Foundations of Geometric Methods in Data Analysis
2017-18

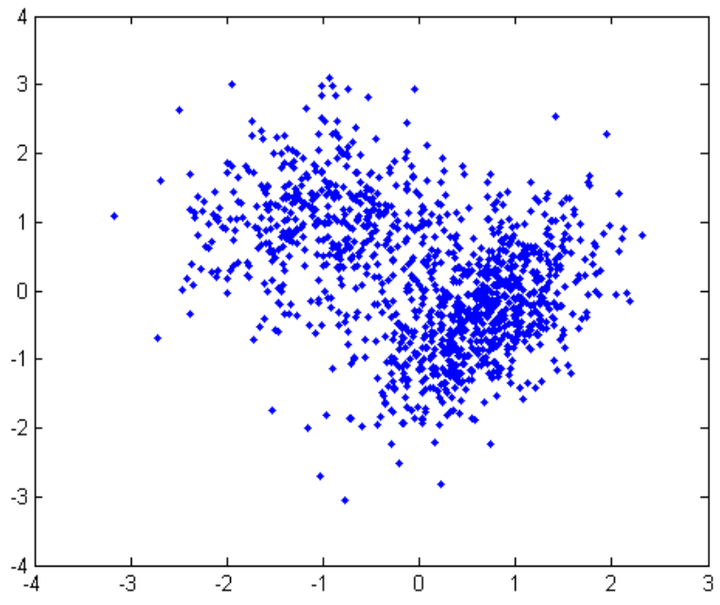
Clustering algorithms and introduction to persistent homology

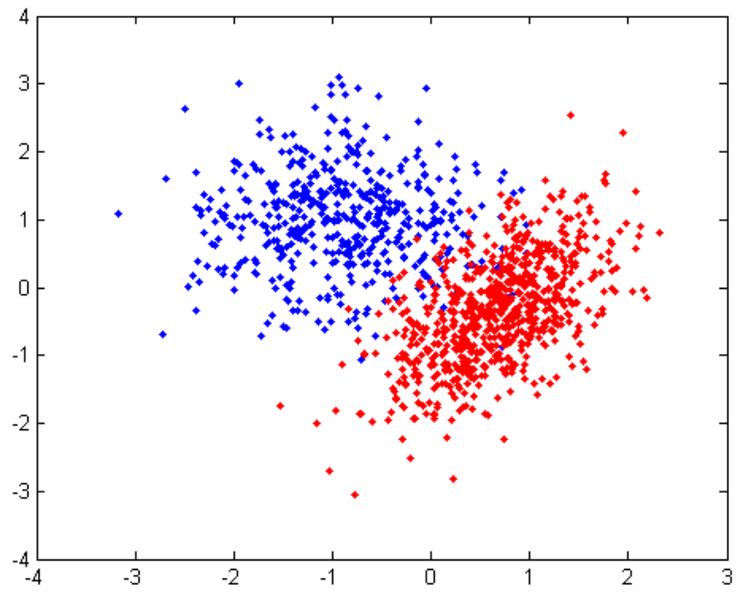
Frédéric Chazal
INRIA Saclay - Ile-de-France
frederic.chazal@inria.fr

Introduction

Clustering: A partition of data into groups of similar observations. The observations in each group (cluster) are similar to each other and dissimilar to observations from other groups.

Input: a finite set of observations: point cloud embedded in an Euclidean space (with coordinates) or a more general metric space (pairwise distance/similarity) matrix.





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

Introduction

Clustering: A partition of data into groups of similar observations. The observations in each group (cluster) are similar to each other and dissimilar to observations from other groups.

Not a single/universal notion of cluster.

A wealth of approaches:

- Variational
- Spectral
- Density based
- Hierarchical
- etc...

Introduction

Clustering: A partition of data into groups of similar observations. The observations in each group (cluster) are similar to each other and dissimilar to observations from other groups.

Not a single/universal notion of cluster.

A wealth of approaches:

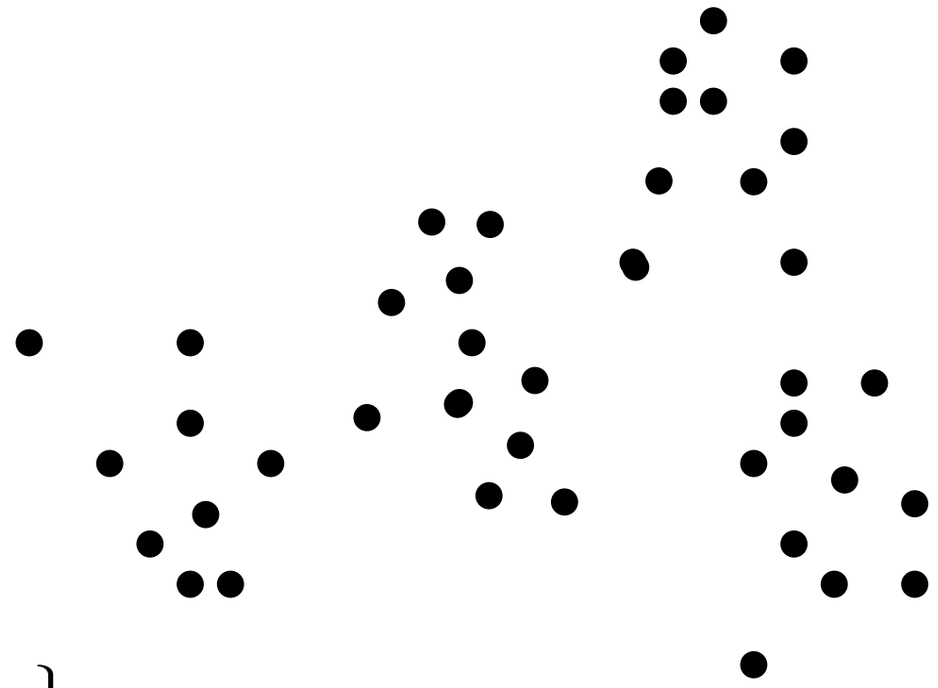
- Variational
- Spectral
- Density based
- Hierarchical
- etc...

In this lecture:

- a few basic classical "algorithms" motivating and bringing us to the introduction of persistent homology.

The k-means algorithm

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

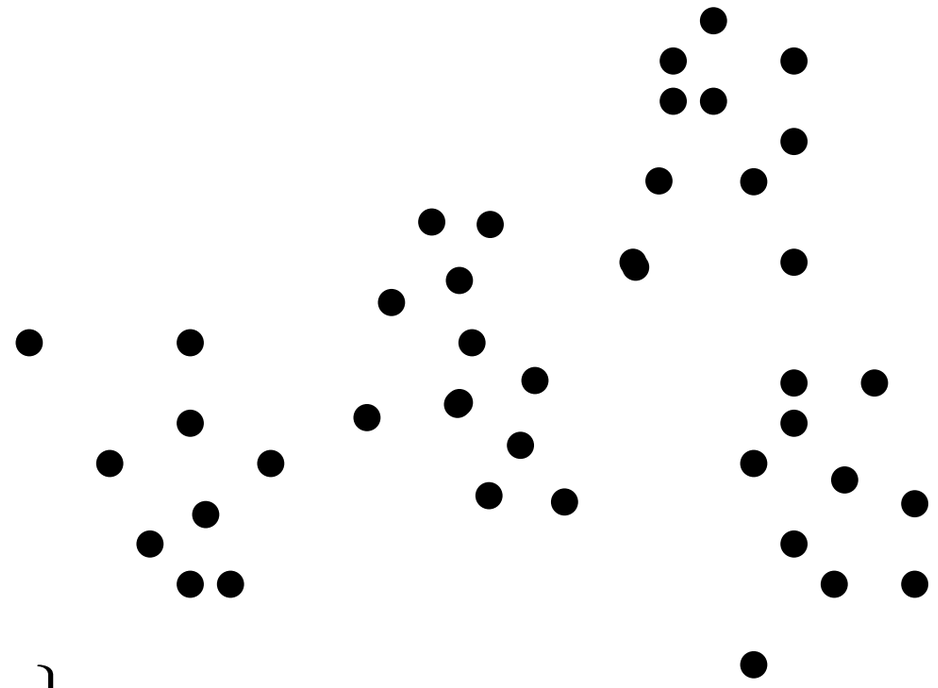


Goal: Find a set of k points $L = \{y_1, \cdots y_k\}$ that minimizes

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

The k-means algorithm

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



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

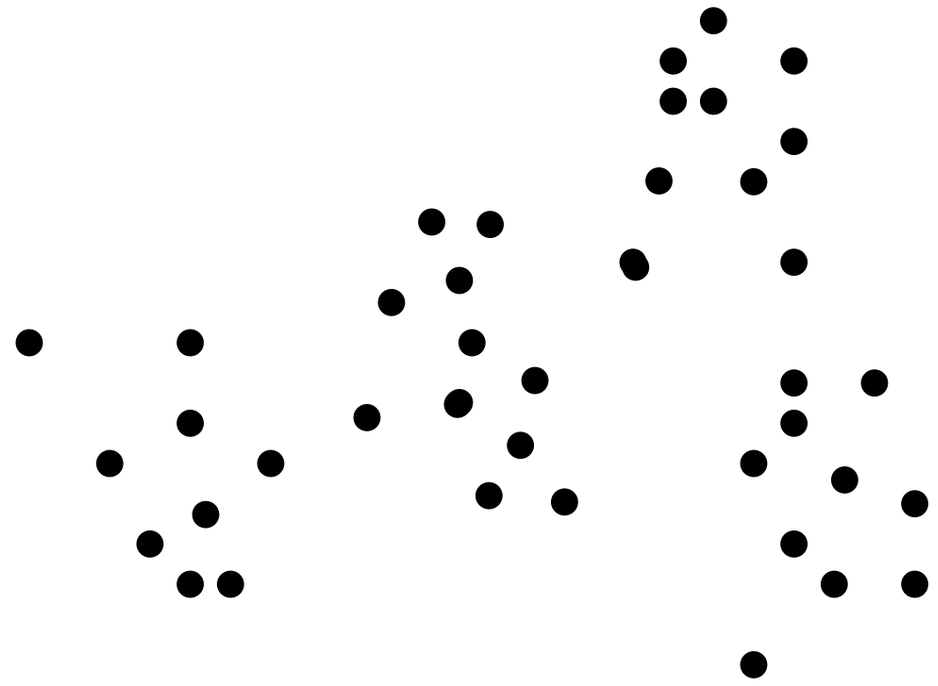
$$E = \sum_{i=1}^{N} d(x_i, L)^2$$

This is a NP hard problem!

The Lloyd's algorithm: a very simple local search algorithm $\rightarrow$ local minimum.

The k-means algorithm

The Lloyd's algorithm



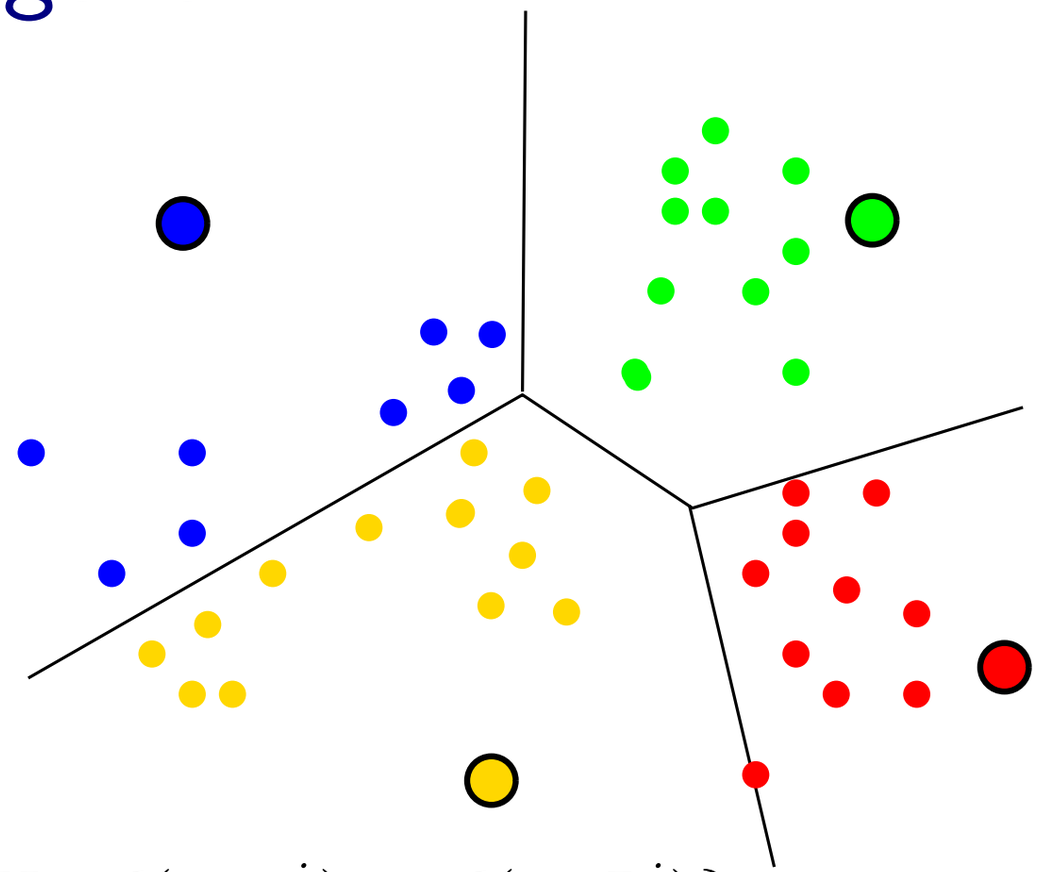
- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

The k-means algorithm

The Lloyd's algorithm



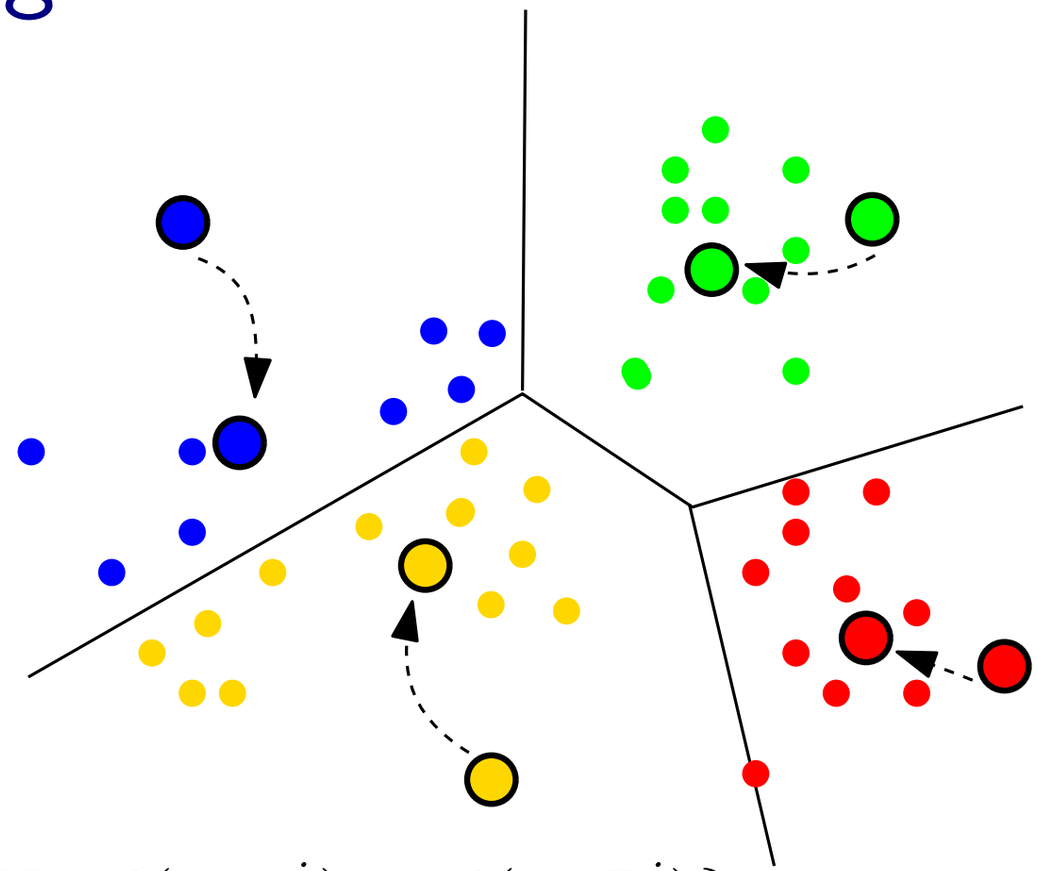
- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

The k-means algorithm

The Lloyd's algorithm



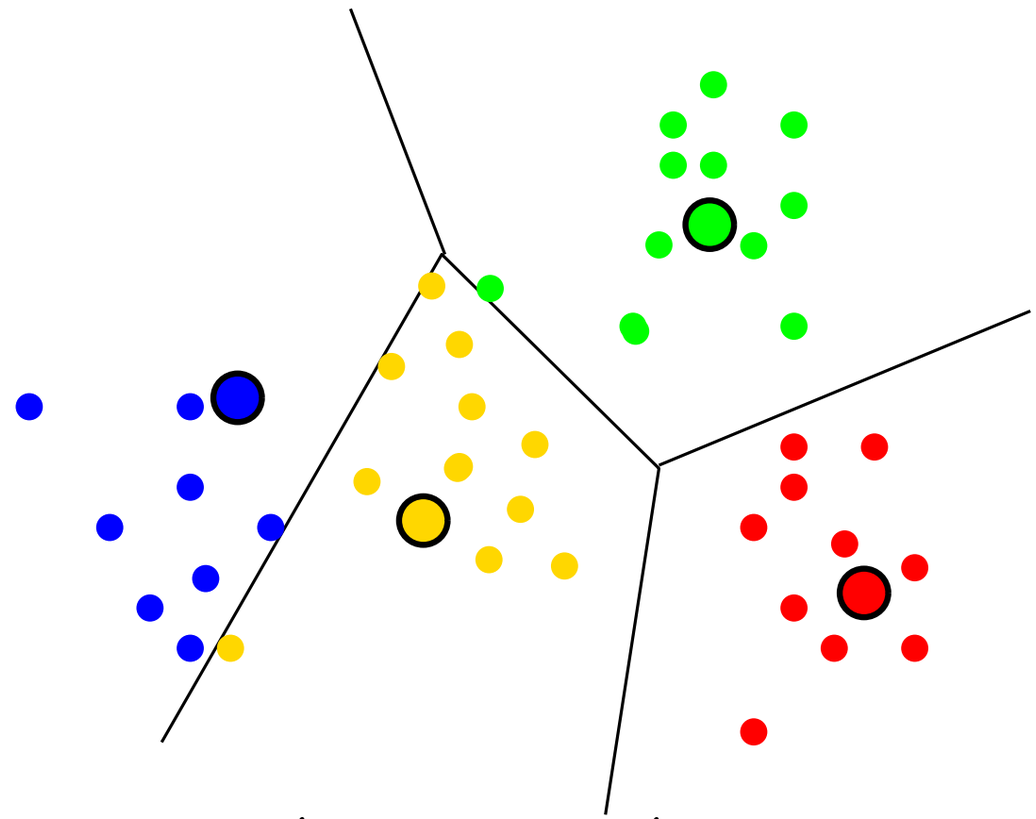
- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

The k-means algorithm

The Lloyd's algorithm



- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

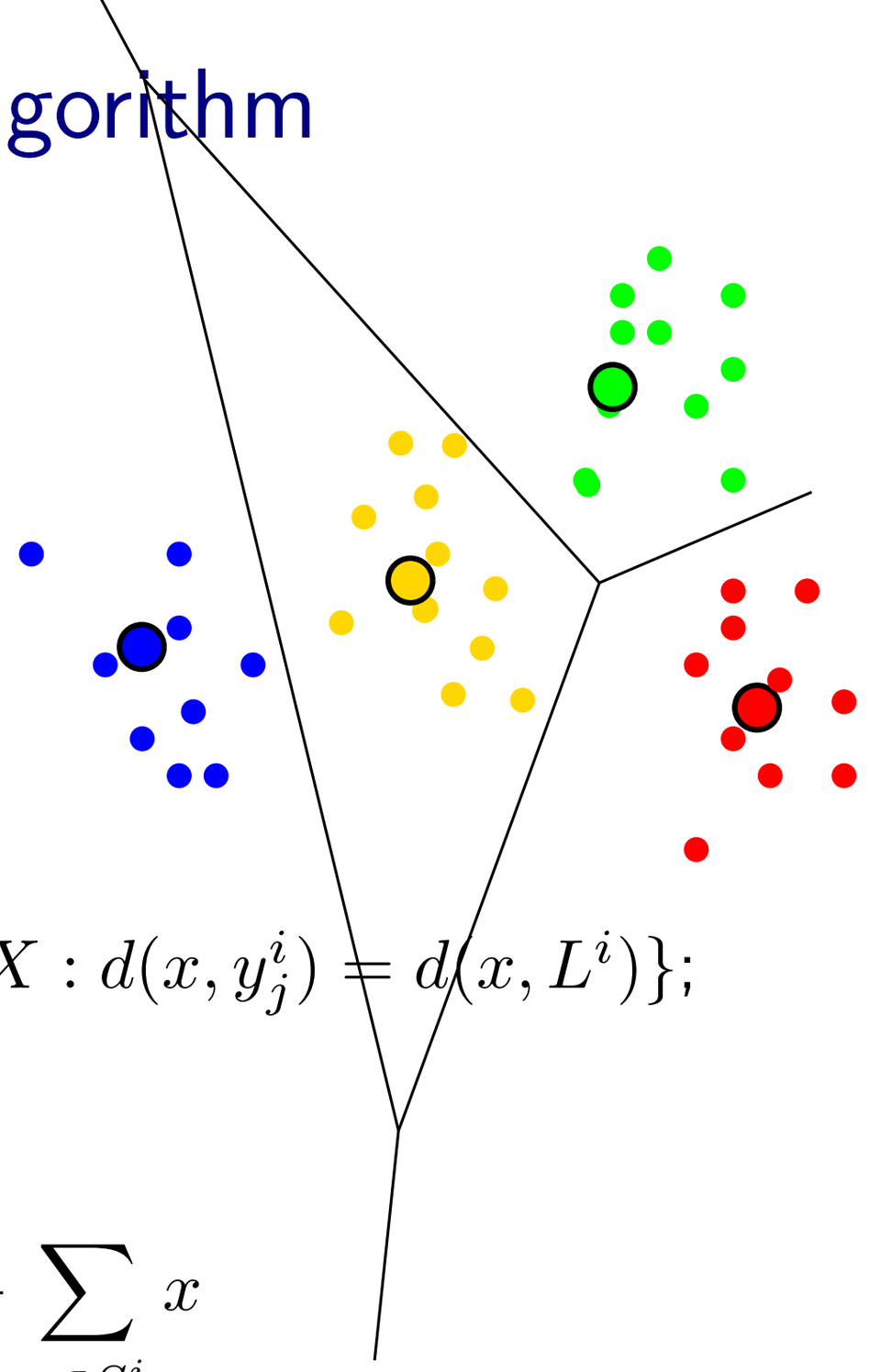
The k-means algorithm

The Lloyd's algorithm

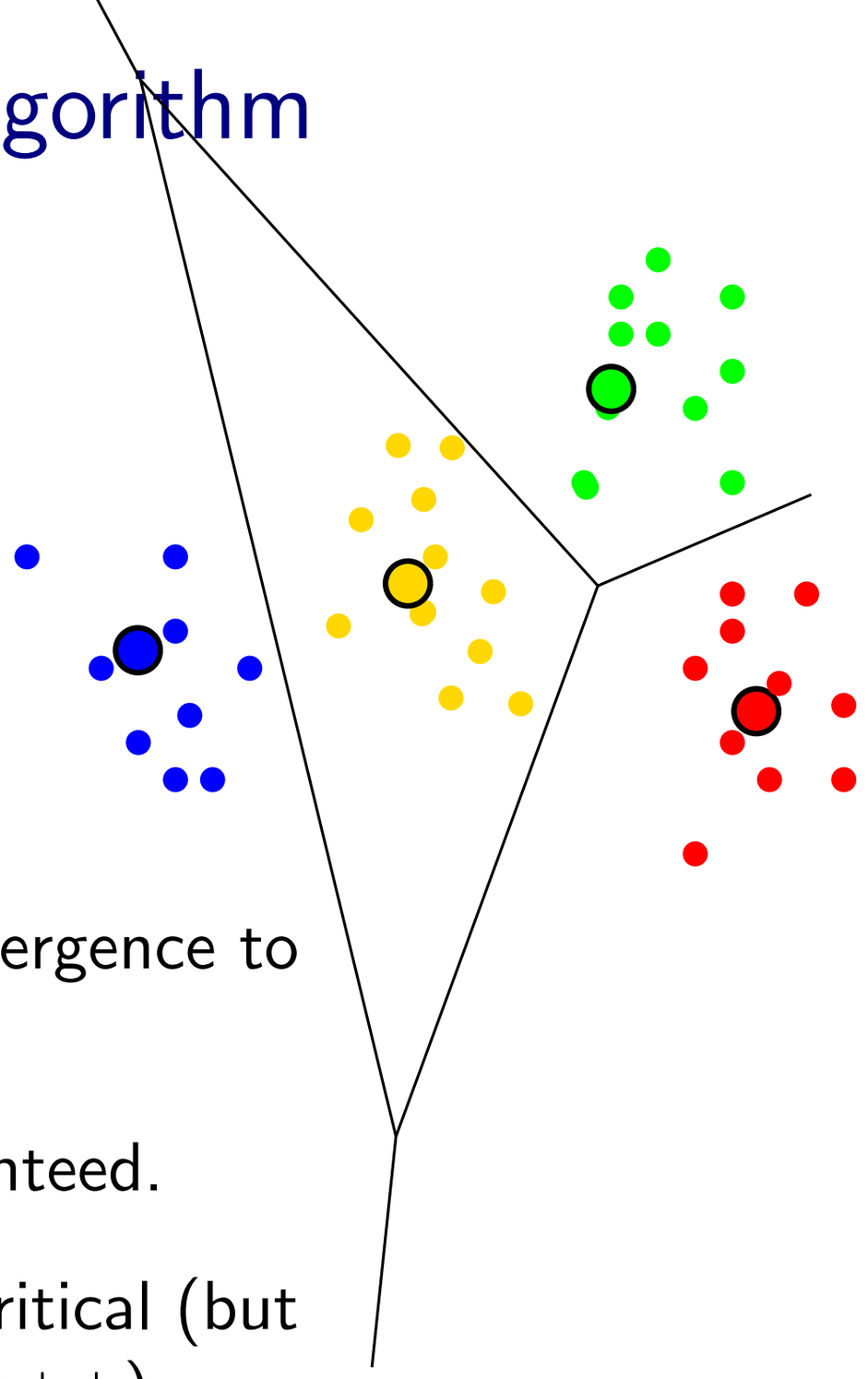
- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence



The k-means algorithm



Warning:

- Lloyd's algorithm does not ensure convergence to a global minimum!
- The speed of convergence is not guaranteed.
- the choice of the initial seeds may be critical (but there exists some strategies $\rightarrow$ k-means++).

Hierarchical clustering algorithms

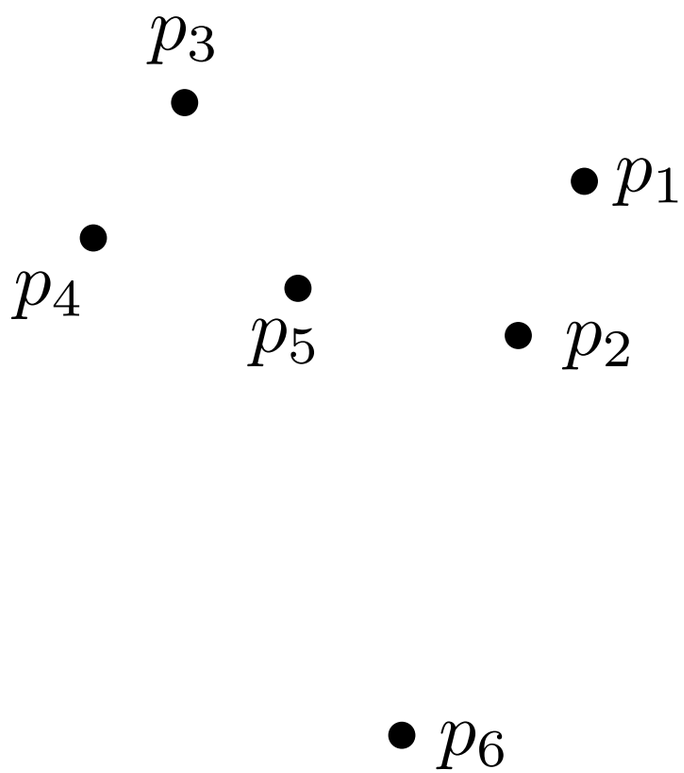
Build a hierarchy of clusters (nested family of clustering partitions)

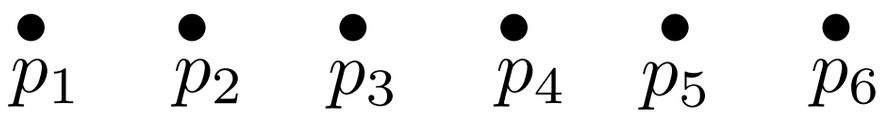
Hierarchical clustering algorithms

Build a hierarchy of clusters (nested family of clustering 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. number of clusters).



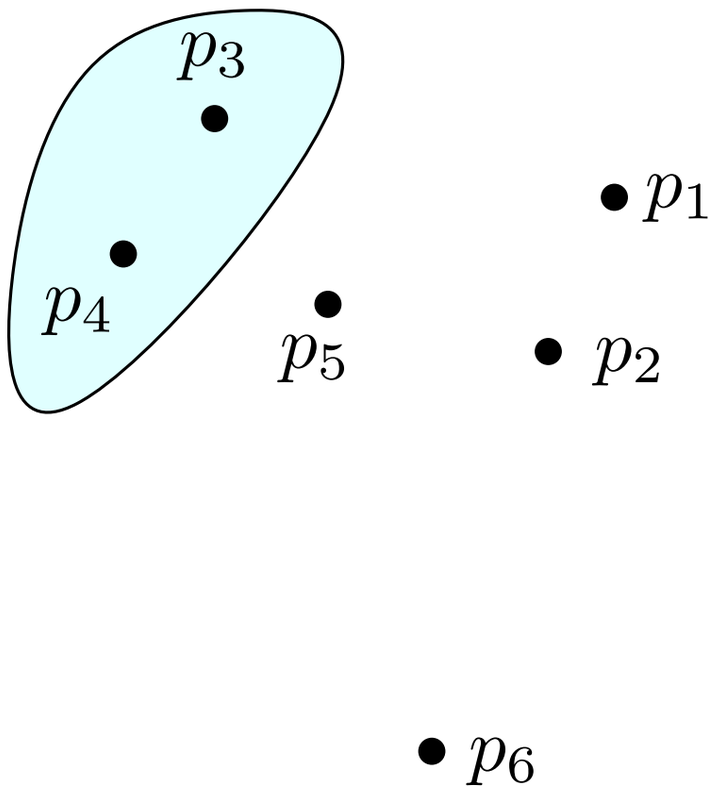


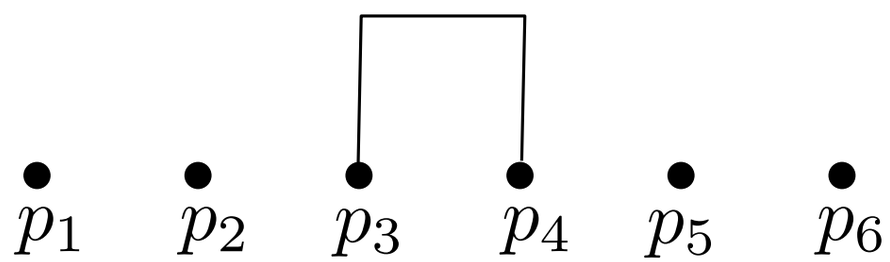
Hierarchical clustering algorithms

Build a hierarchy of clusters (nested family of clustering 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. number of clusters).



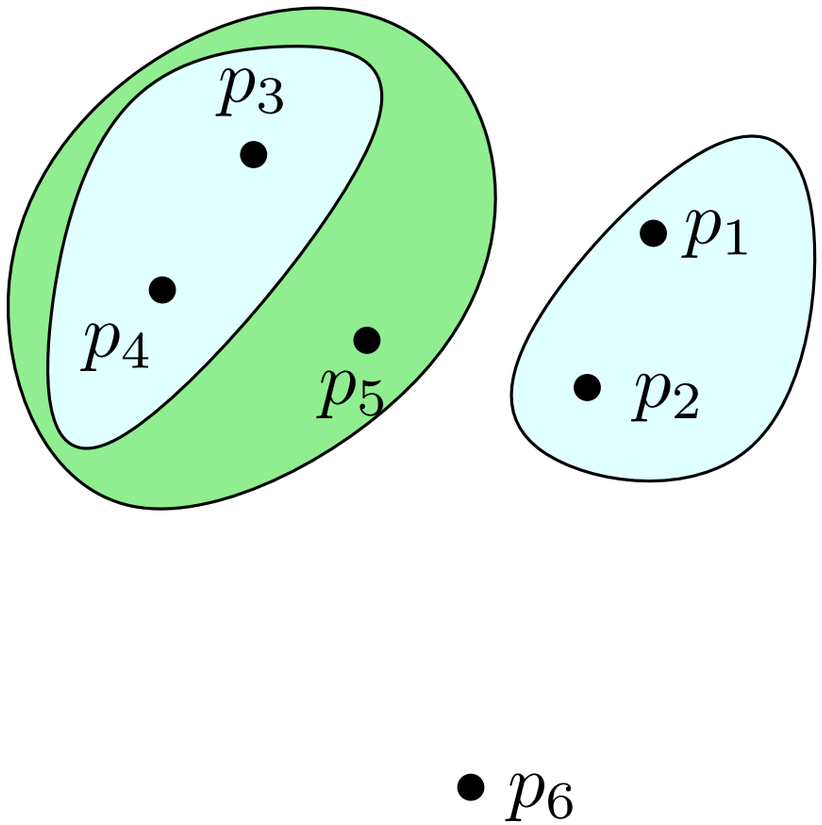


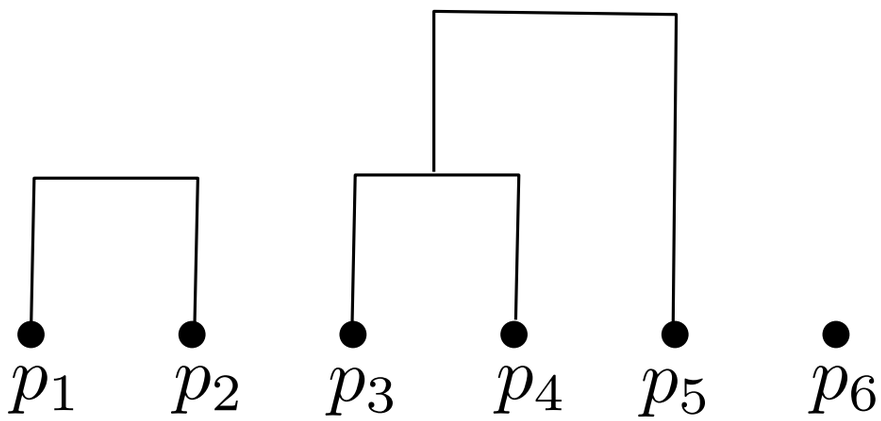
Hierarchical clustering algorithms

Build a hierarchy of clusters (nested family of clustering 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. number of clusters).



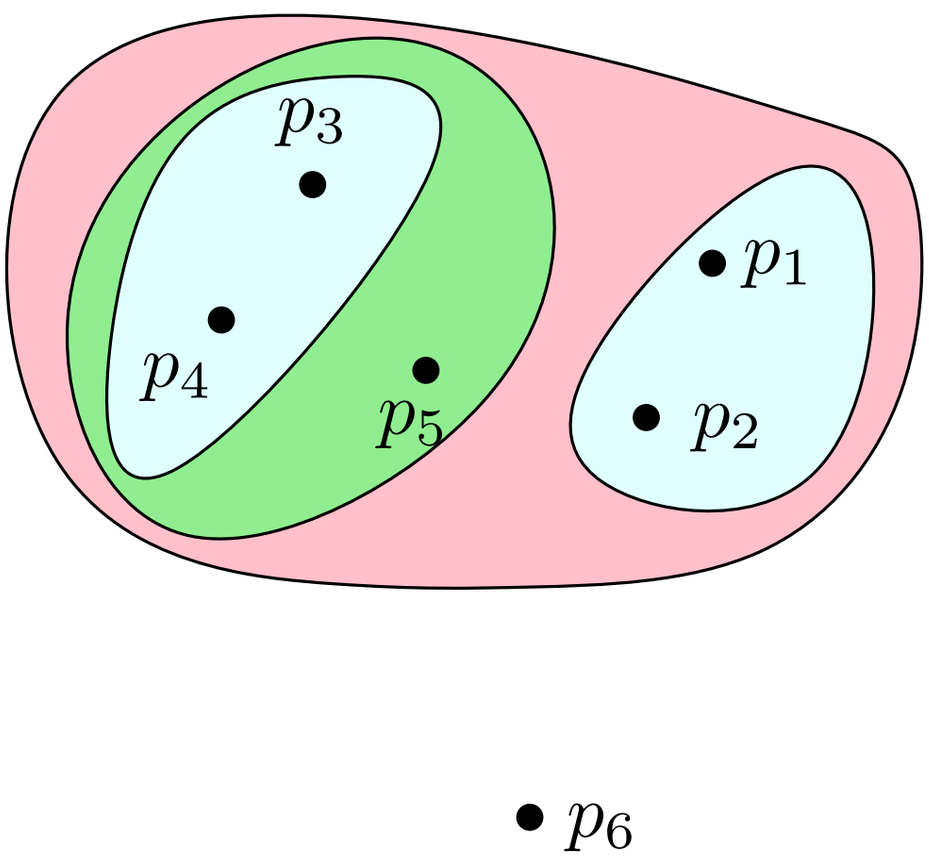


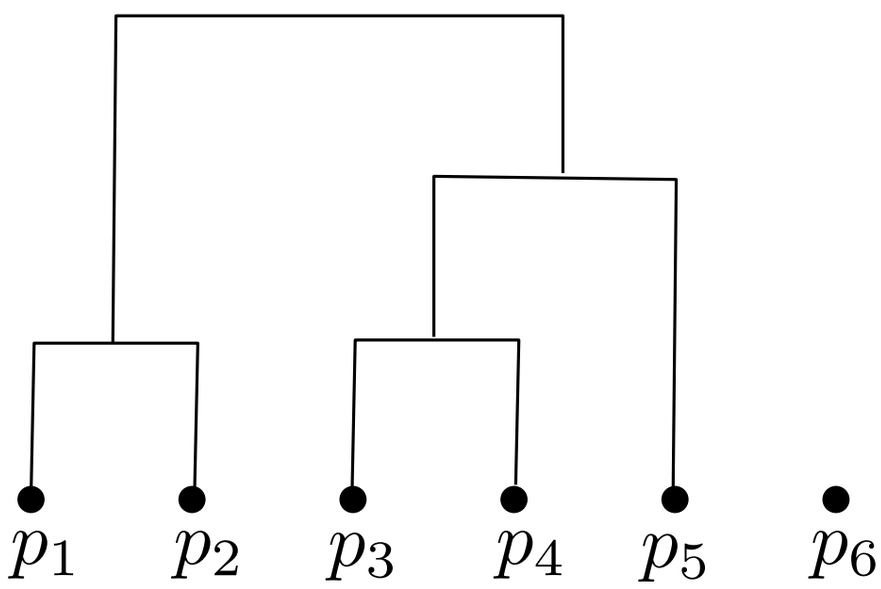
Hierarchical clustering algorithms

Build a hierarchy of clusters (nested family of clustering 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. number of clusters).



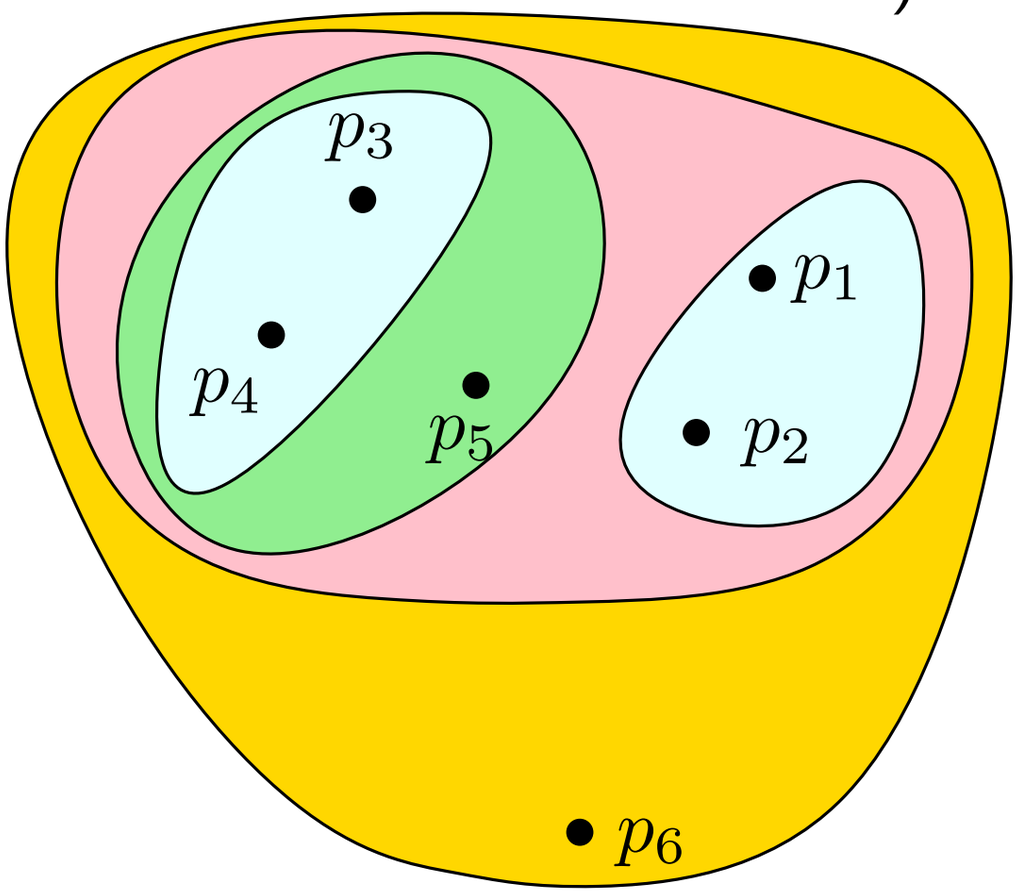


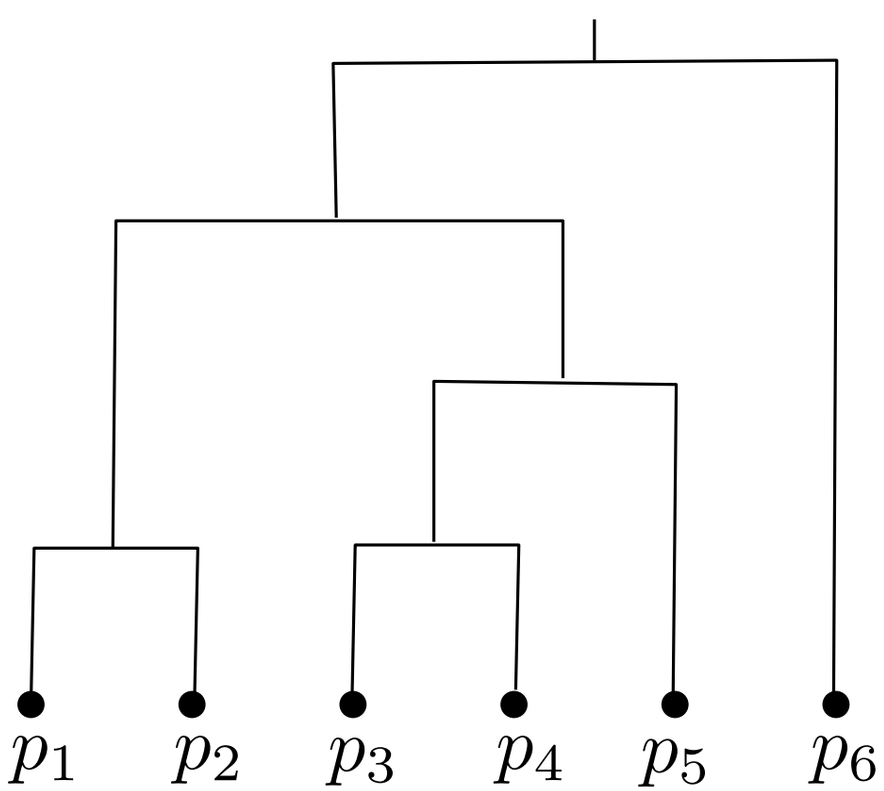
Hierarchical clustering algorithms

Build a hierarchy of clusters (nested family of clustering 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. number of clusters).



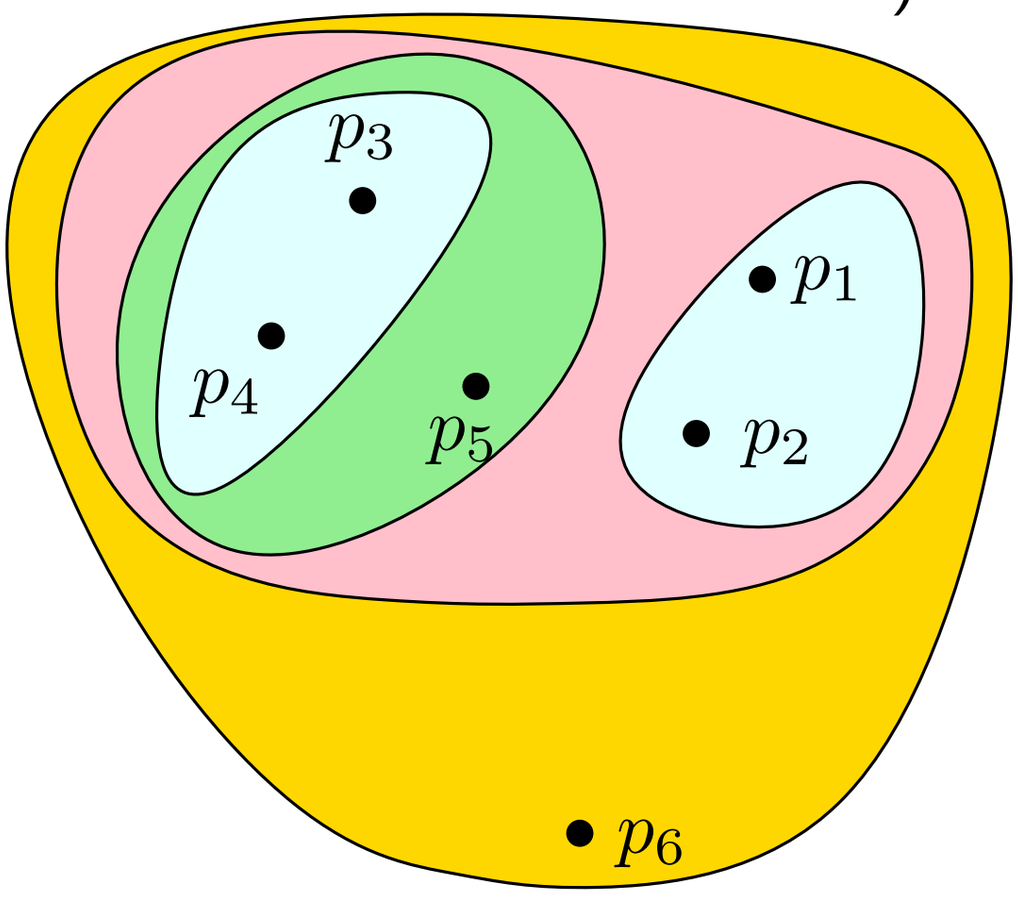


Hierarchical clustering algorithms

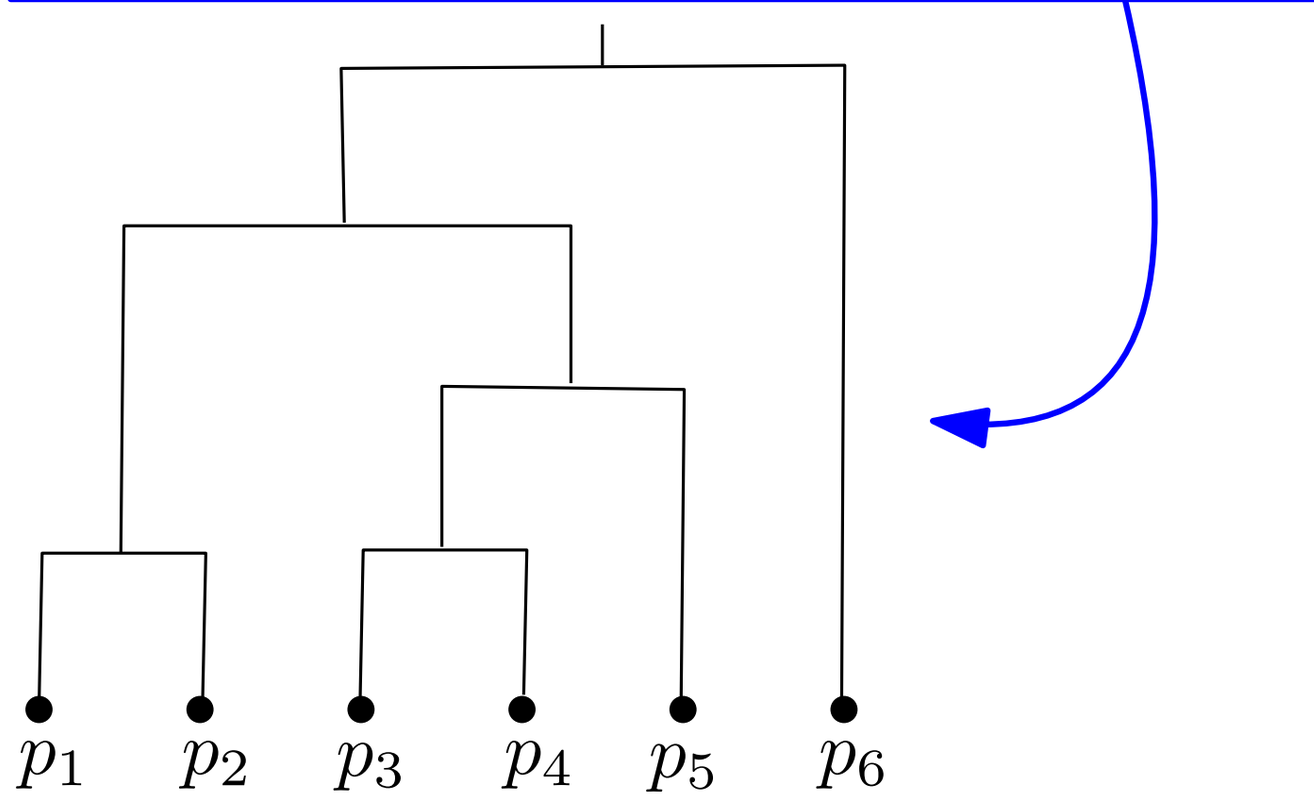
Build a hierarchy of clusters (nested family of clustering partitions)

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. number of clusters).



Dendogram, i;e. a tree such that:
- each leaf node is a singleton,
- each node represent 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

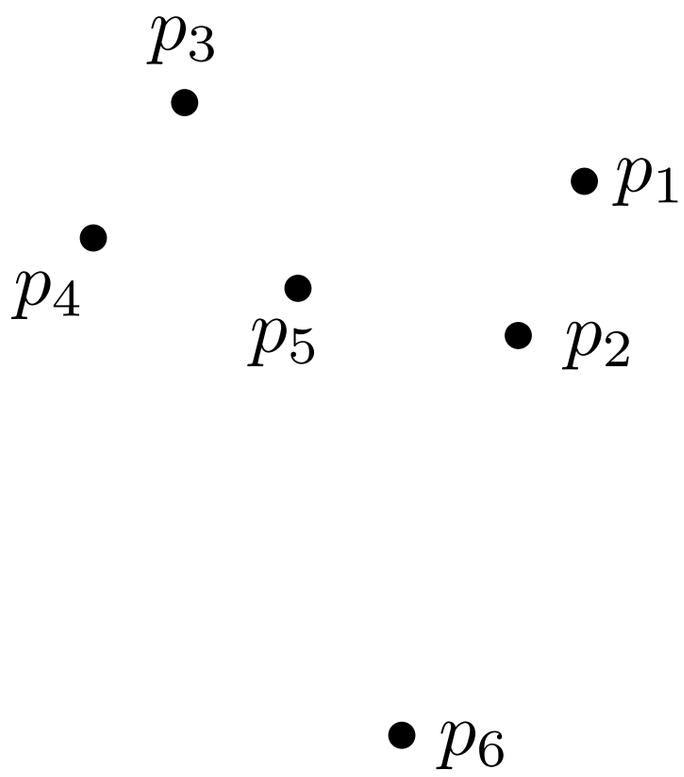
Build a hierarchy of clusters (nested family of clustering partitions)

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. number of clusters).

Dividing (top-down)

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



Hierarchical clustering algorithms

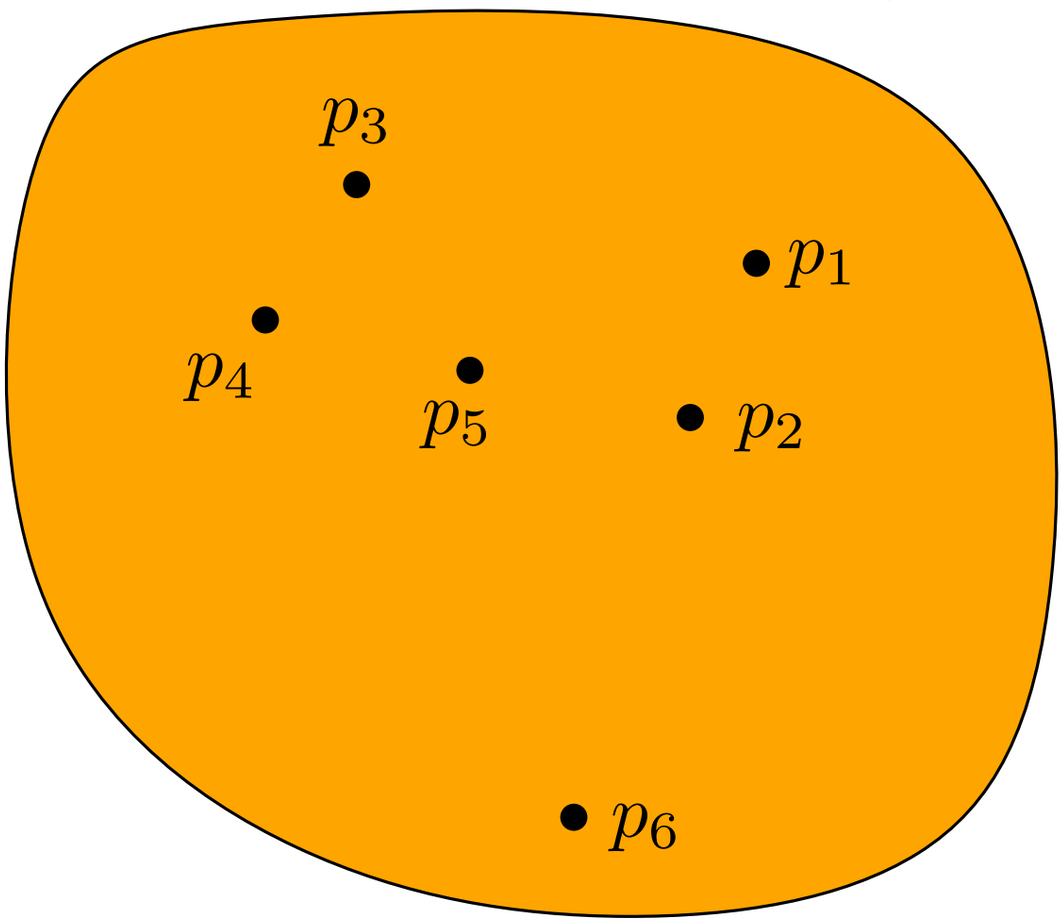
Build a hierarchy of clusters (nested family of clustering partitions)

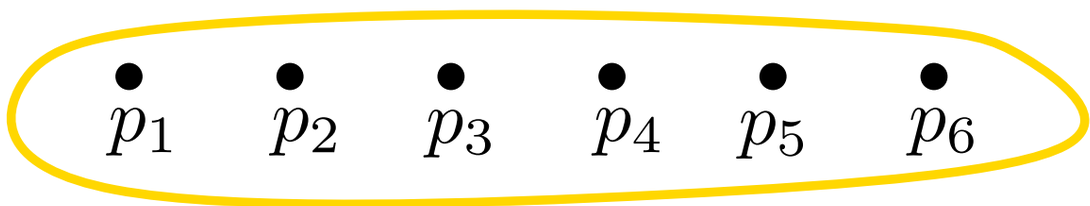
Agglomerative (bottom-up)

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

Dividing (top-down)

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





Hierarchical clustering algorithms

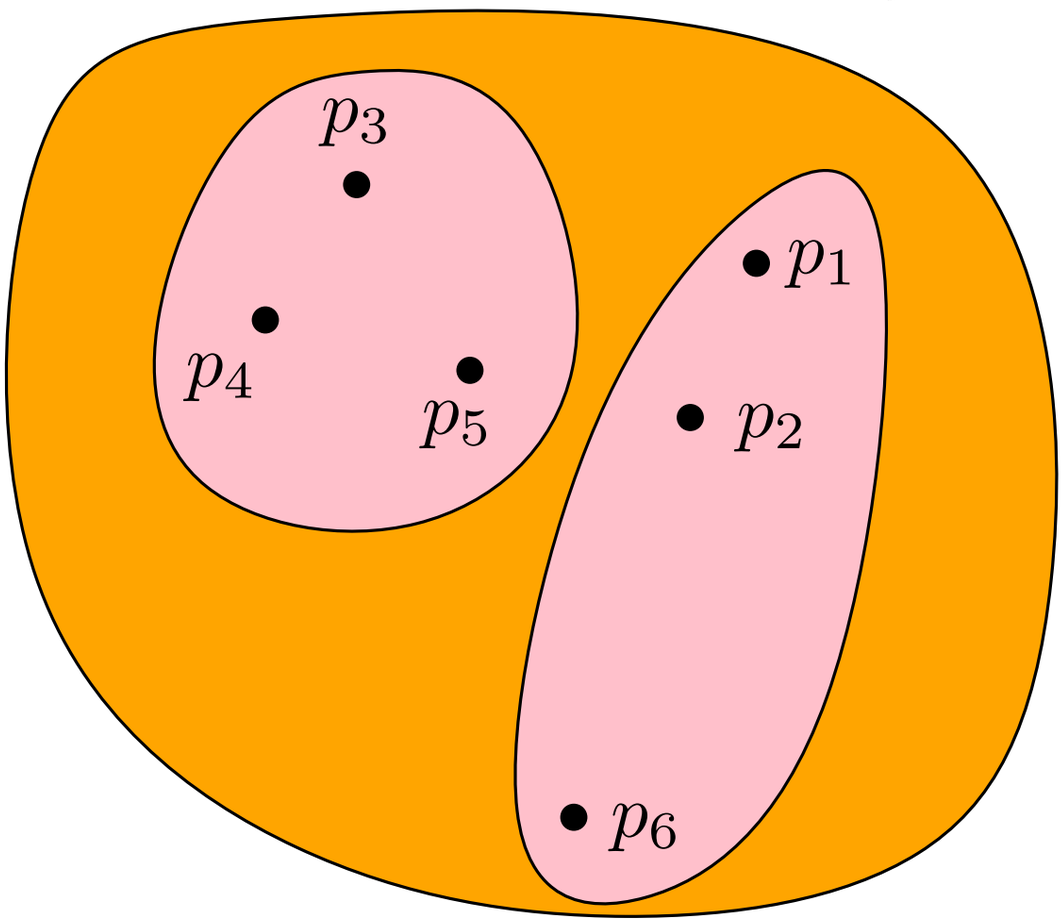
Build a hierarchy of clusters (nested family of clustering partitions)

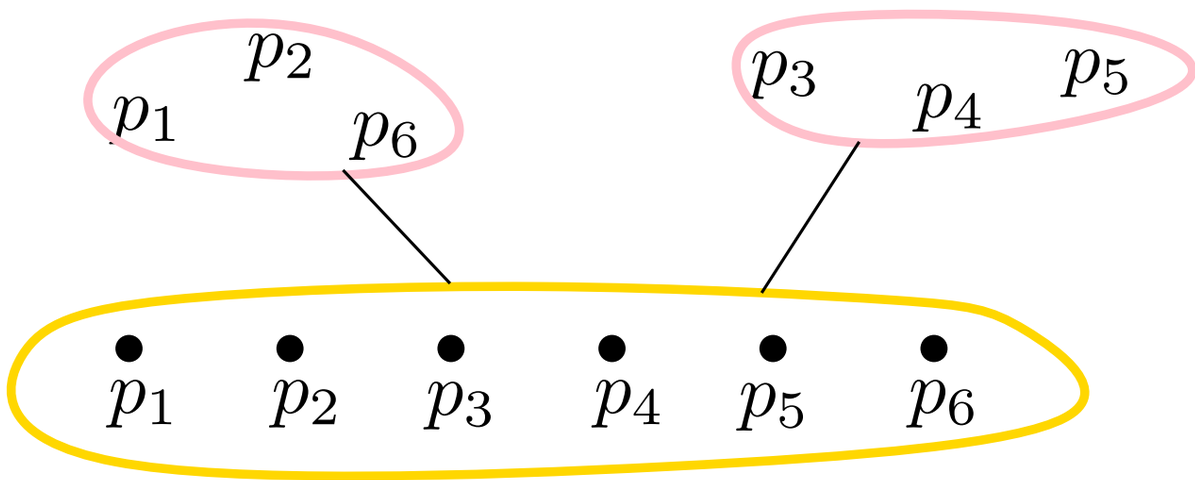
Agglomerative (bottom-up)

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

Dividing (top-down)

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





Hierarchical clustering algorithms

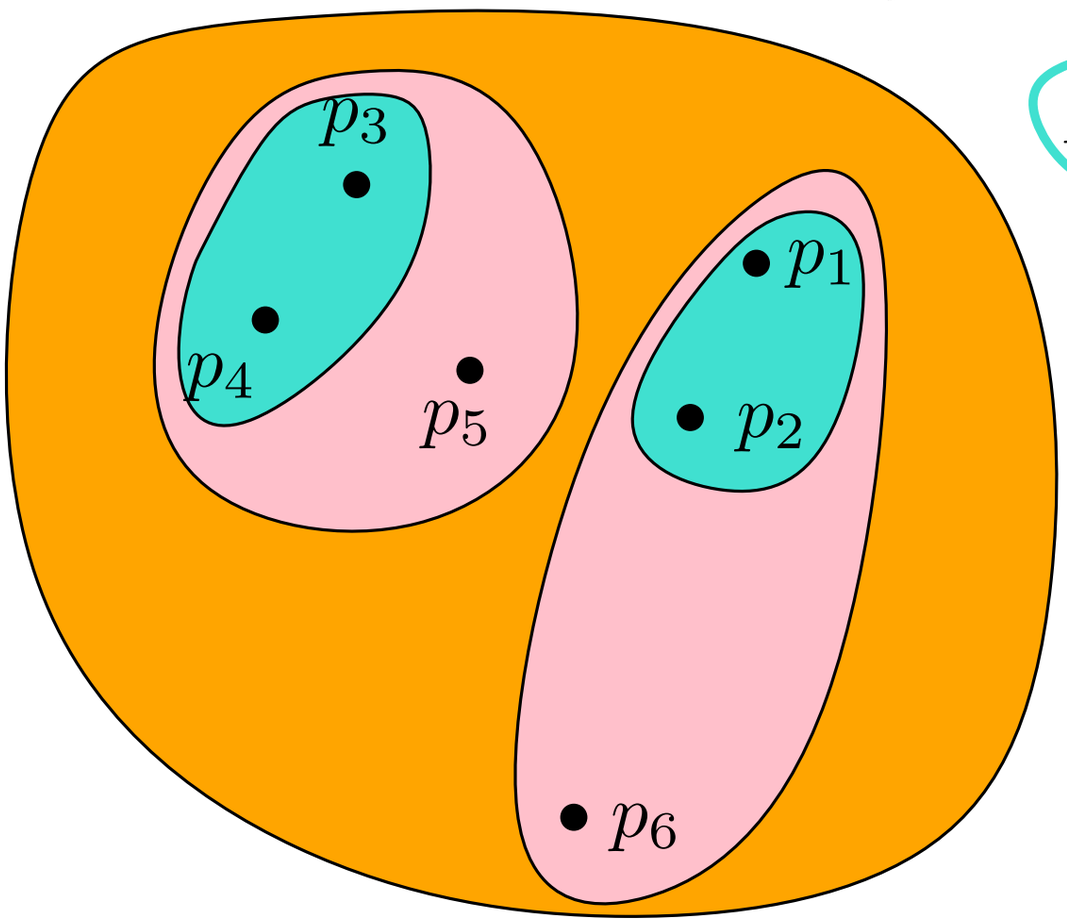
Build a hierarchy of clusters (nested family of clustering partitions)

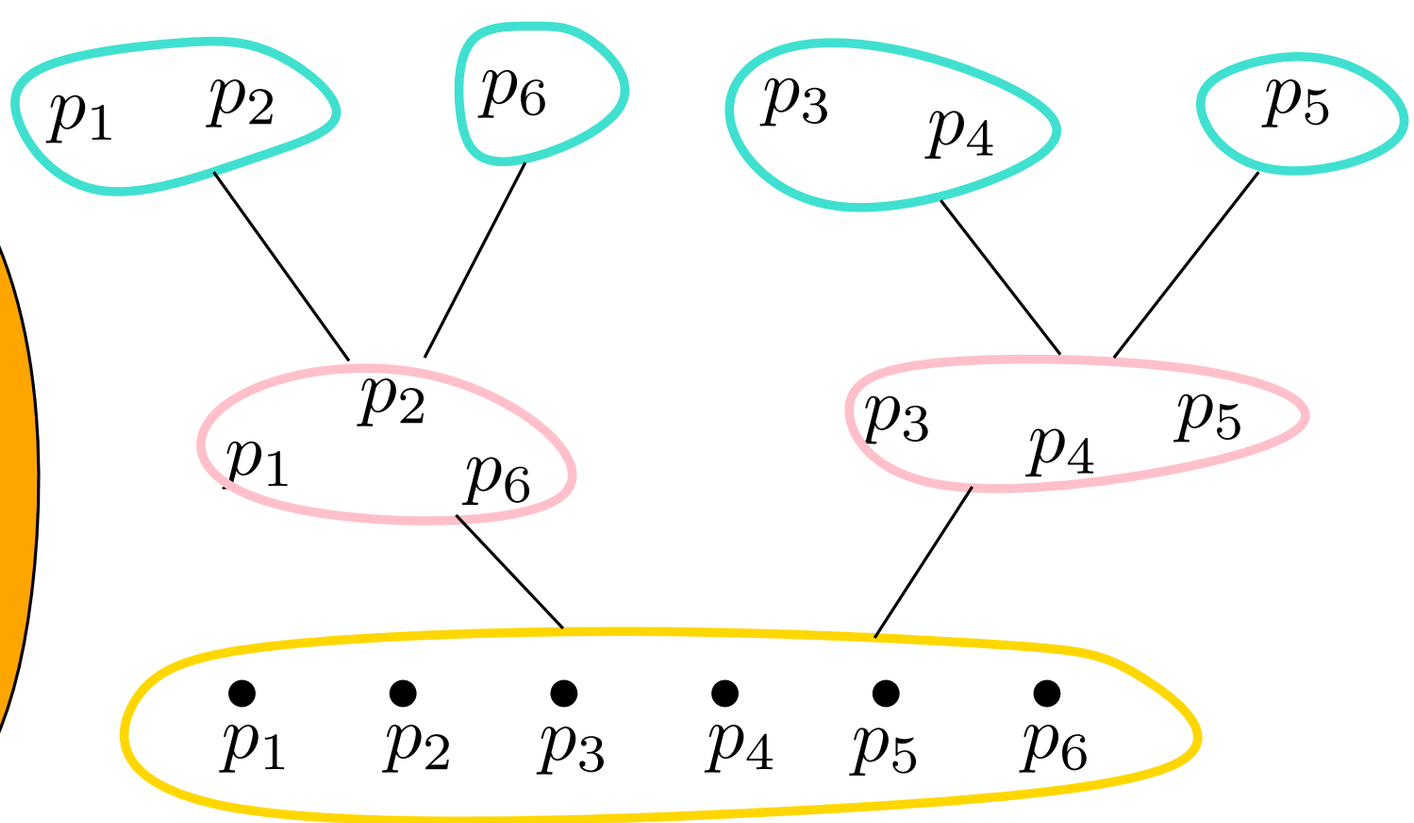
Agglomerative (bottom-up)

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

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 $X_n = \{x_1, \cdots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $(d_{i,j})$).

Given two clusters $C, C' \subseteq X_n$ let

$$d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$$

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 dendogram

Single linkage clustering

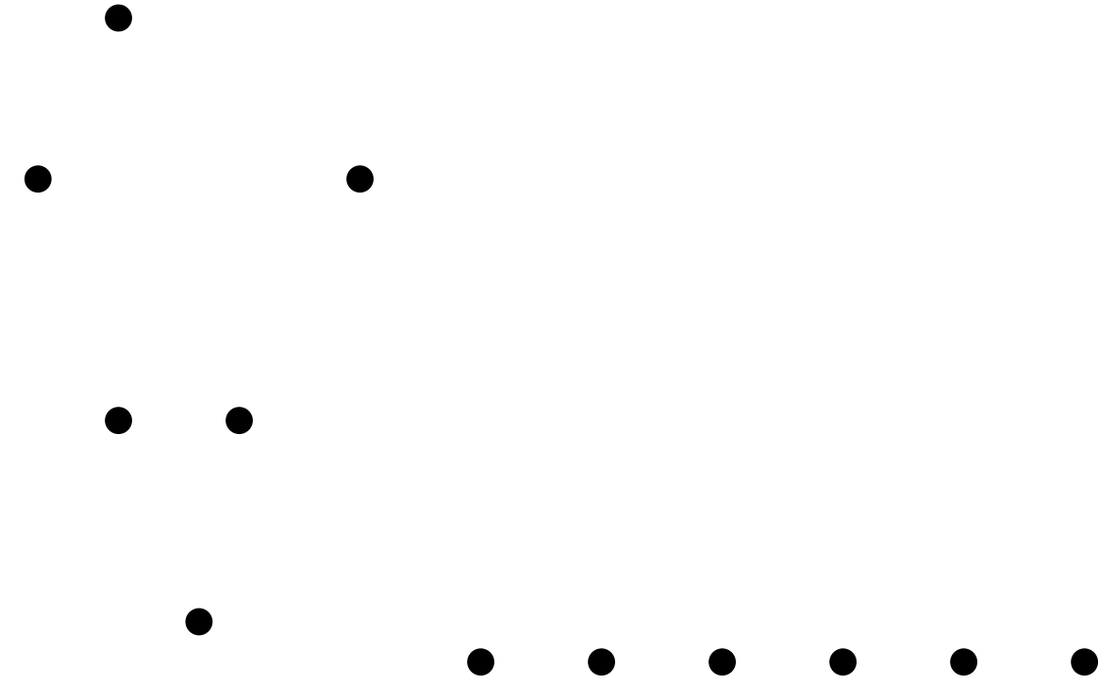
Input: A set $X_n = \{x_1, \cdots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $(d_{i,j})$).

Given two clusters $C, C' \subseteq X_n$ let

$$d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$$

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 dendogram

Single linkage clustering

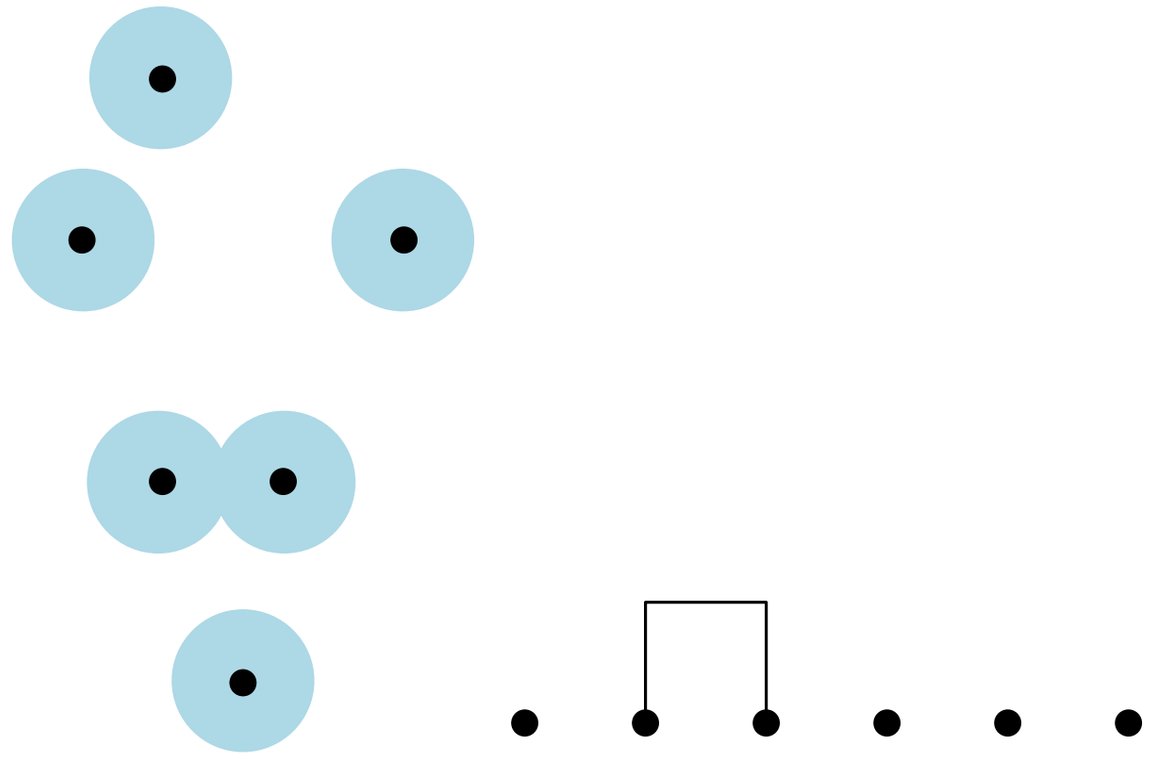
Input: A set $X_n = \{x_1, \cdots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $(d_{i,j})$).

Given two clusters $C, C' \subseteq X_n$ let

$$d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$$

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 dendogram

Single linkage clustering

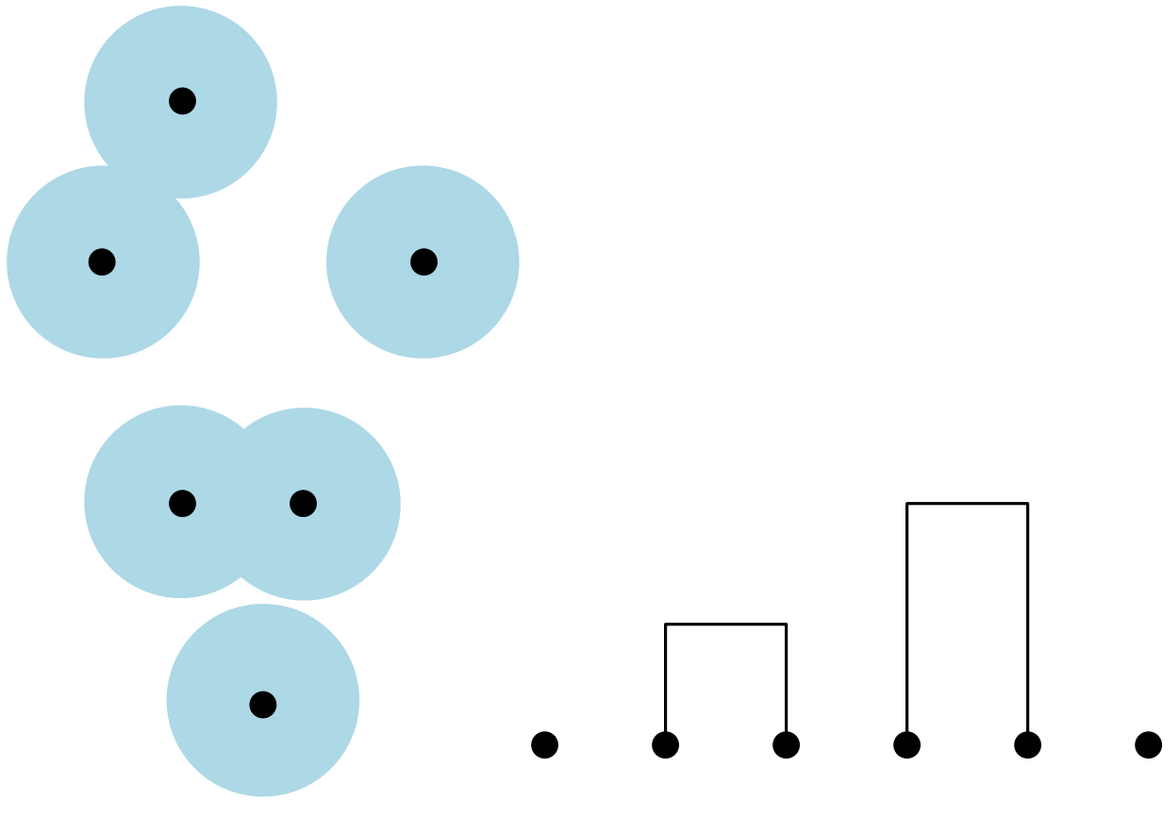
Input: A set $X_n = \{x_1, \cdots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $(d_{i,j})$).

Given two clusters $C, C' \subseteq X_n$ let

$$d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$$

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 dendogram

Single linkage clustering

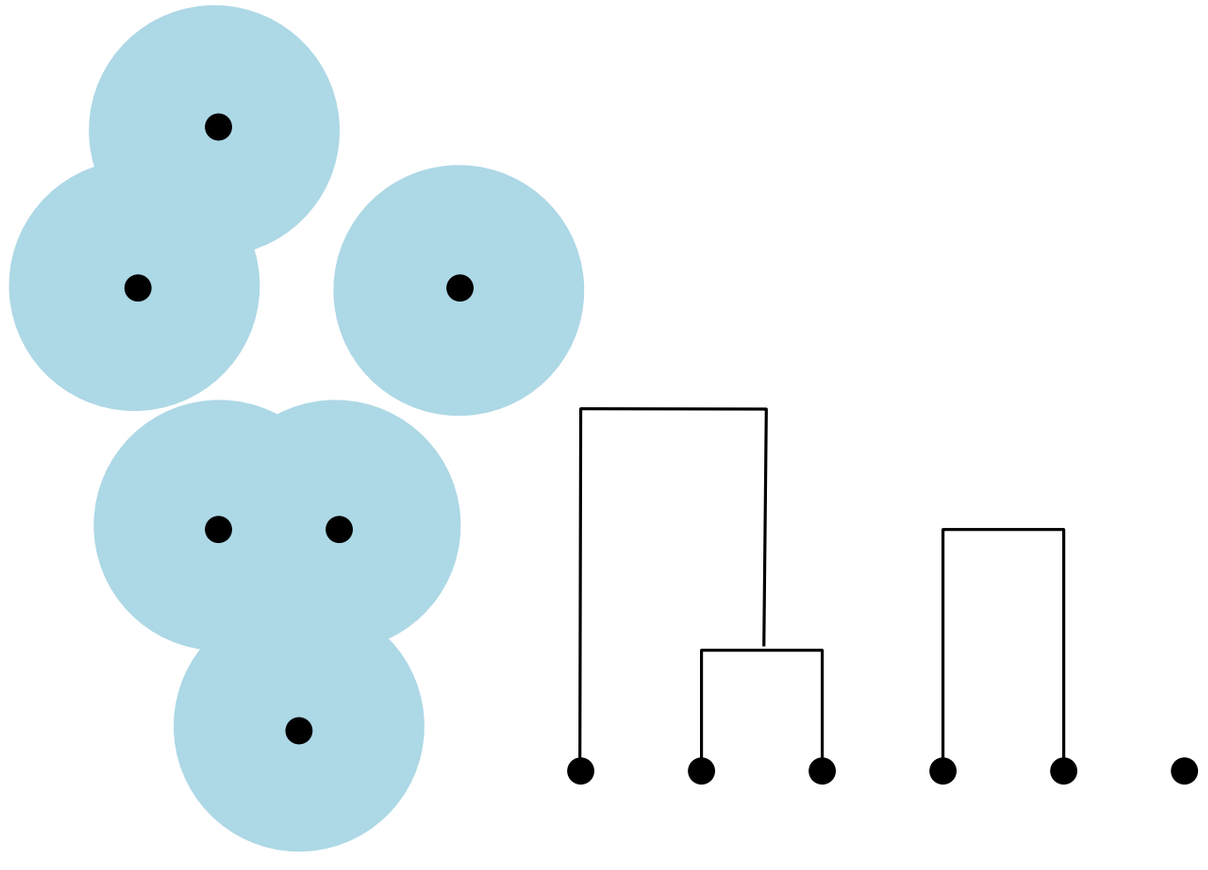
Input: A set $X_n = \{x_1, \cdots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $(d_{i,j})$).

Given two clusters $C, C' \subseteq X_n$ let

$$d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$$

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 dendogram

Single linkage clustering

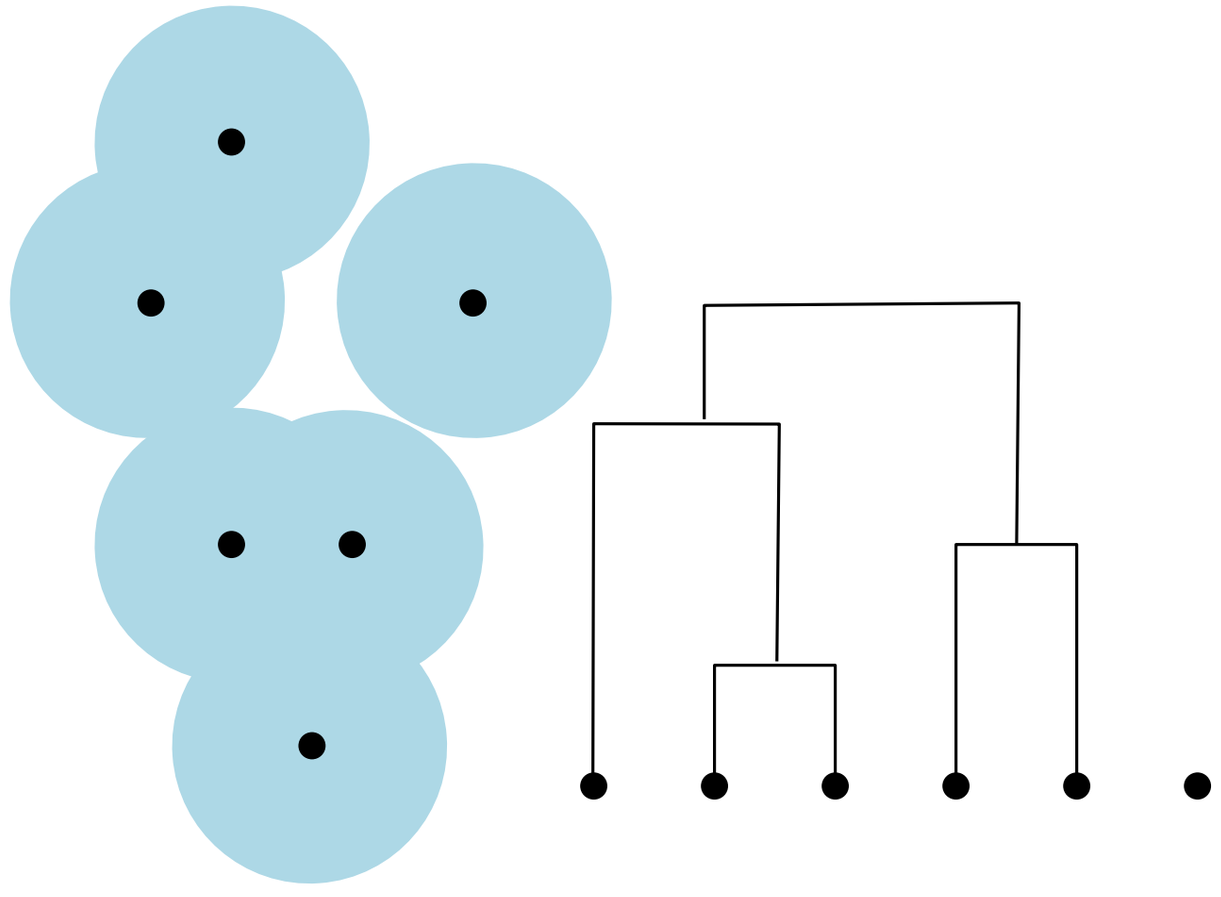
Input: A set $X_n = \{x_1, \cdots, x_n\}$ in a metric space (X, d) (or just a matrix of pairwise dissimilarities $(d_{i,j})$).

Given two clusters $C, C' \subseteq X_n$ let

$$d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$$

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 dendogram

Single linkage clustering

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

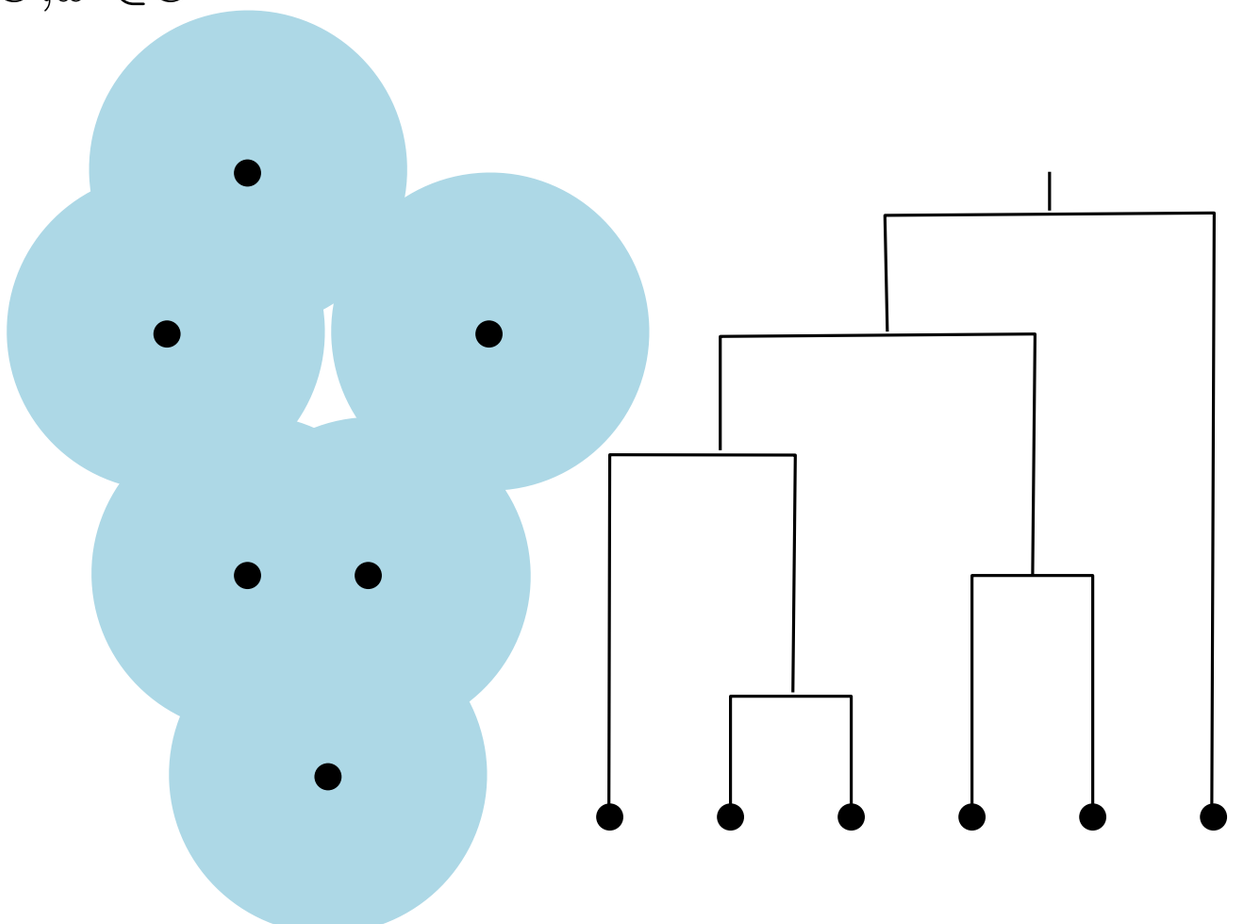
Given two clusters $C, C' \subseteq X_n$ let

$$d(C, C') = \inf_{x \in C, x' \in C'} d(x, x')$$

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

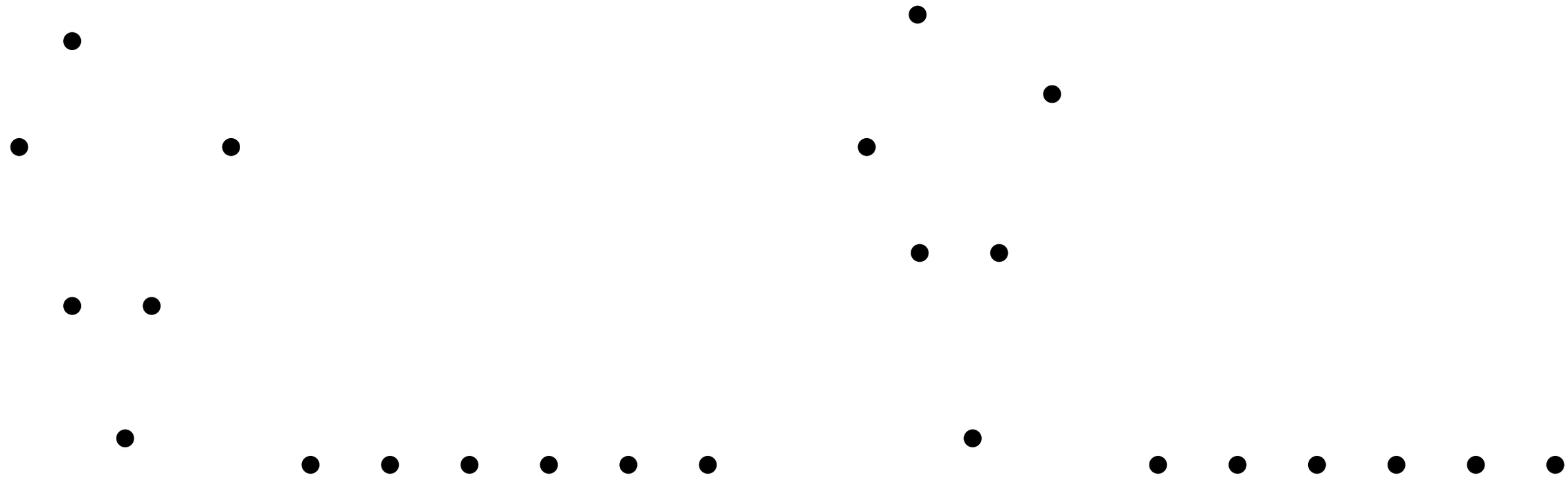
2. At each step, merge the two closest clusters until it remains a single cluster (containing all data points).

Output:: the resulting dendogram

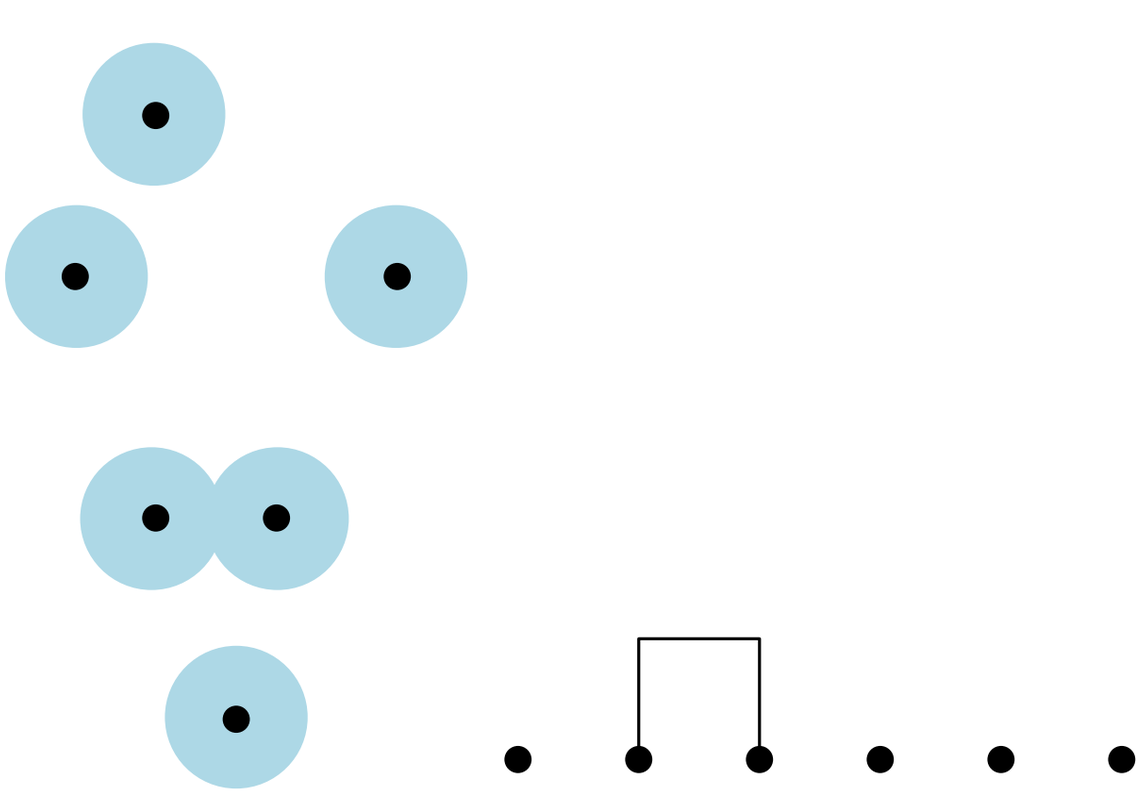


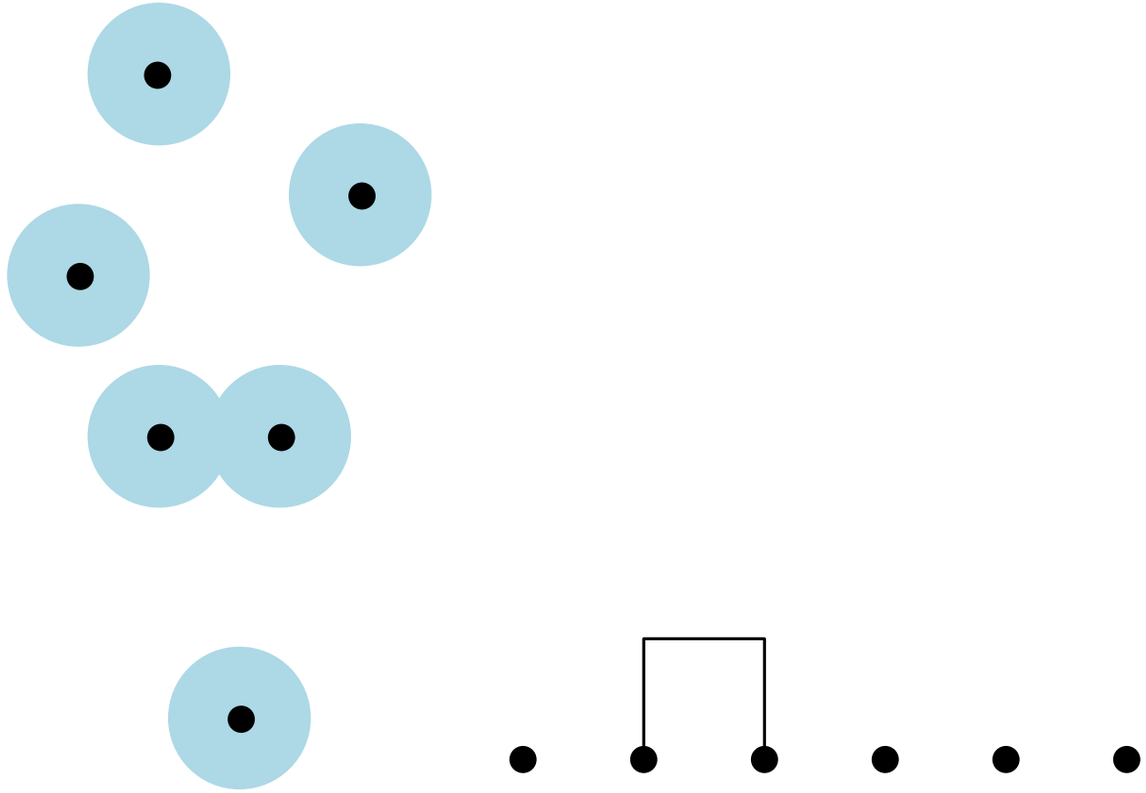
The unstability of dendograms

The unstability of dendograms

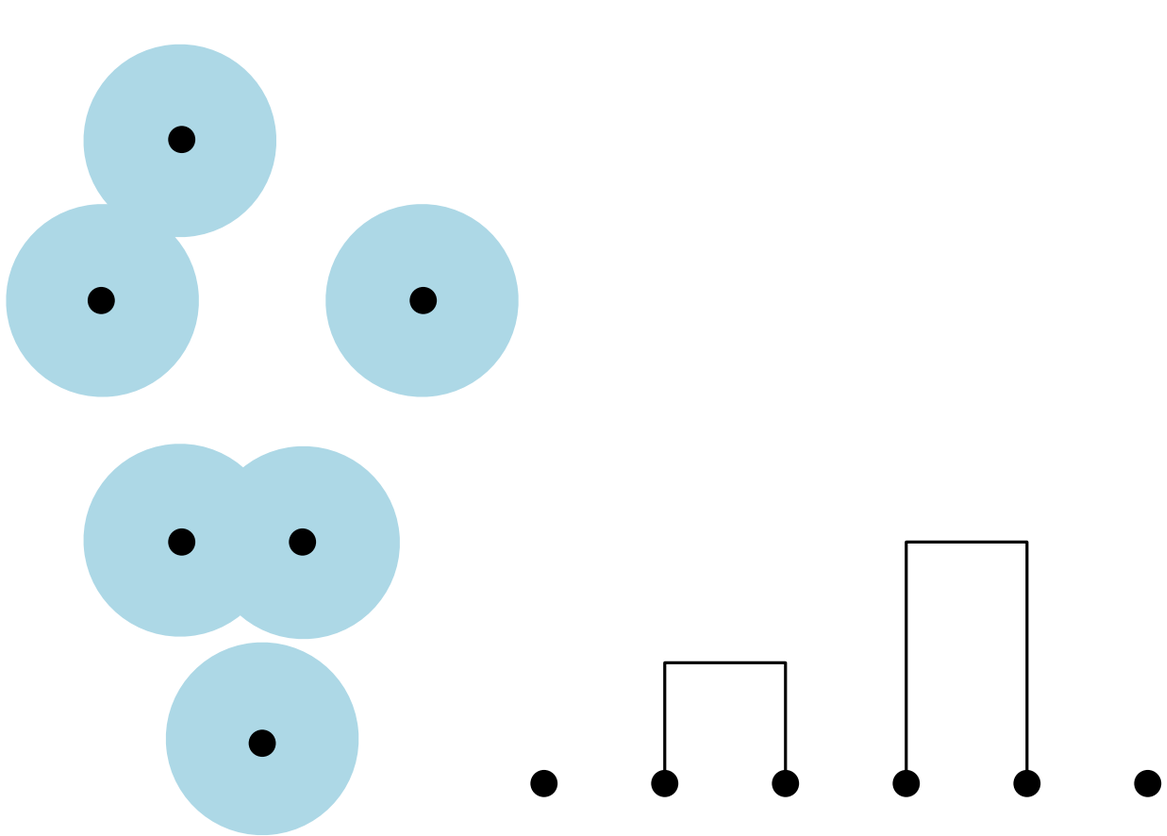


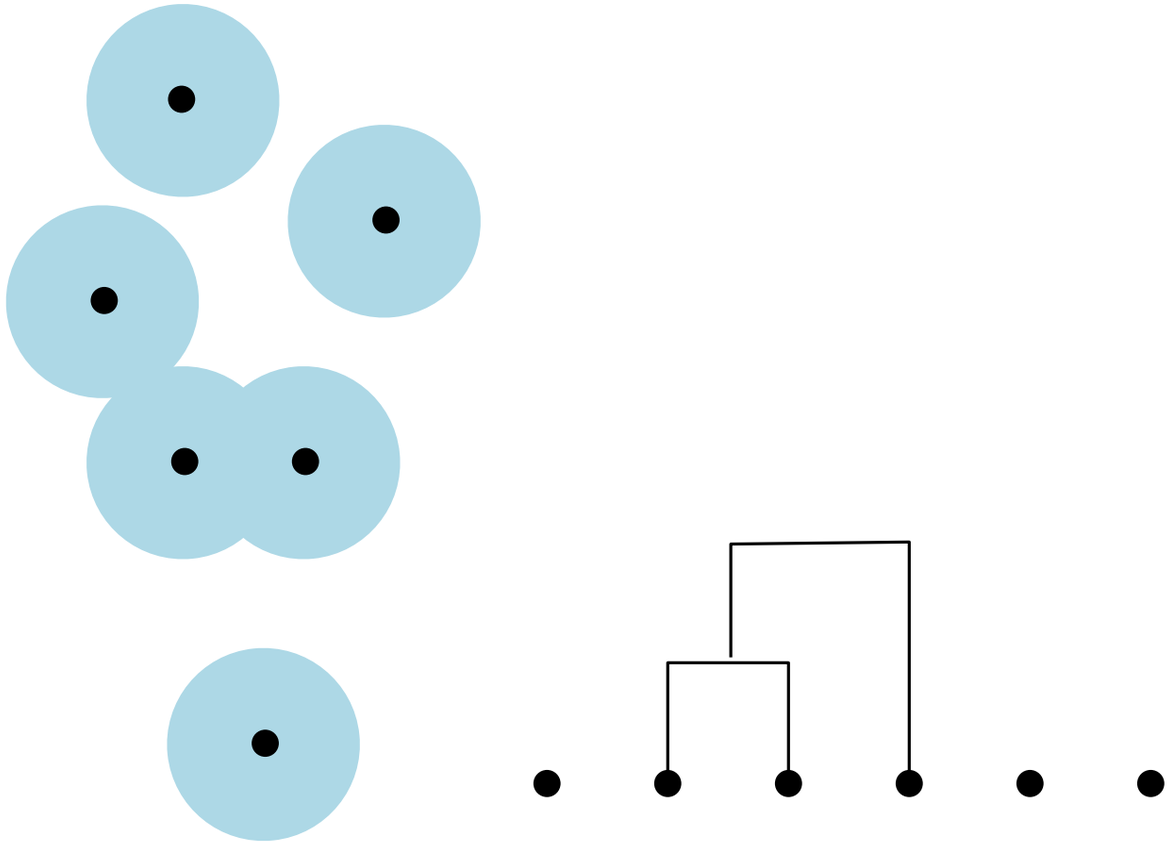
The unstability of dendograms



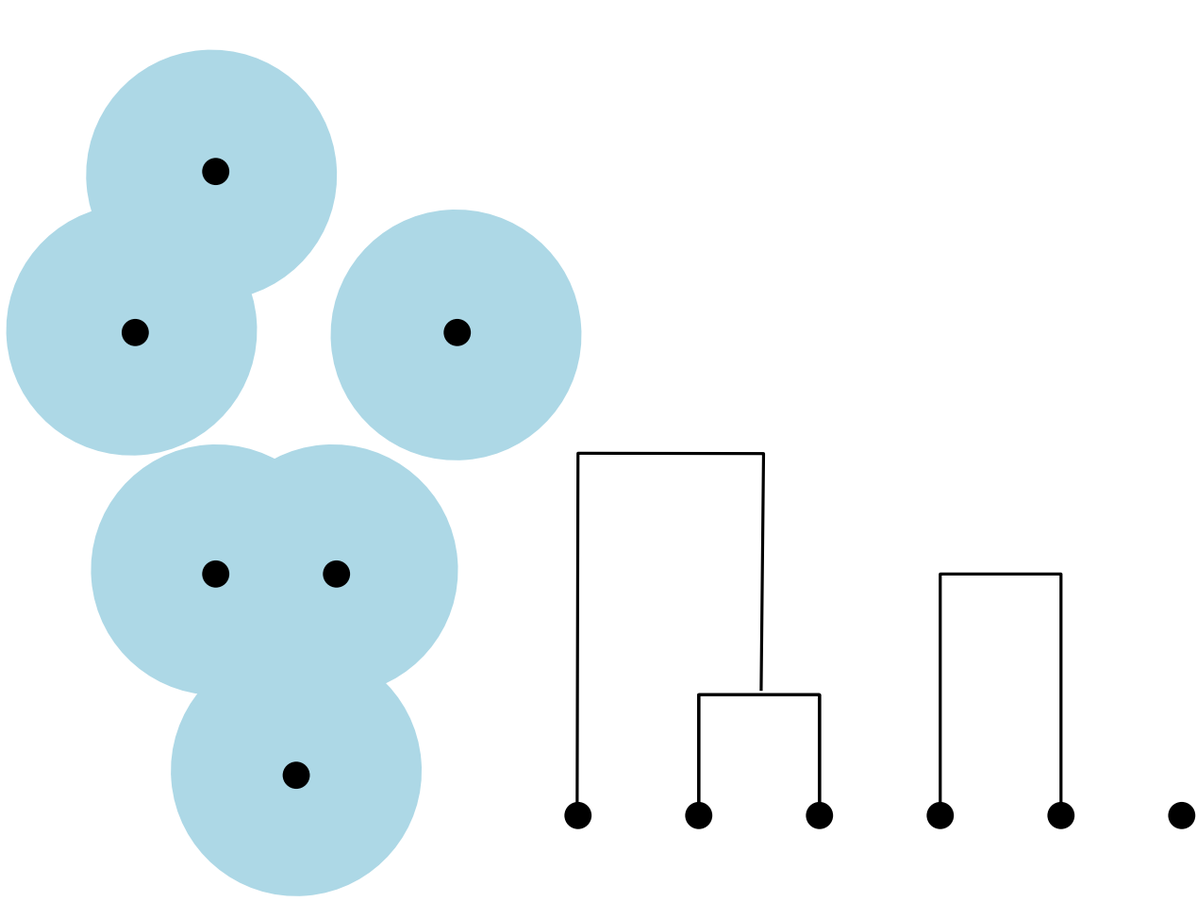


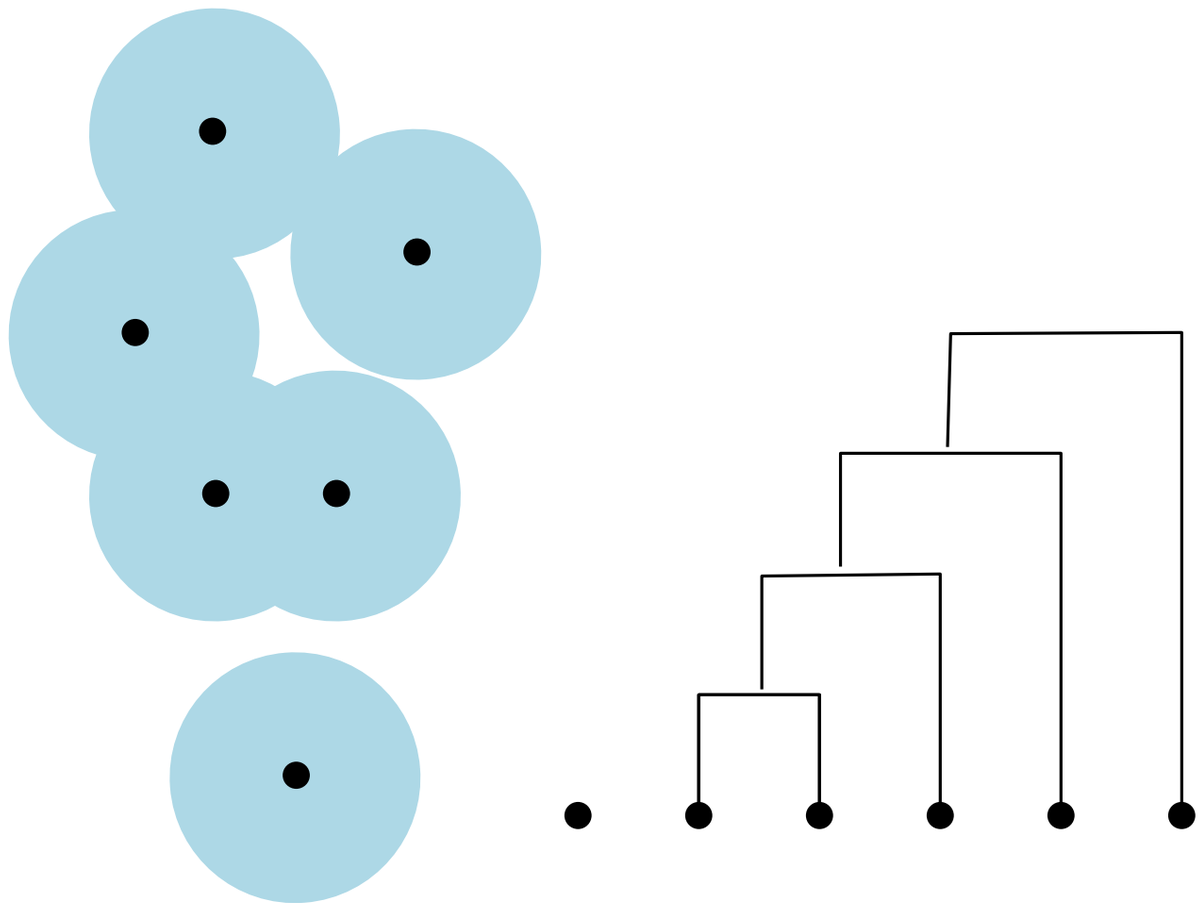
The unstability of dendograms



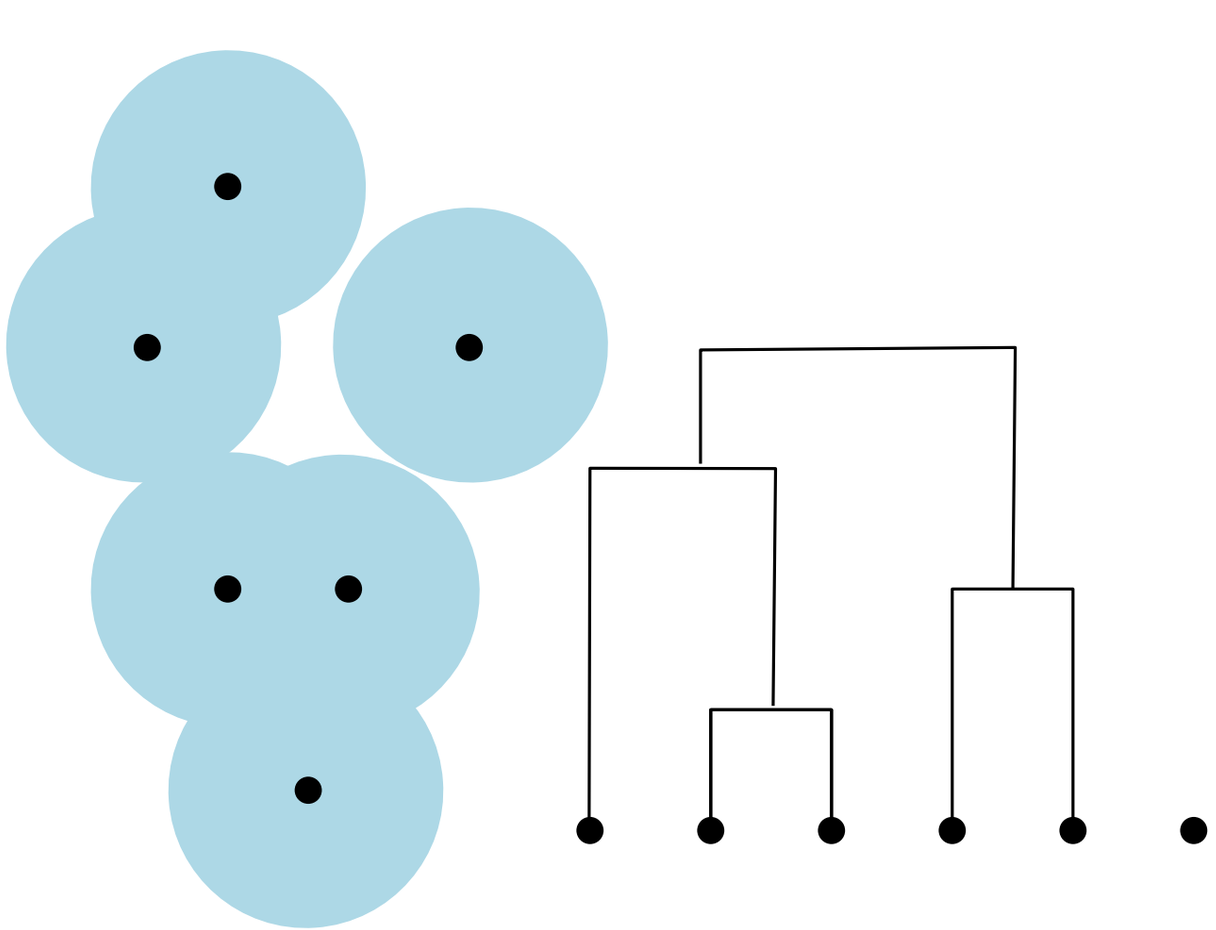


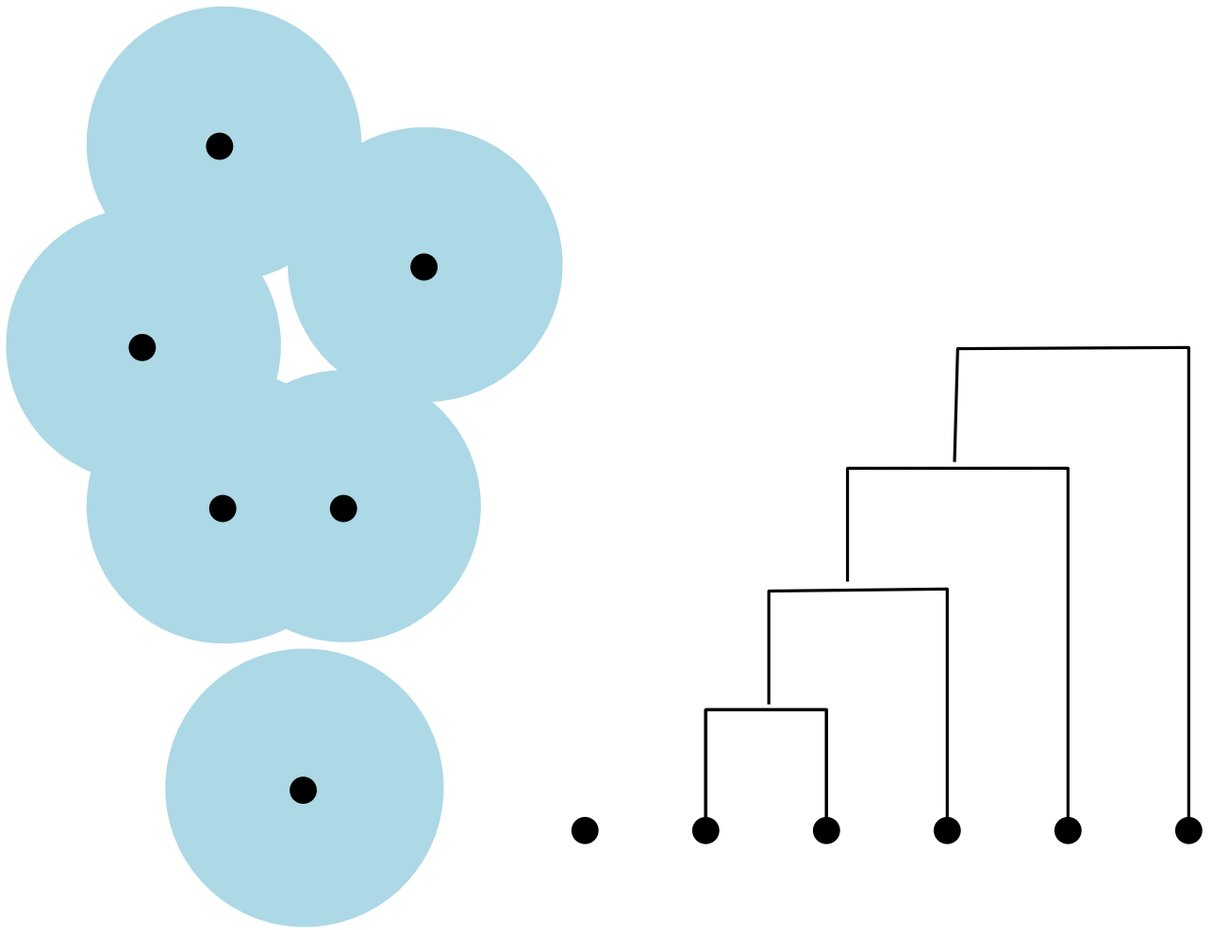
The unstability of dendograms



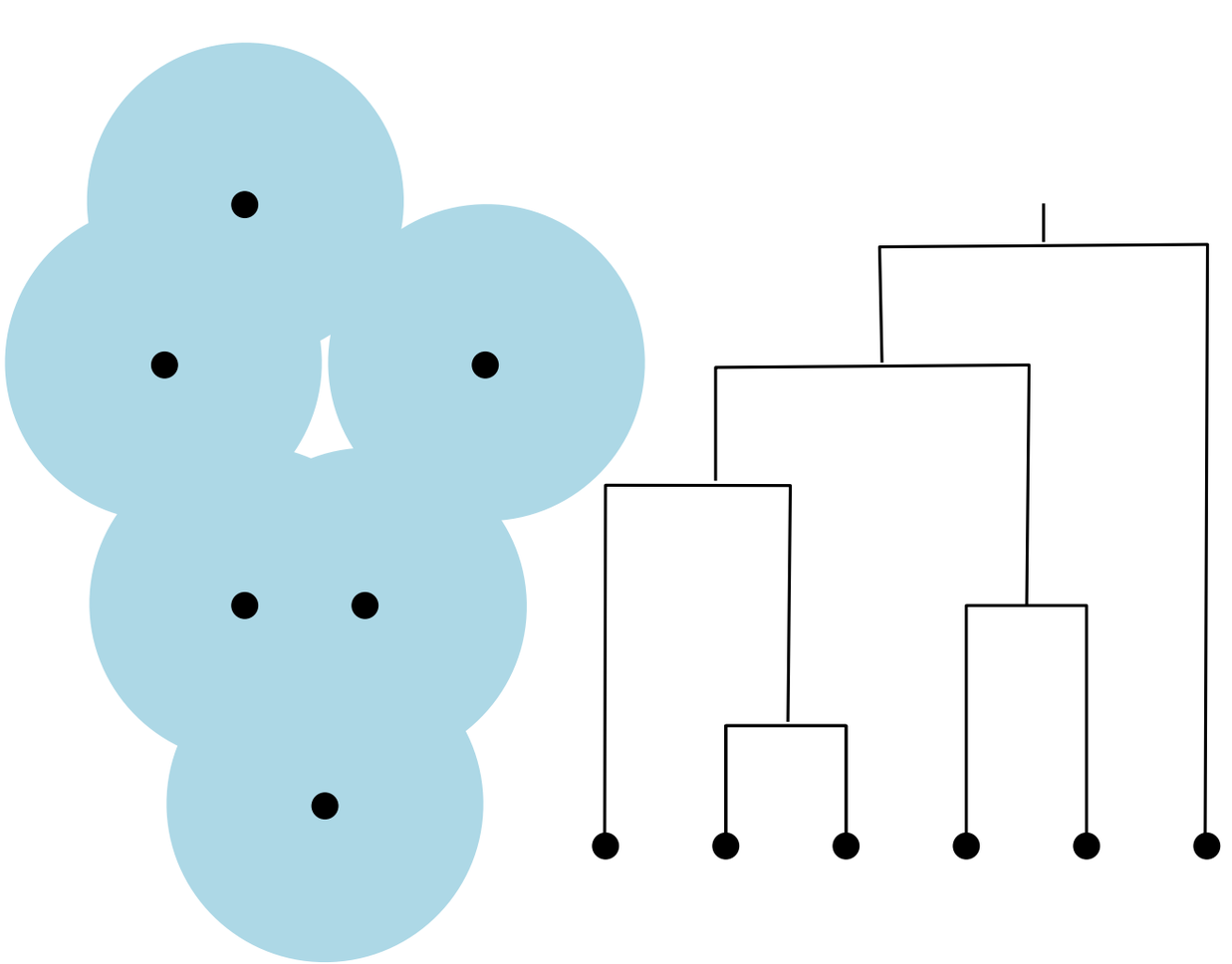


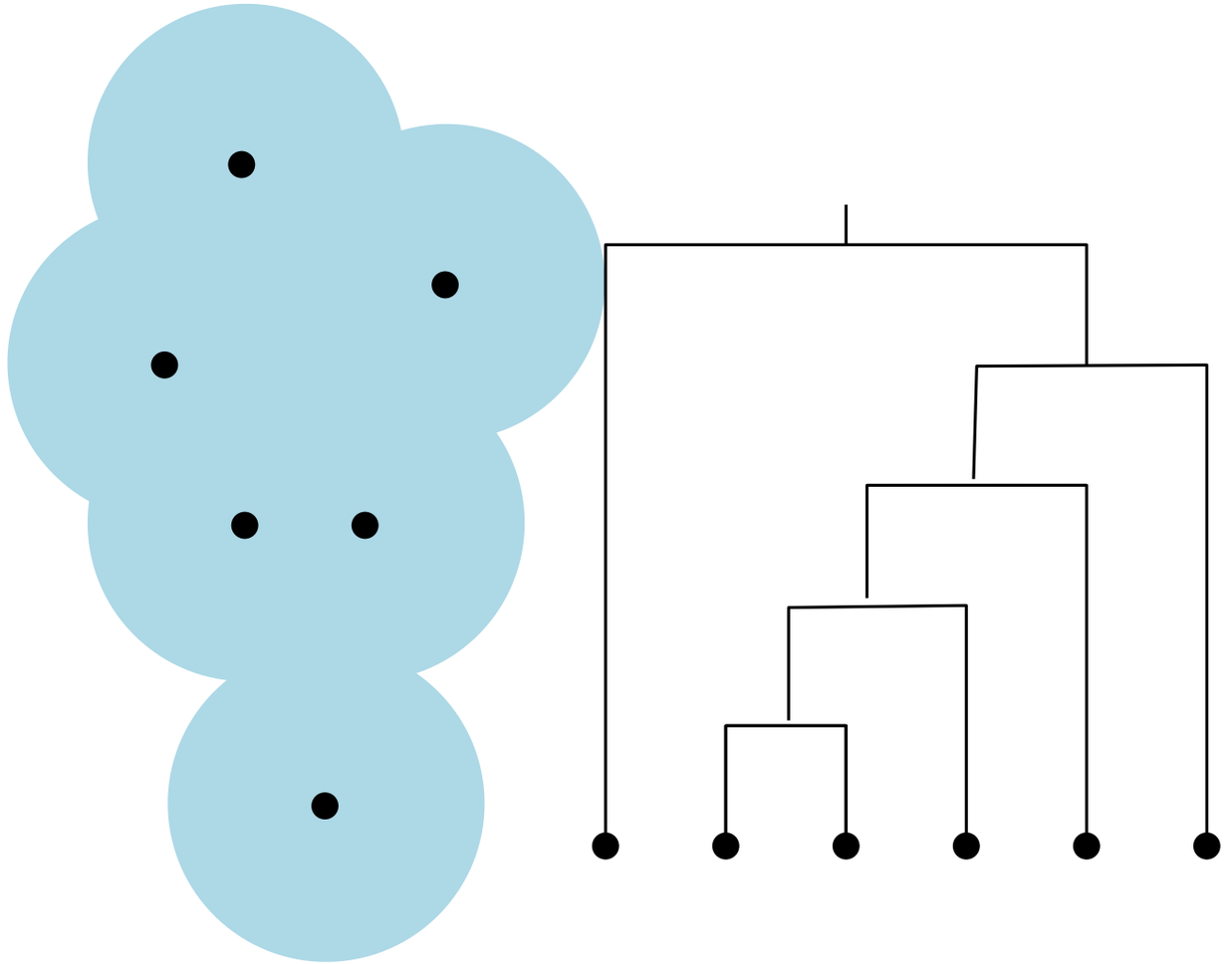
The unstability of dendograms



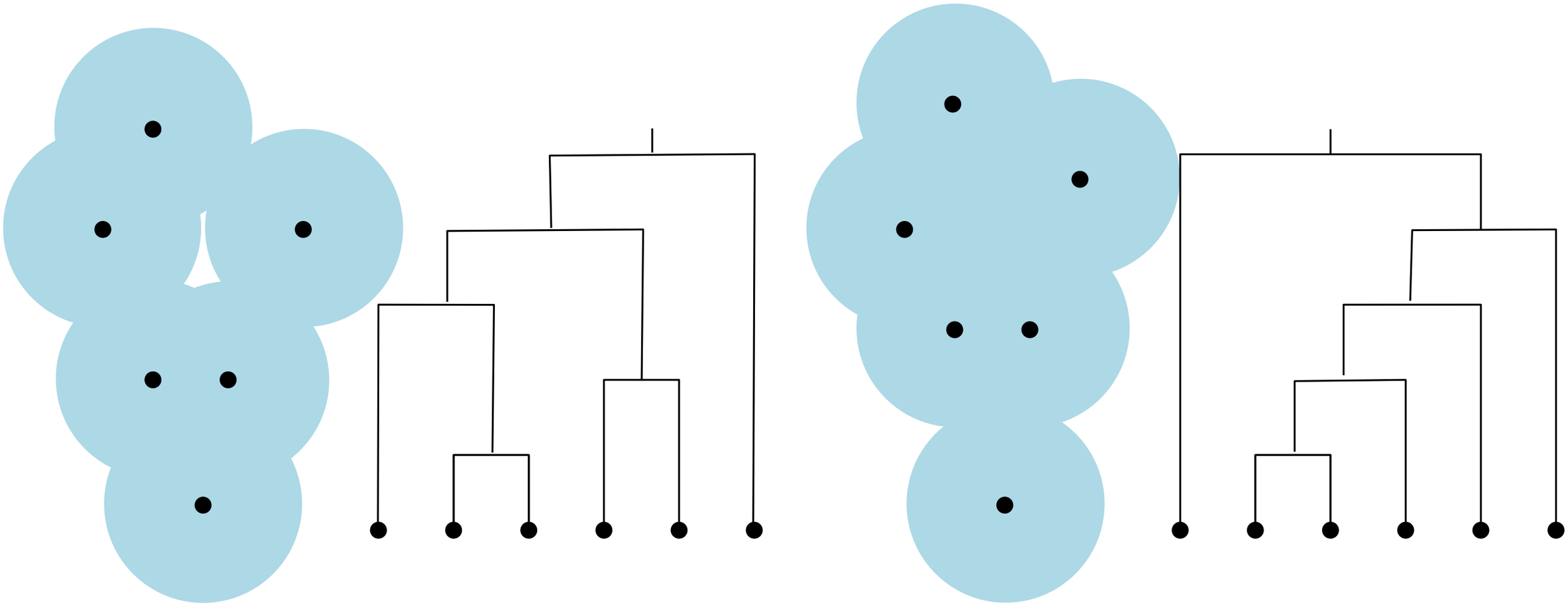


The unstability of dendograms



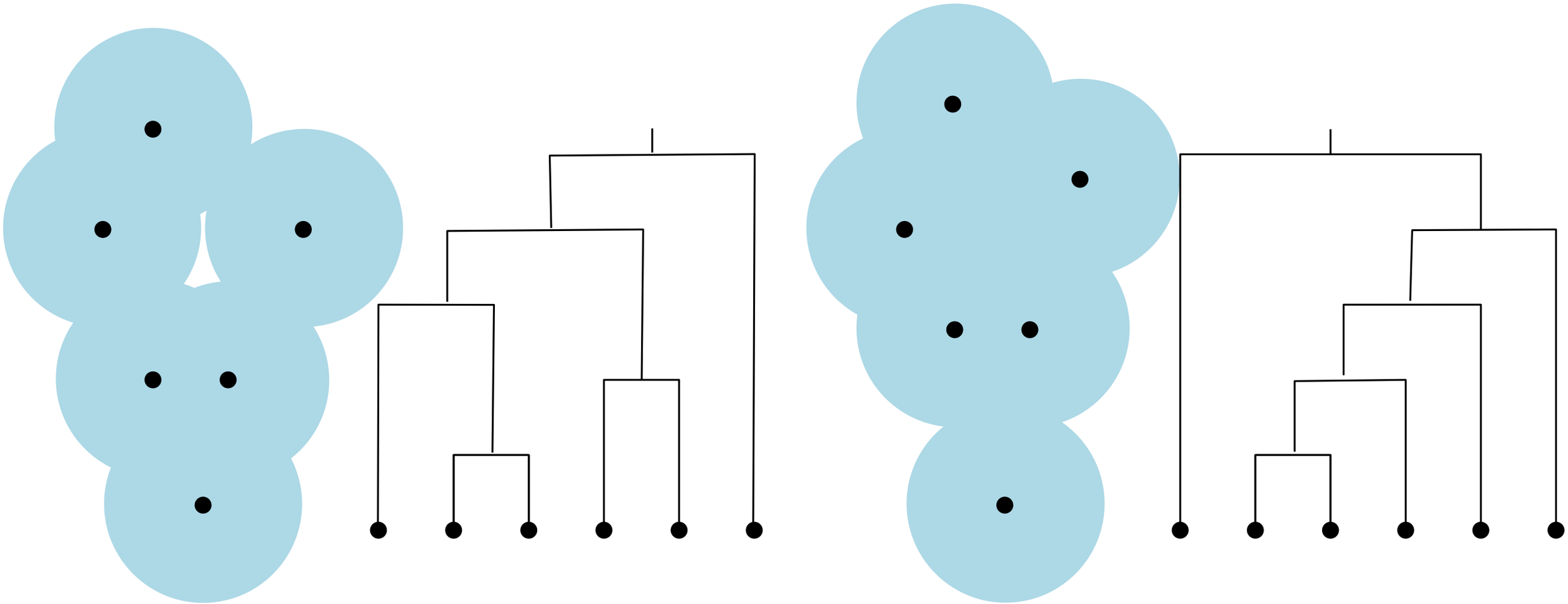


The unstability of dendograms



- Small perturbations on the input data may lead to wide change in the structure of the trees.
- However, the "merging times" remain stable.
- (For Euclidean data), the single linkage clustering keeps track of the evolution of the connected components of the distance function to the data.

The unstability of dendograms

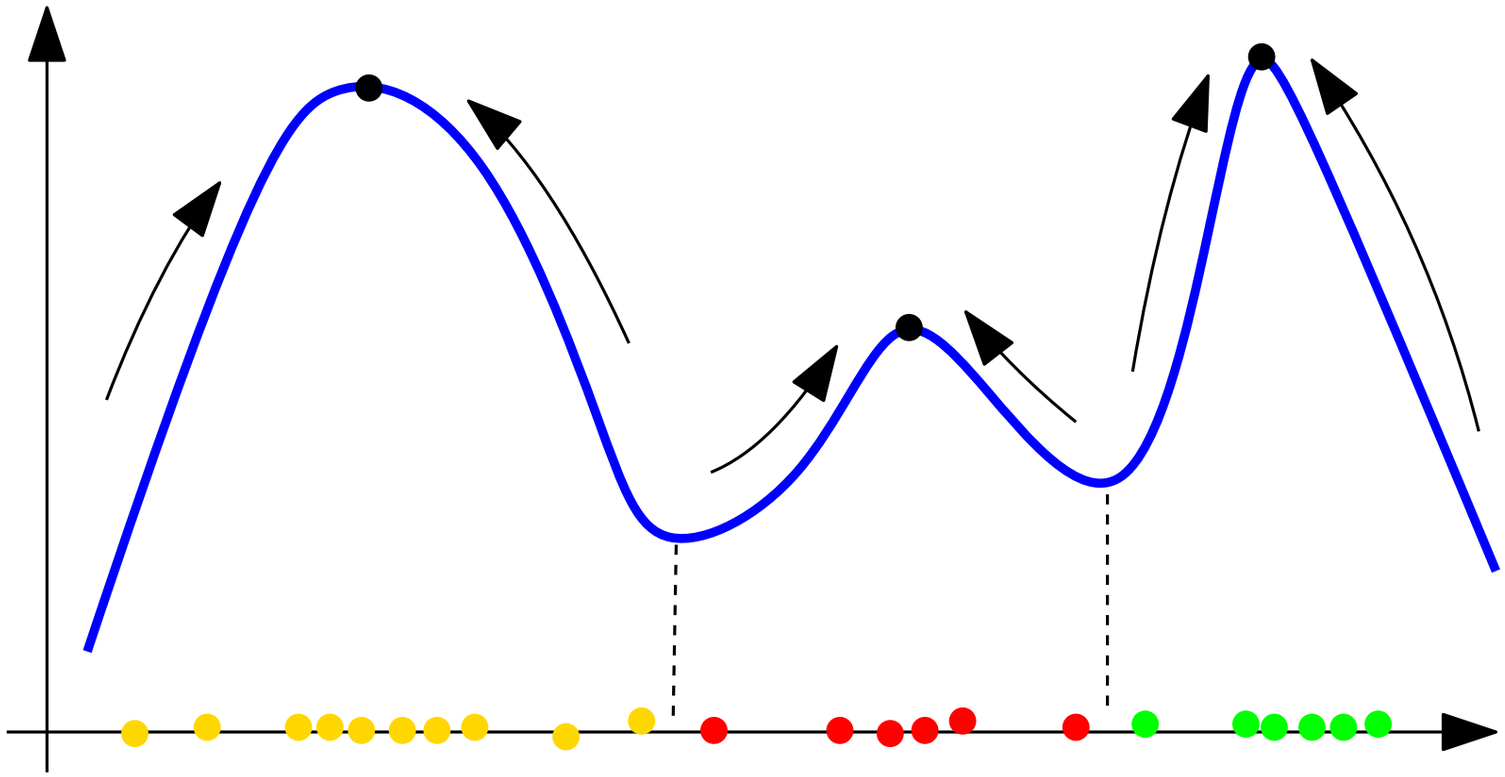


- Small perturbations on the input data may lead to wide change in the structure of the trees.

Persistent homology!

- However, the "merging times" remain stable.
- (For Euclidean data), the single linkage clustering keeps track of the evolution of the connected components of the distance function to the data.

Mode seeking clustering

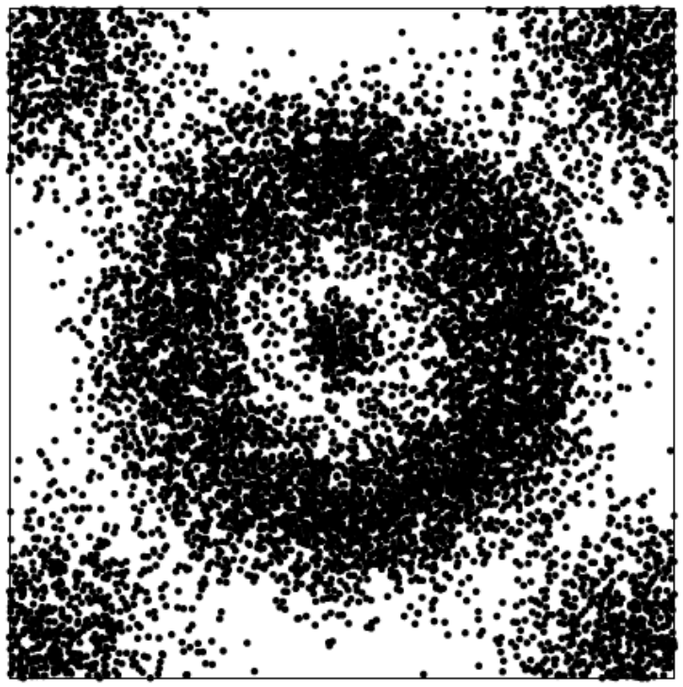


- Data points are sampled according to some (unknown) probability density.
- Clusters = the basins of attractions of the density

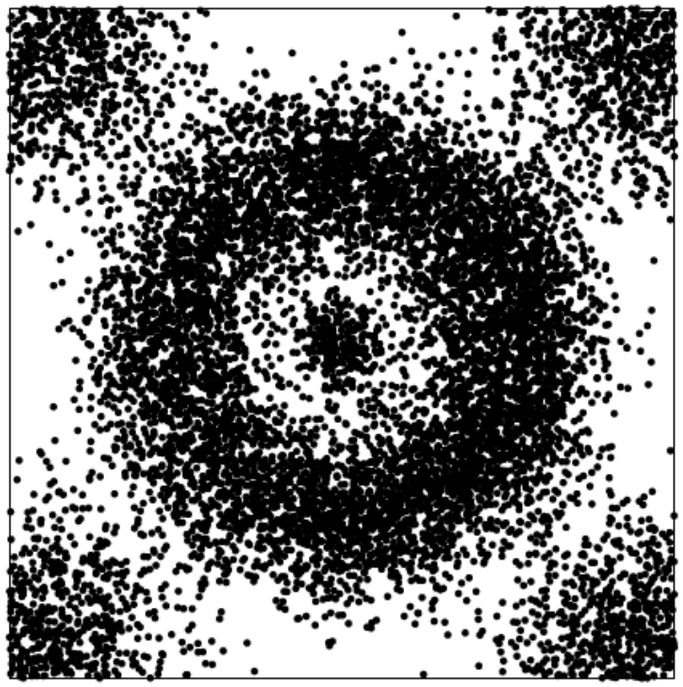
Two approaches:

- **Iterative**, such as, e.g., Mean Shift [Comaniciu et al, IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002].
- **Graph-based**, such as, e.g., [Koontz et al, IEEE Trans. on Computers 1976].

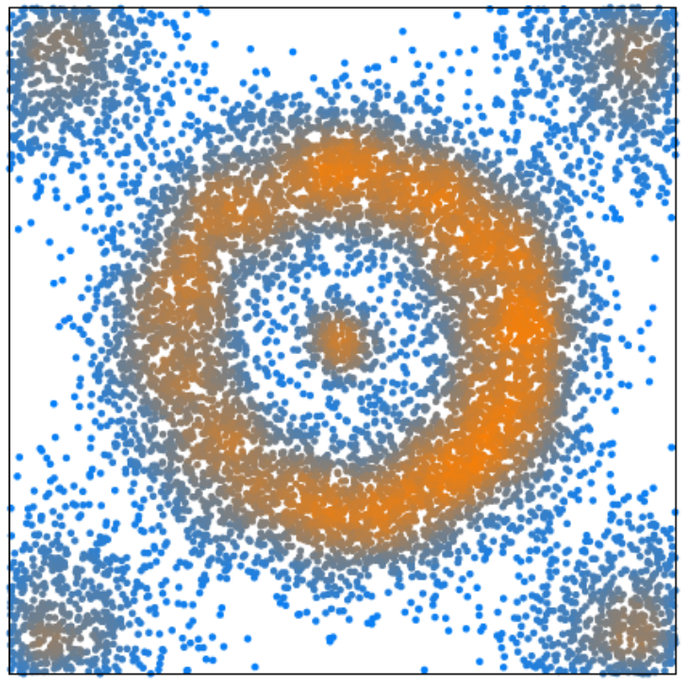
The Koonz, Narendra and Fukunaga algorithm (1976)



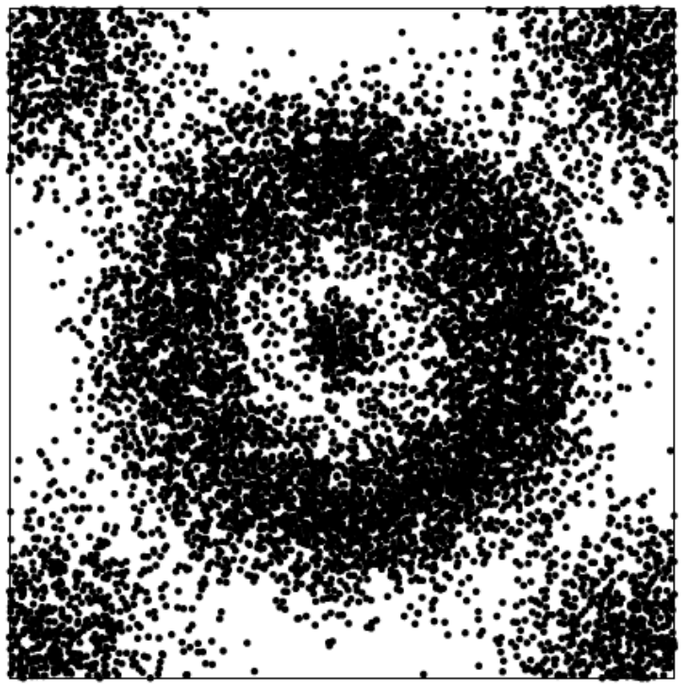
The Koonz, Narendra and Fukunaga algorithm (1976)



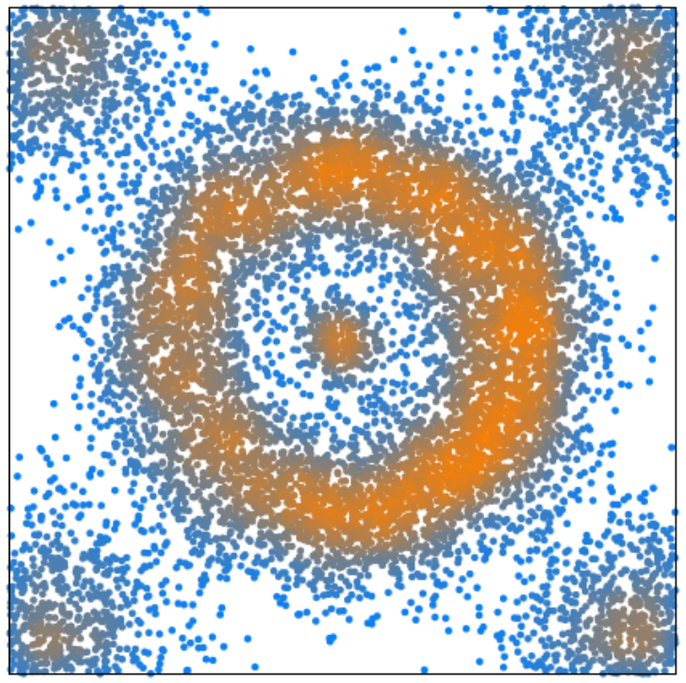
Density estimation



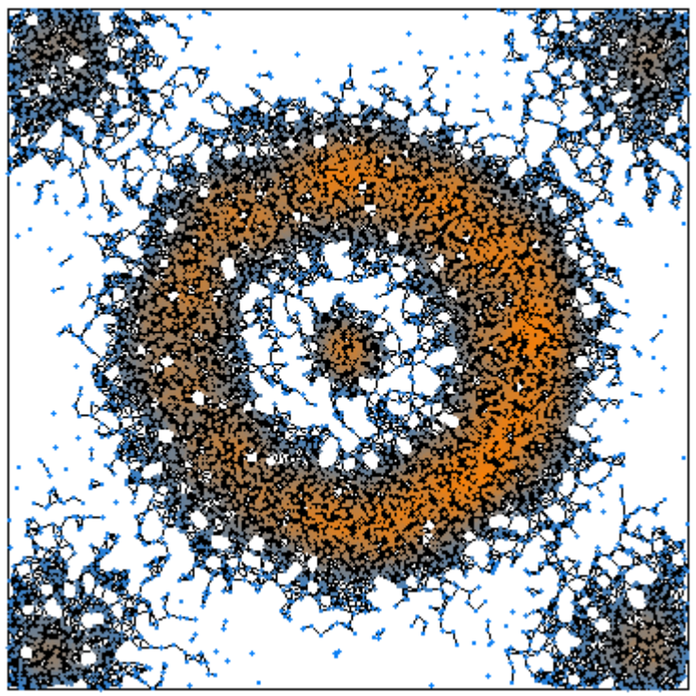
The Koonz, Narendra and Fukunaga algorithm (1976)



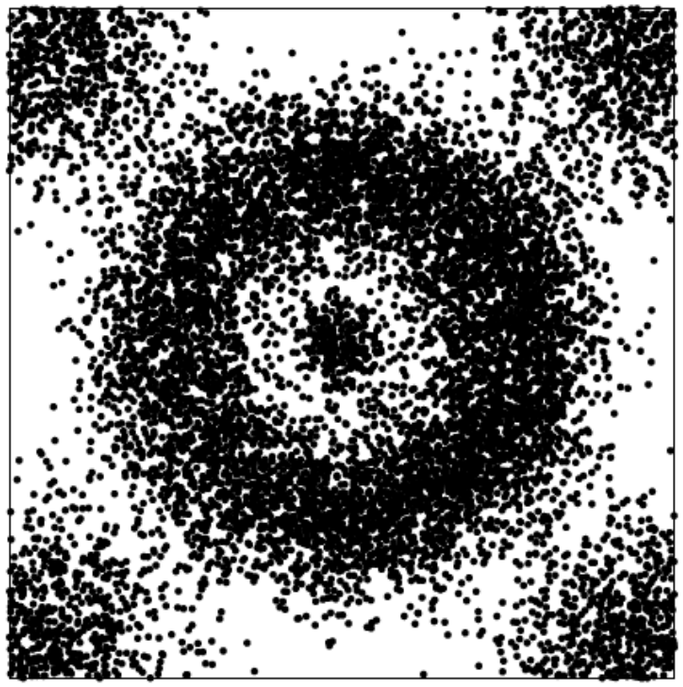
Density estimation



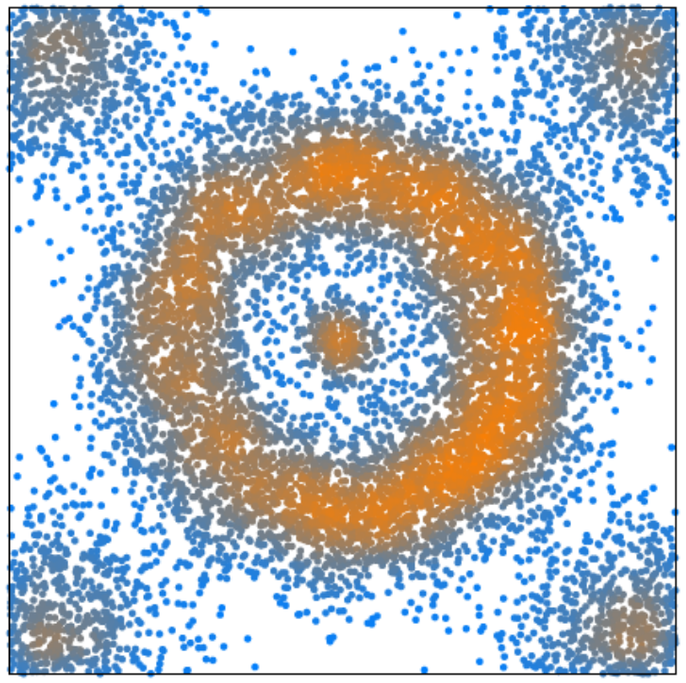
Neighborhood graph



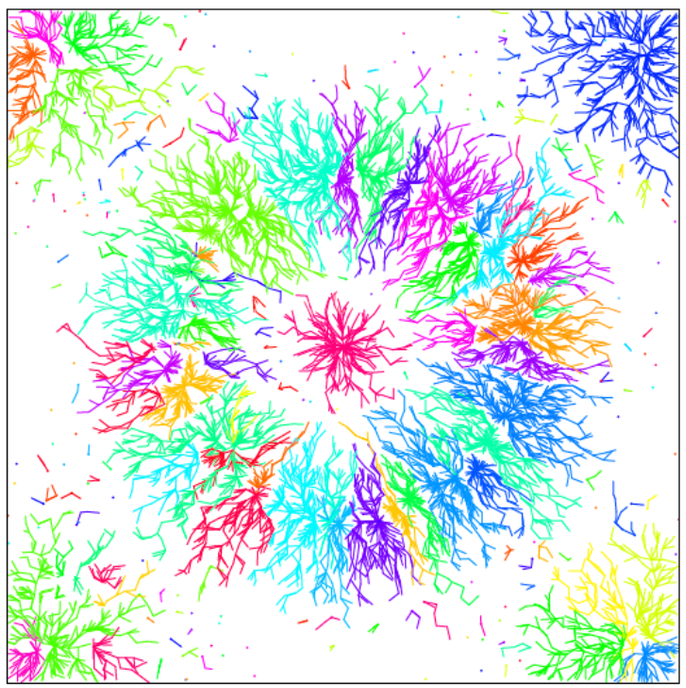
The Koonz, Narendra and Fukunaga algorithm (1976)



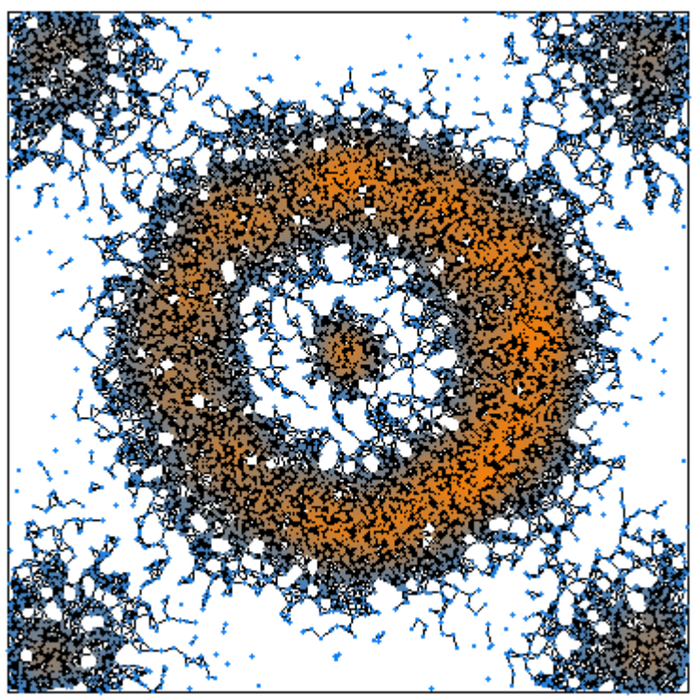
Density estimation



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

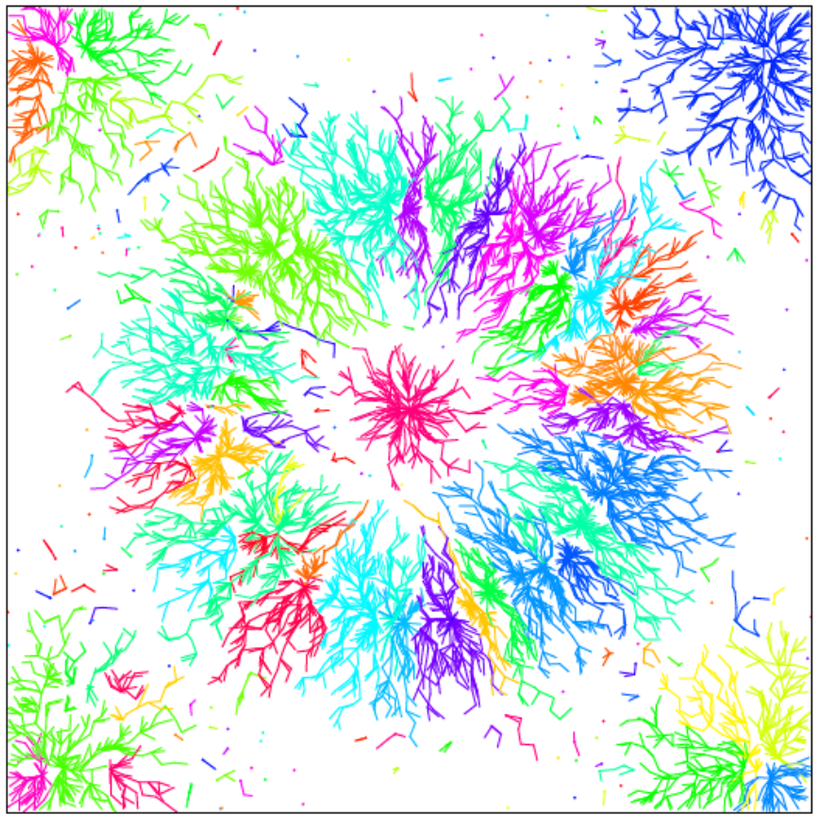
Sort the vertex indices $\{1, 2, \cdots, n\}$ in decreasing order: $\hat{f}(1) \geq \hat{f}(2) \geq \cdots \geq \hat{f}(n)$;
Initialize a union-find data structure (disjoint-set forest) $\mathcal{U}$ and two vectors g, r of size n;

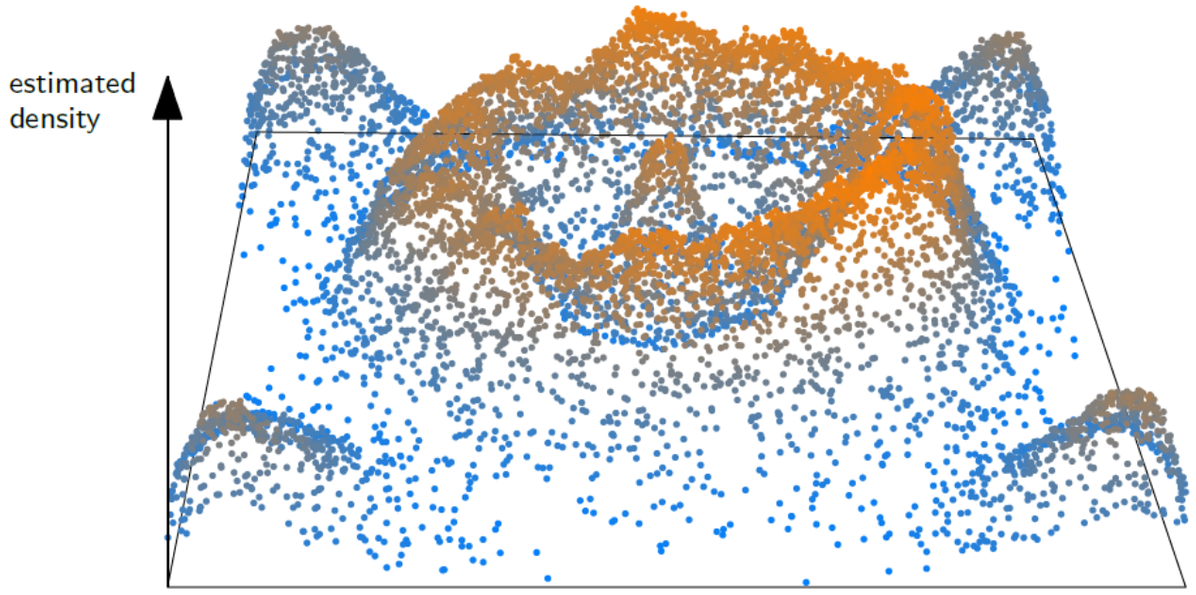
for $i = 1$ to n **do**
    Let N be the set of neighbors of i in G that have indices higher than i;
    if $N = \emptyset$
        Create a new entry e in $\mathcal{U}$ and attach vertex i to it;
        $r(e) \leftarrow i$ *// $r(e)$ stores the root vertex associated with the entry e*
    else
        $g(i) \leftarrow \text{argmax}_{j \in \mathcal{N}} \hat{f}(j)$ *// $g(i)$ stores the approximate gradient at vertex$i$*
        $e_i \leftarrow \mathcal{U}.\texttt{find}(g(i))$;
        Attach vertex i to the entry e_i;

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

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

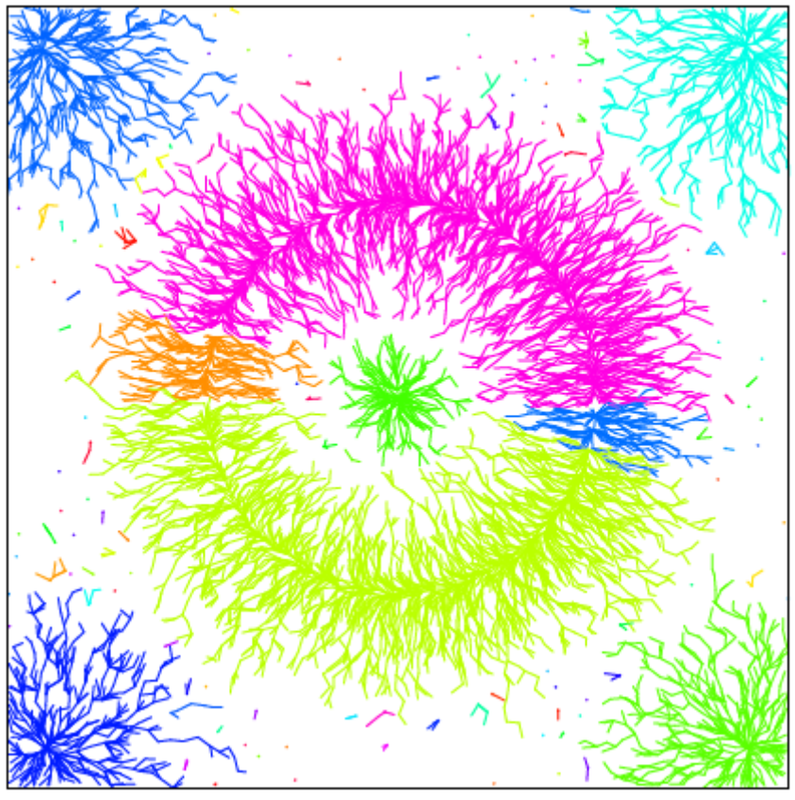


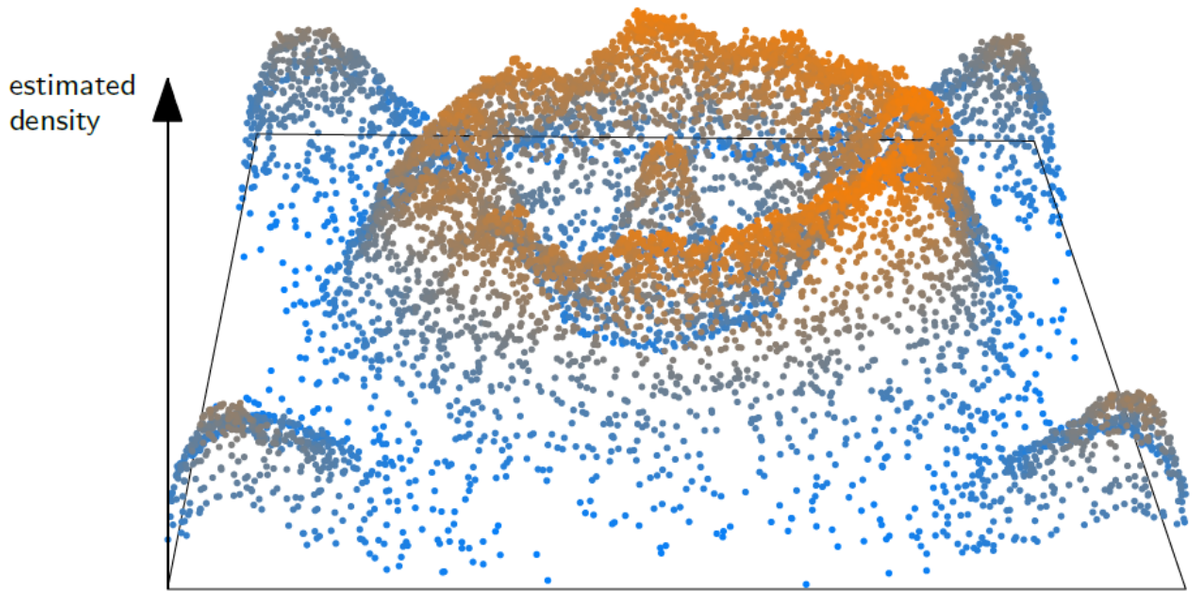


- As many clusters as local maxima of the density estimate $\rightarrow$ sensitivity to noise!

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:





- As many clusters as local maxima of the density estimate $\rightarrow$ sensitivity to noise!
- The choice of the neighborhood graph may result in wide changes in the output.

The Koonz, Narendra and Fukunaga algorithm (1976)

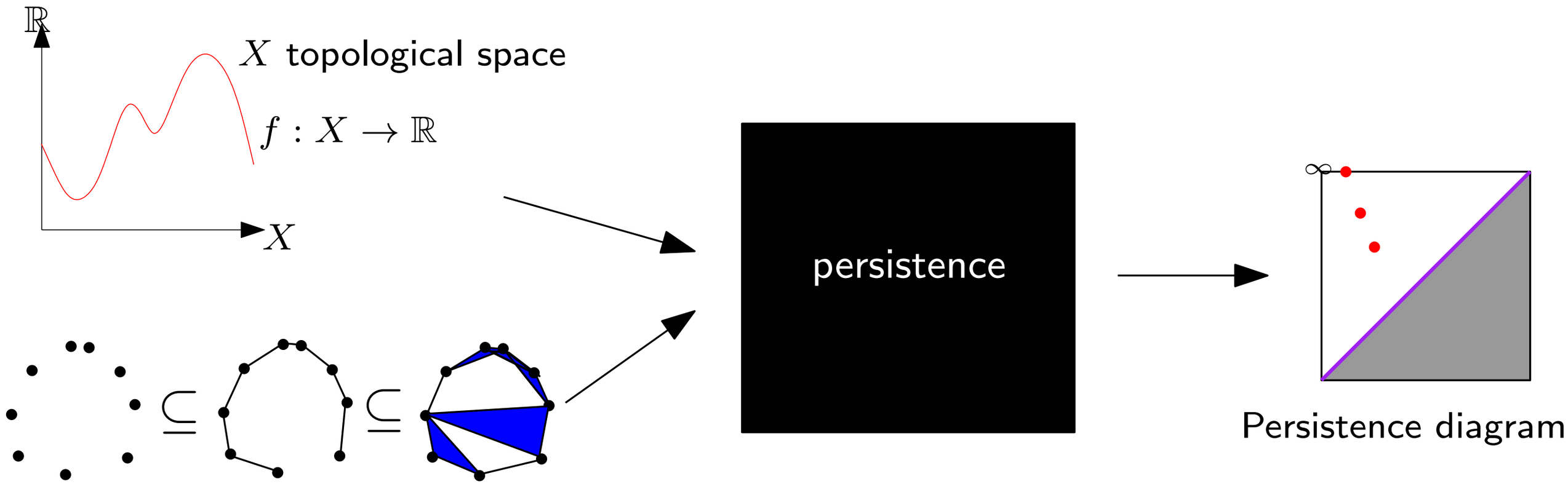
Drawbacks:

- As many clusters as local maxima of the density estimate $\rightarrow$ sensitivity to noise!
- The choice of the neighborhood graph may result in wide changes in the output.

Approaches to overcome these issues:

- Smooth out the density estimate (e.g. mean-shift)... But pb of choice of a smoothing param.
- Merge clusters: various way to do that. In the following: use persistent homology!

Persistent homology

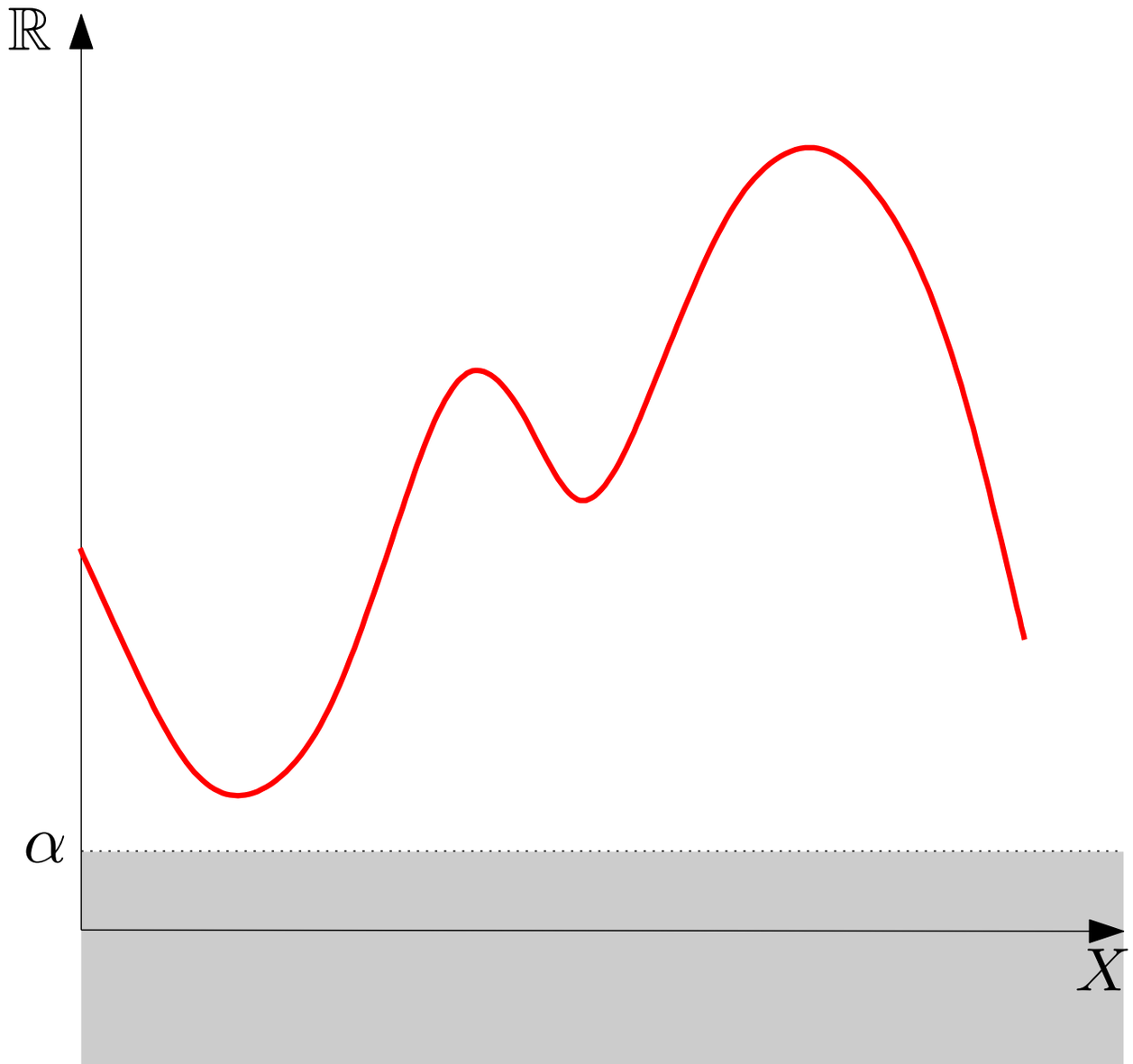


Nested spaces

- A general mathematical framework to encode the evolution of the topology (homology) of families of nested spaces (filtered complex, sublevel sets,...).
- Formalized by H. Edelsbrunner (2002) et al and G. Carlsson et al (2005) - wide development during the last decade.
- Multiscale topological information.
- Barcodes/persistence diagrams can be efficiently computed.
- Stability properties

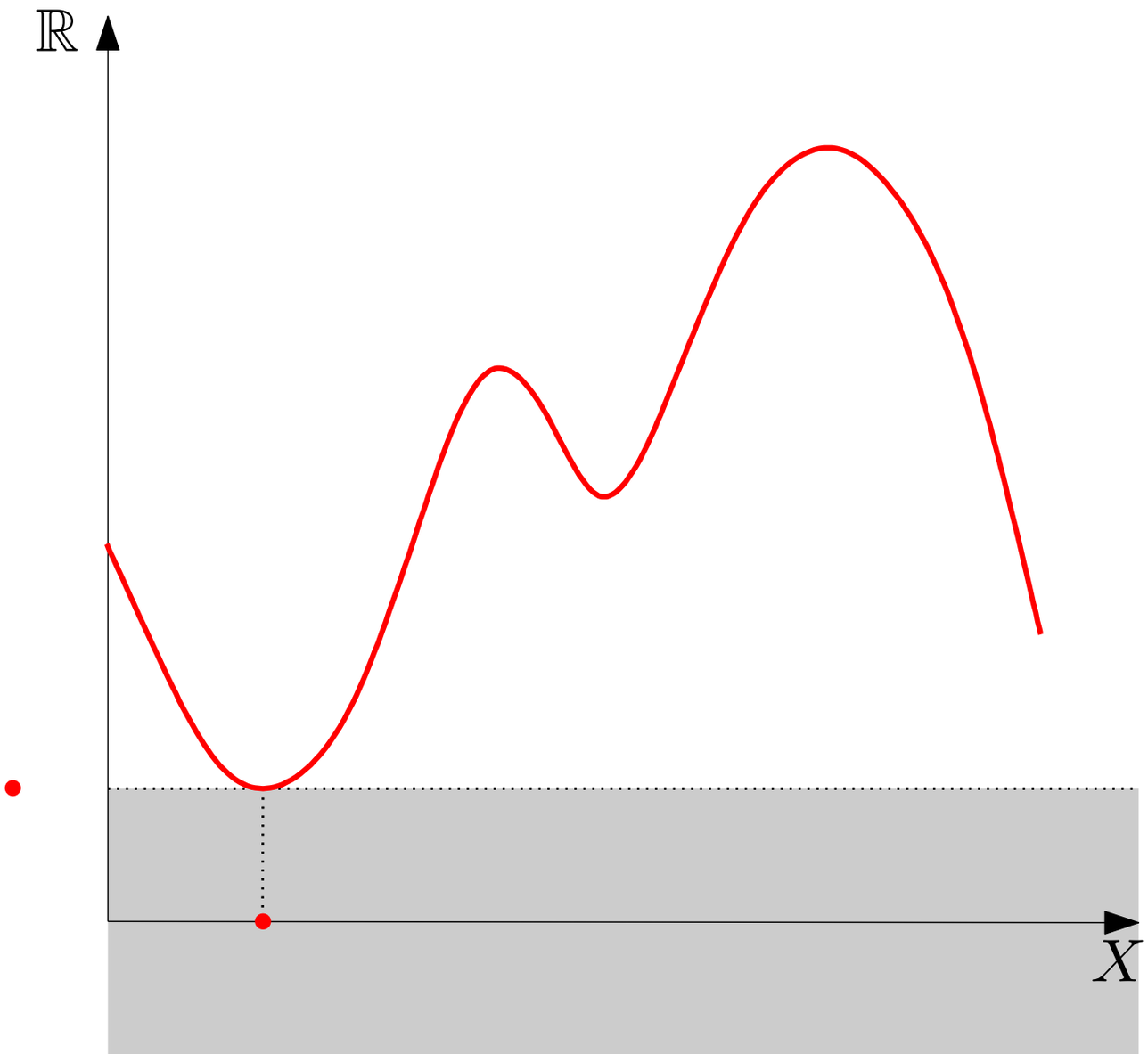
(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.



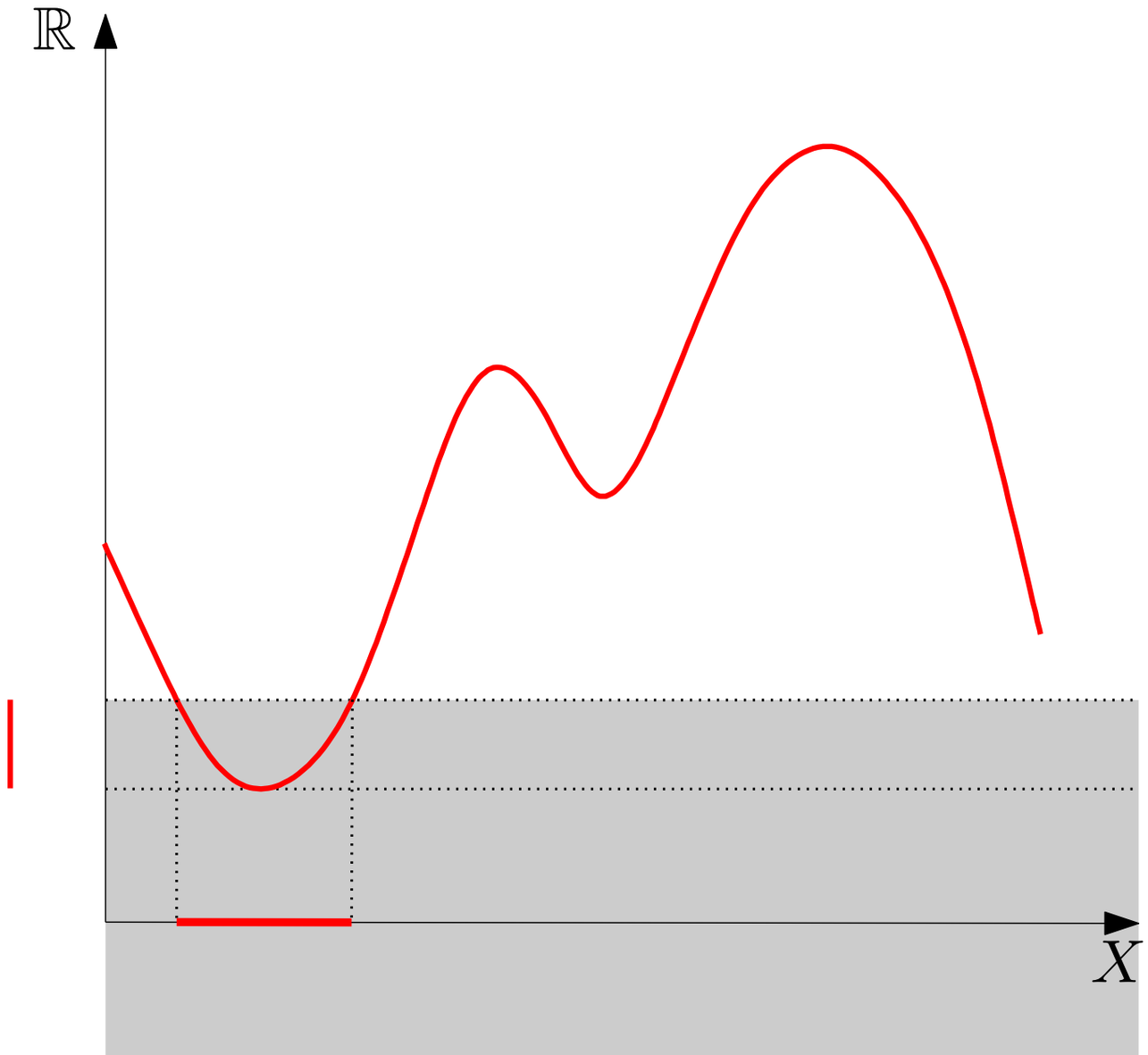
(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.



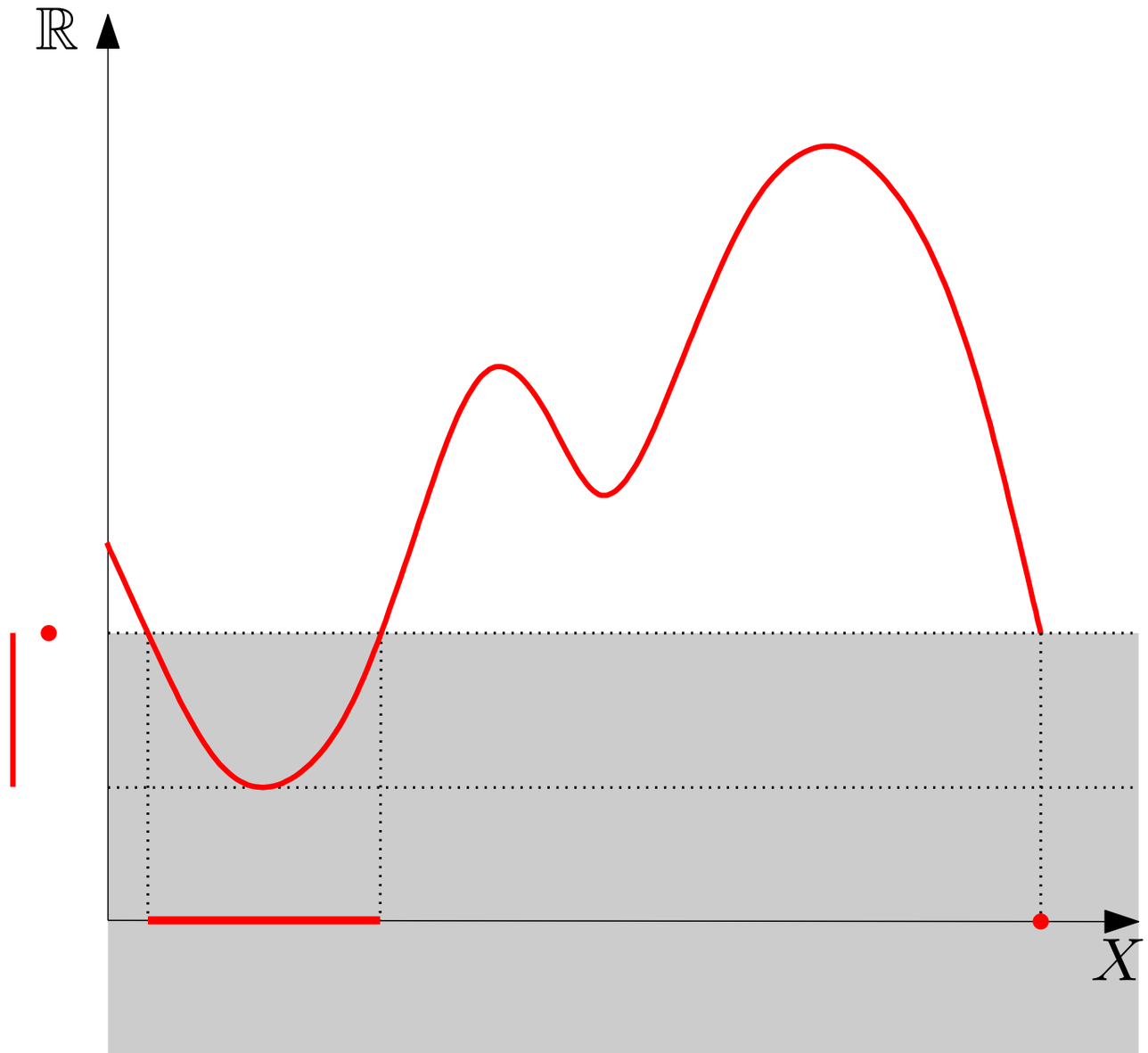
(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.



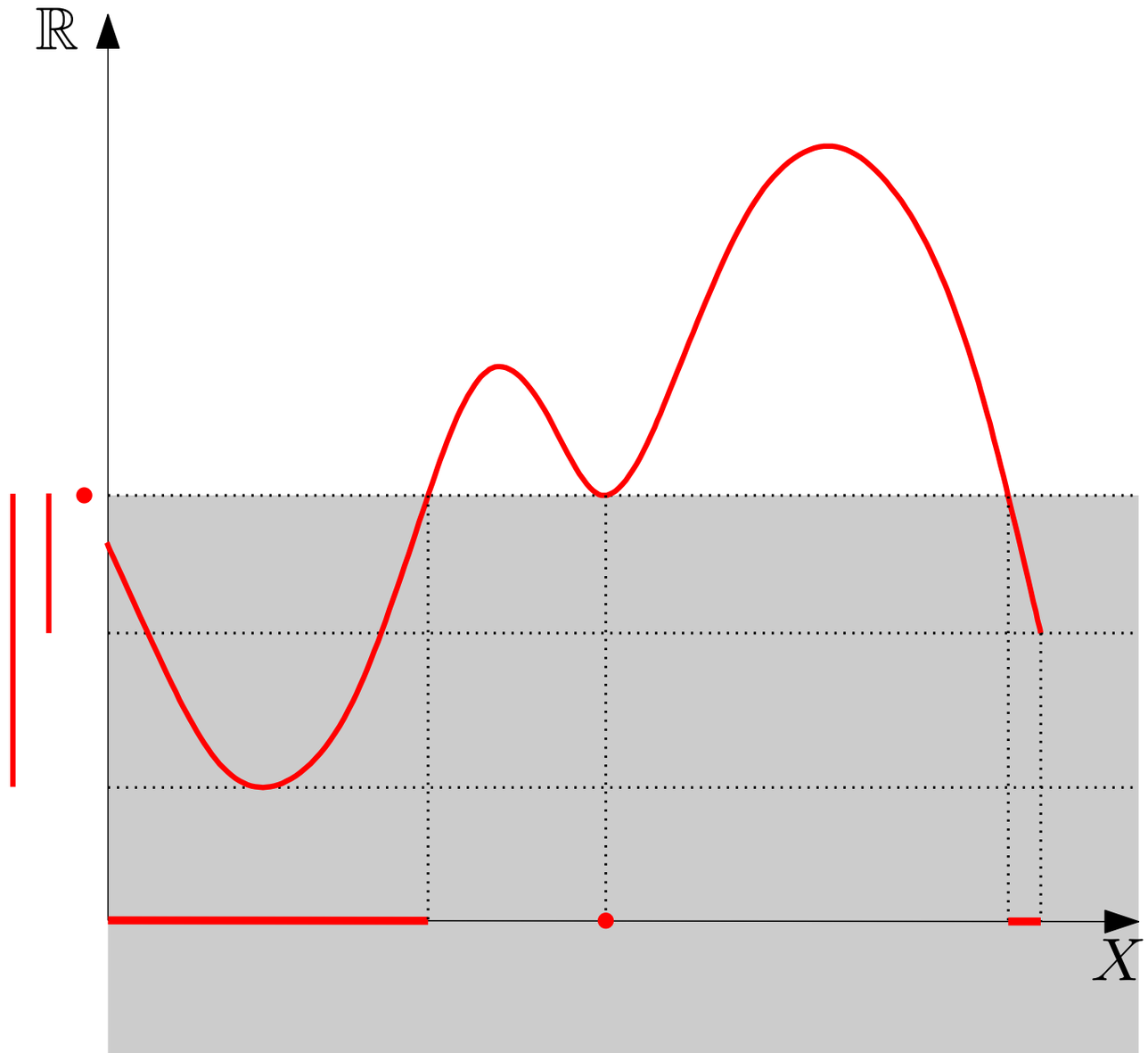
(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.



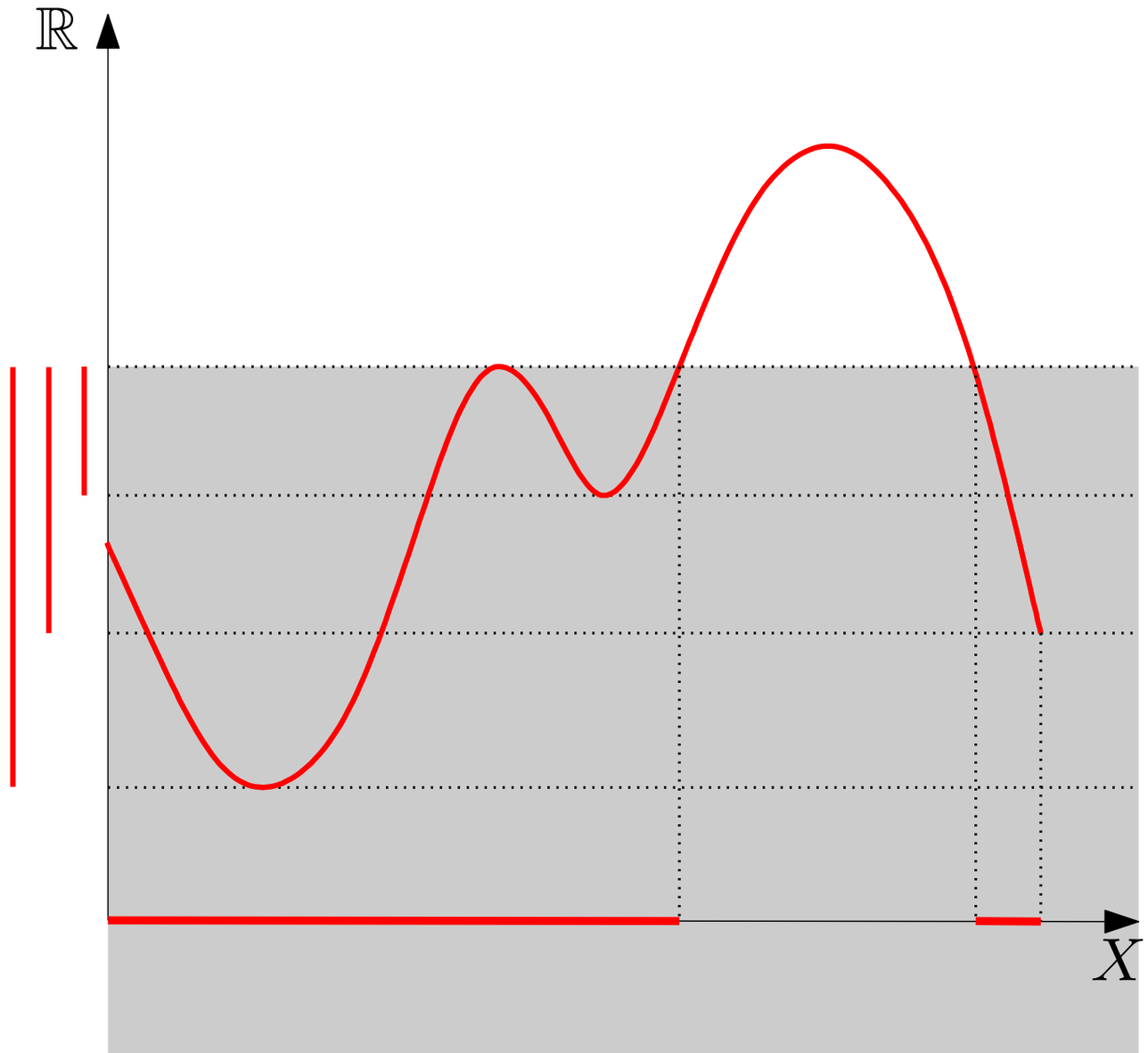
(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.



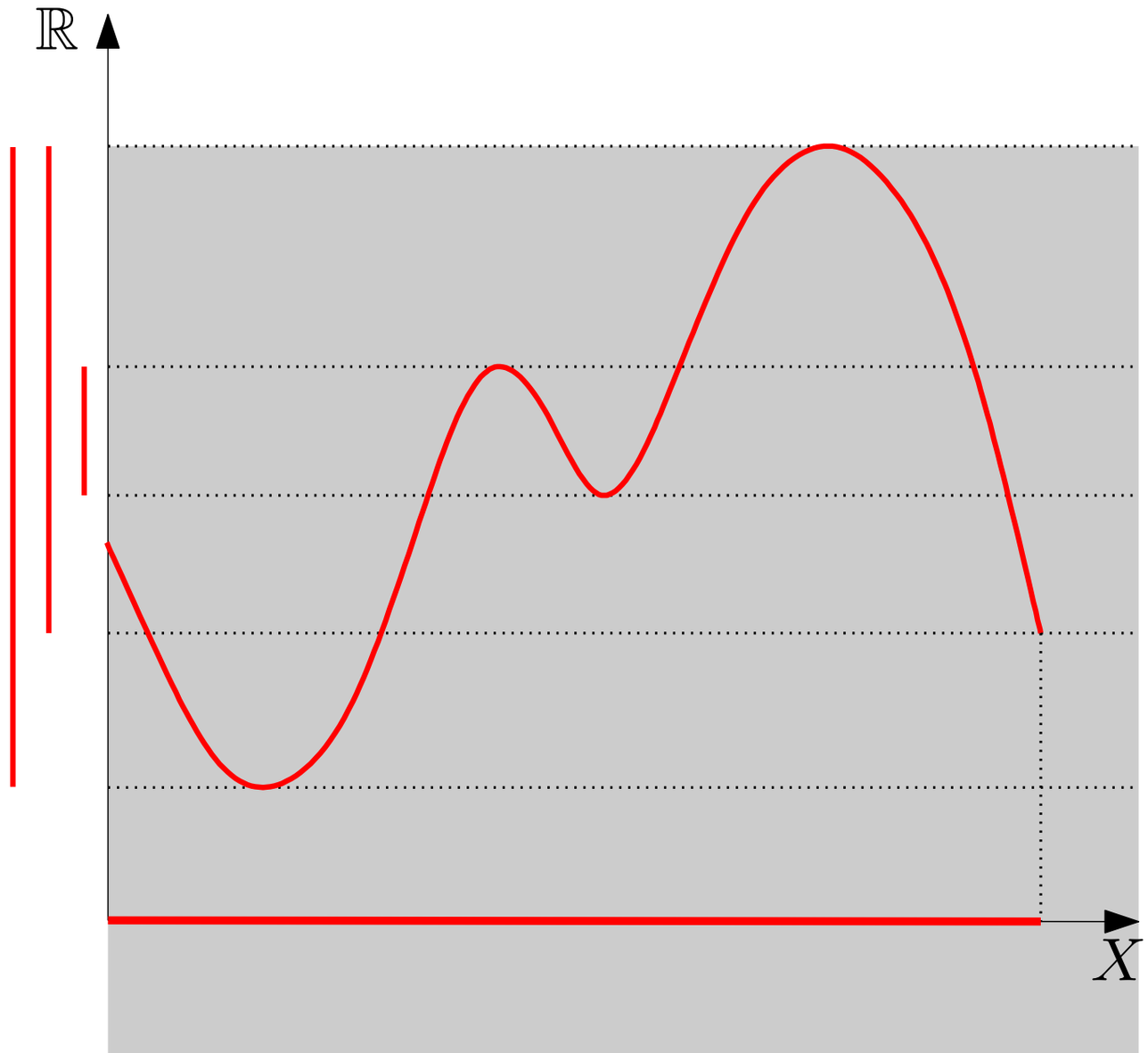
(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.



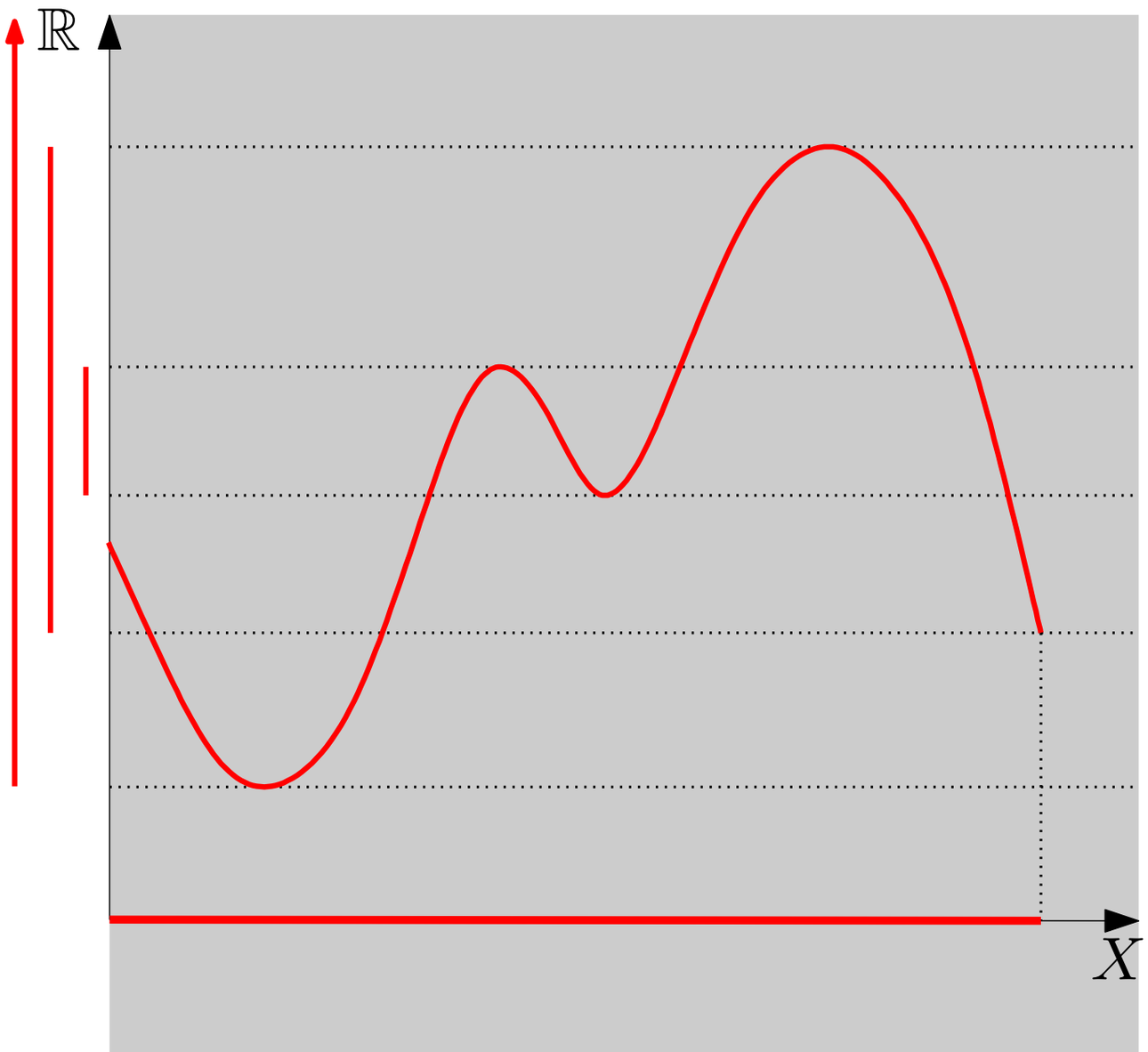
(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.



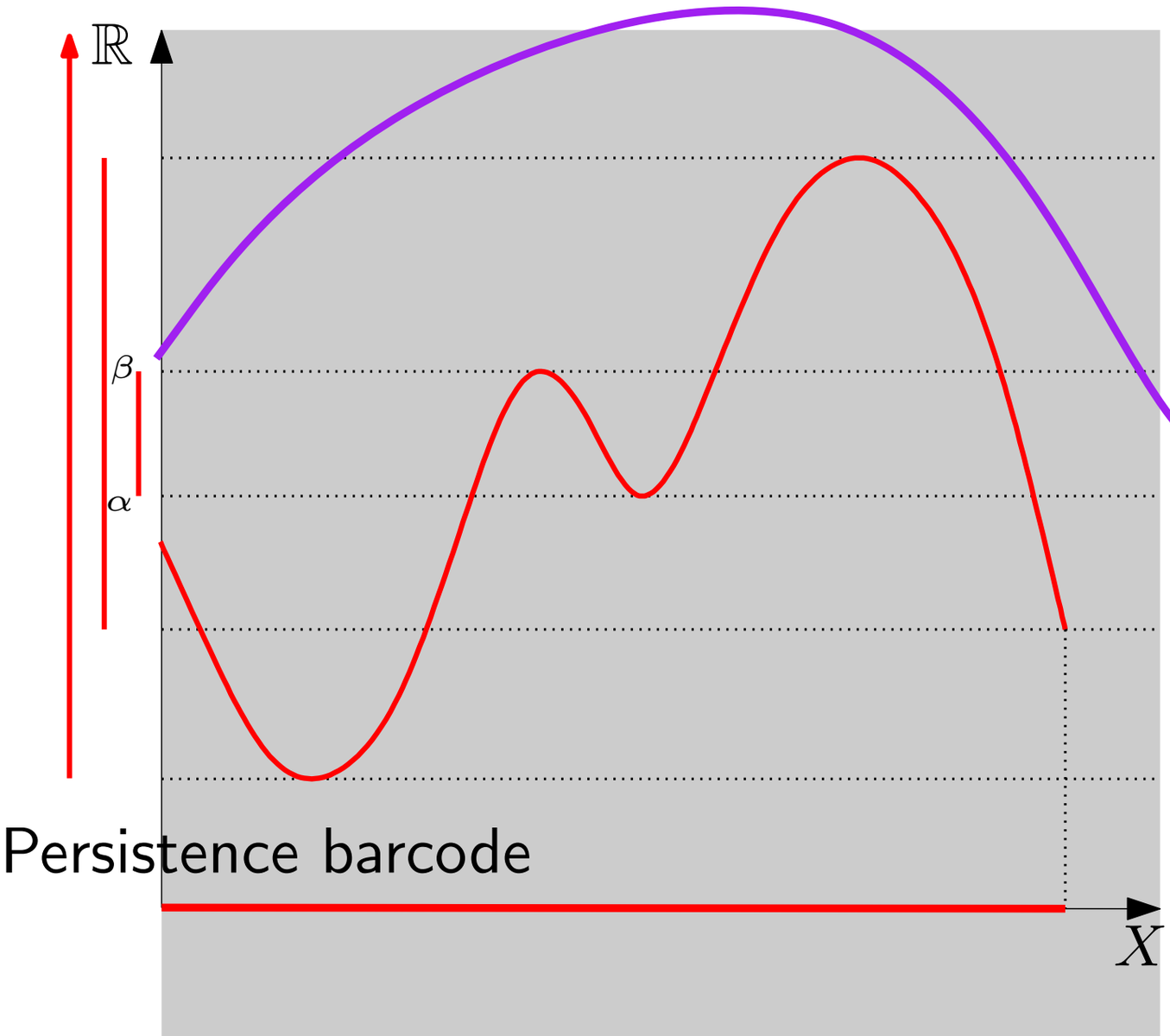
(0-dimensional) persistent homology for functions

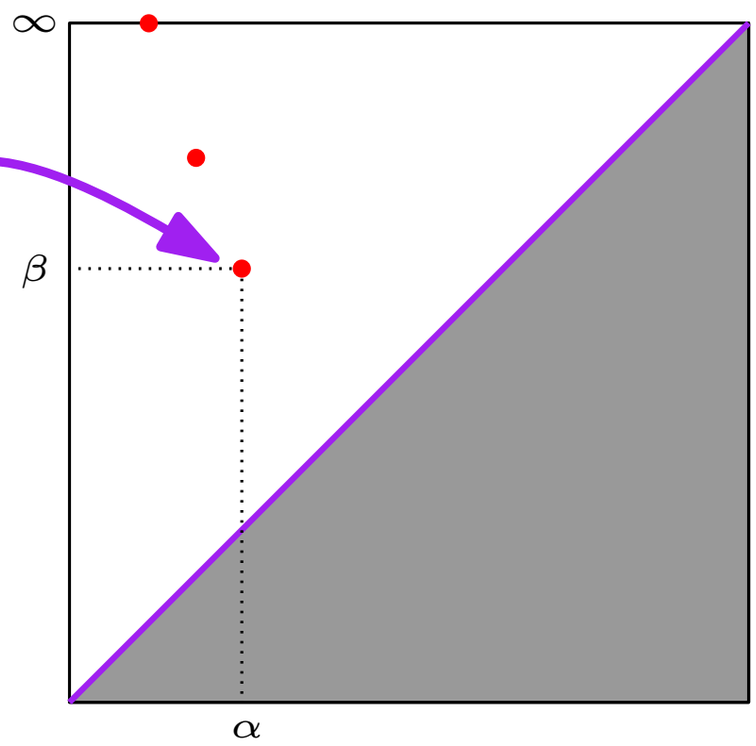
- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.
- Finite set of intervals (barcode) encodes births/deaths of topological features.



(0-dimensional) persistent homology for functions

- Nested family (filtration) of sublevel-sets $f^{-1}((-\infty, \alpha])$ for $\alpha = -\infty$ to $+\infty$.
- Track evolution of connectedness throughout the family.
- Finite set of intervals (barcode) encodes births/deaths of topological features.



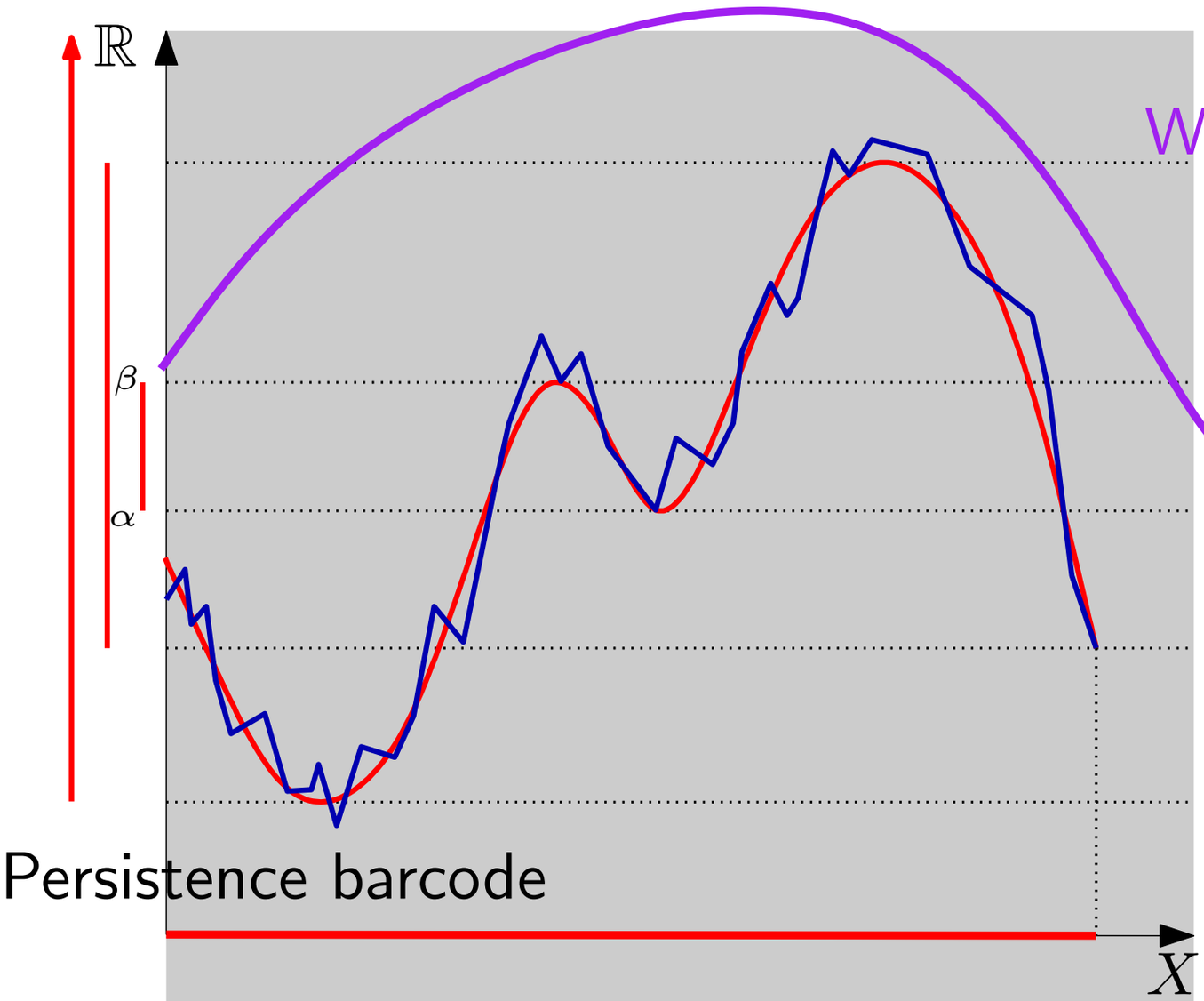


Persistence diagram

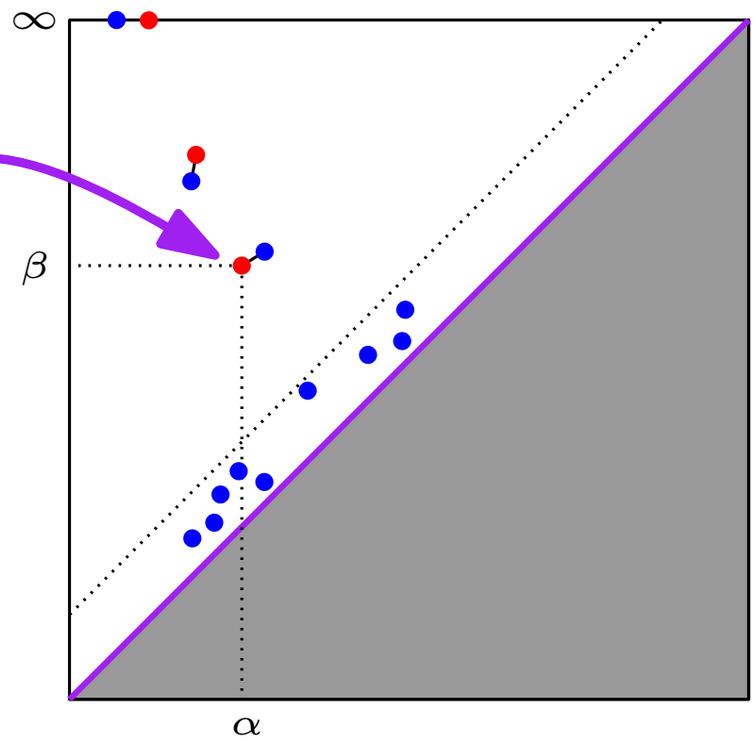
(0-dimensional) persistent homology for functions

Theorem (Stability):
For any *tame* functions $f, g : \mathbb{X} \to \mathbb{R}$, $\mathrm{d}(\mathrm{D}_f, \mathrm{D}_g) \leq \|f - g\|_\infty$.

[Cohen-Steiner, Edelsbrunner, Harer 05], [C., Cohen-Steiner, Glisse, Guibas, Oudot 09], [C., de Silva, Glisse, Oudot 12]

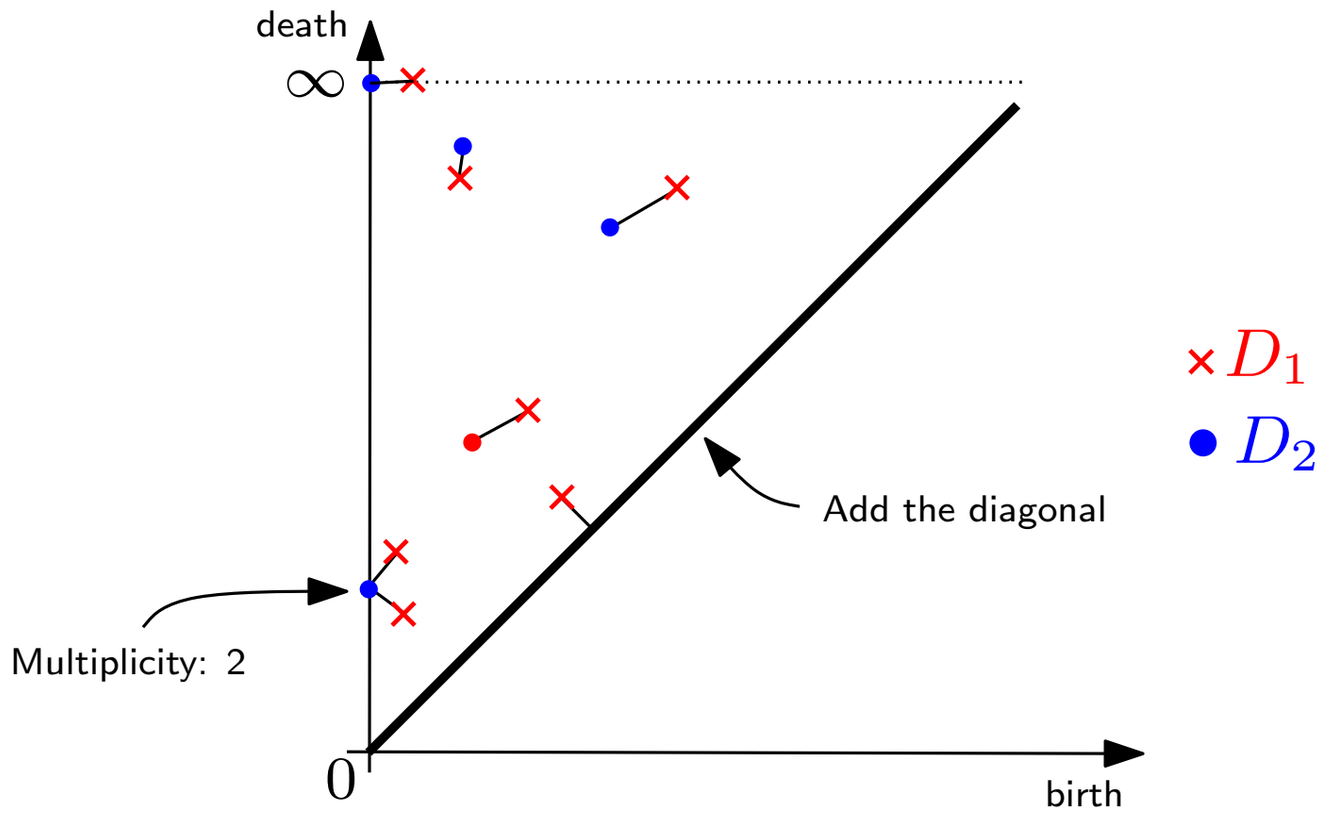


What if f is slightly perturbed?



Persistence diagram

Distance between persistence diagrams



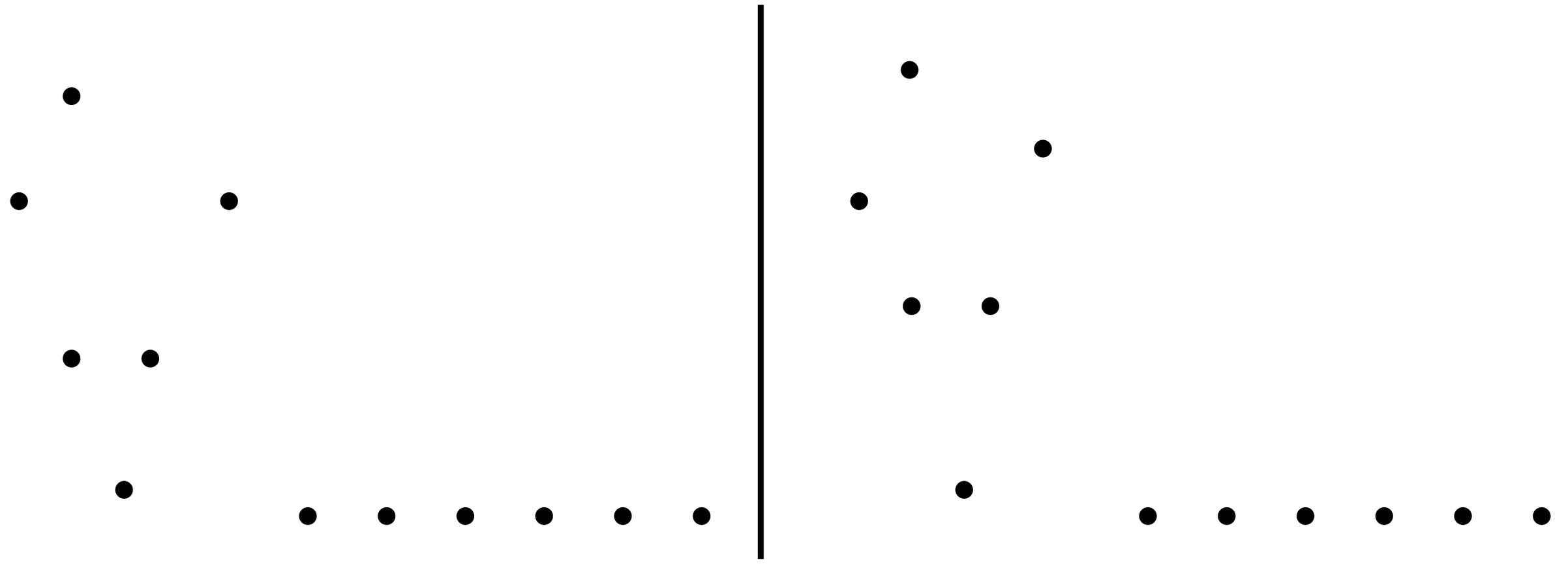
The bottleneck distance between two diagrams D_1 and D_2 is

$$d_B(D_1, D_2) = \inf_{\gamma \in \Gamma} \sup_{p \in D_1} \|p - \gamma(p)\|_\infty$$

where Γ is the set of all the bijections between D_1 and D_2 and $\|p - q\|_\infty = \max(|x_p - x_q|, |y_p - y_q|)$.

The example of distance functions

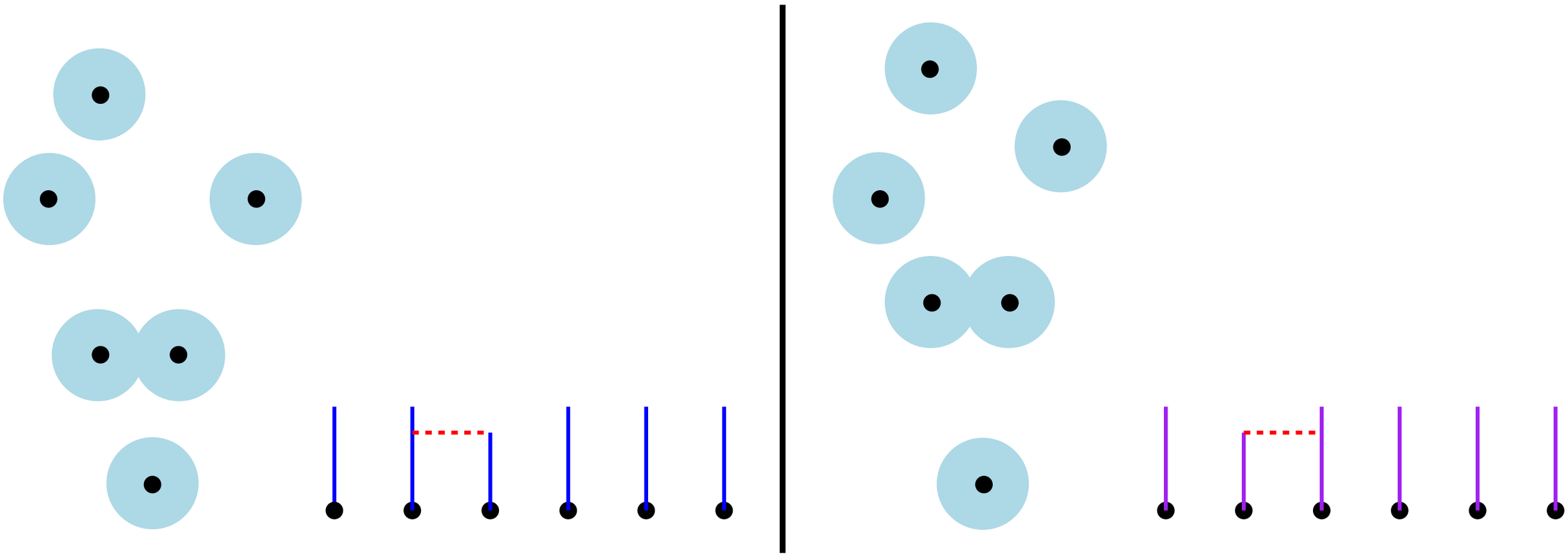
The example of distance functions



$$f_P : \quad \mathbb{R}^2 \to \mathbb{R}$$
$$x \mapsto \min_{p \in P} \|x - p\|_2$$

Link with single linkage clustering

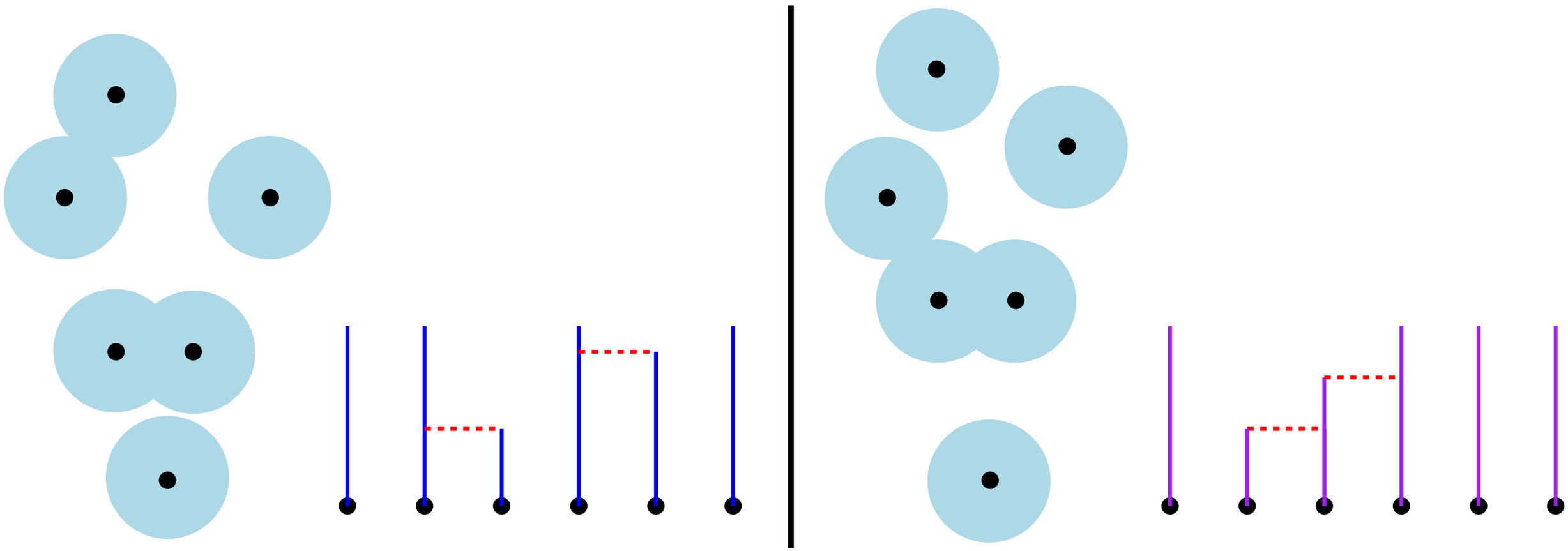
The example of distance functions



$$f_P : \quad \mathbb{R}^2 \to \mathbb{R}$$
$$x \mapsto \min_{p \in P} \|x - p\|_2$$

Link with single linkage clustering

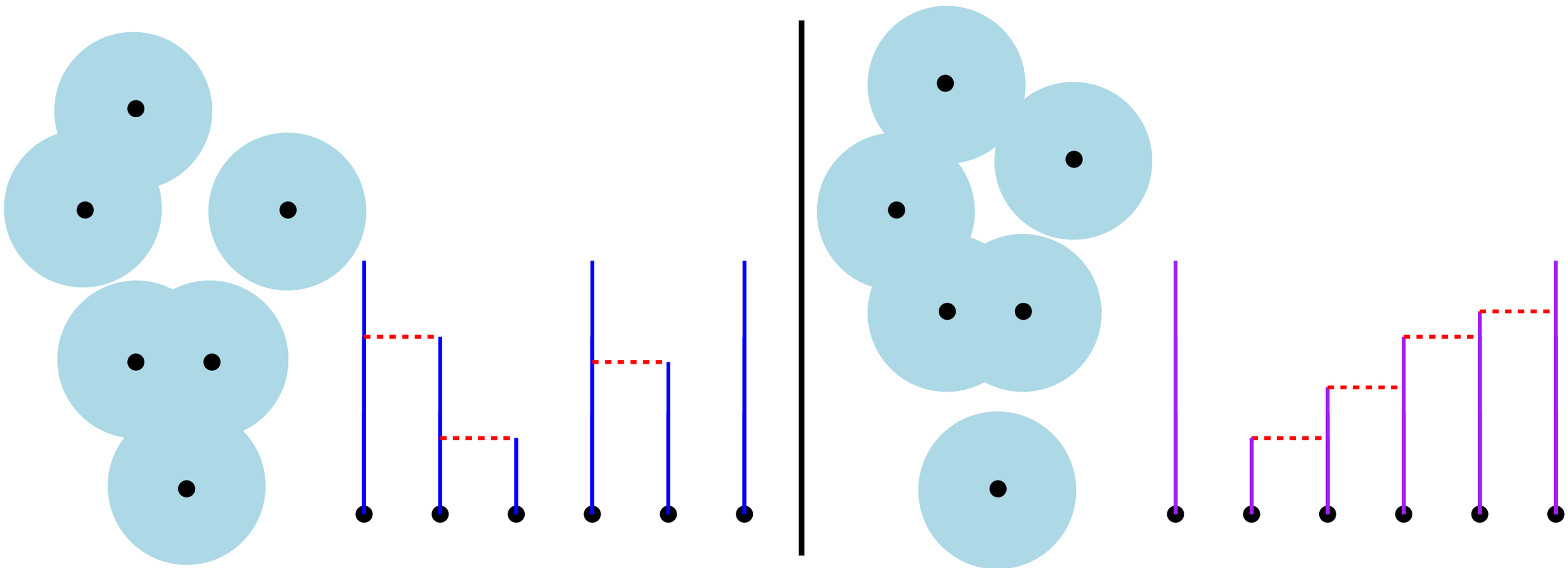
The example of distance functions



$$f_P : \quad \mathbb{R}^2 \to \mathbb{R}$$
$$x \mapsto \min_{p \in P} \|x - p\|_2$$

Link with single linkage clustering

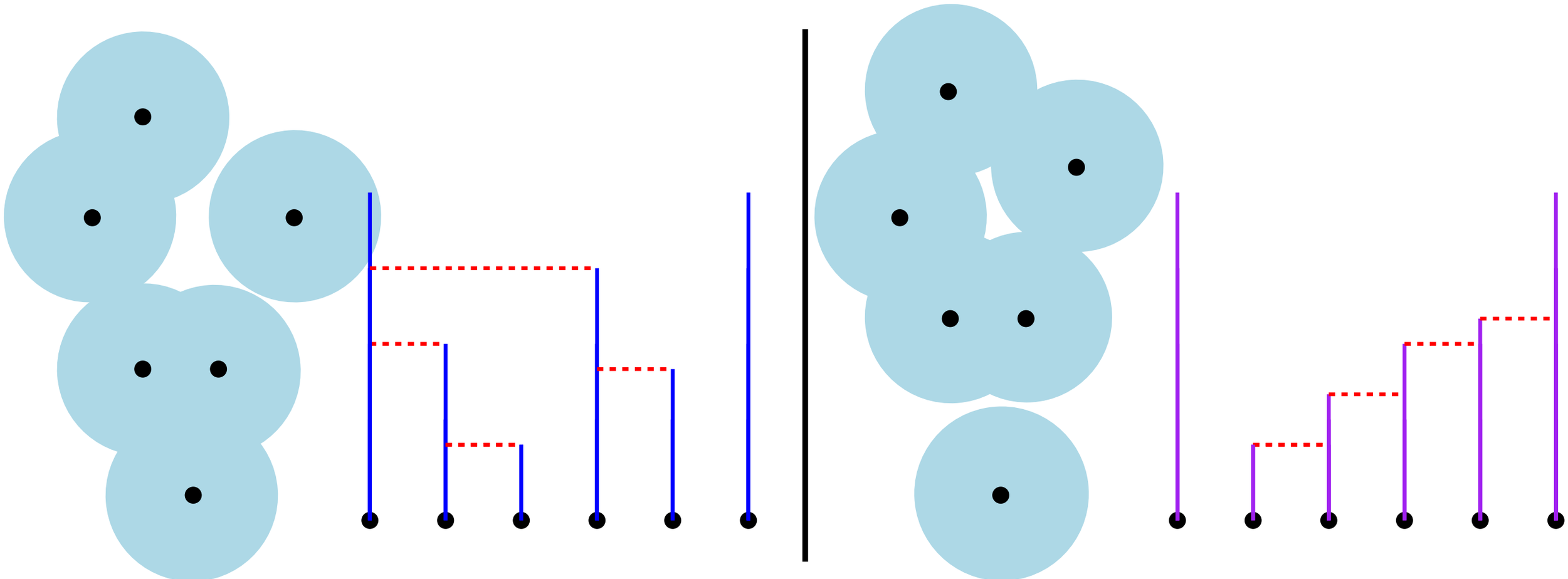
The example of distance functions



$$f_P : \quad \mathbb{R}^2 \to \mathbb{R}$$
$$x \mapsto \min_{p \in P} \|x - p\|_2$$

Link with single linkage clustering

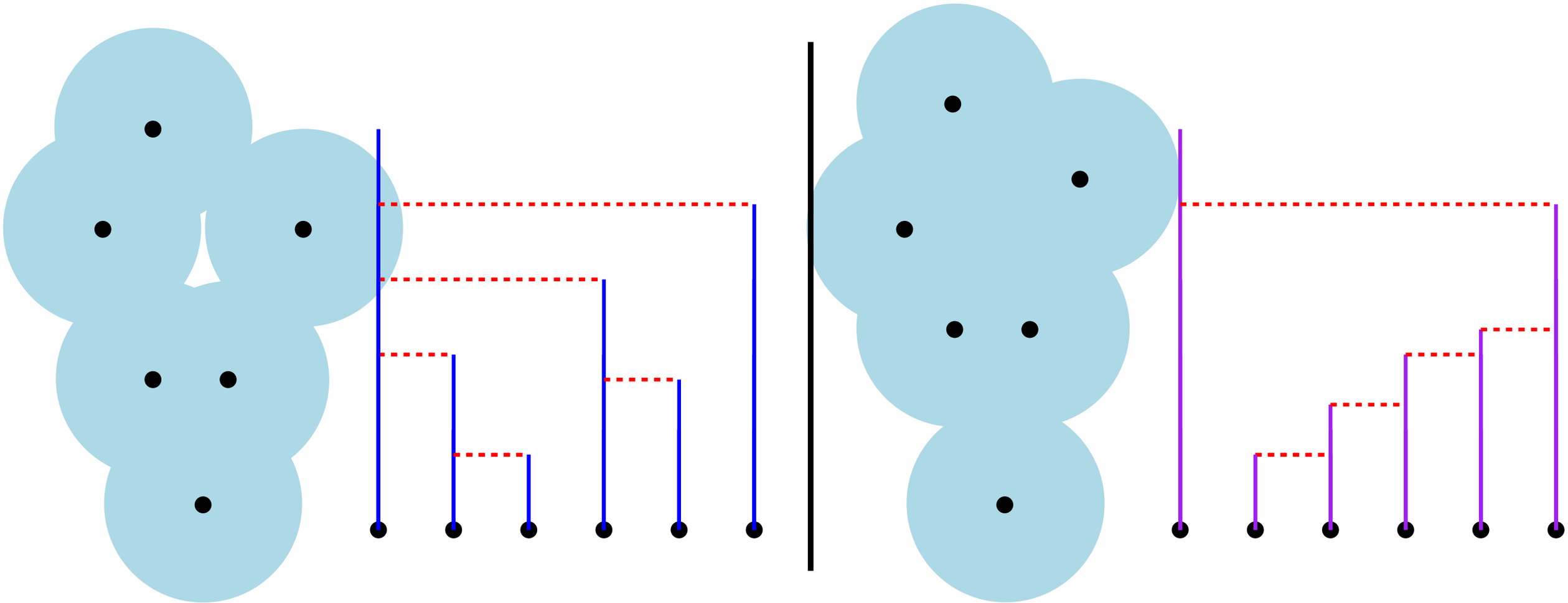
The example of distance functions



$$f_P : \quad \mathbb{R}^2 \to \mathbb{R}$$
$$x \mapsto \min_{p \in P} \|x - p\|_2$$

Link with single linkage clustering

The example of distance functions



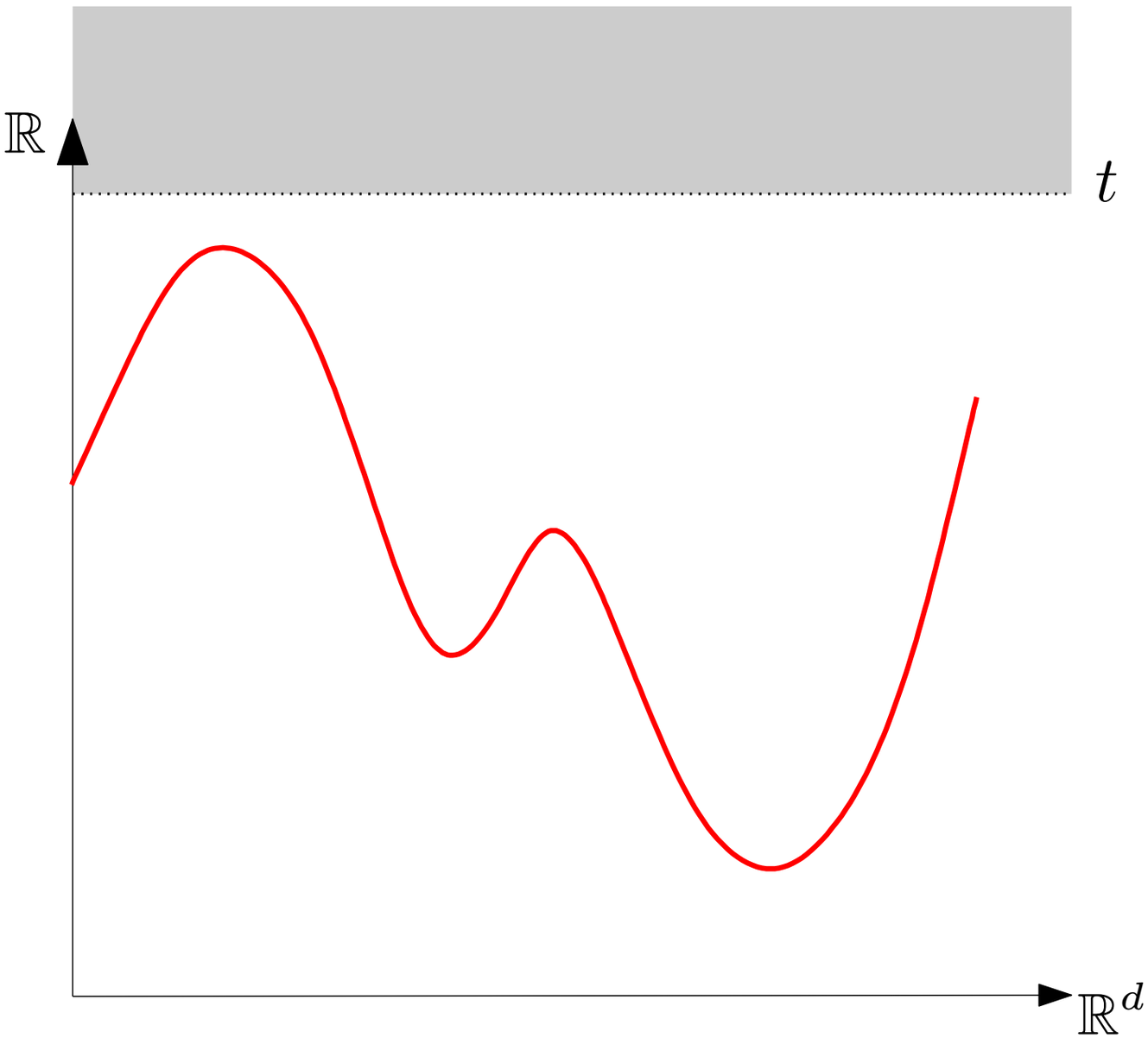
$$f_P : \quad \mathbb{R}^2 \to \mathbb{R}$$
$$x \mapsto \min_{p \in P} \|x - p\|_2$$

Link with single linkage clustering

The case of density and back to mode seeking

Given a probability density f:

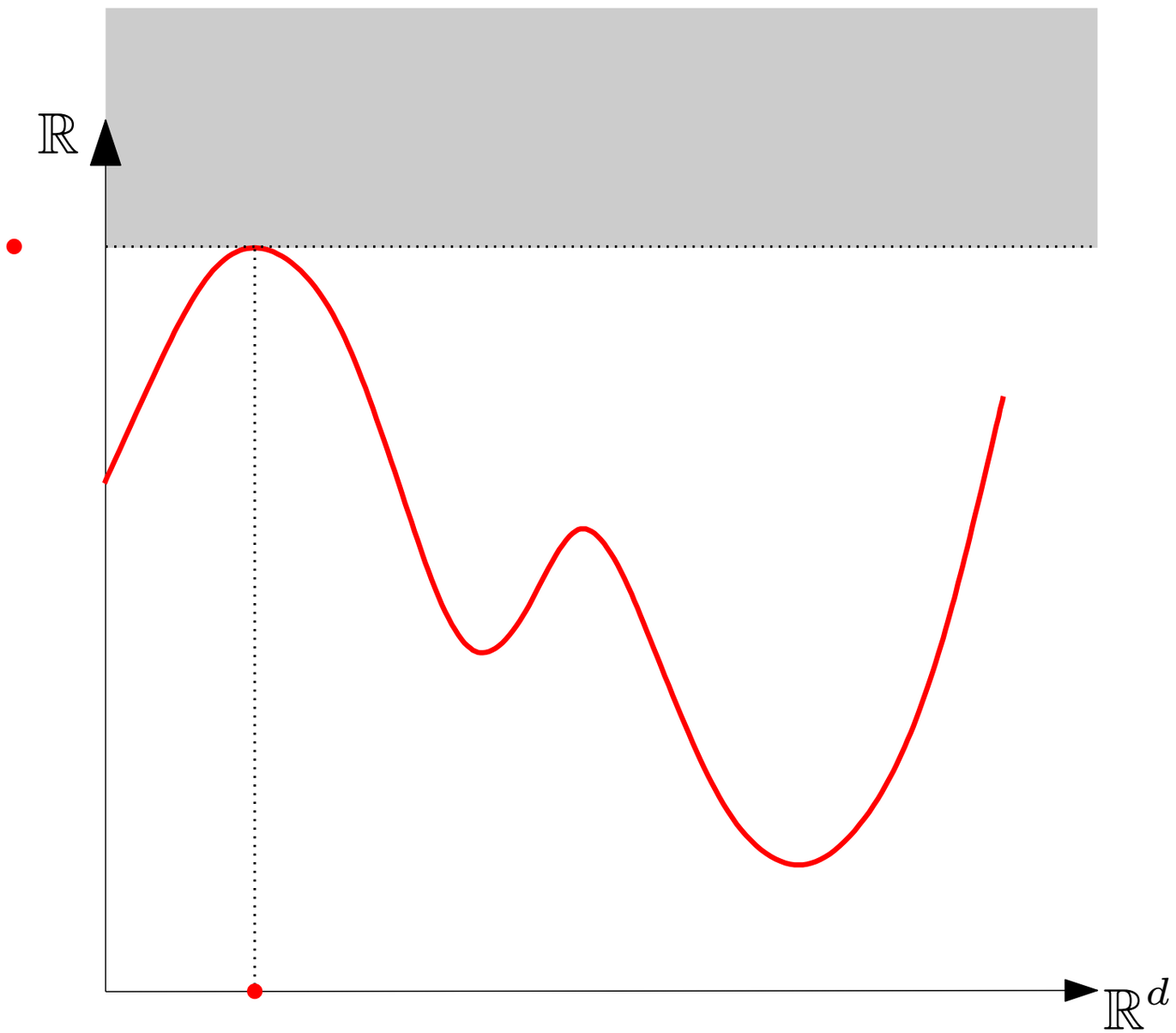
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

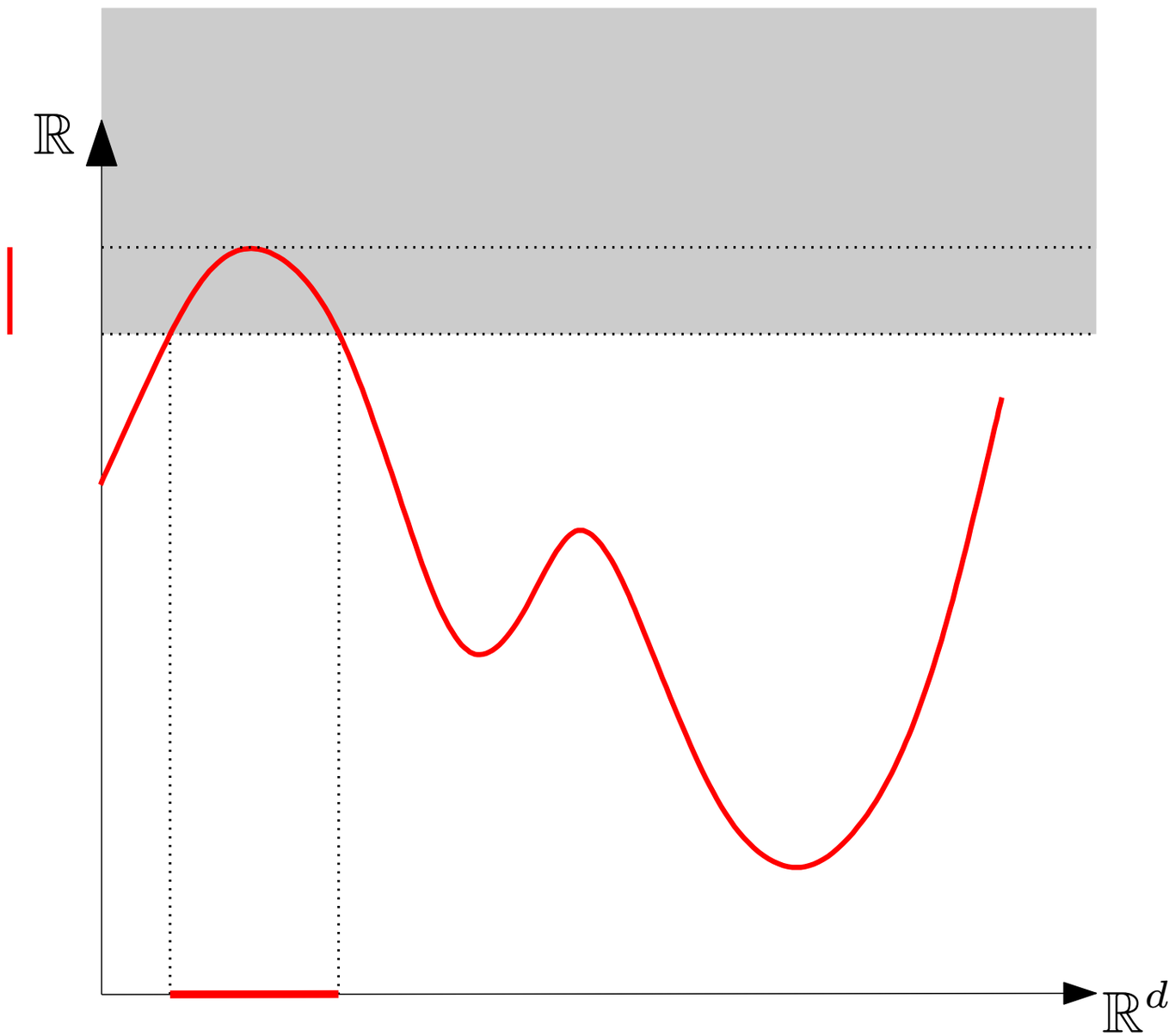
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

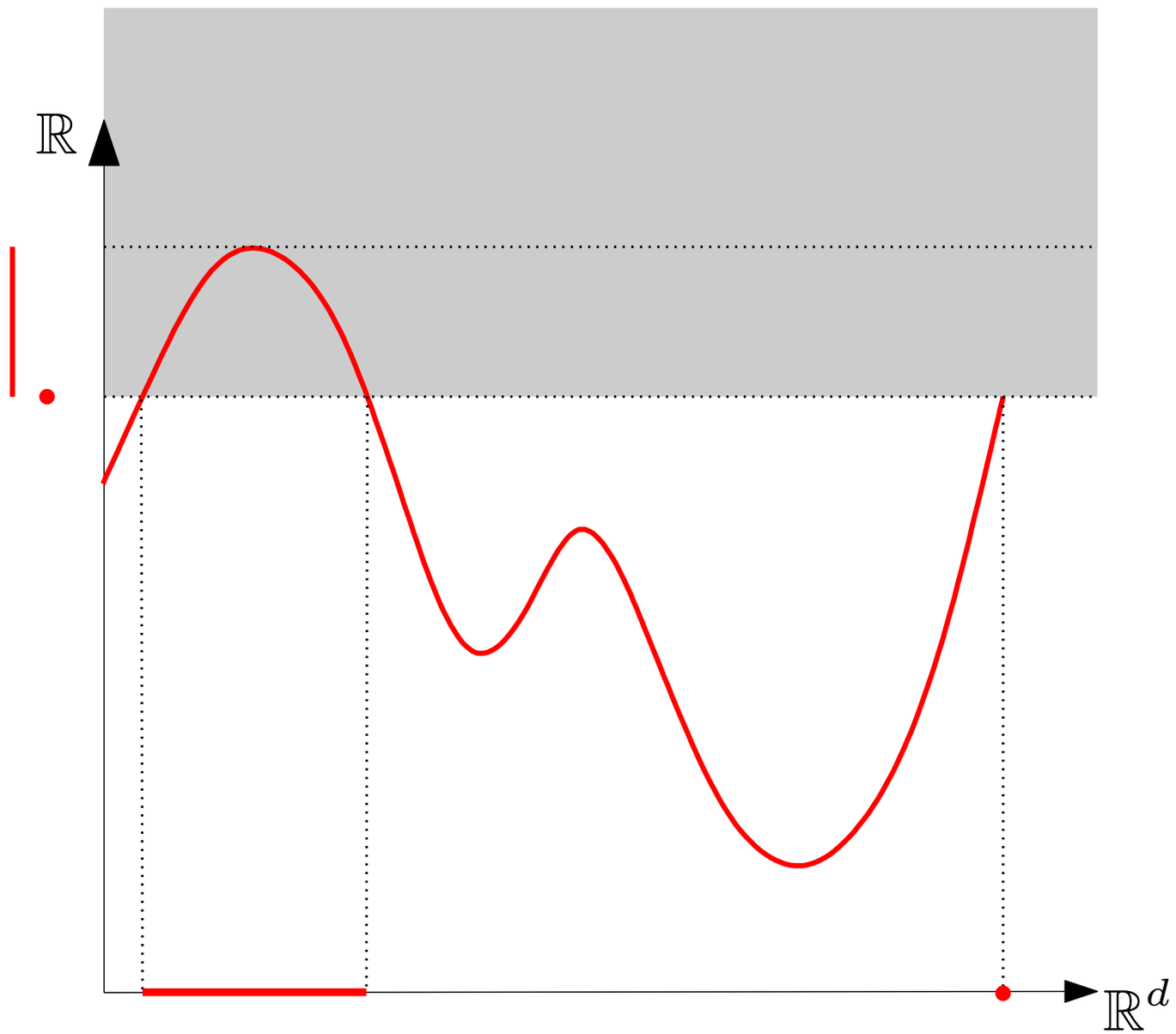
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

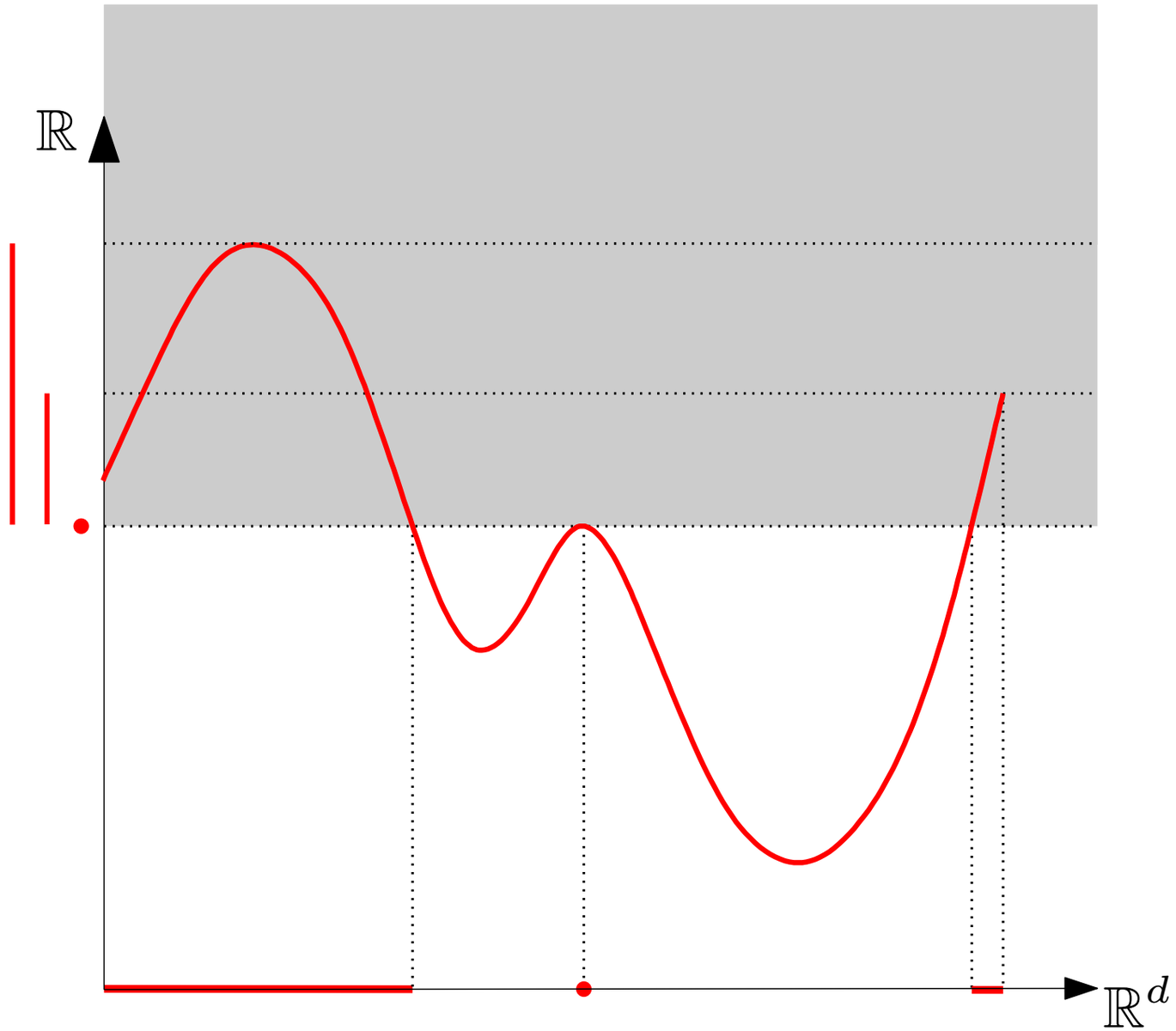
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

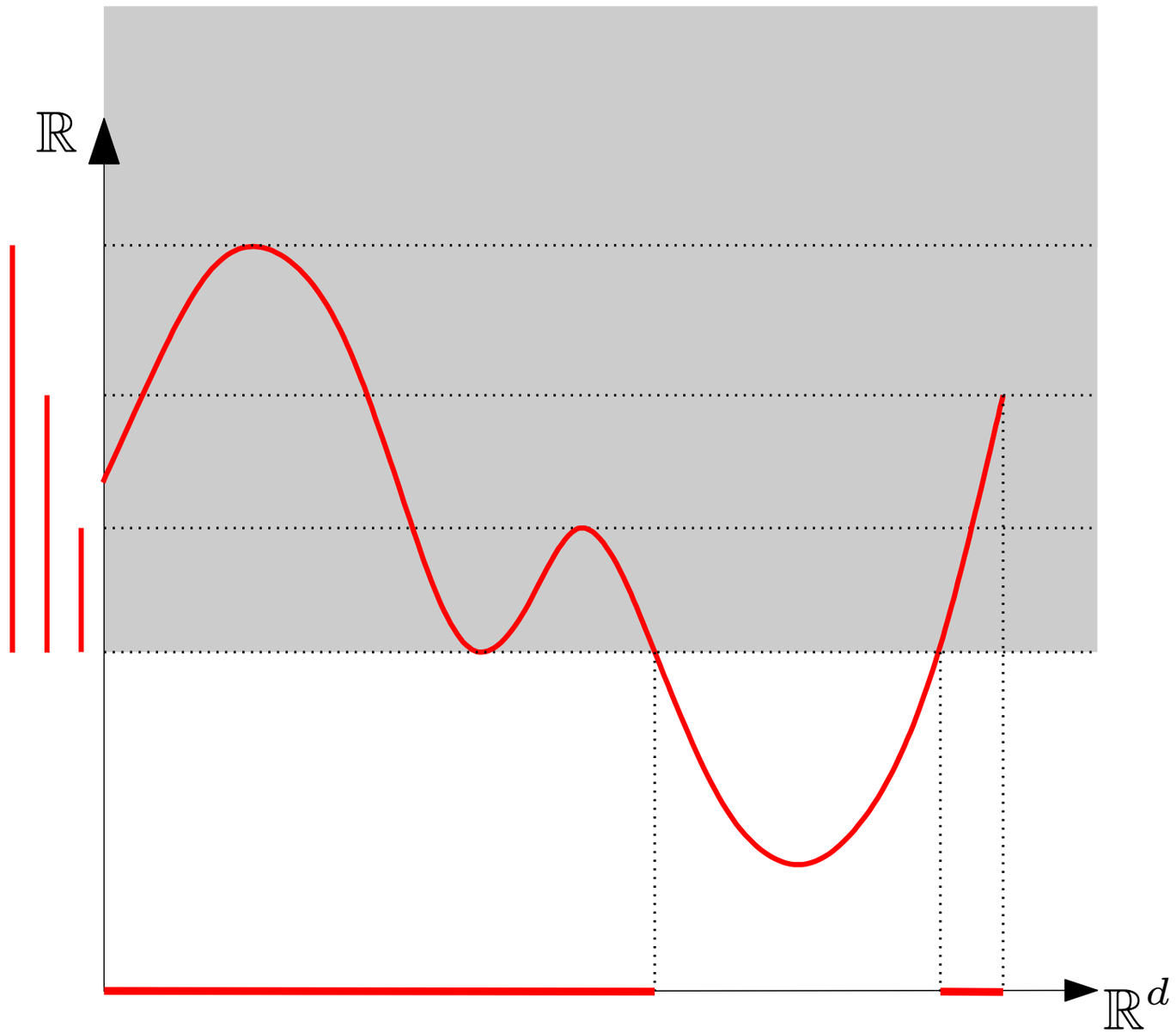
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

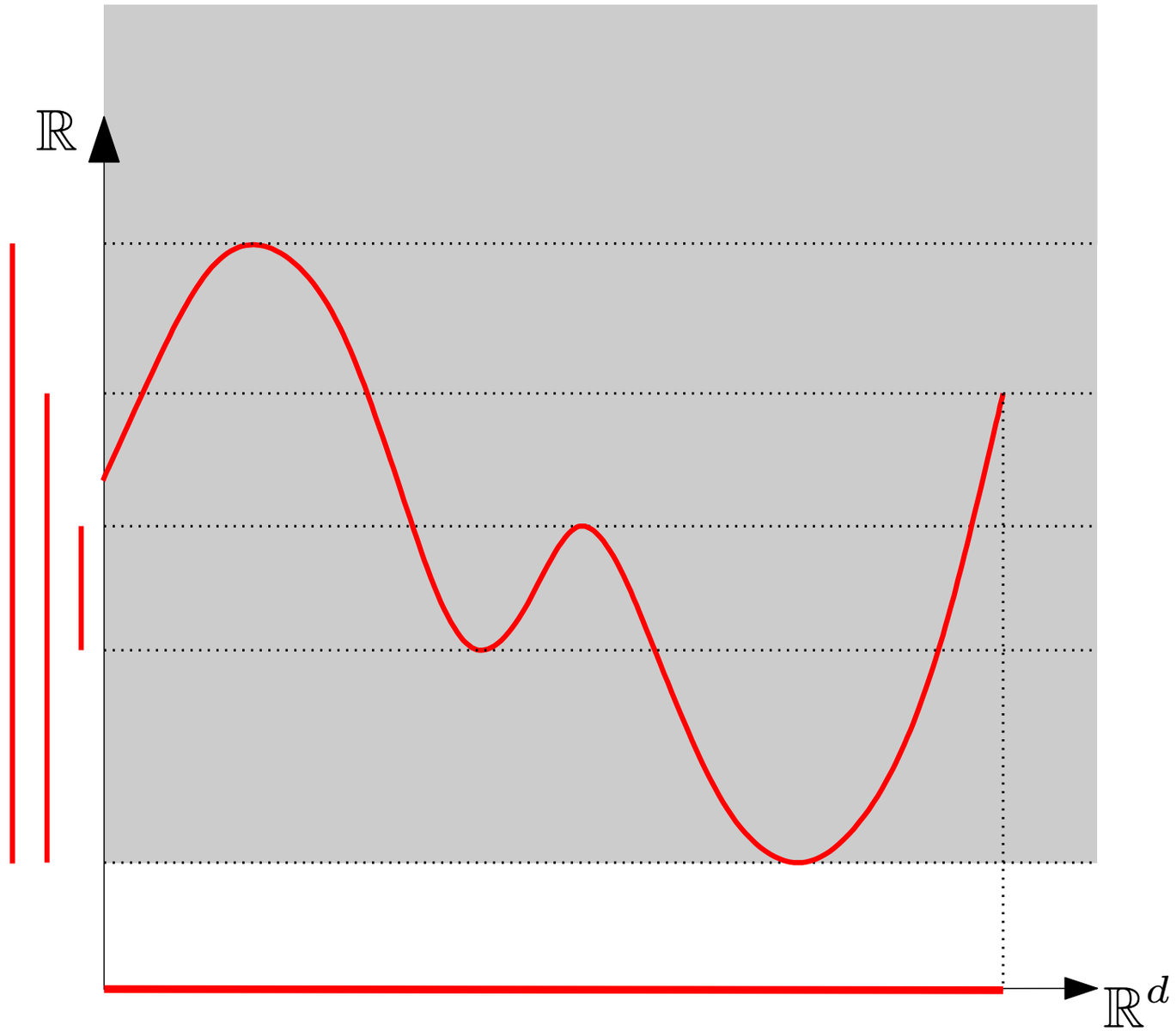
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

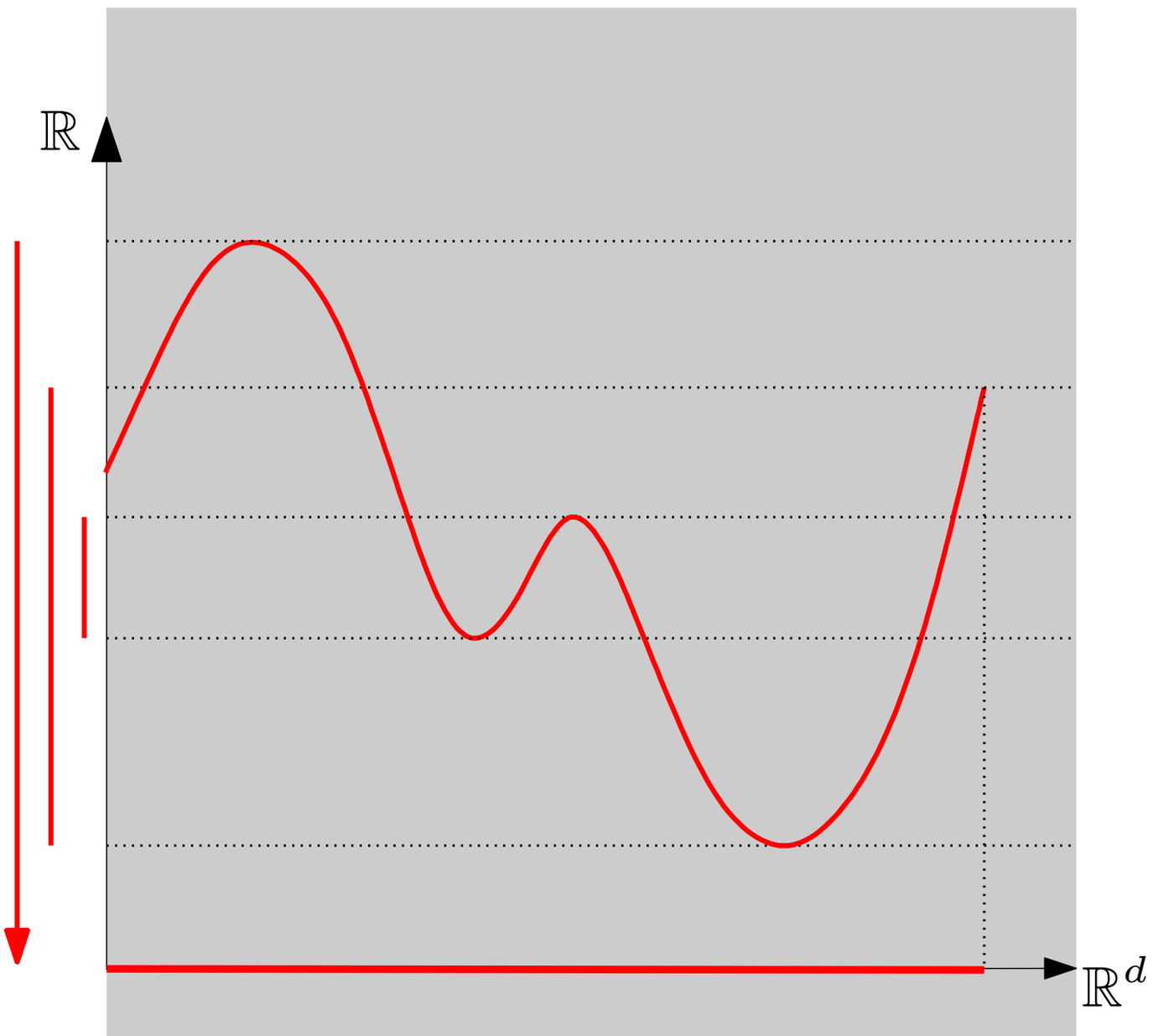
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

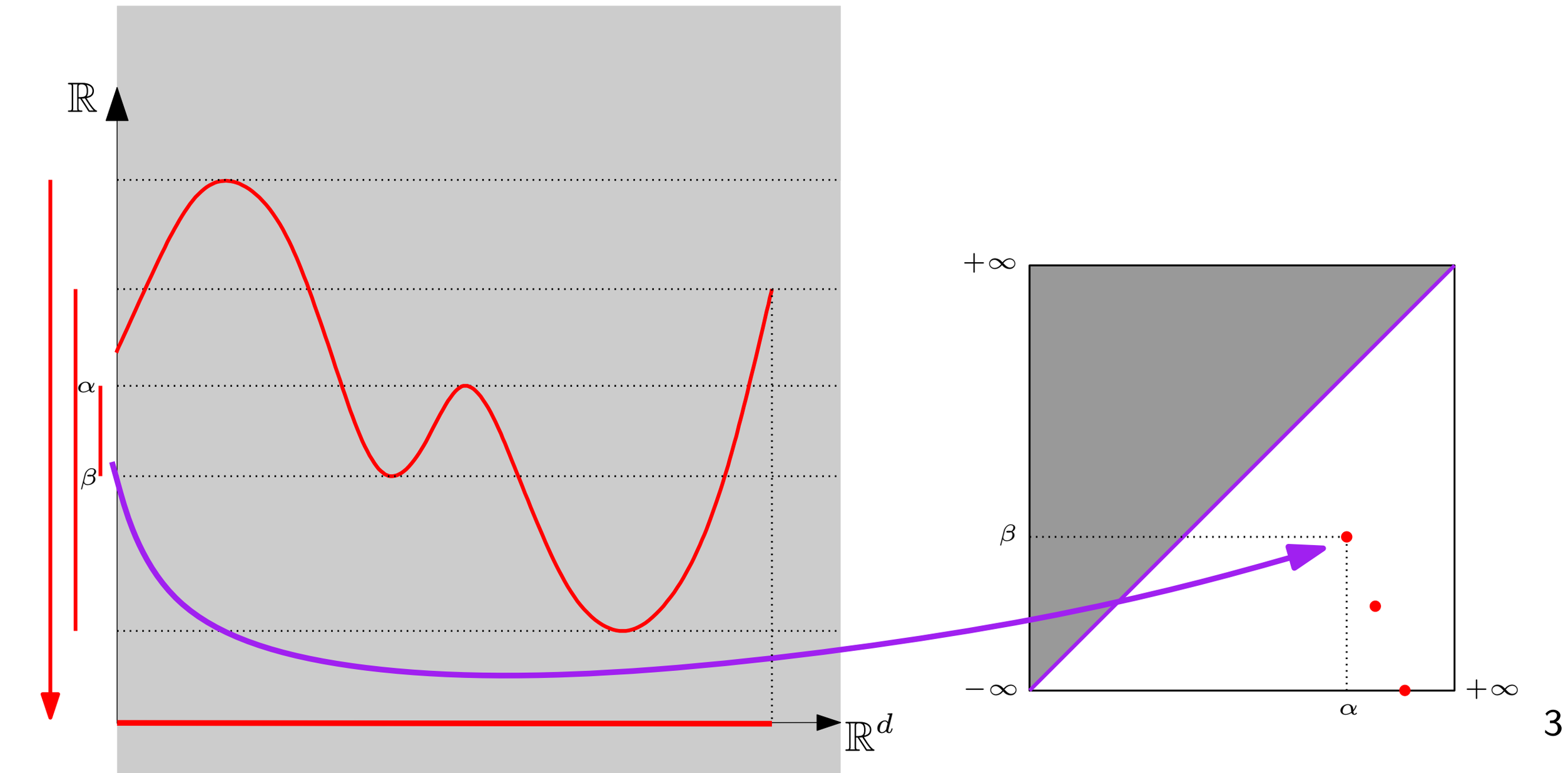
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



The case of density and back to mode seeking

Given a probability density f:

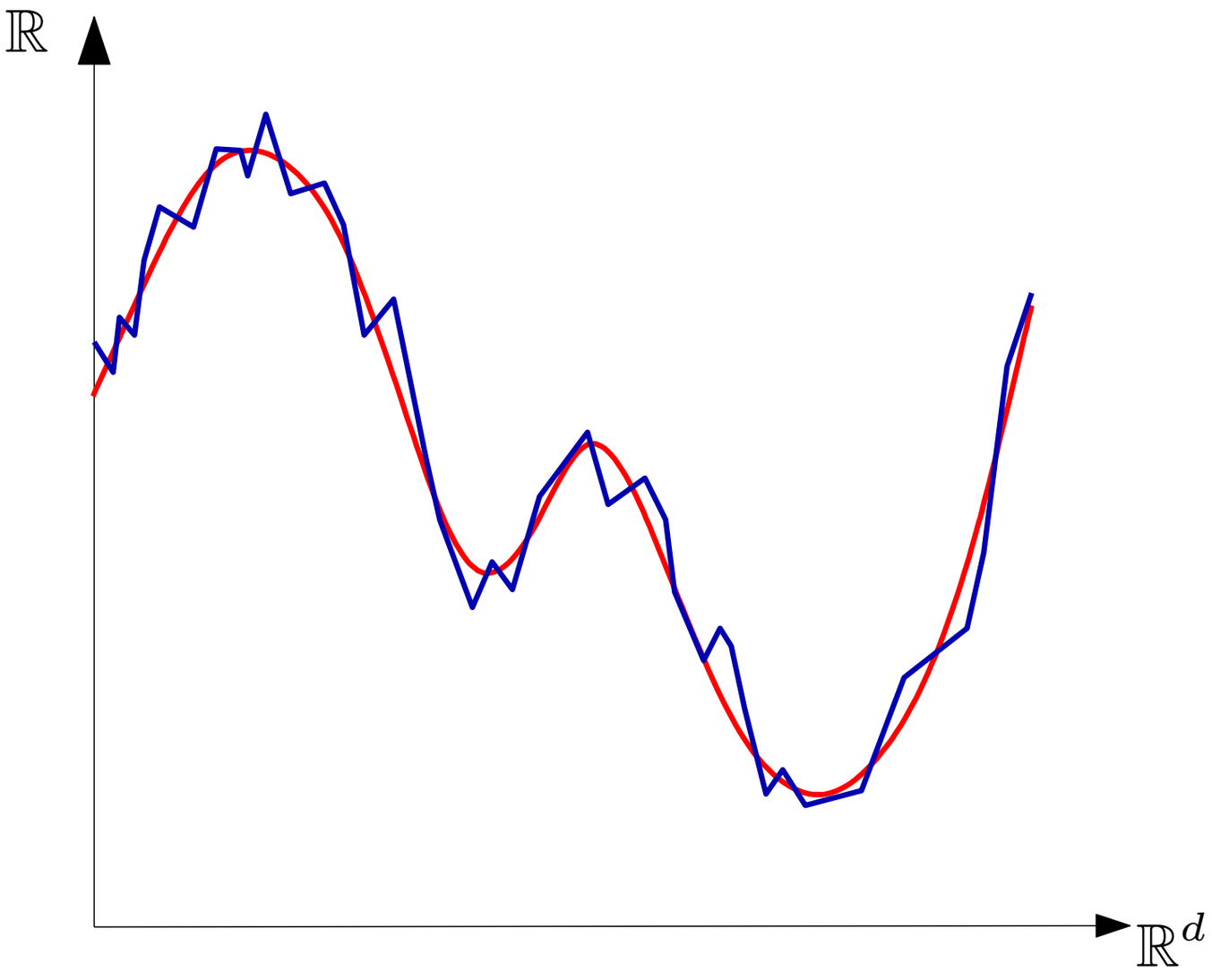
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way

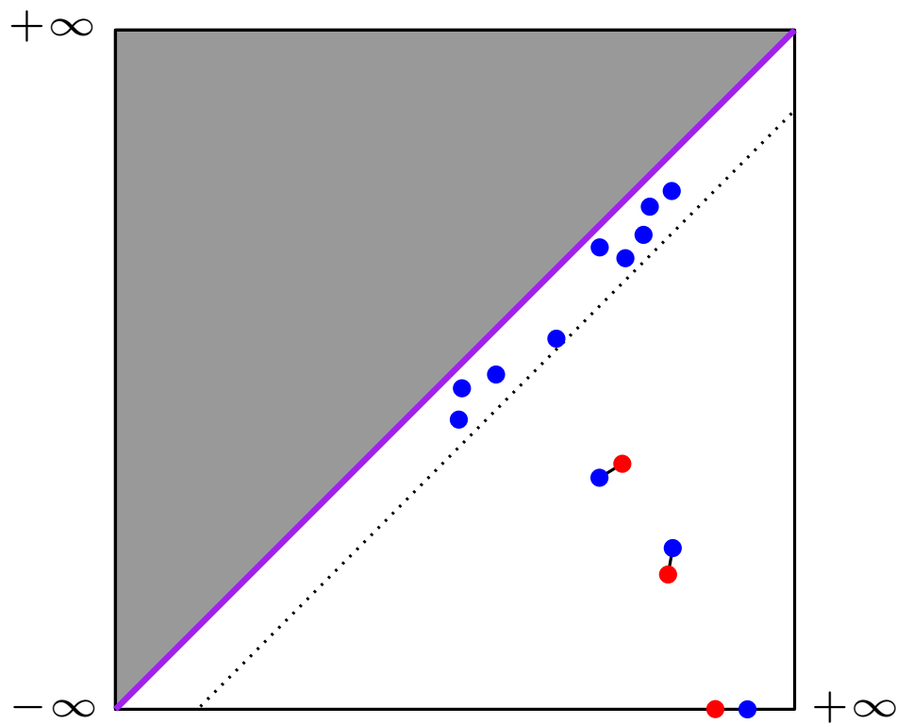


The case of density and back to mode seeking

Given an estimator $\hat{f}$:

Stability Theorem $\Rightarrow d_B(Df, D\hat{f}) \leq \|f - \hat{f}\|_\infty$.

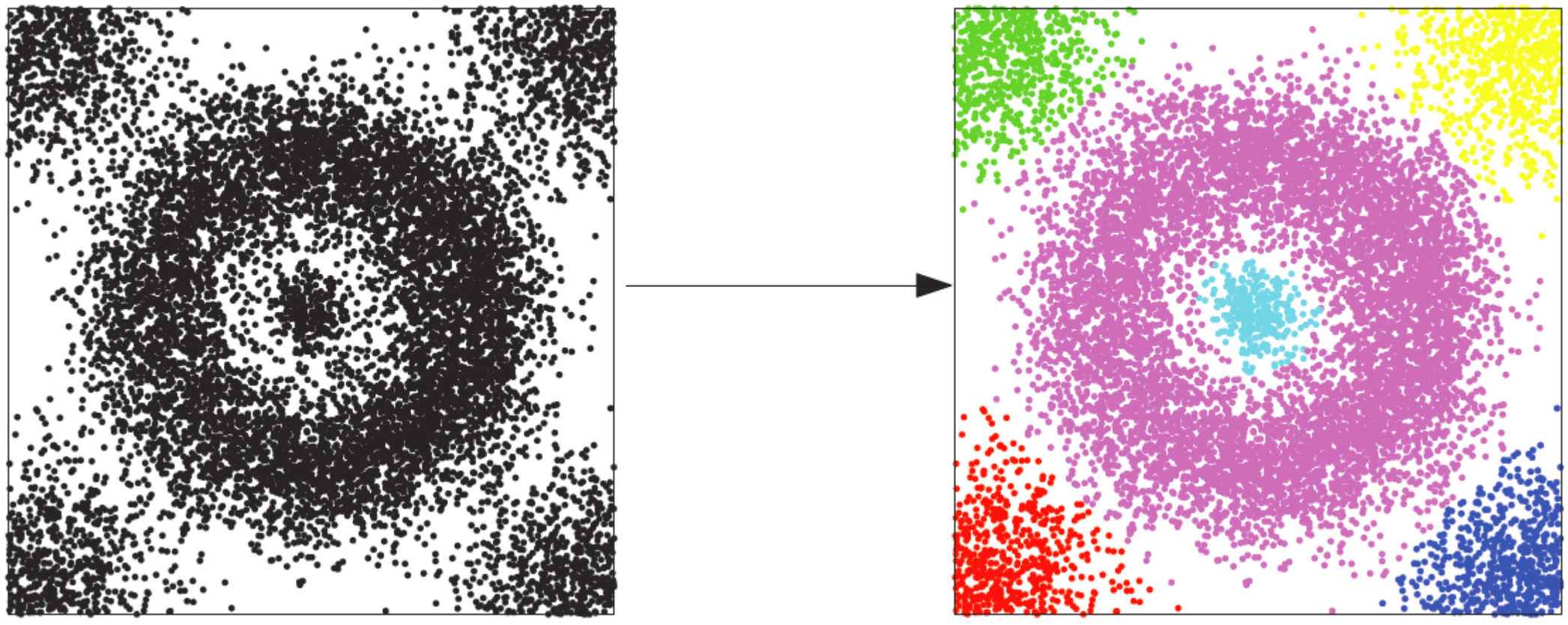




Persistence-based clustering

Combine a mode seeking approach with (0-dim) persistence computation.

[C.,Guibas,Oudot,Skraba - J. ACM 2013]



Input:

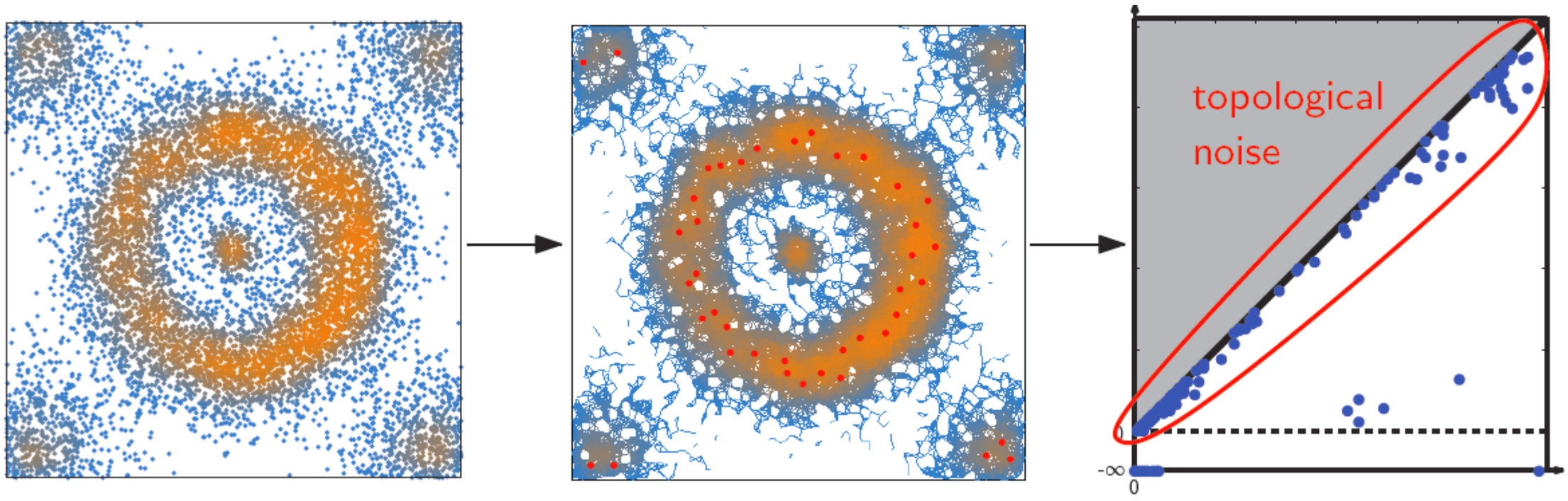
1. A finite set X of observations (point cloud with coordinates or pairwise distance matrix),
2. A real valued function f defined on the observations (e.g. density estimate).

Goal: Partition the data according to the basins of attraction of the peaks of f

Persistence-based clustering

Combine a mode seeking approach with (0-dim) persistence computation.

[C.,Guibas,Oudot,Skraba - J. ACM 2013]



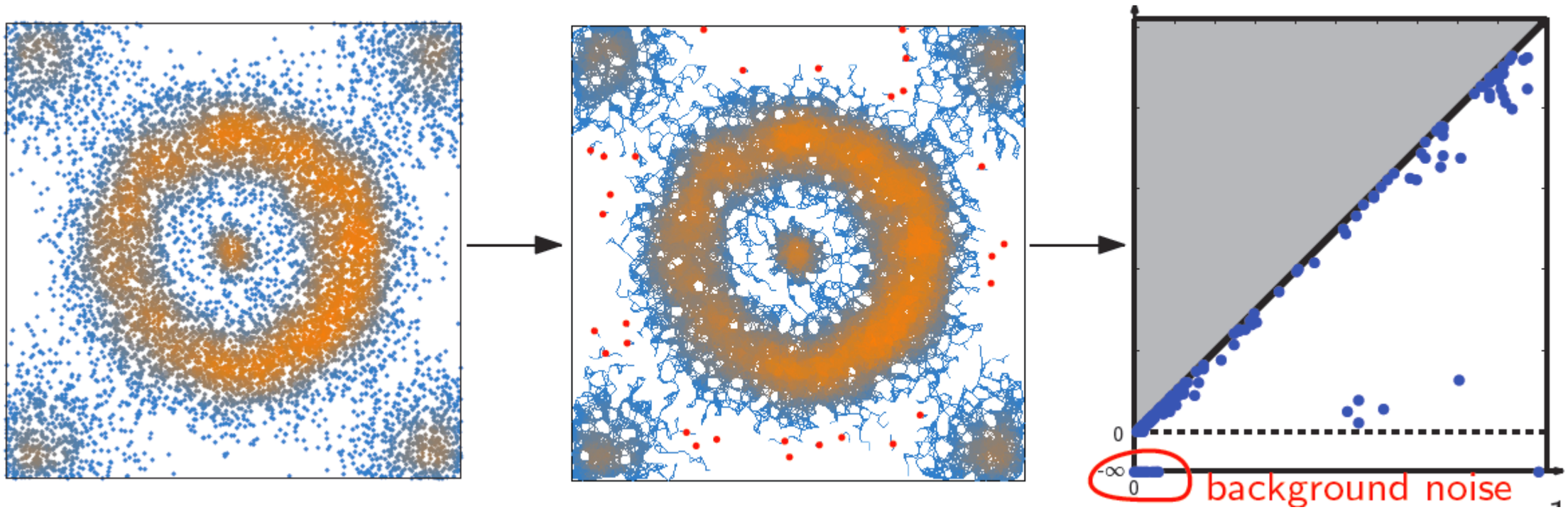
1. Build a neighborhing graph G on top of X.

2. Compute the (0-dim) persistence of f to identify prominent peaks $\rightarrow$ number of clusters (union-find algorithm).

Persistence-based clustering

Combine a mode seeking approach with (0-dim) persistence computation.

[C.,Guibas,Oudot,Skraba - J. ACM 2013]



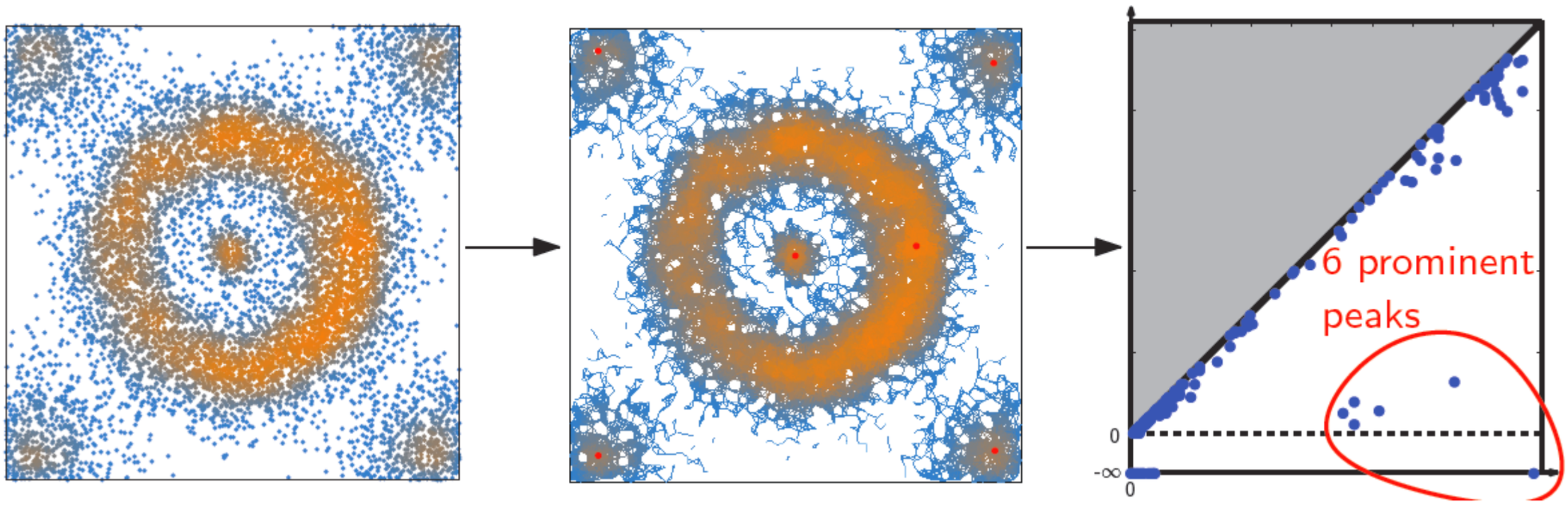
1. Build a neighborhing graph G on top of X.

2. Compute the (0-dim) persistence of f to identify prominent peaks $\rightarrow$ number of clusters (union-find algorithm).

Persistence-based clustering

Combine a mode seeking approach with (0-dim) persistence computation.

[C.,Guibas,Oudot,Skraba - J. ACM 2013]



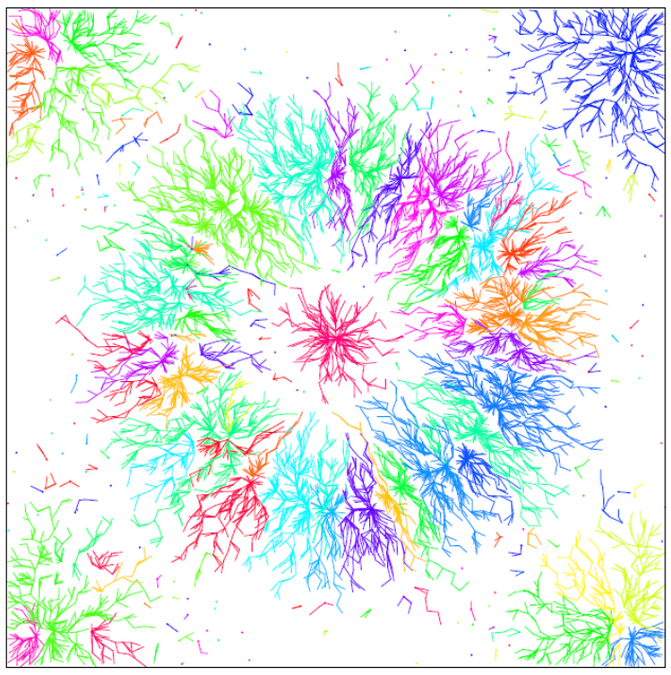
1. Build a neighborhing graph G on top of X.

2. Compute the (0-dim) persistence of f to identify prominent peaks $\rightarrow$ number of clusters (union-find algorithm).

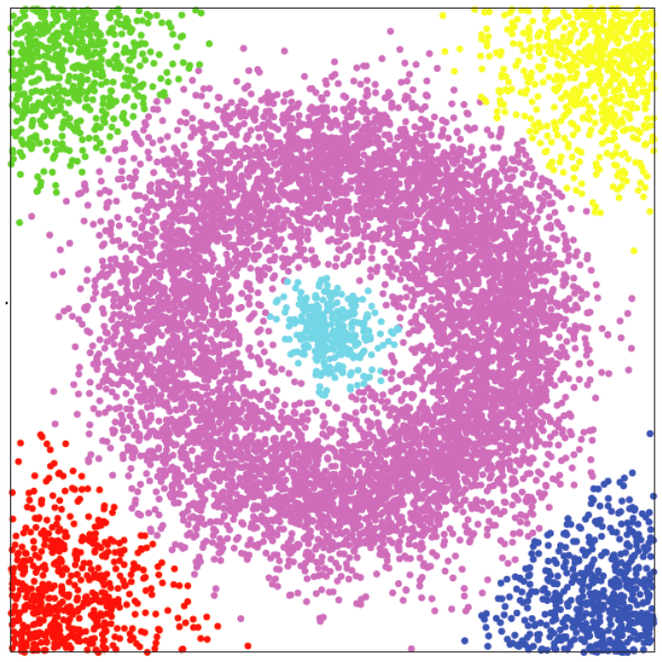
Persistence-based clustering

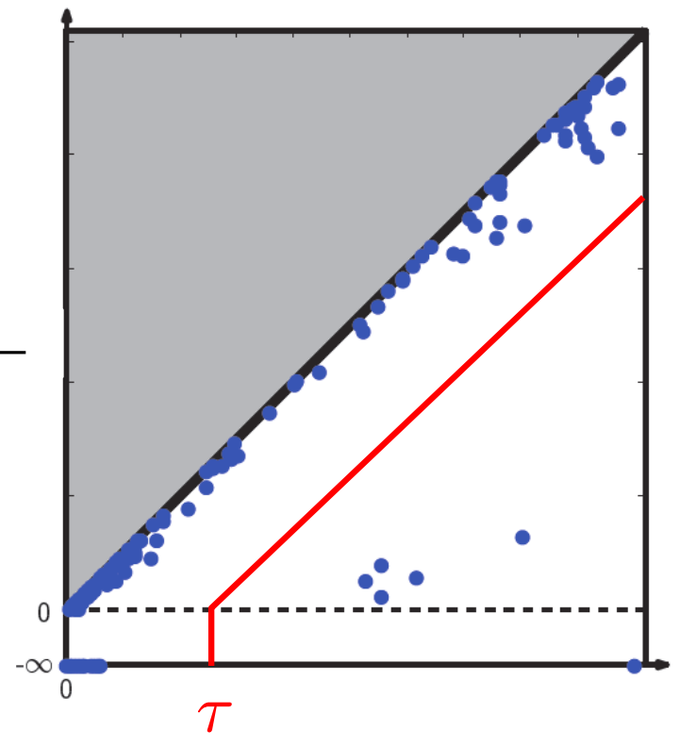
Combine a mode seeking approach with (0-dim) persistence computation.

[C.,Guibas,Oudot,Skraba - J. ACM 2013]



$\tau = 0$





1. Build a neighborhing graph G on top of X.

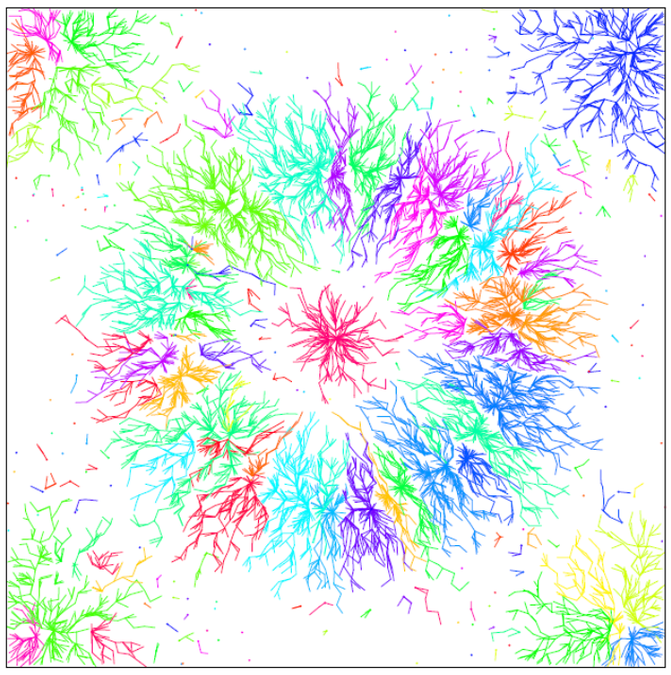
2. Compute the (0-dim) persistence of f to identify prominent peaks $\rightarrow$ number of clusters (union-find algorithm).

3. Chose a threshold $\tau > 0$ and use the persistence algorithm to merge components with prominence less than τ.

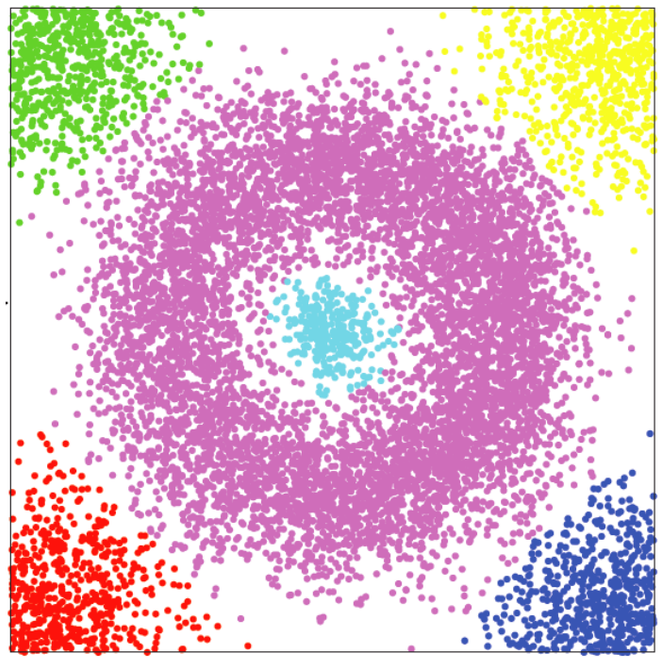
Persistence-based clustering

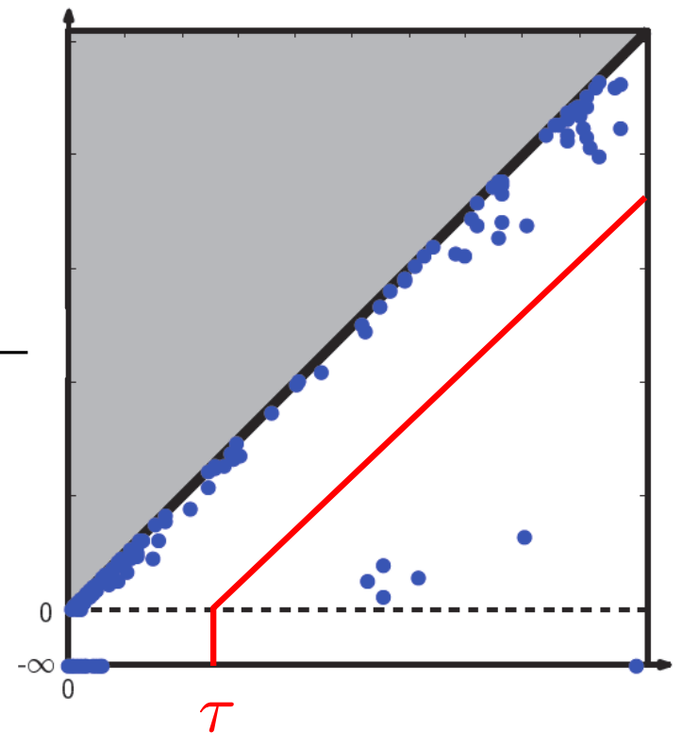
Combine a mode seeking approach with (0-dim) persistence computation.

[C.,Guibas,Oudot,Skraba - J. ACM 2013]



$\tau = 0$





Complexity of the algorithm: $\mathrm{O}(n \log n)$

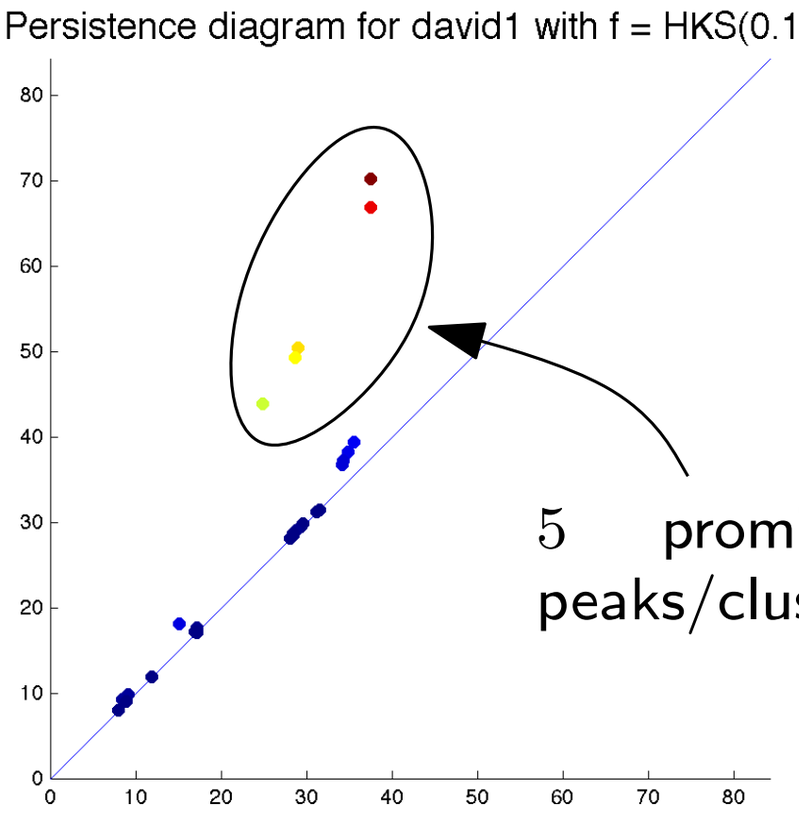
Theoretical guarantees:

- Stability of the number of clusters (w.r.t. perturbations of X and f).
- Partial stability of clusters: well identified stable parts in each cluster.

"soft " clustering

Application to non-rigid shape segmentation

[Skraba, Ovsjanikov, C.,Guibas, NORDIA 10]



X : a 3D shape

f = HKS function on X

Problem: some part of clusters are unstable $\rightarrow$ dirty segments

Application to non-rigid shape segmentation

[Skraba, Ovsjanikov, C.,Guibas, NORDIA 10]

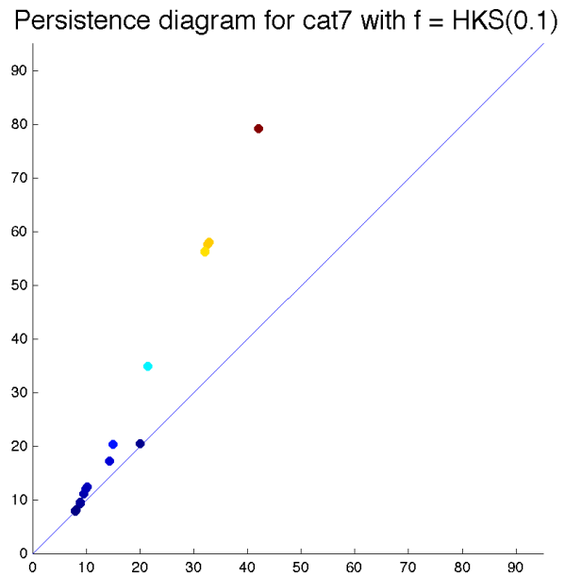
Problem: some part of clusters are unstable $\rightarrow$ dirty segments

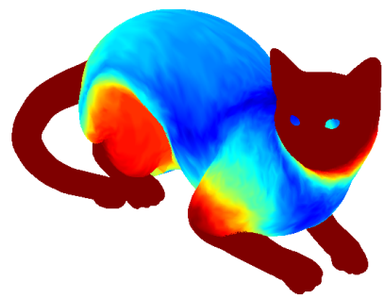
Idea:

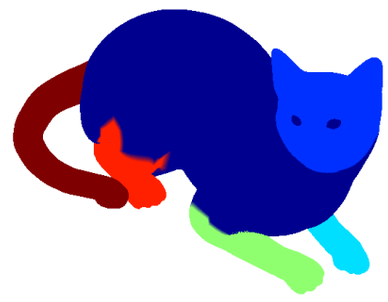
- Run the persistence based algorithm several times on random perturbations of f (size bounded by the "persistence" gap).
- Partial stability of clusters allows to establish correspondences between clusters across the different runs $\rightarrow$ for any $x \in X$, a vector giving the probability for x to belong to each cluster.

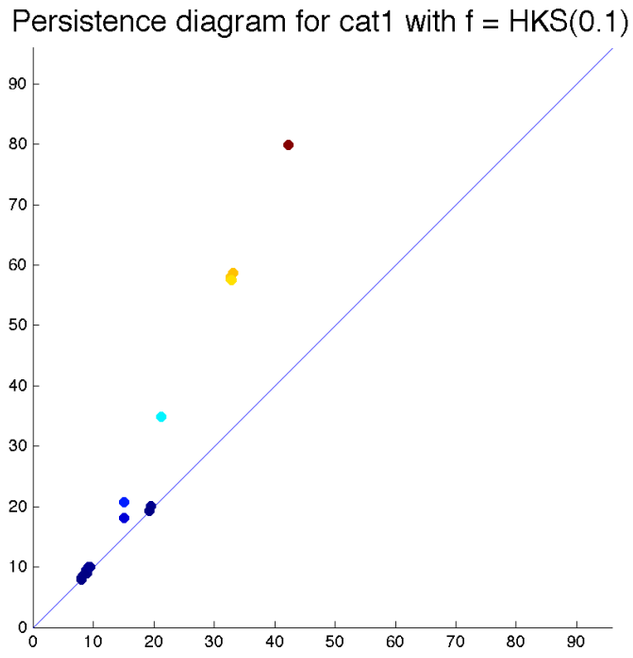
Application to non-rigid shape segmentation

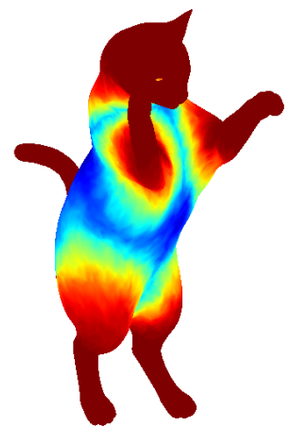
[Skraba, Ovsjanikov, C.,Guibas, NORDIA 10]







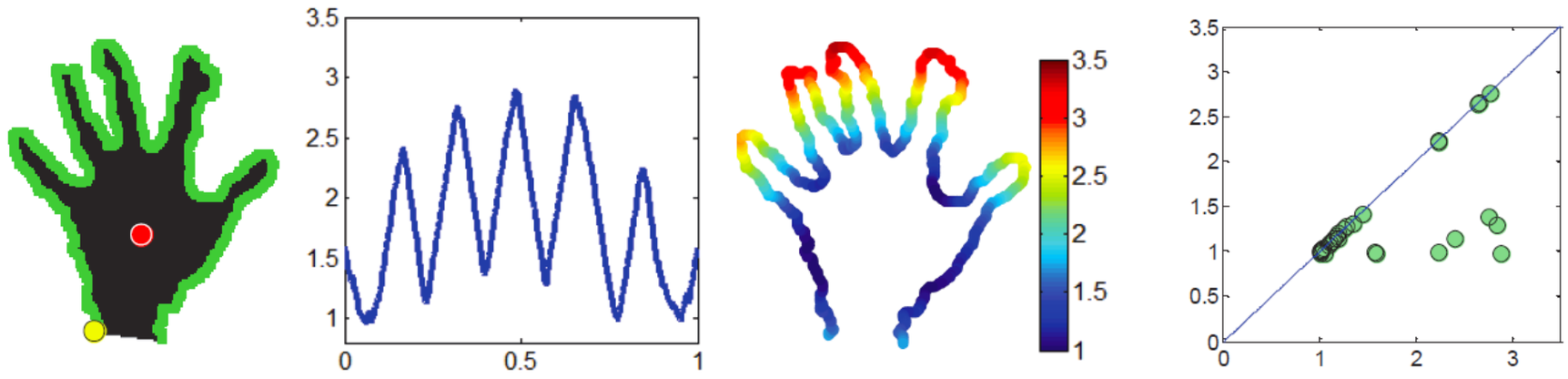




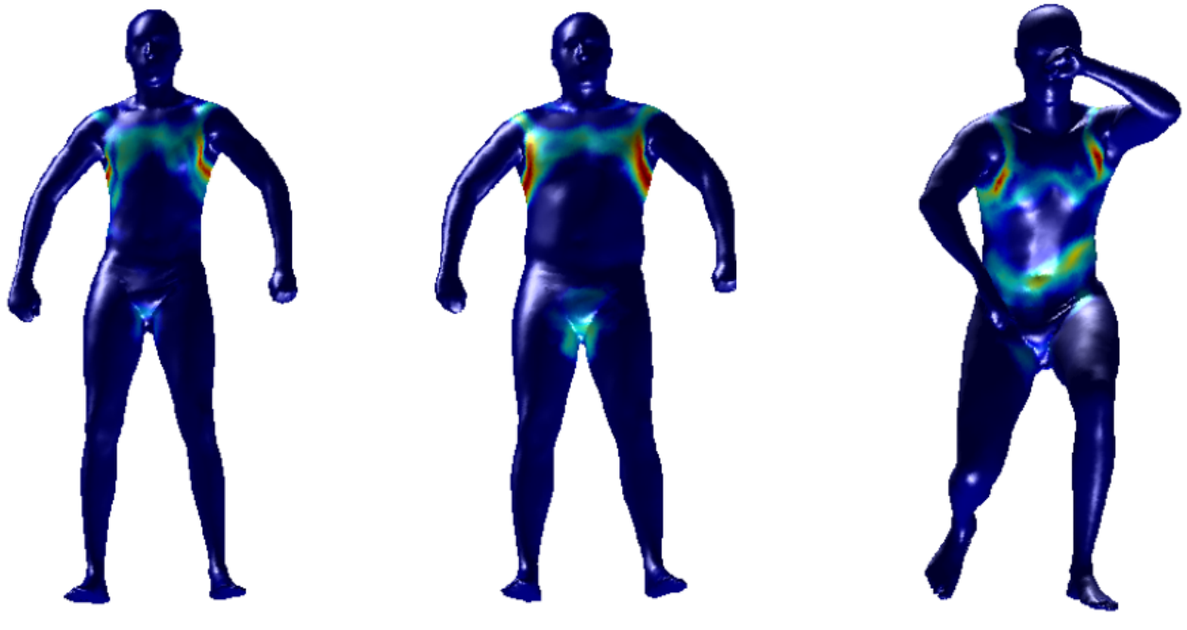
Other applications: classification, object recognition

Examples:

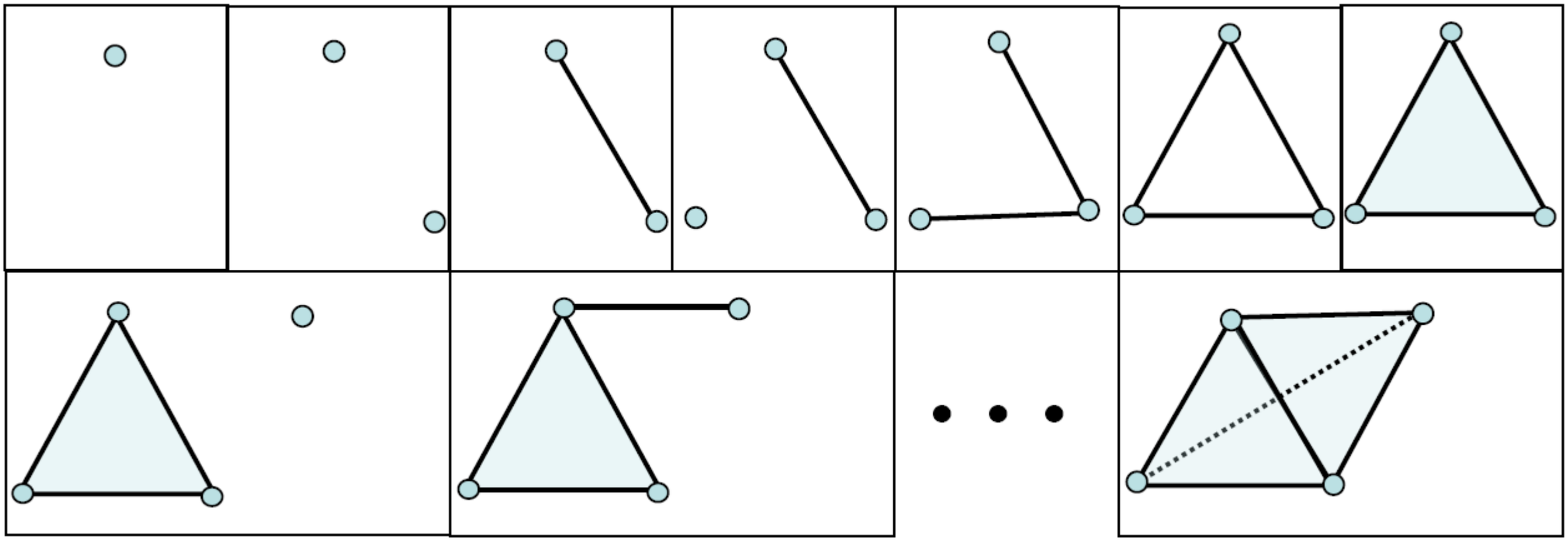
- Hand gesture recognition [Li, Ovsjanikov, C. - CVPR'14]



- Persistence-based pooling for shape recognition [Bonis, Ovsjanikov, Oudot, C. 2015]



Filtrations of simplicial complexes



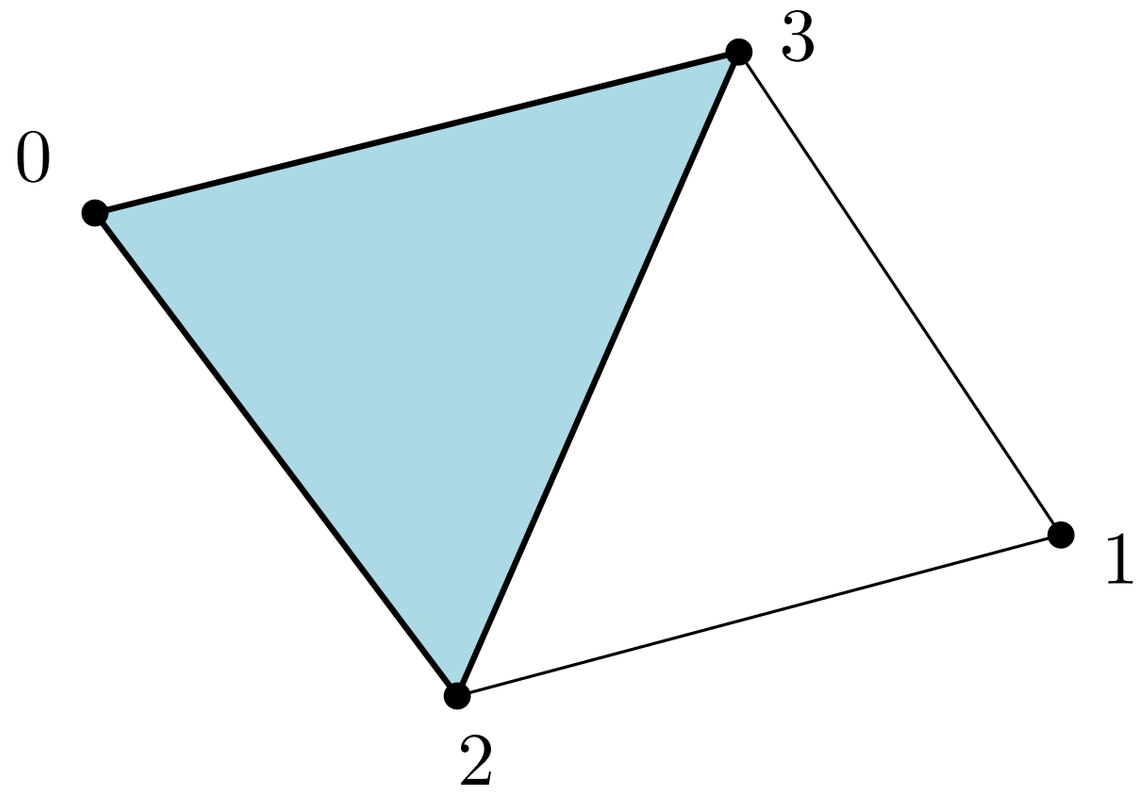
A filtration of a (finite) simplicial complex K is a sequence of subcomplexes such that

i) $\emptyset = K^0 \subset K^1 \subset \cdots \subset K^m = K$,

ii) $K^{i+1} = K^i \cup \sigma^{i+1}$ where σ^{i+1} is a simplex of K.

There are many ways to build filtrations - see the following of the course.

Sublevel set filtration associated to a function

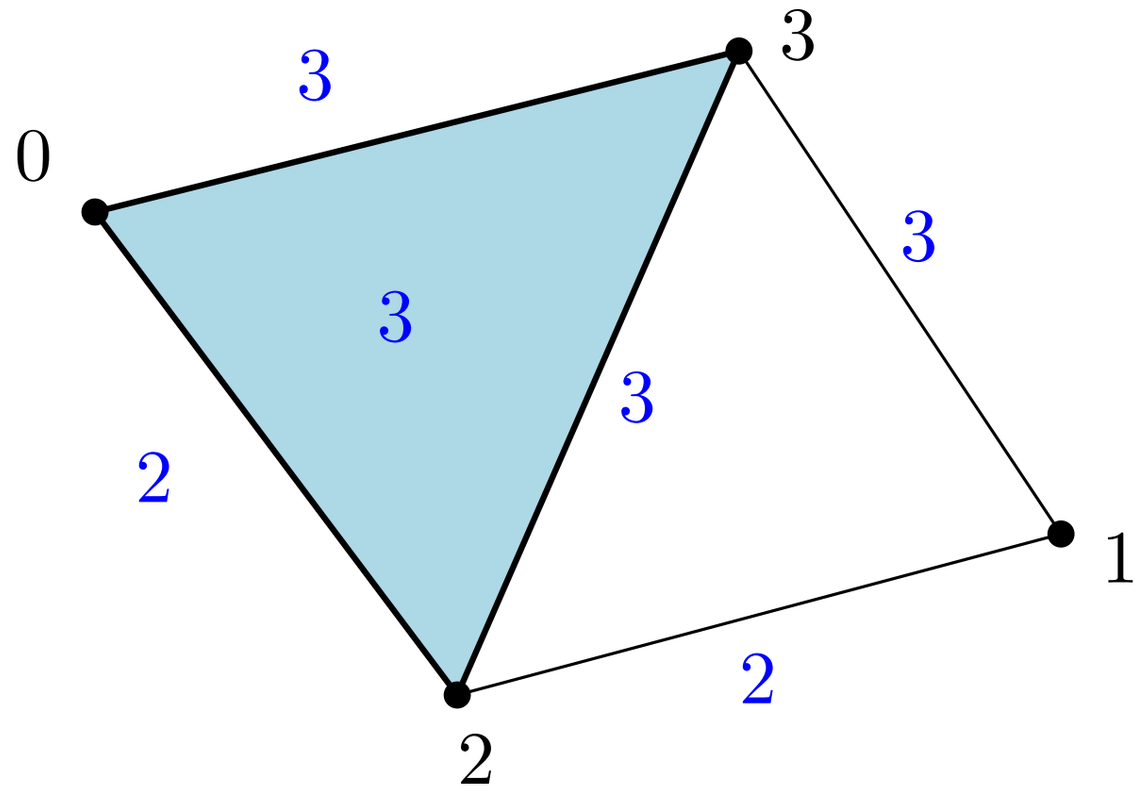


- f a real valued function defined on the vertices of K
- For $\sigma = [v_0, \cdots, v_k] \in K$, $f(\sigma) = \max_{i=0,\cdots,k} f(v_i)$
- The simplices of K are ordered according increasing f values (and dimension in case of equal values on different simplices).

Exercise: show that this is a filtration

General case: if $f : X \to \mathbb{R}$, X a topological space, $\forall t \leq t' \in \mathbb{R}$, $f^{-1}((-\infty, t]) \subseteq f^{-1}((-\infty, t']) \to$ filtration of X by the sublevel sets of f.

Sublevel set filtration associated to a function



Note: the upper level set filtration is defined similarly

- f a real valued function defined on the vertices of K
- For $\sigma = [v_0, \cdots, v_k] \in K$, $f(\sigma) = \max_{i=0,\cdots,k} f(v_i)$
- The simplices of K are ordered according increasing f values (and dimension in case of equal values on different simplices).

Exercise: show that this is a filtration

General case: if $f : X \to \mathbb{R}$, X a topological space, $\forall t \leq t' \in \mathbb{R}$, $f^{-1}((-\infty, t]) \subseteq f^{-1}((-\infty, t']) \to$ filtration of X by the sublevel sets of f.