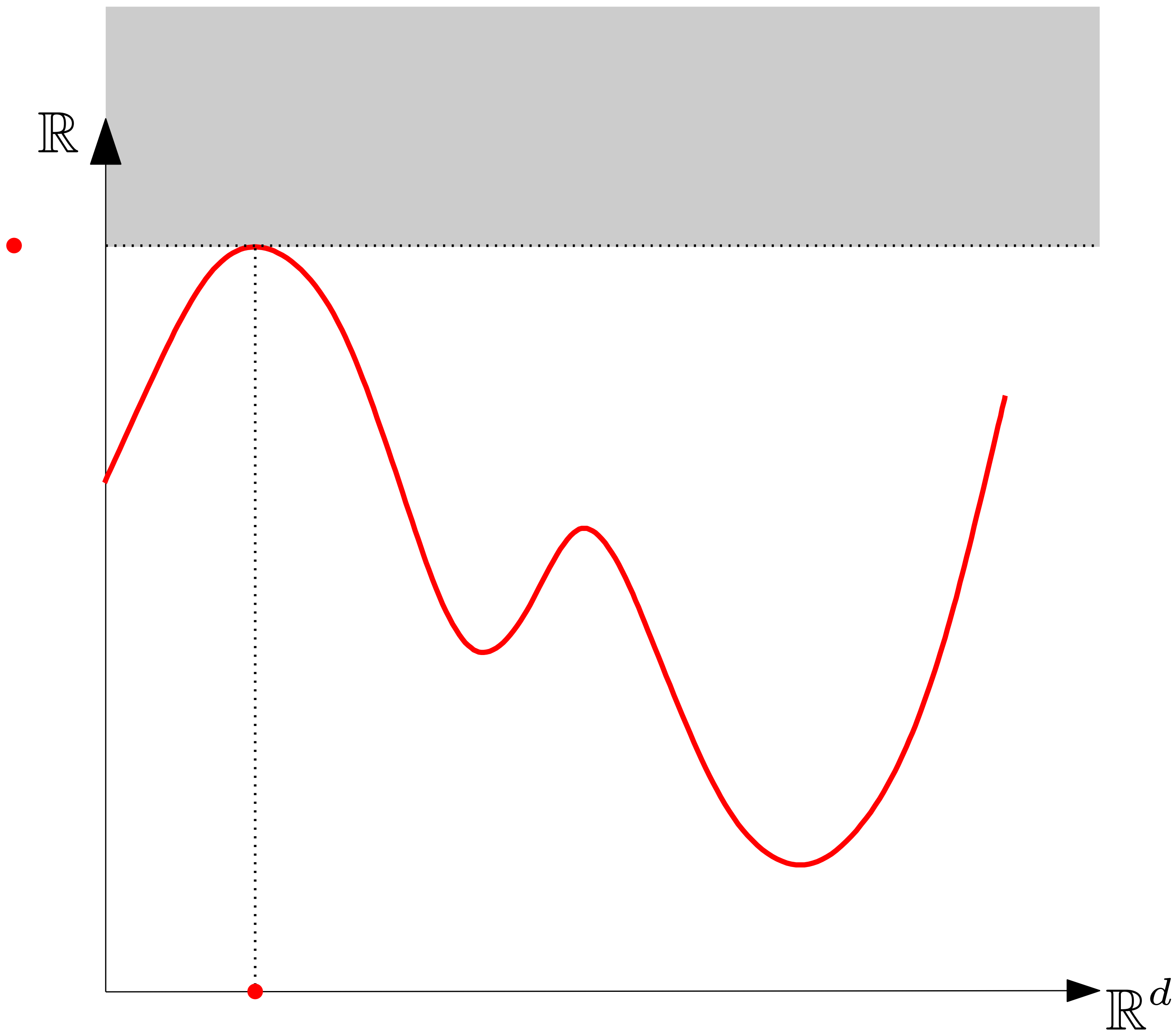
Reminder: 0-dimensional PH of density

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.



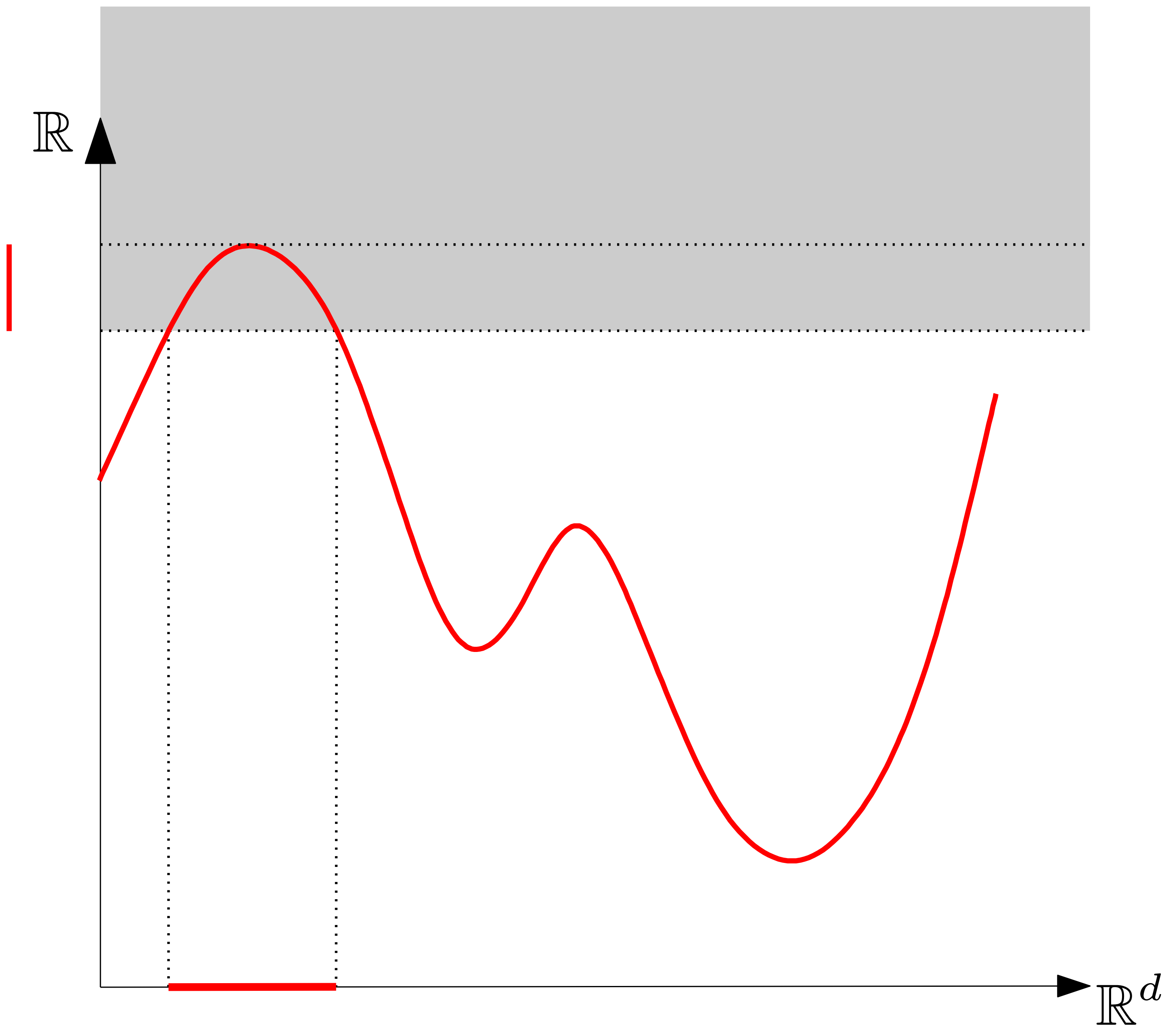
Reminder: 0-dimensional PH of density

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.



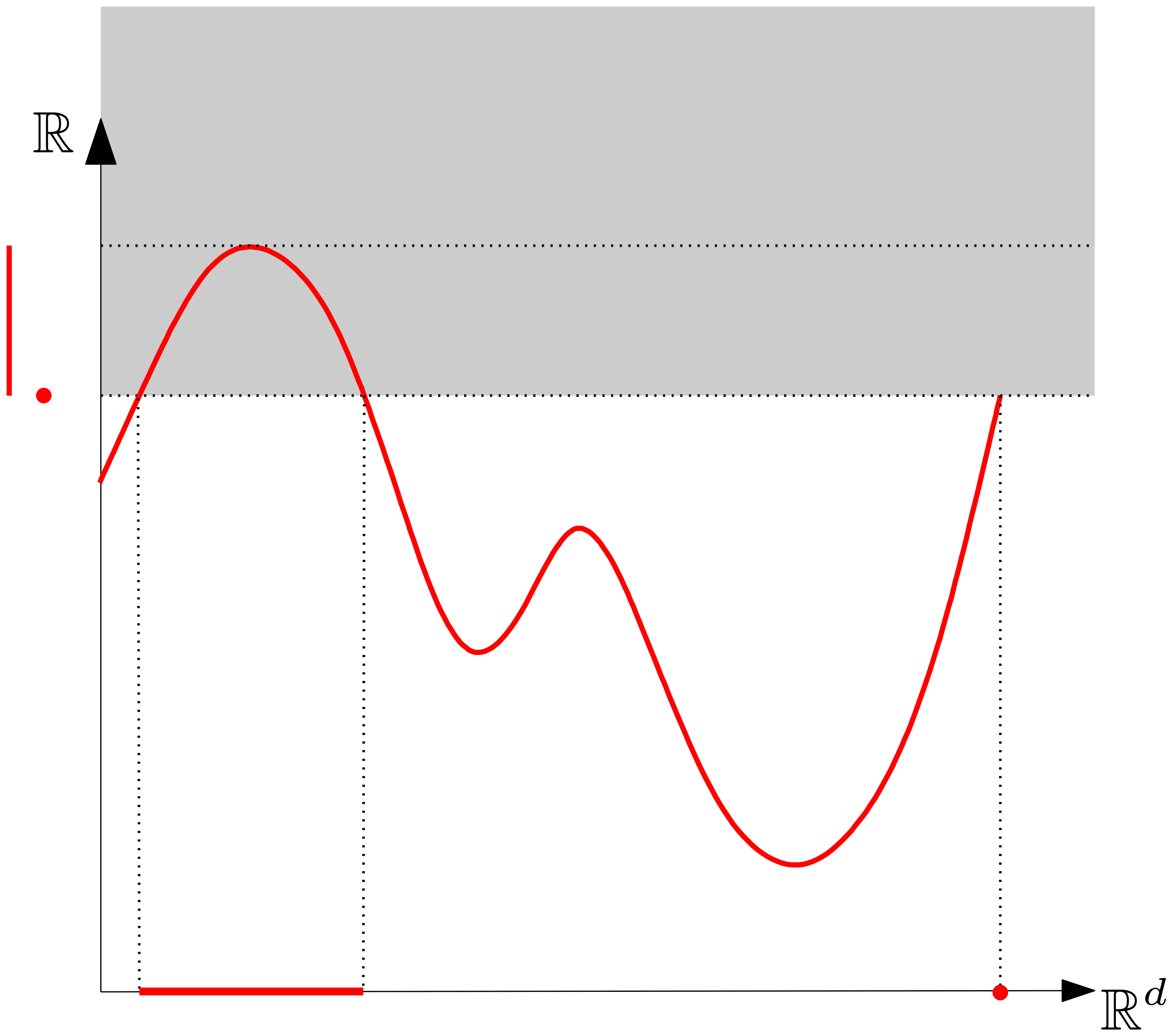
Reminder: 0-dimensional PH of density

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.



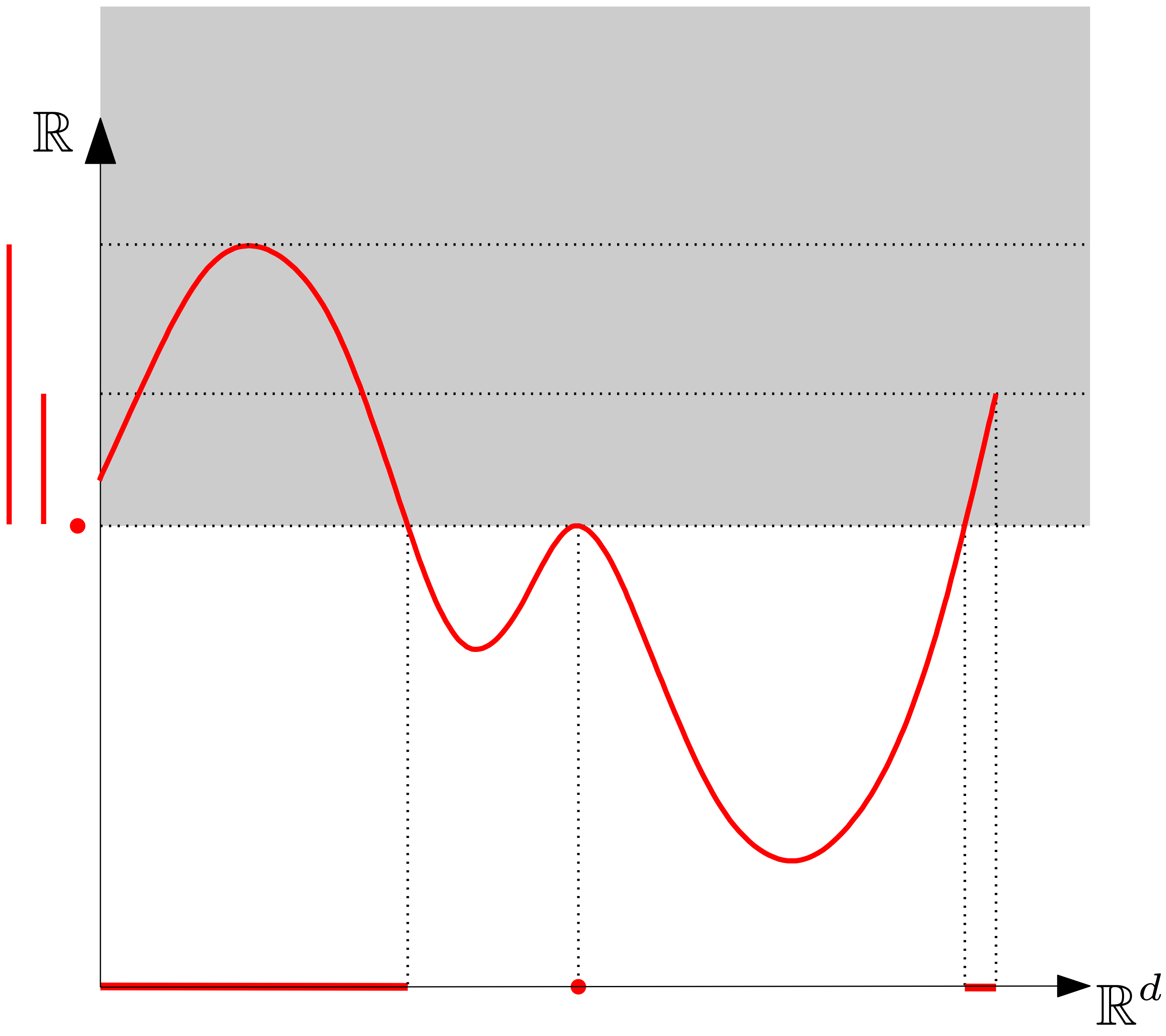
Reminder: 0-dimensional PH of density

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.



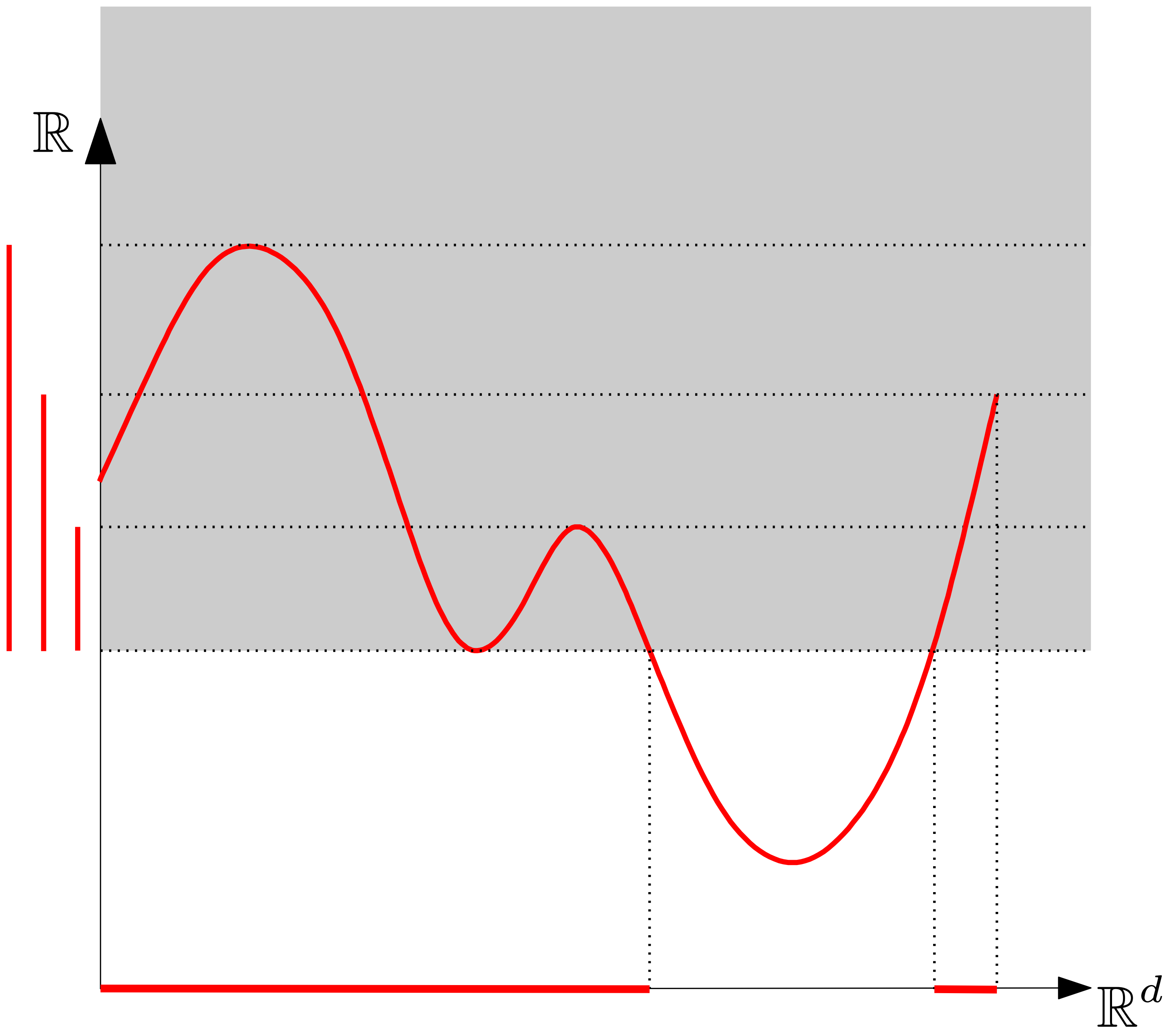
Reminder: 0-dimensional PH of density

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.



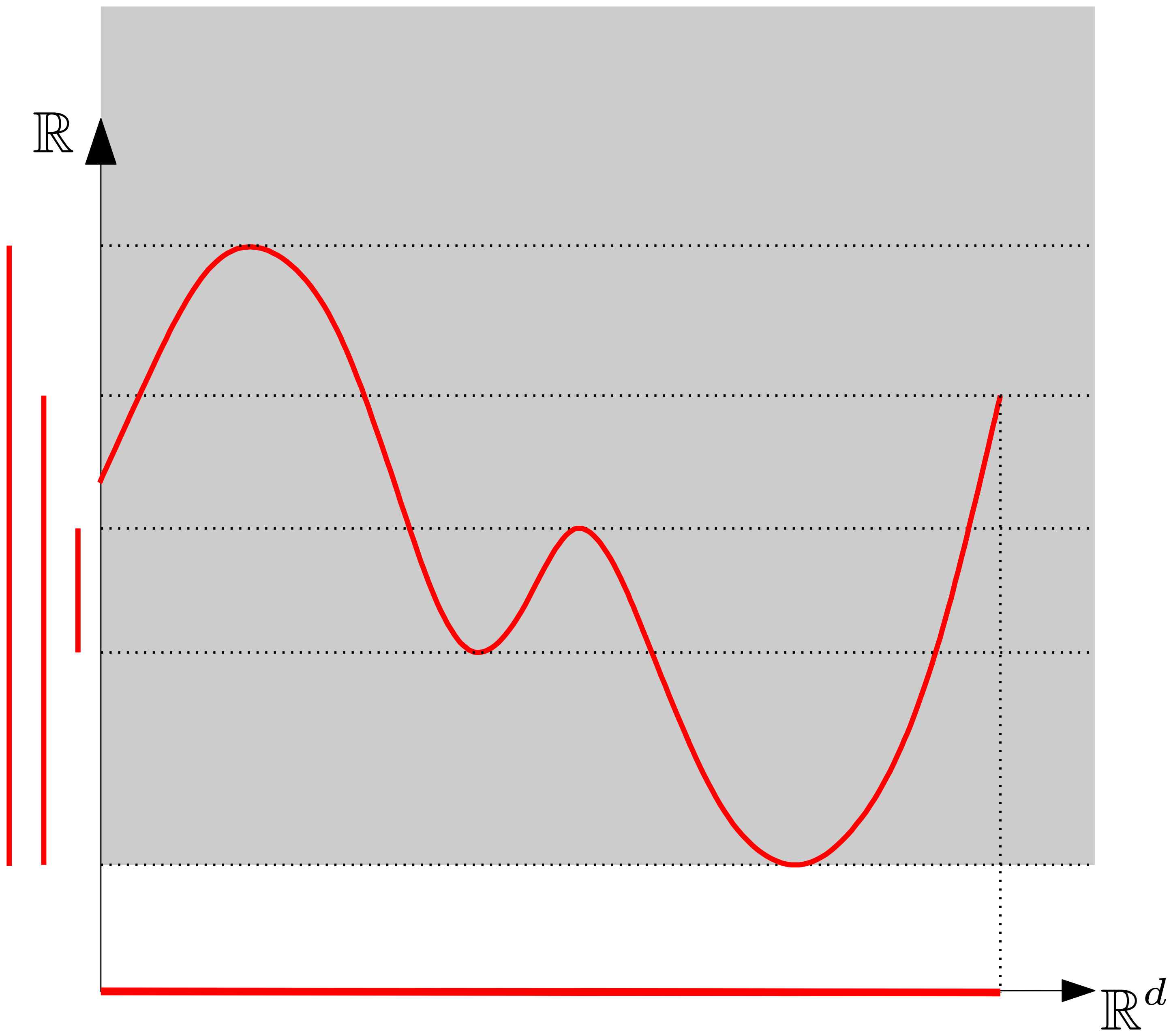
Reminder: 0-dimensional PH of density

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.



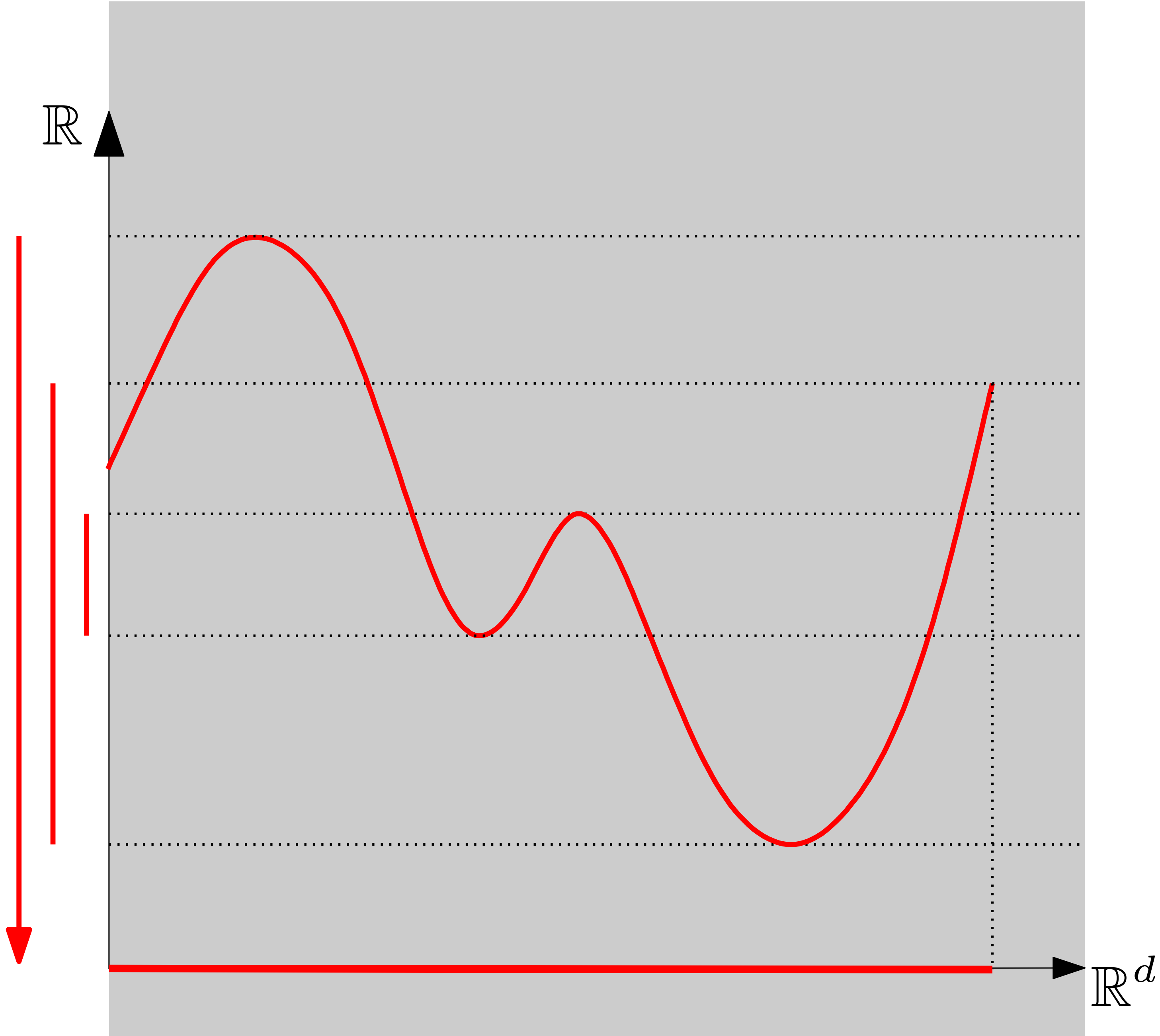
Reminder: 0-dimensional PH of density

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.



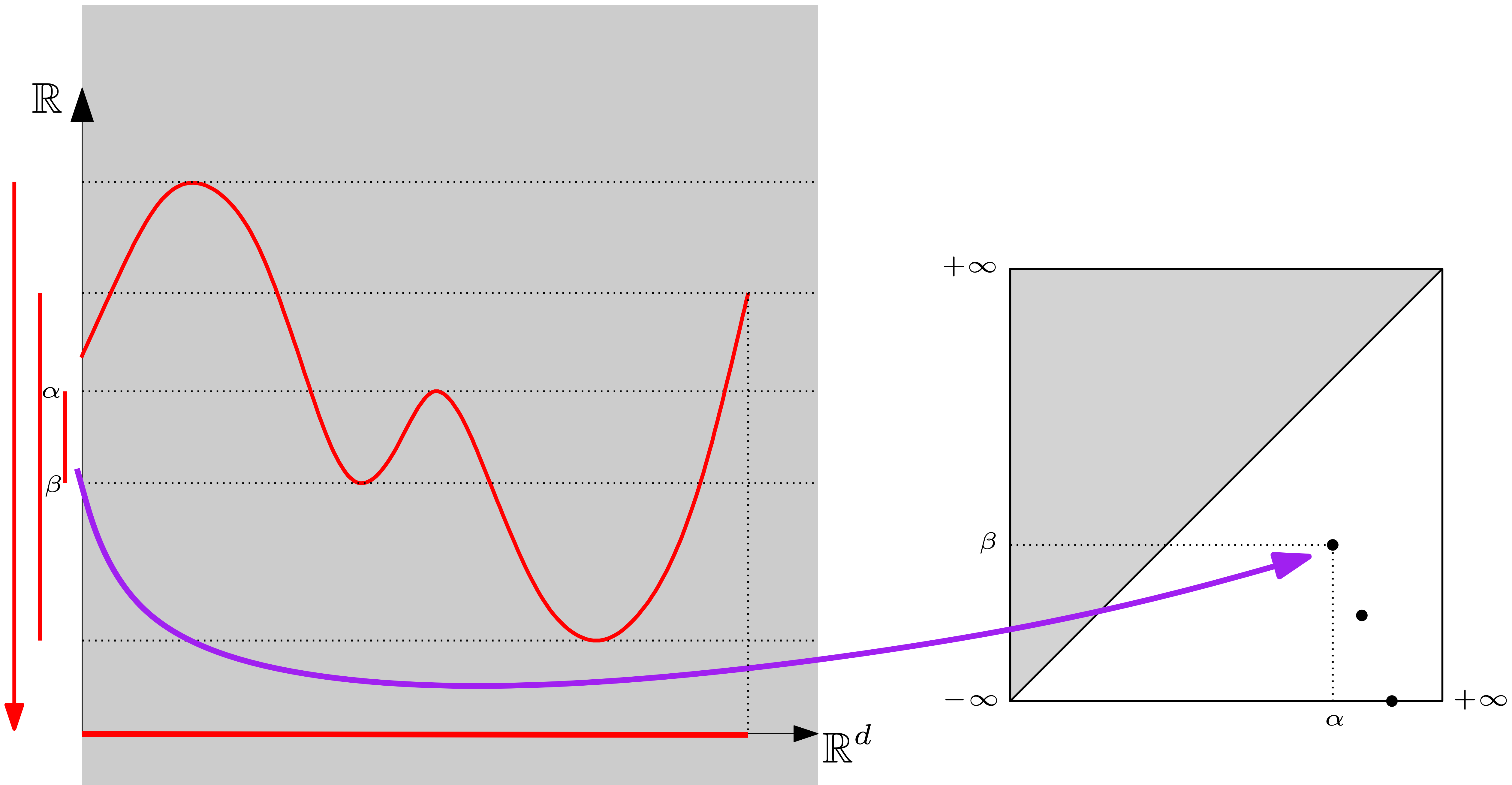
Reminder: 0-dimensional PH of density

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

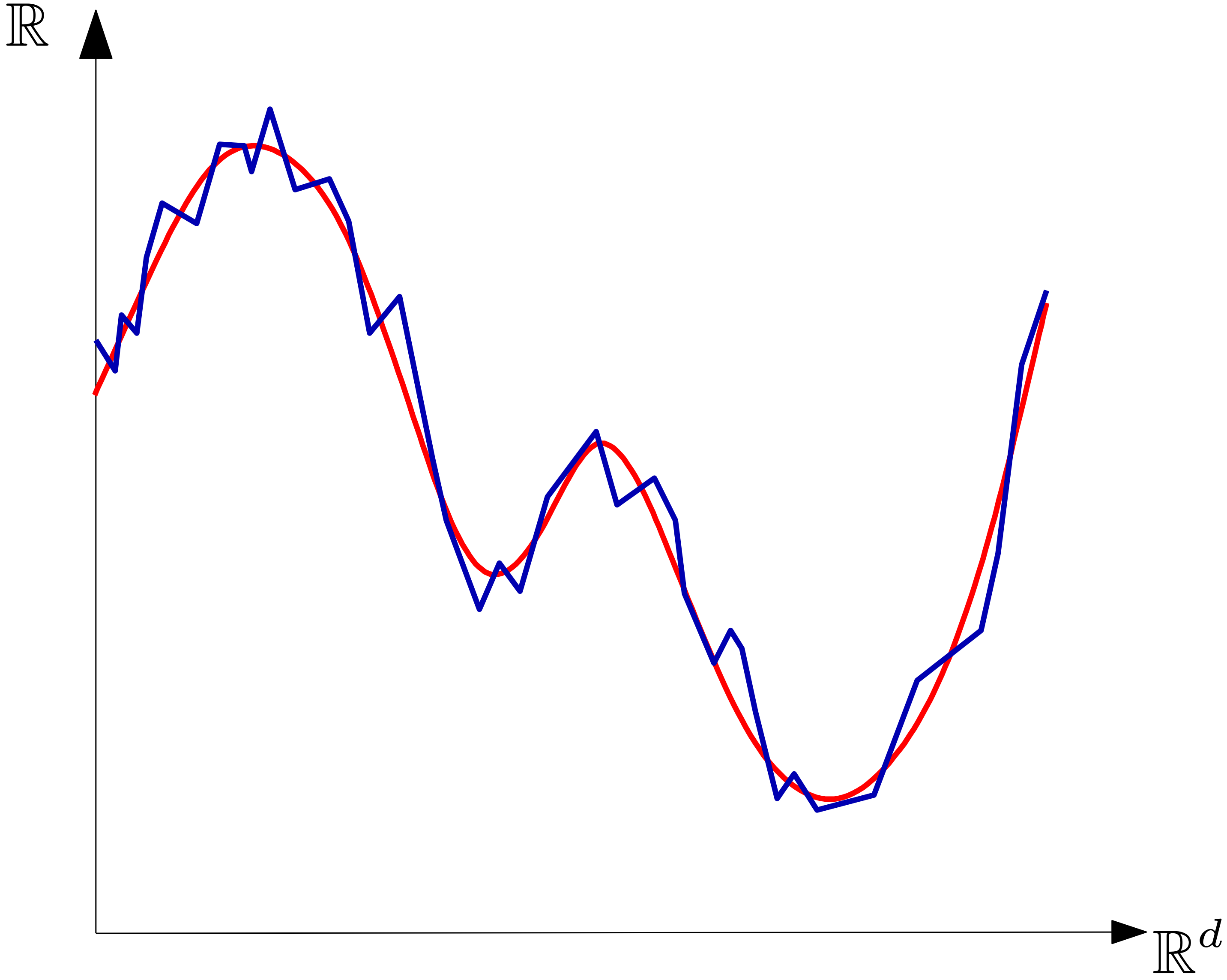
Given a probability density f, we will consider the superlevel-set filtration $f^{-1}([t,+\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-set filtration.

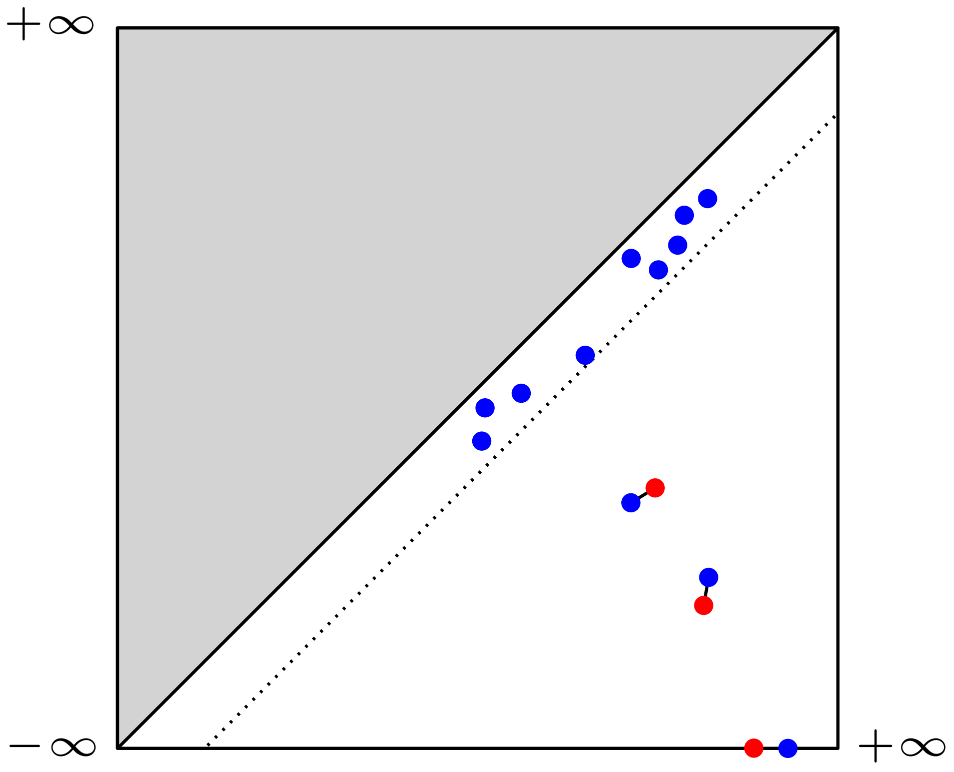


Reminder: 0-dimensional PH of density

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

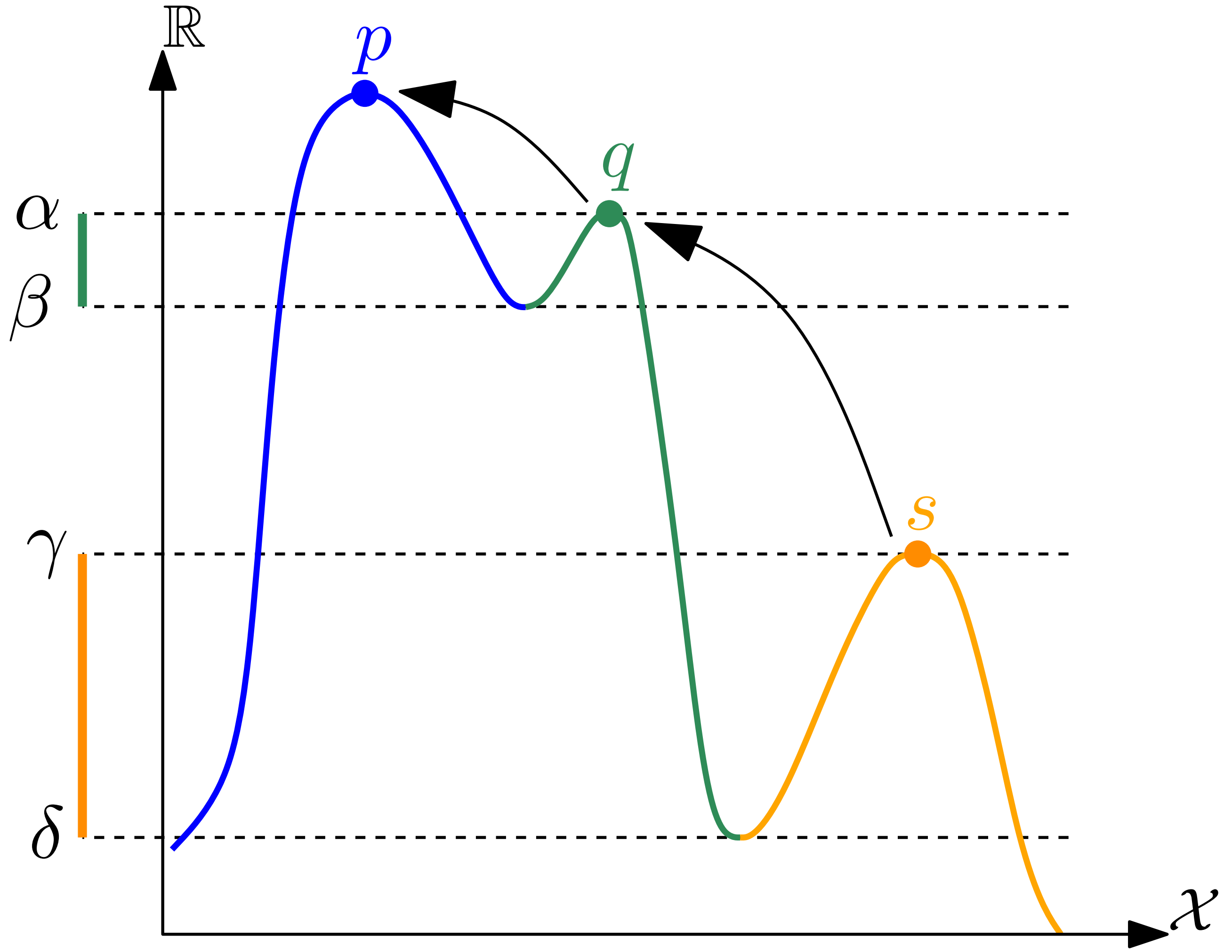
$$d_b(D_f, D_{\hat{f}}) \leq \|f - \hat{f}\|_\infty.$$





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.

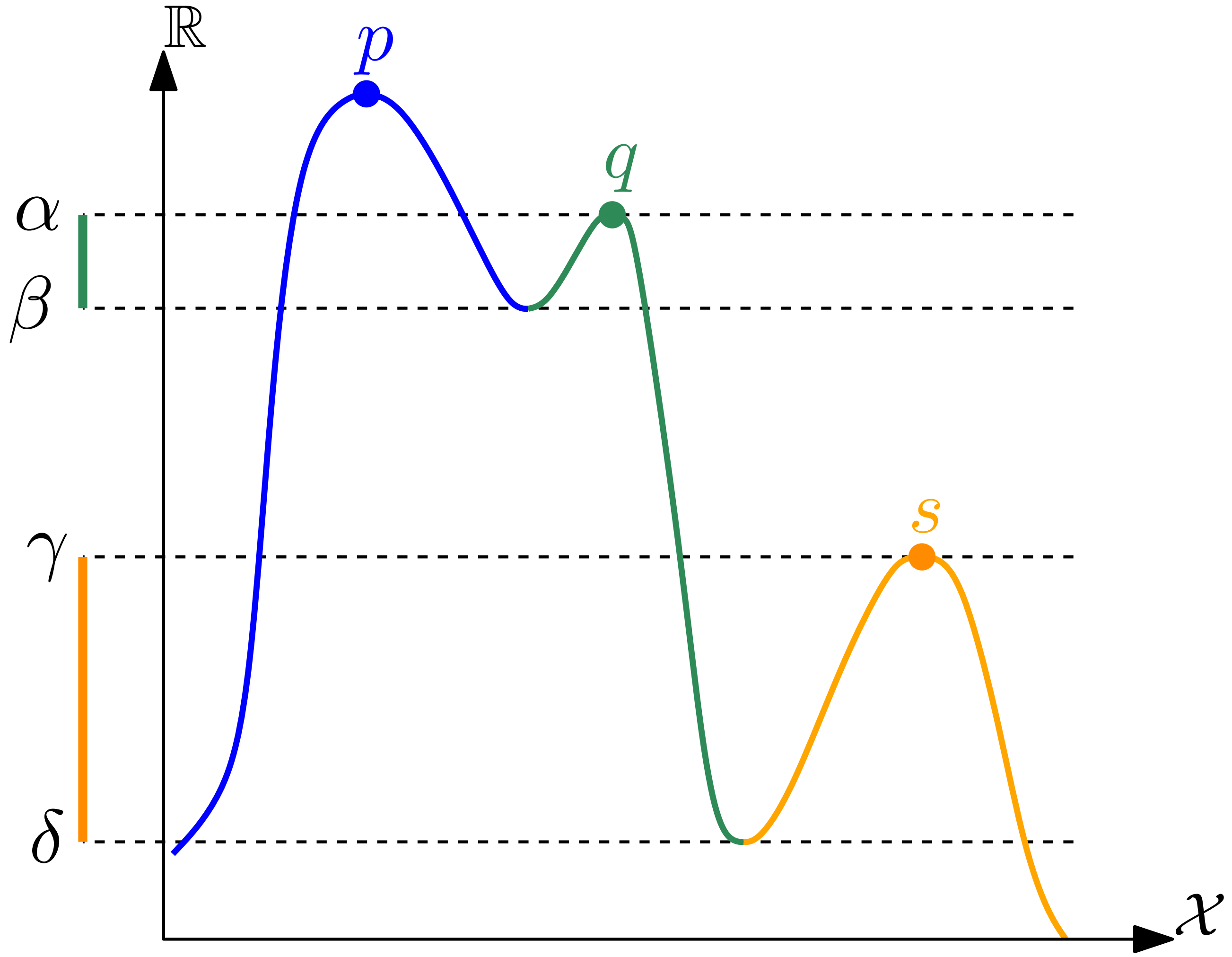


Building a hierarchy of cluster with 0-dimensional PH

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) $> \tau$!

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

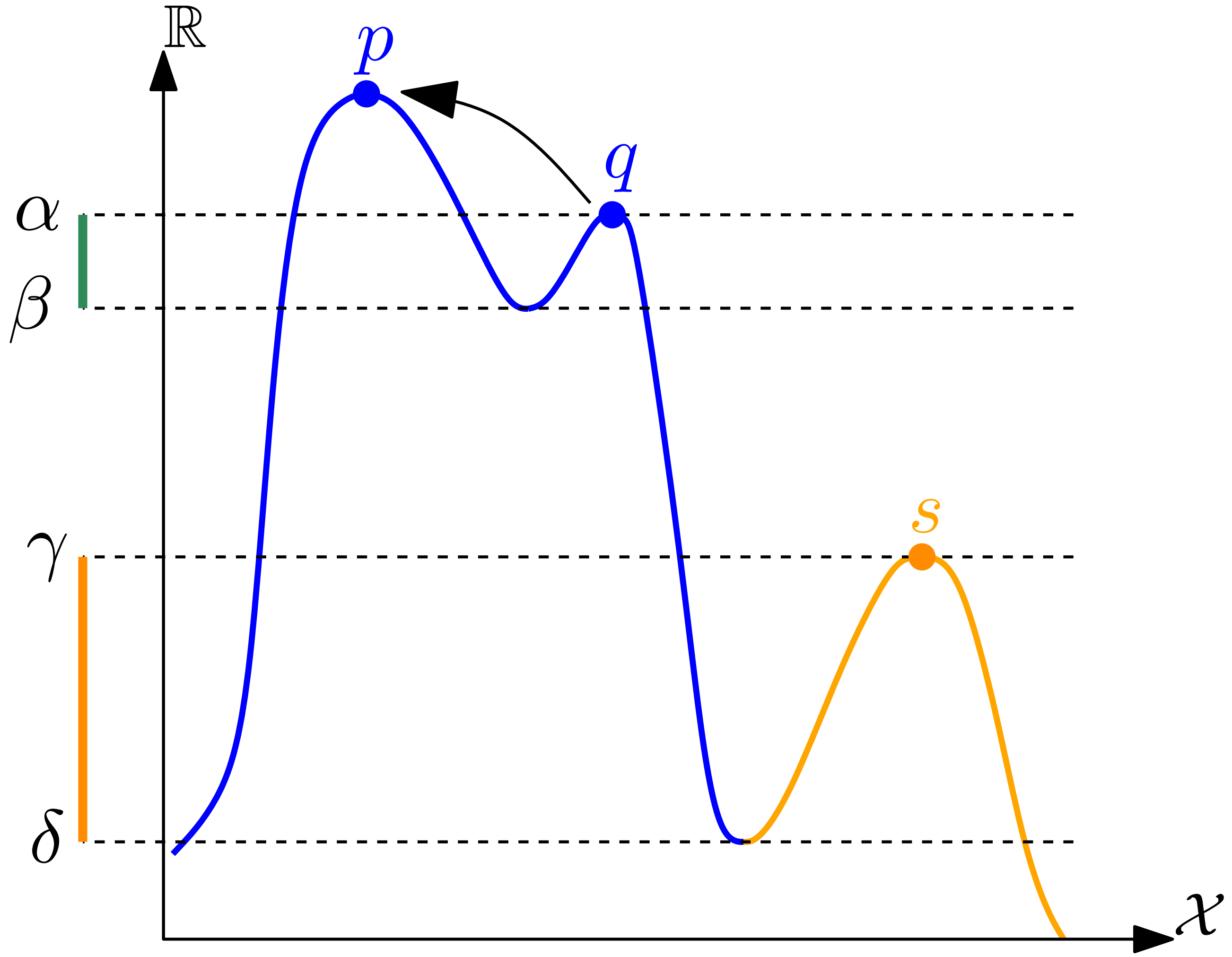


Building a hierarchy of cluster with 0-dimensional PH

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) $> \tau$!

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

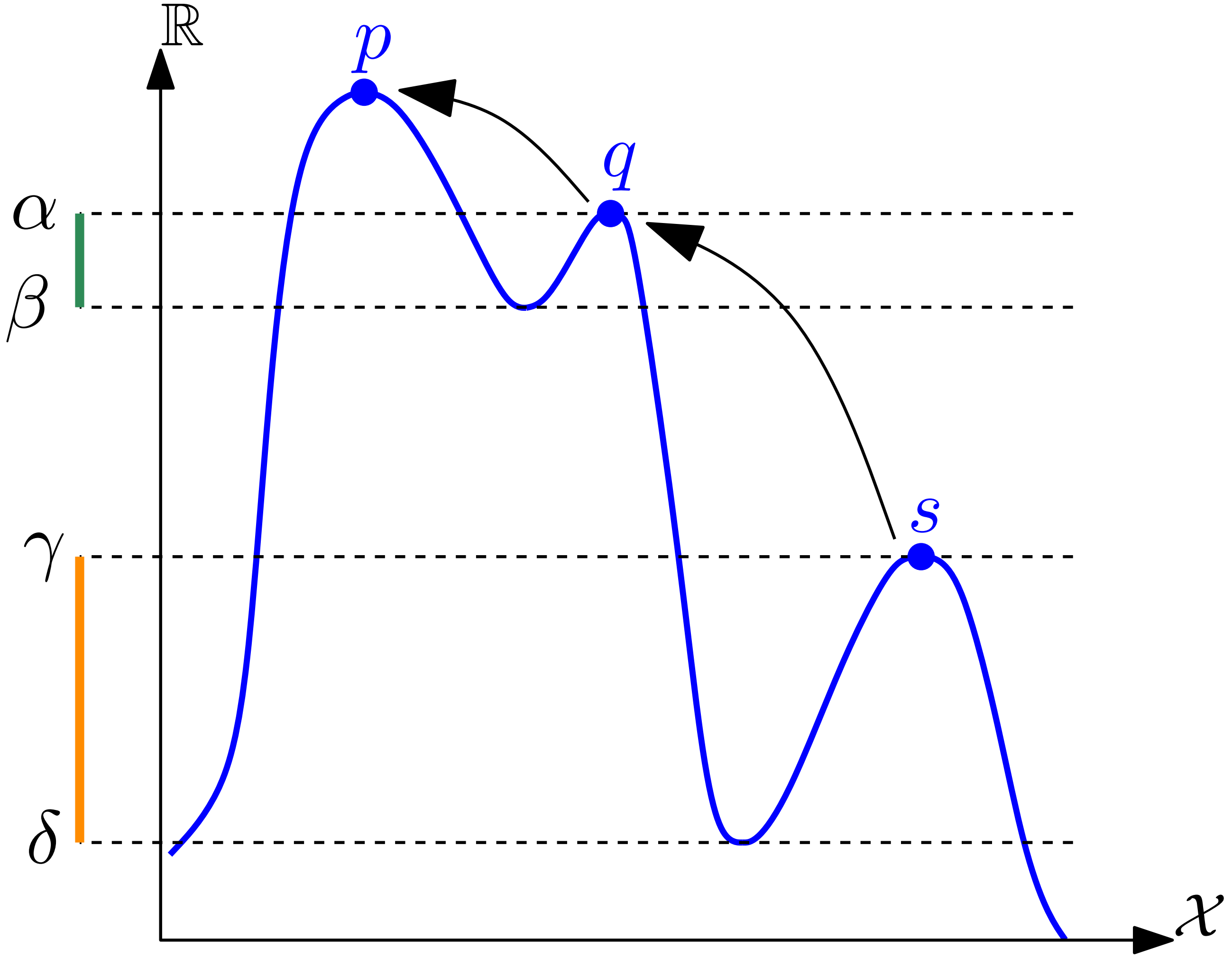


Building a hierarchy of cluster with 0-dimensional PH

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) $> \tau$!

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

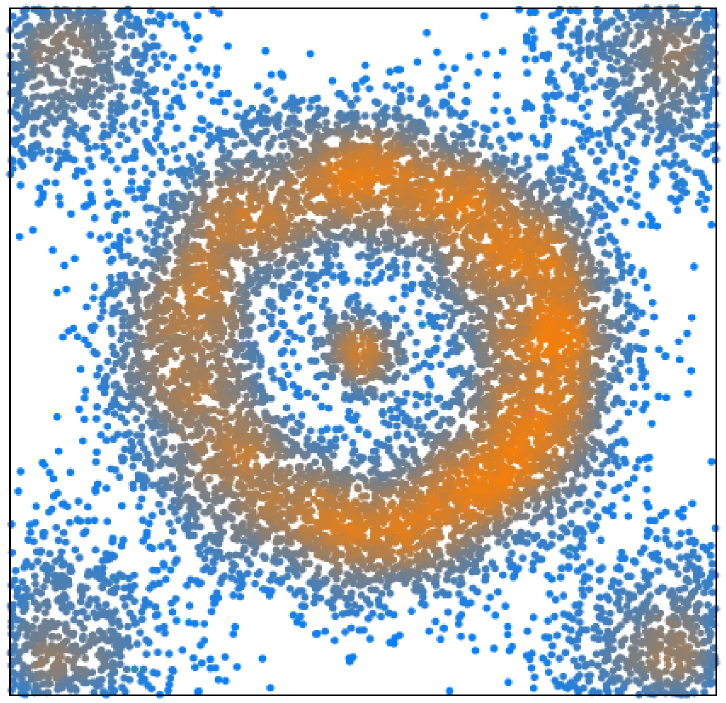


ToMATo: Topological Mode Analysis Tool

[*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

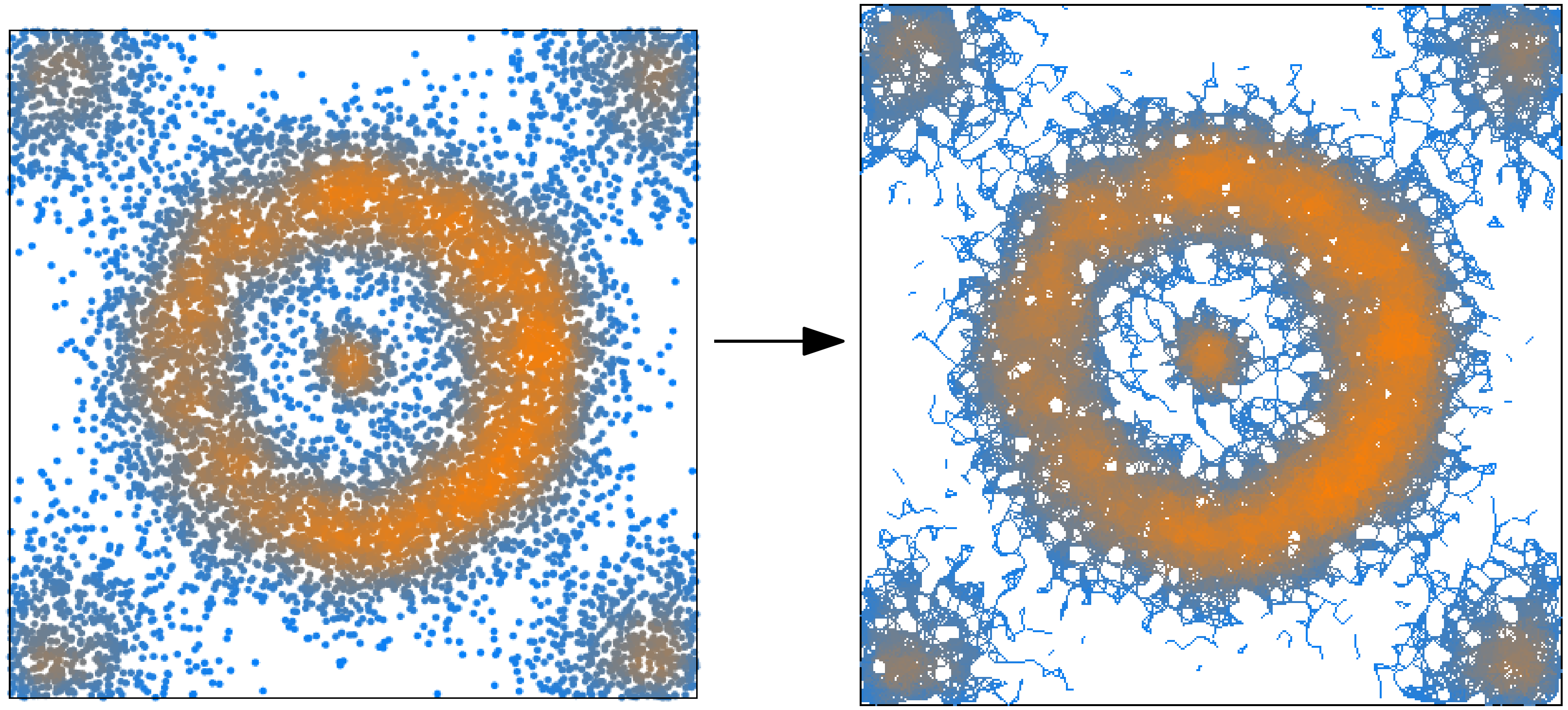
[*Persistence-Based Clustering in Riemannian Manifolds*, Chazal, Oudot, Skraba, Guibas, J. ACM, 2013]

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 $\hat{f}$.

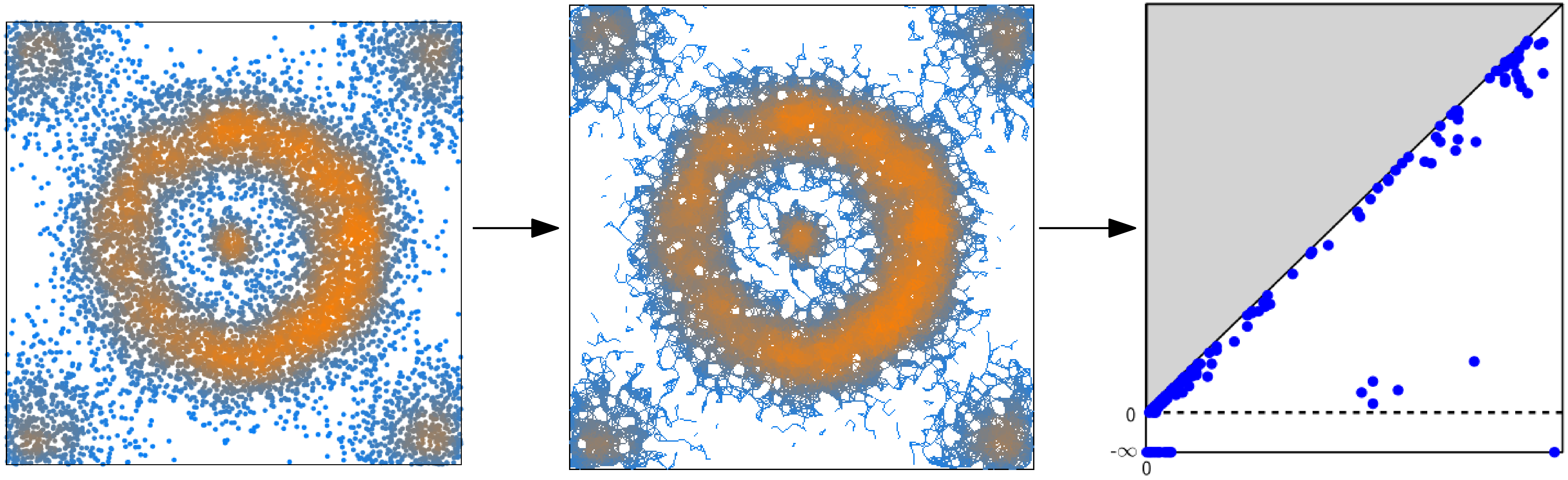
   (sort data points by **decreasing** estimated density values)

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 → 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:

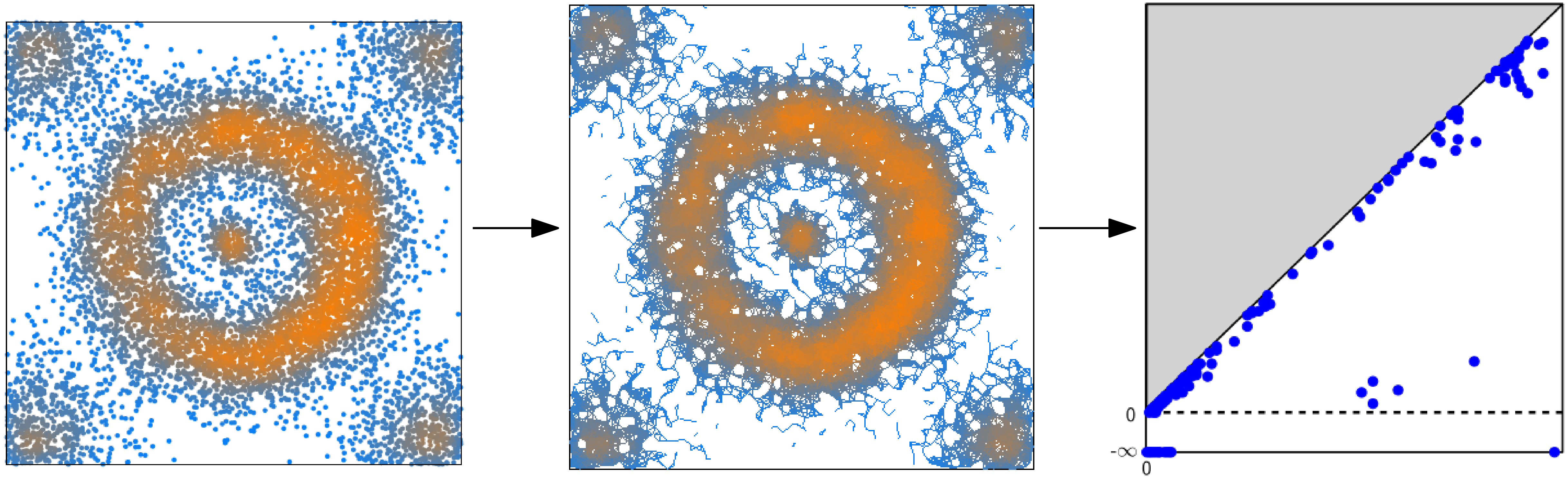
1. the algorithm sorts the vertices by decreasing density values,

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 $\hat{f}$.

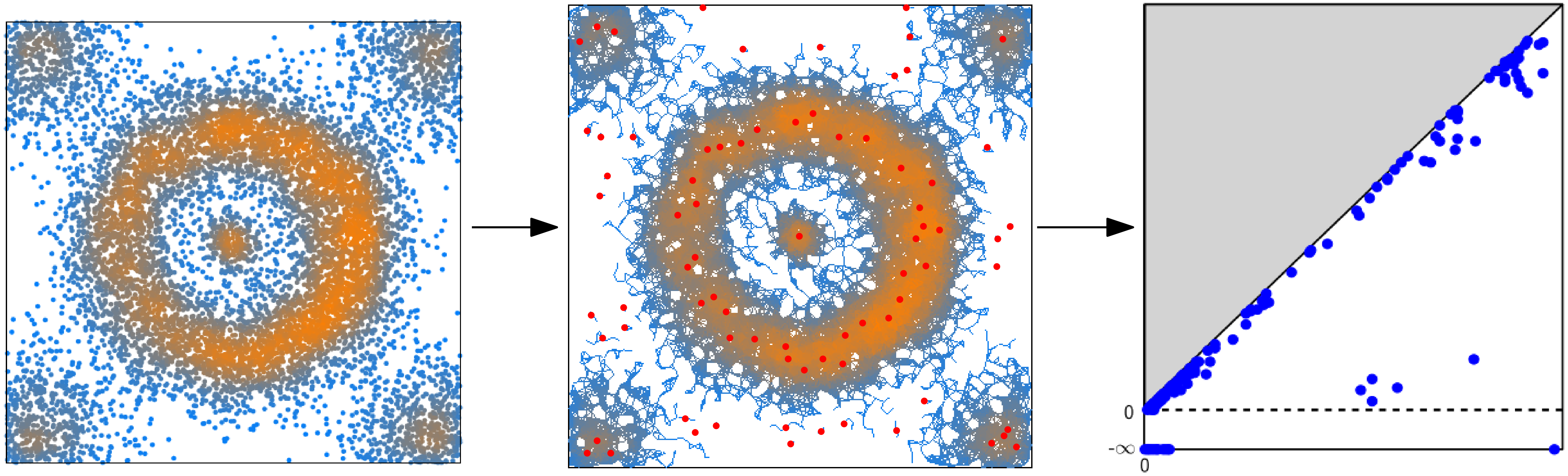
   (sort data points by **decreasing** estimated density values)

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 → union-find data structure)



Estimating the correct number of clusters

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

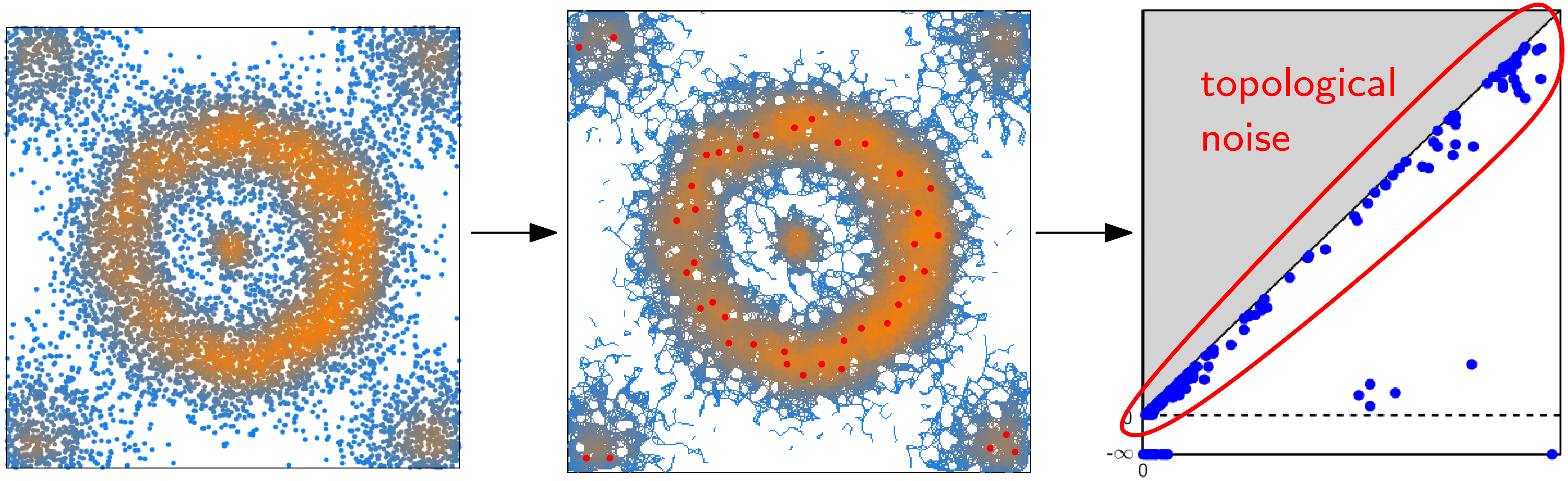
   (sort data points by **decreasing** estimated density values)

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 → union-find data structure)



Estimating the correct number of clusters

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

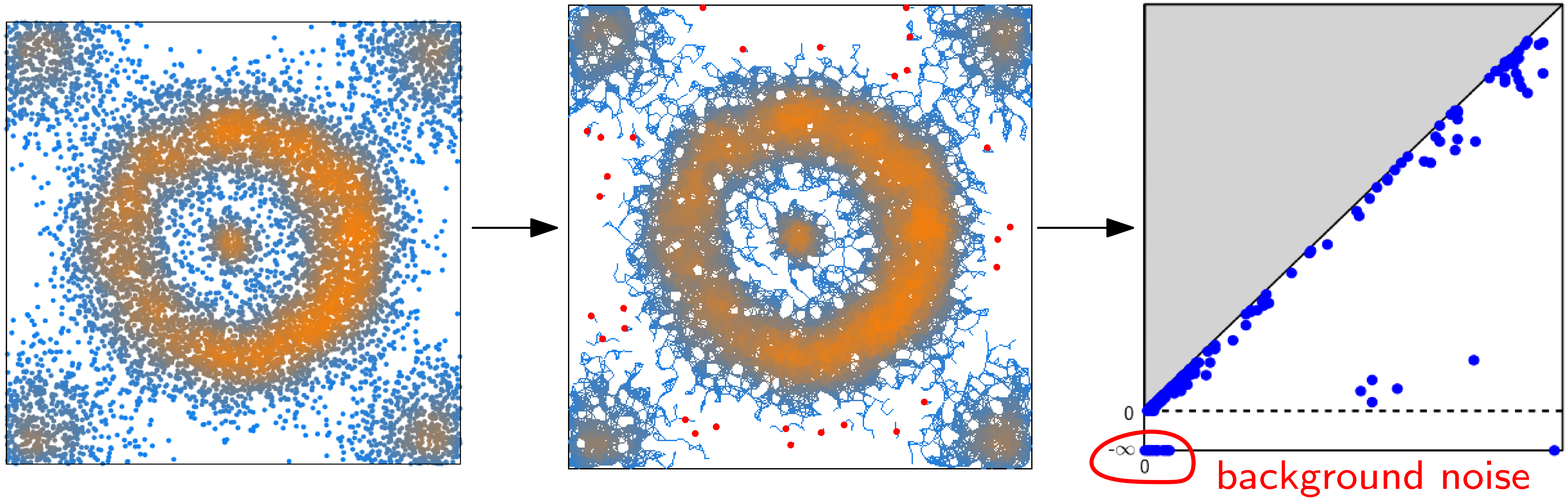
   (sort data points by **decreasing** estimated density values)

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 → union-find data structure)



Estimating the correct number of clusters

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

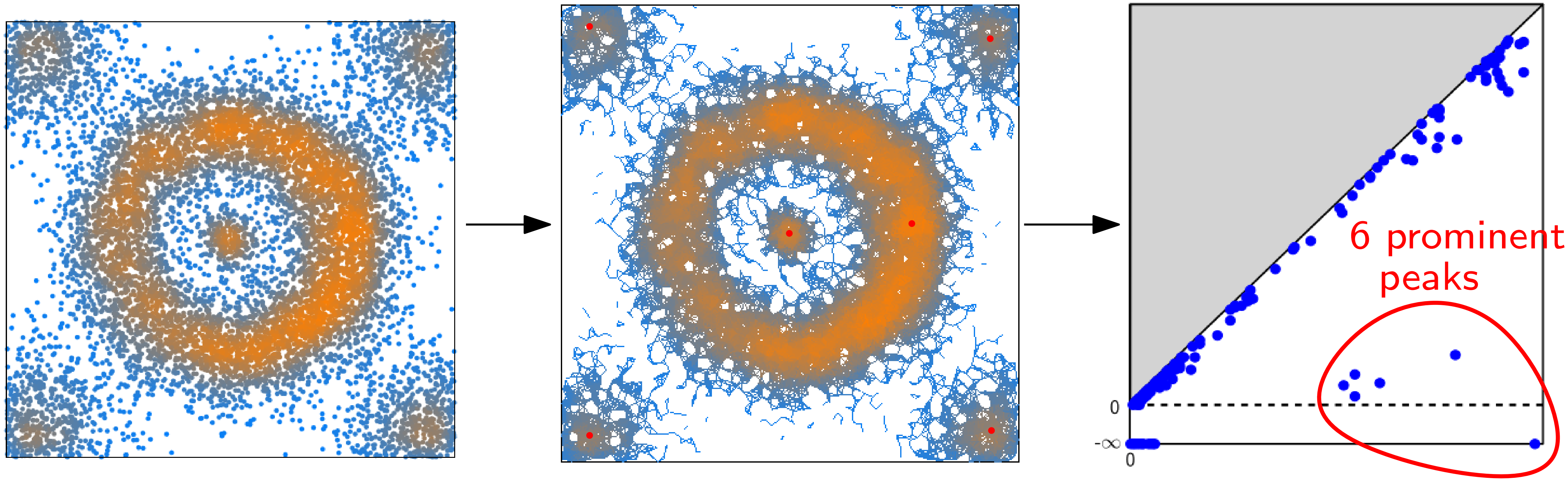
   (sort data points by **decreasing** estimated density values)

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 → union-find data structure)



6 prominent peaks

Estimating the correct number of clusters

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

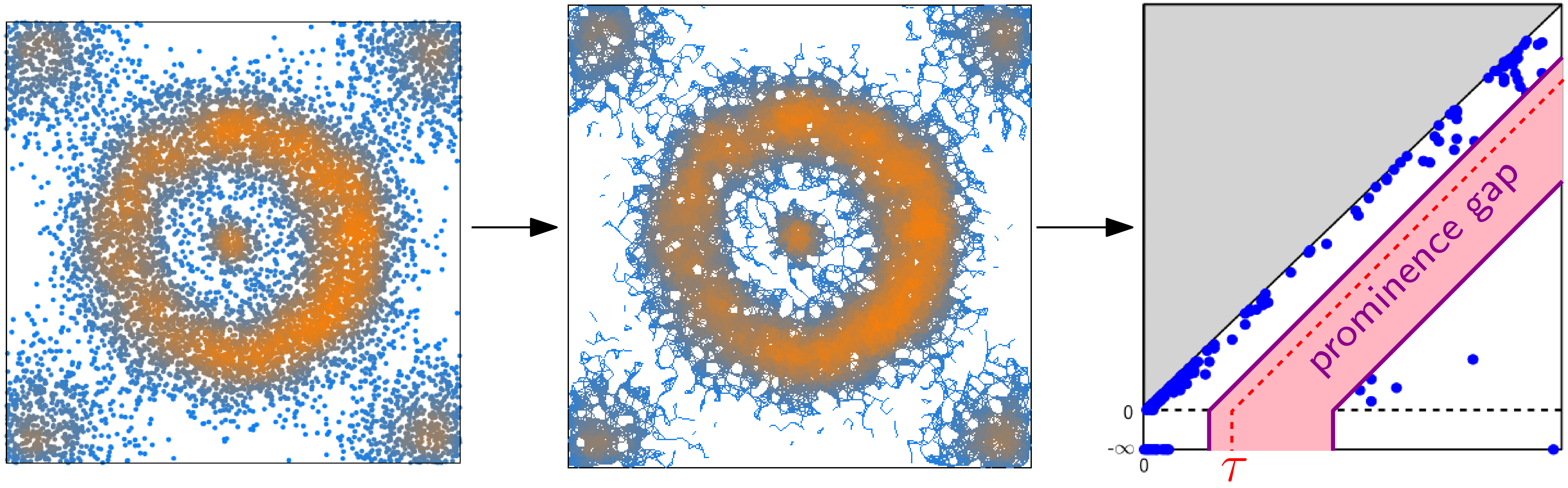
   (sort data points by **decreasing** estimated density values)

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 → 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:

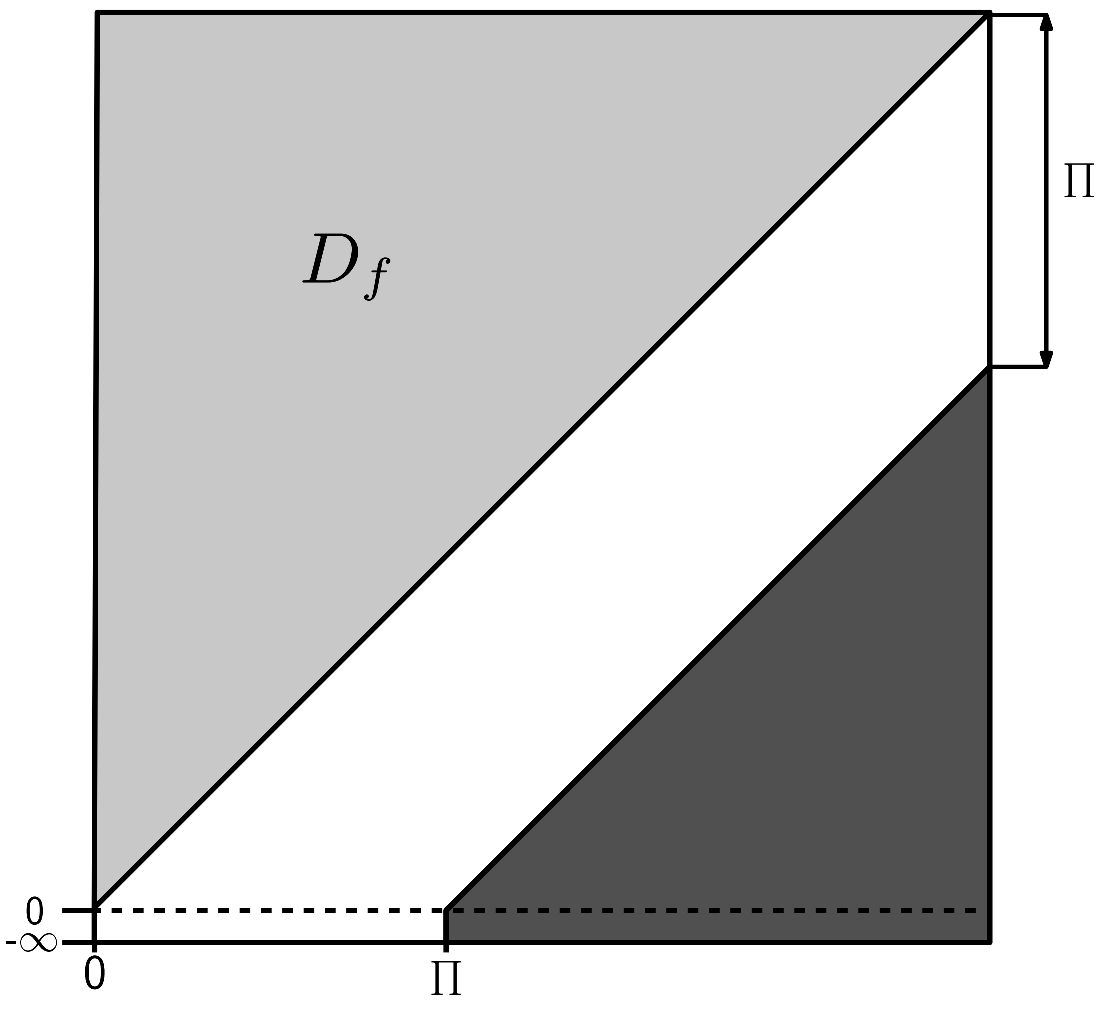
- $f : \mathbb{R}^d \to \mathbb{R}$ a c-Lipschitz probability density function,
- $P \subset \mathbb{R}^d$ a finite set of n points sampled i.i.d. according to f,
- $\hat{f} : P \to \mathbb{R}$ a density estimator s.t. $\eta := \max_{p \in P} |\hat{f}(p) - f(p)| < \Pi/5$,
- $G = (P, E)$ the δ-neighborhood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

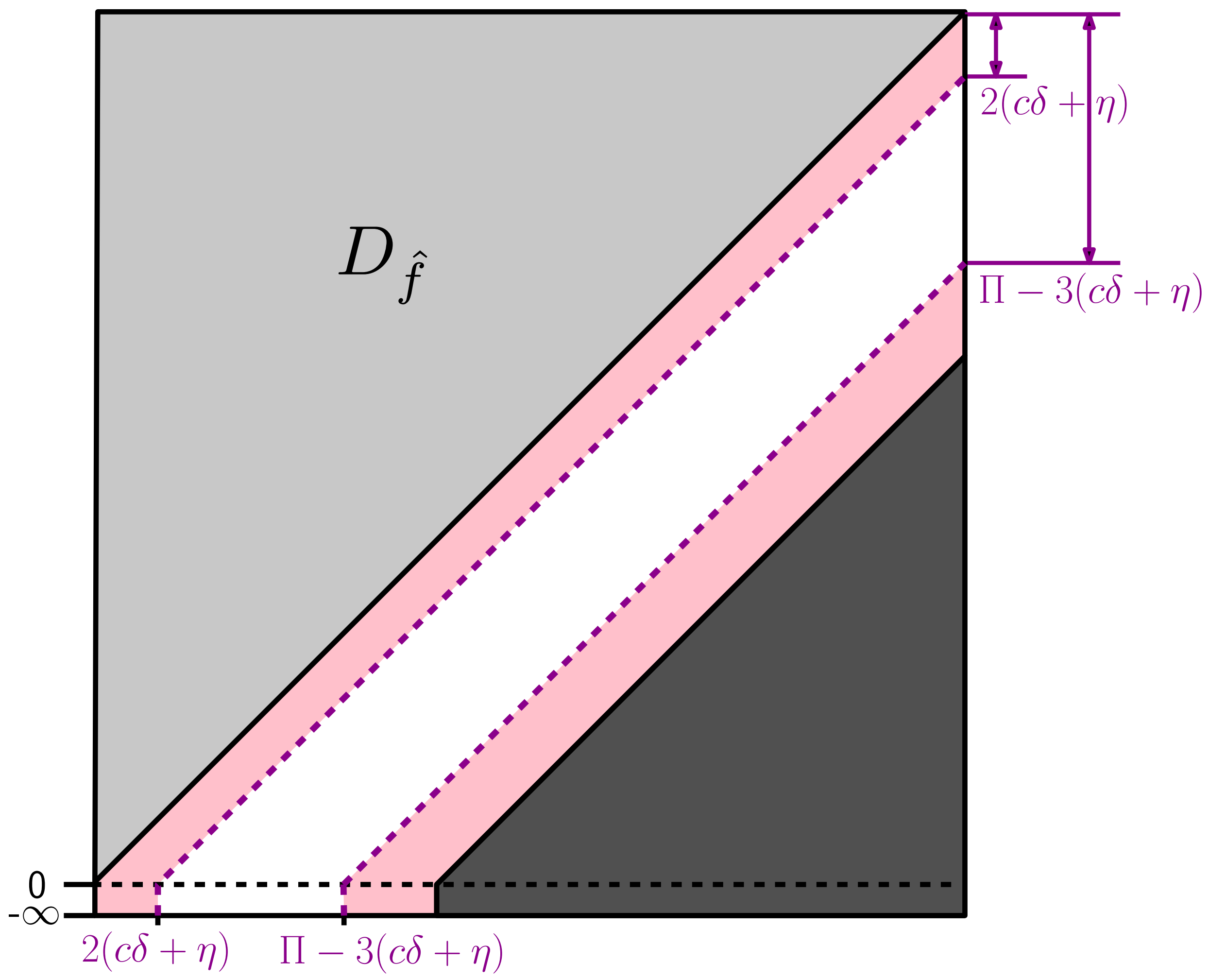
Note: Π is the prominence of the least prominent peak of f

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 $\tau \geq 0$.

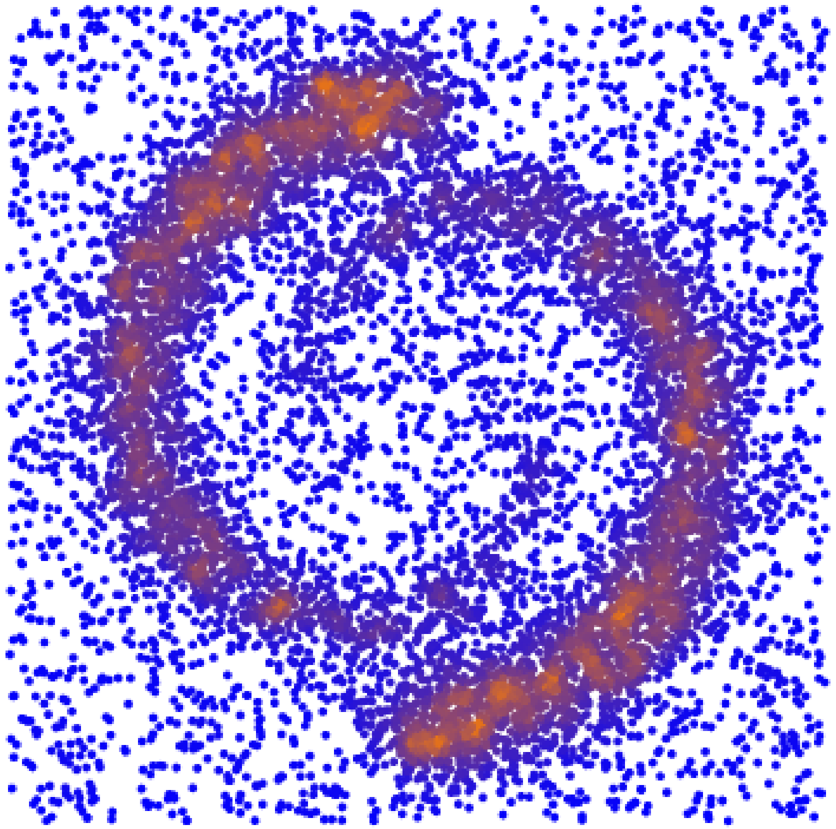
```
Sort the vertex indices $\{1, 2, \ldots, n\}$ in decreasing order: $\hat{f}(1) \geq \cdots \geq \hat{f}(n)$.
Initialize a union-find data structure $\mathcal{U}$ and two lists $g, r$ of length $n$.
for $i \in \{1, \ldots, n\}$:
    Let $\mathcal{N}$ be the set of neighbors of $i$ in $G$ that have indices lower than $i$
    if $\mathcal{N} = \emptyset$:
        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:
        $g[i] \leftarrow \mathrm{argmax}\{\hat{f}(j) : j \in \mathcal{N}\}$ ($g[i]$ stores the approximate gradient at vertex $i$)
        $e_i \leftarrow \mathcal{U}.\mathtt{Find}(g[i])$
        Attach vertex $i$ to the entry $e_i$: $\mathcal{U}.\mathtt{Union}(i, e_i)$
        for $j \in \mathcal{N}$:
            $e \leftarrow \mathcal{U}.\mathtt{Find}(j)$
            if $e \neq e_i$ and $\min\{\hat{f}(r[e]),\ \hat{f}(r[e_i])\} < \hat{f}(i) + \tau$:
                $\mathcal{U}.\mathtt{Union}(e,\ e_i)$
                $r[e \cup e_i] \leftarrow \mathrm{argmax}\{\hat{f}(r[e]),\ \hat{f}(r[e_i])\}$
                $e_i \leftarrow e \cup e_i$
```

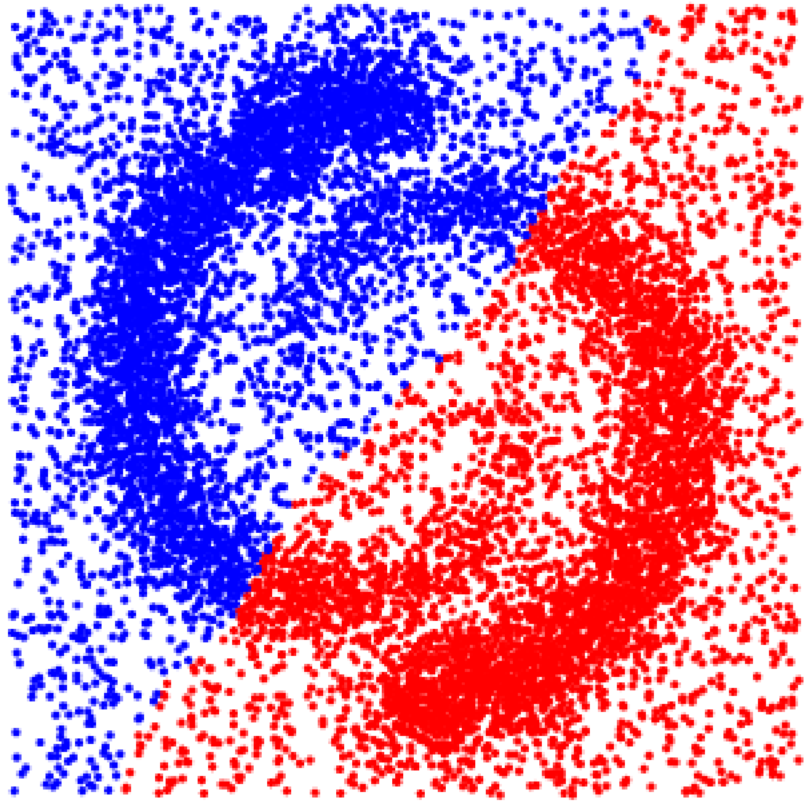
cluster merges with persistence

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

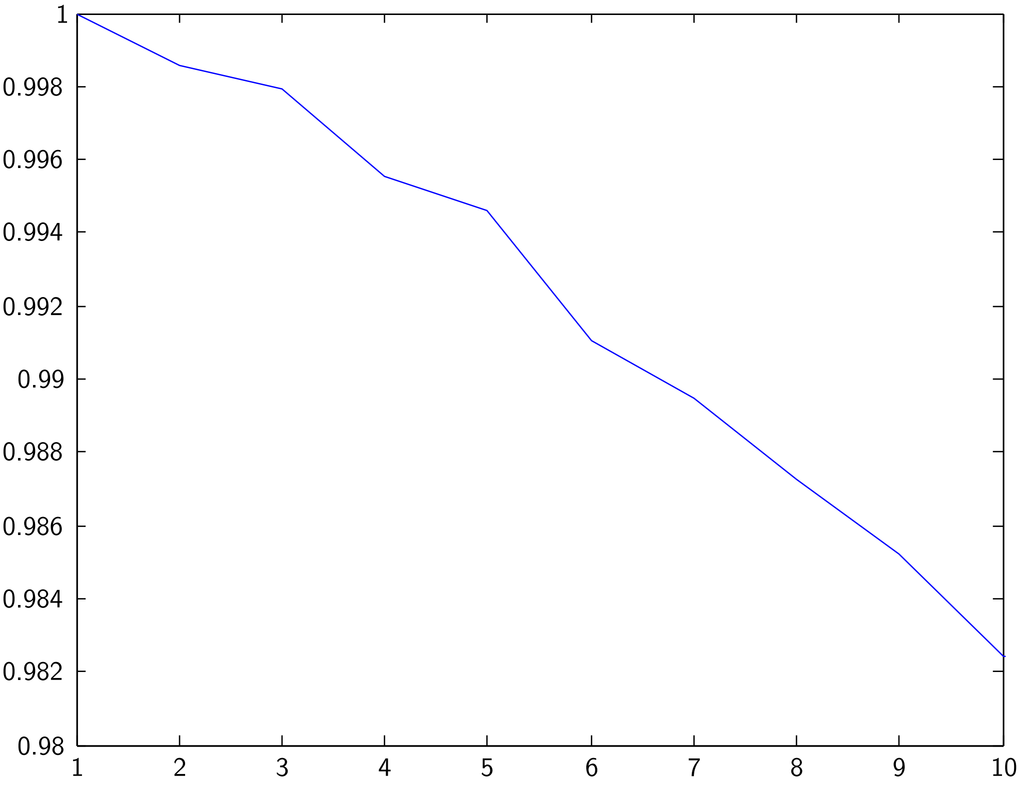
Experimental results

Synthetic Data



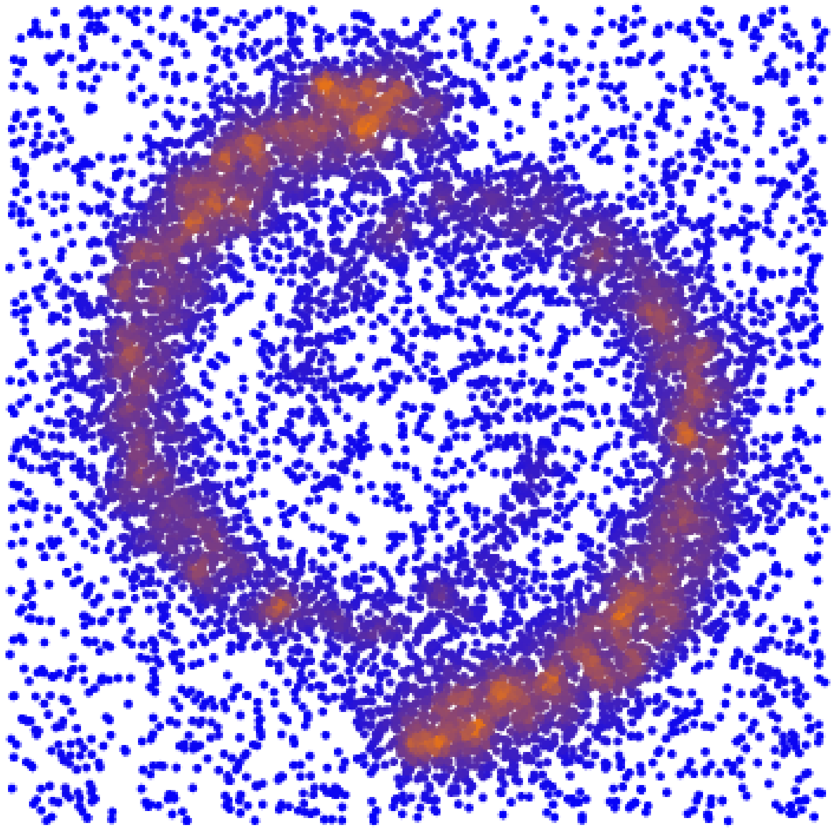


Spectral clustering
(k-means in eigenspace)



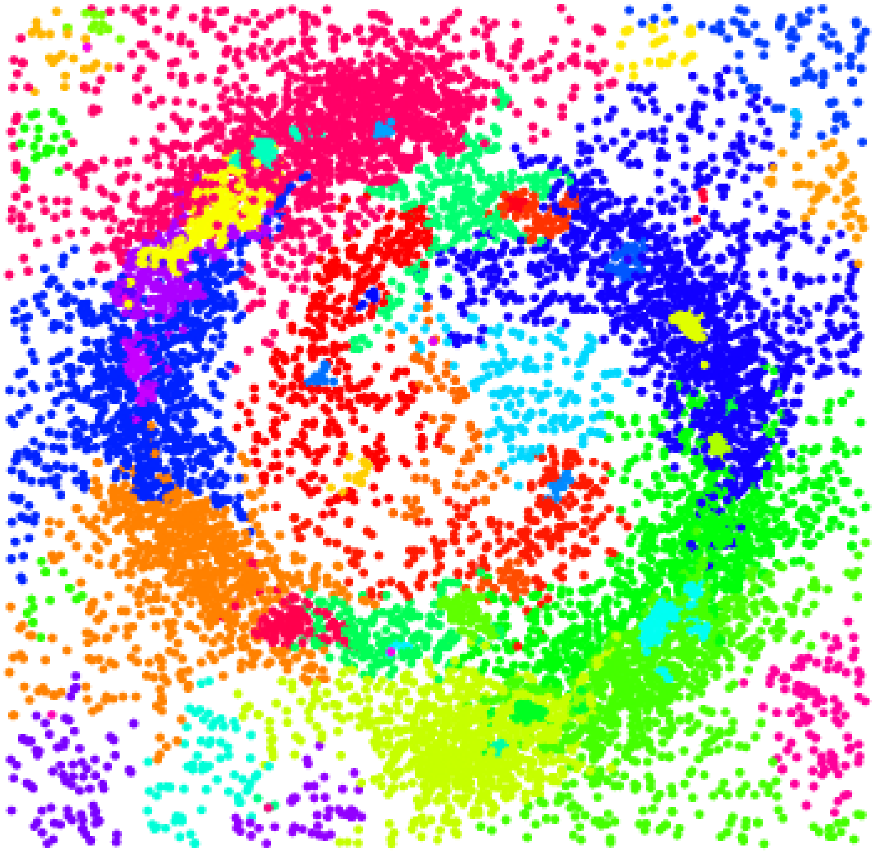
Experimental results

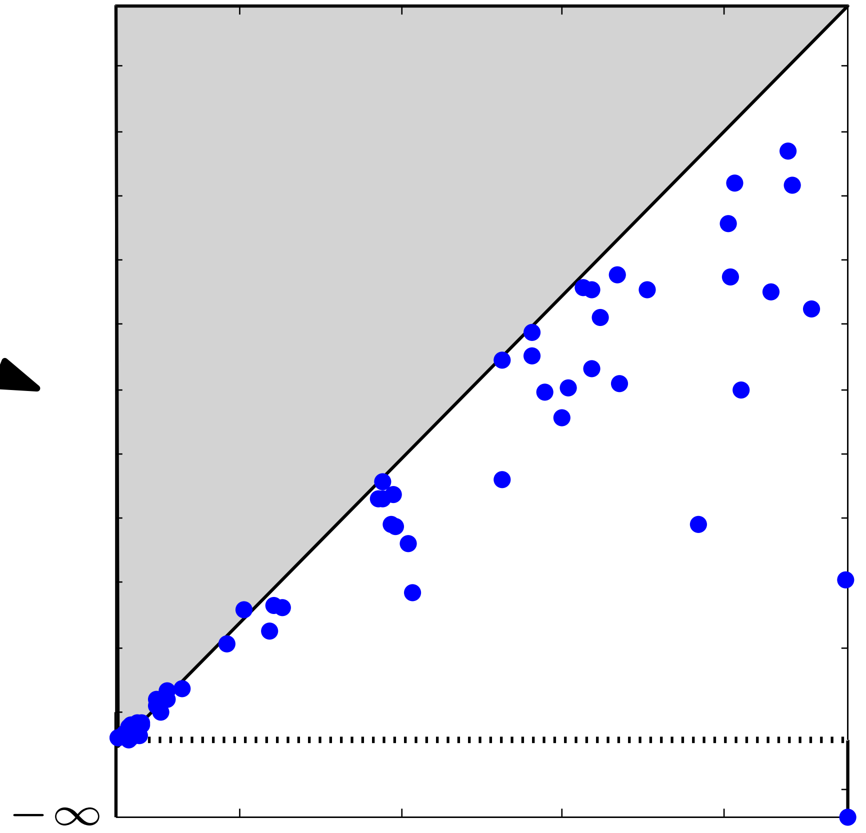
Synthetic Data



ToMATo

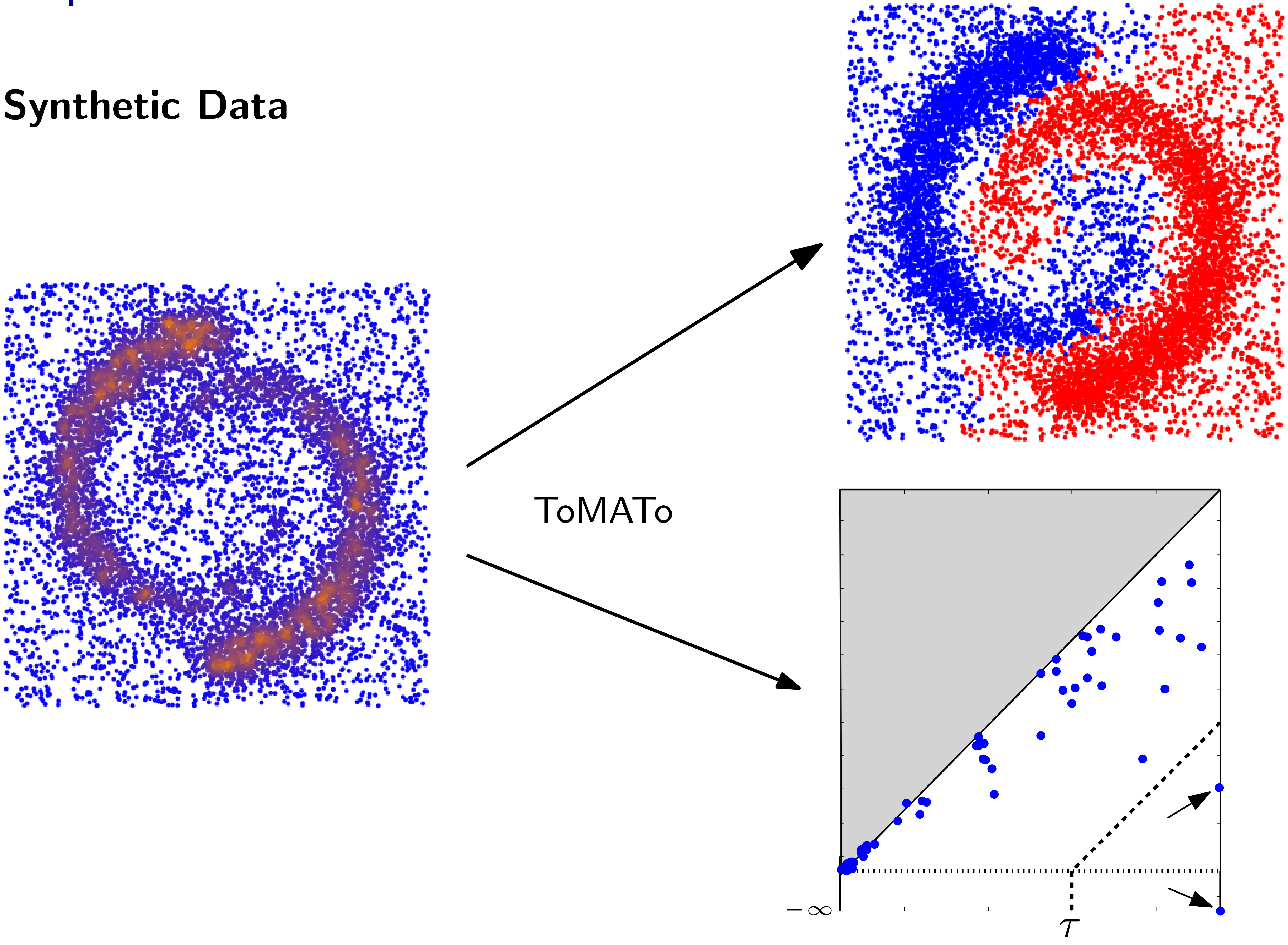
$\tau = 0$





Experimental results

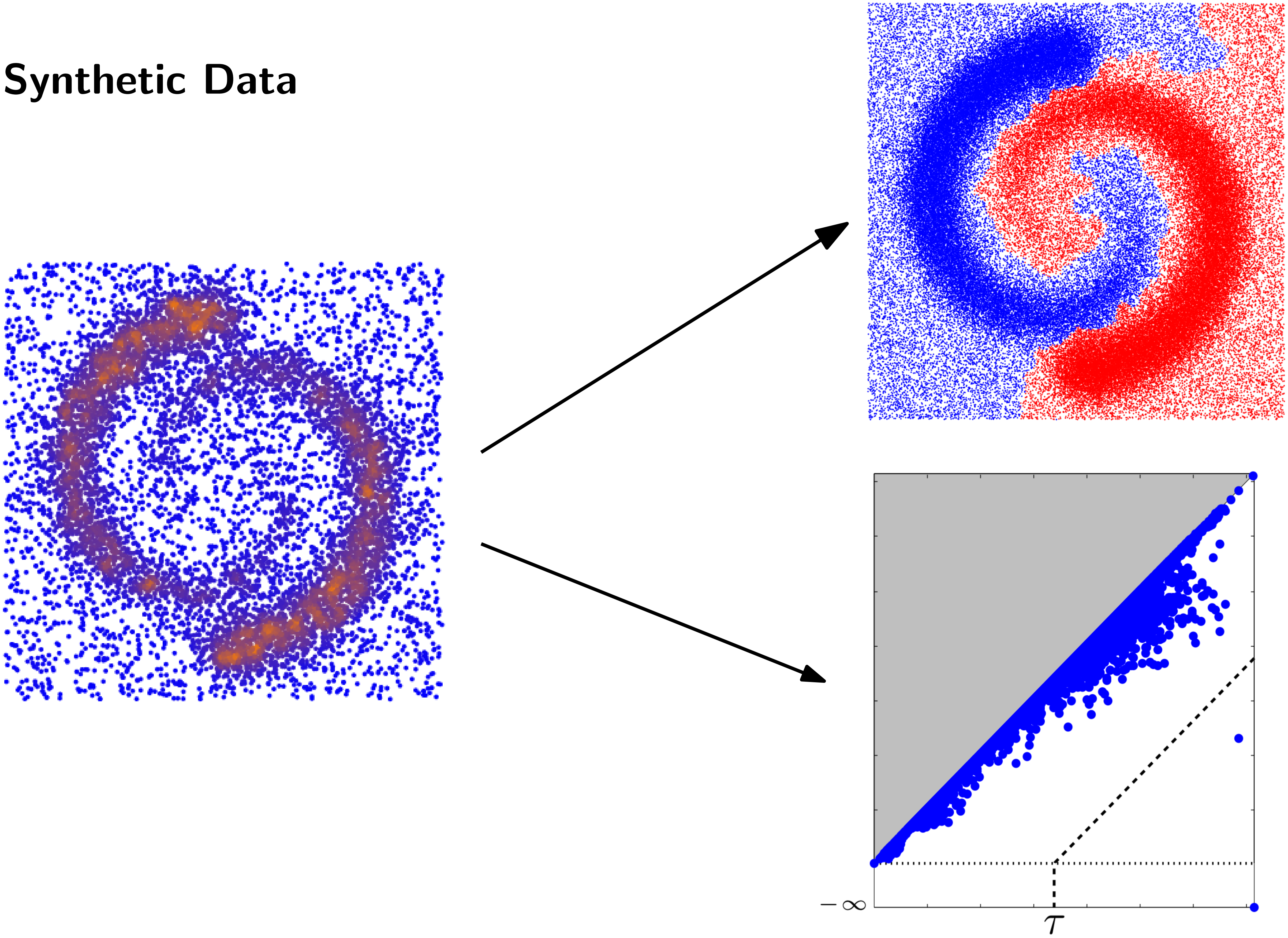
Synthetic Data



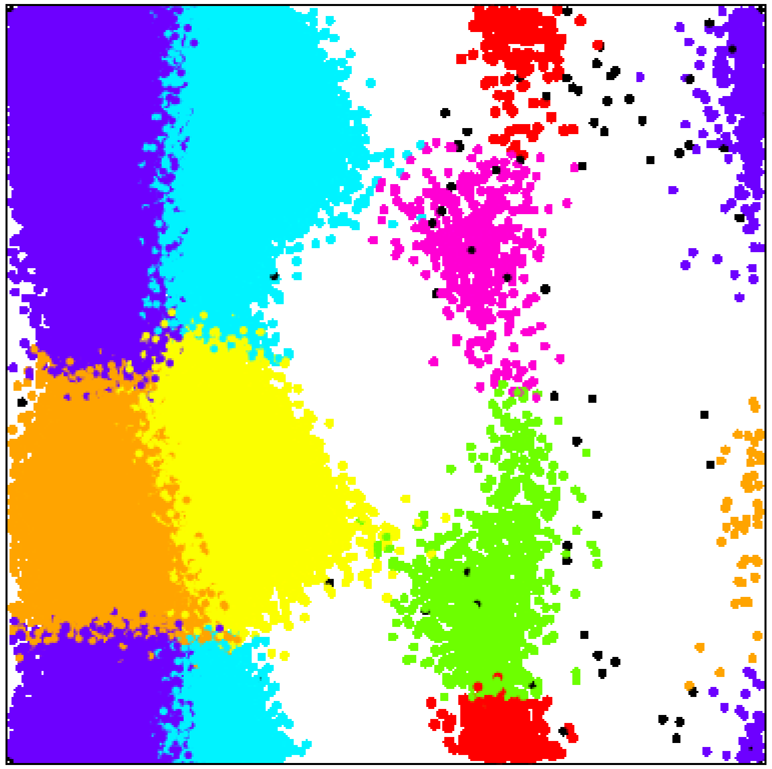
ToMATo

Experimental results

Synthetic Data



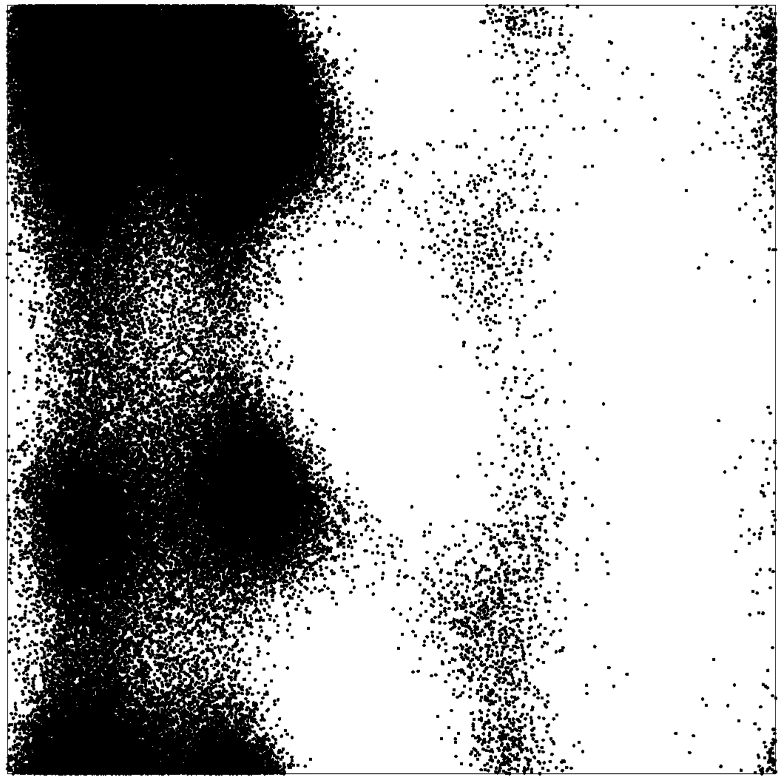
Experimental results

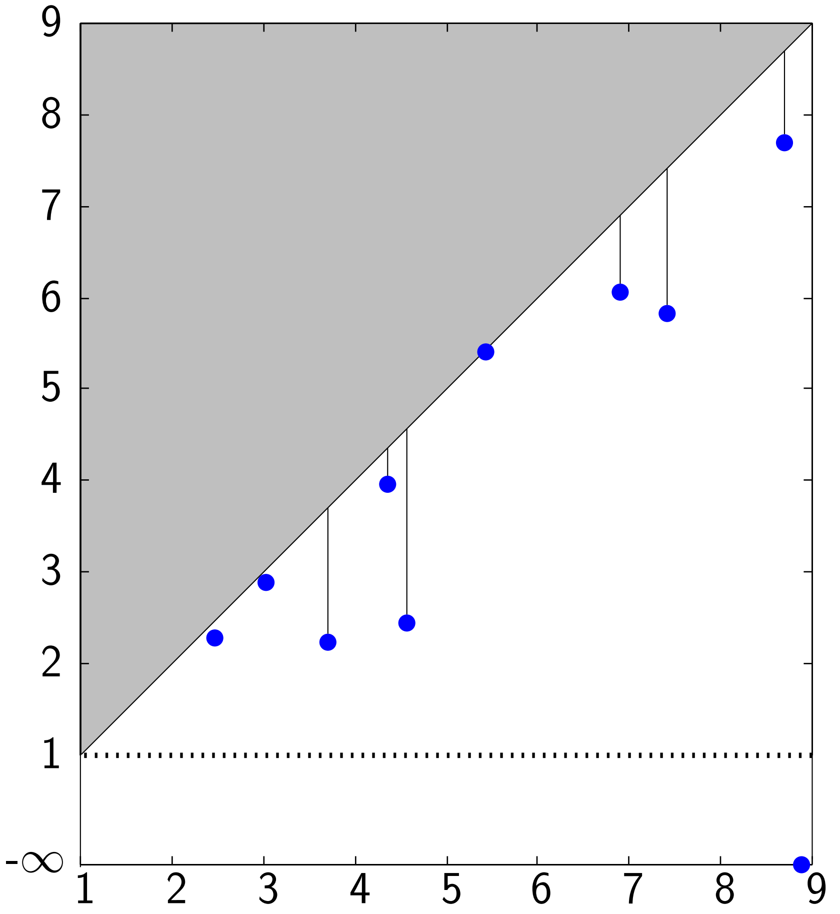


Biological Data

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

with RMSD distance (non-Euclidean).





Common belief: 6 metastable states.

PD shows anywhere between 4 and 7 clusters.

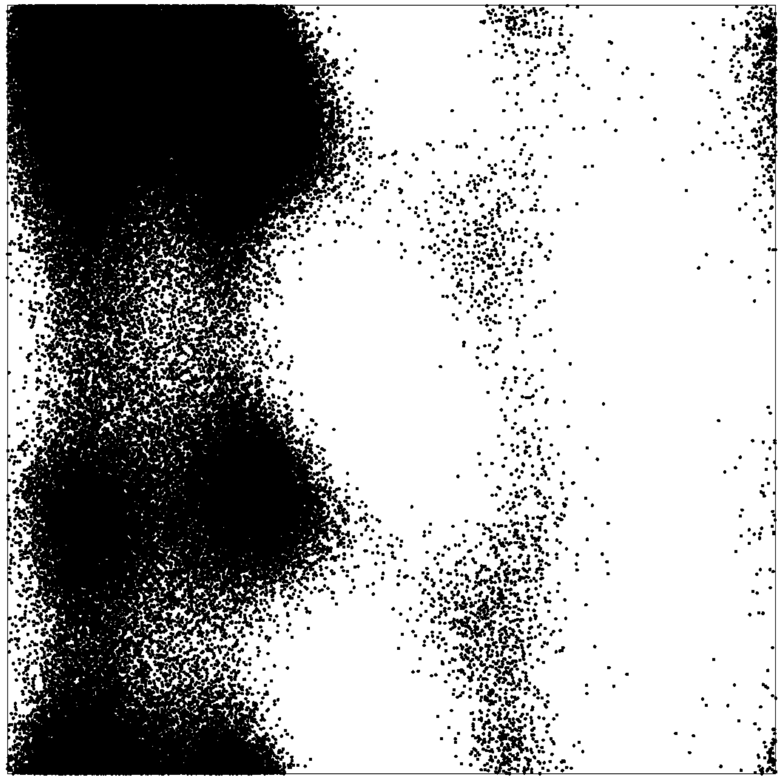
Experimental results

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

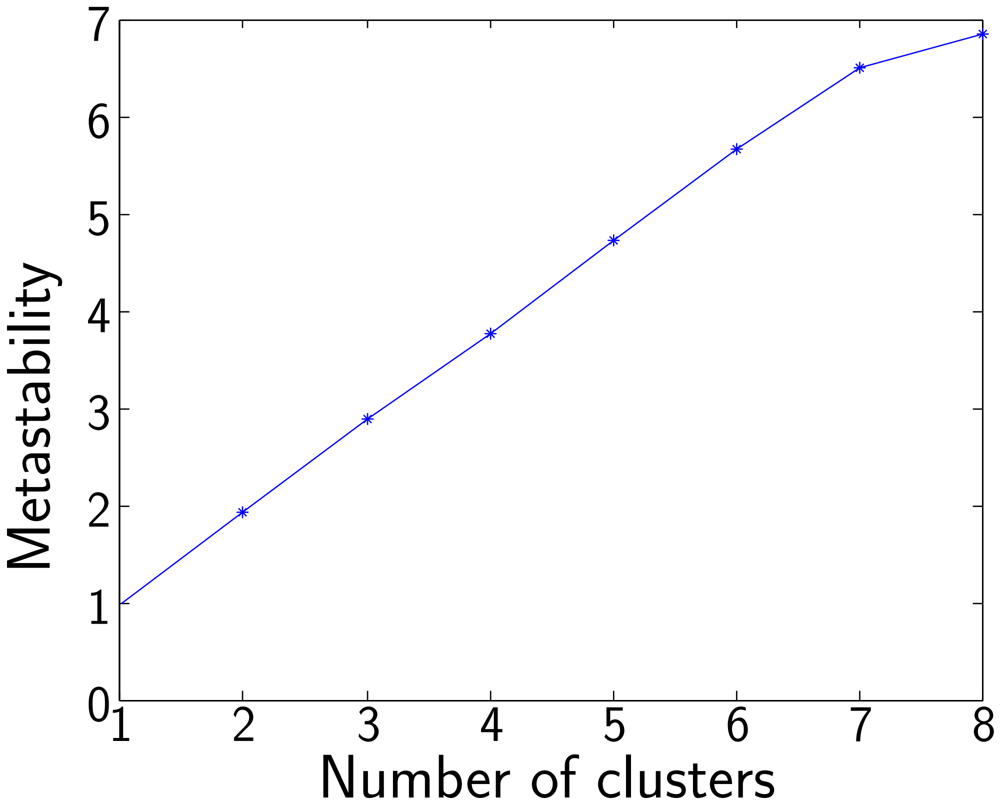
Biological Data

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.

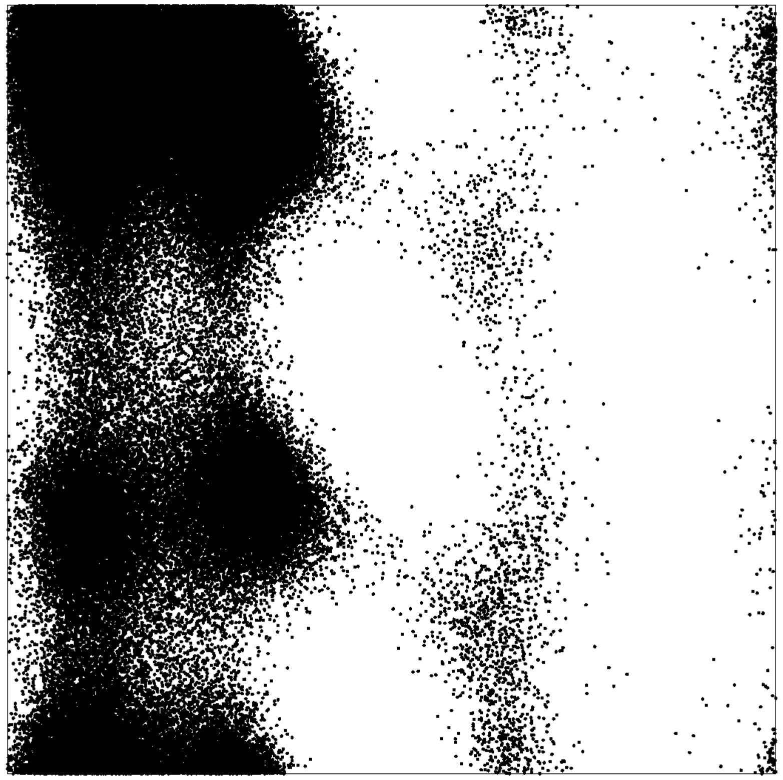
Experimental results

[*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).



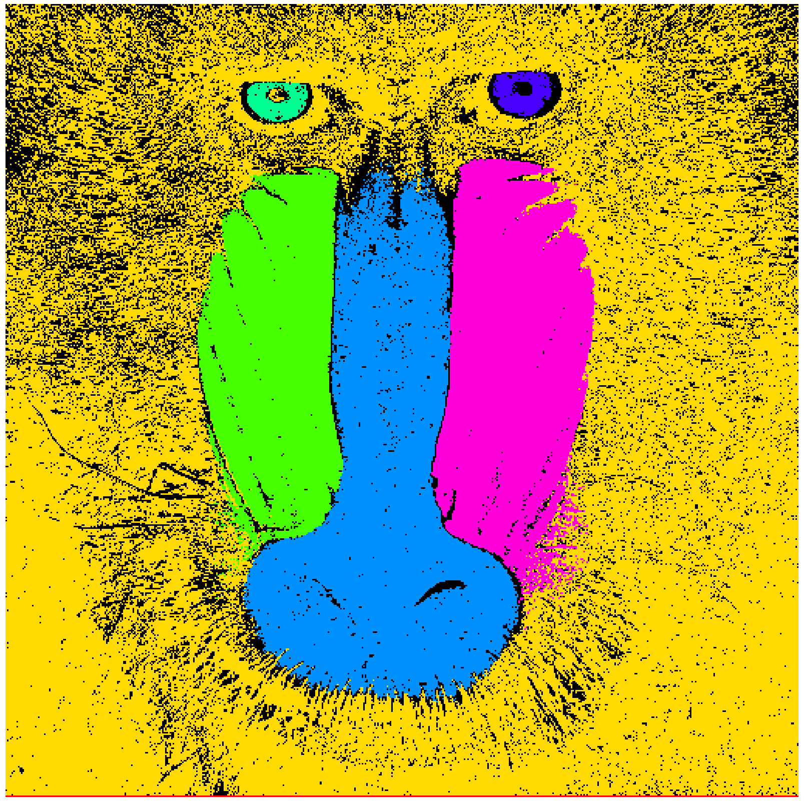
Note: Spectral Clustering takes a week of tweaking, while ToMATo runs out-of-the-box in a few minutes.

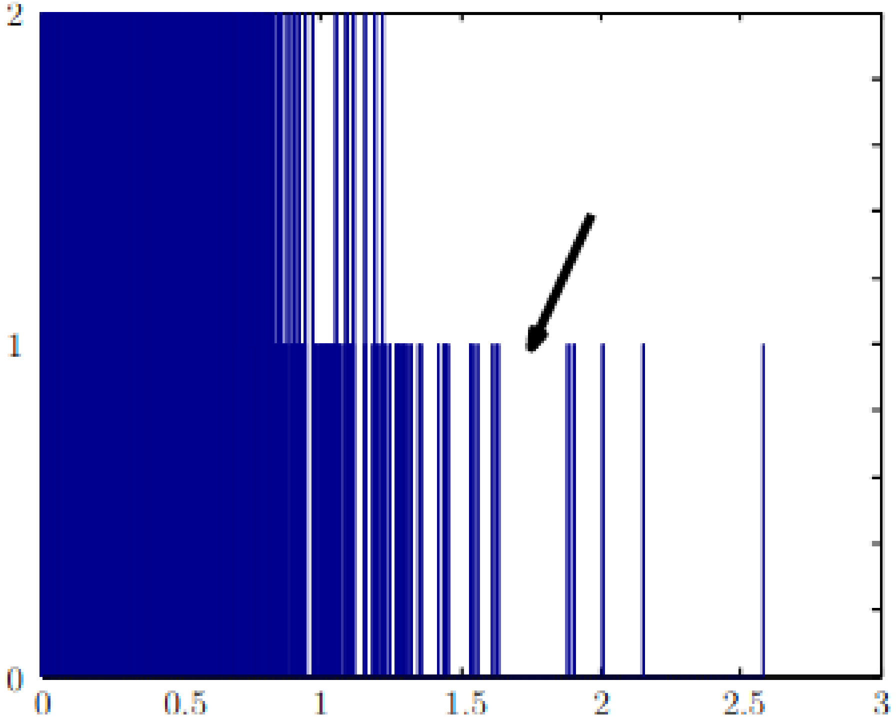
Experimental results

Image Segmentation

Density is estimated in 3D color space.

Neighborhood graph is built in image domain.



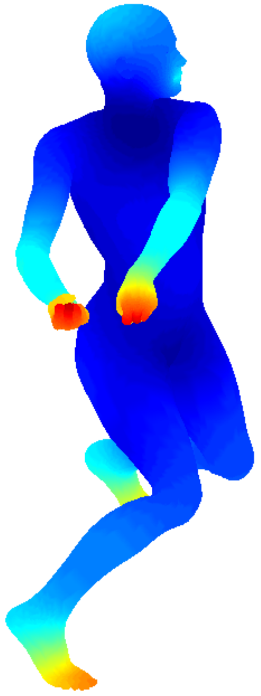


Distribution of prominences does not usually show a clear unique gap.

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

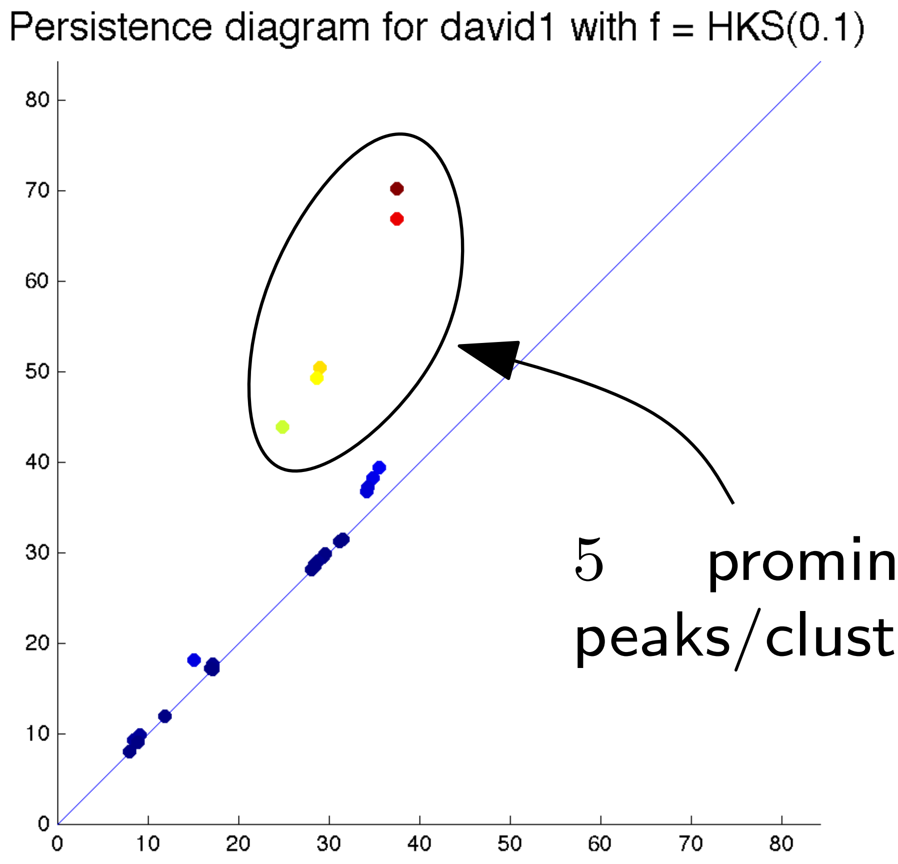
Application to non-rigid shape segmentation

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

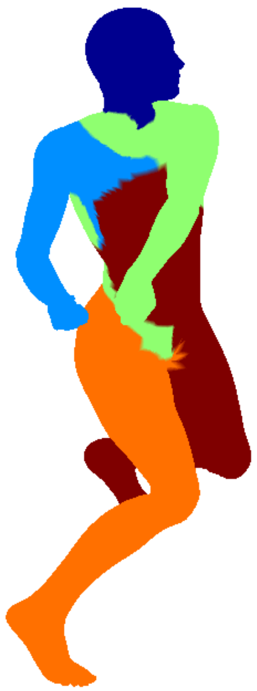


X : a 3D shape

f = HKS function on X

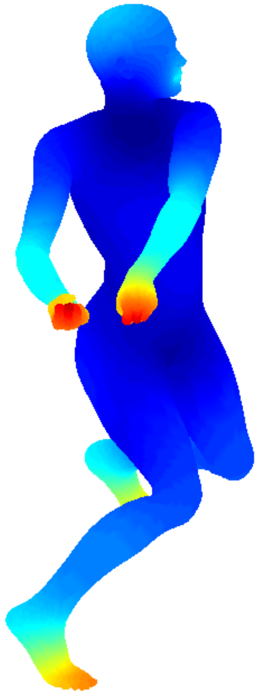


5 prominent peaks/clusters



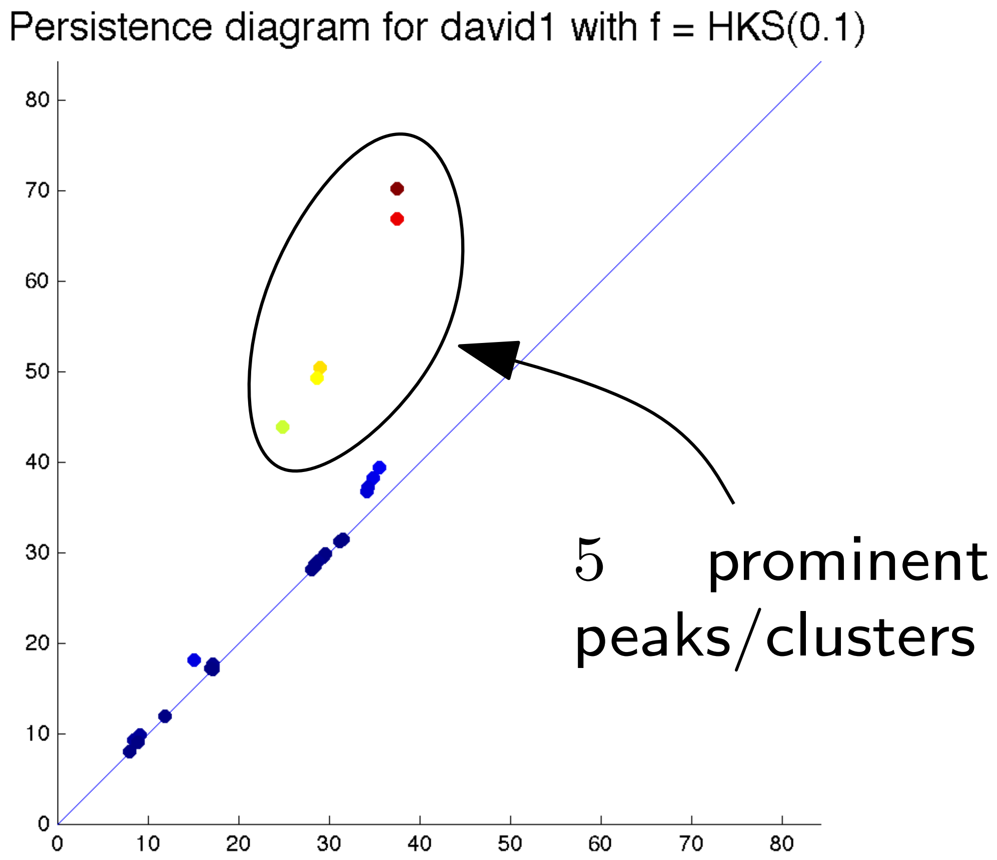
Application to non-rigid shape segmentation

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

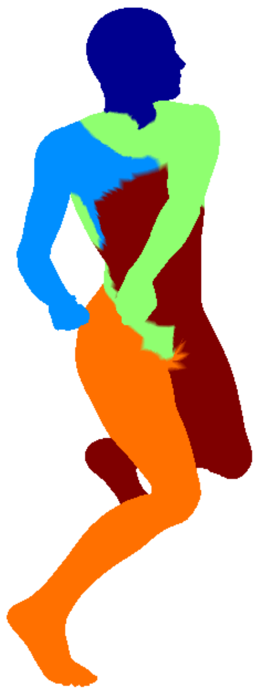


X : a 3D shape

f = HKS function on X



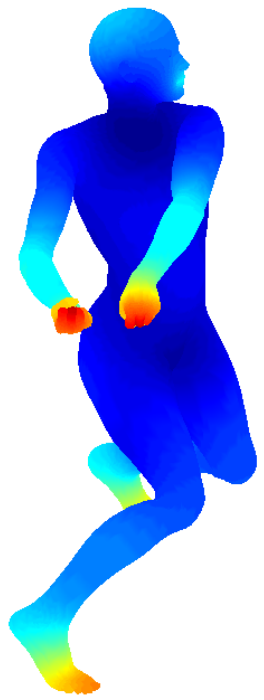
5 prominent peaks/clusters



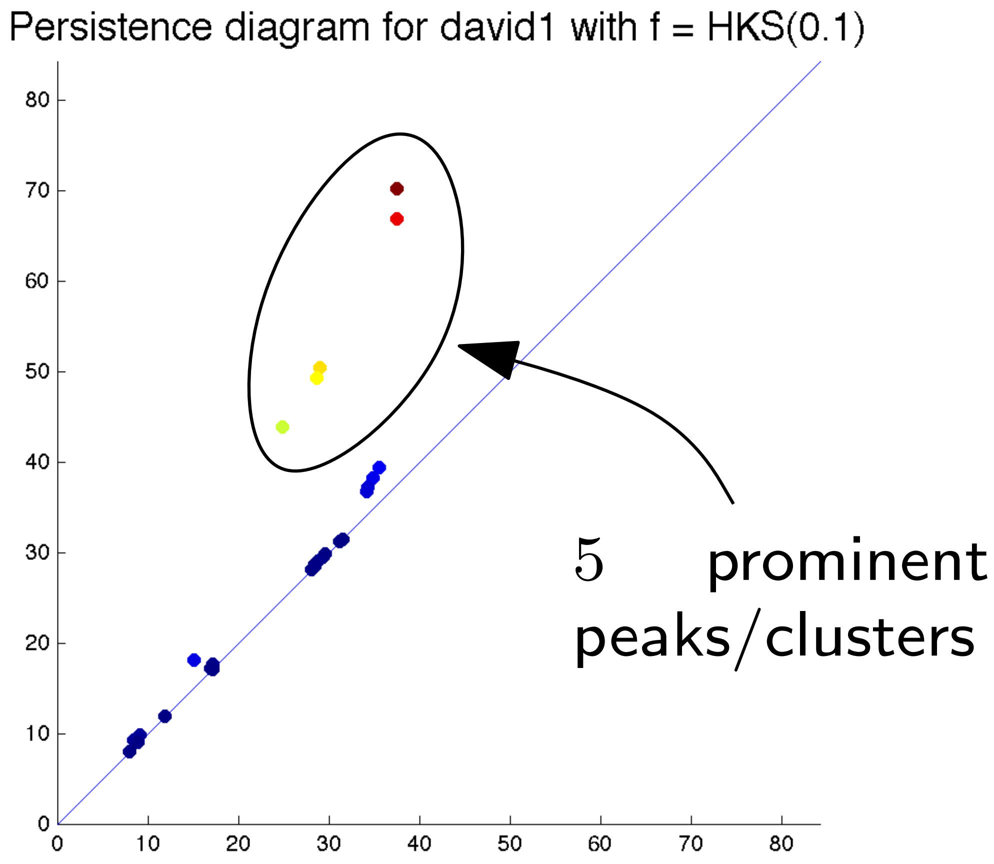
Problem: cluster boundaries are unstable, which gives dirty segments.

Application to non-rigid shape segmentation

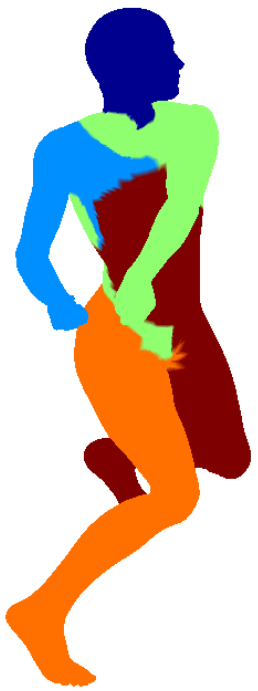
[*Persistence-Based Segmentation of Deformable Shapes*, Skraba, Ovsjanikov, Chazal, Guibas, Proc. CVPR 2010]



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



5 prominent peaks/clusters

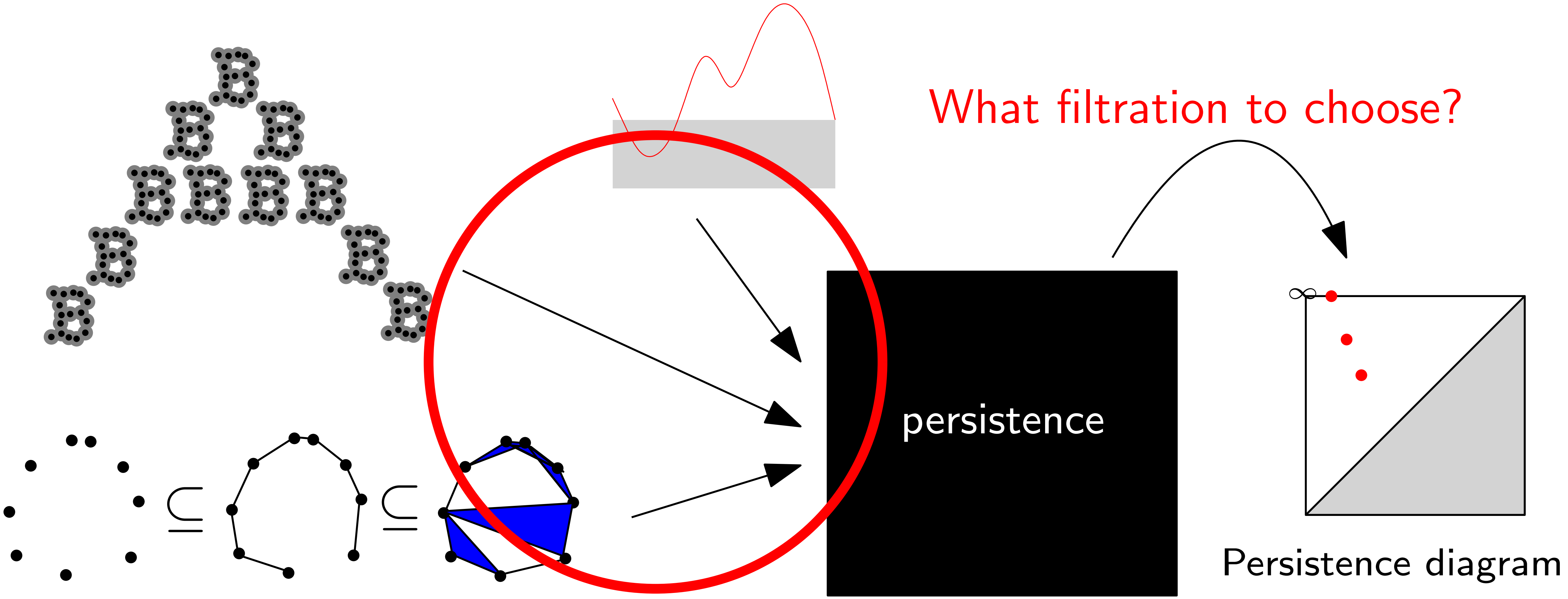


Problem: cluster boundaries are unstable, which gives dirty segments.

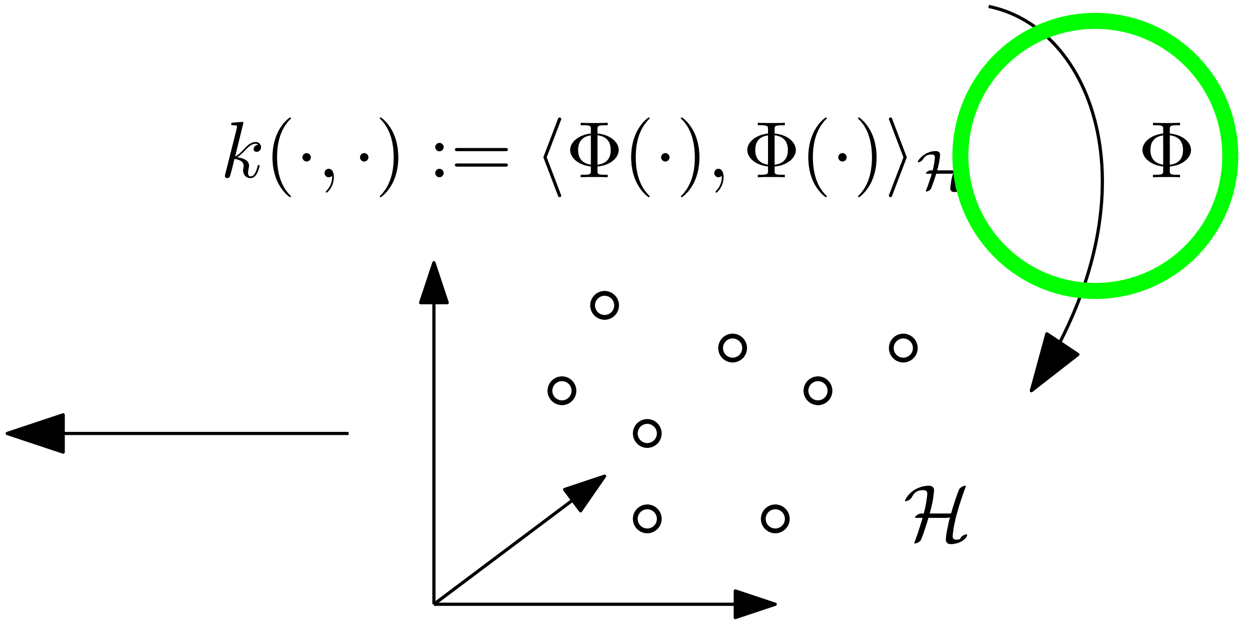
Topological Machine Learning (II): Guiding ML models

1. Hierarchical and Mode Seeking Clustering
2. Topology-based Clustering
3. Topology-based Optimization

Persistence diagrams and optimization



What filtration to choose?



$k(\cdot,\cdot) := \langle \Phi(\cdot), \Phi(\cdot) \rangle_{\mathcal{H}}$

- Classifier (RF, SVM, NN etc.)
- Dim. red. (PCA, MDS, UMAP, t-SNE)
- Clustering (DBSCAN, K-means, etc.)

Etc.

What representation to choose? → PersLay

Problem setting

Q: How to define ∇D?

Problem setting

Q: How to define ∇D?

Q: Given a parameterized family of functions $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$, how to define $\nabla_\theta D_{f_\theta}$?

Q: Given a point cloud $X \subseteq \mathbb{R}^d$, how to define $\nabla_X D_{\mathrm{Rips}}(X)$?

Problem setting

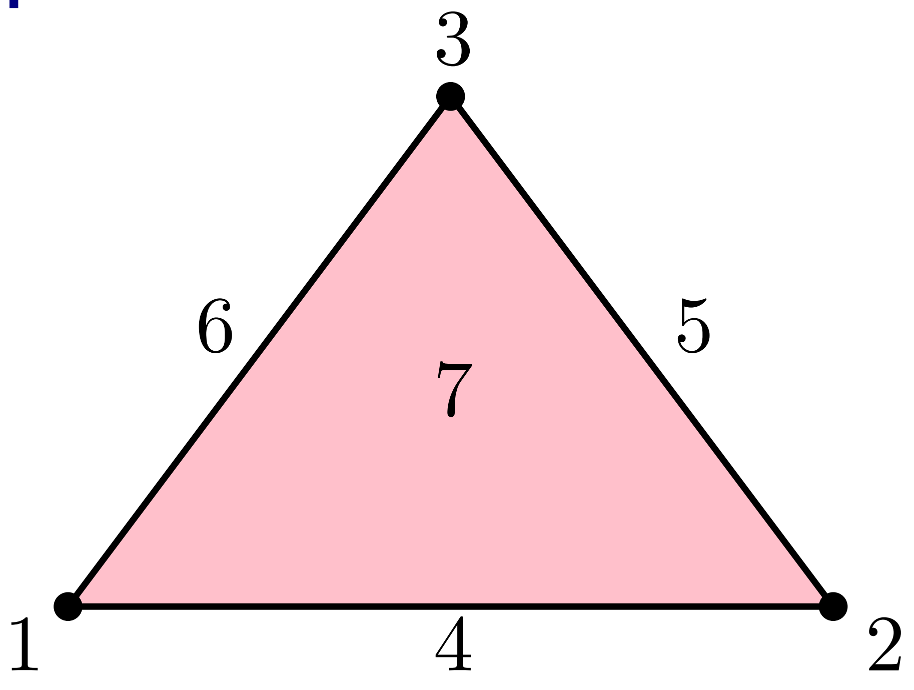
Q: How to define ∇D?

Q: Given a parameterized family of functions $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$, how to define $\nabla_\theta D_{f_\theta}$?

Q: Given a point cloud $X \subseteq \mathbb{R}^d$, how to define $\nabla_X D_{\mathrm{Rips}}(X)$?

Idea: Let's go back to the PD construction...

Computation with matrix reduction



Input: simplicial filtration

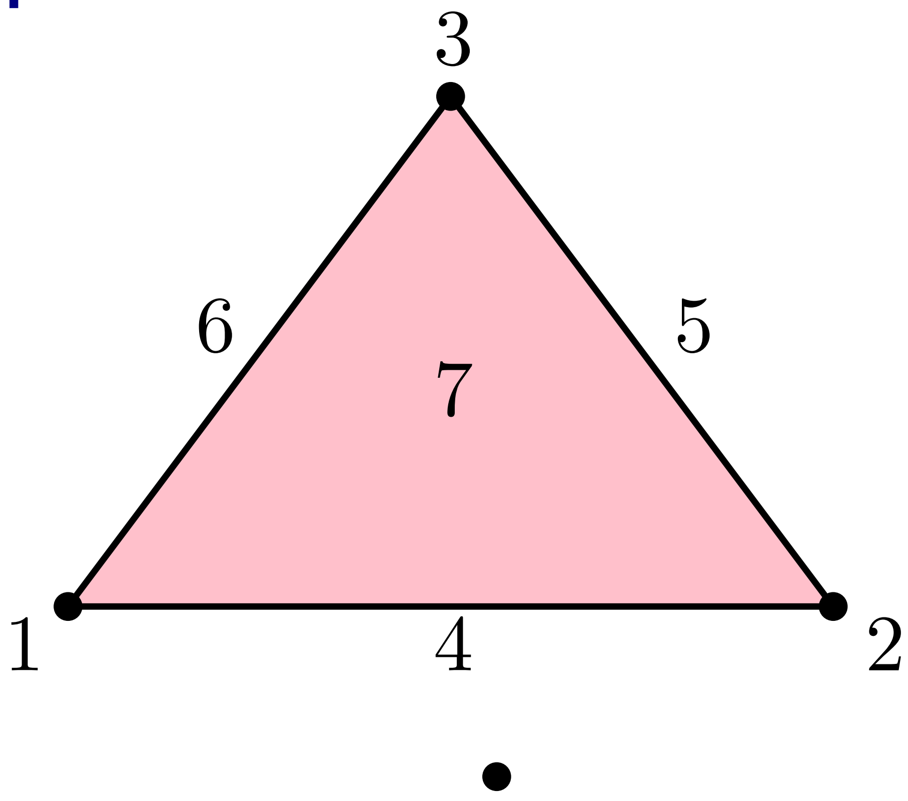
(Persistent) homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*

Computation with matrix reduction

Input: simplicial filtration



(Persistent) homology can be computed by using the fact that each simplex is either:

- *positive*, i.e., it *creates a new homology class*
- *negative*, i.e., it *destroys an homology class*

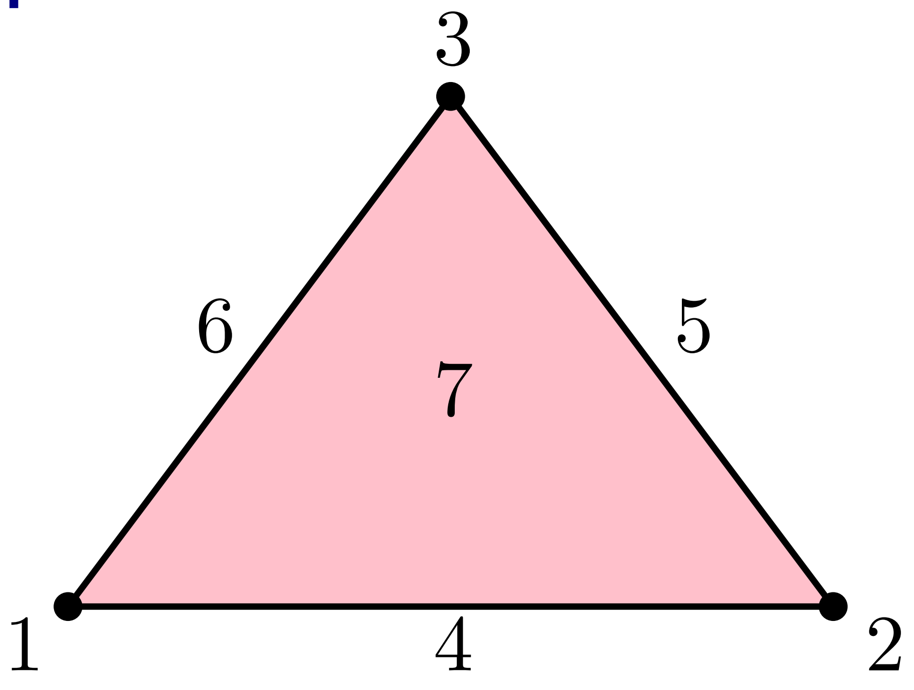
Computation with matrix reduction

Input: simplicial filtration

(Persistent) homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*

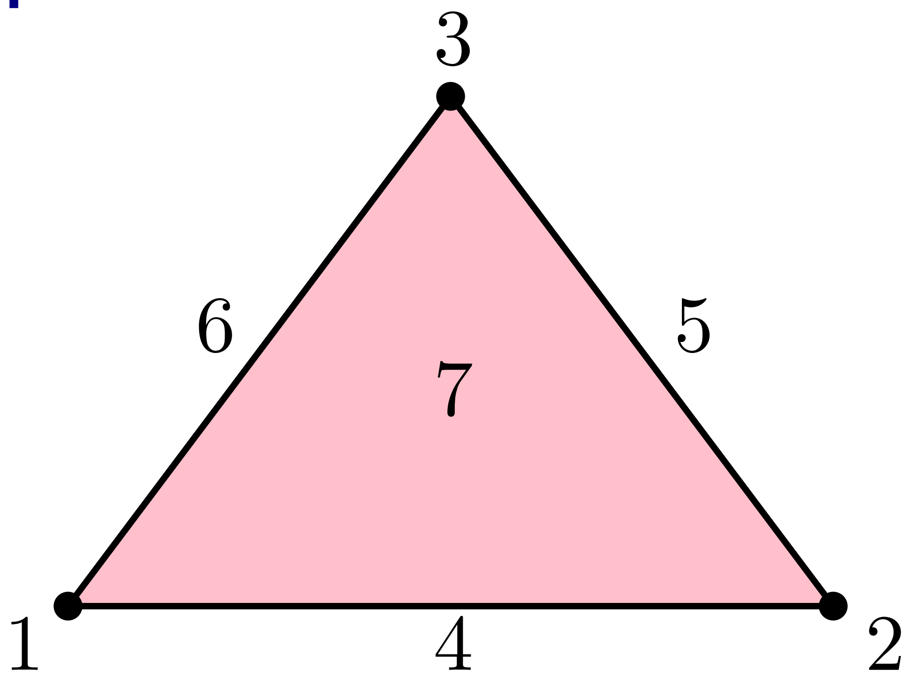


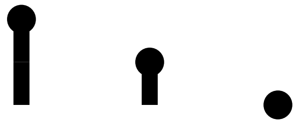
Computation with matrix reduction

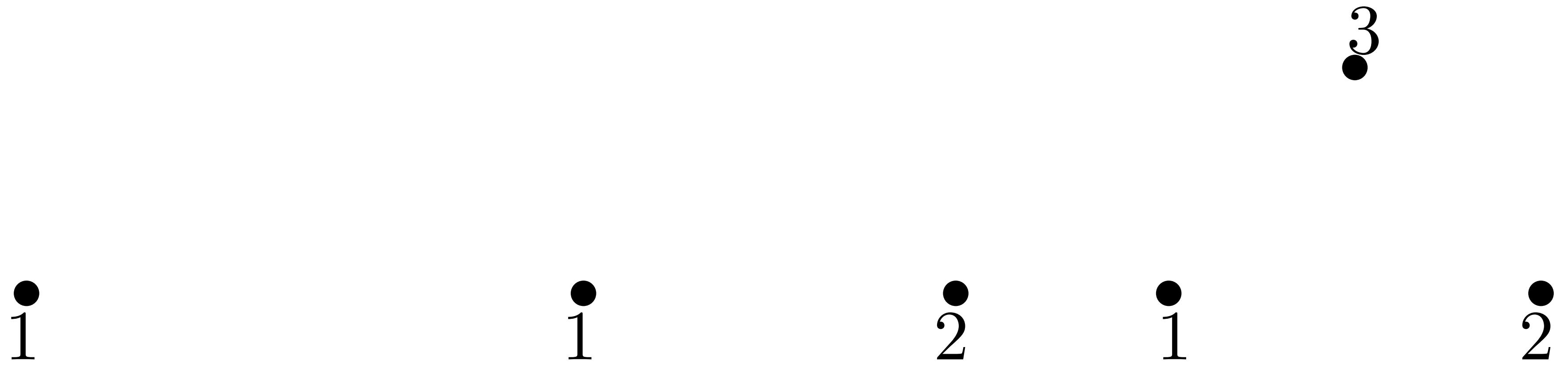
Input: simplicial filtration

(Persistent) homology can be computed by using the fact that each simplex is either:

- *positive*, i.e., it *creates a new homology class*
- *negative*, i.e., it *destroys an homology class*

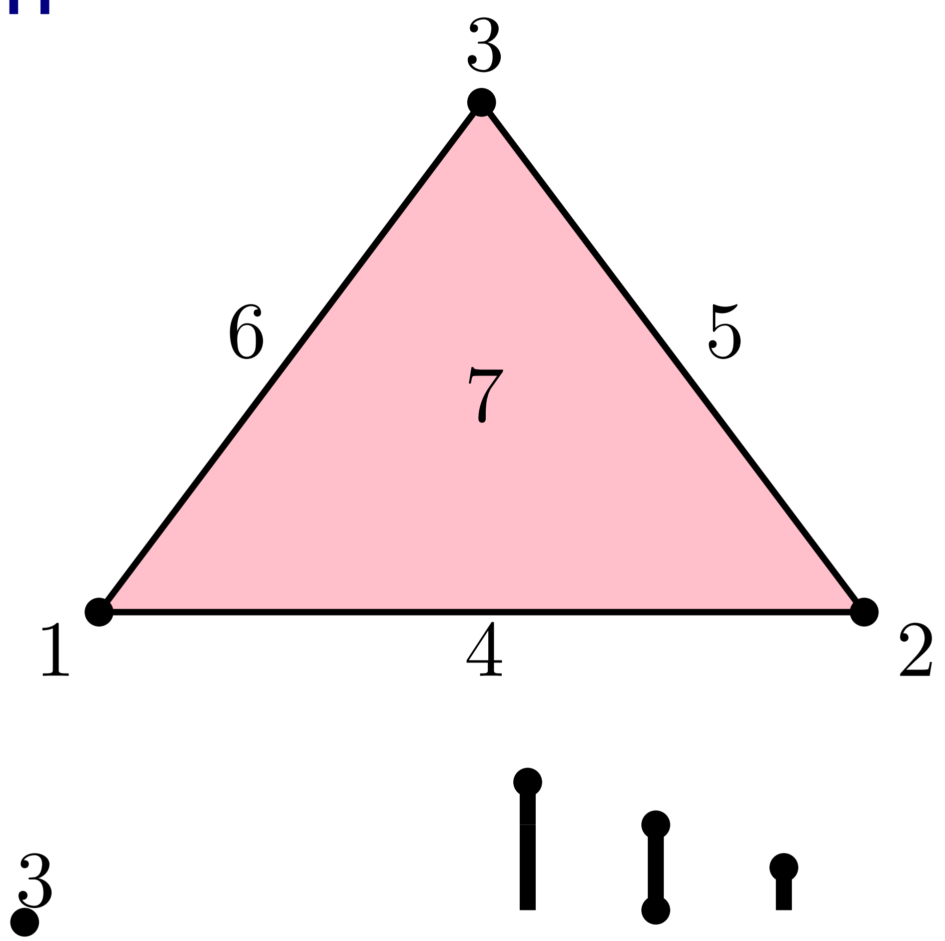






Computation with matrix reduction

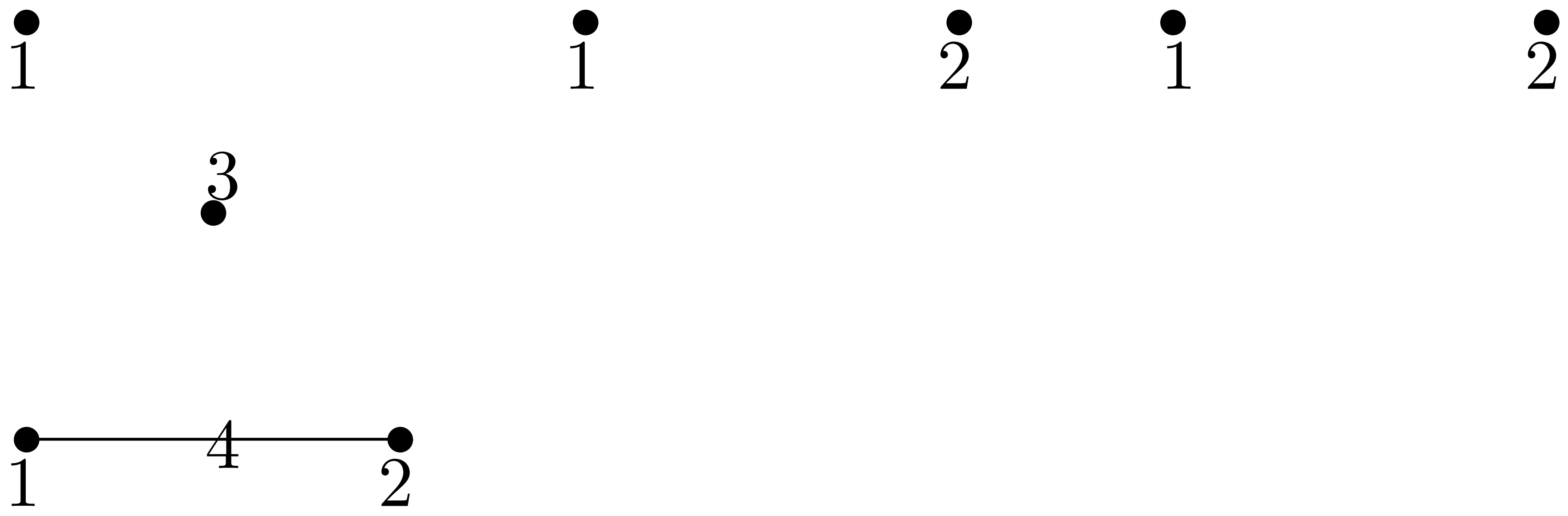
Input: simplicial filtration



(Persistent) homology can be computed by using the fact that each simplex is either:

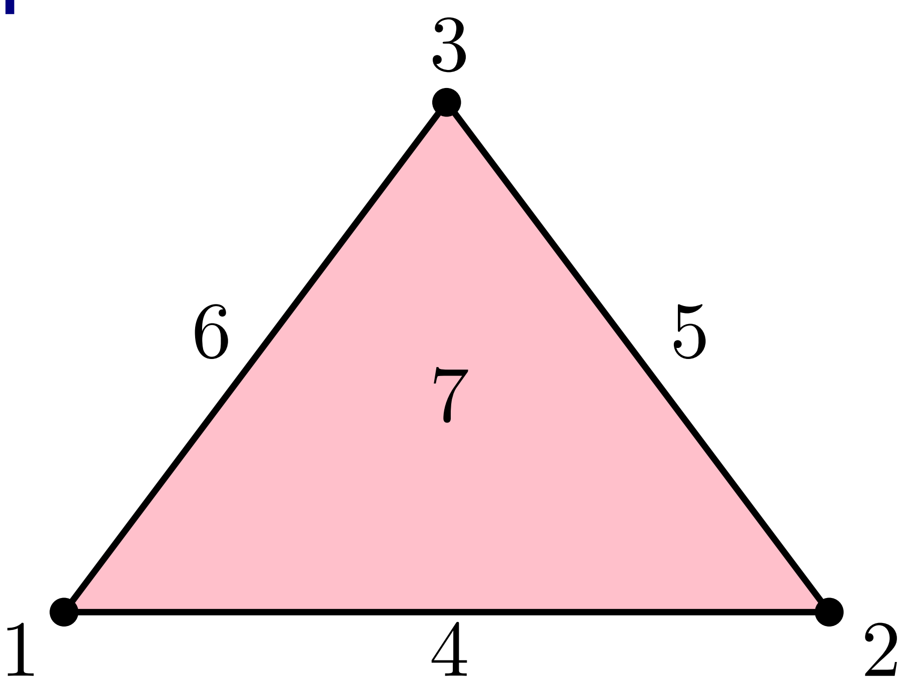
positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*



Computation with matrix reduction

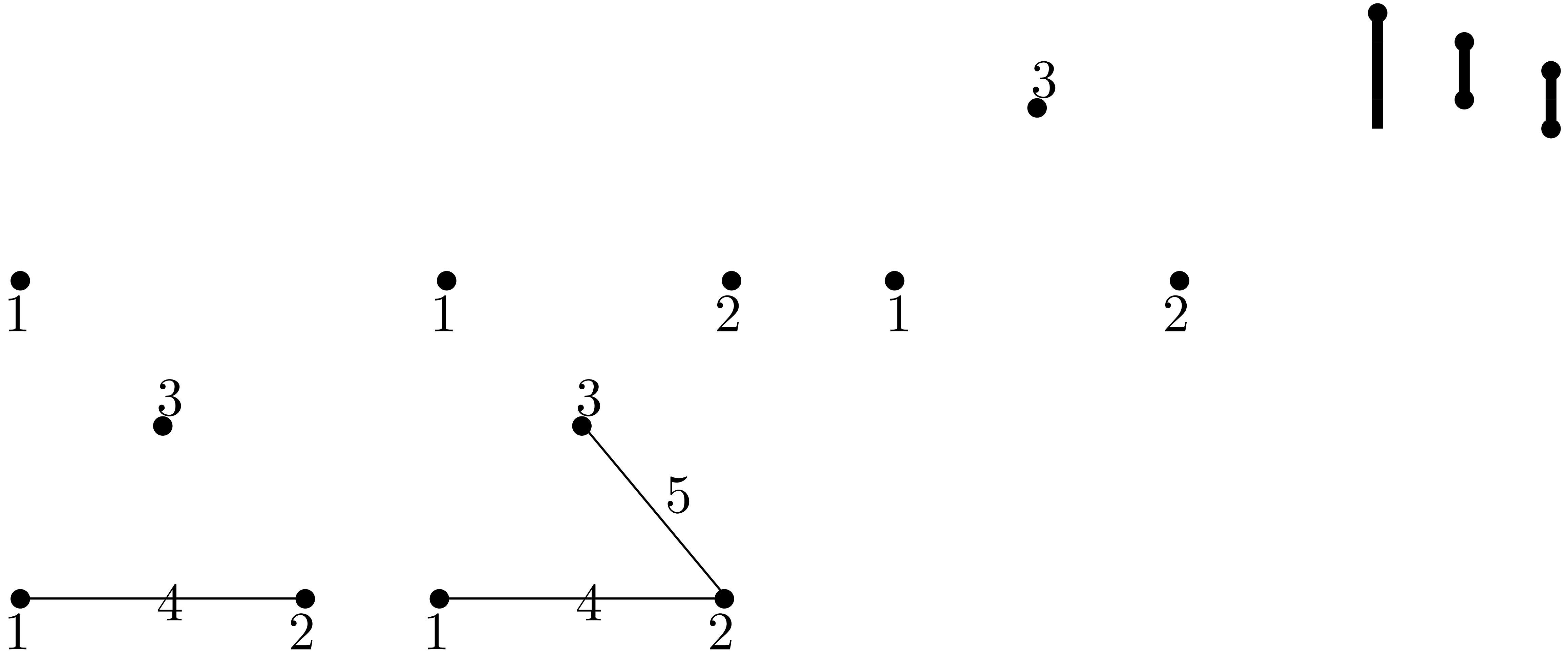
Input: simplicial filtration



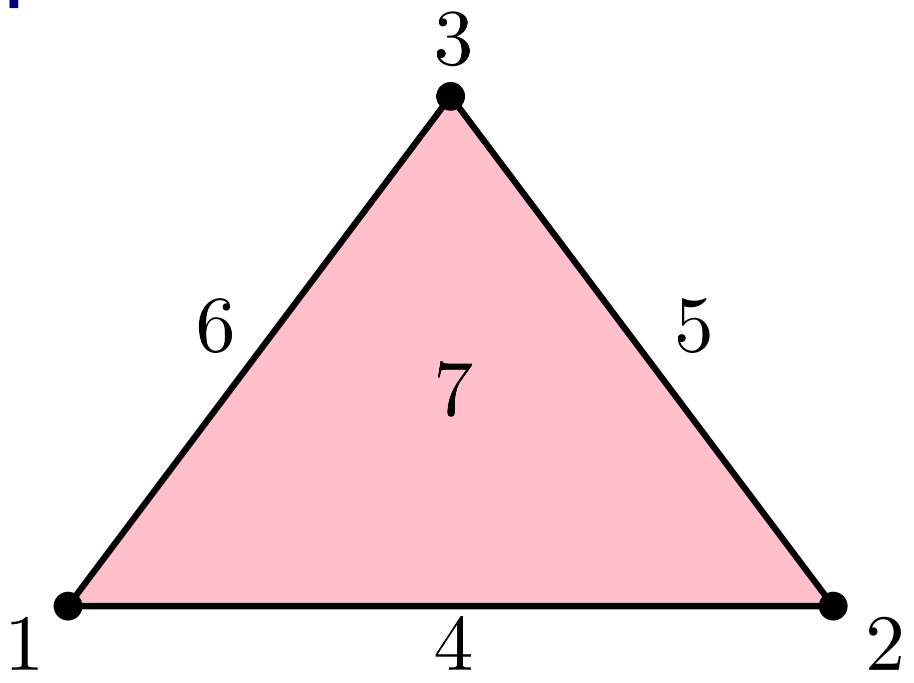
(Persistent) homology can be computed by using the fact that each simplex is either:

positive, i.e., it *creates a new homology class*

negative, i.e., it *destroys an homology class*



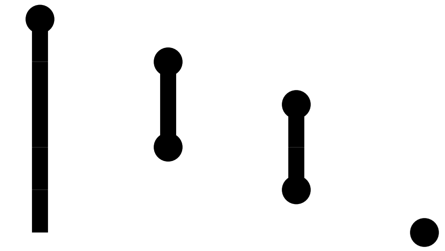
Computation with matrix reduction

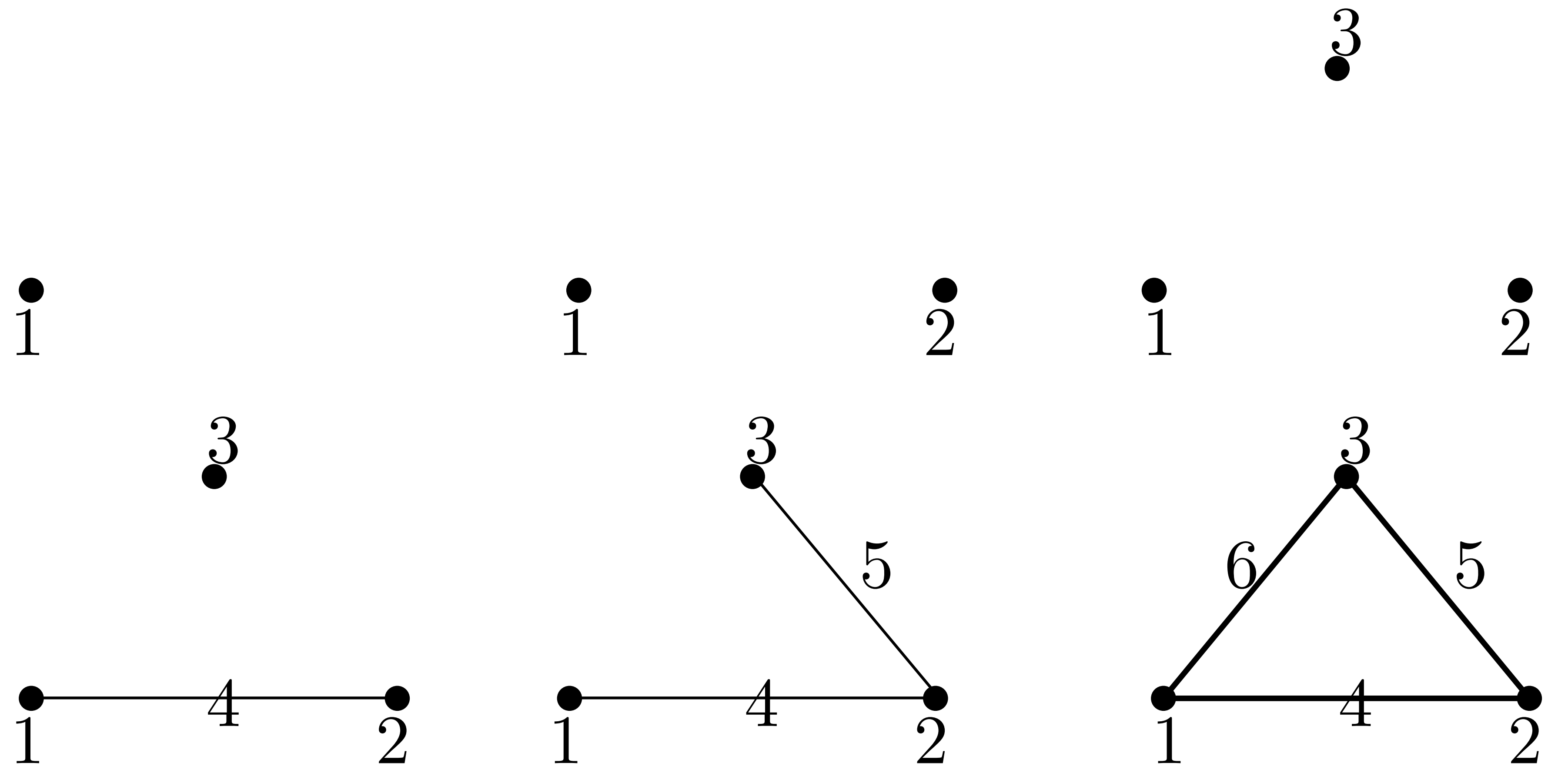


Input: simplicial filtration

(Persistent) homology can be computed by using the fact that each simplex is either:

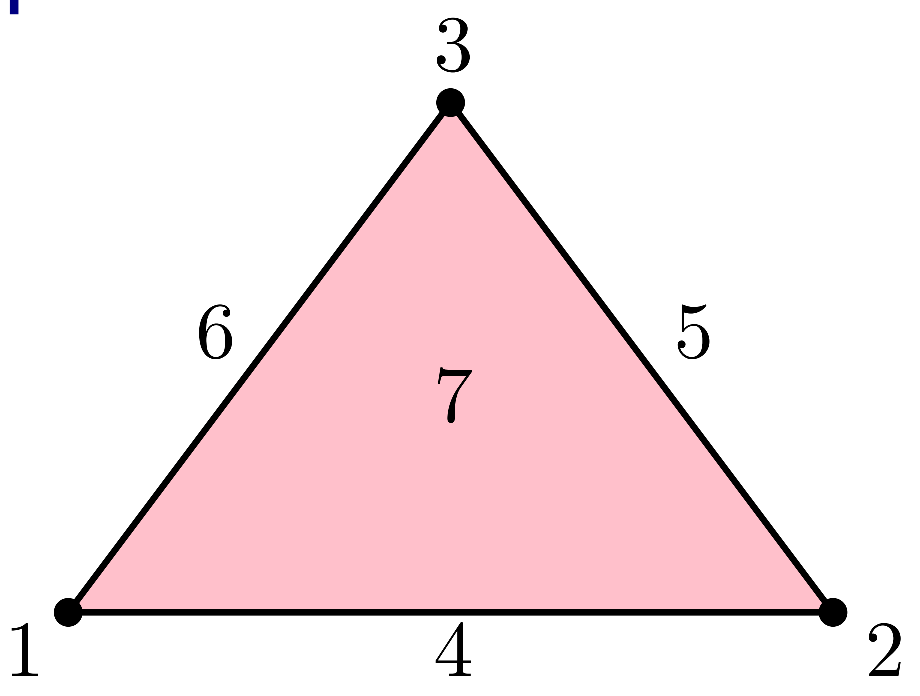
- *positive*, i.e., it *creates a new homology class*
- *negative*, i.e., it *destroys an homology class*





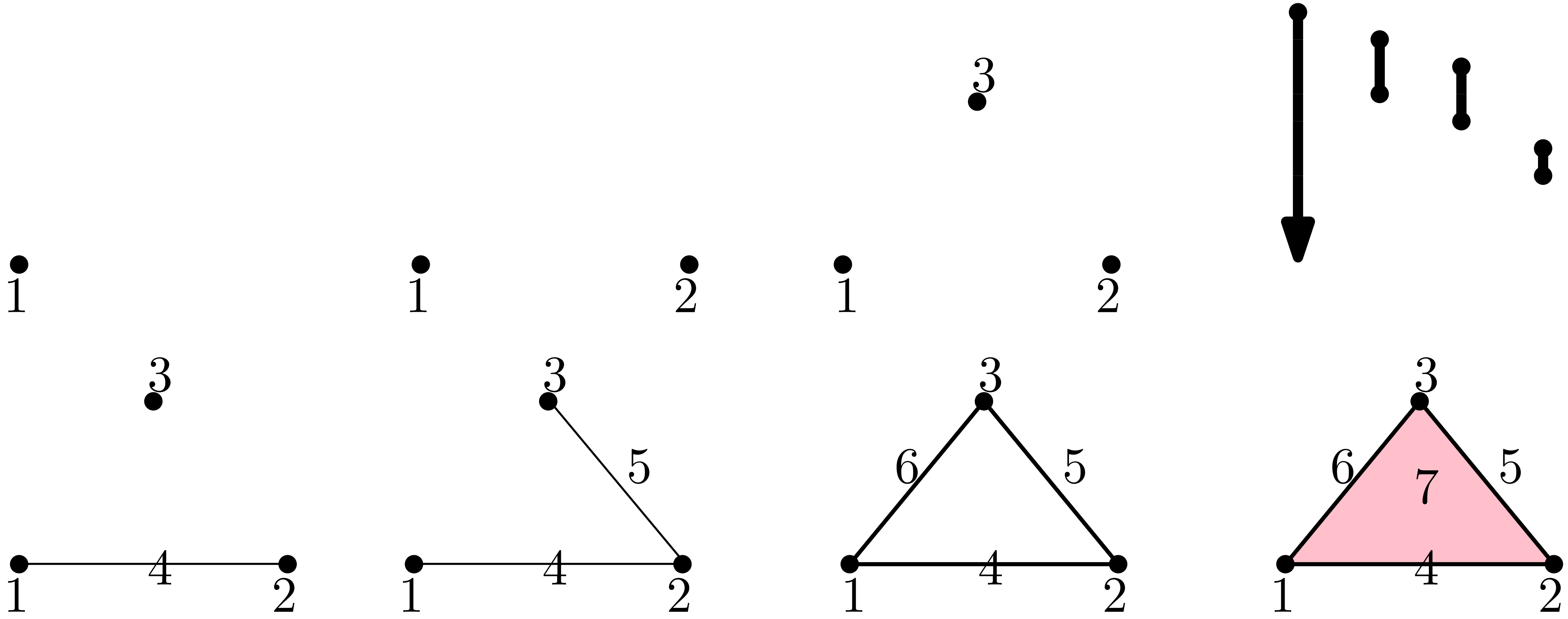
Computation with matrix reduction

Input: simplicial filtration

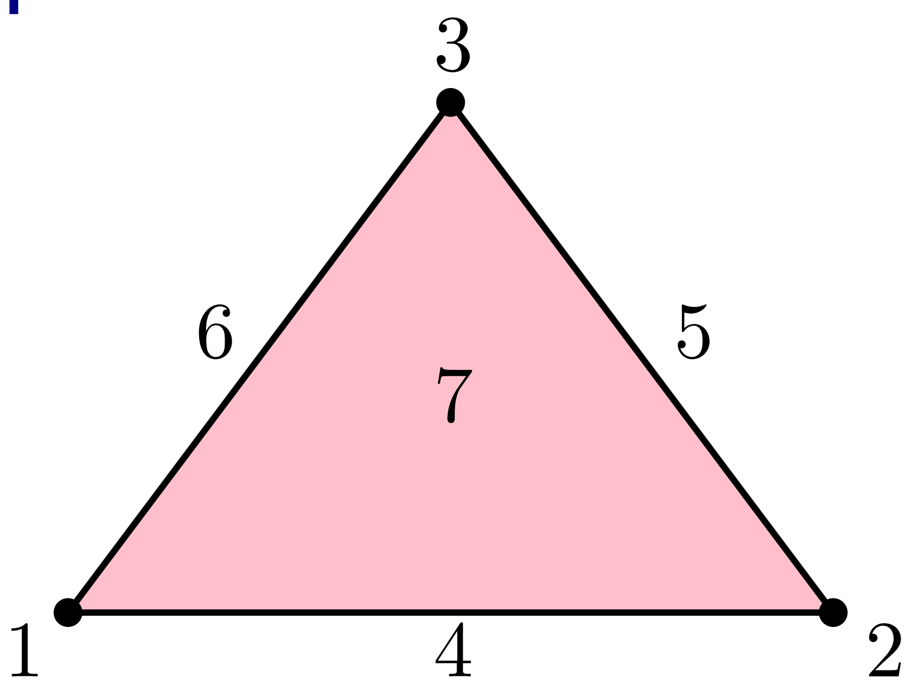


(Persistent) homology can be computed by using the fact that each simplex is either:

- *positive*, i.e., it *creates a new homology class*
- *negative*, i.e., it *destroys an homology class*



Computation with matrix reduction



Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

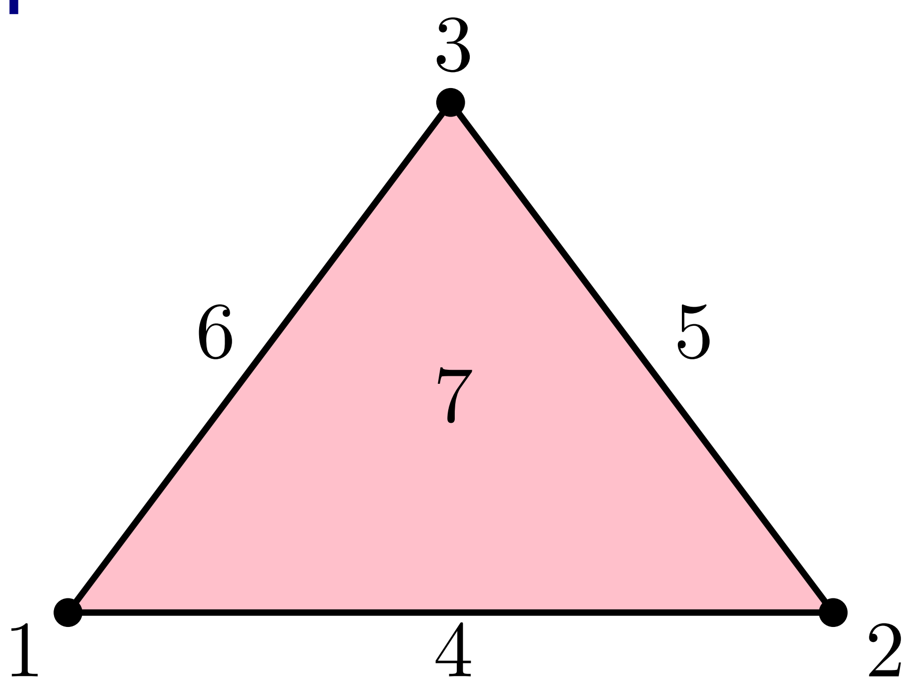
○ simplex pairs give finite intervals:
$[2,4)$, $[3,5)$, $[6,7)$

□ unpaired simplices give infinite intervals: $[1,+\infty)$

	1	2	3	4	5	6	7
1				$*$		$*$	
2				$*$	$*$		
3					$*$	$*$	
4							$*$
5							$*$
6							$*$
7							

	1	2	3	4	5	6	7
1				$*$			
2				1	$*$		
3					1		
4							$*$
5							$*$
6							1
7							

Computation with matrix reduction



Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

○ simplex pairs give finite intervals:
$[2,4)$, $[3,5)$, $[6,7)$

□ unpaired simplices give infinite intervals: $[1,+\infty)$

A persistence diagram D is made of all $(\mathcal{F}(\sigma_+),\mathcal{F}(\sigma_-)) \in \mathbb{R}^2$ where σ_+ (resp. σ_-) is positive (resp. negative), and $\mathcal{F}$ is the filtration function.

	1	2	3	4	5	6	7
1				*			
2				1	*		
3					1		
4							*
5							*
6							1
7							

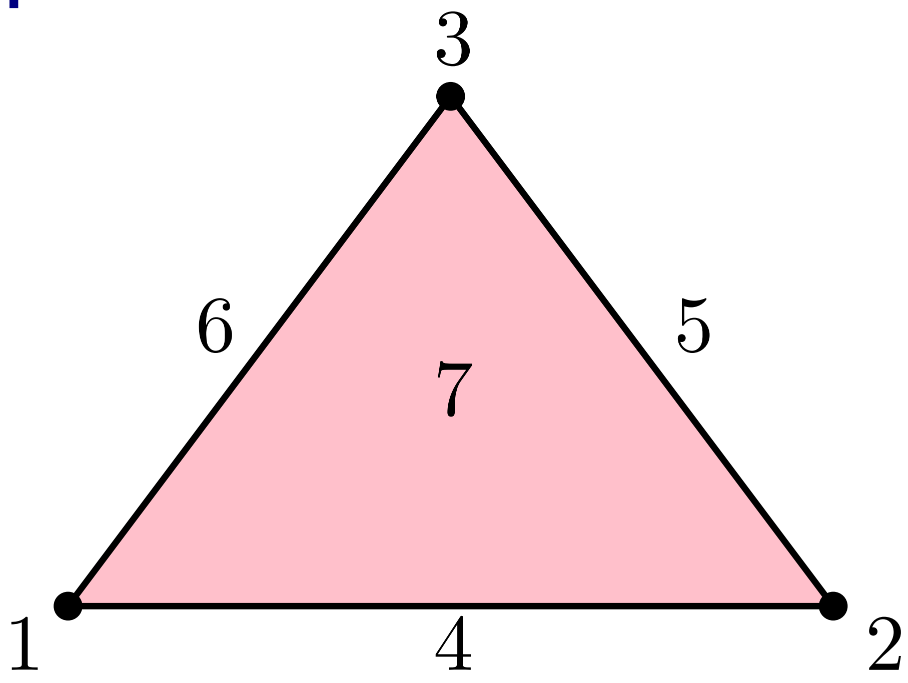
Computation with matrix reduction

Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

○ simplex pairs give finite intervals:
$[2,4)$, $[3,5)$, $[6,7)$

□ unpaired simplices give infinite intervals: $[1,+\infty)$



A persistence diagram D is made of all $(\mathcal{F}(\sigma_+),\mathcal{F}(\sigma_-)) \in \mathbb{R}^2$ where σ_+ (resp. σ_-) is positive (resp. negative), and $\mathcal{F}$ is the filtration function.

Thus we can define the gradient of a point $p = (\mathcal{F}(\sigma_+),\mathcal{F}(\sigma_-)) \in D$ as

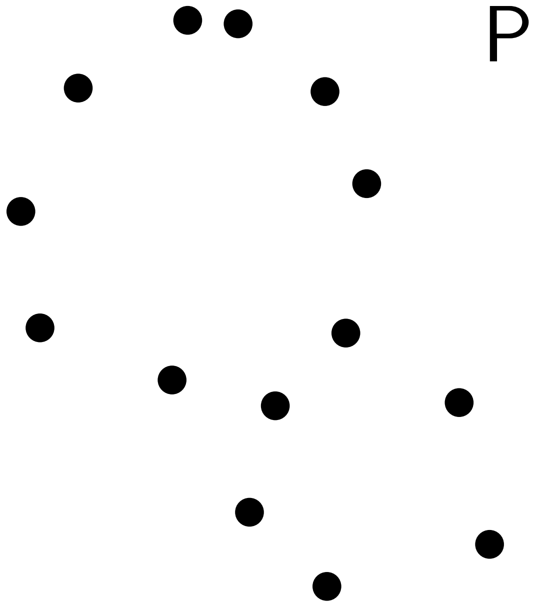
$$\nabla p = [\nabla\mathcal{F}(\sigma_+), \nabla\mathcal{F}(\sigma_-)]$$

	1	2	3	4	5	6	7
$\boxed{1}$				$*$			
2				①	$*$		
3					①		
4							$*$
5							$*$
6							①
7							

Example: Vietoris-Rips gradient

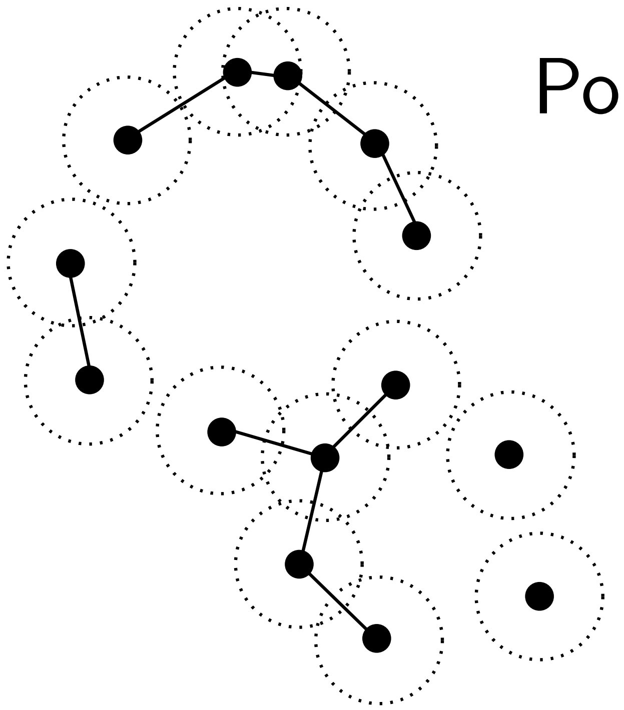
Q: Define and compute Vietoris-Rips gradient?

Example: Vietoris-Rips gradient

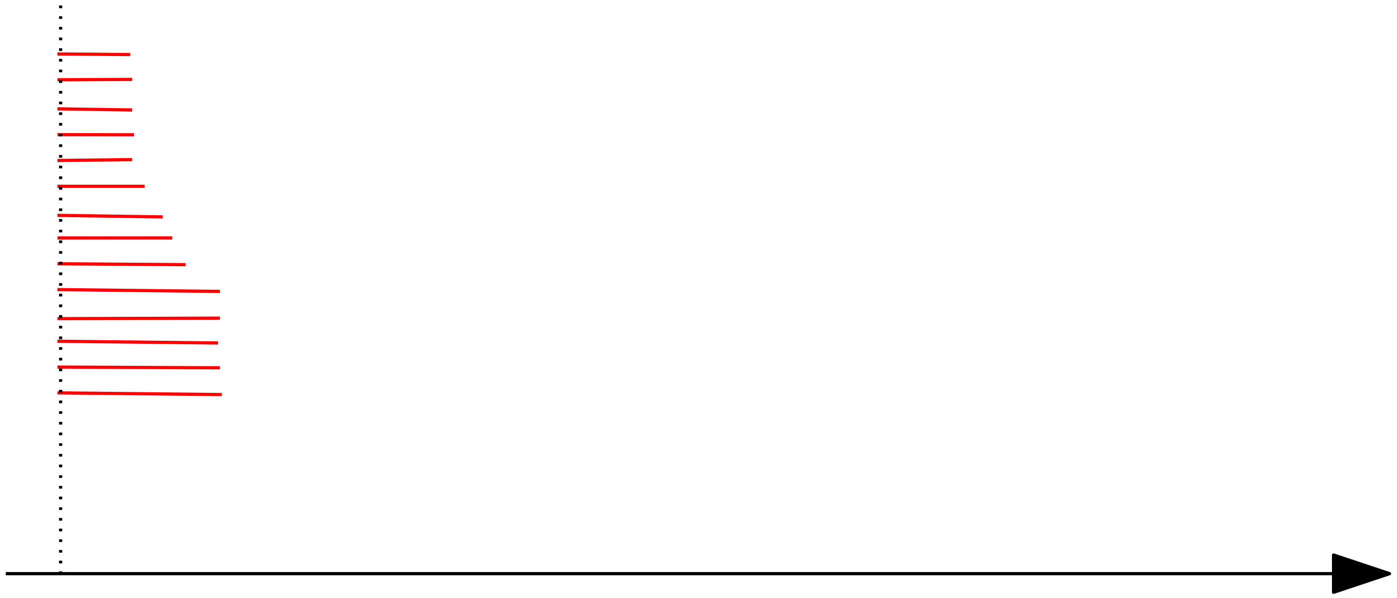


Point cloud $\hat{X}_n$

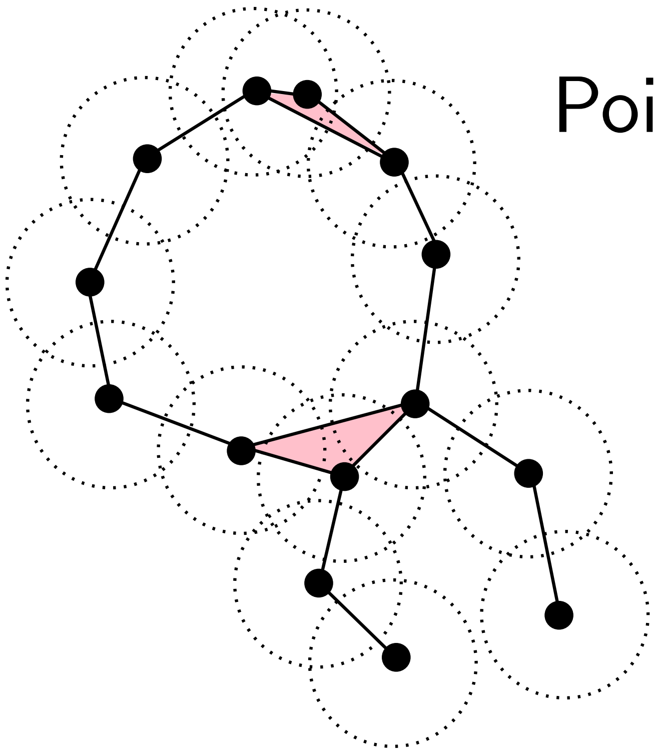
Example: Vietoris-Rips gradient



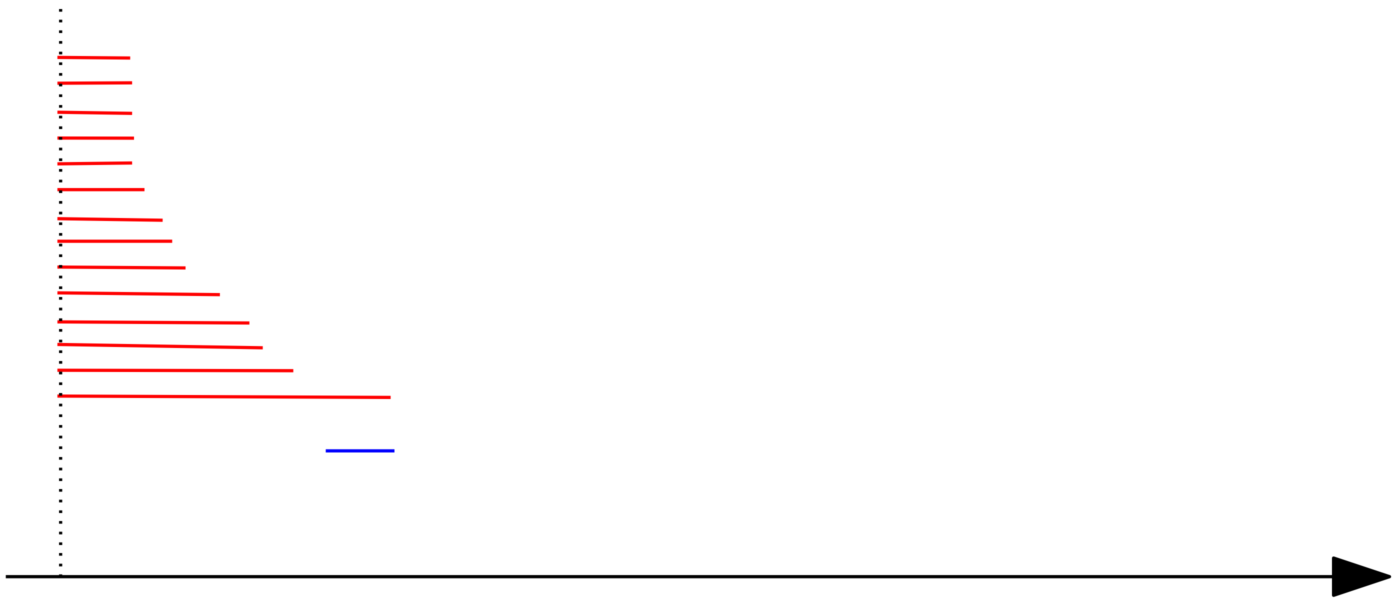
Point cloud $\hat{X}_n$



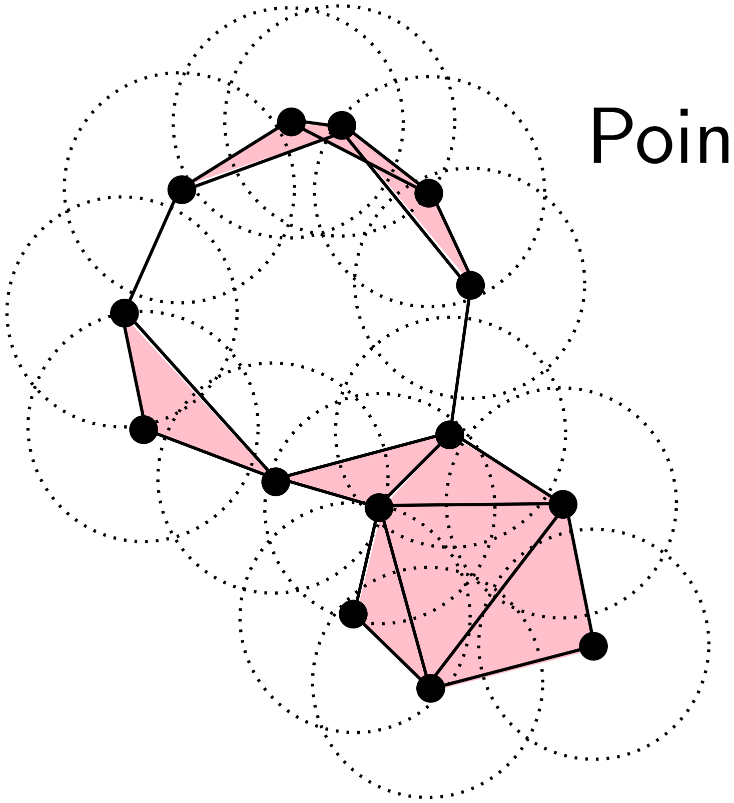
Example: Vietoris-Rips gradient



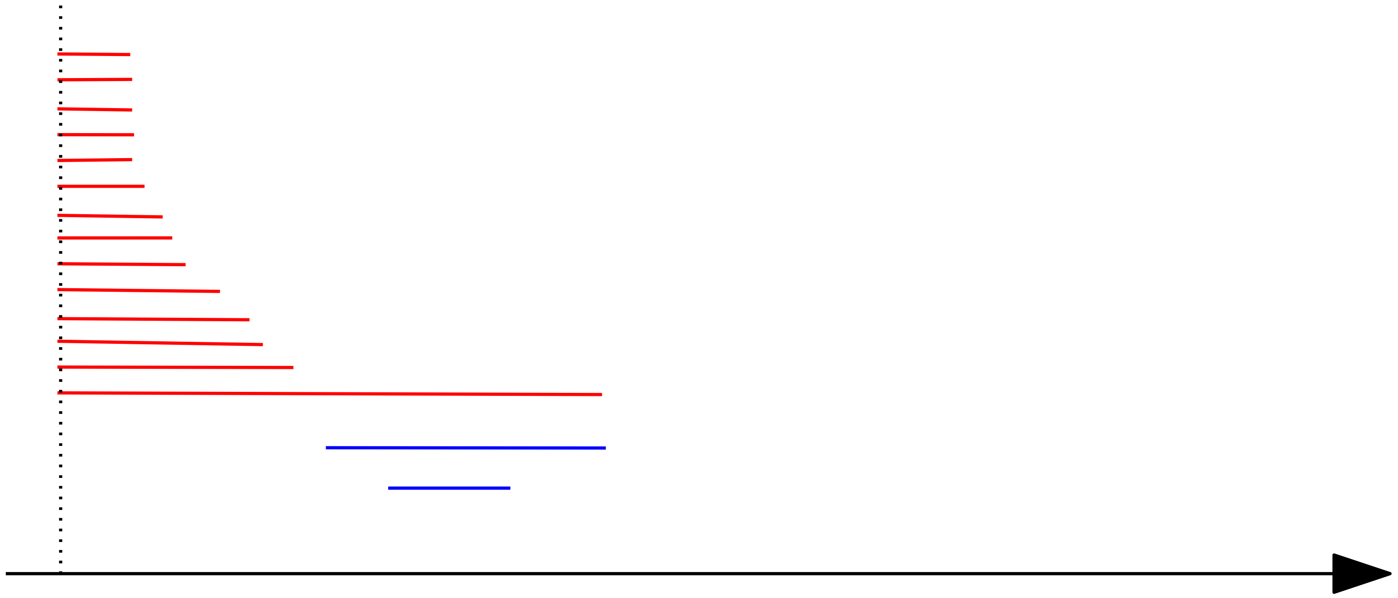
Point cloud $\hat{X}_n$



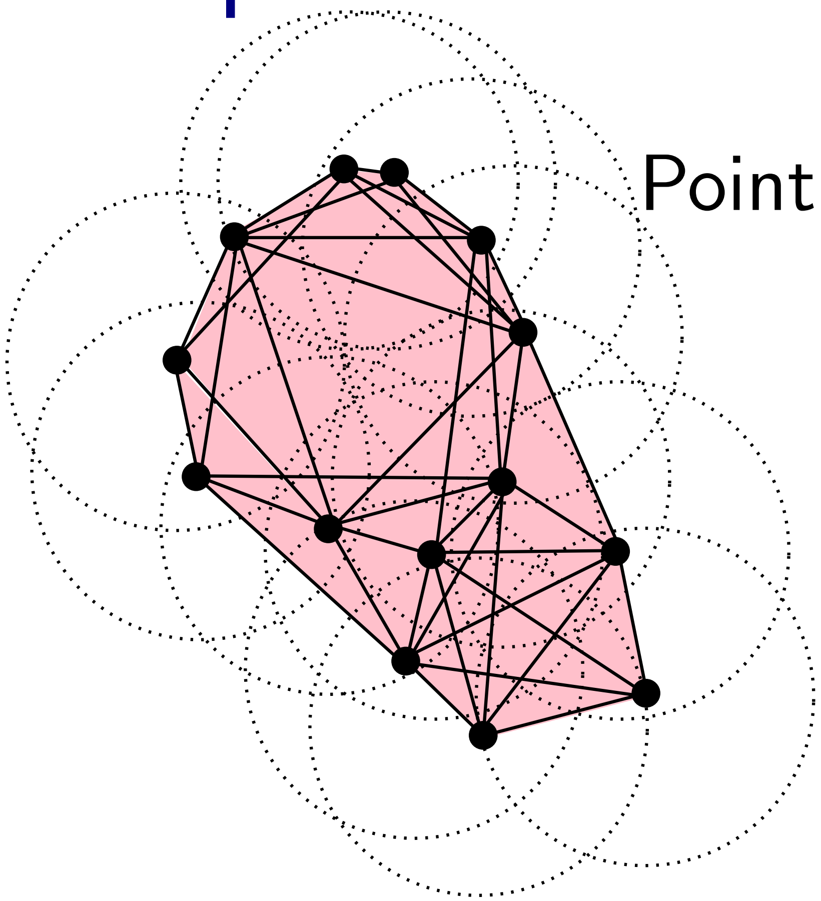
Example: Vietoris-Rips gradient



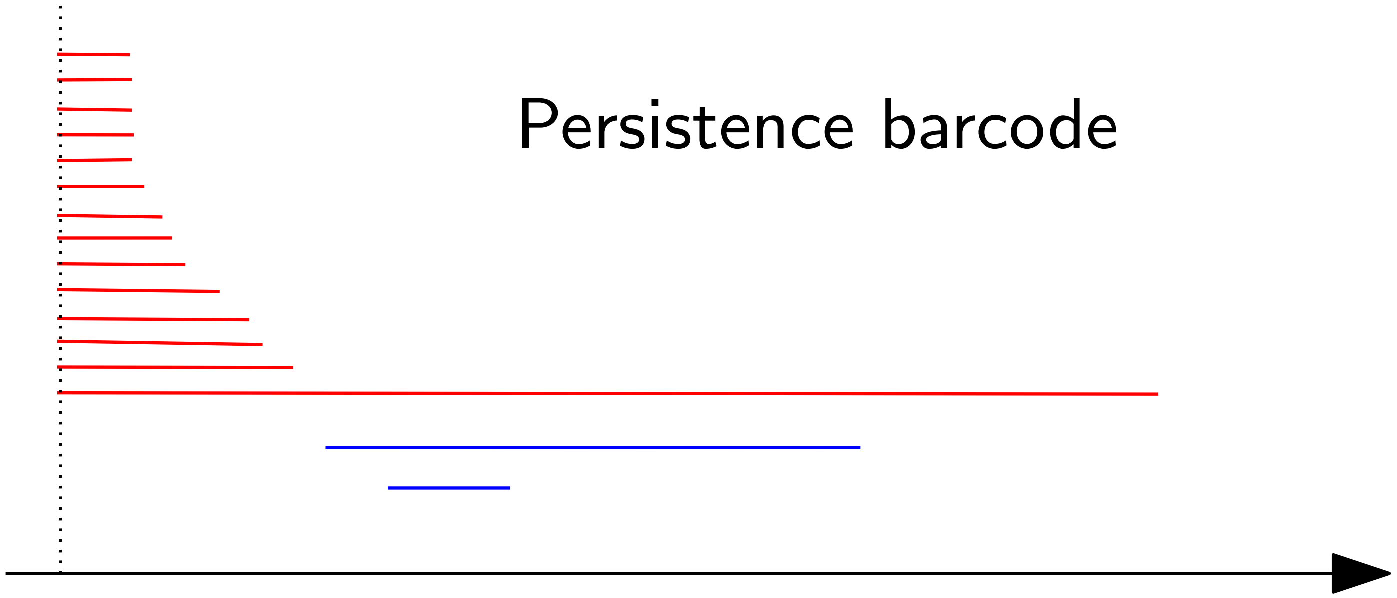
Point cloud $\hat{X}_n$



Example: Vietoris-Rips gradient

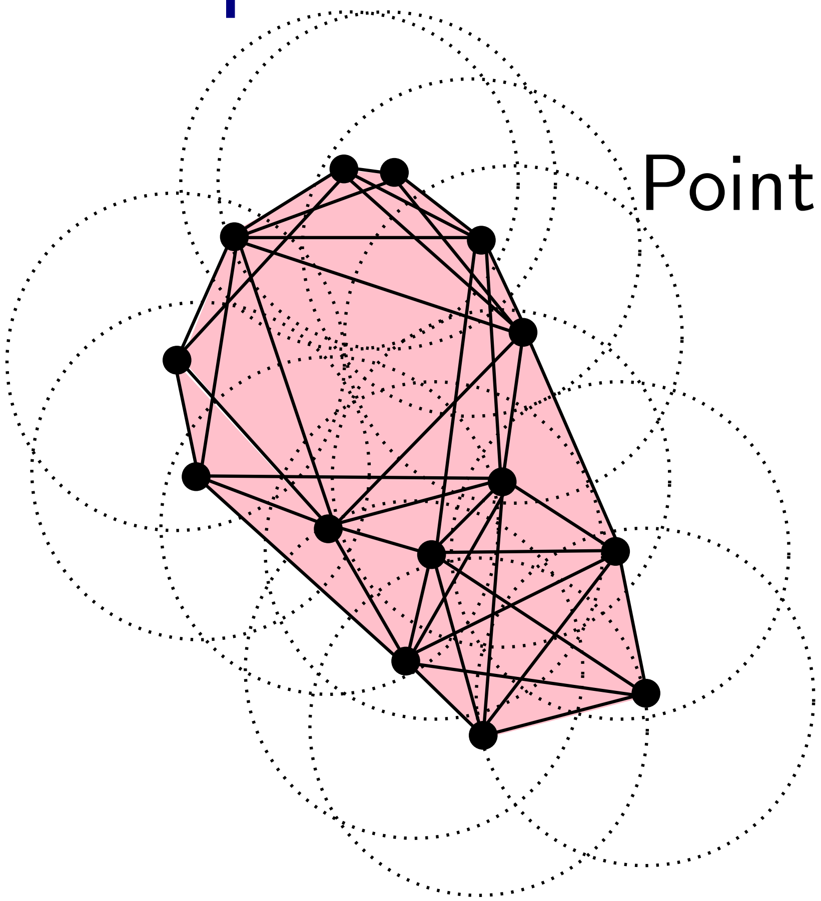


Point cloud $\hat{X}_n$

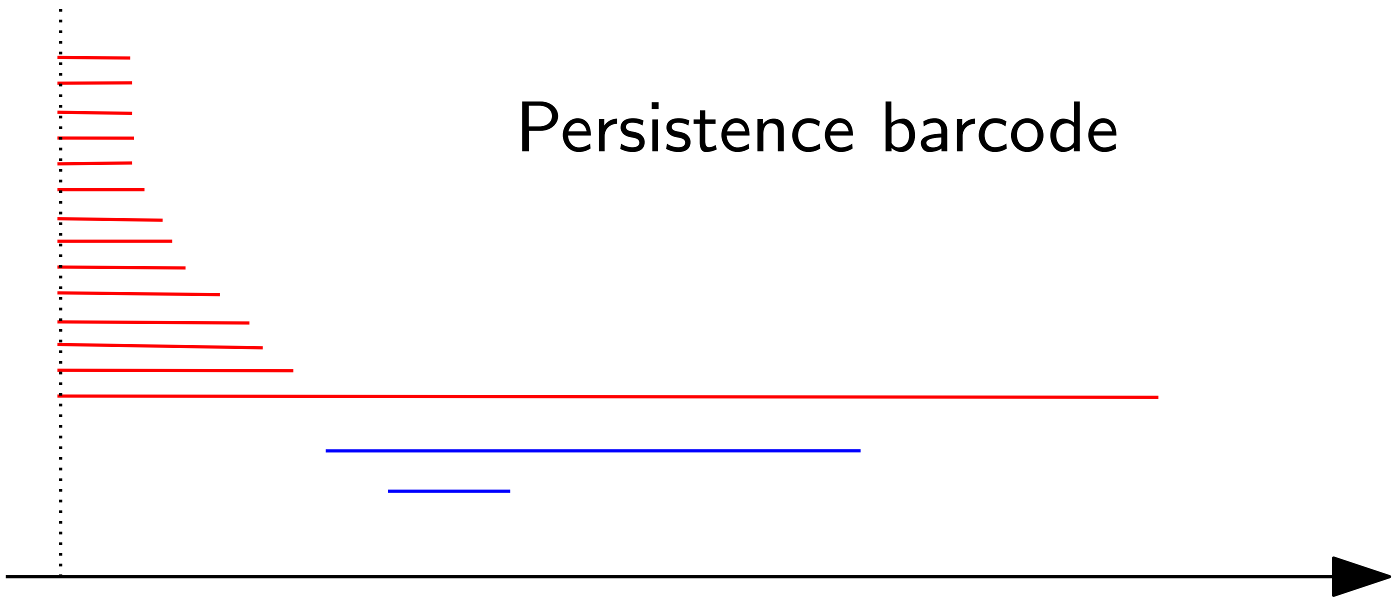


Persistence barcode

Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$

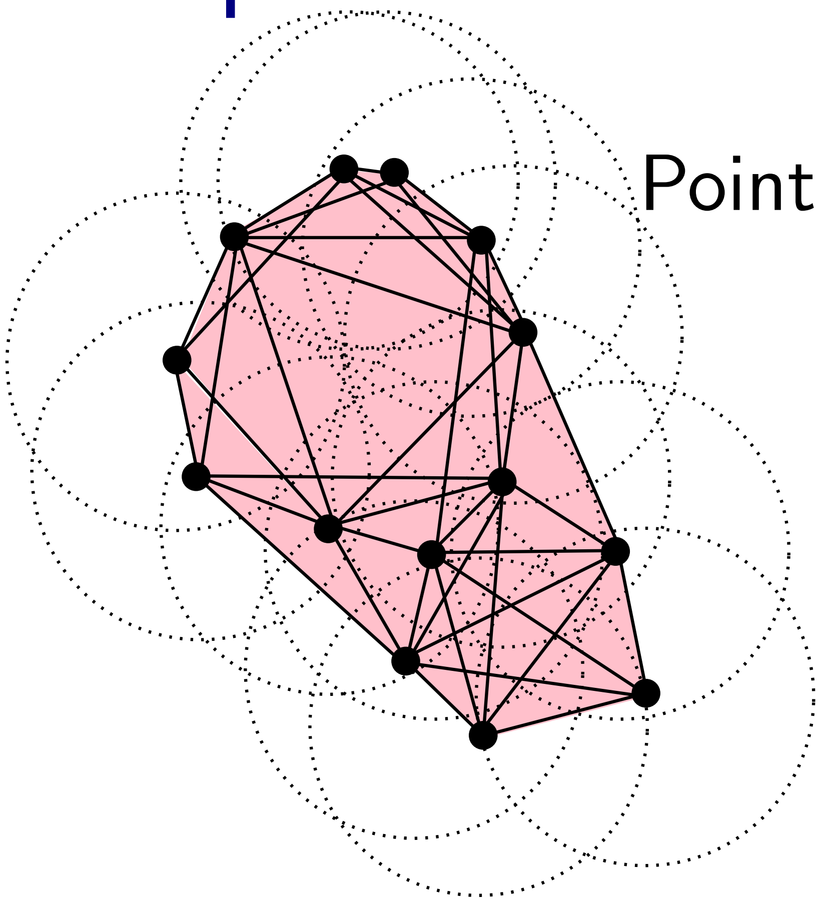


Persistence barcode

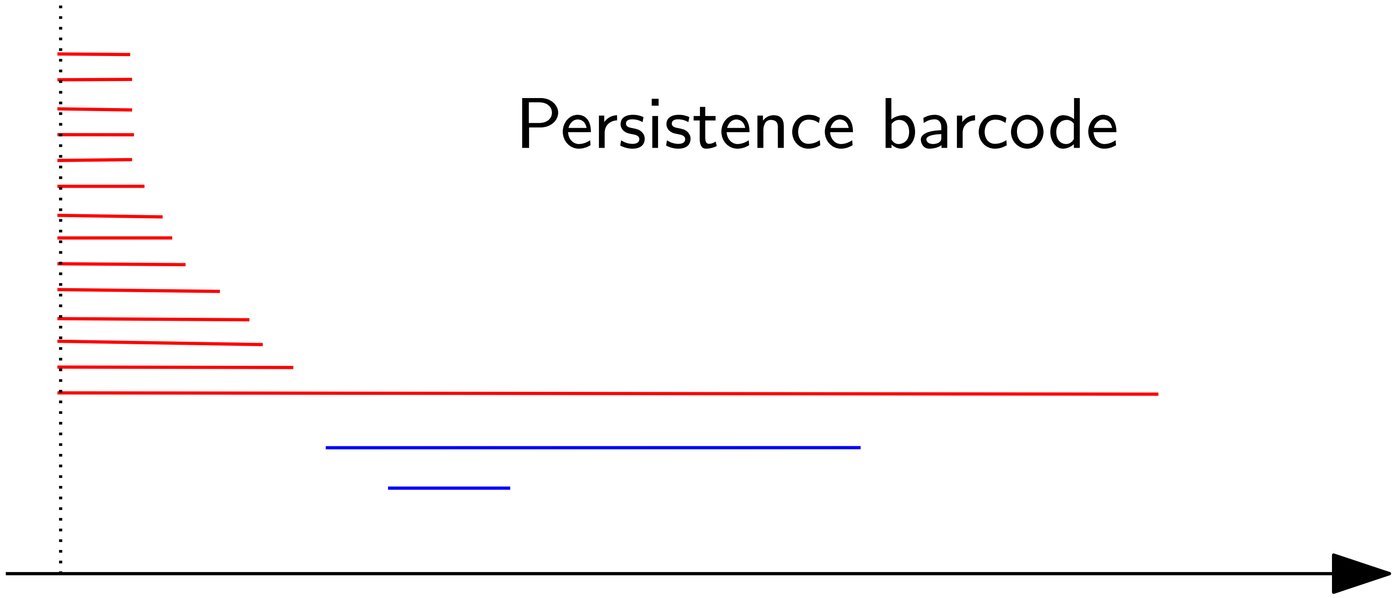
Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_{i,j} \|v_i - v_j\|$$

Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$



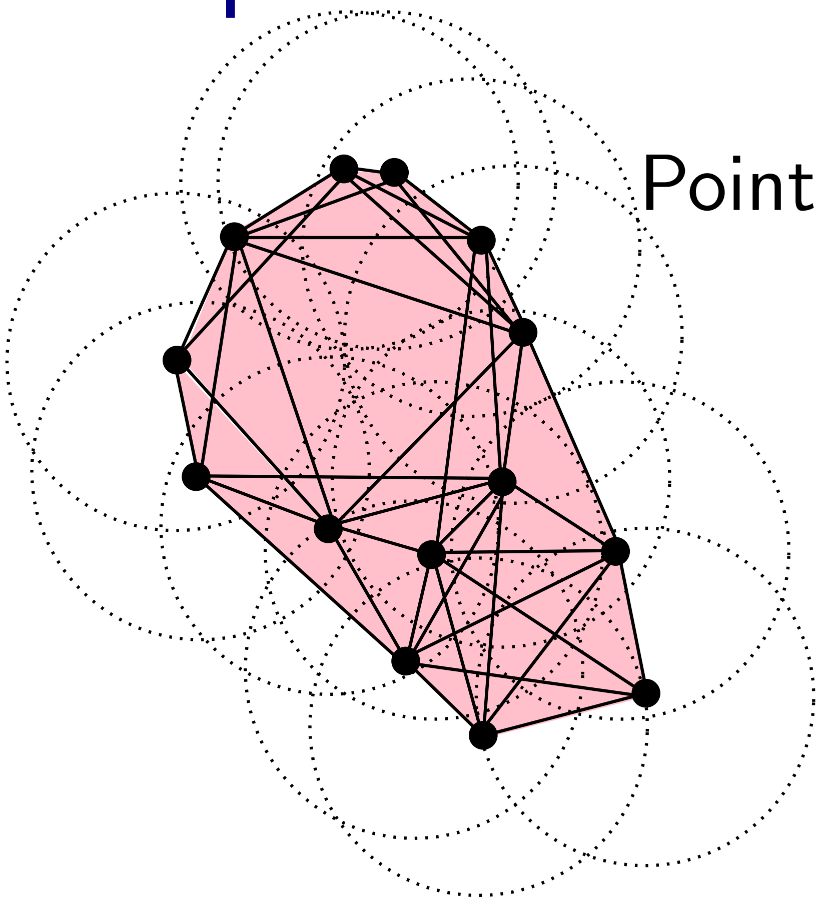
Persistence barcode

Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

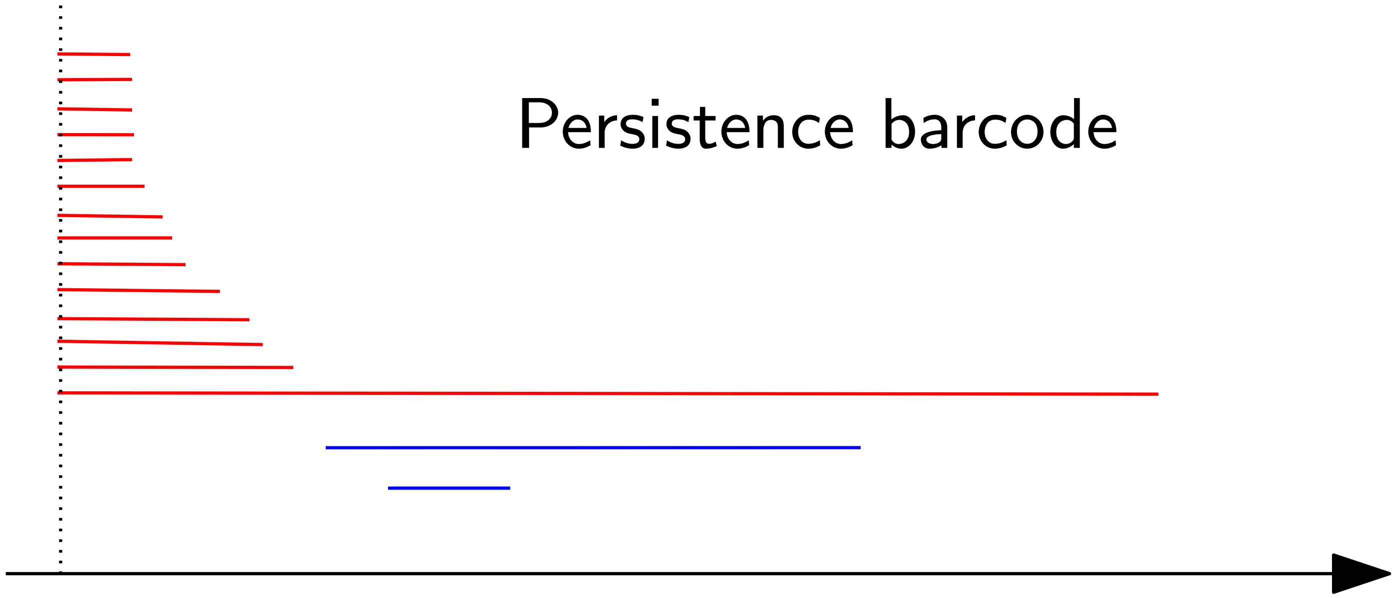
$$\mathcal{F}(\sigma) = \max_{i,j} \|v_i - v_j\|$$

Let $p = (\mathcal{F}(\sigma_+), \mathcal{F}(\sigma_-)) \in D_{\mathrm{Rips}}(X)$

Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$



Persistence barcode

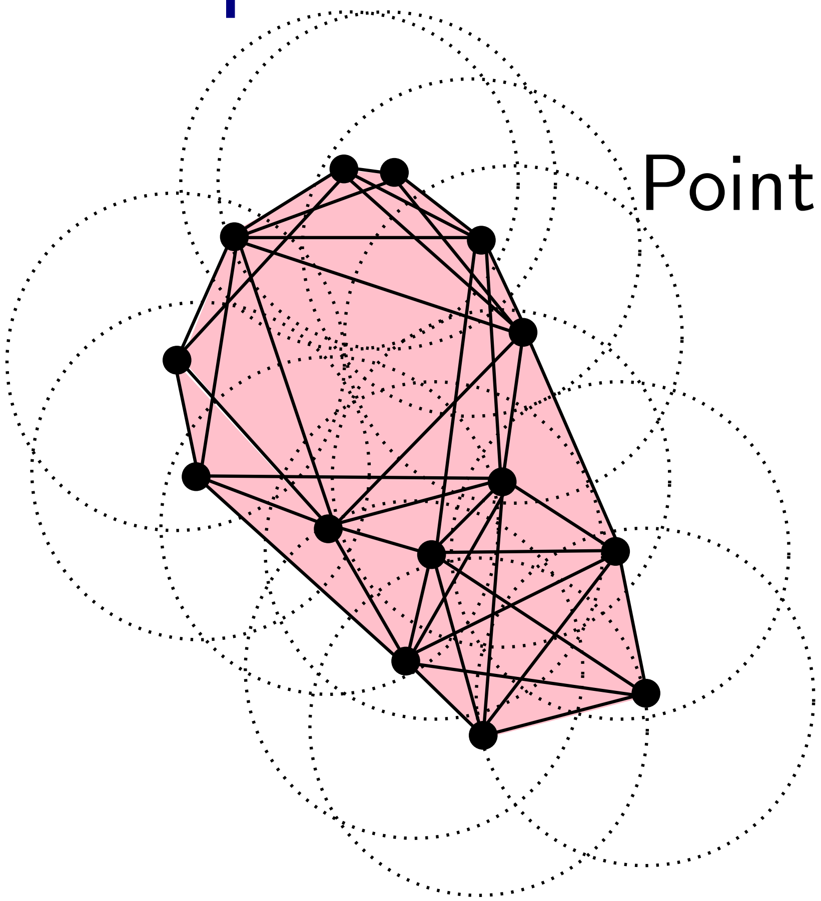
Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_{i,j} \|v_i - v_j\|$$

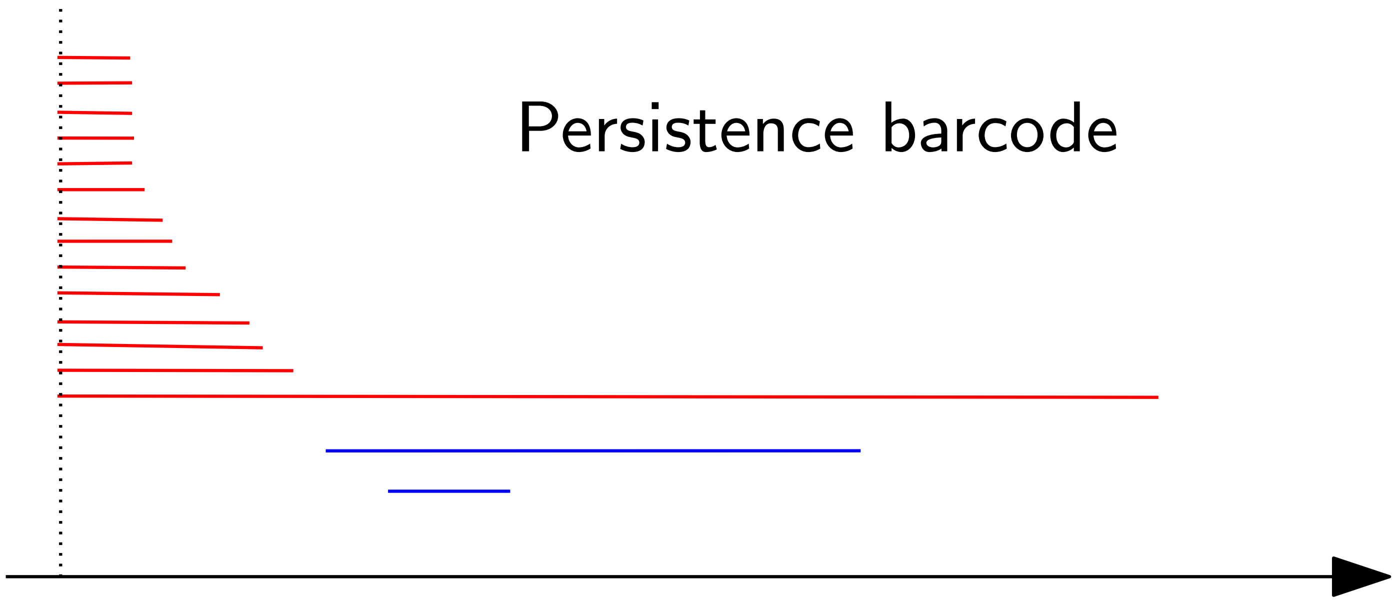
Let $p = (\mathcal{F}(\sigma_+), \mathcal{F}(\sigma_-)) \in D_{\mathrm{Rips}}(X)$

with $\sigma_+ = \{v_0, \ldots, v_k\}$ and $\sigma_- = \{w_0, \ldots, w_{k+1}\}$

Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$



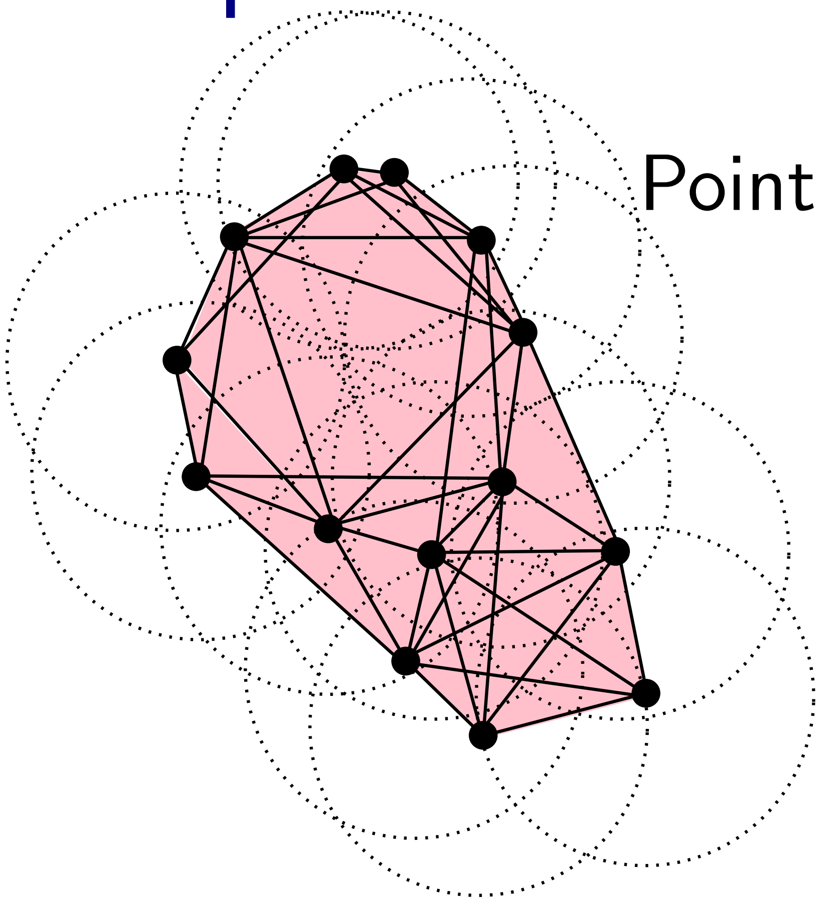
Persistence barcode

$$\nabla_X p = \left[\frac{\partial}{\partial X}\|v_{i^*} - v_{j^*}\|, \frac{\partial}{\partial X}\|w_{a^*} - w_{b^*}\|\right]$$

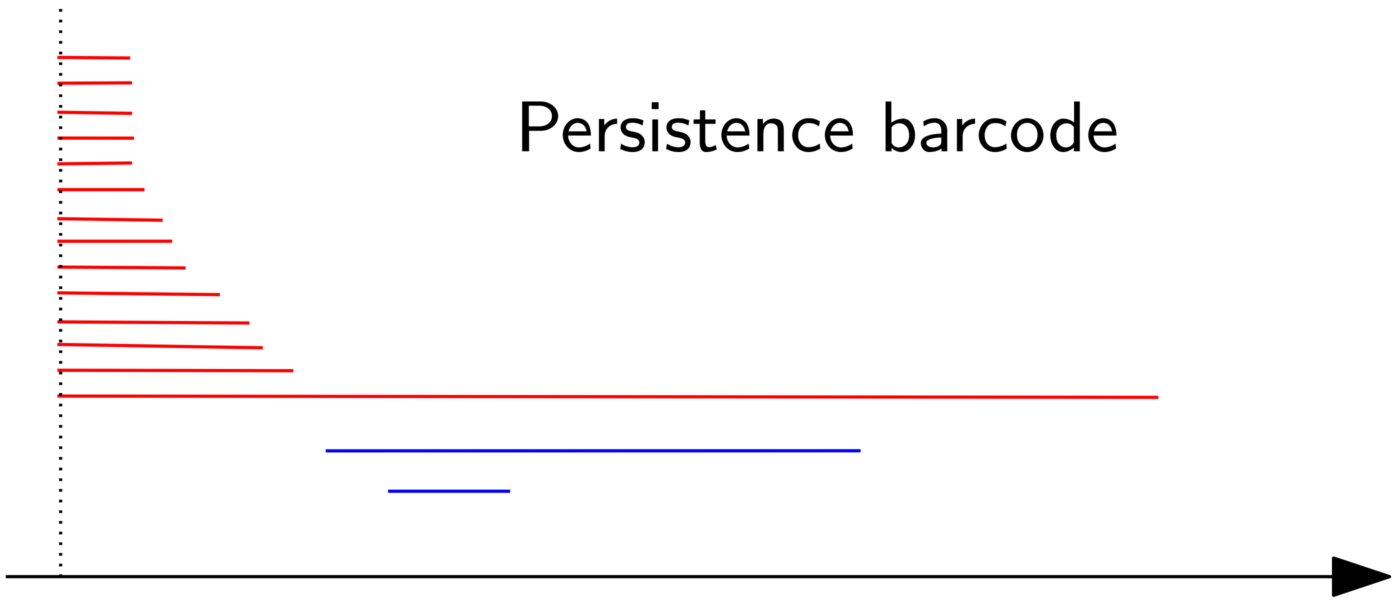
Let $p = (\mathcal{F}(\sigma_+), \mathcal{F}(\sigma_-)) \in D_{\mathrm{Rips}}(X)$

with $\sigma_+ = \{v_0, \ldots, v_k\}$ and $\sigma_- = \{w_0, \ldots, w_{k+1}\}$

Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$



Persistence barcode

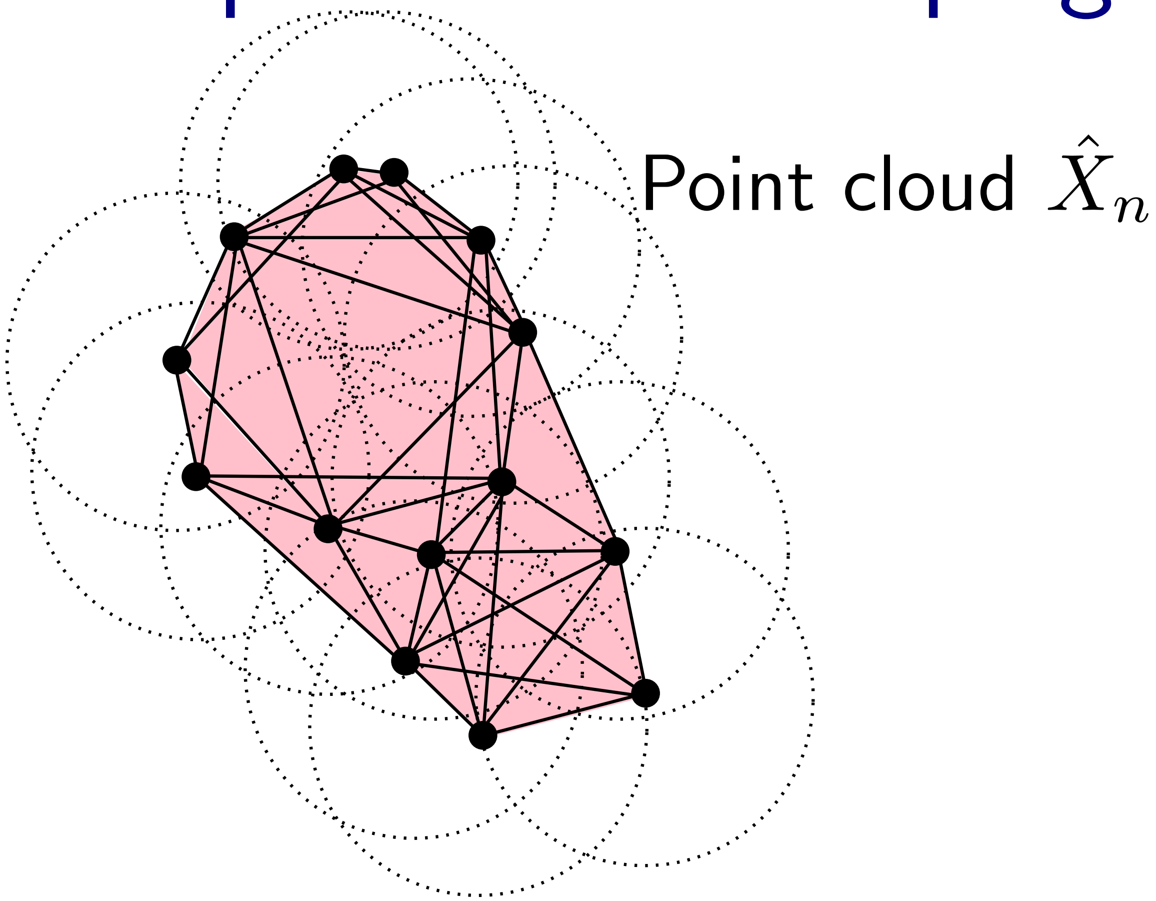
$$\nabla_X p = \left[\frac{\partial}{\partial X}\|v_{i^*} - v_{j^*}\|, \frac{\partial}{\partial X}\|w_{a^*} - w_{b^*}\|\right]$$

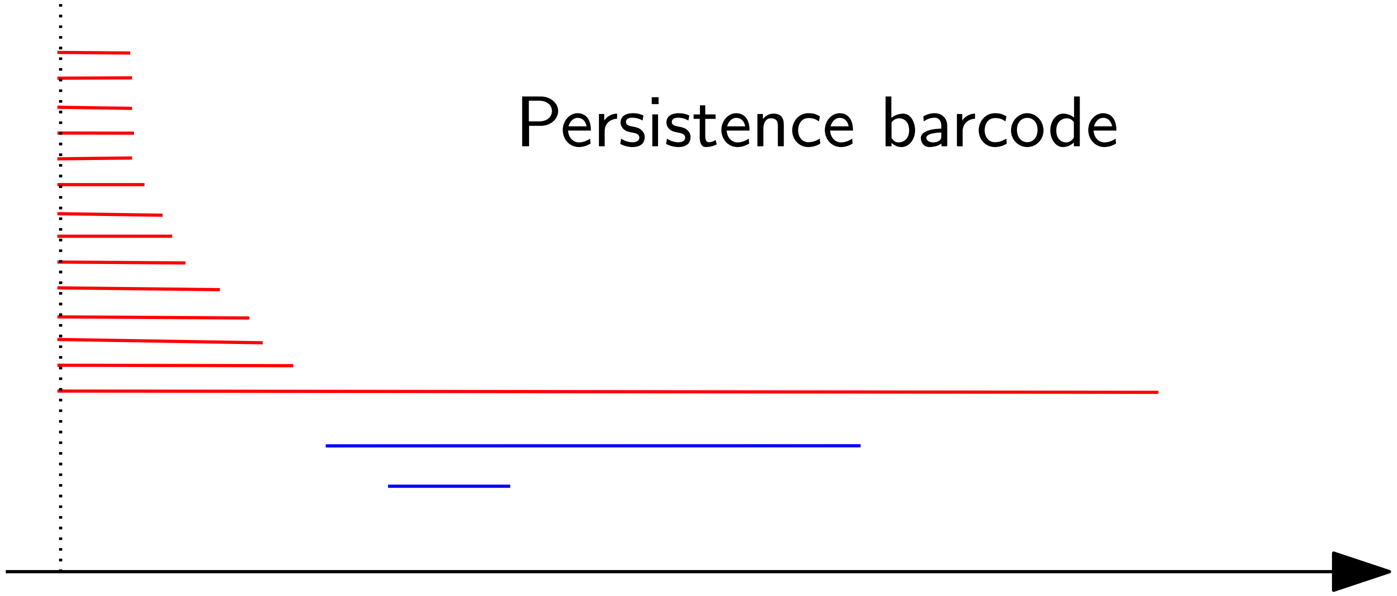
$\frac{\partial}{\partial v_i^{(d)}}\|v_{i^*} - v_{j^*}\| = (-)\frac{1}{\|v_{i^*} - v_{j^*}\|}(v_{i^*}^{(d)} - v_{j^*}^{(d)})$ if $i = i^*$ (j^*) and 0 otherwise

Let $p = (\mathcal{F}(\sigma_+), \mathcal{F}(\sigma_-)) \in D_{\mathrm{Rips}}(X)$

with $\sigma_+ = \{v_0, \ldots, v_k\}$ and $\sigma_- = \{w_0, \ldots, w_{k+1}\}$

Example: Vietoris-Rips gradient



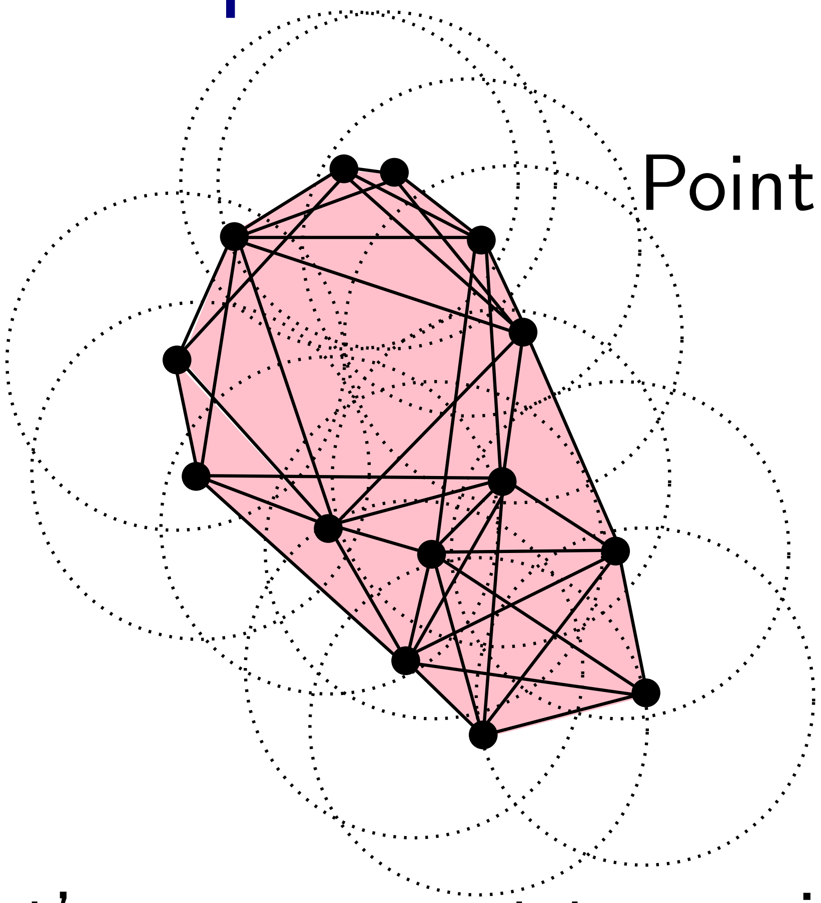


$$\nabla_X p = \left[\frac{\partial}{\partial X}\|v_{i^*} - v_{j^*}\|, \frac{\partial}{\partial X}\|w_{a^*} - w_{b^*}\|\right]$$

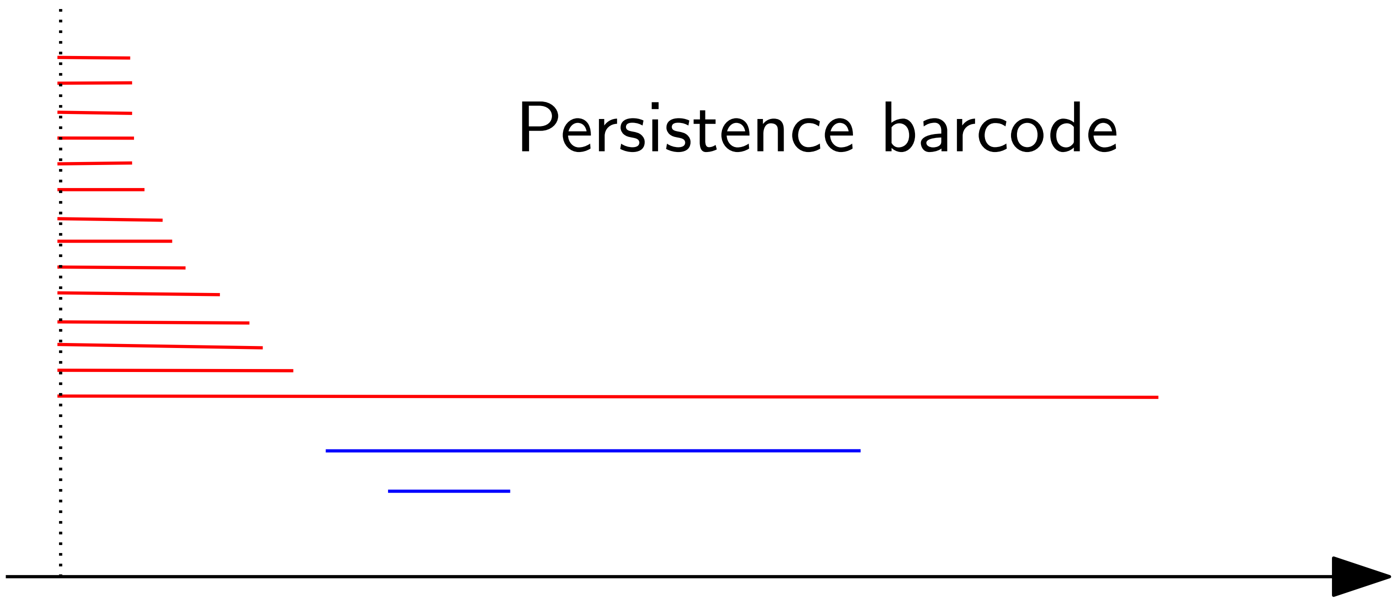
$$\frac{\partial}{\partial v_i^{(d)}}\|v_{i^*} - v_{j^*}\| = (-)\frac{1}{\|v_{i^*} - v_{j^*}\|}(v_{i^*}^{(d)} - v_{j^*}^{(d)}) \text{ if } i = i^* \ (j^*) \text{ and } 0 \text{ otherwise}$$

With this gradient rule, one can do gradient descent with any function of persistence!

Example: Vietoris-Rips gradient

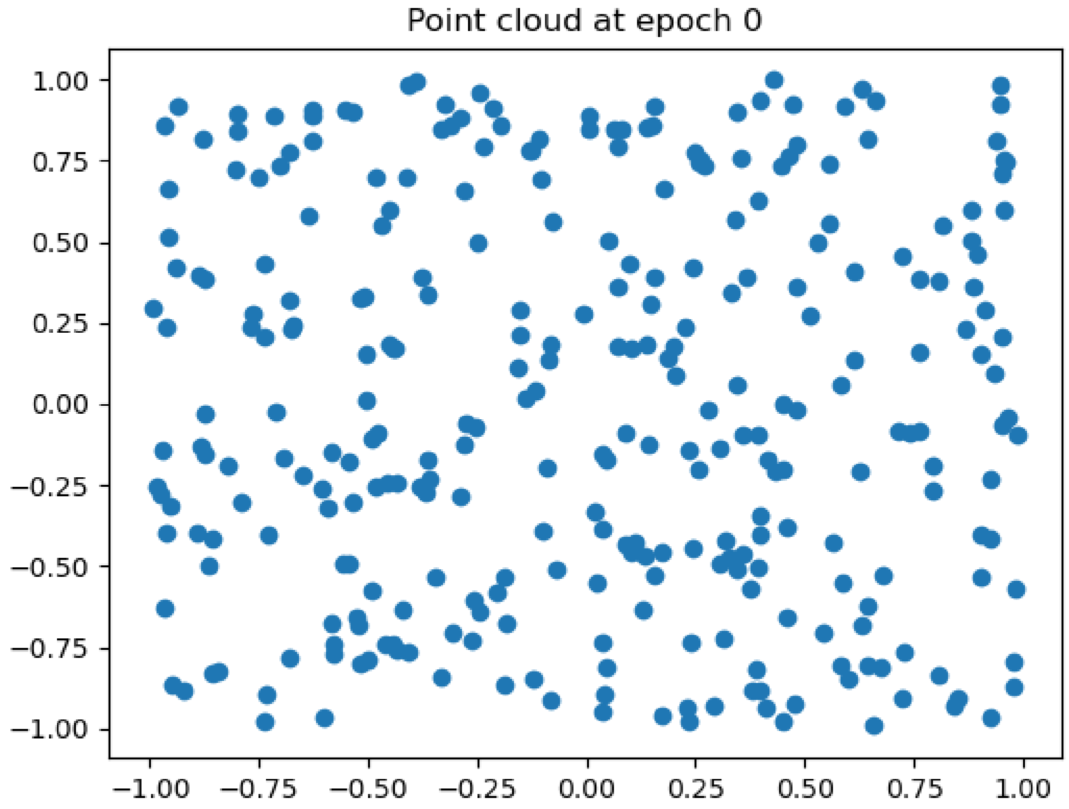


Point cloud $\hat{X}_n$

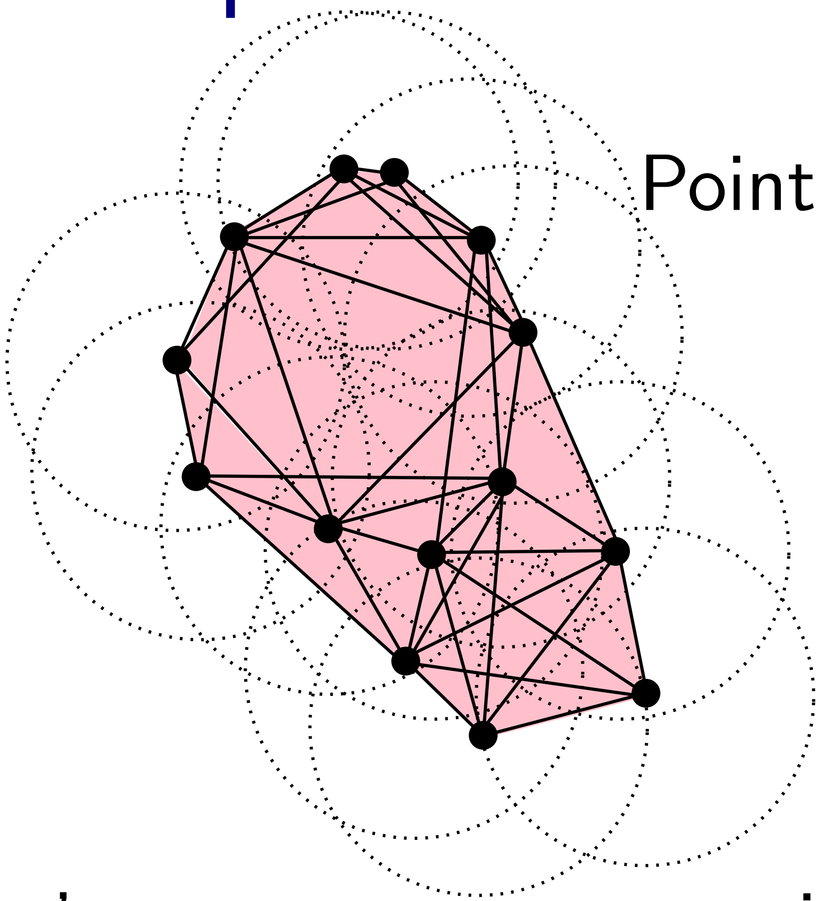


Persistence barcode

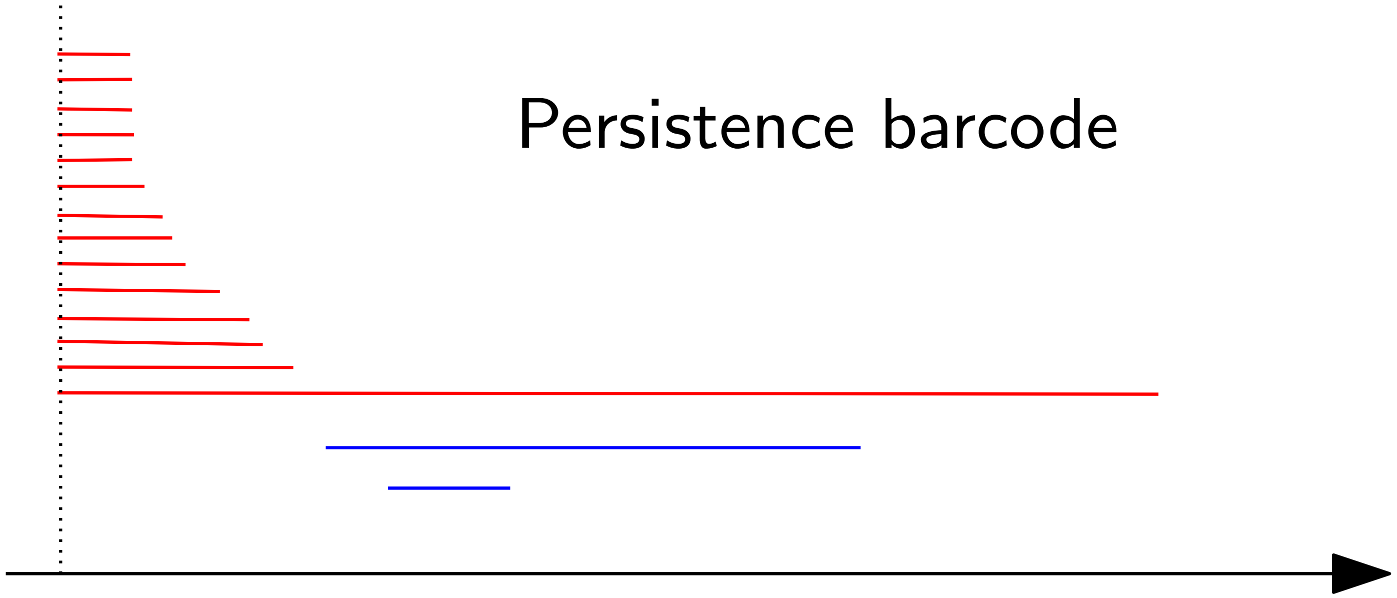
Let's say we want to maximize the number of holes in that point cloud.



Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$



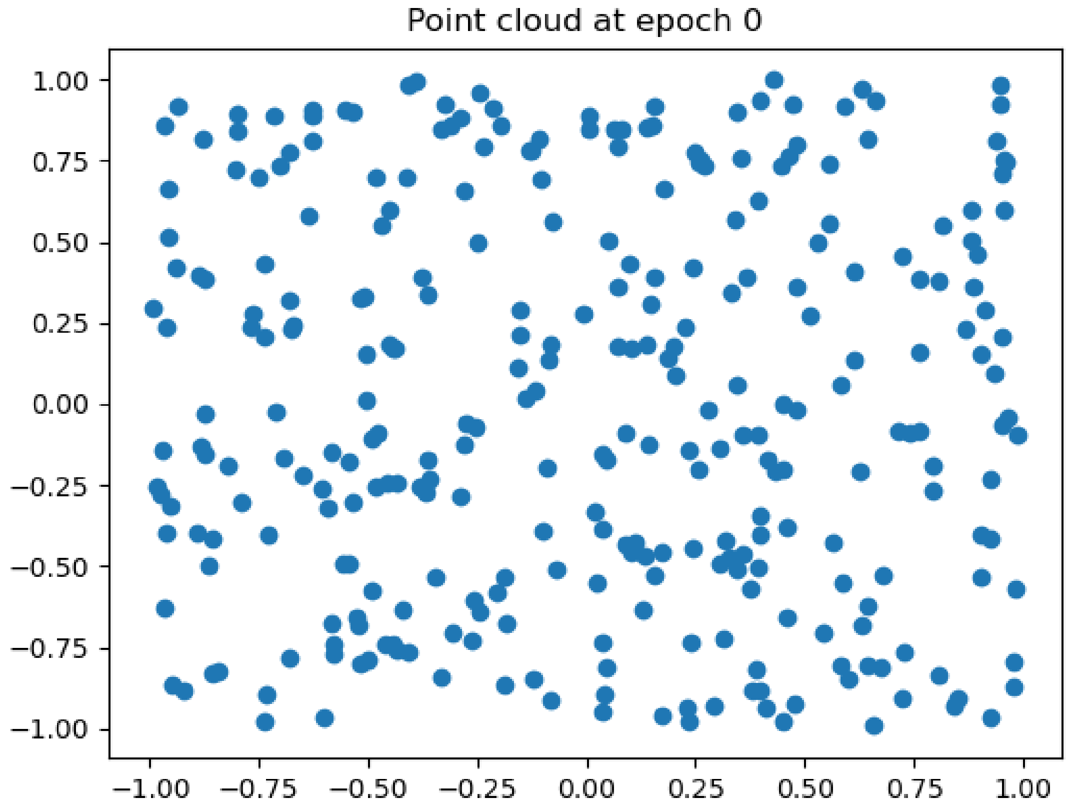
Persistence barcode

Let's say we want to maximize the number of holes in that point cloud.

We can use gradient descent to minimize loss

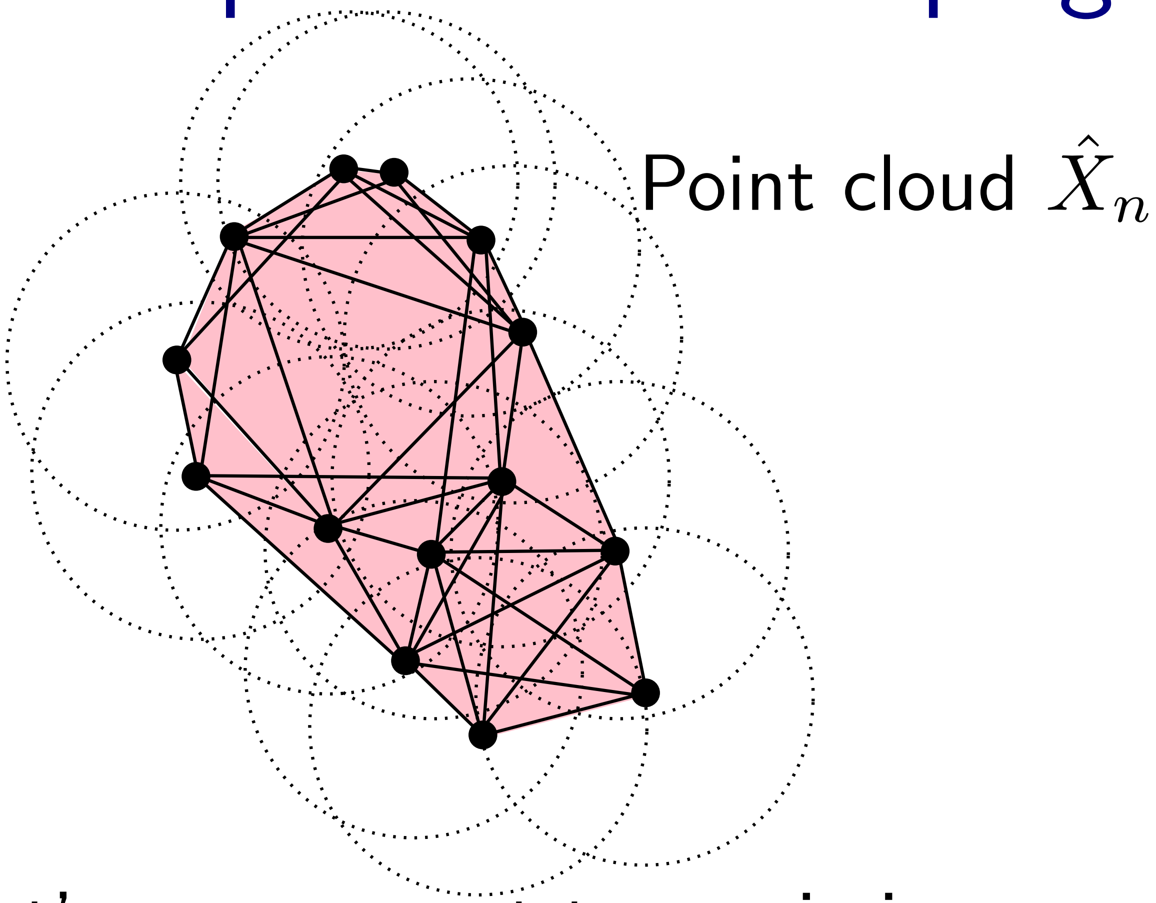
$$\mathcal{L}(X) = -\sum_p \|p\|_2^2,$$

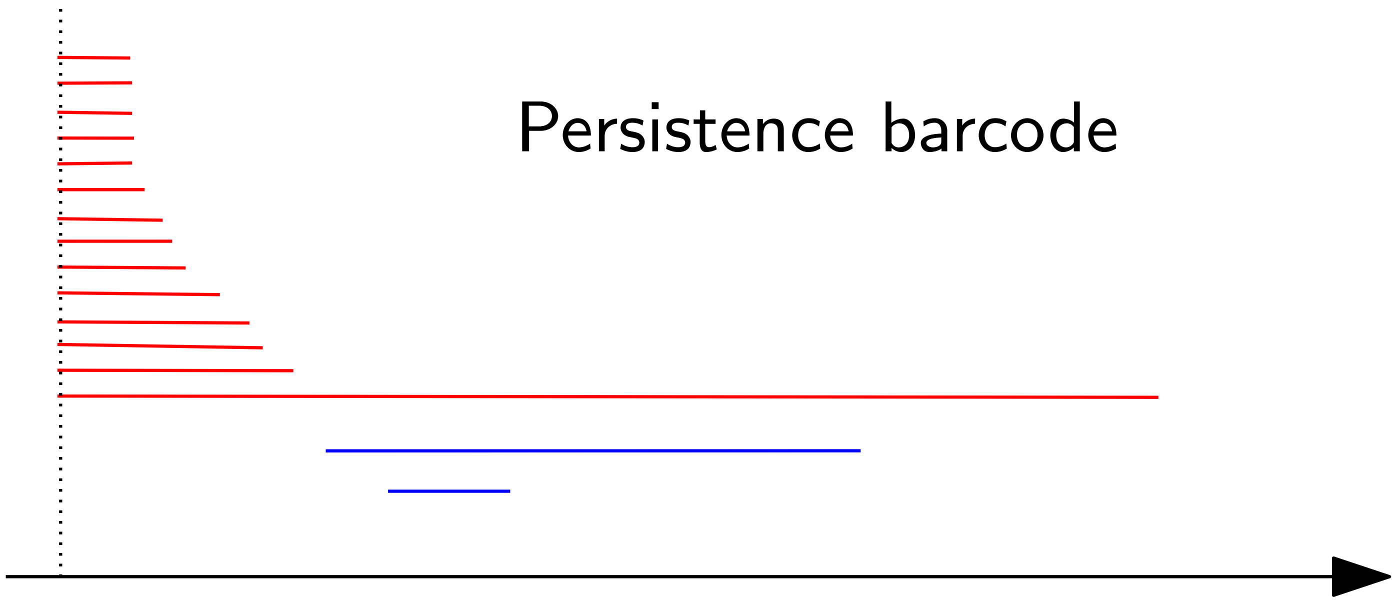
with $p \in D_{\mathrm{Rips}}(X)$ (in hom. 1).



Point cloud at epoch 0

Example: Vietoris-Rips gradient



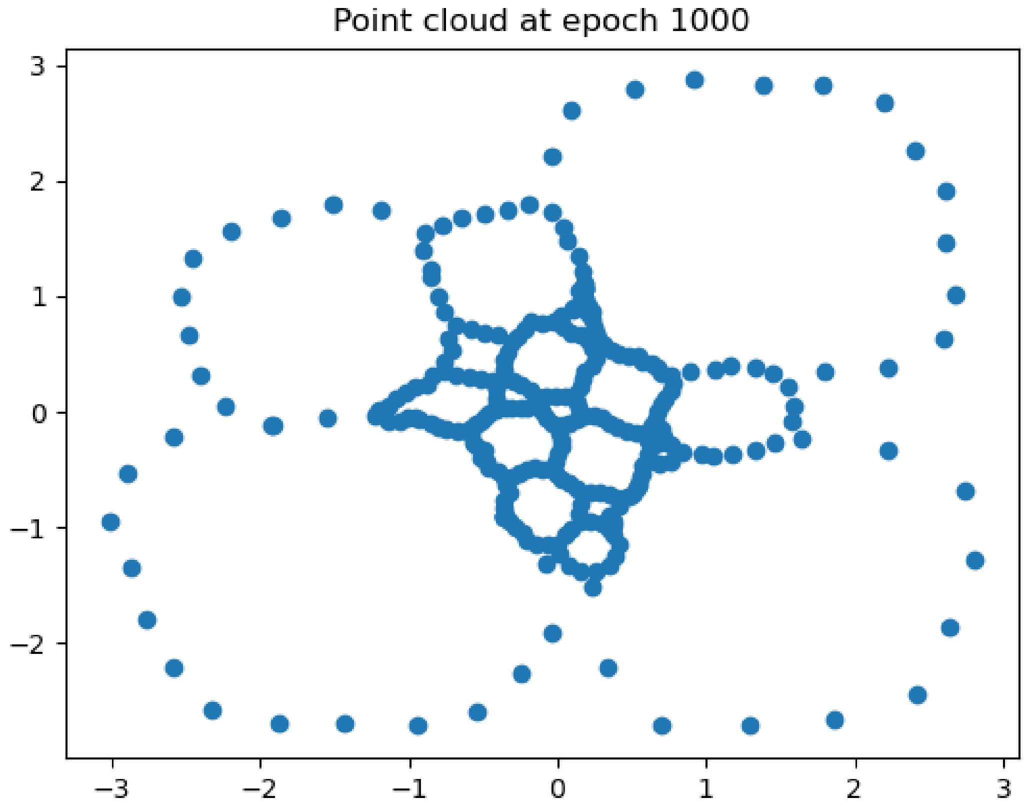


Let's say we want to maximize the number of holes in that point cloud.

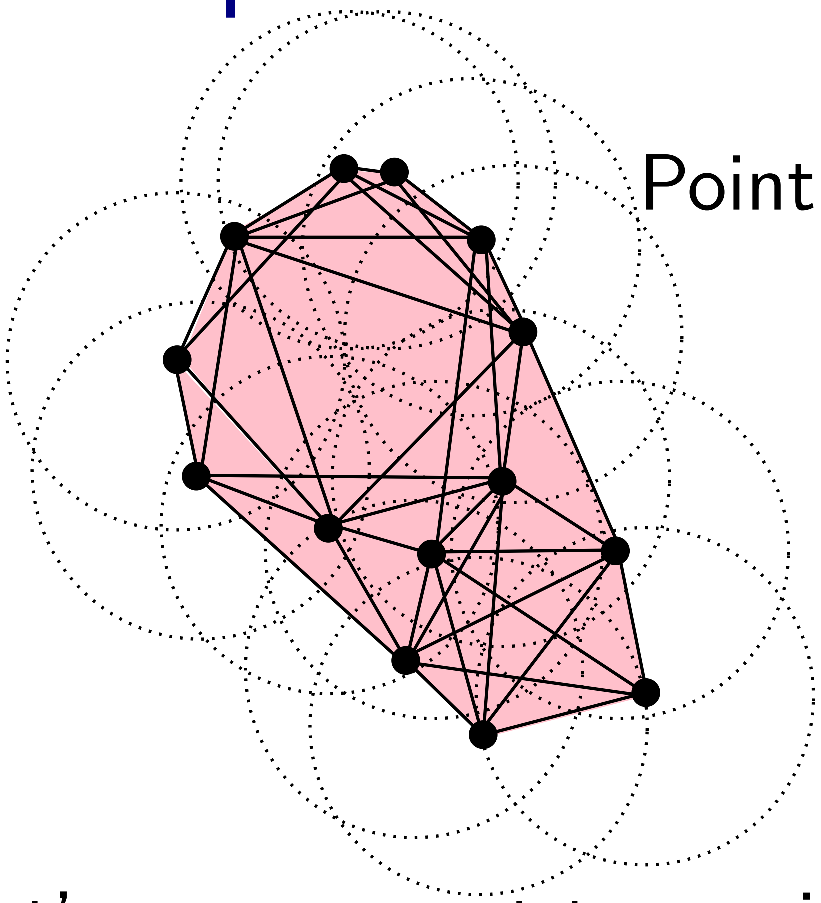
We can use gradient descent to minimize loss

$$\mathcal{L}(X) = -\sum_{p} \|p\|_2^2,$$

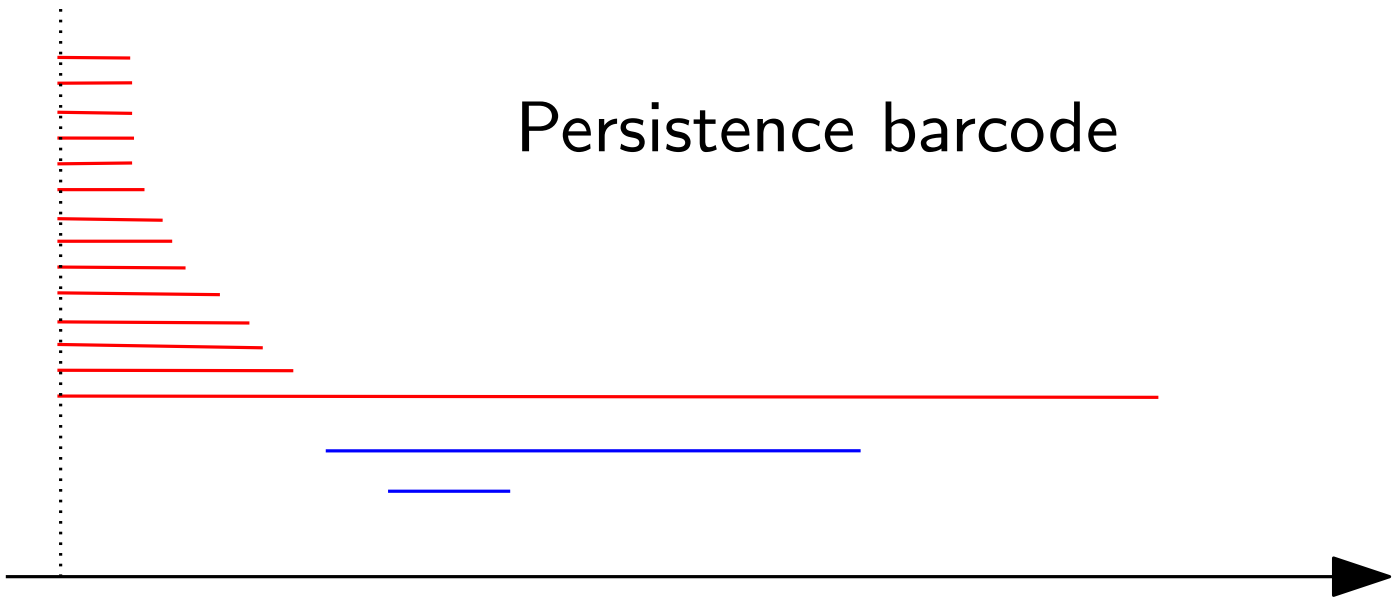
with $p \in D_{\text{Rips}}(X)$ (in hom. 1).



Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$



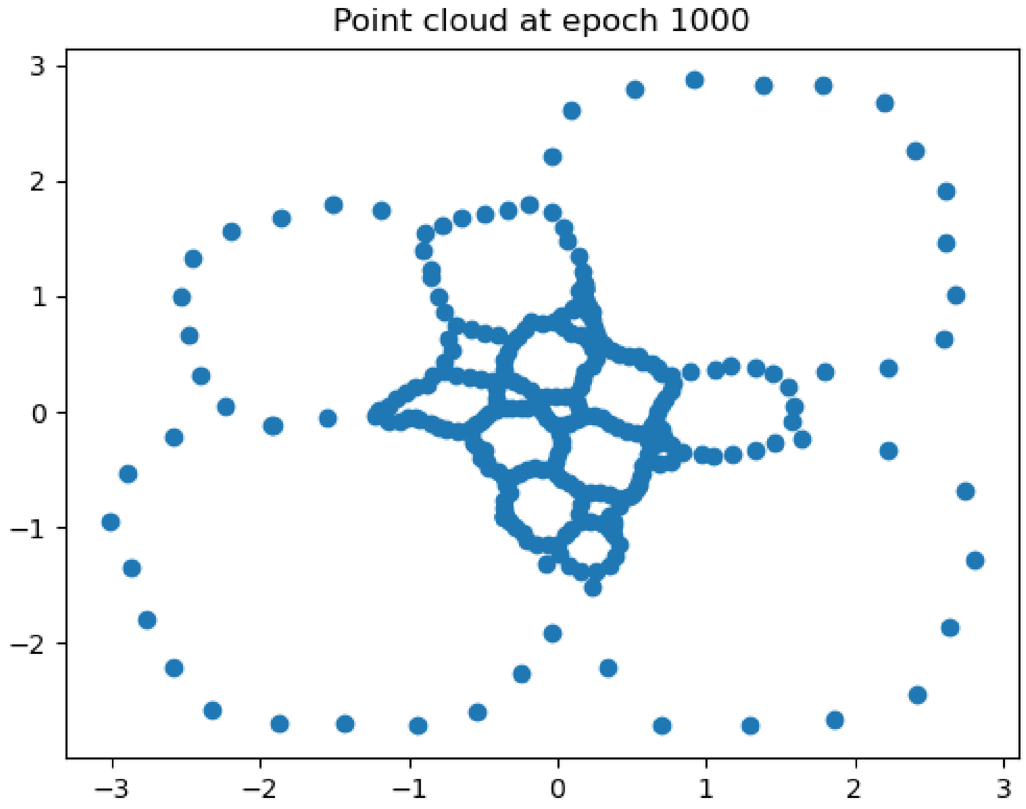
Persistence barcode

Let's say we want to maximize the number of holes in that point cloud.

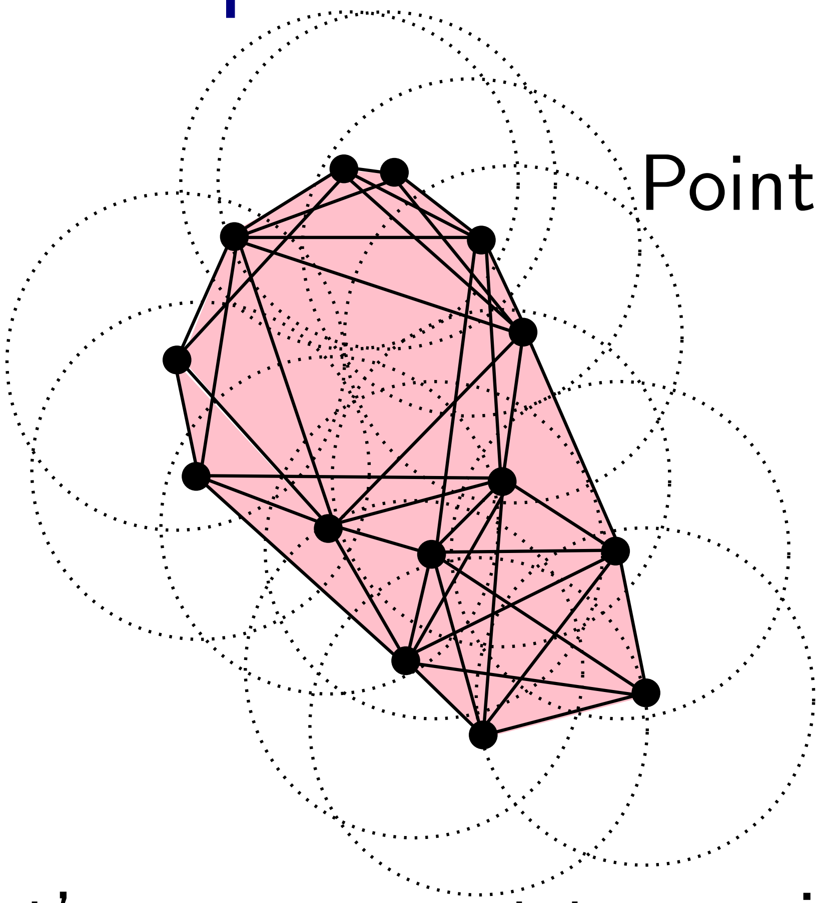
We can use gradient descent to minimize loss

$$\mathcal{L}(X) = -\sum_p \|p\|_2^2 + d(X, C),$$

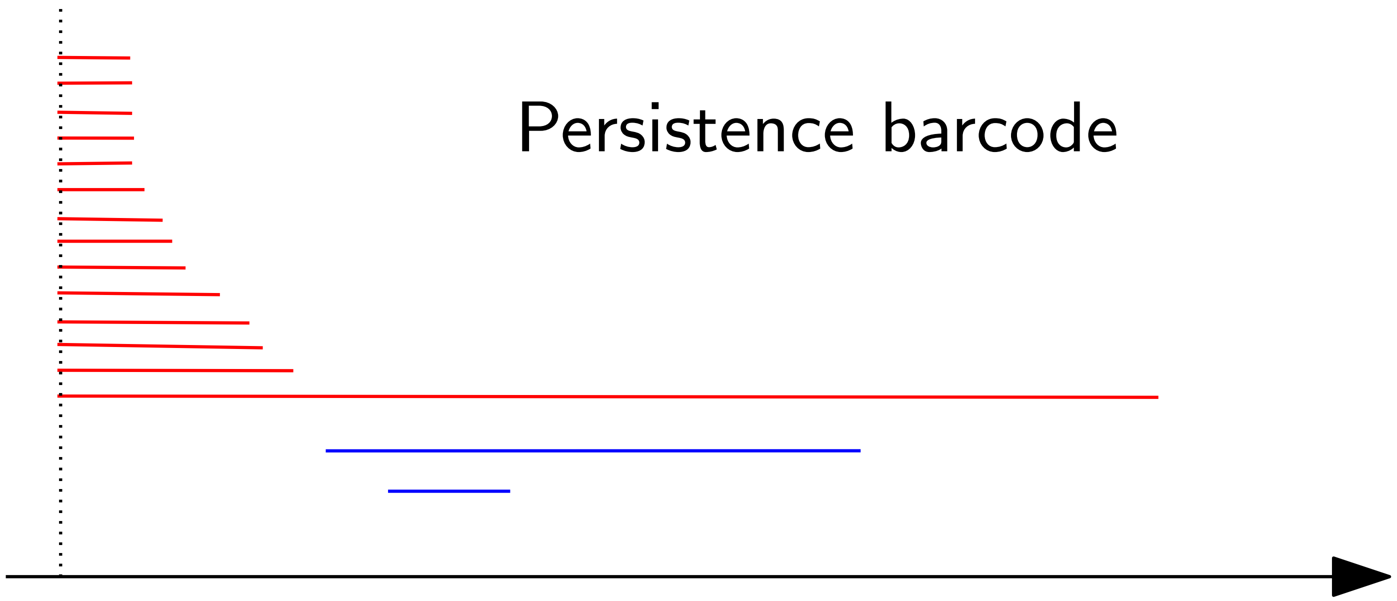
with $p \in D_{\mathrm{Rips}}(X)$ and C unit square.



Example: Vietoris-Rips gradient



Point cloud $\hat{X}_n$



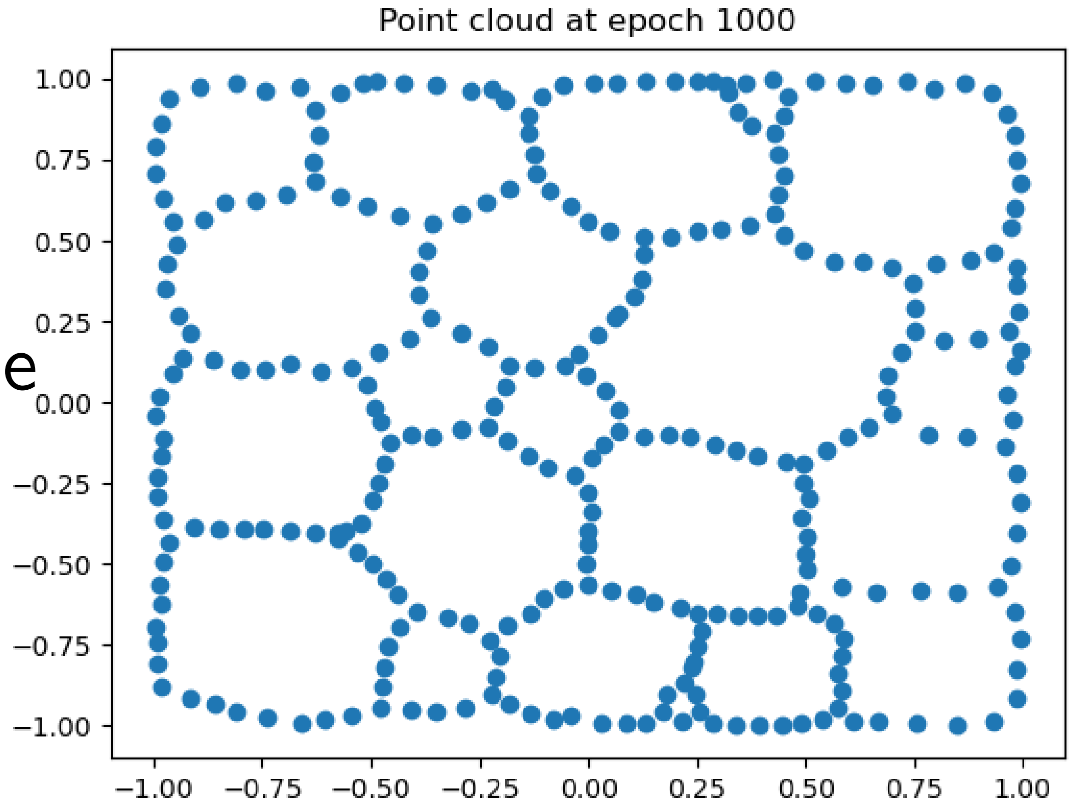
Persistence barcode

Let's say we want to maximize the number of holes in that point cloud.

We can use gradient descent to minimize loss

$$\mathcal{L}(X) = -\sum_p \|p\|_2^2 + d(X, C),$$

with $p \in D_{\text{Rips}}(X)$ and C unit square.



Example: Sublevel sets

Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_i \; f_\theta(v_i)$$

Example: Sublevel sets

Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \mathrm{max}_i \; f_\theta(v_i)$$

$$\nabla_\theta p = \left[\tfrac{\partial}{\partial \theta} f_\theta(v_{i^*}), \tfrac{\partial}{\partial \theta} f_\theta(w_{a^*}) \right]$$

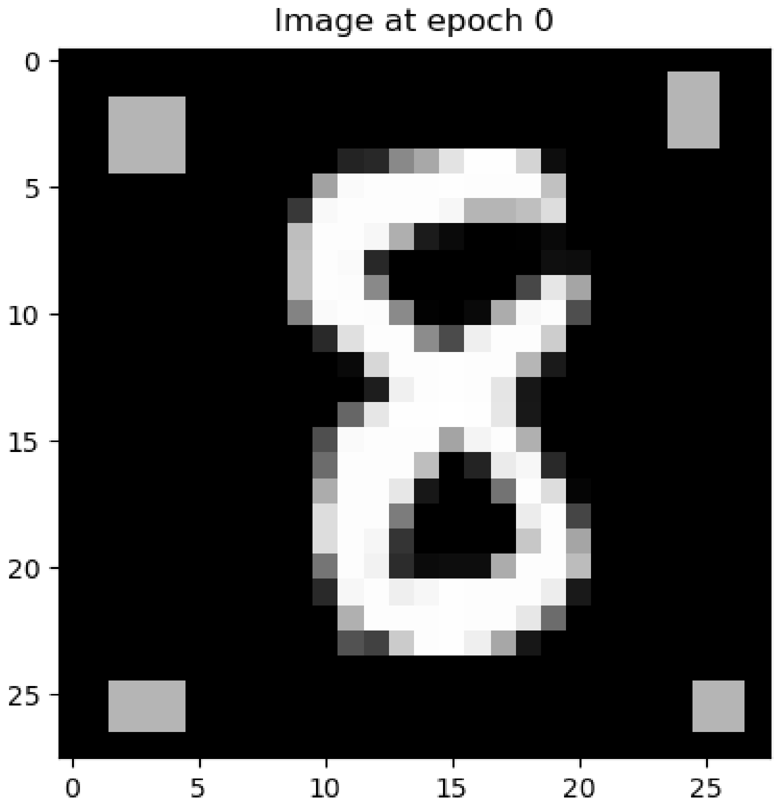
Example: Sublevel sets

Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_i \, f_\theta(v_i)$$

$$\nabla_\theta p = \left[\tfrac{\partial}{\partial\theta} f_\theta(v_{i^*}), \tfrac{\partial}{\partial\theta} f_\theta(w_{a^*})\right]$$

Let's say we want to remove the stains in that image.



Example: Sublevel sets

Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_i \ f_\theta(v_i)$$

$$\nabla_\theta p = \left[\tfrac{\partial}{\partial\theta} f_\theta(v_{i^*}), \tfrac{\partial}{\partial\theta} f_\theta(w_{a^*})\right]$$

Let's say we want to remove the stains in that image.

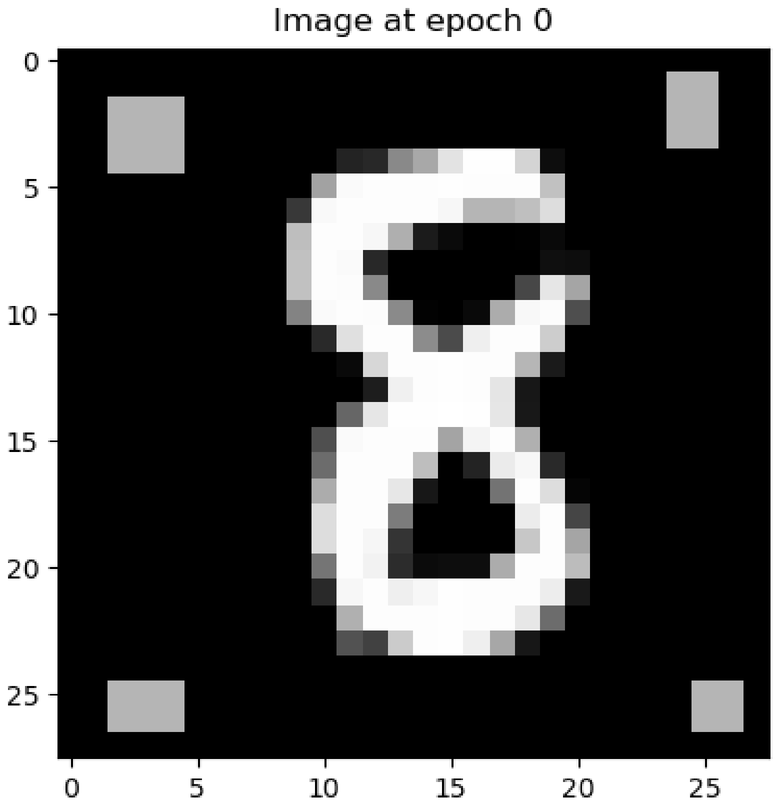


Image at epoch 0

We can use gradient descent to minimize loss

$$\mathcal{L}(X) = \sum_p \|p\|_2^2,$$

with $p \in D_{\text{Pixel}}(I)$ (in hom. 0).

Example: Sublevel sets

Given k-dim. simplex $\sigma = [v_0, \dots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_i \ f_\theta(v_i)$$

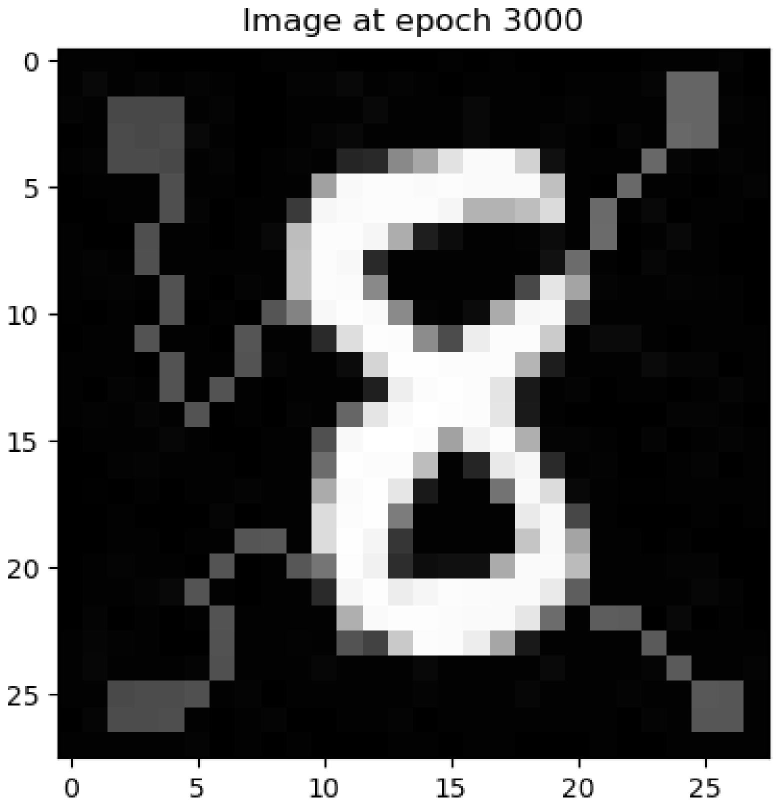
$$\nabla_\theta p = \left[\tfrac{\partial}{\partial \theta} f_\theta(v_{i^*}), \tfrac{\partial}{\partial \theta} f_\theta(w_{a^*}) \right]$$

Let's say we want to remove the stains in that image.

We can use gradient descent to minimize loss

$$\mathcal{L}(X) = \sum_p \|p\|_2^2,$$

with $p \in D_{\mathrm{Pixel}}(I)$ (in hom. 0).



Example: Sublevel sets

Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_i \ f_\theta(v_i)$$

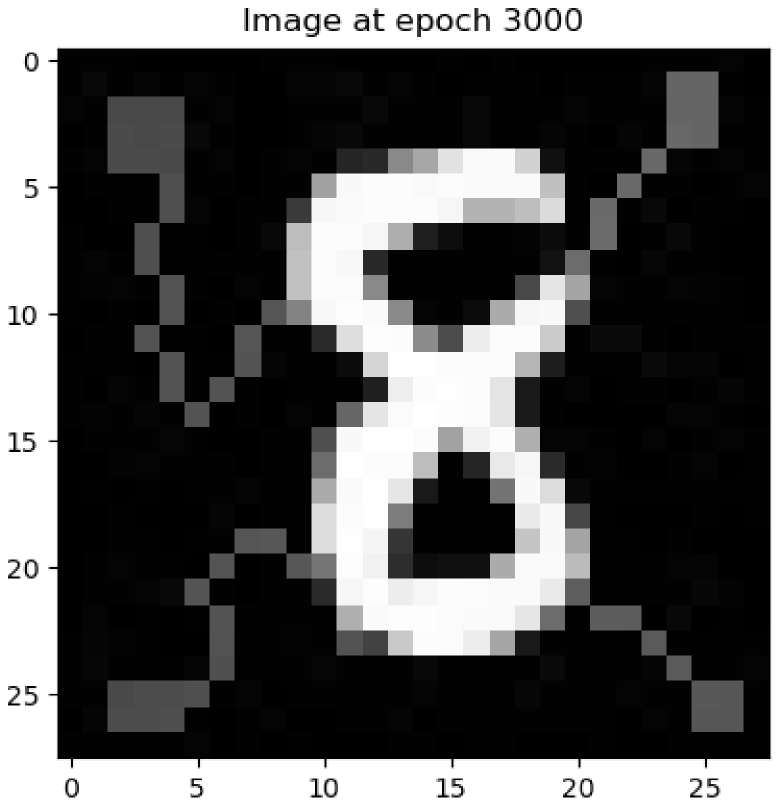
$$\nabla_\theta p = \left[\tfrac{\partial}{\partial\theta} f_\theta(v_{i^*}), \tfrac{\partial}{\partial\theta} f_\theta(w_{a^*})\right]$$

Let's say we want to remove the stains in that image.

We can use gradient descent to minimize loss

$$\mathcal{L}(X) = \sum_p \|p\|_2^2 + \sum_{P \in I} \max\{|P|, |1-P|\},$$

with $p \in D_{\text{Pixel}}(I)$.



Example: Sublevel sets

Given k-dim. simplex $\sigma = [v_0, \ldots, v_k]$, one has

$$\mathcal{F}(\sigma) = \max_i \ f_\theta(v_i)$$

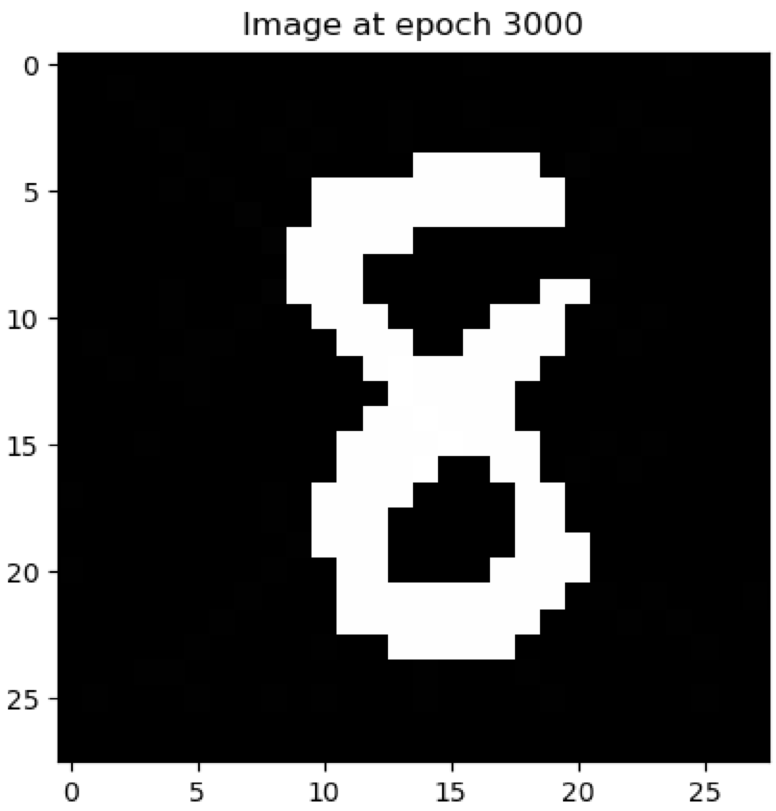
$$\nabla_\theta p = \left[\tfrac{\partial}{\partial\theta} f_\theta(v_{i^*}), \tfrac{\partial}{\partial\theta} f_\theta(w_{a^*})\right]$$

Let's say we want to remove the stains in that image.

We can use gradient descent to minimize loss

$$\mathcal{L}(X) = \sum_p \|p\|_2^2 + \sum_{P \in I} \max\{|P|, |1-P|\},$$

with $p \in D_{\mathrm{Pixel}}(I)$.



Topological gradient descent

[*Optimizing persistent homology based functions*, C., Chazal, Glisse, Ike, Kanna, Umeda, ICML, 2021]

Topological gradient descent

[*Optimizing persistent homology based functions*, C., Chazal, Glisse, Ike, Kanna, Umeda, ICML, 2021]

For a fixed ordering of the simplices in a simplicial complex K, the corresponding persistence diagram always has the same number of points: its gradient is well-defined!

Topological gradient descent

[*Optimizing persistent homology based functions*, C., Chazal, Glisse, Ike, Kanna, Umeda, ICML, 2021]

For a fixed ordering of the simplices in a simplicial complex K, the corresponding persistence diagram always has the same number of points: its gradient is well-defined!

If the ordering changes, the boundary matrix can have a new reduced form and the persistence diagram can have a new, different number of points.

Topological gradient descent

[*Optimizing persistent homology based functions*, C., Chazal, Glisse, Ike, Kanna, Umeda, ICML, 2021]

For a fixed ordering of the simplices in a simplicial complex K, the corresponding persistence diagram always has the same number of points: its gradient is well-defined!

If the ordering changes, the boundary matrix can have a new reduced form and the persistence diagram can have a new, different number of points.

Prop: Let K be a simplicial complex and let $\Phi : A \to \mathbb{R}^{|K|}$ a (parameterized) filtration of K. There exists a partition $A = S \sqcup O_1 \sqcup \cdots \sqcup O_k$ s.t. all the restrictions $\Phi : O_i \to \mathbb{R}^{|K|}$ are differentiable.

The O_i's are the parts of A where the ordering of the simplices of K is preserved, and S is the boundaries of all O_i's.

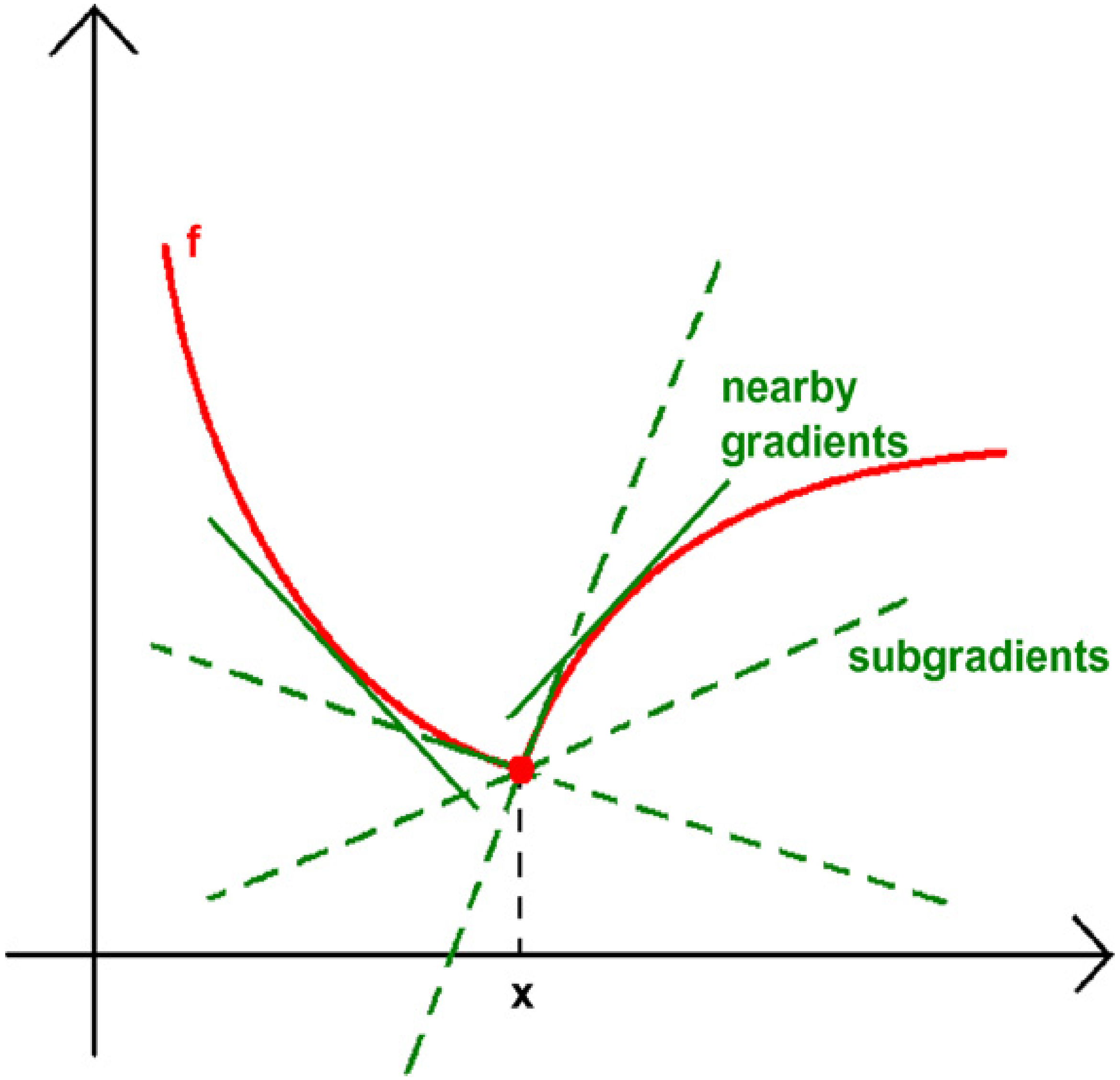
Topological gradient descent

[*Optimizing persistent homology based functions*, C., Chazal, Glisse, Ike, Kanna, Umeda, ICML, 2021]

Def: The *Clarke subdifferential* $\partial\mathcal{L}$ of $\mathcal{L}$ is the set:

$$\partial_x\mathcal{L} = \mathrm{conv}\{\lim_{x_i\to x}\nabla\mathcal{L}(x_i) : \mathcal{L} \text{ is diff. at } x_i\},$$

where conv denotes the convex hull.



Topological gradient descent

[*Optimizing persistent homology based functions*, C., Chazal, Glisse, Ike, Kanna, Umeda, ICML, 2021]

Let $\{\alpha_k\}_k$, $\{\zeta_k\}_k$ s.t.

$\alpha_k \geq 0$, $\sum_k \alpha_k = +\infty$ and $\sum_k \alpha_k^2 < +\infty$

ζ_k random variables s.t. $E[\zeta_k] = 0$ and $E[\|\zeta_k\|^2] < C$ for some $C > 0$

Thm: As long as $\mathcal{L} \circ \mathrm{Pers} \circ \Phi$ is locally Lipschitz, the sequence

$$a_{k+1} = a_k - \alpha_k(g_k + \zeta_k),$$

where $g_k \in \partial_{a_k}(\mathcal{L} \circ \mathrm{Pers} \circ \Phi)$, converges to a critical point of $\mathcal{L} \circ \mathrm{Pers} \circ \Phi$.

Topological stratified gradient descent

[*A gradient sampling algorithm for stratified maps with applications to topological data analysis*, Leygonie, C., Lacombe, Oudot, 2021]

Better guarantees can be obtained by smoothing the gradient definition.

Def: The *smoothed topological gradient* of $\mathrm{Pers} \circ \Phi$ is defined as:

$$\tilde{\nabla}_a = \mathrm{argmin}\{\|g\| \,:\, g \in \mathrm{conv}(S_a)\}$$

where $S_a = \{\nabla_{a'} \,:\, a' \in O_i, O_i \in \mathcal{N}(O_a)\}$, where O_a is the stratum associated to a, and $\mathcal{N}(O_a)$ is the set of strata that are close to O_a.

Intuitively, close strata means that their corresponding orderings are very similar, e.g., they differ by single swaps, or their distance is bounded by $\epsilon > 0$.

Topological stratified gradient descent

[*A gradient sampling algorithm for stratified maps with applications to topological data analysis*, Leygonie, C., Lacombe, Oudot, 2021]

Better guarantees can be obtained by smoothing the gradient definition.

Def: The *smoothed topological gradient* of $\mathrm{Pers} \circ \Phi$ is defined as:

$$\tilde{\nabla}_a = \mathrm{argmin}\{\|g\| \,:\, g \in \mathrm{conv}(S_a)\}$$

where $S_a = \{\nabla_{a'} \,:\, a' \in O_i, O_i \in \mathcal{N}(O_a)\}$, where O_a is the stratum associated to a, and $\mathcal{N}(O_a)$ is the set of strata that are close to O_a.

Intuitively, close strata means that their corresponding orderings are very similar, e.g., they differ by single swaps, or their distance is bounded by $\epsilon > 0$.

Thm: Let $\epsilon > 0$. As long as $\mathcal{L} \circ \mathrm{Pers} \circ \Phi$ is Lipschitz, the sequence

$$a_{k+1} = a_k - \epsilon \cdot \tilde{\nabla}_{a_k} / \|\tilde{\nabla}_{a_k}\|,$$

converges in **finitely many** iterations to $\tilde{a}$ s.t. $\exists \bar{a} \,:\, \tilde{\nabla}_{\bar{a}} = 0$ and $\|\tilde{a} - \bar{a}\| \leq \epsilon$.

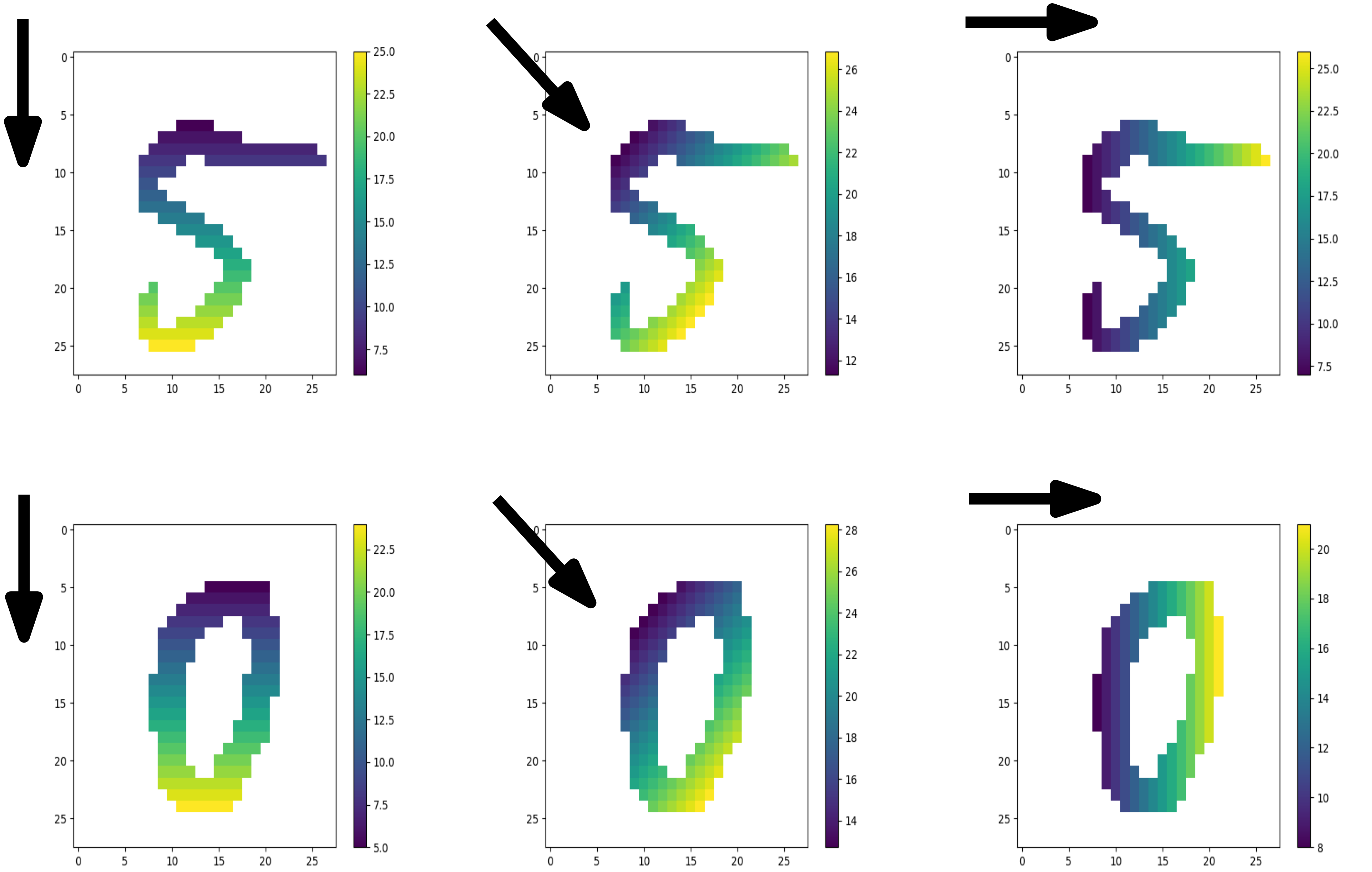
Example: filter selection

Assume we have a supervised classification task. The goal is to find a filtration from a family $\mathcal{F}$ such that the corresponding persistence diagrams give the best classification score.

Example: filter selection

Assume we have a supervised classification task. The goal is to find a filtration from a family $\mathcal{F}$ such that the corresponding persistence diagrams give the best classification score.

Ex: images filtered by a direction parameterized by angle.



Example: filter selection

Assume we have a supervised classification task. The goal is to find a filtration from a family $\mathcal{F}$ such that the corresponding persistence diagrams give the best classification score.

Idea: minimize:

$$\mathcal{L}(f) = \sum_{l} \frac{\sum_{y_i = y_j = l} \mathrm{d}_p(D_f(x_i), D_f(x_j))}{\sum_{y_i = l} \mathrm{d}_p(D_f(x_i), D_f(x_j))},$$

one can also use Sliced Wasserstein for speedup.

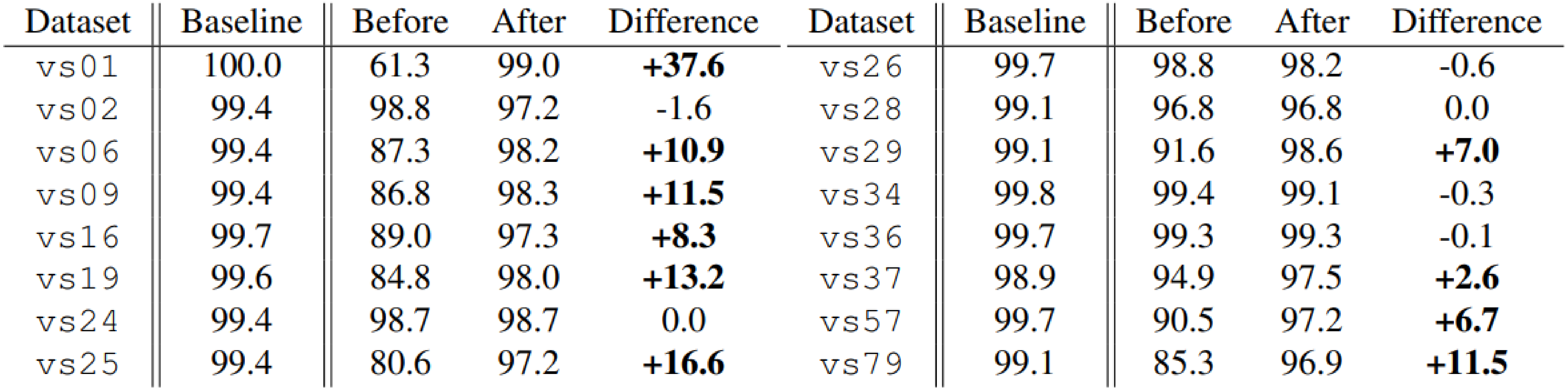
Example: filter selection

Assume we have a supervised classification task. The goal is to find a filtration from a family $\mathcal{F}$ such that the corresponding persistence diagrams give the best classification score.

Idea: minimize:

$$\mathcal{L}(f) = \sum_{l} \frac{\sum_{y_i=y_j=l} \mathrm{d}_p(D_f(x_i), D_f(x_j))}{\sum_{y_i=l} \mathrm{d}_p(D_f(x_i), D_f(x_j))},$$

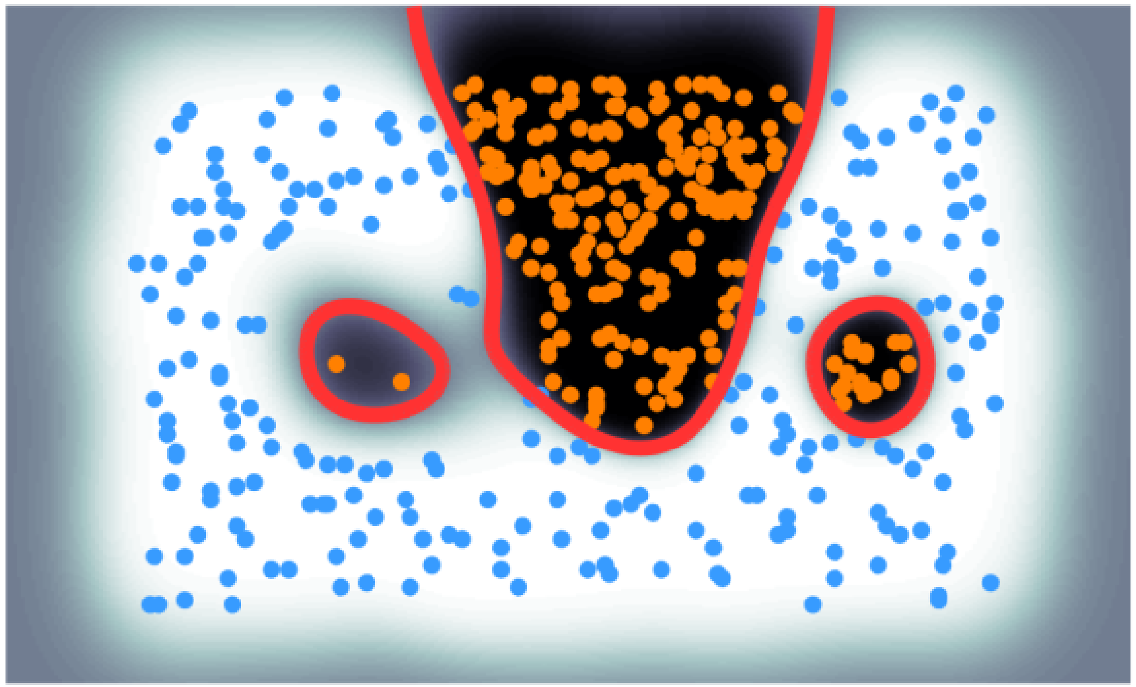
one can also use Sliced Wasserstein for speedup.

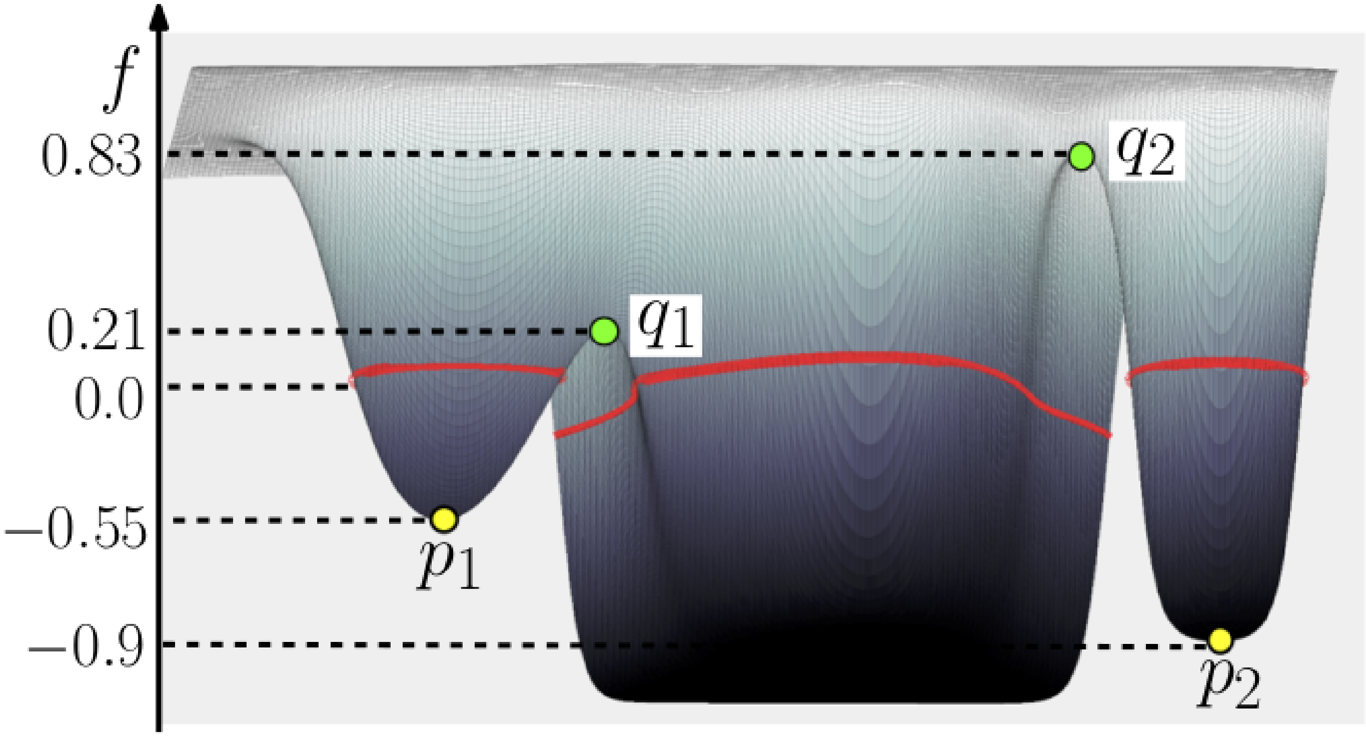


Dataset	Baseline	Before	After	Difference
vs01	100.0	61.3	99.0	**+37.6**
vs02	99.4	98.8	97.2	-1.6
vs06	99.4	87.3	98.2	**+10.9**
vs09	99.4	86.8	98.3	**+11.5**
vs16	99.7	89.0	97.3	**+8.3**
vs19	99.6	84.8	98.0	**+13.2**
vs24	99.4	98.7	98.7	0.0
vs25	99.4	80.6	97.2	**+16.6**
vs26	99.7	98.8	98.2	-0.6
vs28	99.1	96.8	96.8	0.0
vs29	99.1	91.6	98.6	**+7.0**
vs34	99.8	99.4	99.1	-0.3
vs36	99.7	99.3	99.3	-0.1
vs37	98.9	94.9	97.5	**+2.6**
vs57	99.7	90.5	97.2	**+6.7**
vs79	99.1	85.3	96.9	**+11.5**

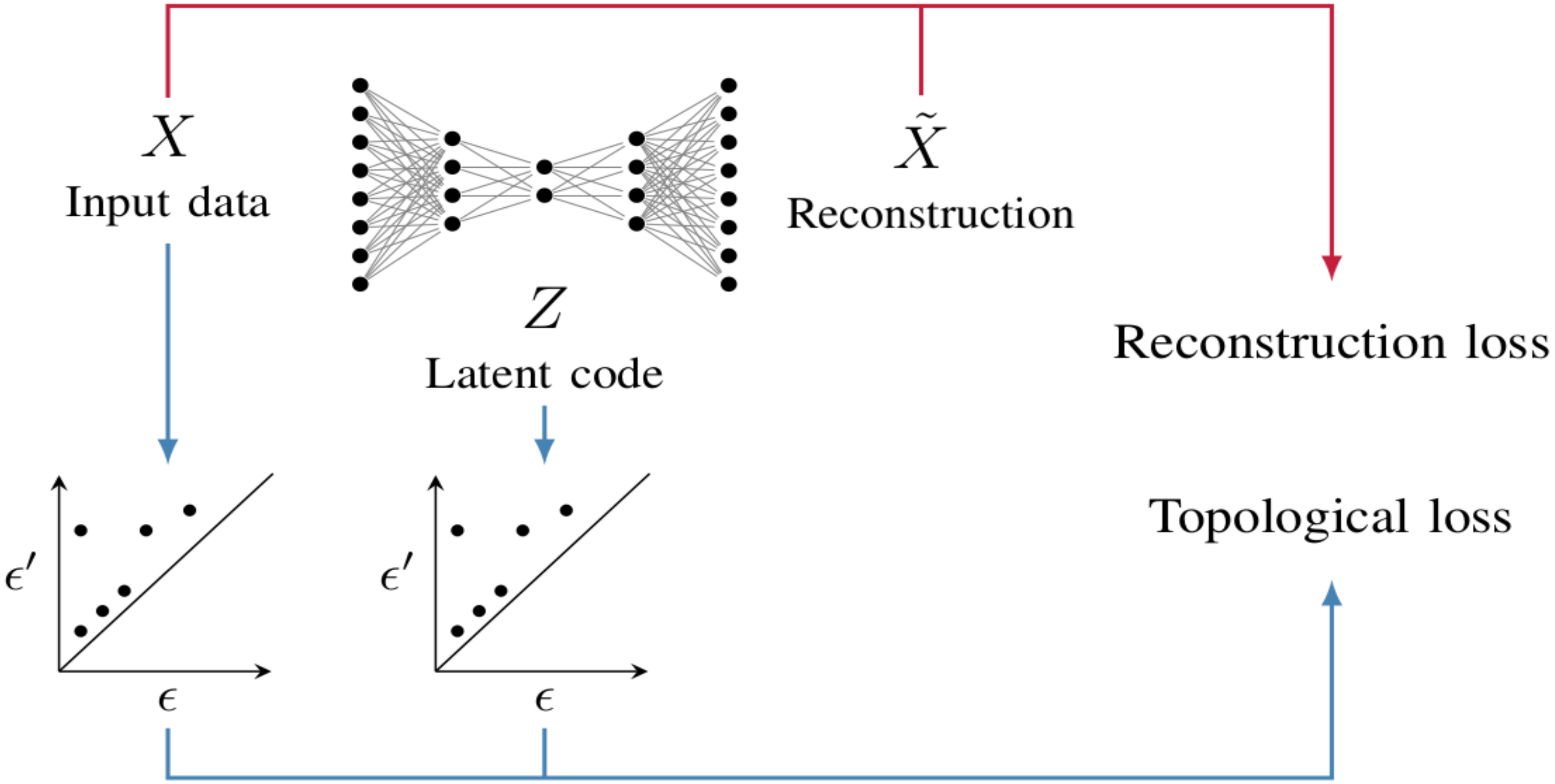
Application: model regularization

[*A Topological Regularizer for Classifiers via Persistent Homology*, Chen, Ni, Bai, Wang, AISTATS, 2019]





[*Topological autoencoders*, Moor, Horn, Rieck, Borgwardt, ICML, 2020]



X
Input data

Z
Latent code

$\tilde{X}$
Reconstruction

Reconstruction loss

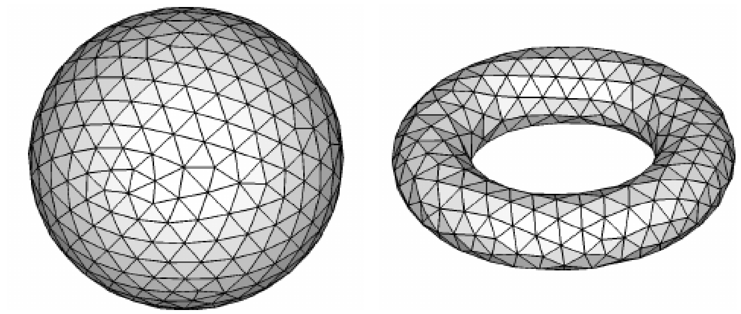
Topological loss

Take home message

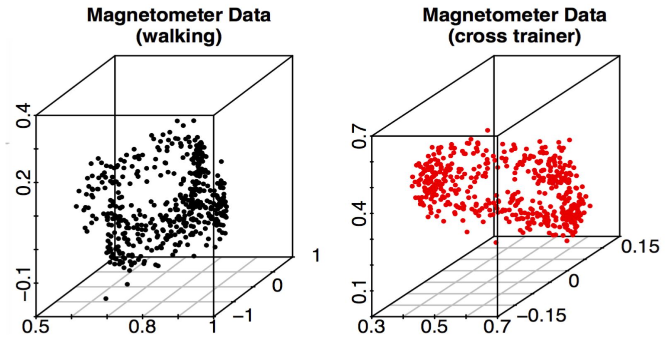
Topological Data Analysis is:

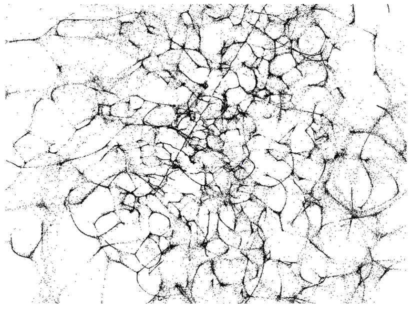
a mathematically grounded framework...

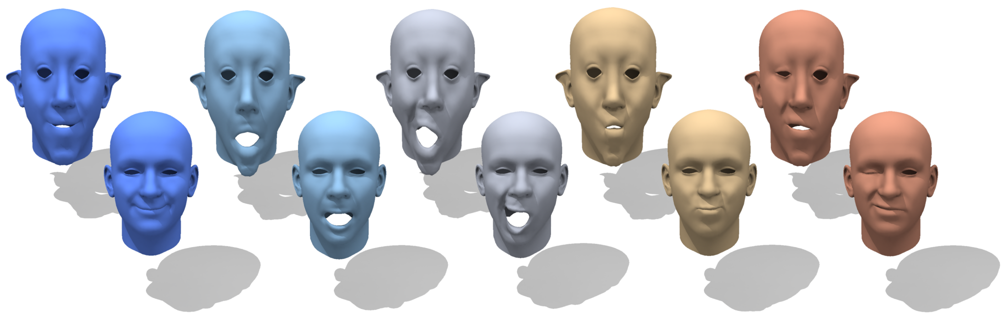
$$H_k = Z_k / B_k$$



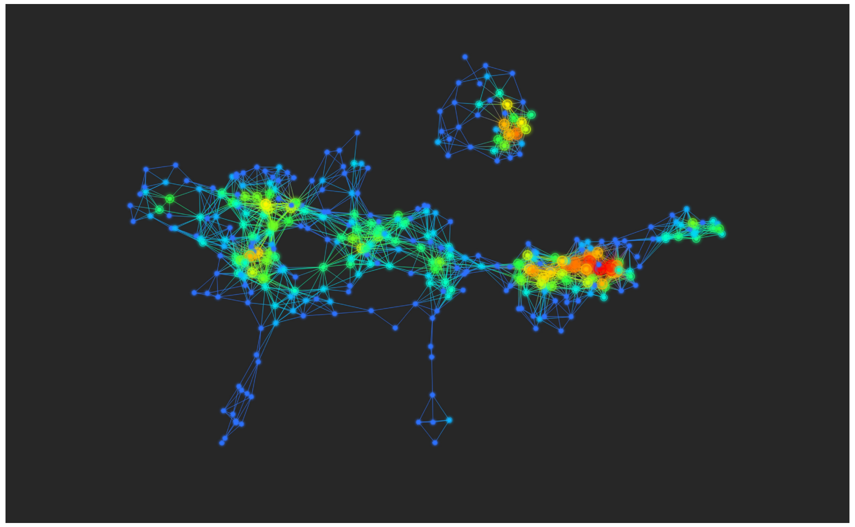
...that applies to a wide variety of data sets...



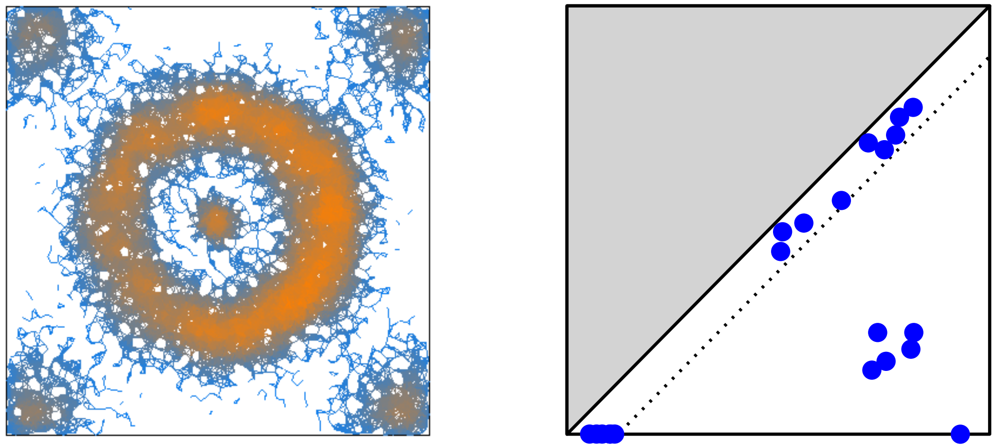




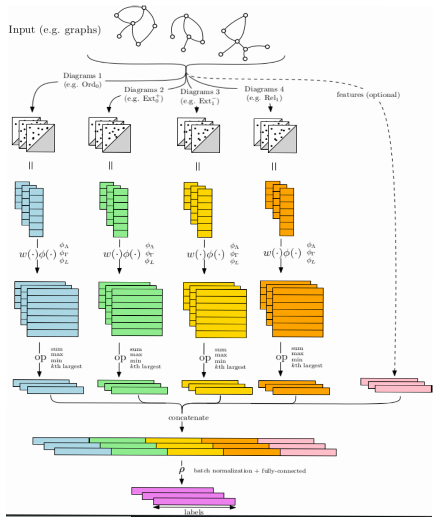
...for a wide variety of tasks.



Exploratory data analysis



Topological inference



Topological machine learning