2nd Workshop on Topological Methods in Data Analysis
4th - 6th October 2021, Heidelberg University

Simplicial Complexes and (Persistent) Homology

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

Class outline

The classes are about

Topological Data Analysis (TDA)

Class outline

The classes are about

Topological Data Analysis (TDA)

Goal: Study geometric data sets with techniques coming from *topology*.

Class outline

The classes are about

Topological Data Analysis (TDA)

Goal: Study geometric data sets with techniques coming from *topology*.

Question: What is topology?

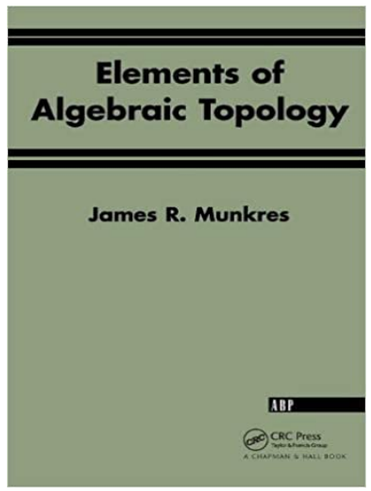
Class outline

The classes are about

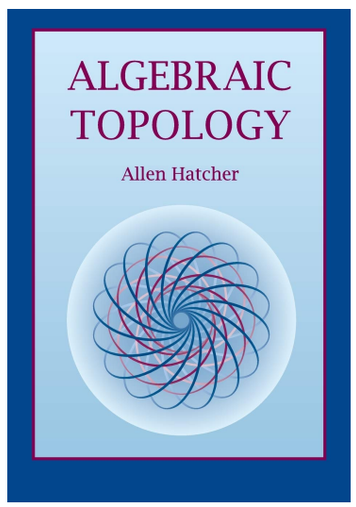
Topological Data Analysis (TDA)

Goal: Study geometric data sets with techniques coming from *topology*.

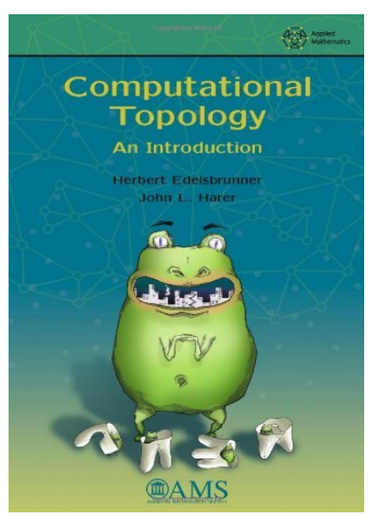
Question: What is topology?



[*Elements of Algebraic Topology*, Munkres, CRC Press, 1984]

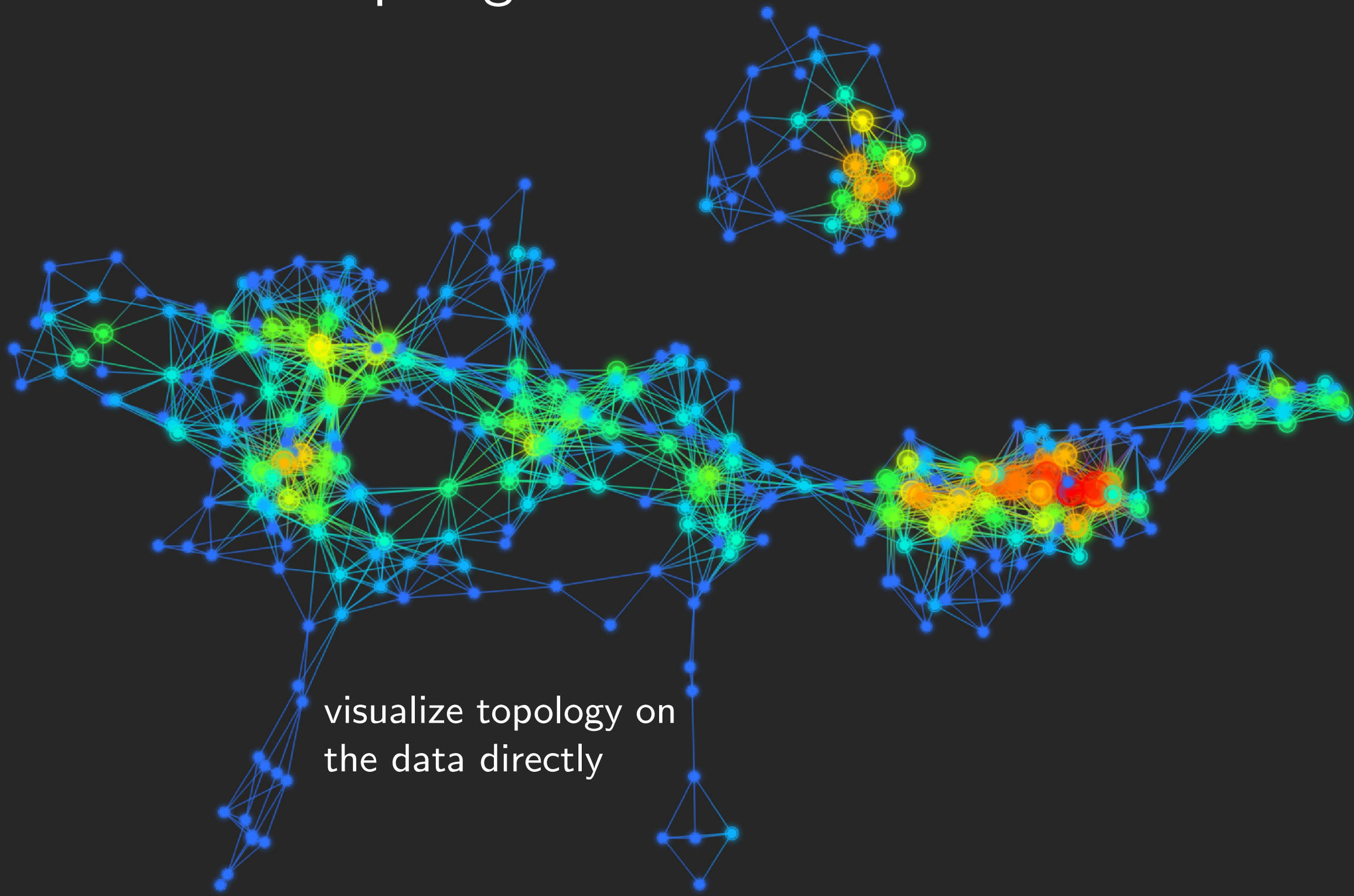


[*Algebraic Topology*, Hatcher, Cambridge University Press, 2002]



[*Computational Topology: an introduction*, Edelsbrunner, Harer, AMS, 2010]

Introduction: topological visualization



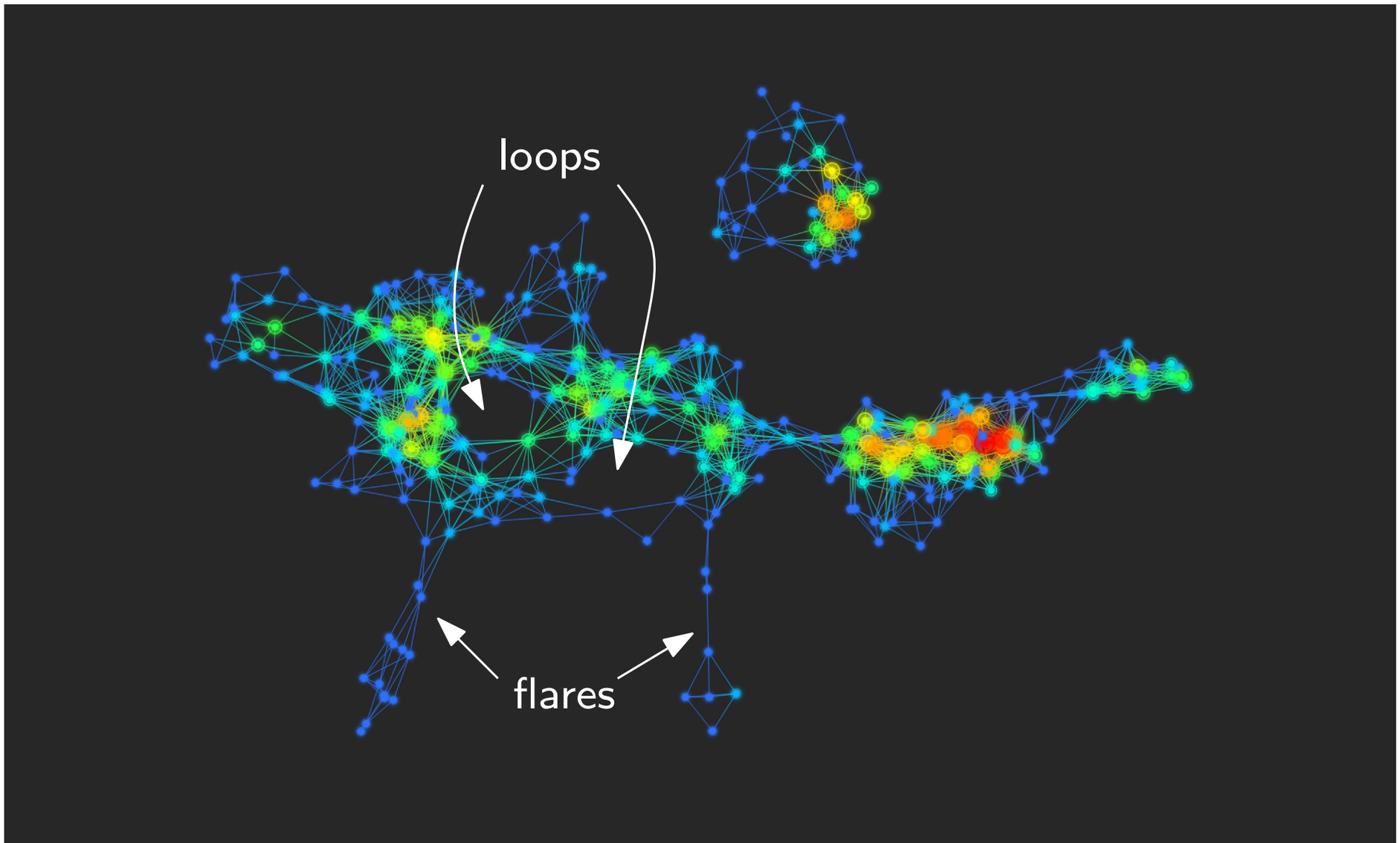
visualize topology on
the data directly

Introduction: topological visualization

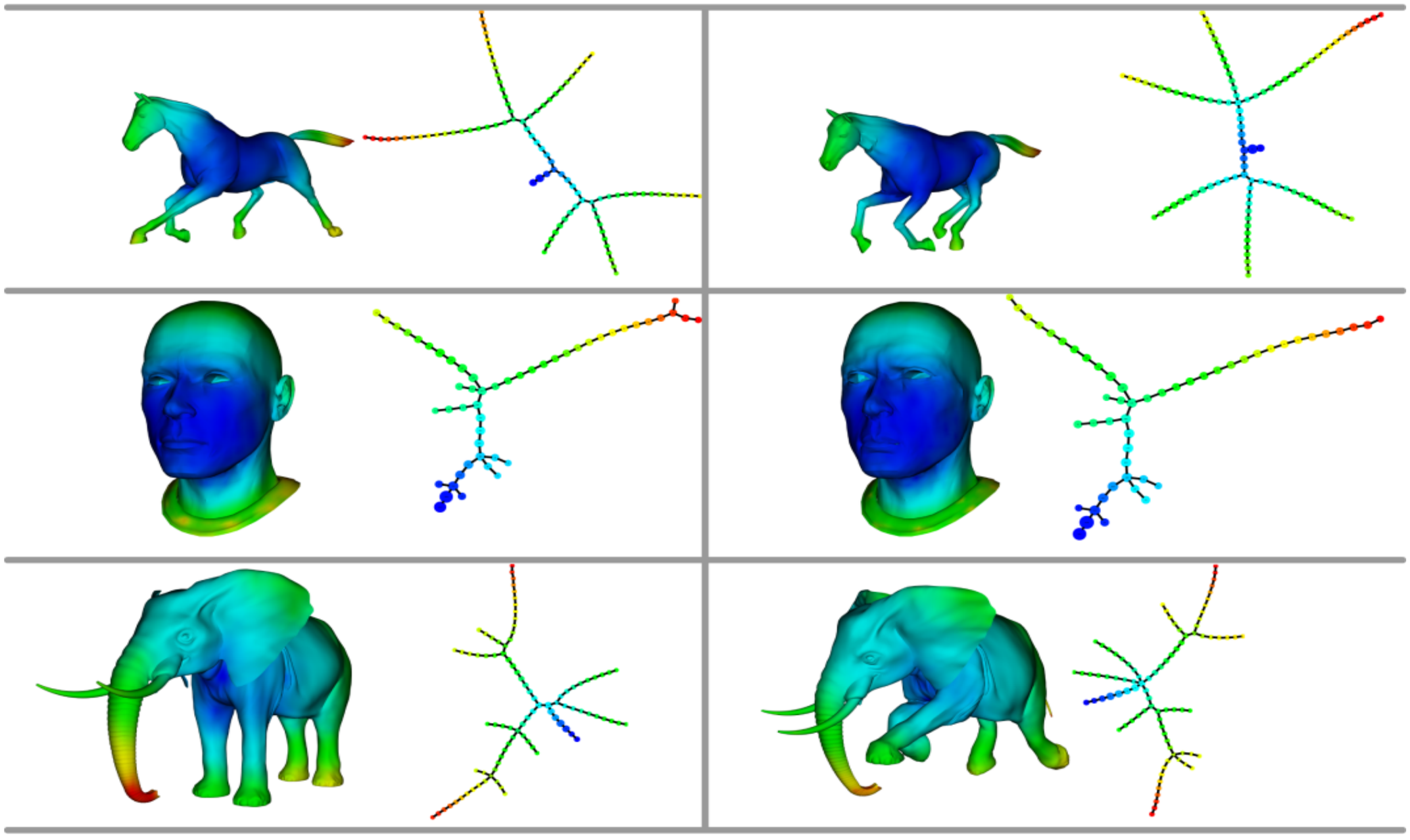
Two types of applications:

- clustering
- feature selection

Principle: identify statistically relevant subpopulations through topological patterns (flares, loops).



Introduction: topological visualization



3d shapes classification

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

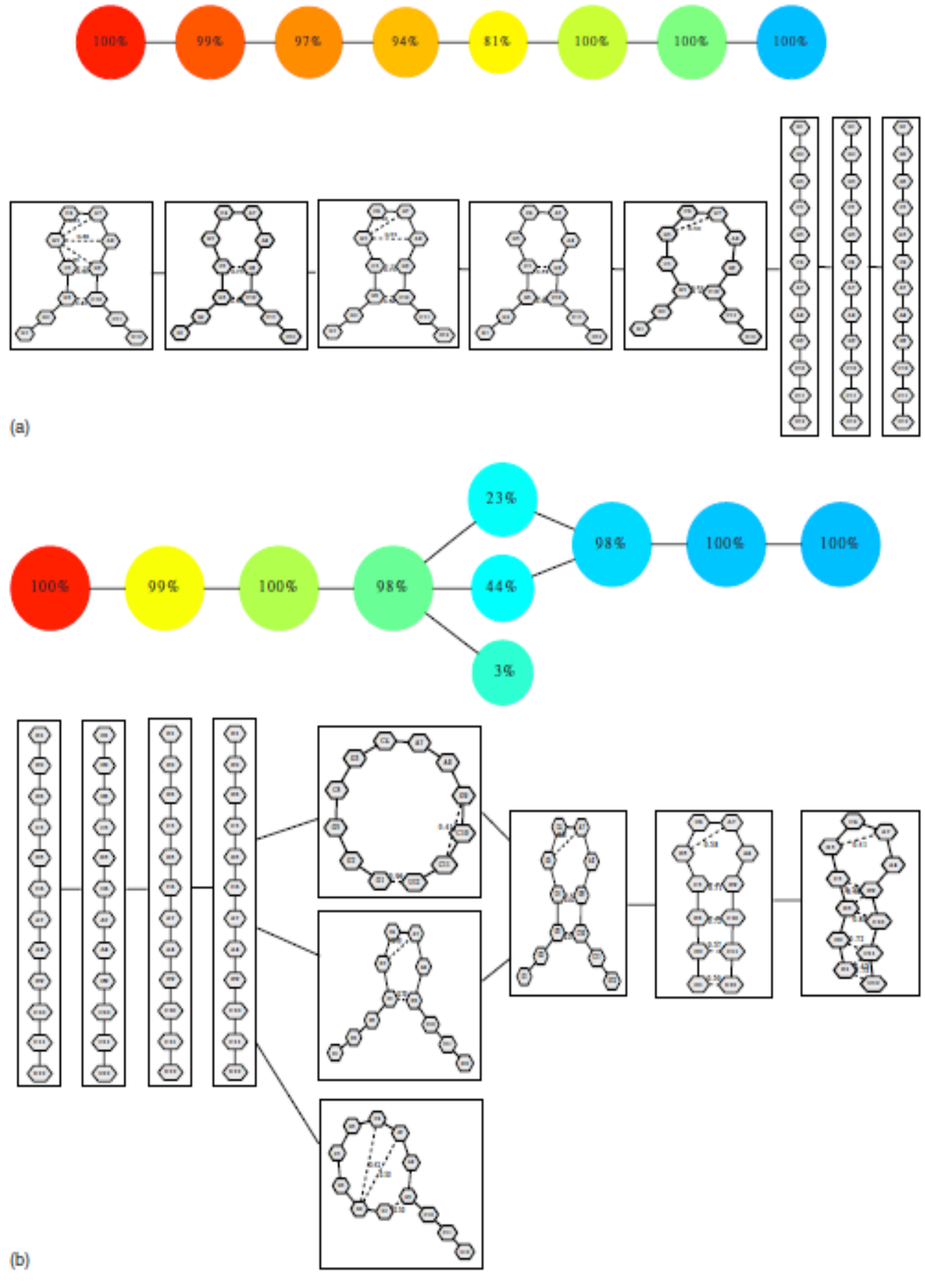
Introduction: topological visualization

[*Topological Methods for Exploring Low-density States in Biomolecular Folding Pathways*, Yao et al., J. Chemical Physics, 2009]

Data: conformations of molecules.

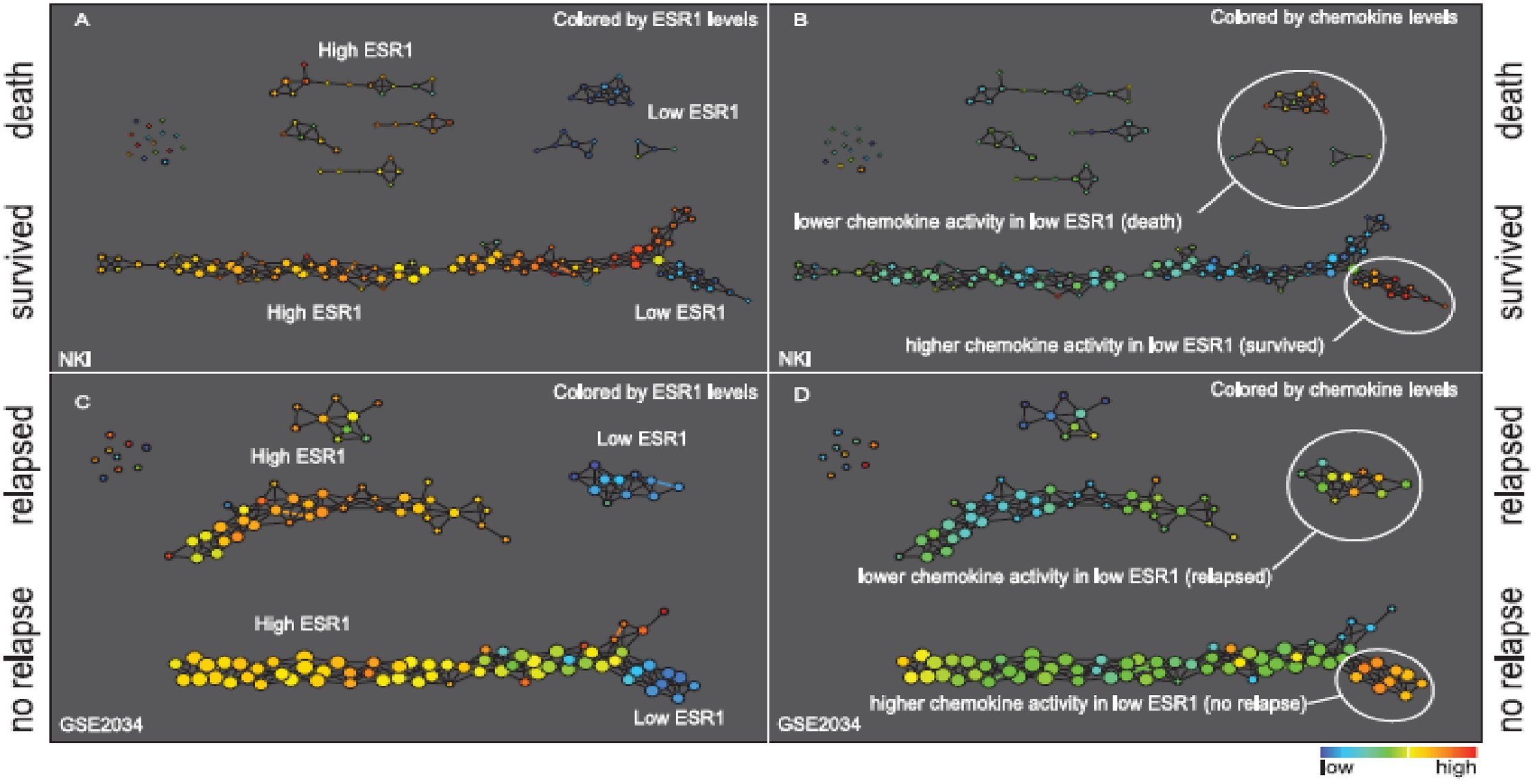
Goal: detect folding pathways.

Idea: 1 loop = 2 pathways.



Introduction: topological visualization

[*Extracting insights from the shape of complex data using topology*, Lum et al., Nature, 2013]



Data: breast cancer patients that went through specific therapy.

Goal: detect variables that influence survival after therapy in breast cancer.

Introduction: topological descriptors built from data

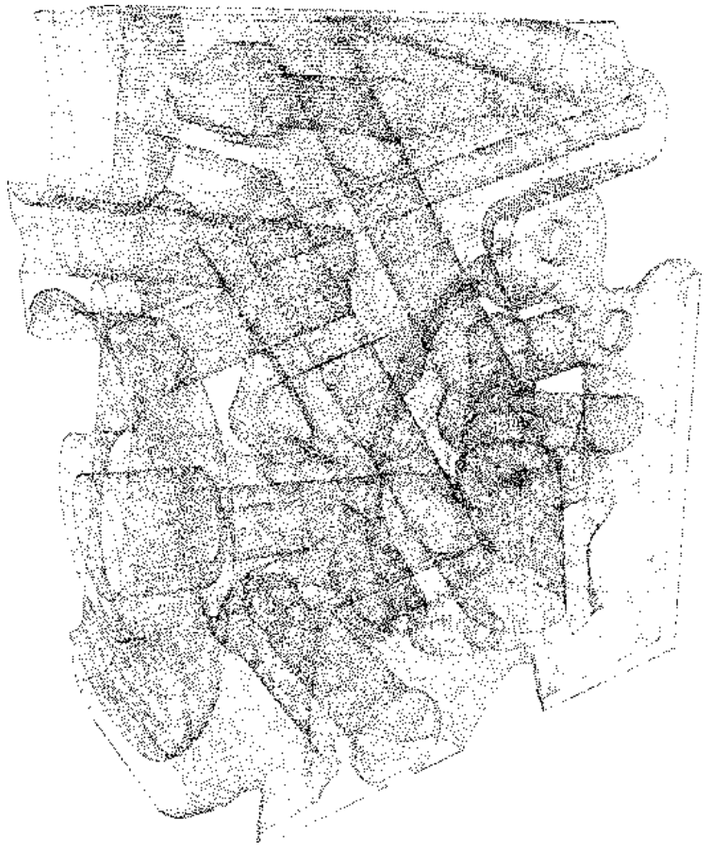
We will see how to build new topological features from data sets...

Introduction: topological descriptors built from data

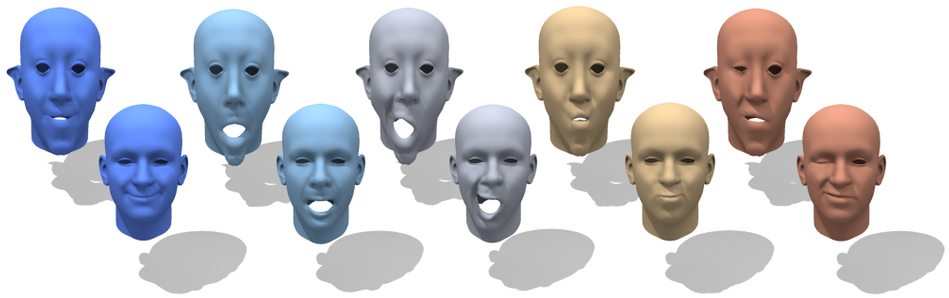
We will see how to build new topological features from data sets...

...but why is that interesting?

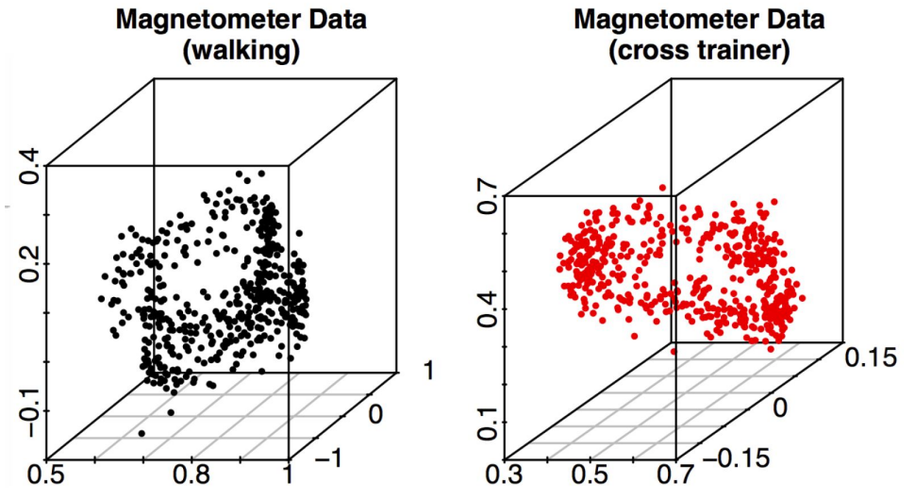
Introduction: topological descriptors built from data



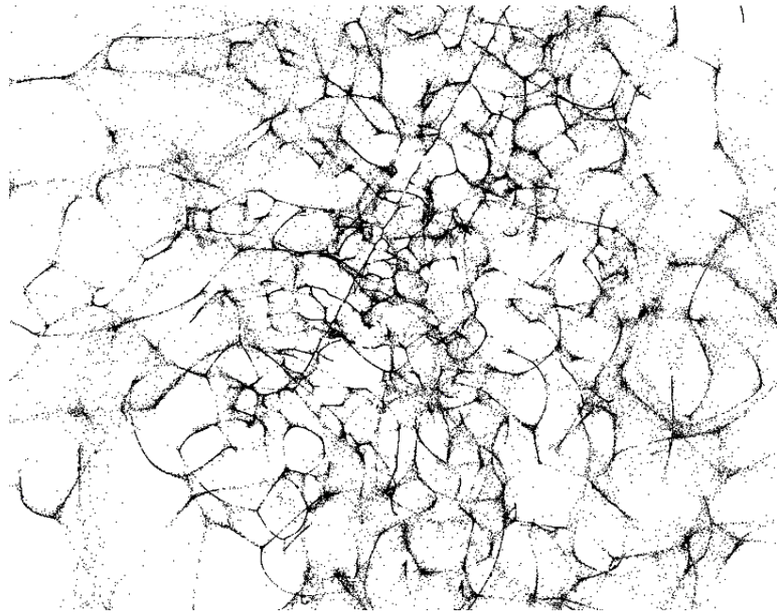
Scans



3D shapes

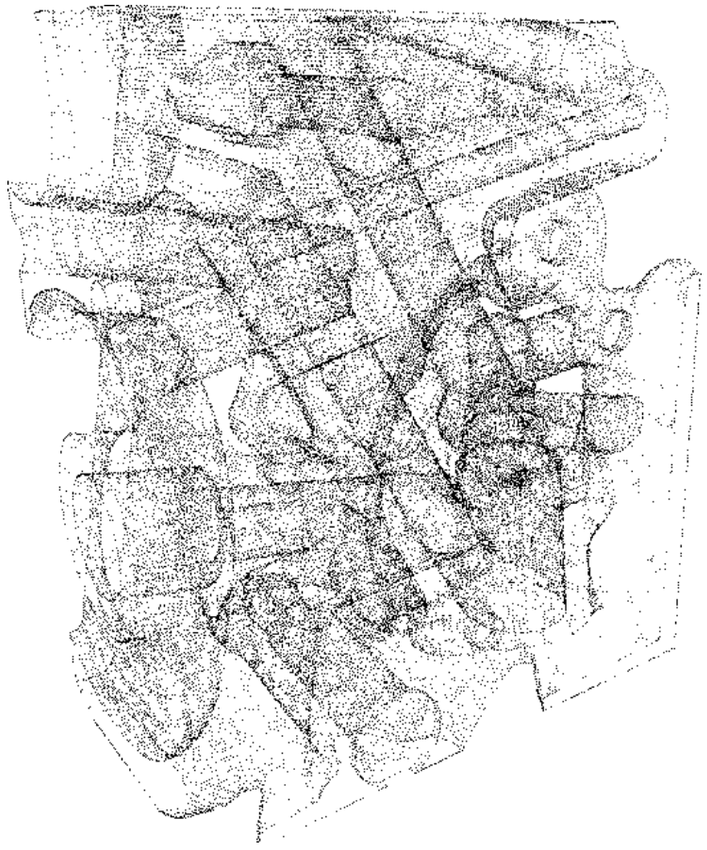


Magnetometer

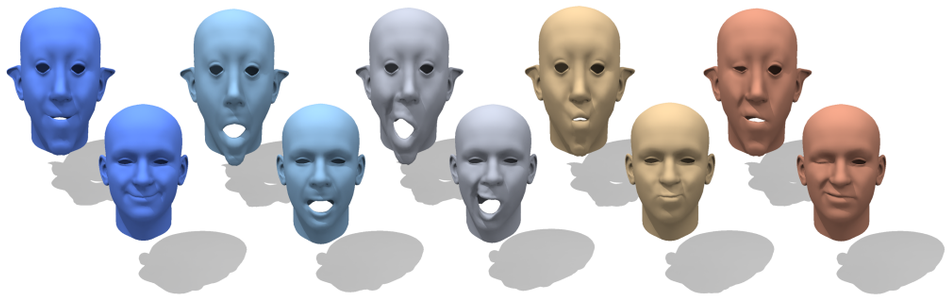


Galaxies

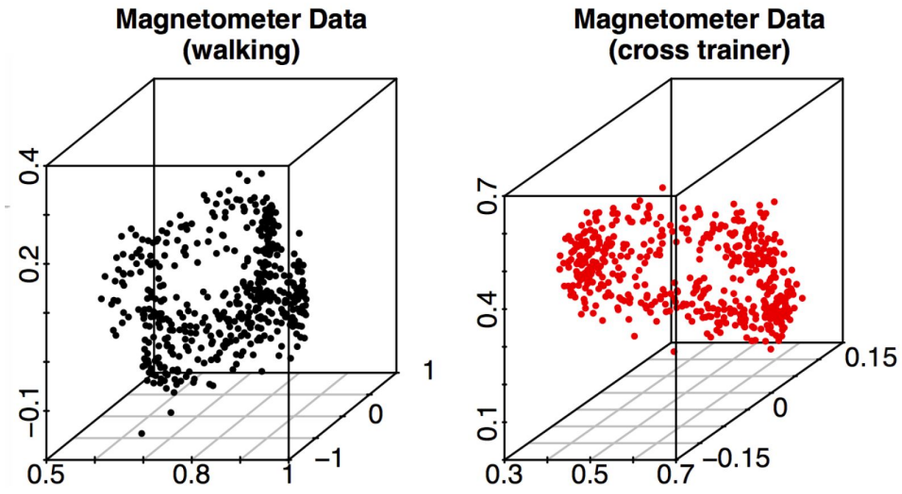
Introduction: topological descriptors built from data



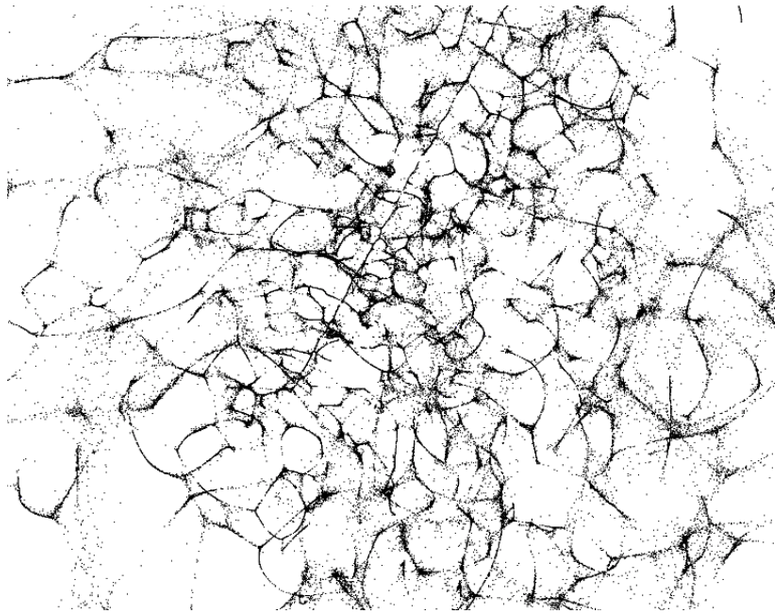
Scans



3D shapes



Magnetometer

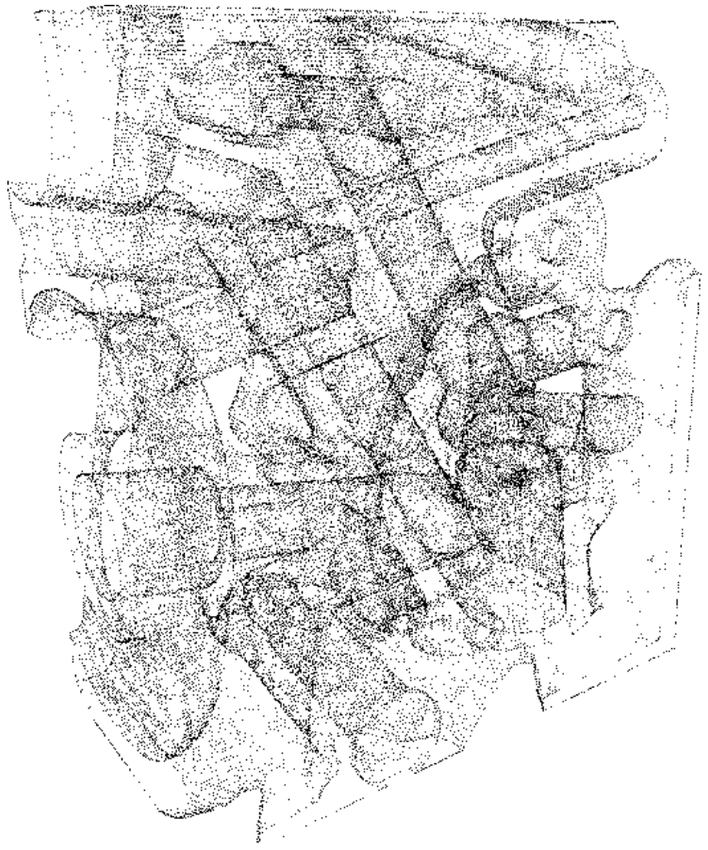


Galaxies

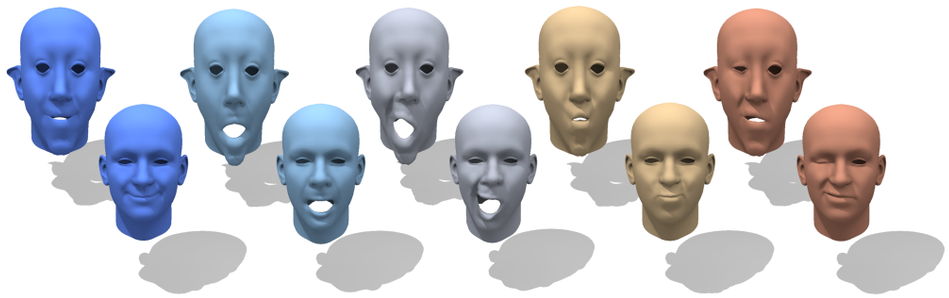
Data often come as (sampling of) metric spaces or sets/spaces endowed with a similarity measure with, possibly complex, topological/geometric structure.

Data carrying geometric information is usually high dimensional.

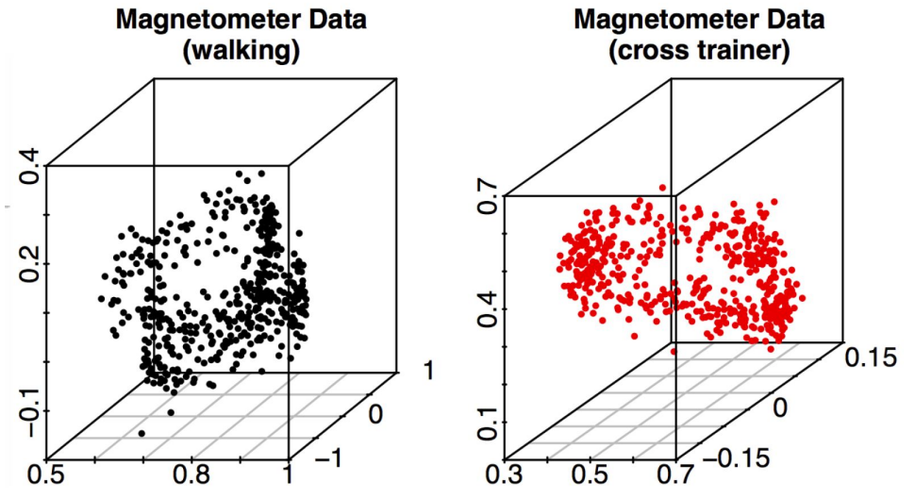
Introduction: topological descriptors built from data



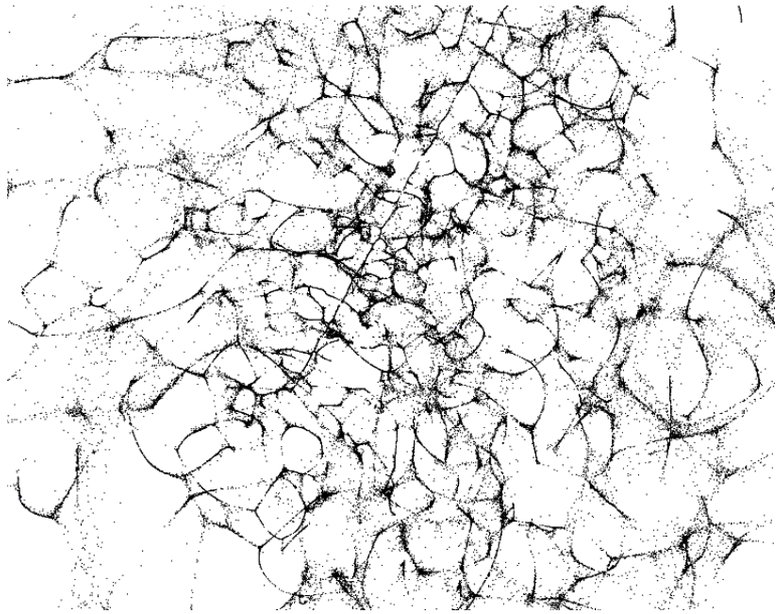
Scans



3D shapes



Magnetometer

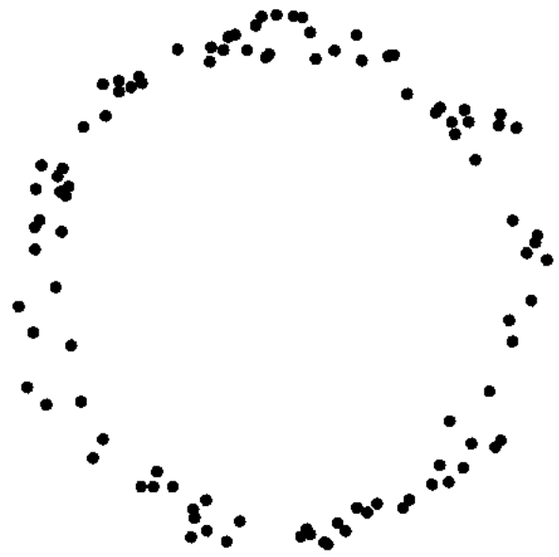


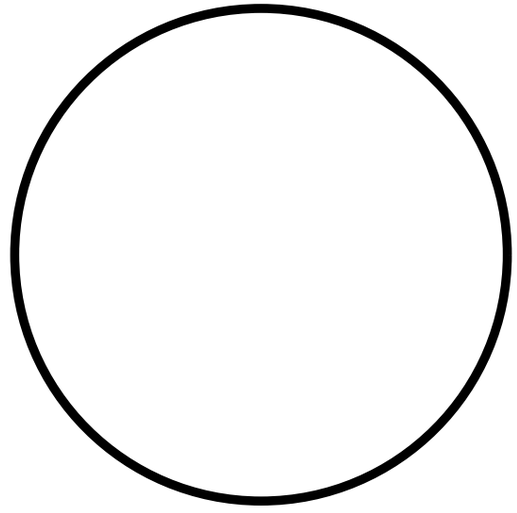
Galaxies

Features from Topological Data Analysis allow to:
- infer relevant topological and geometric features of these spaces.
- take advantage of topol./geom. information for further processing of data (classification, recognition, learning, clustering, parametrization...).

Challenges and advantages

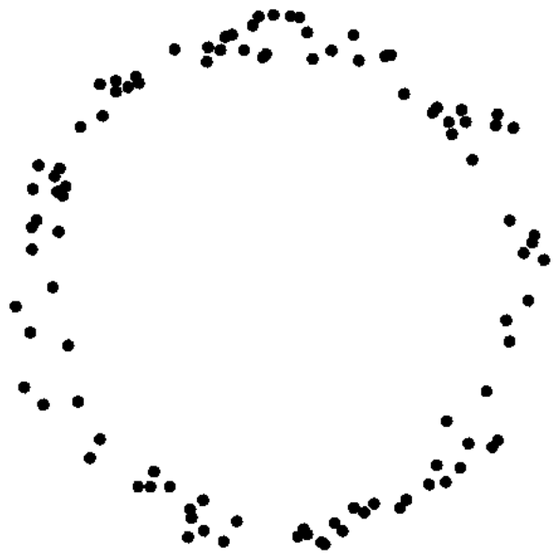
Problem: how to define the *topology* of a data set?

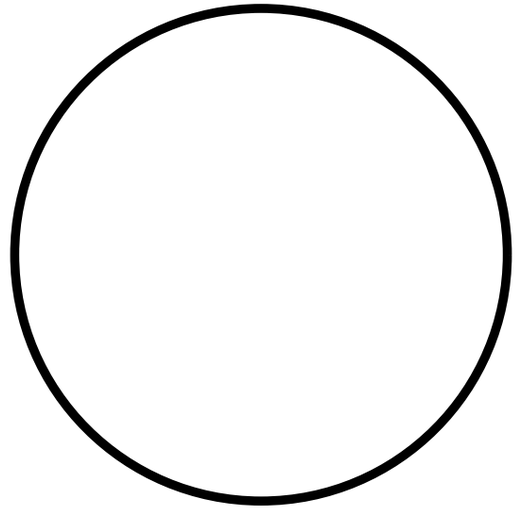




Challenges and advantages

Problem: how to define the *topology* of a data set?





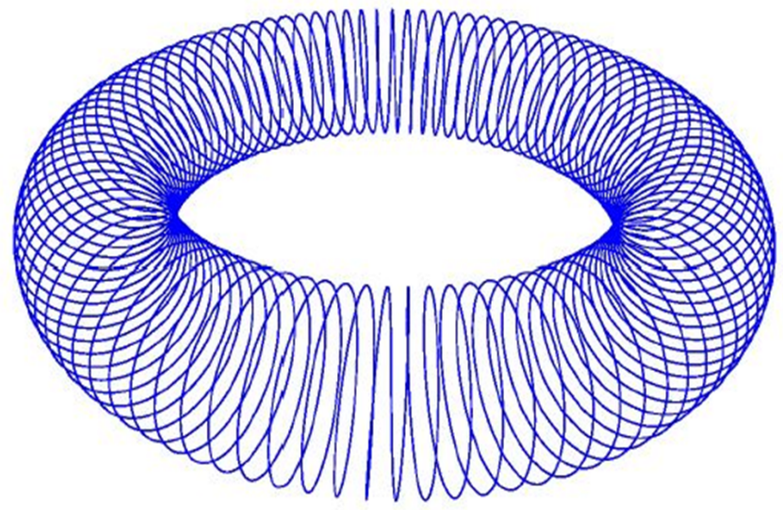
Challenges and goals:

$\rightarrow$ no direct access to topological/geometric information: need of intermediate constructions with *simplicial complexes*;

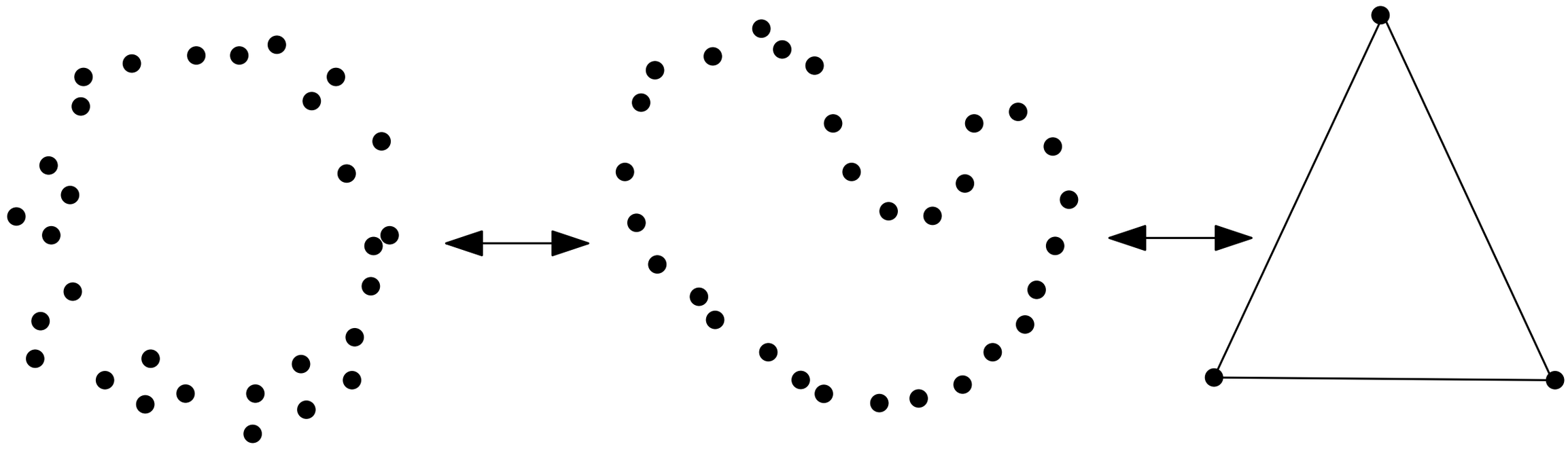
$\rightarrow$ distinguish topological "signal" from noise;

$\rightarrow$ topological information may be multiscale;

$\rightarrow$ statistical analysis of topological information.



Challenges and advantages



Advantages:

→ **coordinate invariance:** topological features/invariants do not rely on any coordinate system ⇒ no need to have data with coordinates, or to embed data in spaces with coordinates... but the metric (distance/similarity between data points) is important.

→ **deformation invariance:** topological features are invariant under homeomorphism and reparameterization.

→ **compressed representation:** topology offers a set of tools to summarize the data in compact ways while preserving its topological structure.

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

Q: What is the most basic brick (space) topology can work on?

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

Q: What is the most basic brick (space) topology can work on?

A: The so-called *topological spaces*.

Def: A *topological space* is a set X equipped with a *topology*, i.e., a family $\mathcal{O}$ of subsets of X, called the *open sets* of X, such that:
(i) the empty set $\emptyset$ and X are elements of $\mathcal{O}$,
(ii) any union of elements of $\mathcal{O}$ is an element of $\mathcal{O}$,
(iii) any finite intersection of elements of $\mathcal{O}$ is an element of $\mathcal{O}$.

Open sets are the tools that allow to define *continuity*, which is the primary notion that allow to compare spaces in topology.

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

Q: What is the most basic brick (space) topology can work on?

A: The so-called *topological spaces*.

Def: A *topological space* is a set X equipped with a *topology*, i.e., a family $\mathcal{O}$ of subsets of X, called the *open sets* of X, such that:
(i) the empty set $\emptyset$ and X are elements of $\mathcal{O}$,
(ii) any union of elements of $\mathcal{O}$ is an element of $\mathcal{O}$,
(iii) any finite intersection of elements of $\mathcal{O}$ is an element of $\mathcal{O}$.

Open sets are the tools that allow to define *continuity*, which is the primary notion that allow to compare spaces in topology.

Def: a map $f : X \to Y$ is *continuous* if and only if the pre-image $f^{-1}(O_Y) = \{x \in X : f(x) \in O_Y\}$ of any open set $O_Y \subseteq Y$ is an open set of X.

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

A very common family of topological spaces is comprised of the *metric spaces*.

Def: A metric (or distance) on X is a map $d : X \times X \to [0, +\infty)$ such that:
(i) for any $x, y \in X$, $d(x, y) = d(y, x)$,
(ii) for any $x, y \in X$, $d(x, y) = 0$ if and only if $x = y$,
(iii) for any $x, y, z \in X$, $d(x, z) \leq d(x, y) + d(y, z)$.
The set X together with d is a metric space.

The smallest topology containing all the open balls $B(x, r) = \{y \in X : d(x, y) < r\}$ is called the metric topology on X induced by d.

Ex: the standard topology in an Euclidean space is the one induced by the metric defined by the norm: $d(x, y) = \|x - y\|$.

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

Def: Here are the main comparison tools of topology:

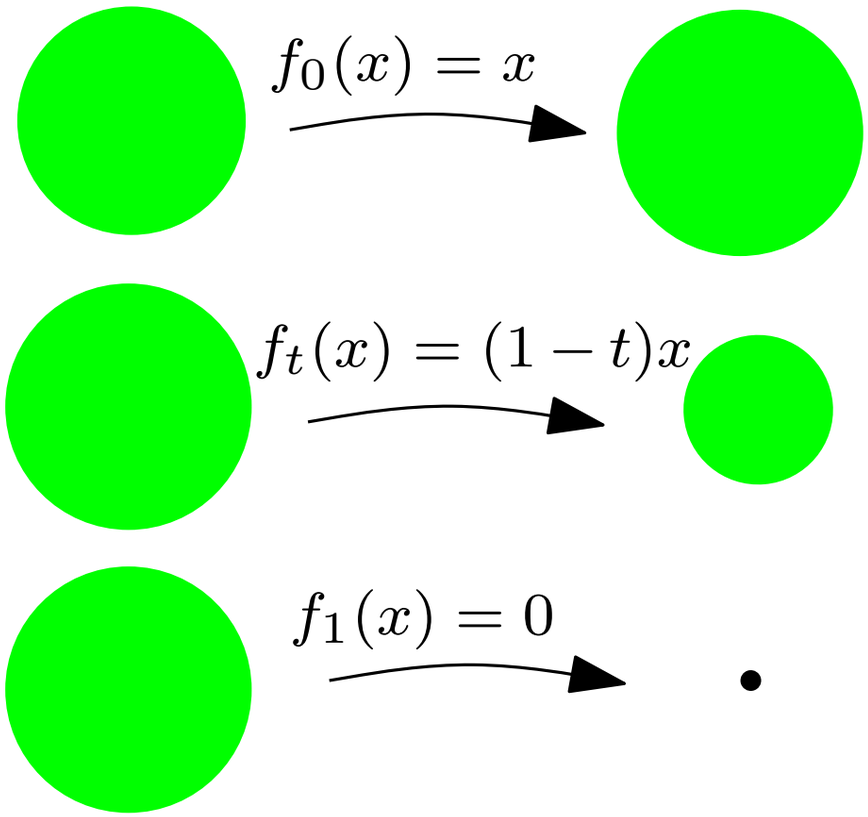
- Two maps $f_0 : X \to Y$ and $f_1 : X \to Y$ are *homotopic* if $\exists$ a continuous map $F : [0,1] \times X \to Y$ s.t. $\forall x \in X$, $F(0,x) = f_0(x)$ and $F_1(1,x) = f_1(x)$. X and Y are homotopy equivalent if $\exists$ continuous maps $f : X \to Y$ and $g : Y \to X$ s.t. $g \circ f$ is homotopic to id_X and $f \circ g$ is homotopic to id_Y.
- X and Y are homeomorphic if $\exists$ a bijection (homeomorphism) $h : X \to Y$ s.t. h and h^{-1} are continuous.
- X and Y are isotopic if $\exists$ a continuous map (isotopy) $F : X \times [0,1] \to Y$ s.t. $F(.,0) = \mathrm{id}_X$, $F(X,1) = Y$ and $\forall t \in [0,1]$, $F(.,t)$ is an homeomorphism.

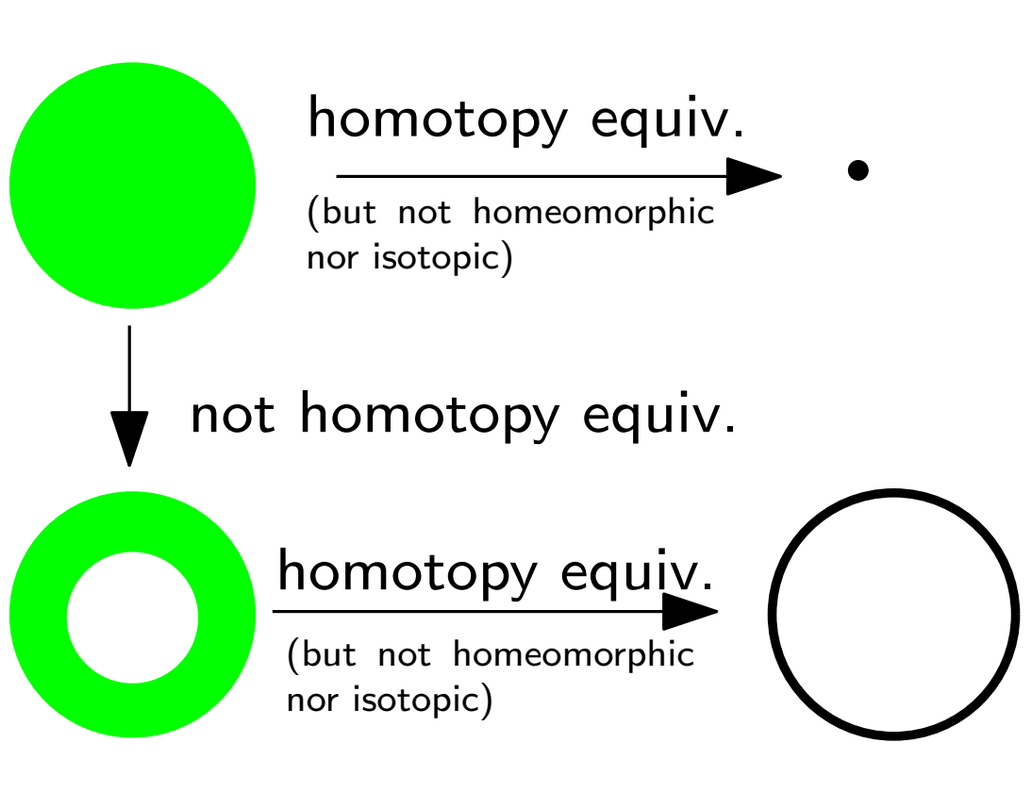
Q: Which notion is stronger/weaker?

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.





A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

Previous examples are particular homotopy equivalences called *deformation retracts*.

Def: If $Y \subseteq X$ and if there exists a continuous map $F : [0,1] \times X \to X$ s.t.:
(i) $\forall x \in X,\ F(0,x) = x$
(ii) $\forall x \in X,\ F(1,x) \in Y$
(iii) $\forall y \in Y,\ \forall t \in [0,1],\ F(t,y) \in Y$
then X and Y are homotopy equivalent. If one replaces condition *(iii)* by $\forall y \in Y,\ \forall t \in [0,1],\ H(t,y) = y$ then H is a deformation retract of X onto Y.

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

Q: Can you find two spaces that are homeomorphic but not isotopic?

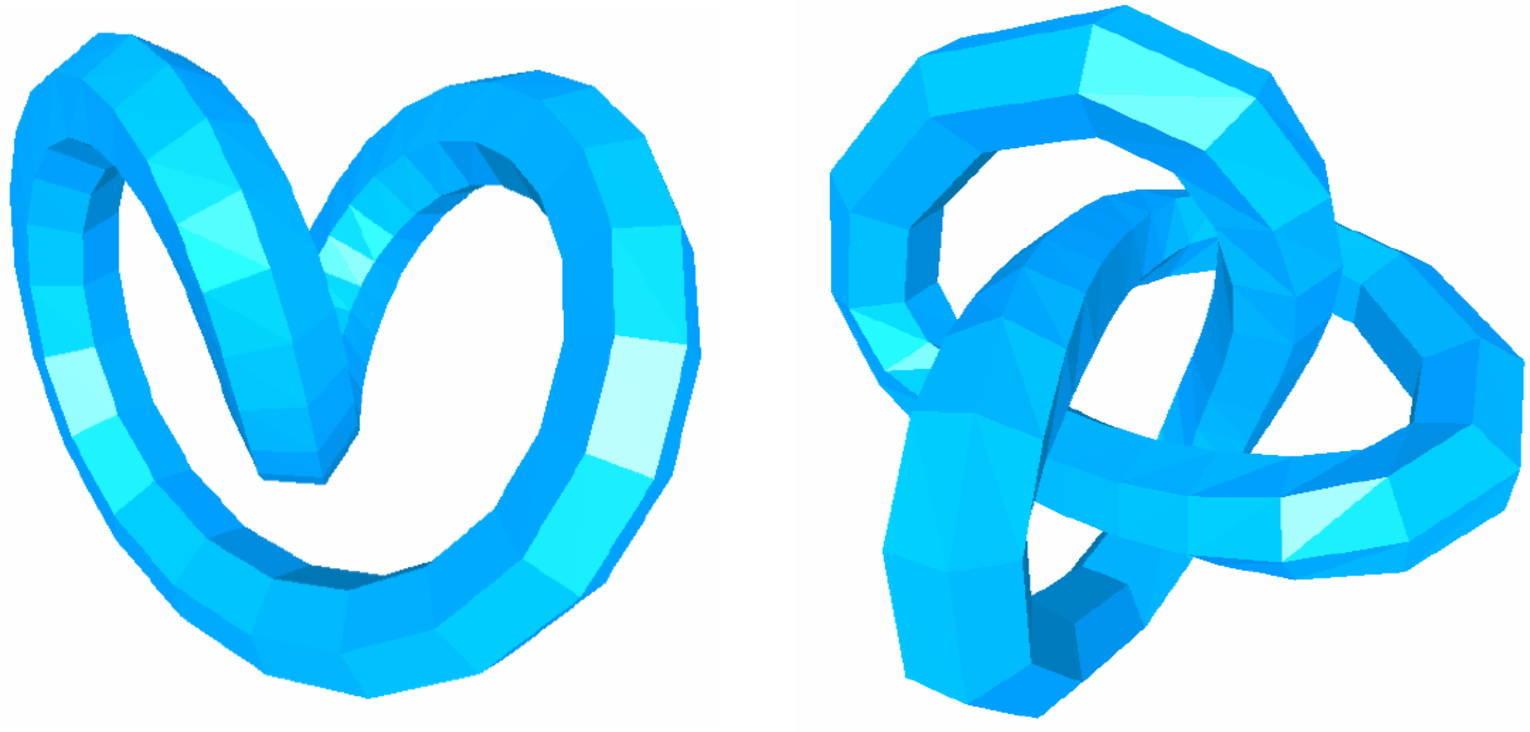
A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

Q: Can you find two spaces that are homeomorphic but not isotopic?

A: Torus and trefoil knot.



A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

Q: Can you find an isotopy between these guys?

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

Pb 1: How to encode topological spaces for computational purposes?

A brief look at topology

Roughly speaking, the goal of topology is to *classify spaces*.

In topology, two spaces are the same (i.e., belong to the same class) if one 'continuously deforms' onto the other.

Pb 1: How to encode topological spaces for computational purposes?

Pb 2: Looking for homotopy equivalences/homeomorphisms/isotopies is extremely difficult. Are there mathematical quantities that are invariant to homotopy equivalences **and** easy to compute?

A topological space fit for computation

Pb 1: How to encode topological spaces for computational purposes?

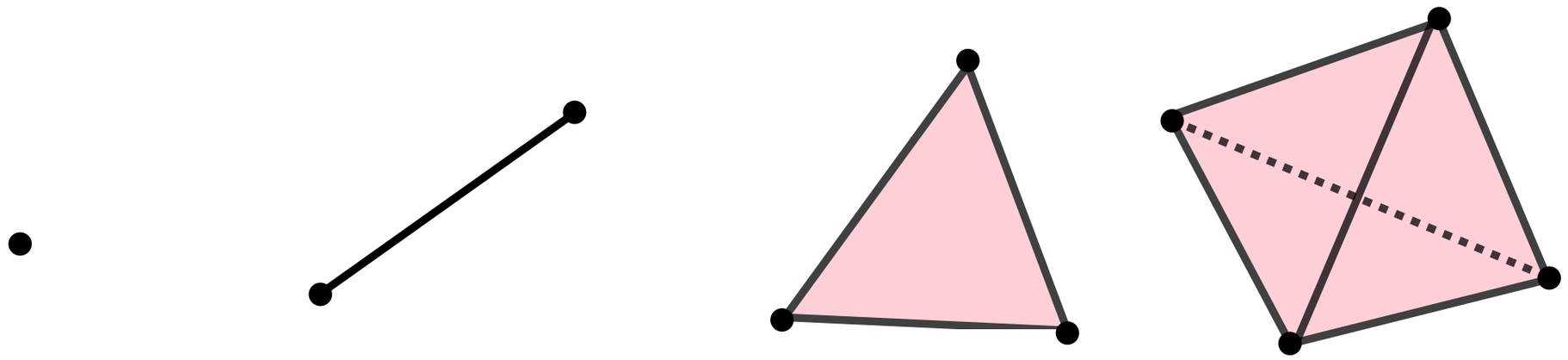
A topological space fit for computation

Pb 1: How to encode topological spaces for computational purposes?

A: Using spaces made of small convex bricks, namely the *simplicial complexes* made of *simplices*.

Simplex and simplicial complex

Simplex and simplicial complex



0-simplex:
vertex

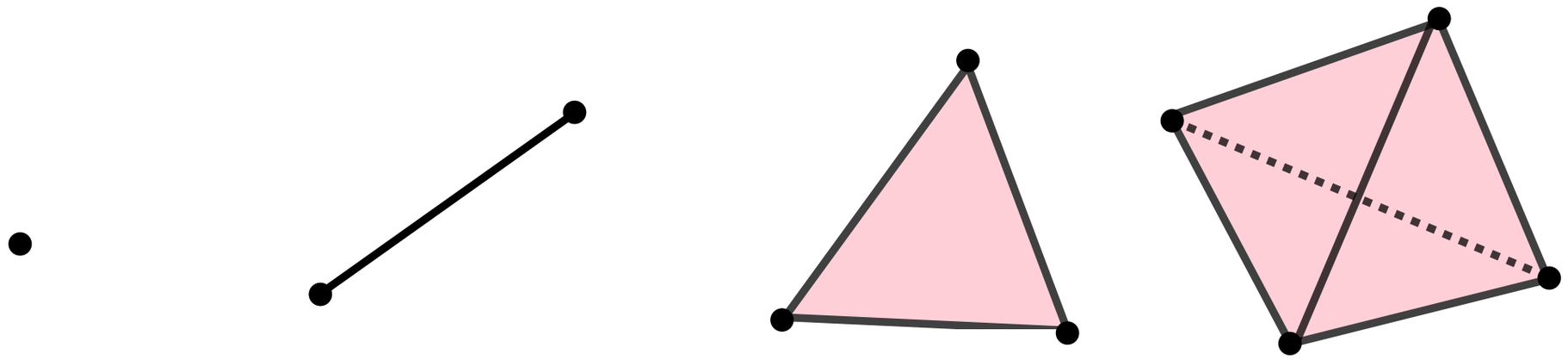
1-simplex:
edge

2-simplex:
triangle

3-simplex:
tetrahedron

etc...

Simplex and simplicial complex



0-simplex: vertex

1-simplex: edge

2-simplex: triangle

3-simplex: tetrahedron

etc...

Def: Given a set $P = \{p_0, \ldots, p_k\} \subset \mathbb{R}^d$ of $k+1$ affinely independent points, the k-dimensional simplex σ (or k-simplex for short) spanned by P is the set of convex combinations

$$\sum_{i=0}^{k} \lambda_i \, p_i, \quad \text{with} \quad \sum_{i=0}^{k} \lambda_i = 1 \quad \text{and} \quad \lambda_i \geq 0.$$

The points $p_0, \ldots, p_k$ are called the vertices of σ.

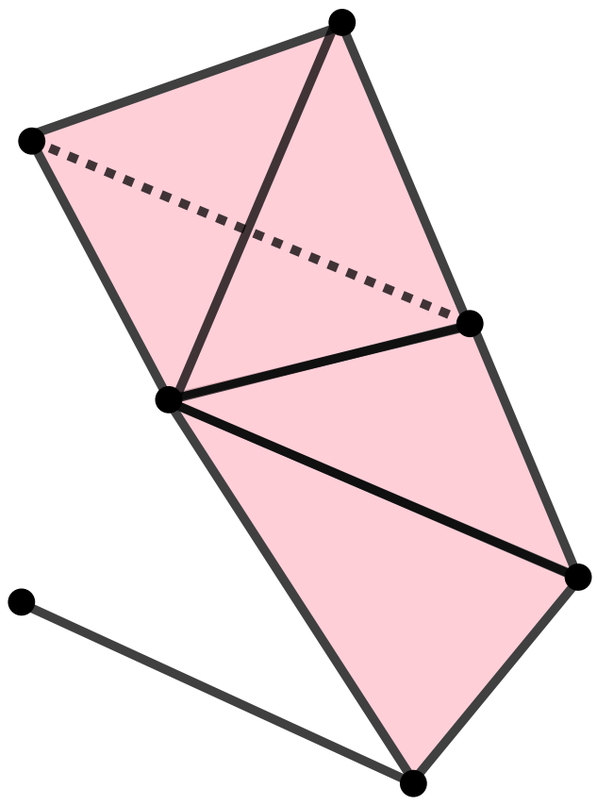
Simplex and simplicial complex

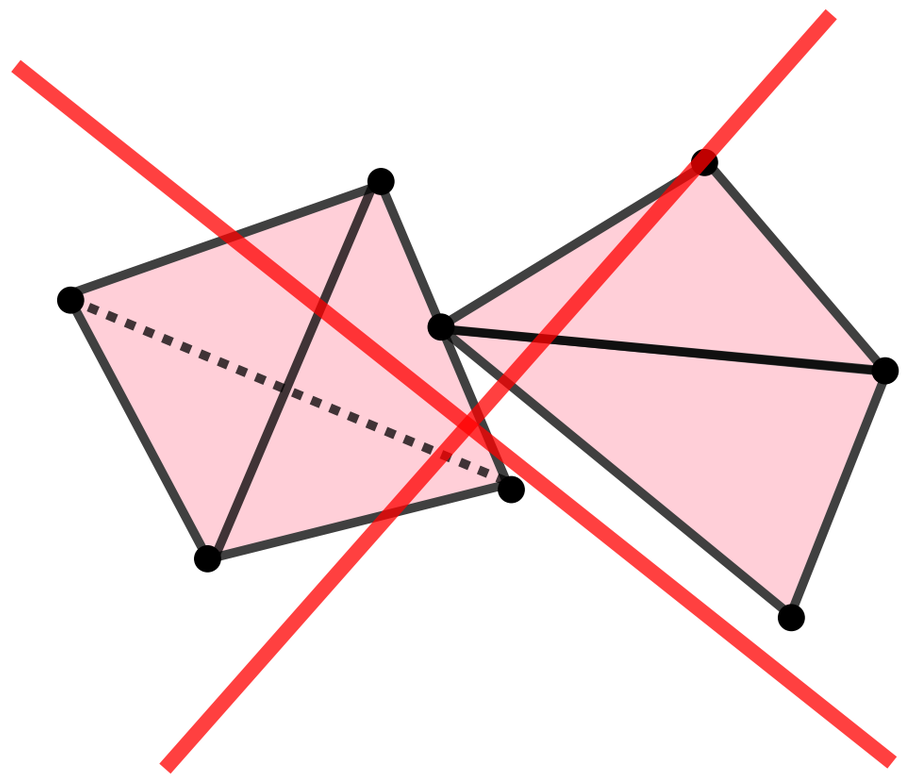
Def: A simplicial complex K in $\mathbb{R}^d$ is a collection of simplices s.t.:

- *(i)* any face of a simplex of K is a simplex of K,
- *(ii)* the intersection of any two simplices of K is either empty or a common face of both.

The underlying space of K (written $|K| \subseteq \mathbb{R}^d$) is the union of its simplices.

Simplex and simplicial complex





Def: A simplicial complex K in $\mathbb{R}^d$ is a collection of simplices s.t.:

- *(i)* any face of a simplex of K is a simplex of K,
- *(ii)* the intersection of any two simplices of K is either empty or a common face of both.

The underlying space of K (written $|K| \subseteq \mathbb{R}^d$) is the union of its simplices.

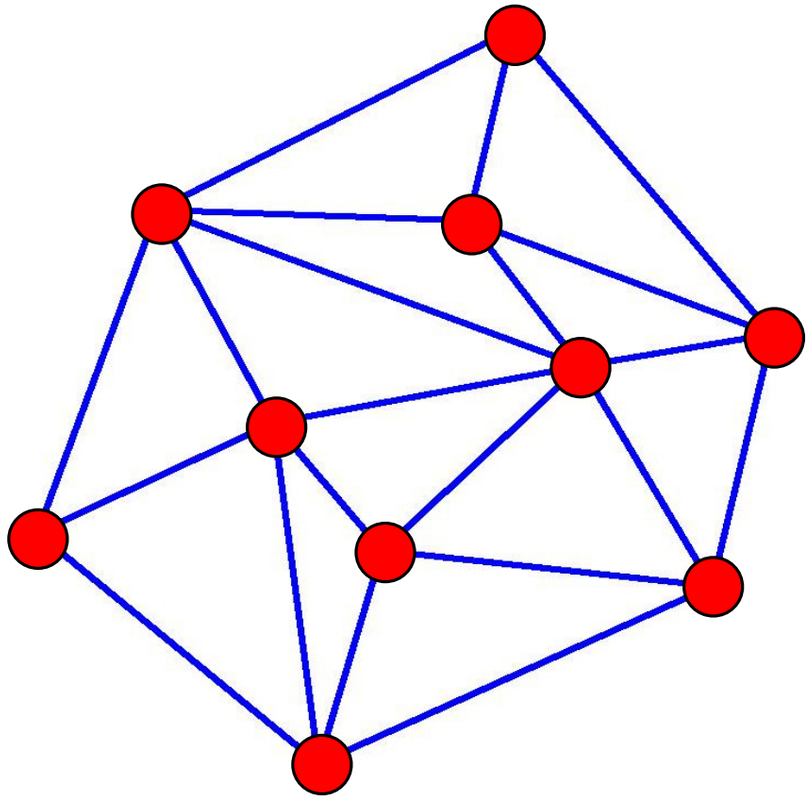
Triangulations

Def: A simplicial complex of dimension d is pure if every simplex is the face of a d-simplex.

Triangulations

Def: A simplicial complex of dimension d is pure if every simplex is the face of a d-simplex.

Def: A triangulation of a point cloud $P \subset \mathbb{R}^d$ is a pure simplicial complex K s.t. $\mathrm{vert}(K) = P$ and $|K| = \mathrm{conv}(P)$.

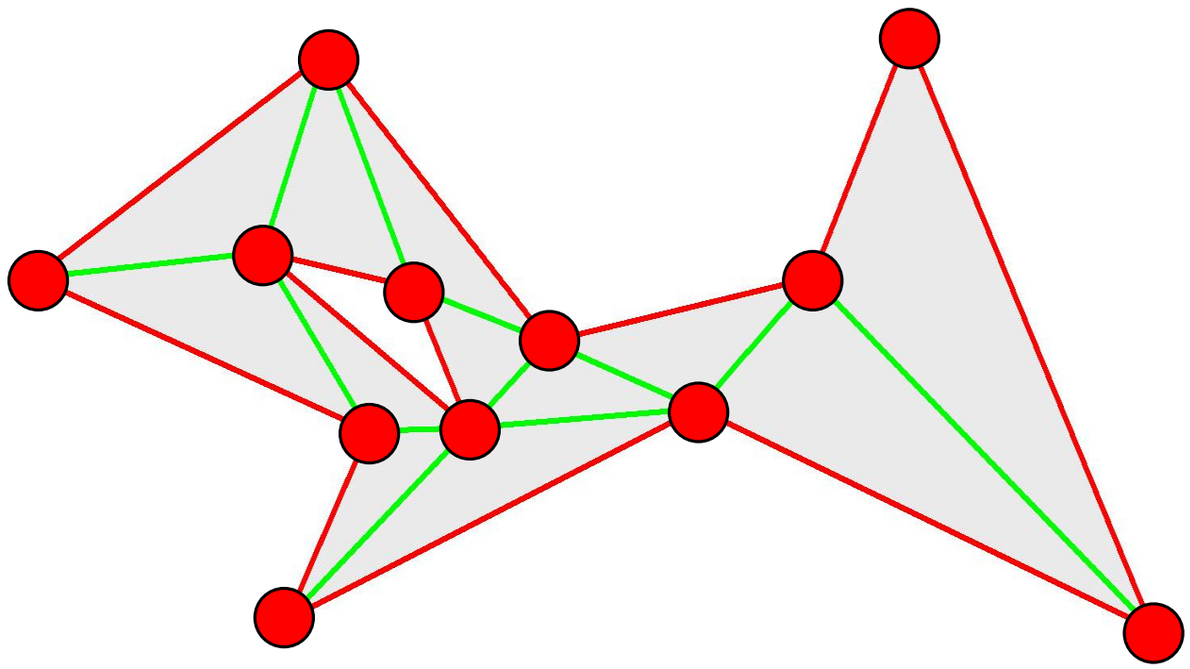


Triangulations

Def: A simplicial complex of dimension d is pure if every simplex is the face of a d-simplex.

Def: A triangulation of a point cloud $P \subset \mathbb{R}^d$ is a pure simplicial complex K s.t. $\mathrm{vert}(K) = P$ and $|K| = \mathrm{conv}(P)$.

Def: A triangulation of a polygonal domain $\Omega \subset \mathbb{R}^d$ is a pure simplicial complex K s.t. $\mathrm{vert}(K) = P$ and $|K| = \Omega$.



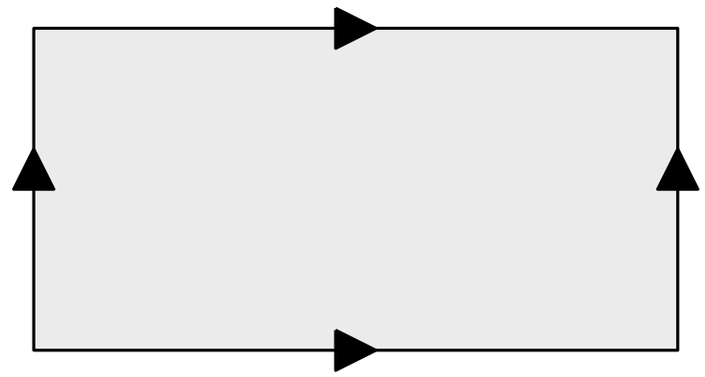
Triangulations

Def: A simplicial complex of dimension d is pure if every simplex is the face of a d-simplex.

Def: A triangulation of a point cloud $P \subset \mathbb{R}^d$ is a pure simplicial complex K s.t. $\mathrm{vert}(K) = P$ and $|K| = \mathrm{conv}(P)$.

Def: A triangulation of a polygonal domain $\Omega \subset \mathbb{R}^d$ is a pure simplicial complex K s.t. $\mathrm{vert}(K) = P$ and $|K| = \Omega$.

Q: Triangulate

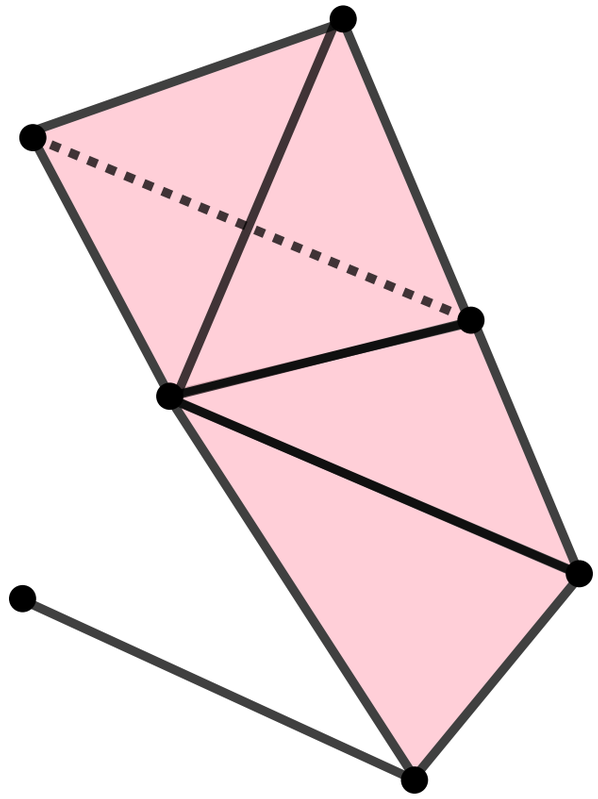


Abstract simplex and simplicial complex

Def: Let $P = \{p_1, \cdots, p_n\}$ be a (finite) set. An abstract simplicial complex K with vertex set P is a set of subsets of P satisfying the two conditions:

- *(i)* the elements of P belong to K,
- *(ii)* if $\tau \in K$ and $\sigma \subseteq \tau$, then $\sigma \in K$.

The elements of K are the simplices.

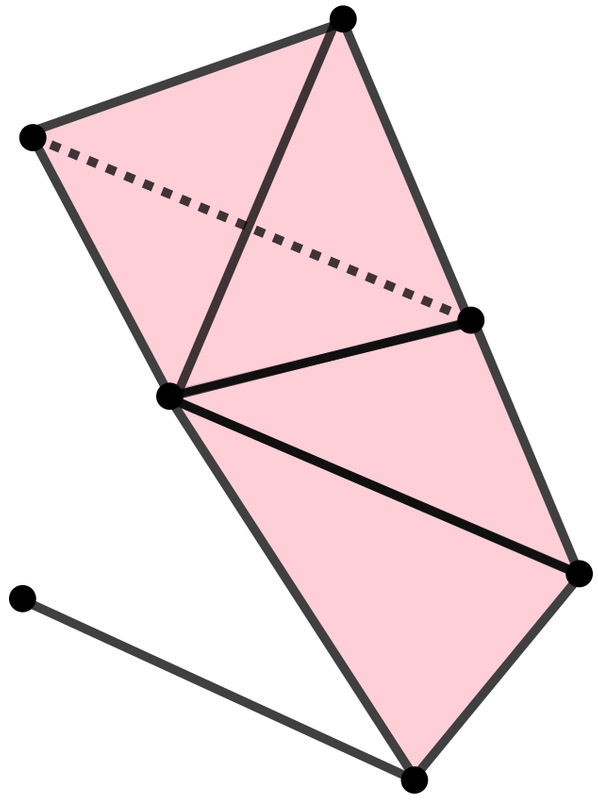


Abstract simplex and simplicial complex

Def: Let $P = \{p_1, \cdots, p_n\}$ be a (finite) set. An abstract simplicial complex K with vertex set P is a set of subsets of P satisfying the two conditions:

- *(i)* the elements of P belong to K,
- *(ii)* if $\tau \in K$ and $\sigma \subseteq \tau$, then $\sigma \in K$.

The elements of K are the simplices.



IMPORTANT

Simplicial complexes can be seen at the same time as geometric/topological spaces (good for topological/geometrical inference) and as combinatorial objects (abstract simplicial complexes, good for computations).

Abstract simplex and simplicial complex

Def: A realization of an abstract simplicial complex K is a geometric simplicial complex K' who is isomorphic to K, i.e., there exists a bijection

$$f : \mathrm{vert}(K) \to \mathrm{vert}(K'),$$

such that $\sigma \in K \iff f(\sigma) \in K'$.

Abstract simplex and simplicial complex

Def: A realization of an abstract simplicial complex K is a geometric simplicial complex K' who is isomorphic to K, i.e., there exists a bijection

$$f : \mathrm{vert}(K) \to \mathrm{vert}(K'),$$

such that $\sigma \in K \iff f(\sigma) \in K'$.

Any abstract simplicial complex with n vertices can be realized in $\mathbb{R}^n$.

Q: Prove it.

Abstract simplex and simplicial complex

Def: A realization of an abstract simplicial complex K is a geometric simplicial complex K' who is isomorphic to K, i.e., there exists a bijection

$$f : \mathrm{vert}(K) \to \mathrm{vert}(K'),$$

such that $\sigma \in K \iff f(\sigma) \in K'$.

Any abstract simplicial complex with n vertices can be realized in $\mathbb{R}^n$.

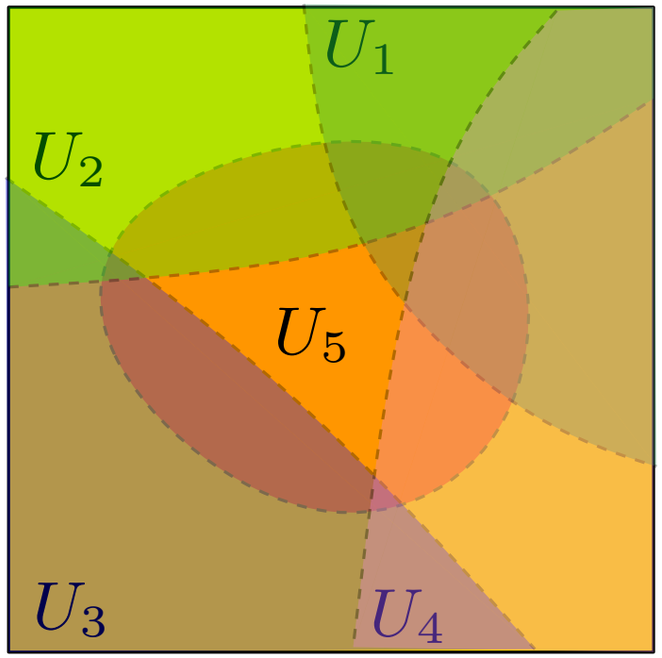
Q: Prove it.

Abstract simplicial complexes and their realizations are *homeomorphic*.

Nerve complex

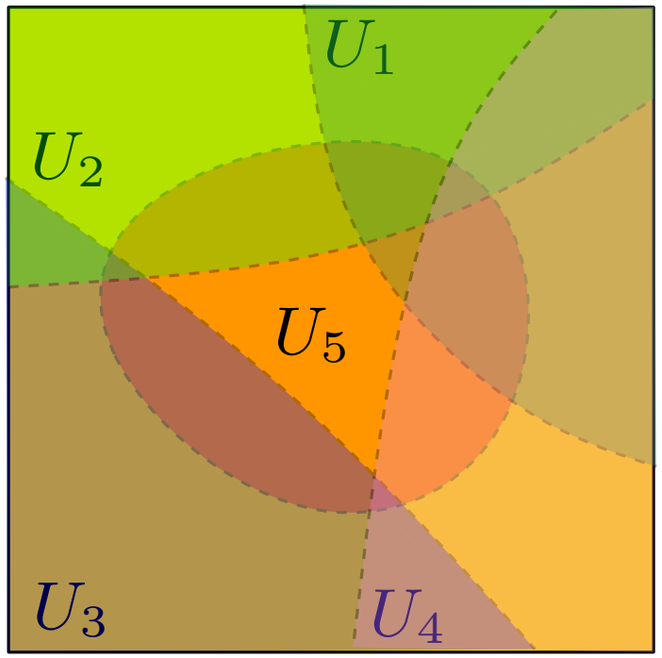
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

Nerve complex



Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

Nerve complex

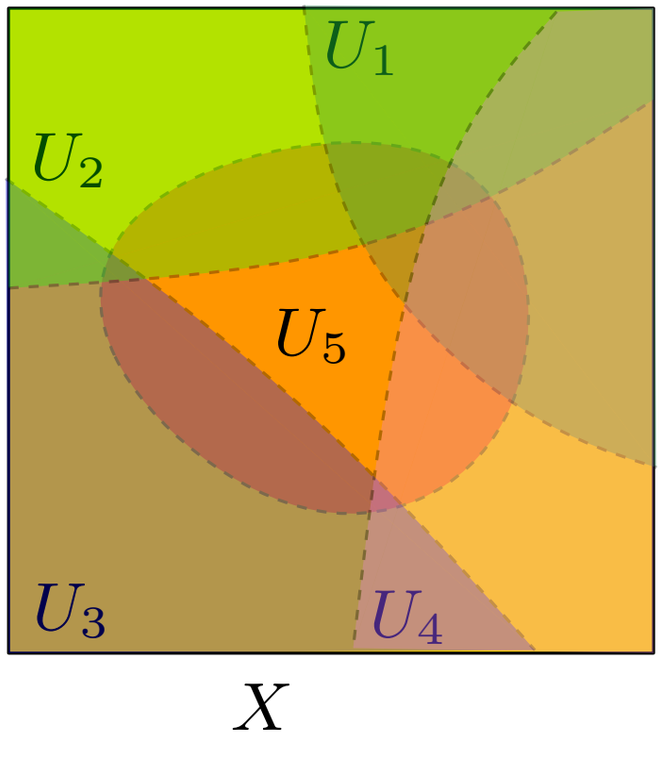


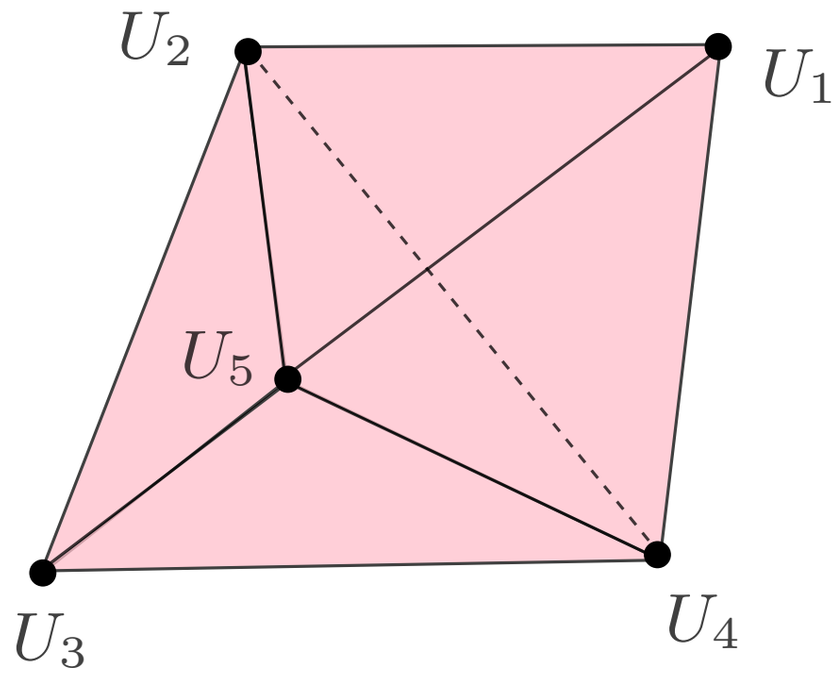
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

Def: Given a cover of a topological space X, $\mathcal{U} = (U_i)_{i \in I}$, its nerve is the abstract simplicial complex $C(\mathcal{U})$ whose vertex set is $\mathcal{U}$ and s.t.

$$\sigma = [U_{i_0}, U_{i_1}, \ldots, U_{i_k}] \in C(\mathcal{U}) \text{ if and only if } \cap_{j=0}^{k} U_{i_j} \neq \emptyset.$$

Nerve complex





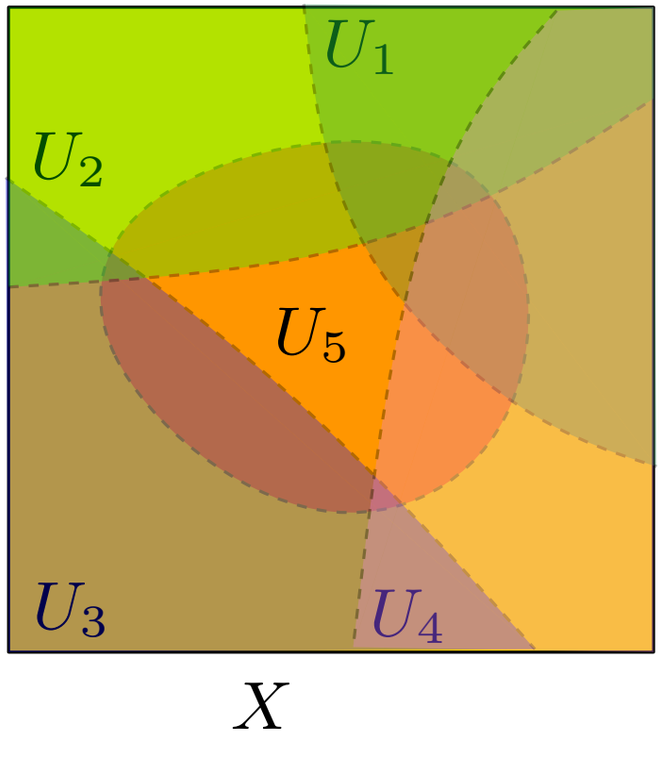
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

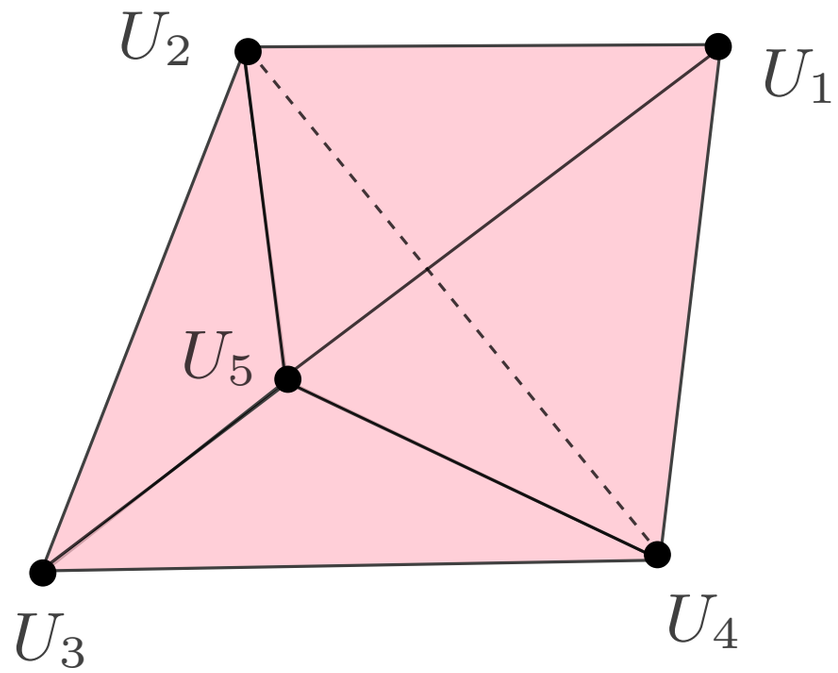
Def: Given a cover of a topological space X, $\mathcal{U} = (U_i)_{i \in I}$, its nerve is the abstract simplicial complex $C(\mathcal{U})$ whose vertex set is $\mathcal{U}$ and s.t.

$$\sigma = [U_{i_0}, U_{i_1}, \ldots, U_{i_k}] \in C(\mathcal{U}) \text{ if and only if } \cap_{j=0}^{k} U_{i_j} \neq \emptyset.$$

Nerve complex

[*On the imbedding of systems of compacta in simplicial complexes*, Borsuk, Fund. Math., 1948]





The Nerve Theorem: Let $\mathcal{U} = (U_i)_{i \in I}$ be a finite open cover of a subset X of $\mathbb{R}^d$ such that any intersection of the U_i's is either empty or contractible. Then X and $C(\mathcal{U})$ are homotopy equivalent.

In particular, every convex set is contractible.

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

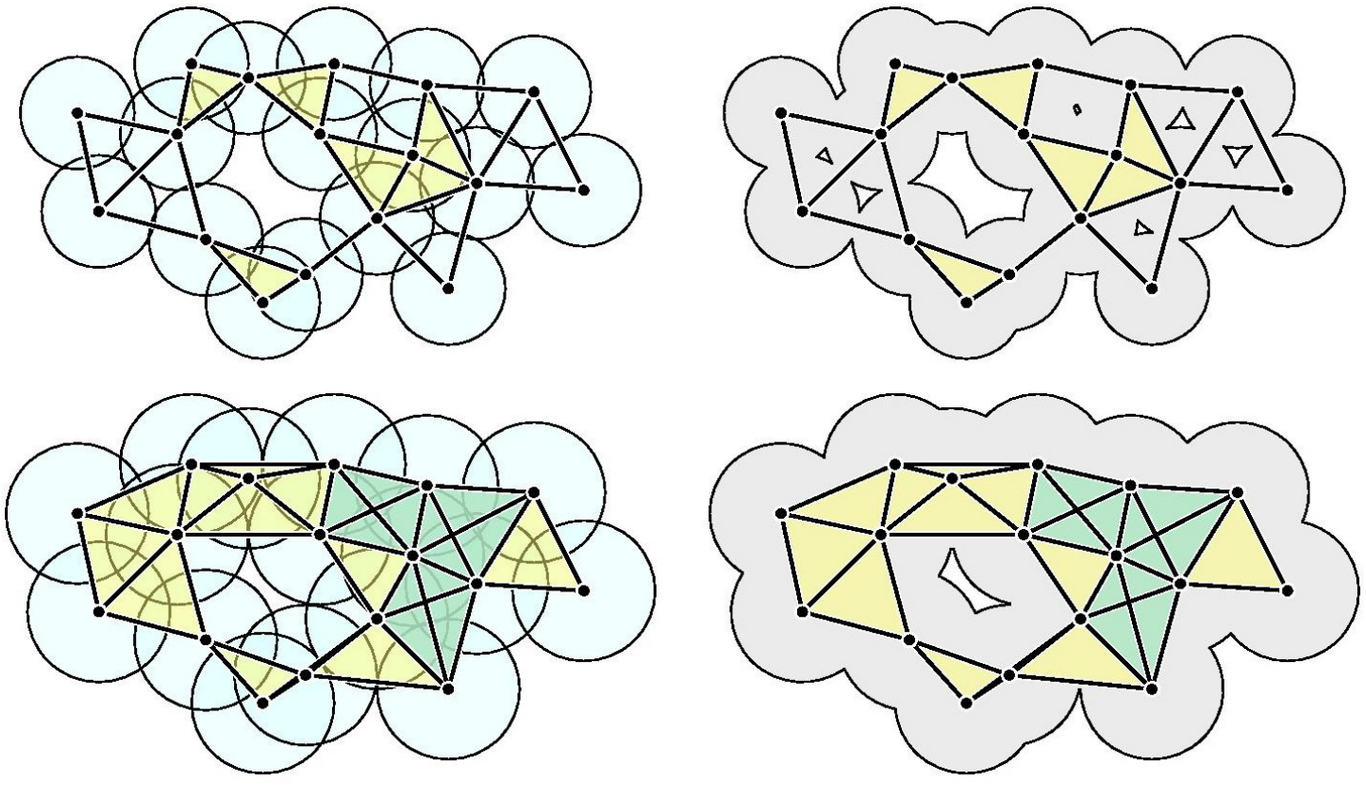
$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Q: Does the Nerve Theorem apply to Čech complexes?



Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\mathrm{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \text{ iif } \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

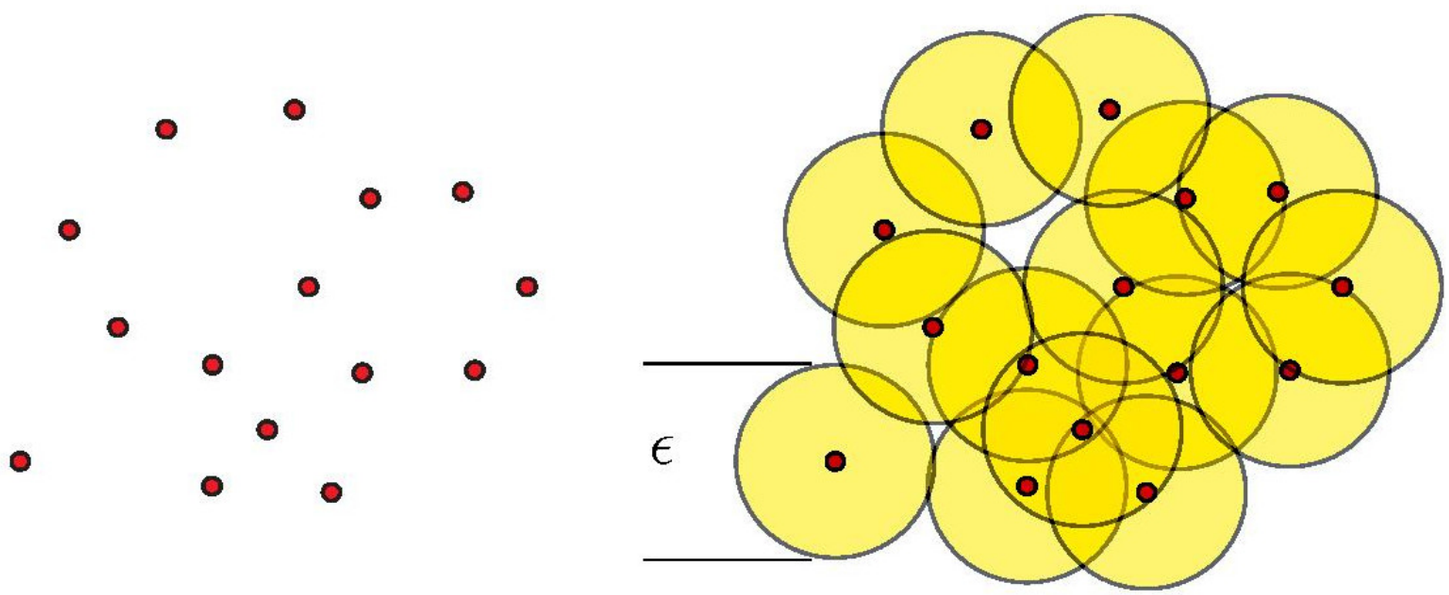
Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Rips complex of radius $r > 0$ is the abstract simplicial complex $R(P, r)$ s.t. $\mathrm{vert}(R(P, r)) = P$ and

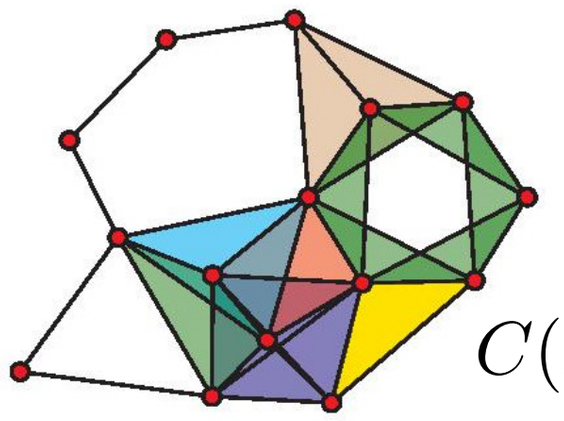
$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in R(P, r) \text{ iif } \|P_{i_j} - P_{i_{j'}}\| \leq 2r, \forall 1 \leq j, j' \leq k.$$

Čech and (Vietoris)-Rips complexes

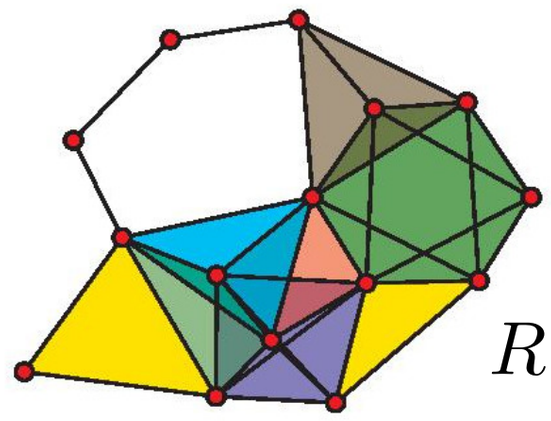
Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\text{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$





$C(P, \epsilon)$



$R(P, \epsilon)$

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\text{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \text{ iif } \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Rips complex of radius $r > 0$ is the abstract simplicial complex $R(P, r)$ s.t. $\text{vert}(R(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in R(P, r) \text{ iif } \|P_{i_j} - P_{i_{j'}}\| \leq 2r, \forall 1 \leq j, j' \leq k.$$

Good news is that Rips and Čech complexes are related:

Čech and (Vietoris)-Rips complexes

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\text{vert}(C(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \text{ iif } \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

Pbm: Čech complexes can be quite hard to compute.

Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Rips complex of radius $r > 0$ is the abstract simplicial complex $R(P, r)$ s.t. $\text{vert}(R(P, r)) = P$ and

$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in R(P, r) \text{ iif } \|P_{i_j} - P_{i_{j'}}\| \leq 2r, \forall 1 \leq j, j' \leq k.$$

Good news is that Rips and Čech complexes are related:

Prop: $R(P, r/2) \subseteq C(P, r) \subseteq R(P, r)$.

Q: Prove it.

Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

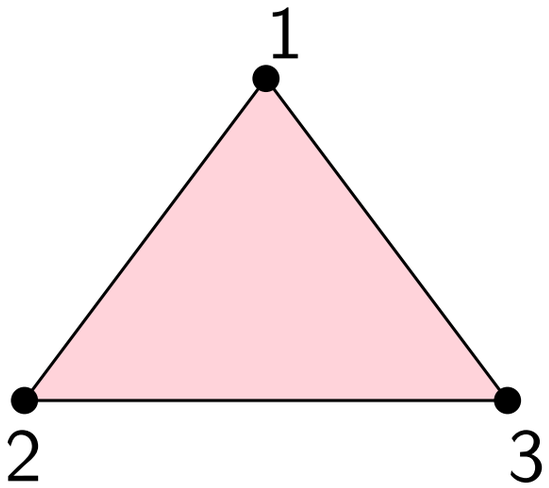
We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

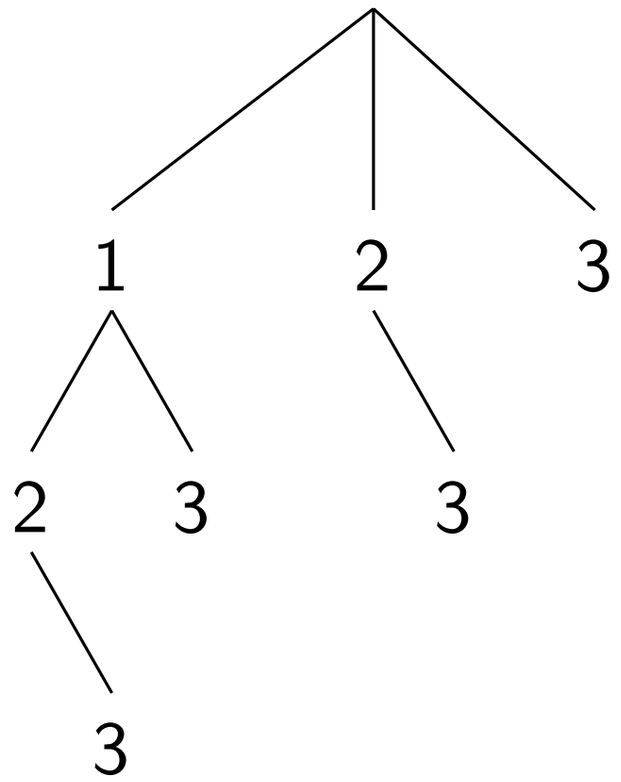
Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Idea: store sorted simplices in a prefix tree (also called *trie)*.



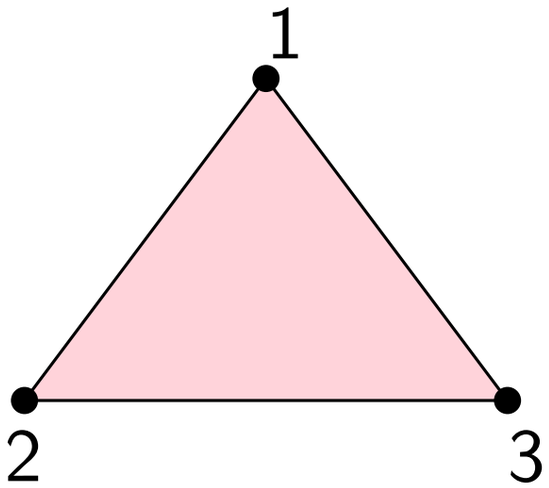


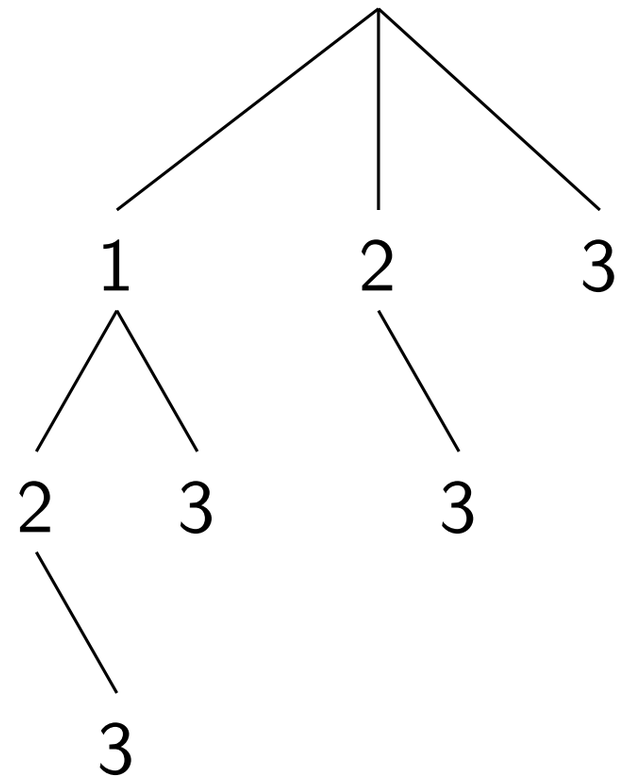
Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Idea: store sorted simplices in a prefix tree (also called *trie)*.





This is called the *simplex tree*.

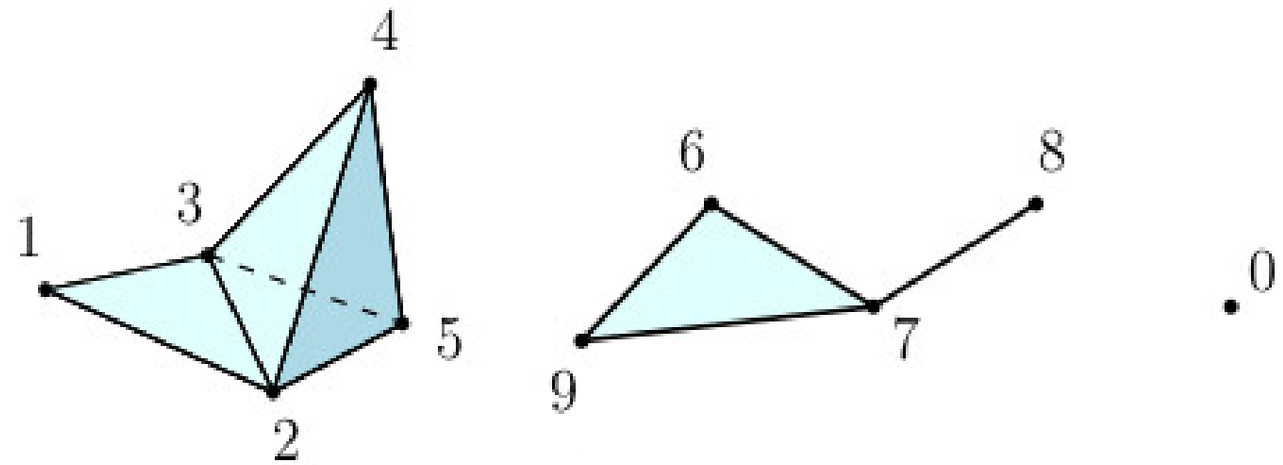
It allows to store all simplices explicitly without storing all adjacency relations, while maintaining low complexity for basic operations.

Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Q: build the simplex tree of

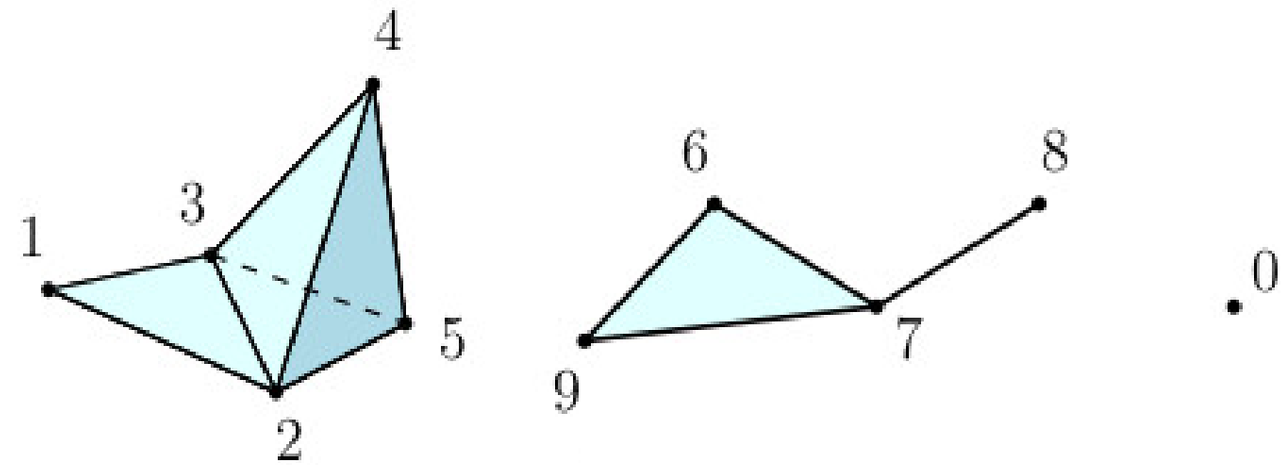


Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Q: build the simplex tree of



Number of nodes in simplex tree = number of simplices

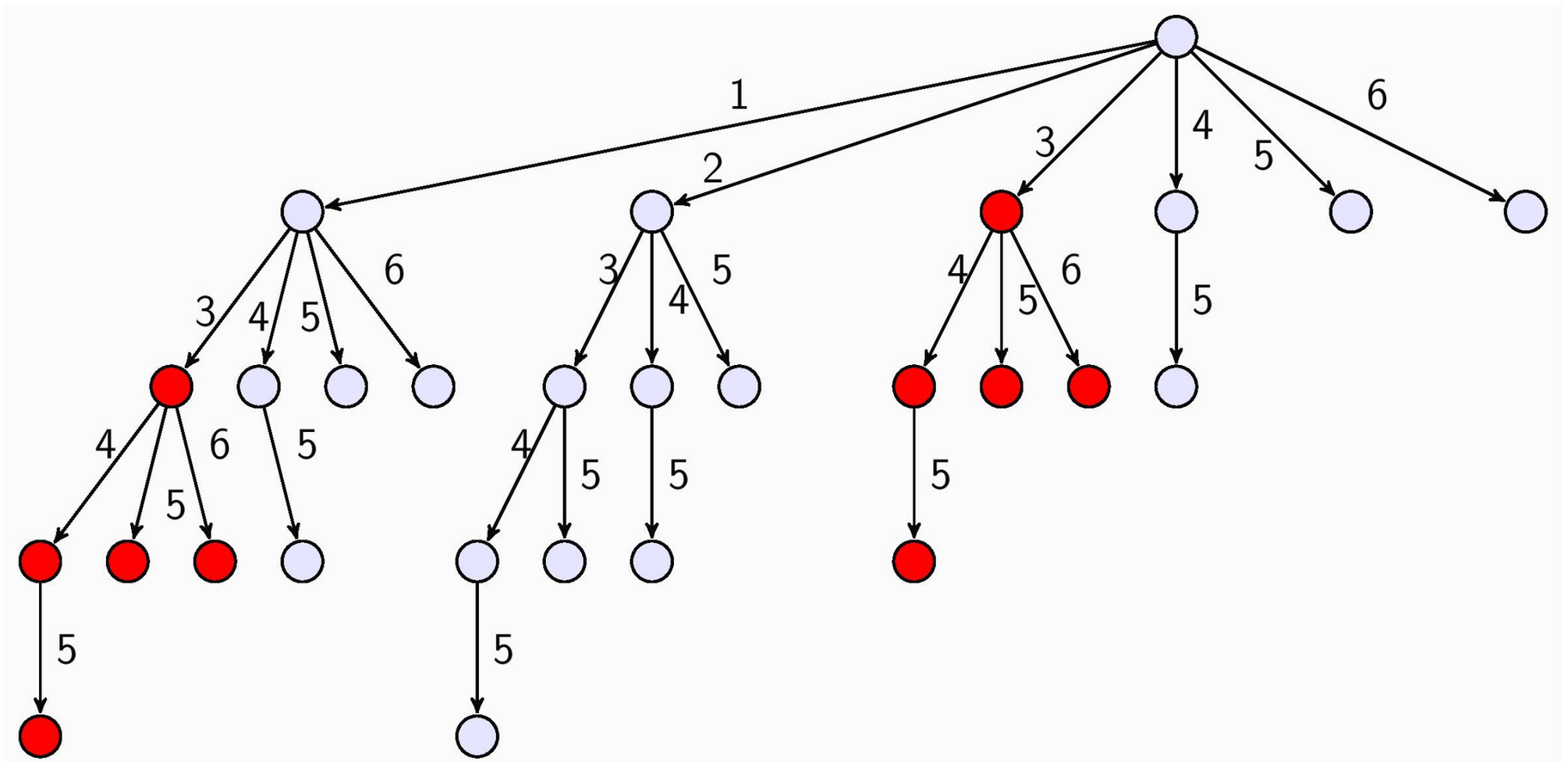
Depth of simplex tree = 1 + dimension of complex

Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Unfortunately, the simplex tree also has redundancies.

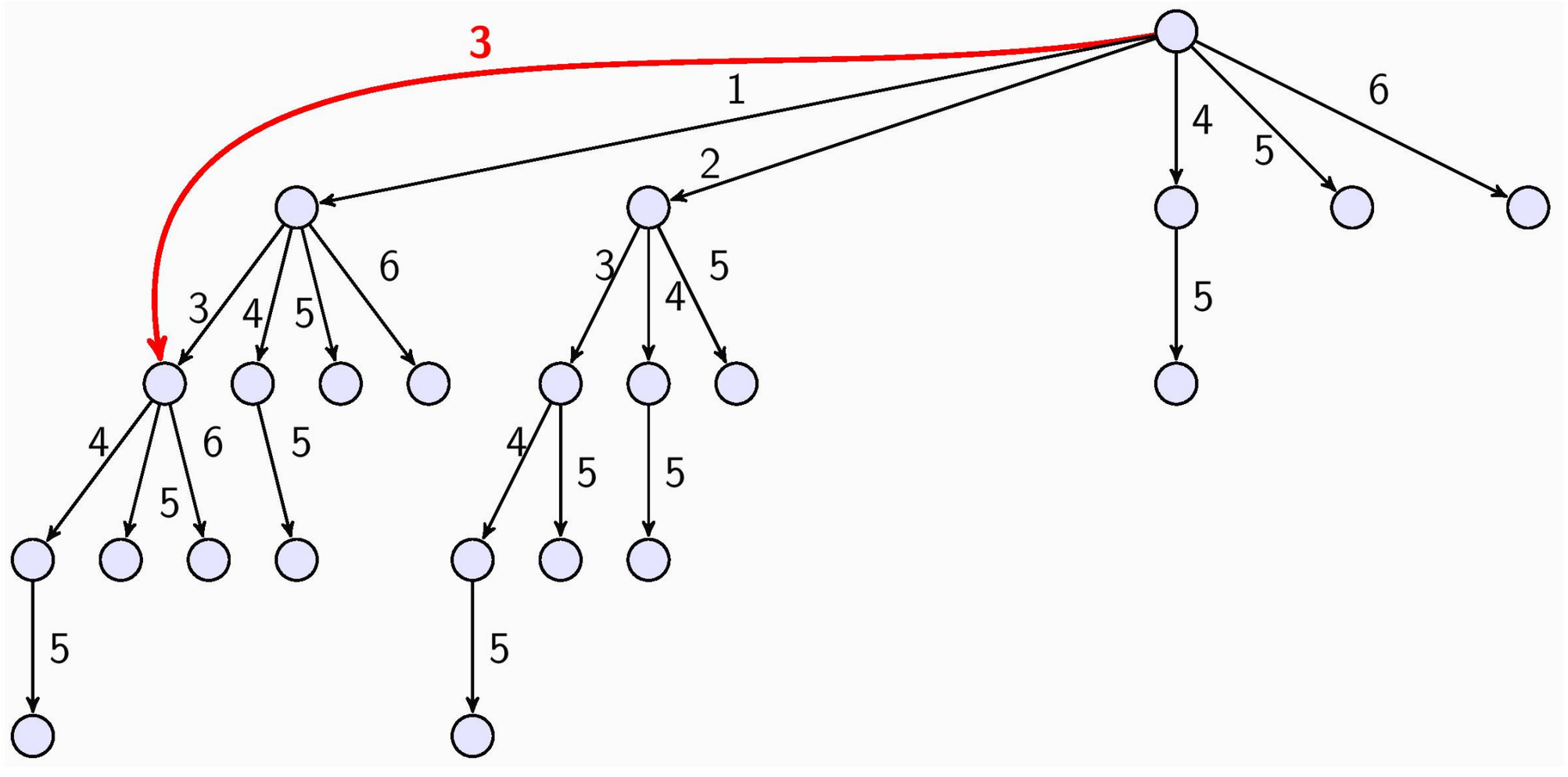


Storing simplicial complexes

[*The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes*, Boissonnat, Maria, Algorithmica, 2014]

We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

Unfortunately, the simplex tree also has redundancies.



An invariant fit for computation

Pb 2: Looking for homotopy equivalences/homeomorphisms/isotopies is extremely difficult. Are there mathematical quantities that are invariant to homotopy equivalences **and** easy to compute?

An invariant fit for computation

Pb 2: Looking for homotopy equivalences/homeomorphisms/isotopies is extremely difficult. Are there mathematical quantities that are invariant to homotopy equivalences **and** easy to compute?

A: The *holes*, encoded in the *homology groups* H_k, $k \in \mathbb{N}$

The homology groups

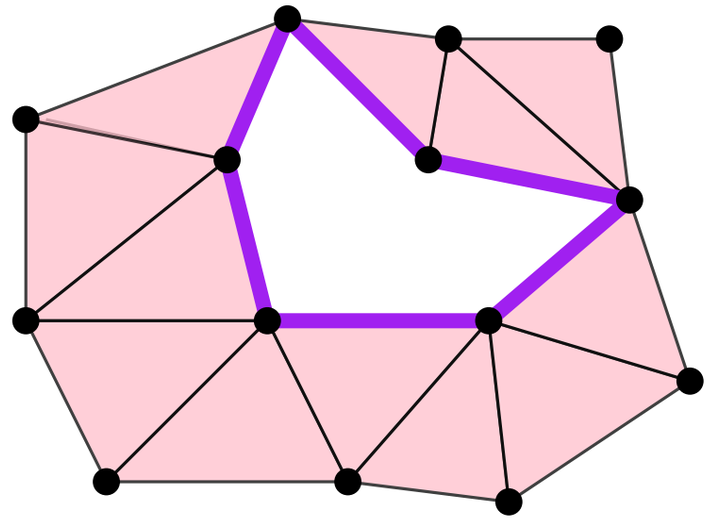
The homology groups

Q: How to characterize a hole in a simplicial complex?

The homology groups

Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

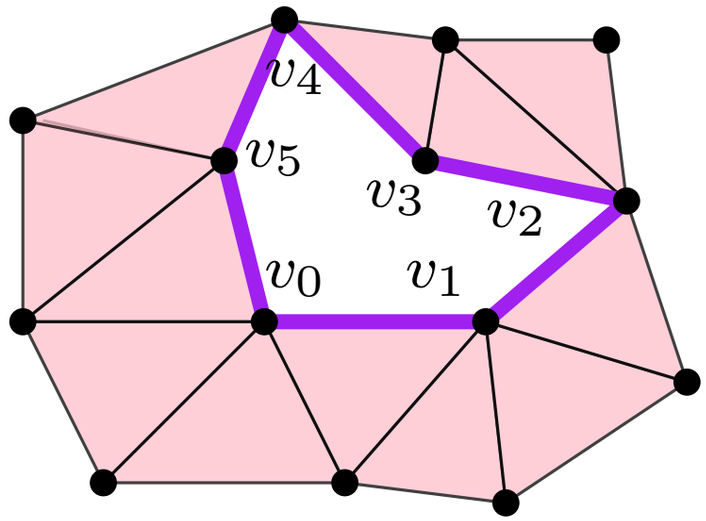


The homology groups

Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

The sequence of 1-dimensional simplices $[v_0, v_1]$, $[v_1, v_2]$, $[v_2, v_3]$, $[v_3, v_4]$, $[v_4, v_5]$, $[v_5, v_0]$ is a hole

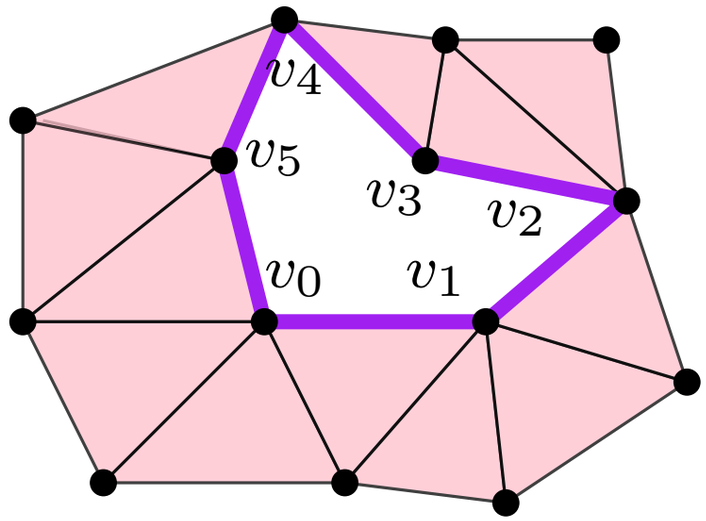


The homology groups

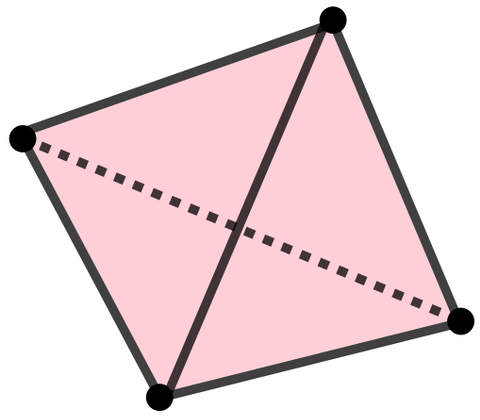
Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

The sequence of 1-dimensional simplices $[v_0, v_1]$, $[v_1, v_2]$, $[v_2, v_3]$, $[v_3, v_4]$, $[v_4, v_5]$, $[v_5, v_0]$ is a hole



But what about higher dimensional holes (like the inside of a tetrahedron)?

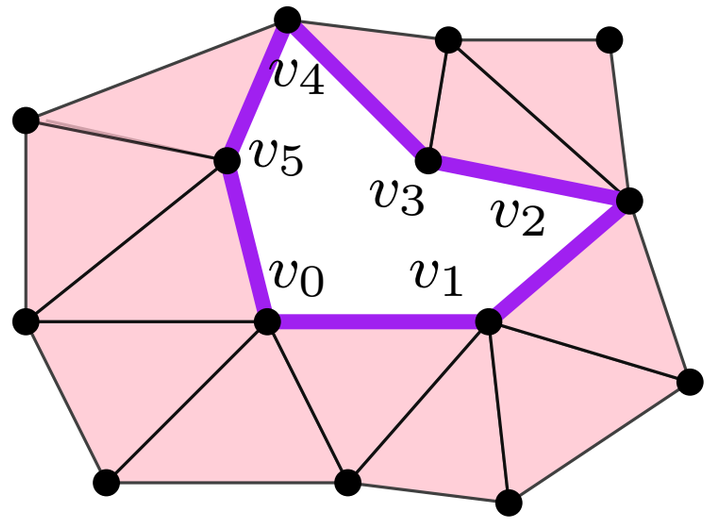


The homology groups

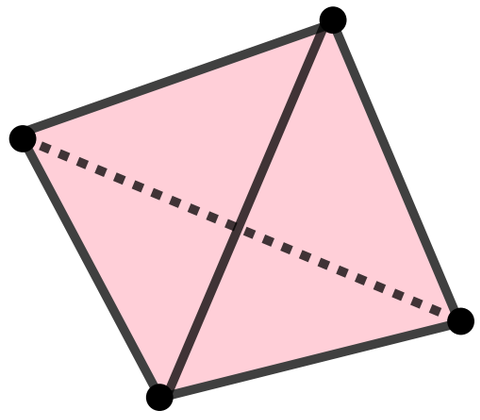
Q: How to characterize a hole in a simplicial complex?

A: A hole (in 1D) is a path whose first and end points are the same, a loop.

The sequence of 1-dimensional simplices $[v_0, v_1]$, $[v_1, v_2]$, $[v_2, v_3]$, $[v_3, v_4]$, $[v_4, v_5]$, $[v_5, v_0]$ is a hole



But what about higher dimensional holes (like the inside of a tetrahedron)?



A: A hole in dimension d is a simplicial complex in which each $(d-1)$-simplex appears an even number of times.

The homology groups

Def: A *d-chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

The homology groups

Def: A *d-chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

The homology groups

Def: A *d-chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

The homology groups

Def: A *d-chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

$$\partial_1 C = \partial_1 [v_0, v_1] + \partial_1 [v_1, v_2] + \partial_1 [v_2, v_3] + \partial_1 [v_3, v_4] + \partial_1 [v_4, v_5] + \partial_1 [v_5, v_0]$$

The homology groups

Def: A *d-chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

$$\partial_1 C = \partial_1 [v_0, v_1] + \partial_1 [v_1, v_2] + \partial_1 [v_2, v_3] + \partial_1 [v_3, v_4] + \partial_1 [v_4, v_5] + \partial_1 [v_5, v_0]$$

$$= [v_0] + [v_1] + [v_1] + [v_2] + [v_2] + [v_3] + [v_3] + [v_4] + [v_4] + [v_5] + [v_5] + [v_0]$$

The homology groups

Def: A *d-chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

$$\partial_1 C = \partial_1 [v_0, v_1] + \partial_1 [v_1, v_2] + \partial_1 [v_2, v_3] + \partial_1 [v_3, v_4] + \partial_1 [v_4, v_5] + \partial_1 [v_5, v_0]$$

$$= [v_0] + [v_1] + [v_1] + [v_2] + [v_2] + [v_3] + [v_3] + [v_4] + [v_4] + [v_5] + [v_5] + [v_0]$$

$$= [v_0] + [v_0] = 0.$$

Def: A *d-cycle* is a d-chain C s.t. $\partial C = 0$.

The homology groups

Def: A *d-chain* is a formal sum of d-simplices with coefficients in $\mathbb{Z}/2\mathbb{Z}$.

$$C = [v_0, v_1] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_5] + [v_5, v_0].$$

Def: The *boundary* of a d-simplex is the chain made of its $(d-1)$-simplices.

$$\partial_n [v_1, \ldots, v_{n+1}] = \sum_{i=1}^{n+1} [v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1}]$$

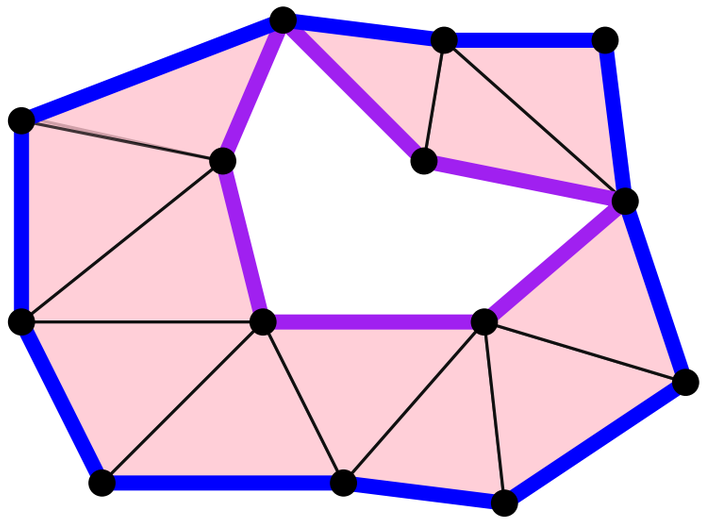
$$\partial_1 C = \partial_1[v_0, v_1] + \partial_1[v_1, v_2] + \partial_1[v_2, v_3] + \partial_1[v_3, v_4] + \partial_1[v_4, v_5] + \partial_1[v_5, v_0]$$

$$= [v_0] + [v_1] + [v_1] + [v_2] + [v_2] + [v_3] + [v_3] + [v_4] + [v_4] + [v_5] + [v_5] + [v_0]$$

$$= [v_0] + [v_0] = 0.$$

Def: A *d-cycle* is a d-chain C s.t. $\partial C = 0$.

Pb: Cycles are not holes!!



The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0$.

Q: Prove it.

The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0$.

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$

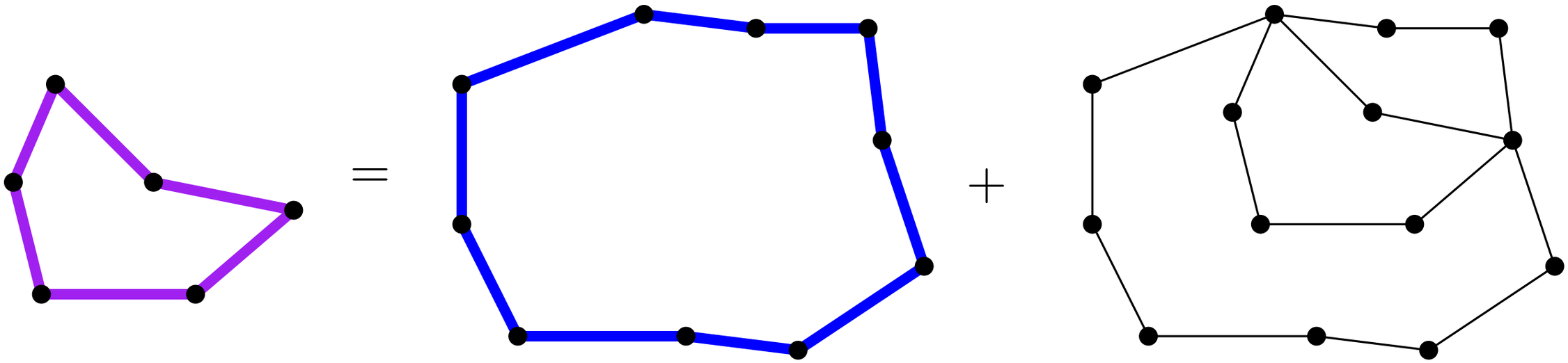
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0$.

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$



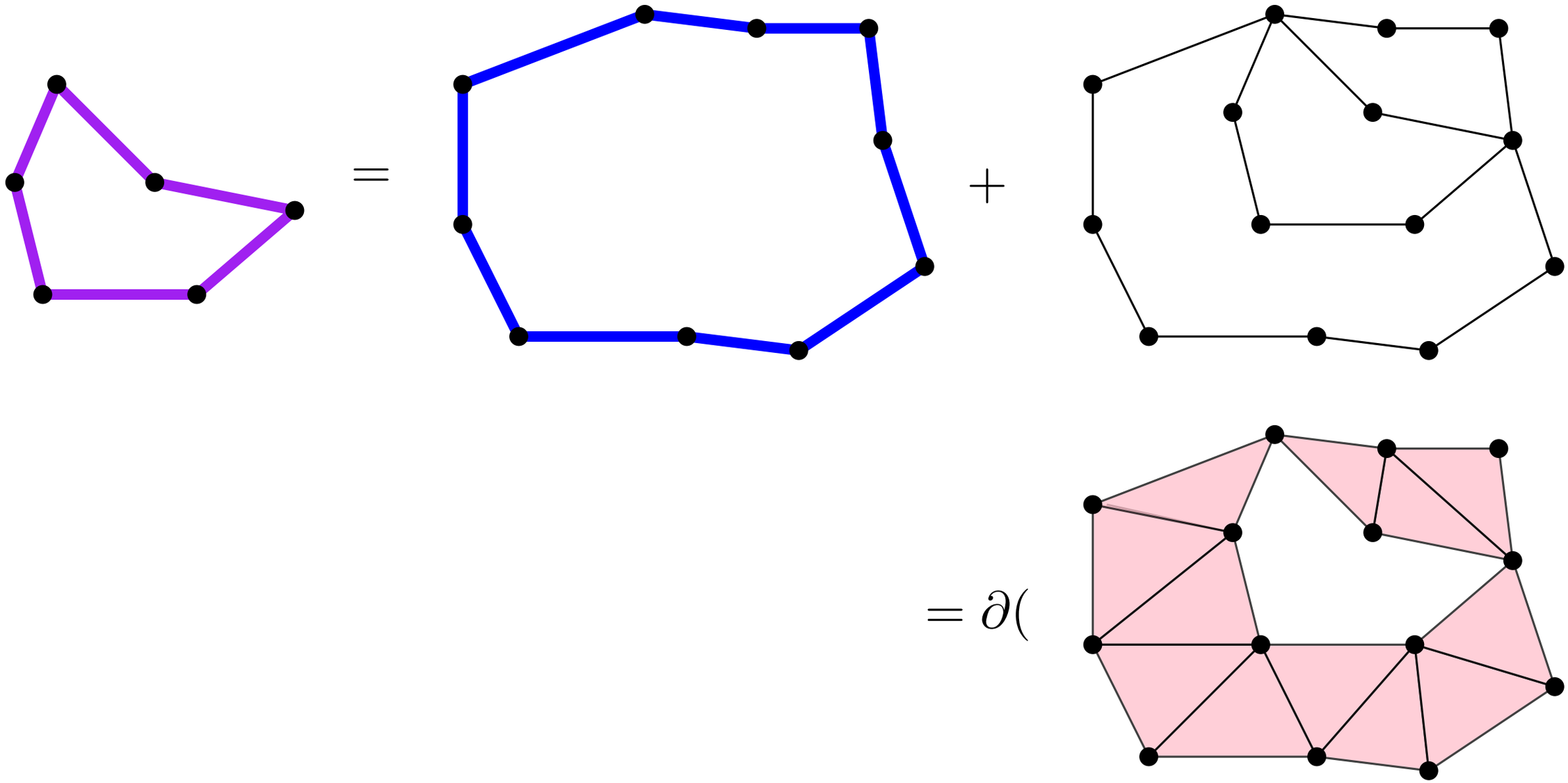
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0$.

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$



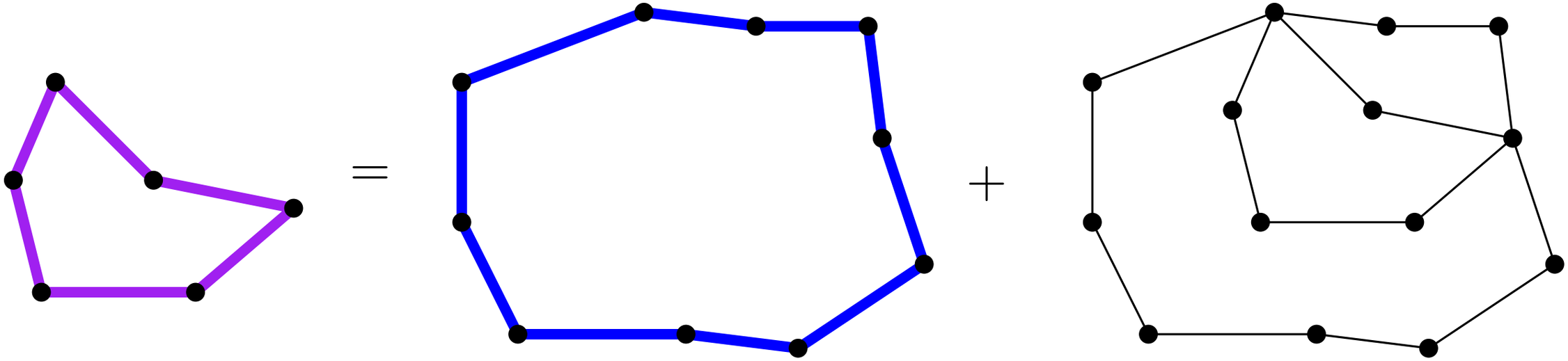
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0.$

Q: Prove it.

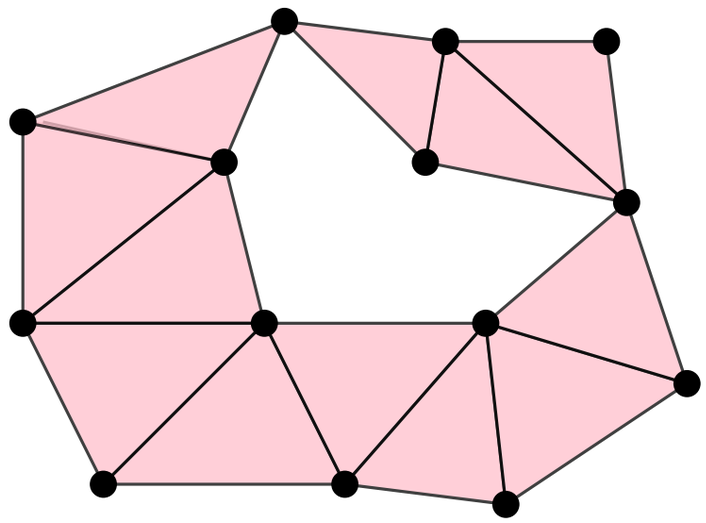
Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$



$$H_k = Z_k / B_k$$

$$= \partial($$



$$)$$

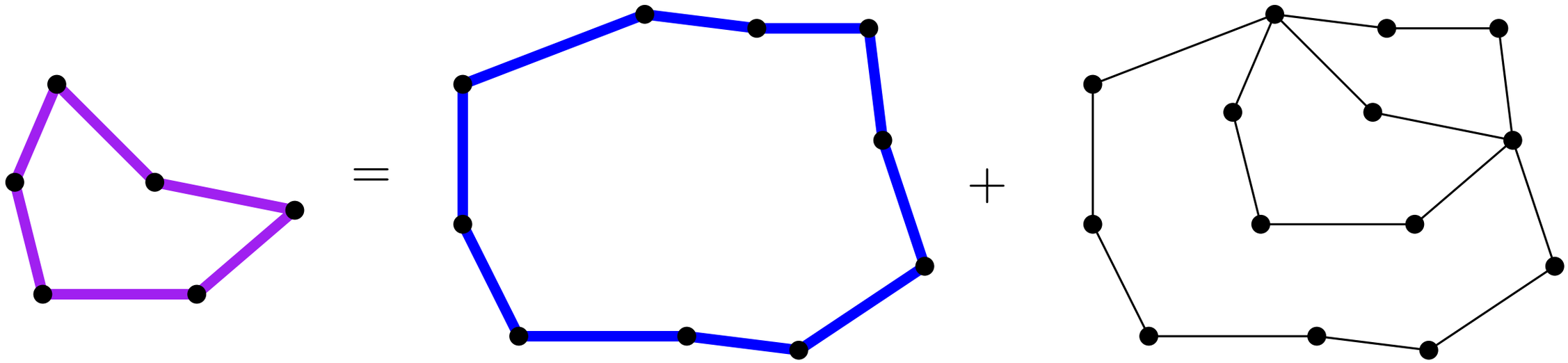
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0$.

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$

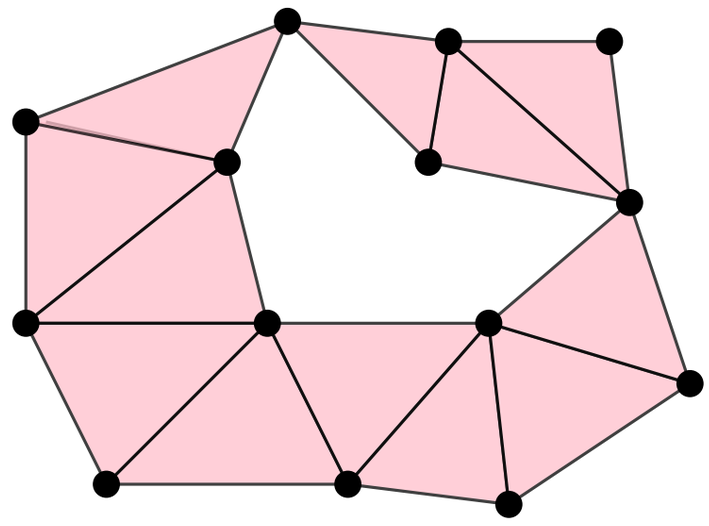


group of k-cycles

$$H_k = Z_k / B_k$$

group of 'cycles minus boundaries'

$\mathrm{im}(\partial_{k+1})$

$= \partial($  $)$

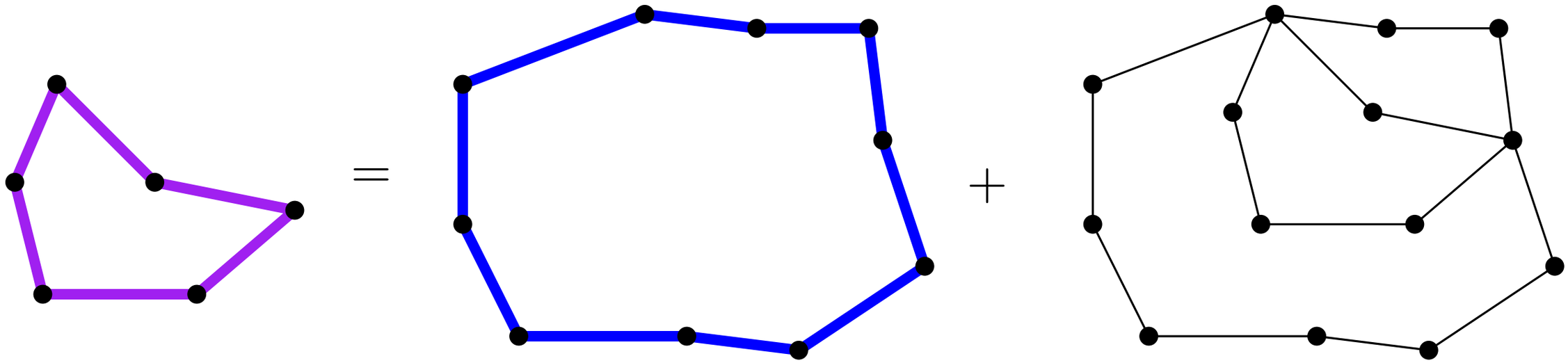
The homology groups

Lemma: $\partial_{n-1} \circ \partial_n = 0$.

Q: Prove it.

Def: Two cycles are the same (homologous) if 'their difference is in $\mathrm{im}(\partial)$':

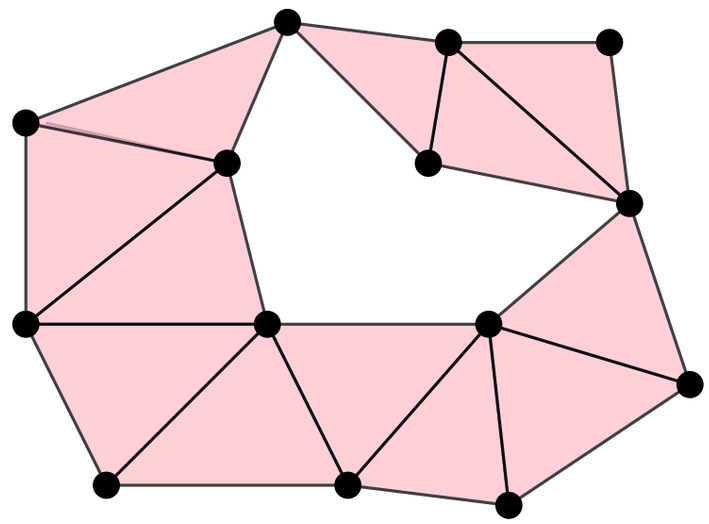
$$C \sim C' \iff C + C' \in \mathrm{im}(\partial)$$



$$H_k = \{[C] \ : \ C \in Z_k\}$$

where

$$[C] = \{C' \ : \ C \sim C'\}$$

$= \partial($  $)$

The homology groups

H_k is a group (vector space) in which each element is an equivalence class of cycles associated to the same hole.

Def: The dimension of H_k is called the *Betti number* β_k.

Minimum number of (classes of) cyles needed to create a basis, i.e., to be able to write *any* cycle as a linear combination of cycles in the basis.

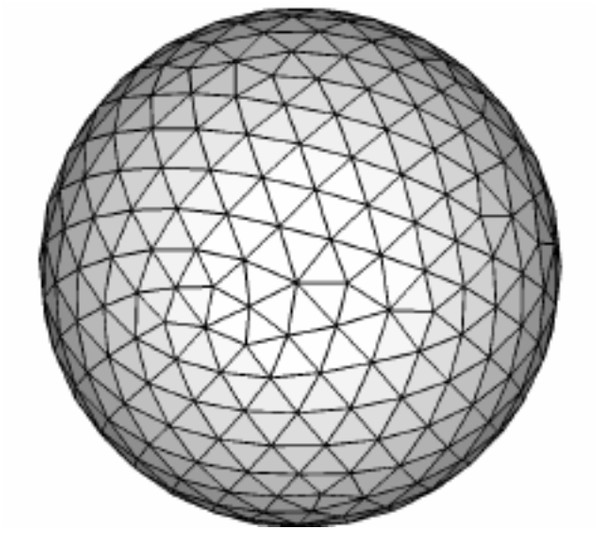
β_0 counts the connected components, β_1 counts the loops, β_2 counts the cavities, and so on...

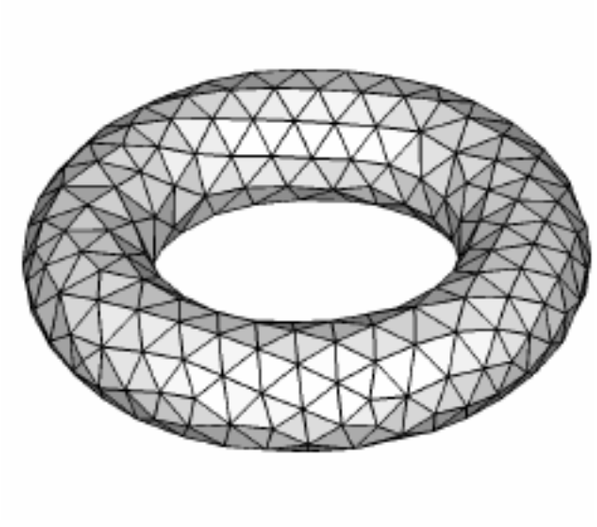
The homology groups

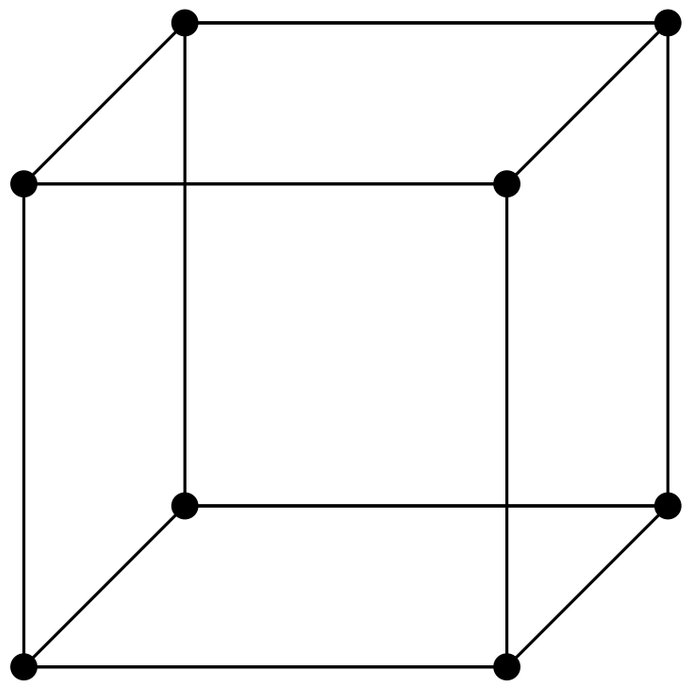
H_k is a group (vector space) in which each element is an equivalence class of cycles associated to the same hole.

Def: The dimension of H_k is called the *Betti number* β_k.

Q: What are the Betti numbers of:





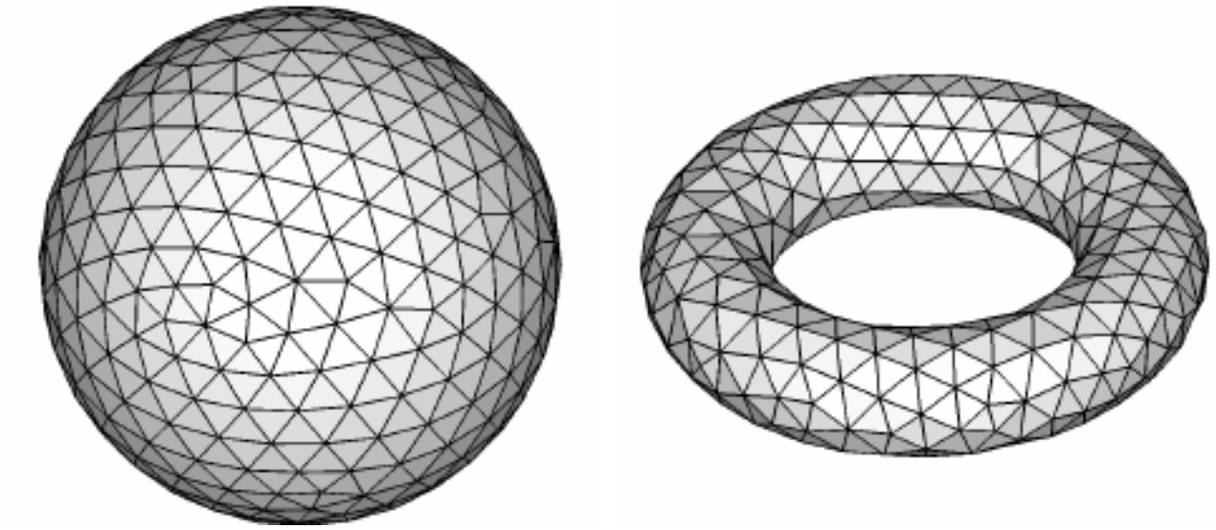


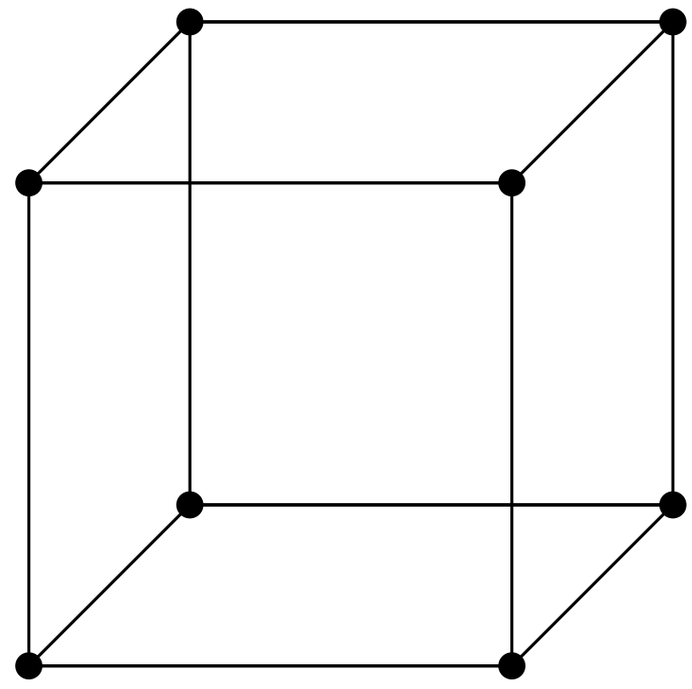
The homology groups

H_k is a group (vector space) in which each element is an equivalence class of cycles associated to the same hole.

Def: The dimension of H_k is called the *Betti number* β_k.

Q: What are the Betti numbers of:





The whole point of homology groups and Betti numbers is that they satisfy:

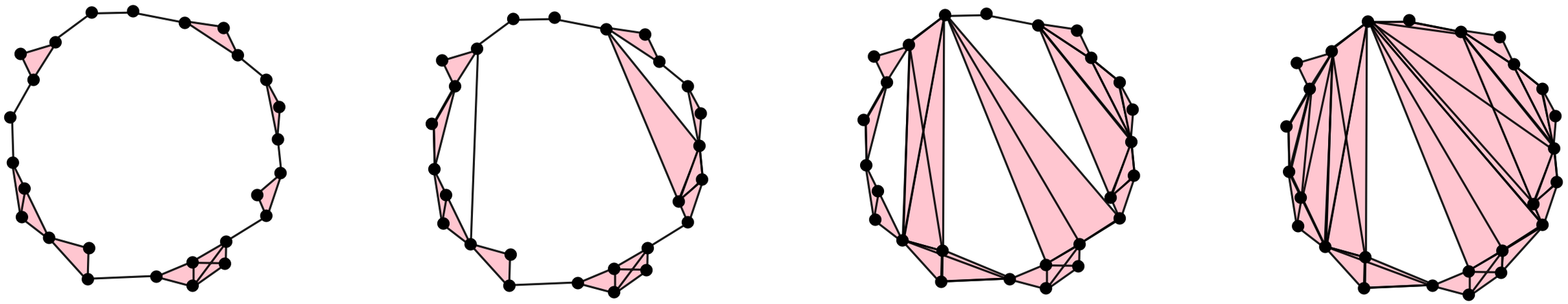
$$H_k(X) \not\sim H_k(Y) \Longrightarrow X \not\sim Y$$

Computation with filtrations and matrix reduction

Algorithms to compute the homology groups of a simplicial complex work by *decomposing* the simplicial complex, with a so-called *filtration*.

Computation with filtrations and matrix reduction

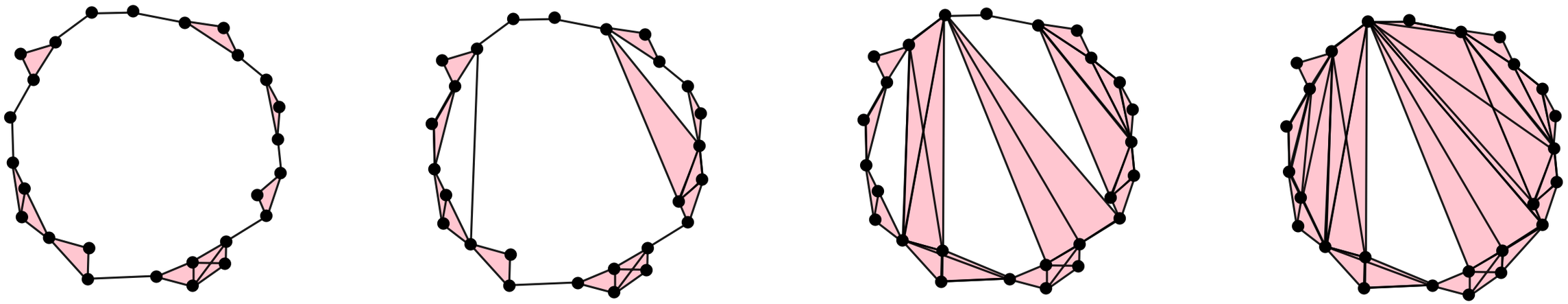
Algorithms to compute the homology groups of a simplicial complex work by *decomposing* the simplicial complex, with a so-called *filtration*.



Def: A filtered simplicial complex S is a family $\{S_a\}_{a\in\mathbb{R}}$ of subcomplexes of some fixed simplicial complex S s.t. $S_a \subseteq S_b$ for any $a \leq b$.

Computation with filtrations and matrix reduction

Algorithms to compute the homology groups of a simplicial complex work by *decomposing* the simplicial complex, with a so-called *filtration*.



Def: A filtered simplicial complex S is a family $\{S_a\}_{a\in\mathbb{R}}$ of subcomplexes of some fixed simplicial complex S s.t. $S_a \subseteq S_b$ for any $a \leq b$.

Def: Let f be a real valued function defined on the vertices of K. For $\sigma = [v_0, \ldots, v_k] \in K$, let $f(\sigma) = \max_{i=0,\ldots,k} f(v_i)$, and order the simplices of K in increasing order w.r.t. the function f values (and break ties with dimension in case some simplices have the same function value).

Q: Show that this is a filtration.

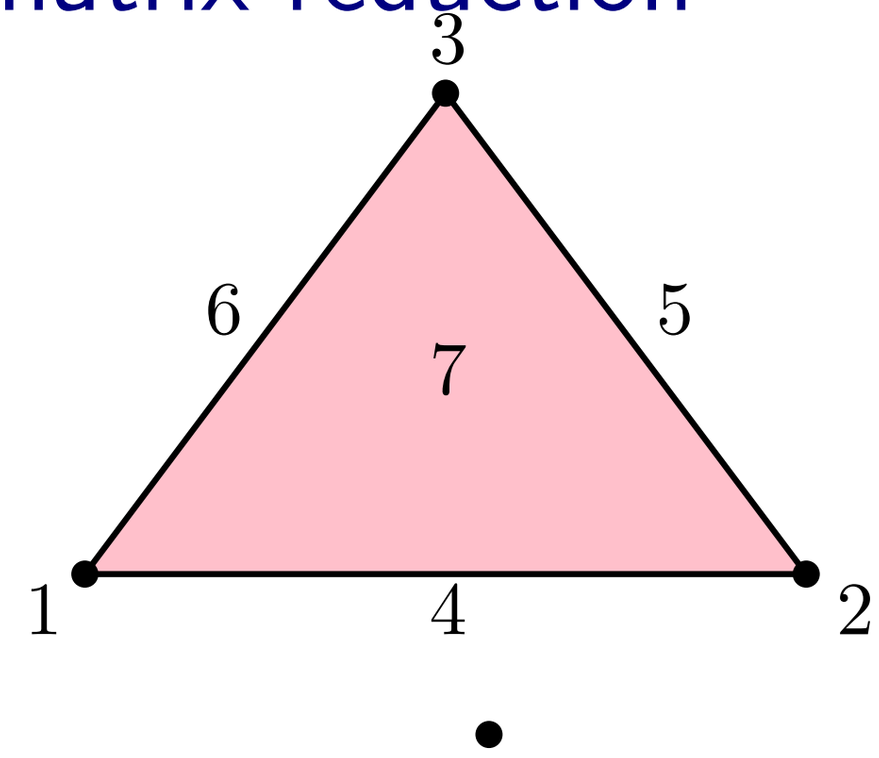
Computation with filtrations and matrix reduction

Input: simplicial filtration

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*



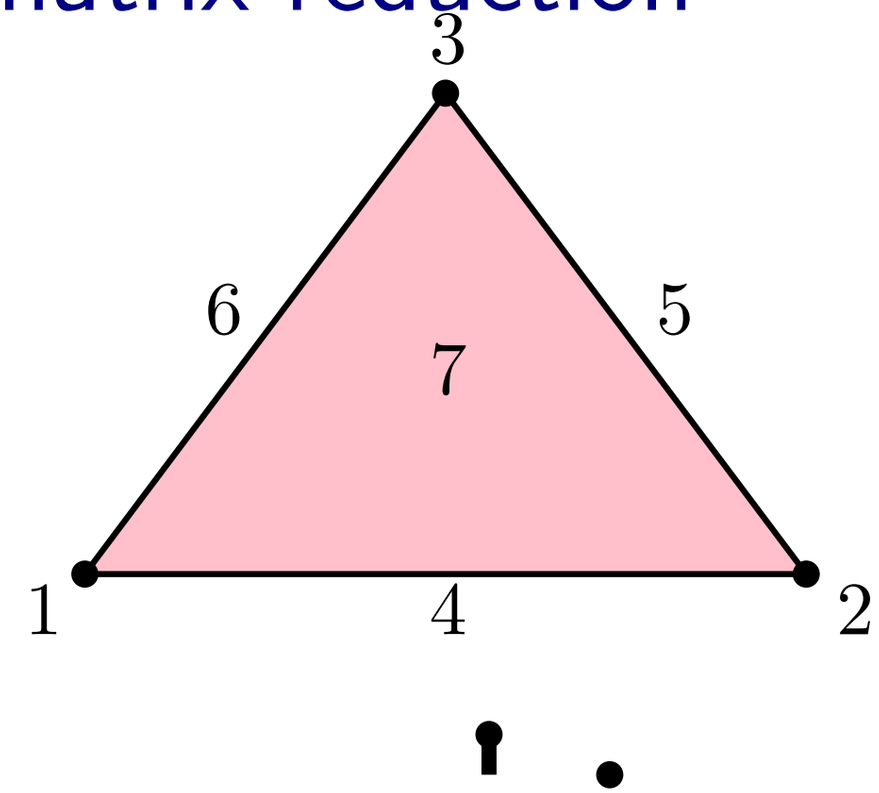
Computation with filtrations and matrix reduction

Input: simplicial filtration

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*



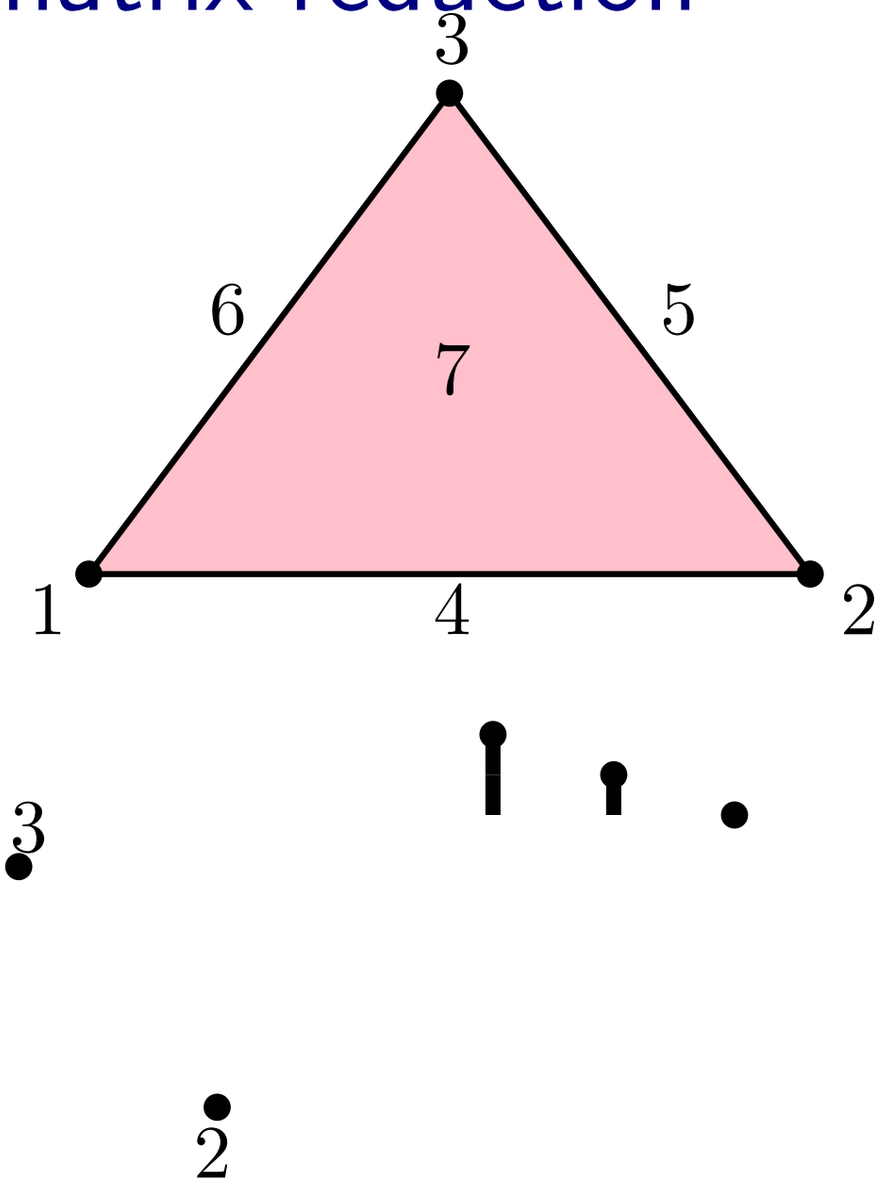
Computation with filtrations and matrix reduction

Input: simplicial filtration

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*



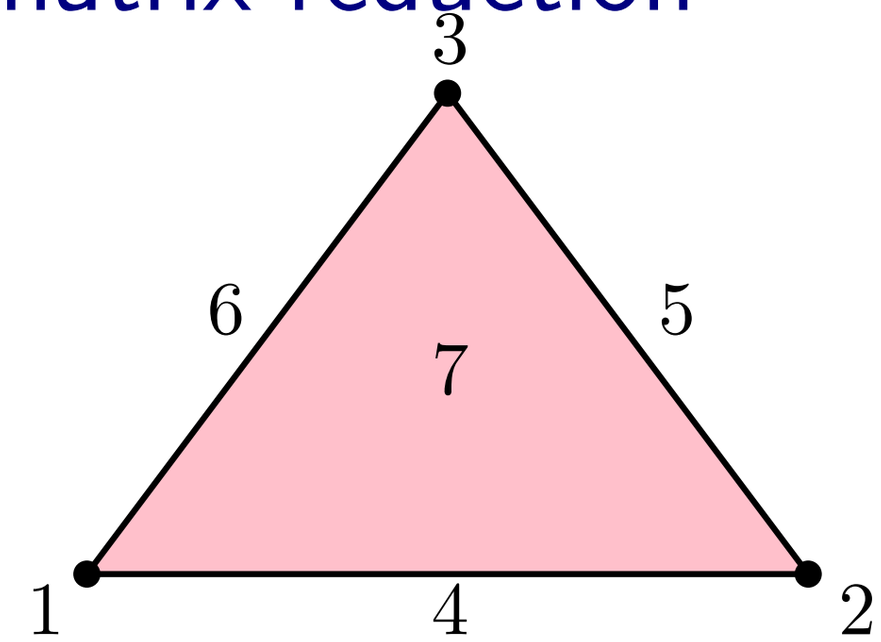
Computation with filtrations and matrix reduction

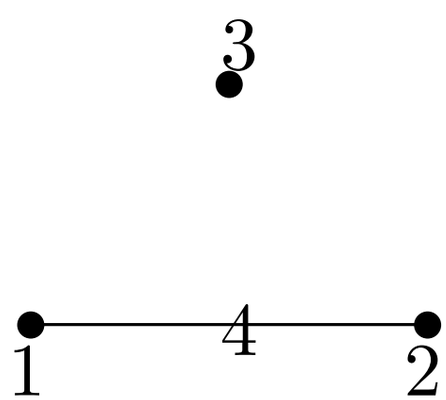
Input: simplicial filtration

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*





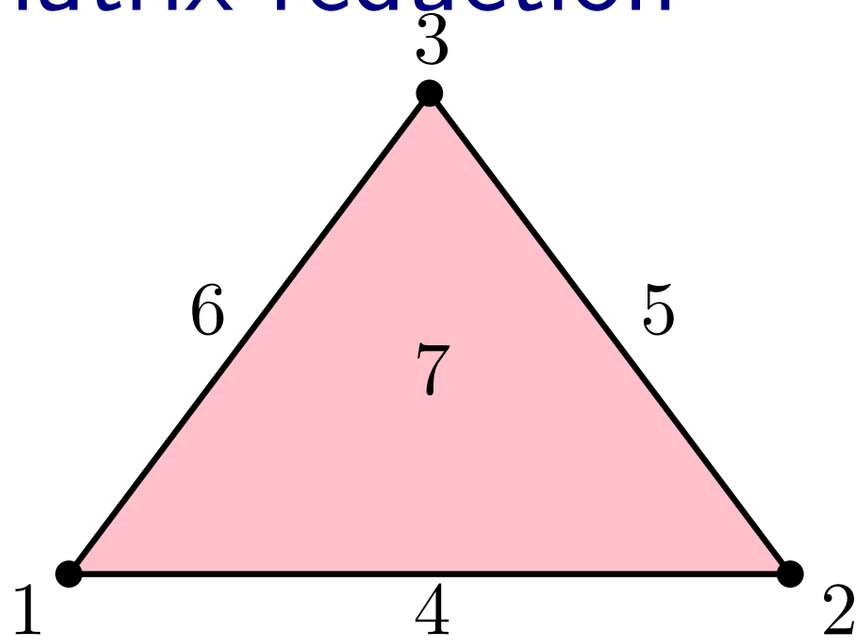
Computation with filtrations and matrix reduction

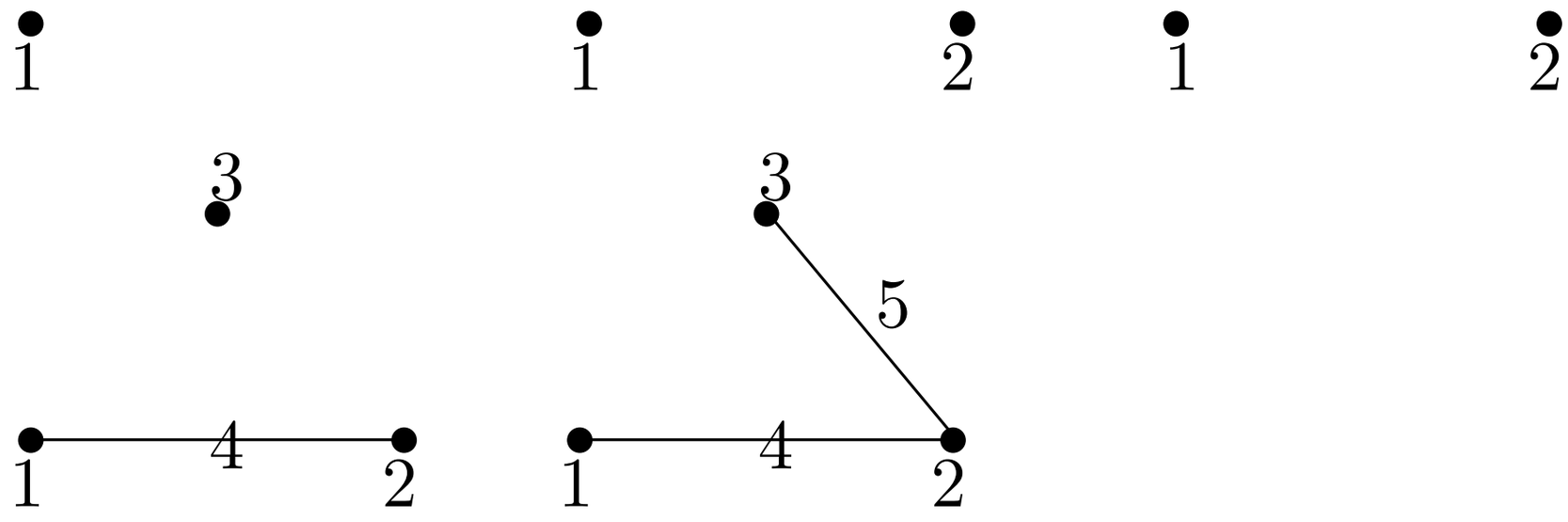
Input: simplicial filtration

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*





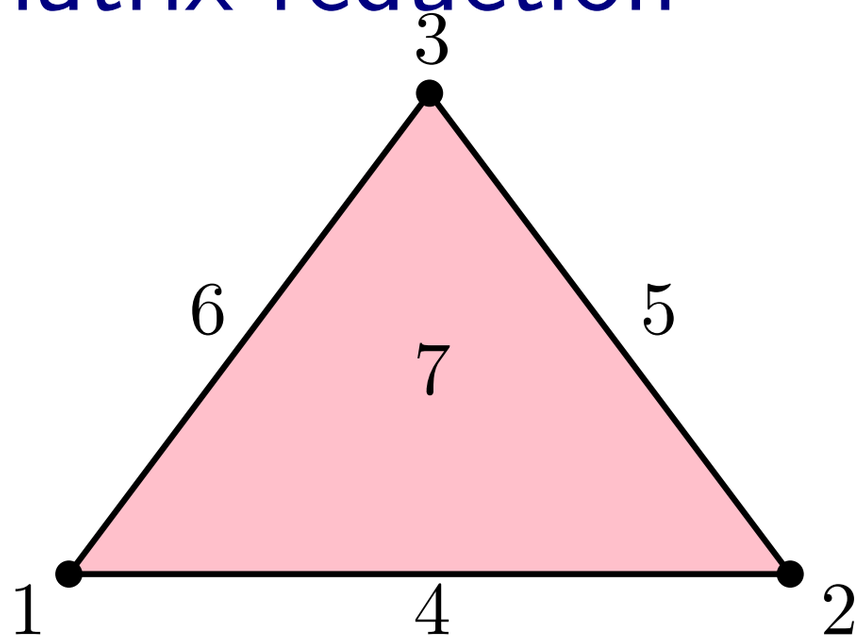
Computation with filtrations and matrix reduction

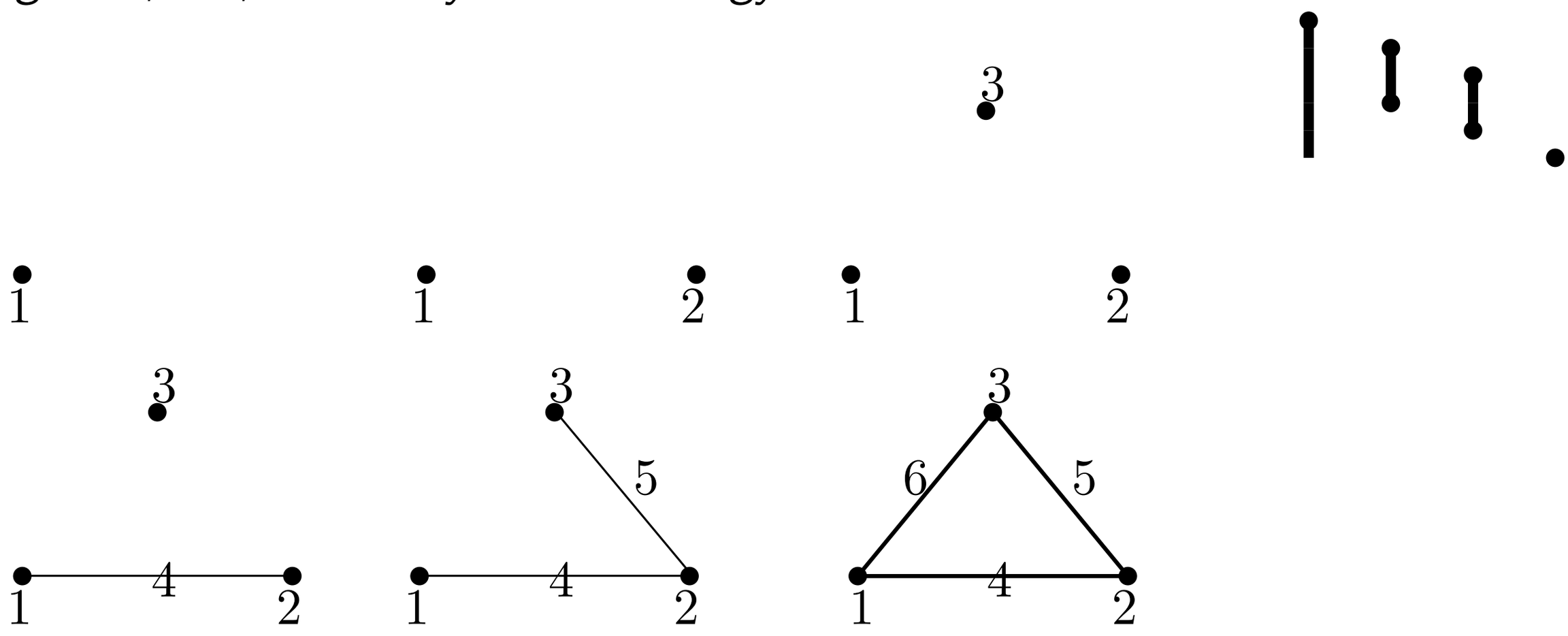
Input: simplicial filtration

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*





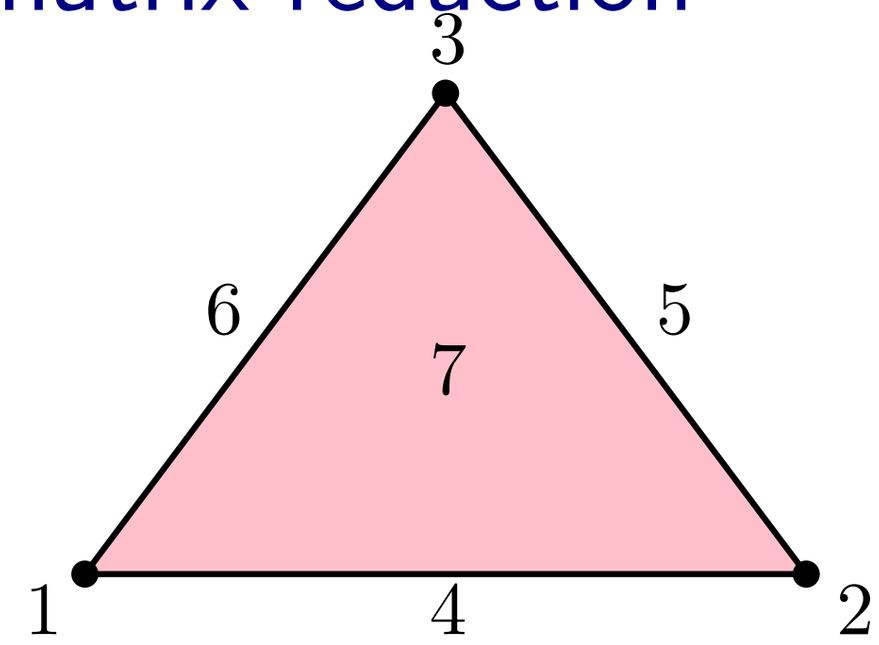
Computation with filtrations and matrix reduction

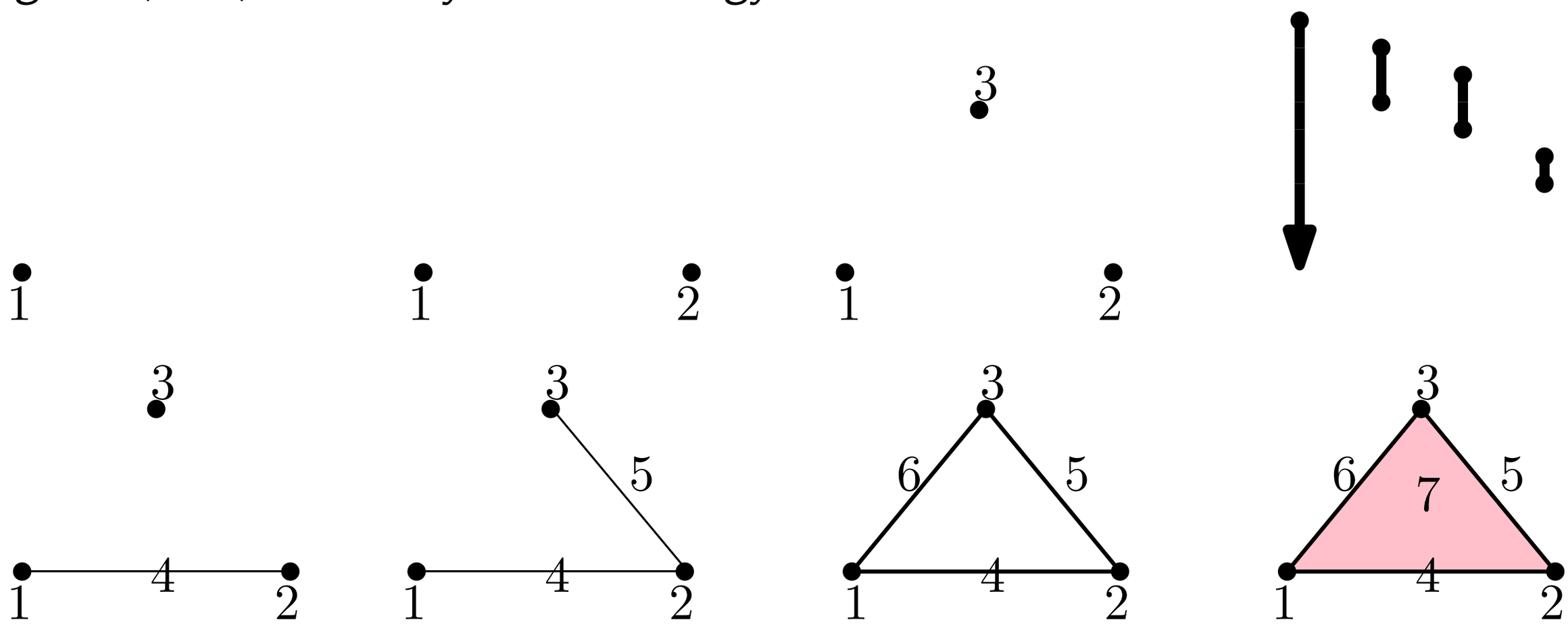
Input: simplicial filtration

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*





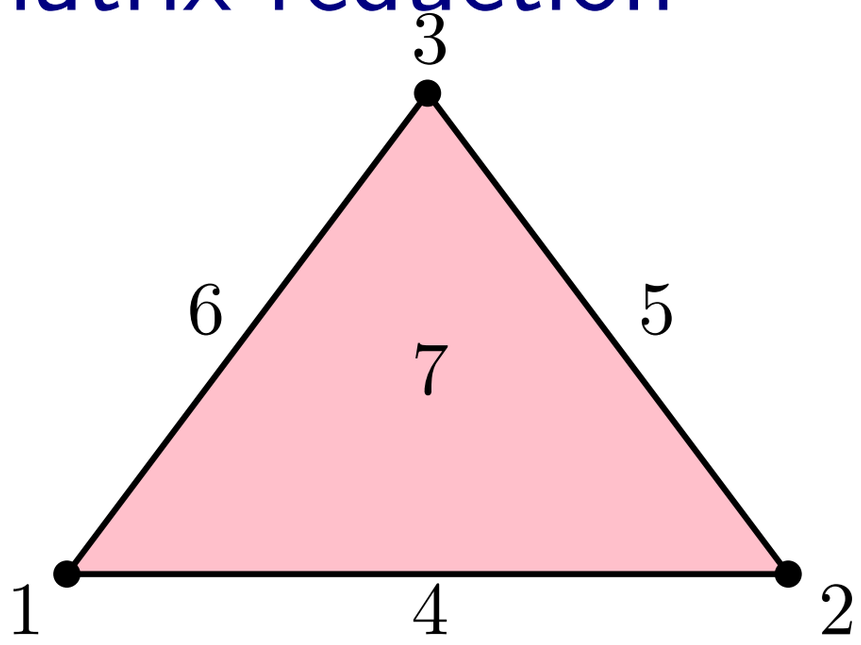
Computation with filtrations and matrix reduction

Input: simplicial filtration

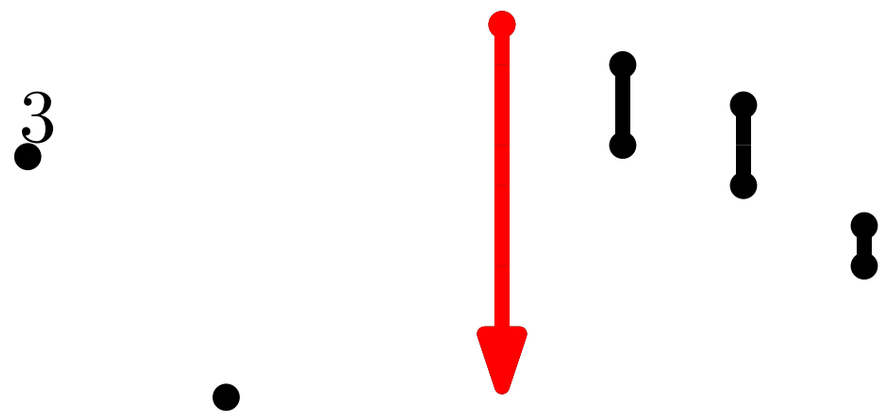
Homology can be computed by using the fact that each simplex is either:

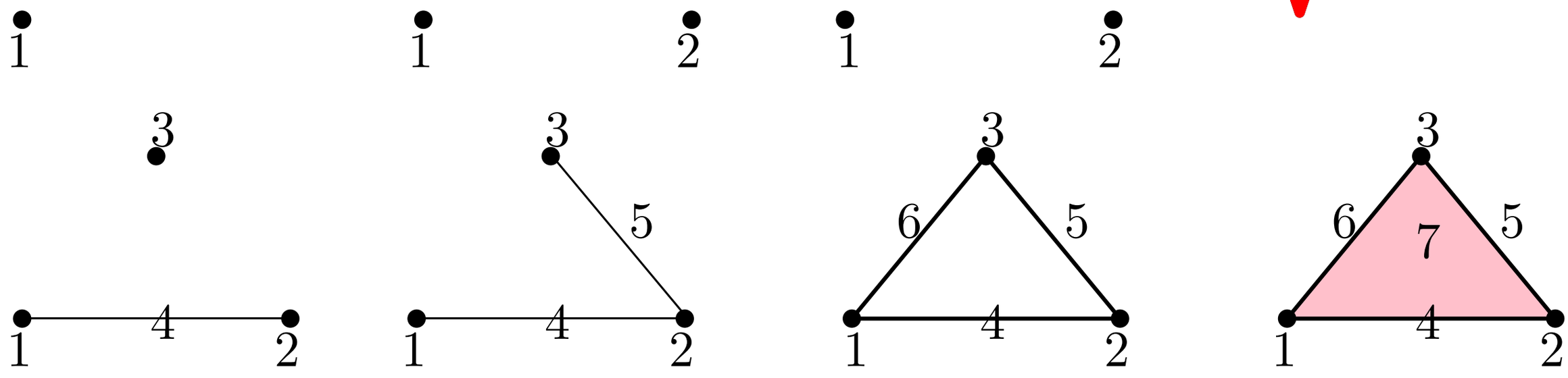
positive, i.e., it *creates a new homology class*

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



The Betti number is equal to the number of bars that are still alive when the full complex is reached in the filtration





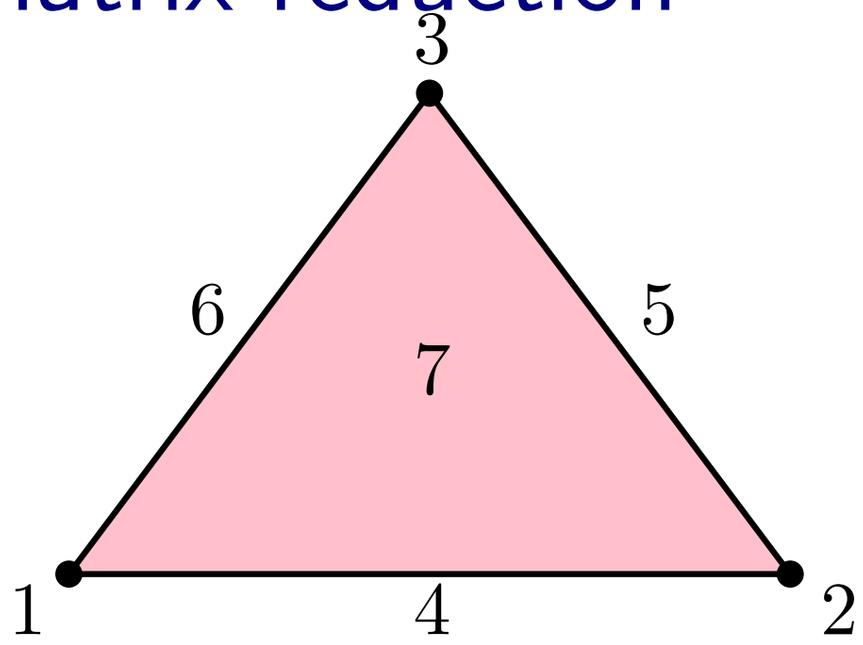
Computation with filtrations and matrix reduction

Input: simplicial filtration

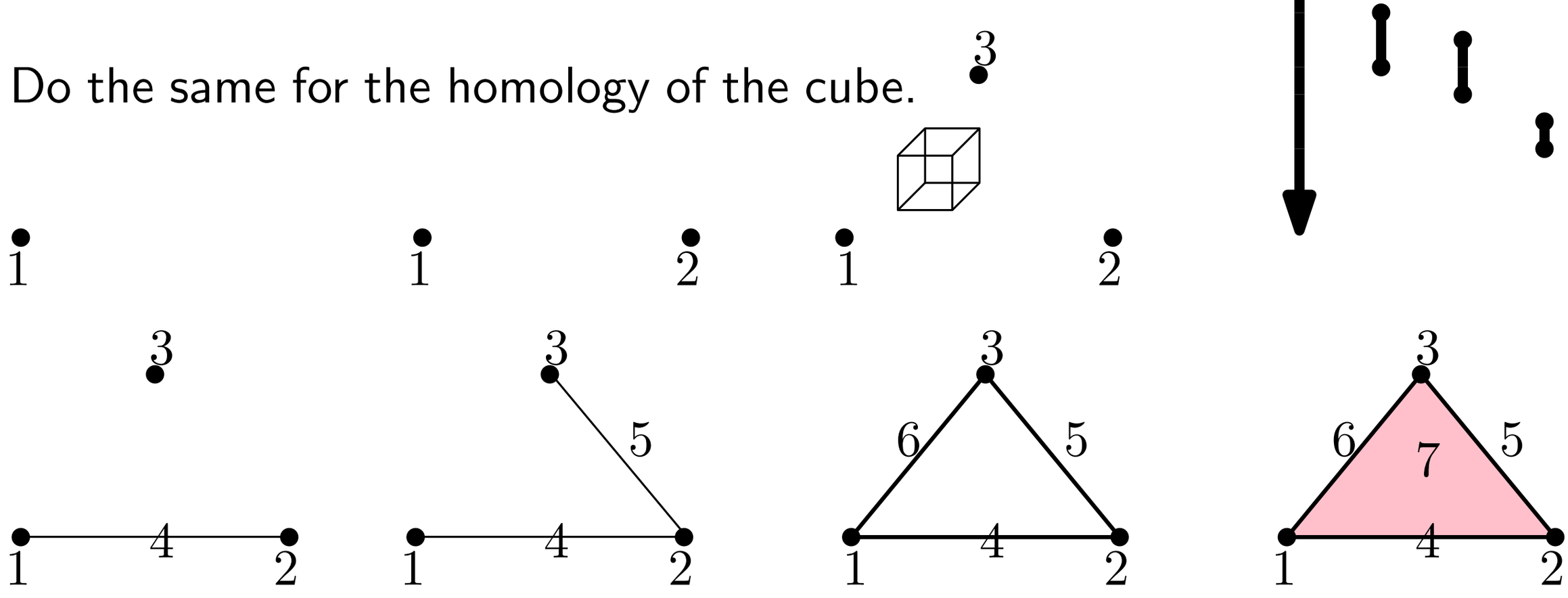
Homology can be computed by using the fact that each simplex is either:

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

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



Q: Do the same for the homology of the cube.

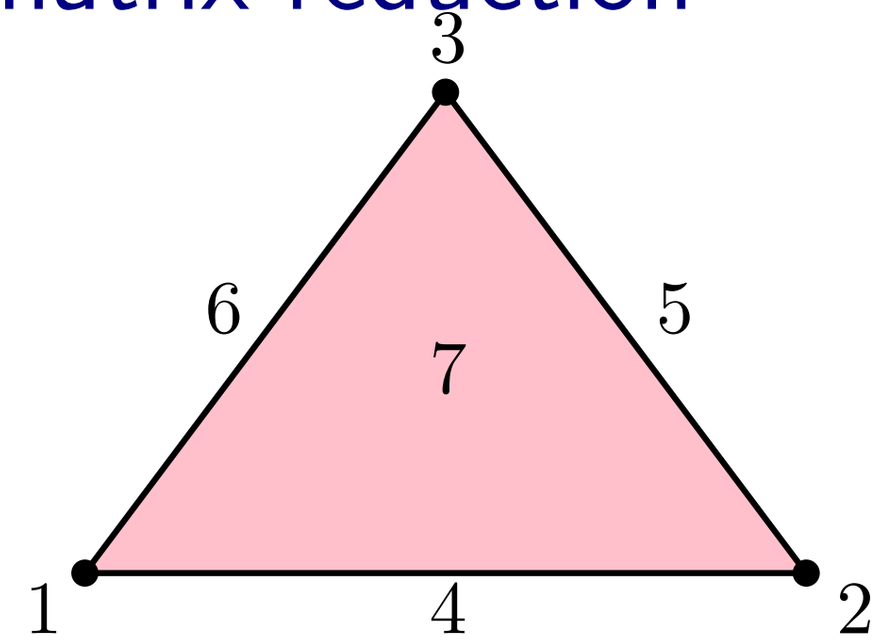


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

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

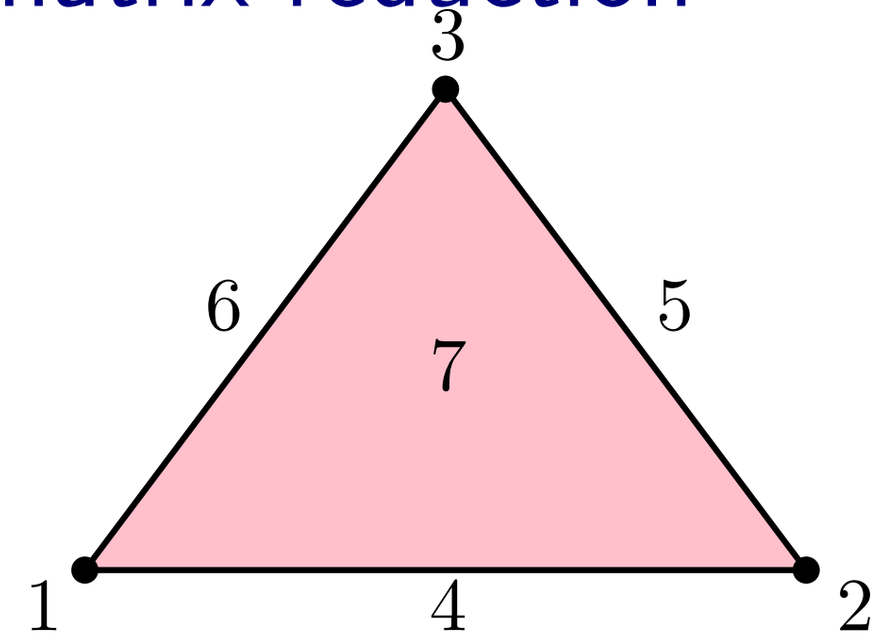


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•			
2				•			
3							
4							
5							
6							
7							

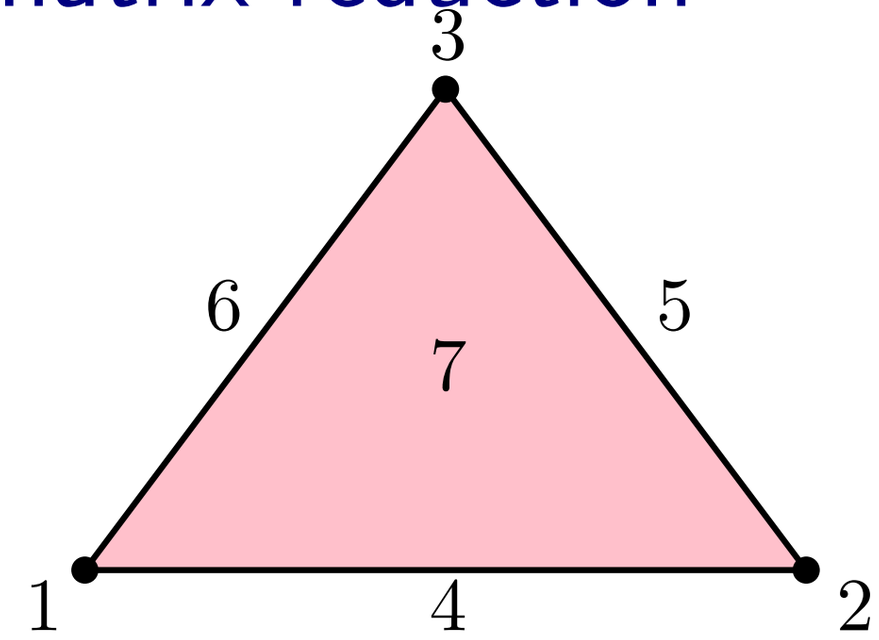


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•			
2				•	•		
3					•		
4							
5							
6							
7							

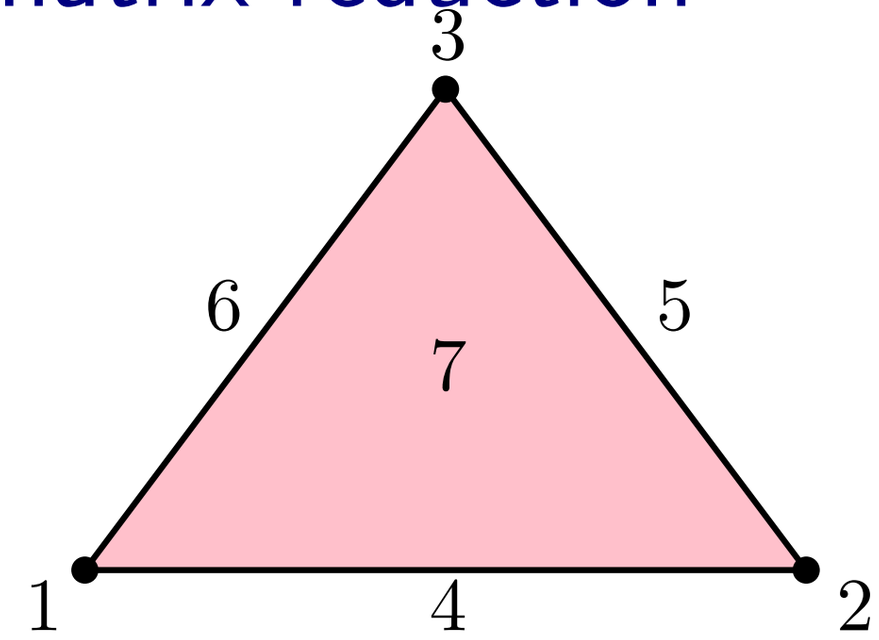


Computation with filtrations and matrix reduction

Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							
5							
6							
7							

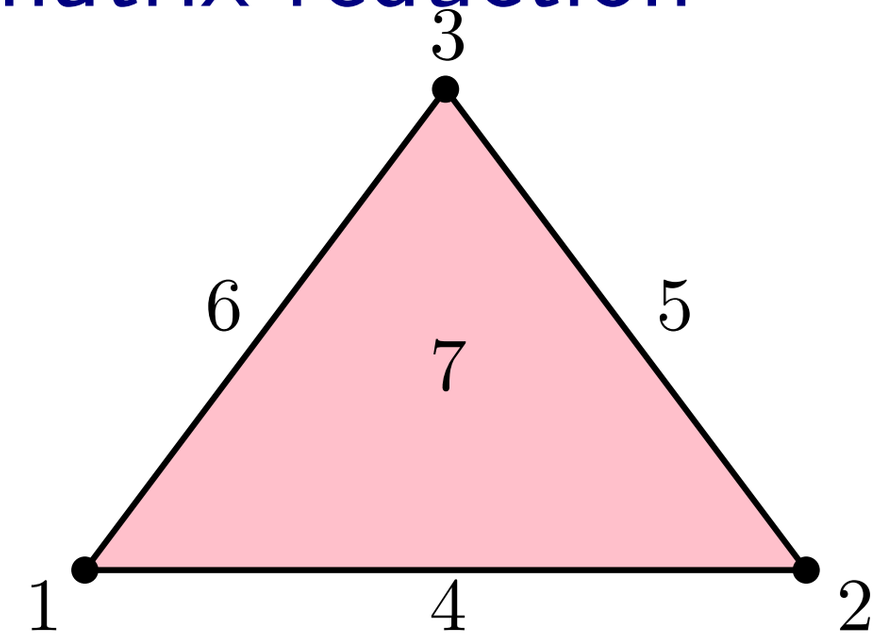


Computation with filtrations and matrix reduction

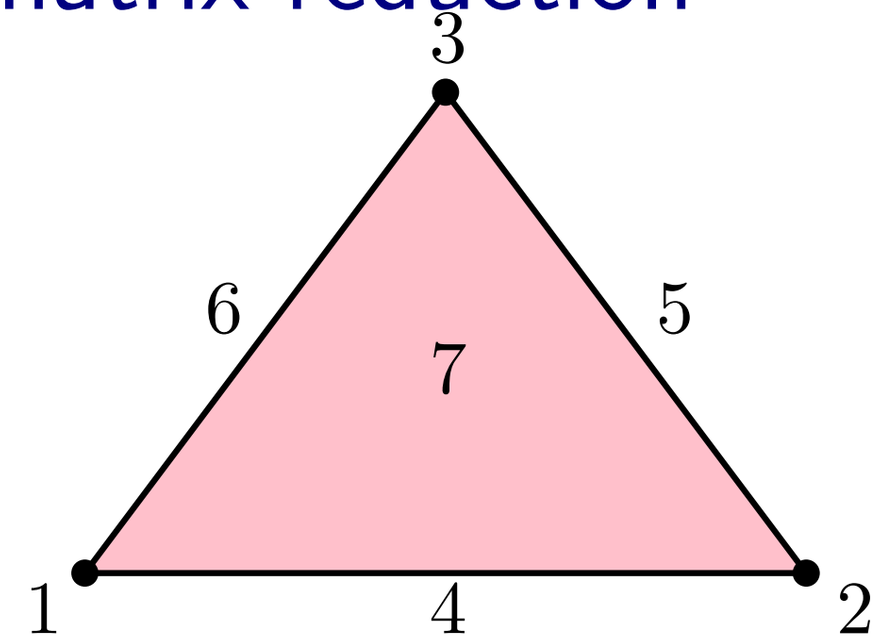
Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							•
5							•
6							•
7							



Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

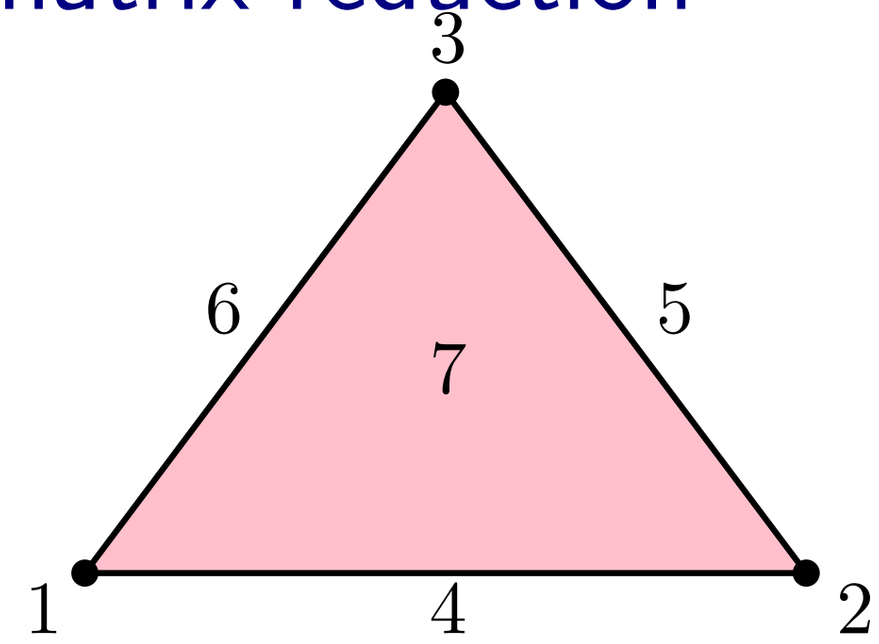
	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							•
5							•
6							•
7							

`for` j=1 `to` m `do:`

`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

col(j) =col(j)+col(k)

Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

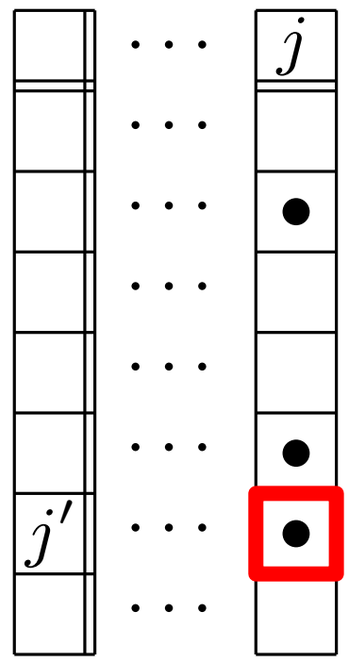
	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							•
5							•
6							•
7							

`for` j=1 `to` m `do:`

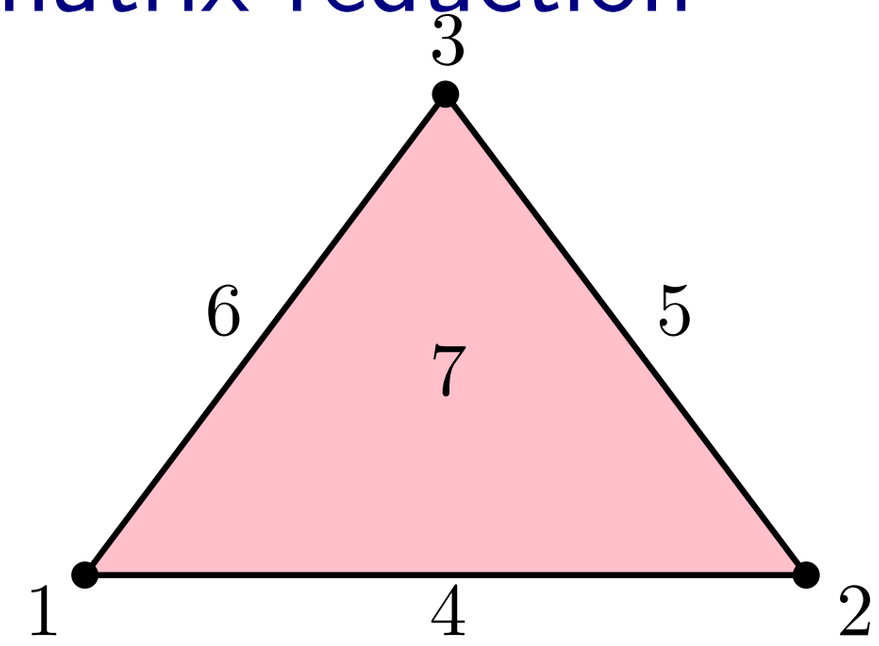
`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

col(j) =col(j)+col(k)

low$(j) = j'$



Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•		
3					•	•	
4							•
5							•
6							•
7							

5	6
	•
•	
•	•

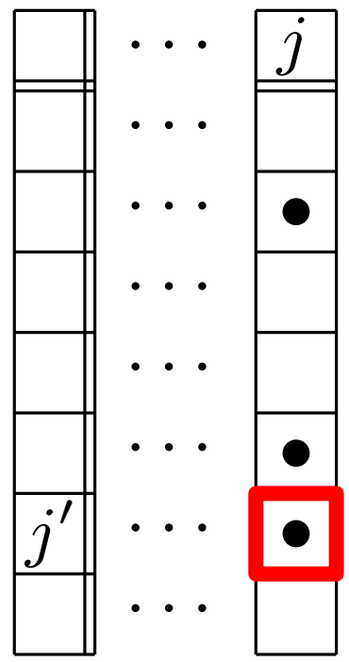
$6 = 6+5$

`for` j=1 `to` m `do:`

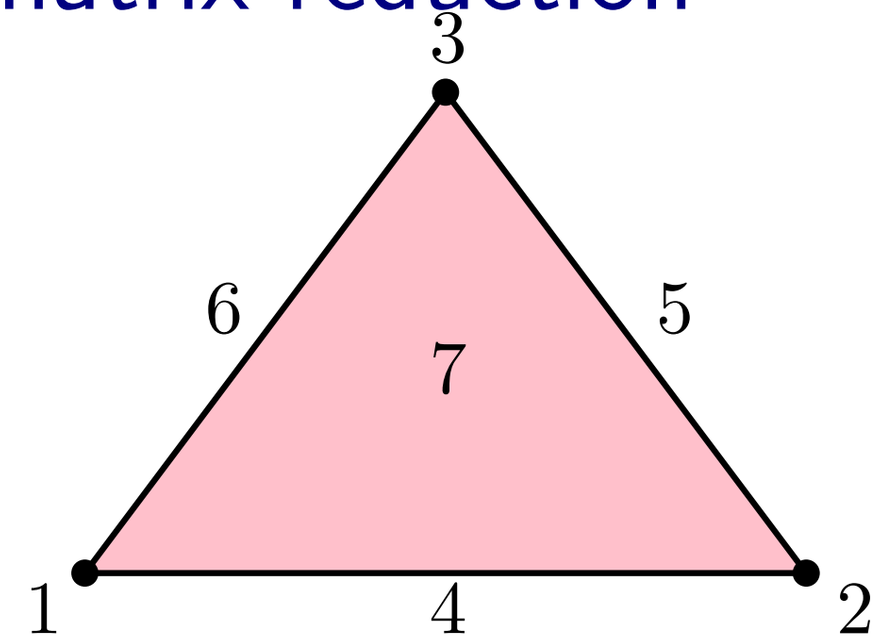
`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

col(j) =col(j)+col(k)

low$(j) = j'$



Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•	•	
3					•		
4							•
5							•
6							•
7							

for j=1 to m do:

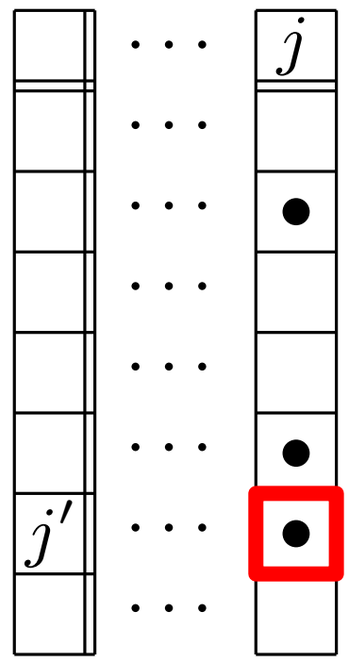
while $\exists k < j$ s.t. low(k) ==low(j) do:

col(j) =col(j)+col(k)

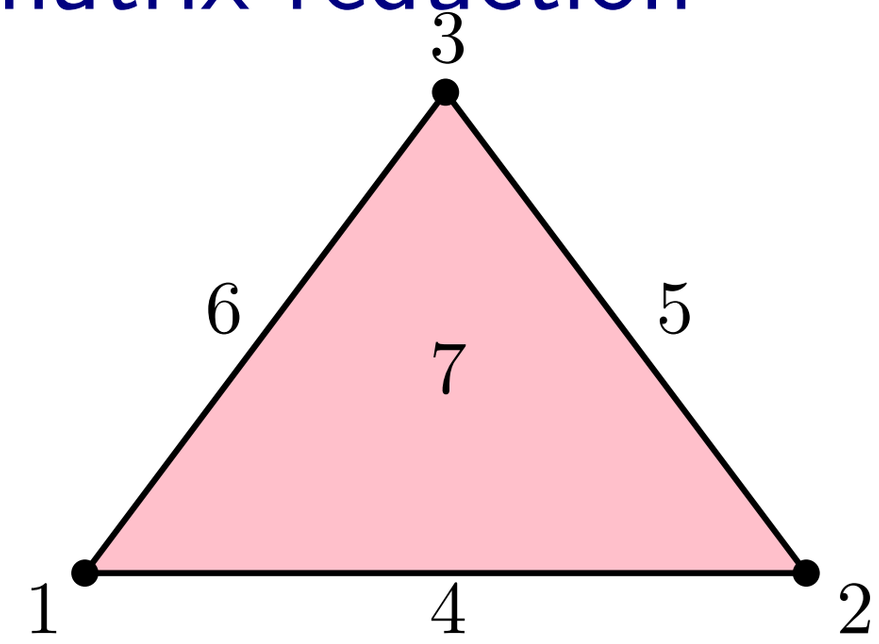
5	6
	•
•	•
•	

6 = 6+5

low$(j) = j'$



Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•		•	
2				•	•	•	
3					•		
4							•
5							•
6							•
7							

5	6
	•
•	•
•	

4		6
•		•
•		•

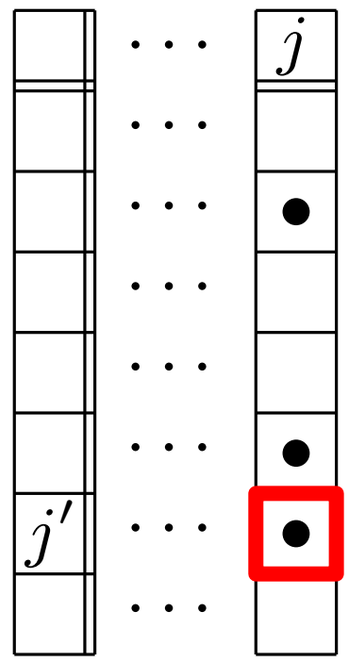
$6 = 6{+}5$
$6 = 6{+}4$

`for` j=1 `to` m `do:`

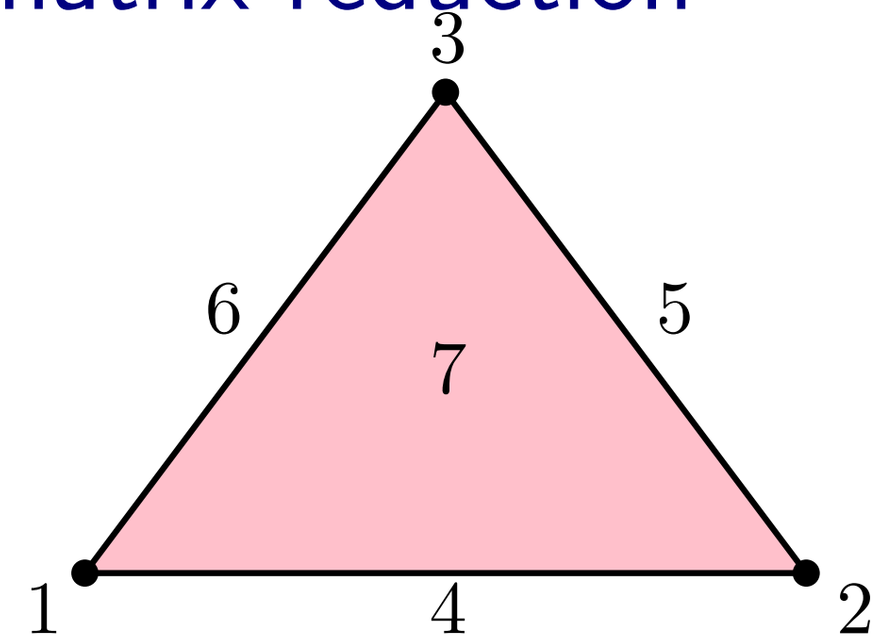
`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

col(j) =col(j)+col(k)

low$(j) = j'$



Computation with filtrations and matrix reduction



Input: simplicial filtration

given as *boundary matrix*

	1	2	3	4	5	6	7
1				•			
2				•	•		
3					•		
4							•
5							•
6							•
7							

5	6
	•
•	•
•	

4		6
•		
•		

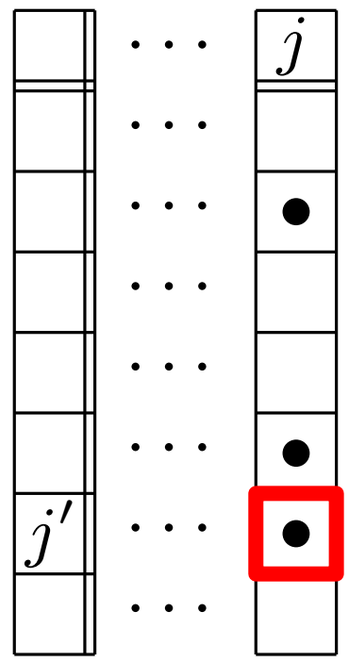
$6 = 6{+}5$
$6 = 6{+}4$

`for` j=1 `to` m `do:`

`while` $\exists k < j$ `s.t.` low(k) ==low(j) `do:`

col(j) =col(j)+col(k)

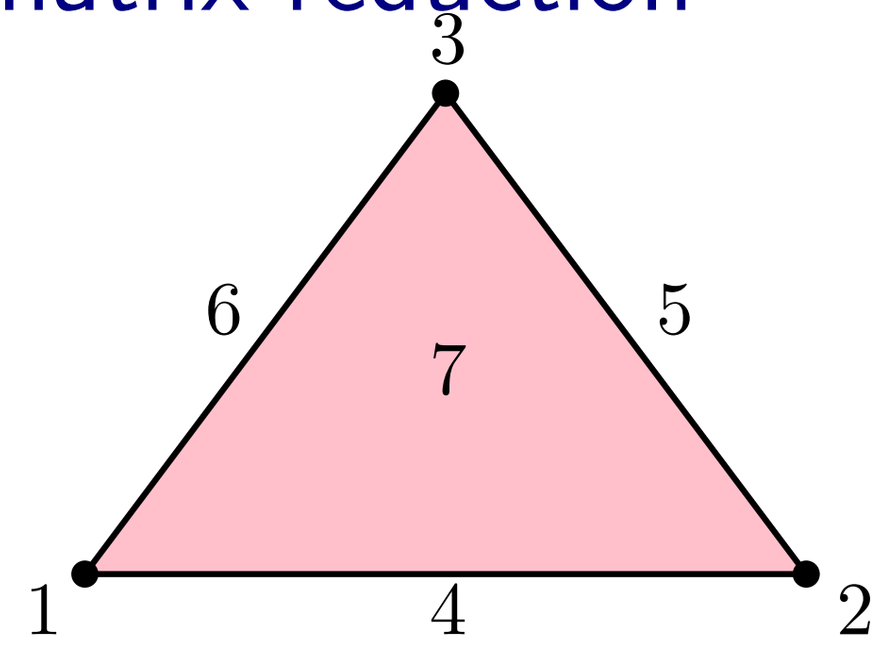
low$(j) = j'$



Computation with filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix

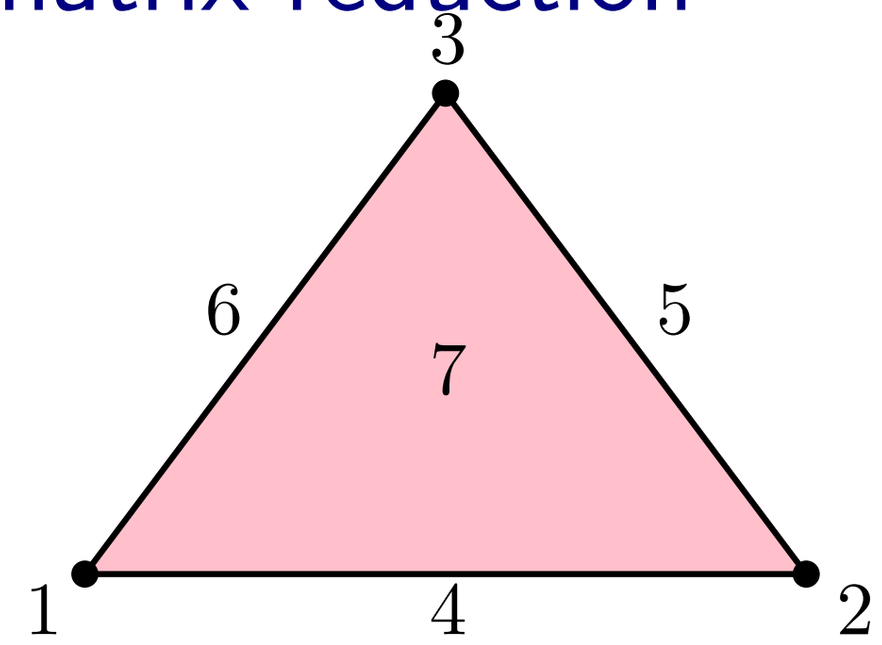


	1	2	3	4	5	6	7
1				*		*	
2				*	*		
3					*	*	
4							*
5							*
6							*
7							

Computation with filtrations and matrix reduction

Input: simplicial filtration

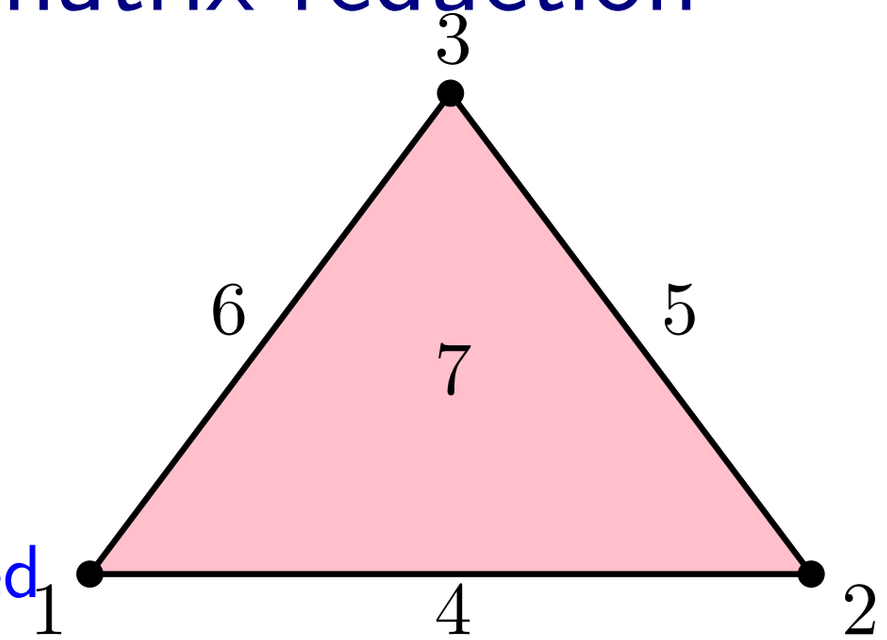
Output: boundary matrix
reduced to column-echelon form



	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 filtrations and matrix reduction



Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

◯ some positive-negative simplices are paired
$[2,4)$, $[3,5)$, $[6,7)$

□ unpaired simplices provide homology basis: $[1,+\infty)$

	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 filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

Q: Complexity?

Computation with filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

Q: Complexity?

PLU factorization:

- Gaussian elimination
- fast matrix multiplication (divide-and-conquer)
- random projections?

Computation with filtrations and matrix reduction

Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

Q: Complexity?

PLU factorization:

- Gaussian elimination

- PLEX / JavaPLEX (http://appliedtopology.github.io/javaplex/)

- Dionysus (http://www.mrzv.org/software/dionysus/)

- Perseus (http://www.sas.upenn.edu/~vnanda/perseus/)

- Gudhi (http://gudhi.gforge.inria.fr/)

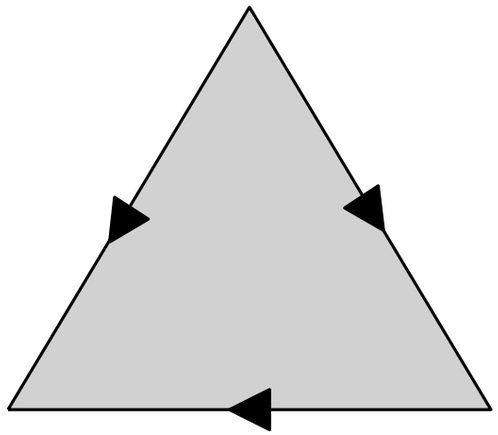
- PHAT (https://bitbucket.org/phat-code/phat)

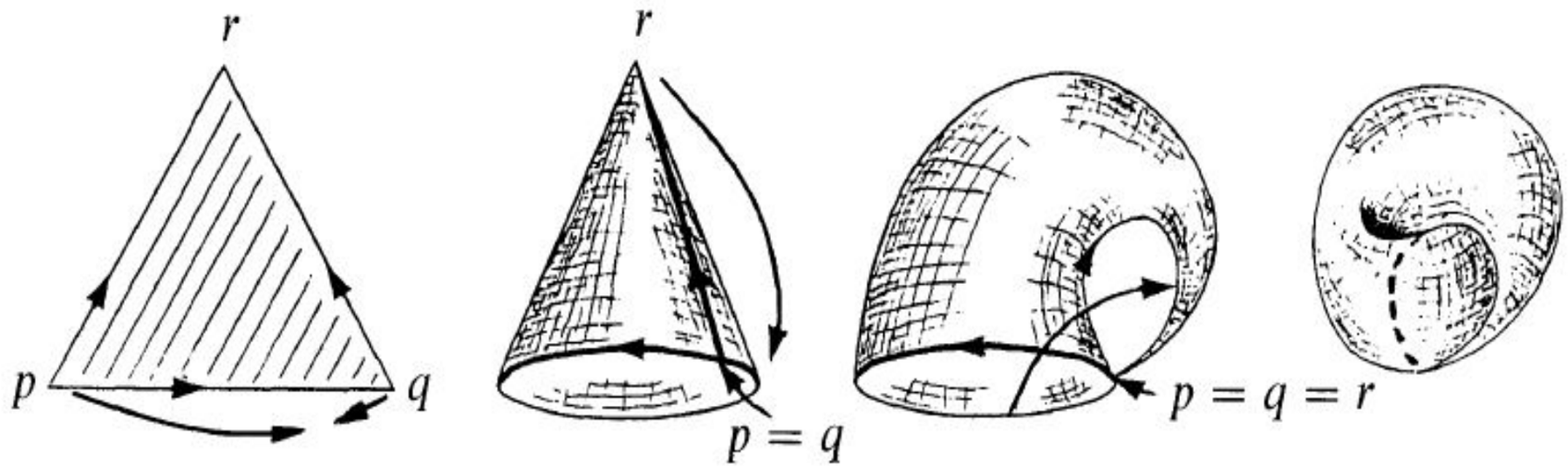
- DIPHA (https://github.com/DIPHA/dipha/)

- CTL (https://github.com/appliedtopology/ctl)

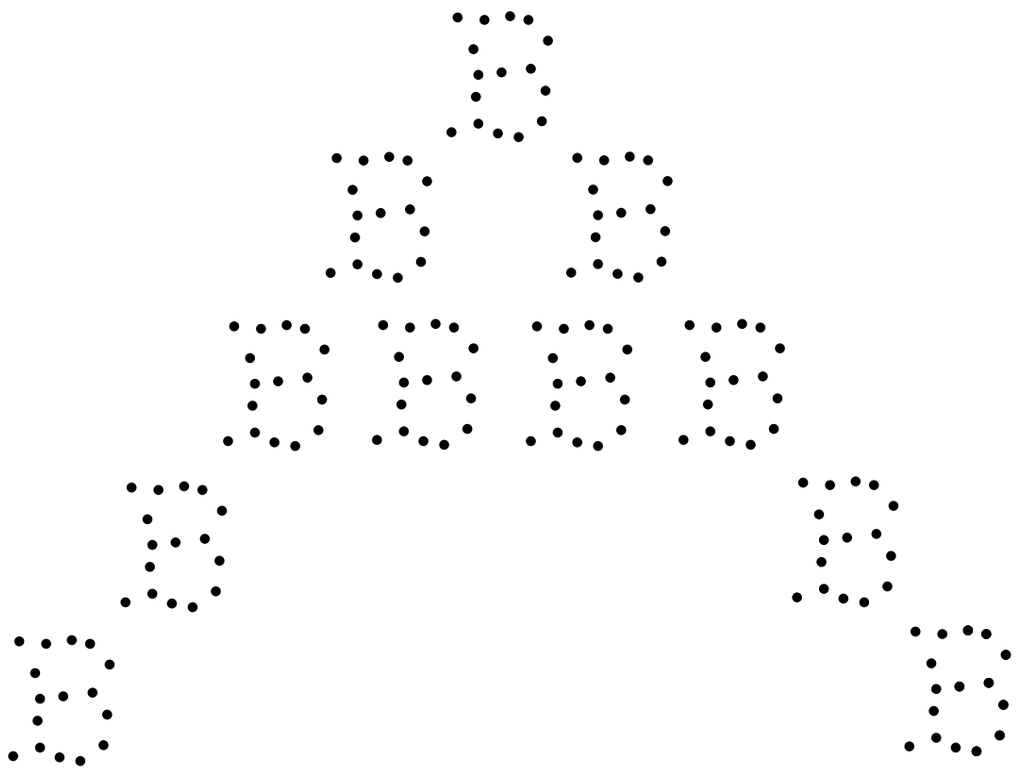
Computation with filtrations and matrix reduction

Q: Triangulate and compute homology of dunce cap:



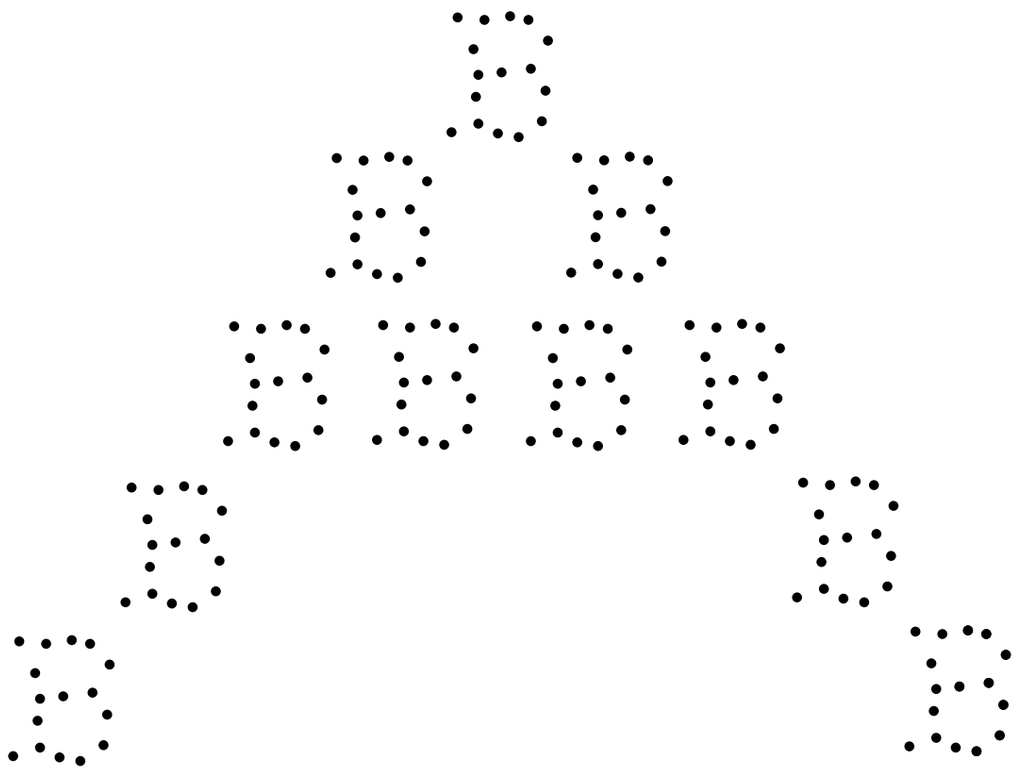


Problems with homology



First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Problems with homology

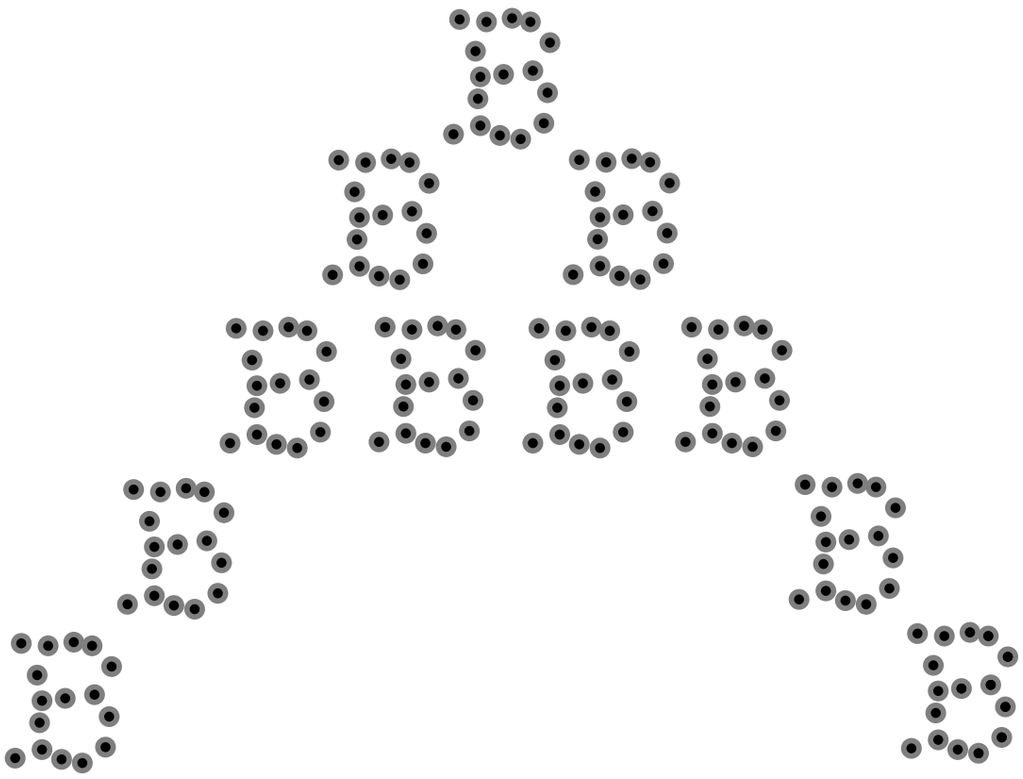


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology

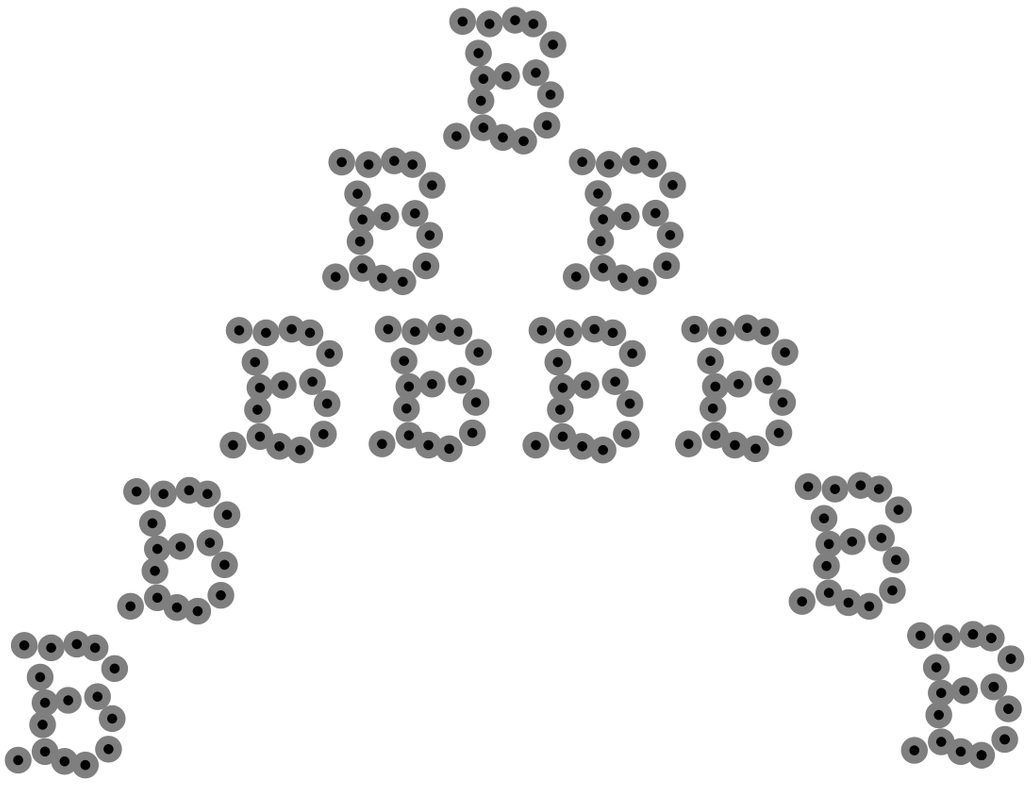


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology

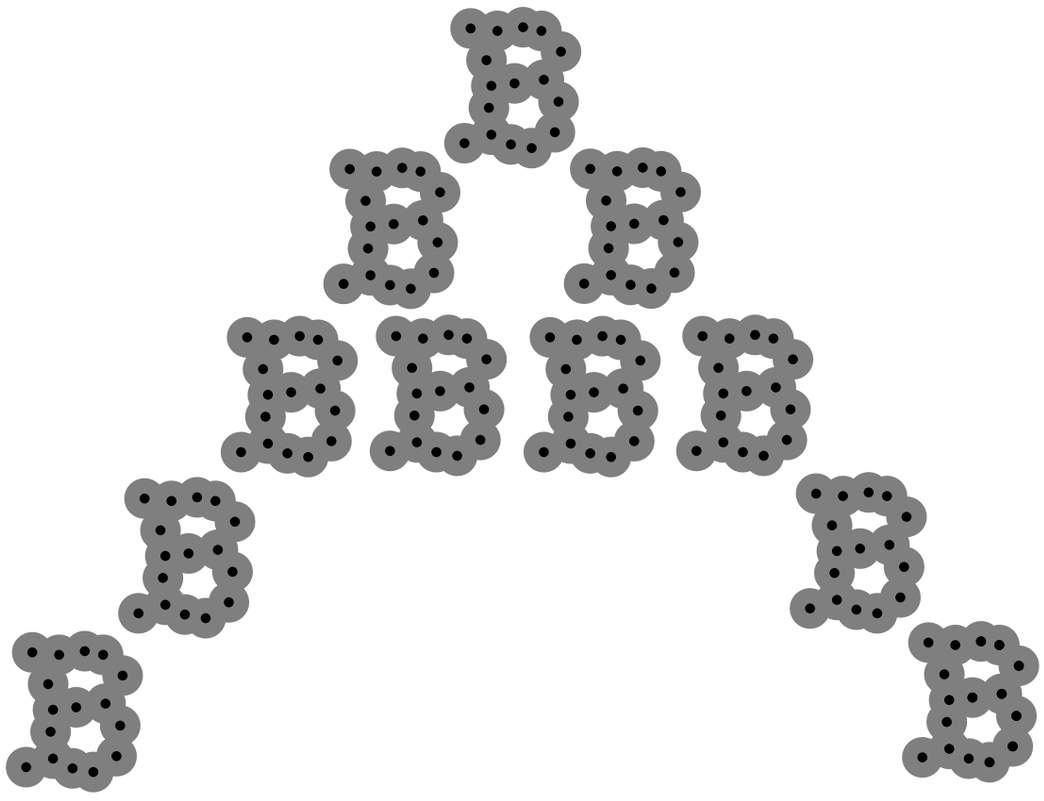


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology

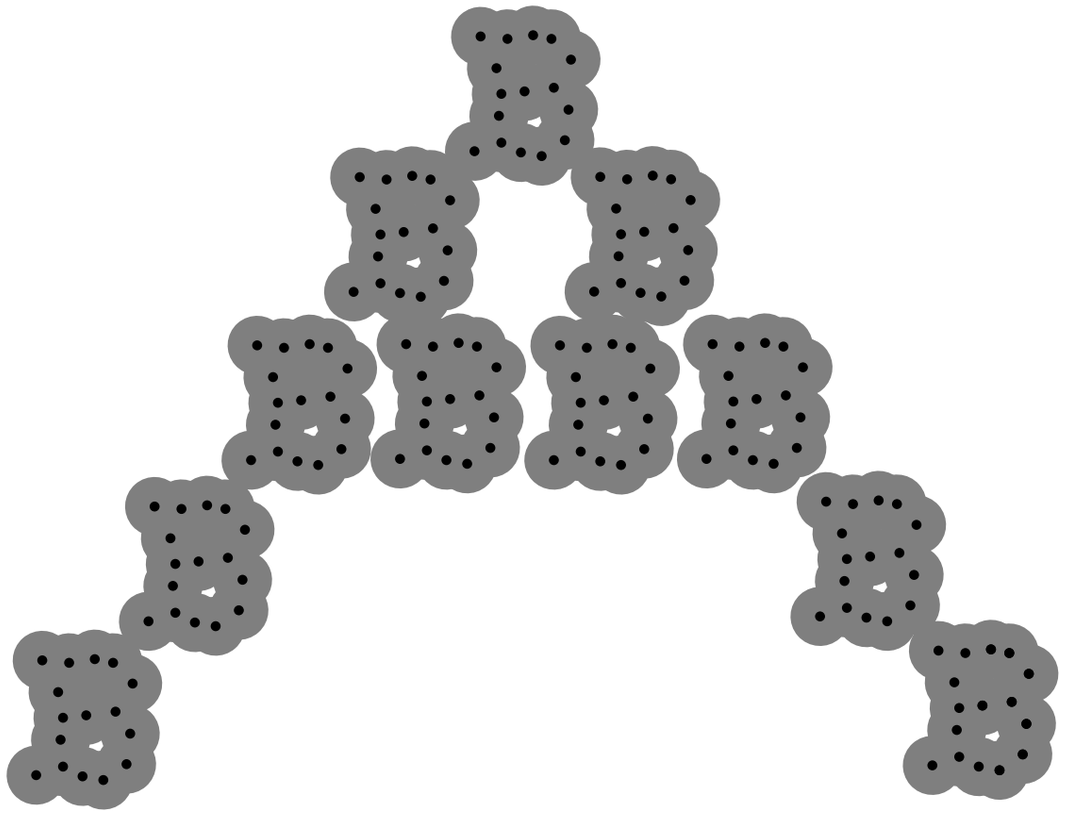


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology

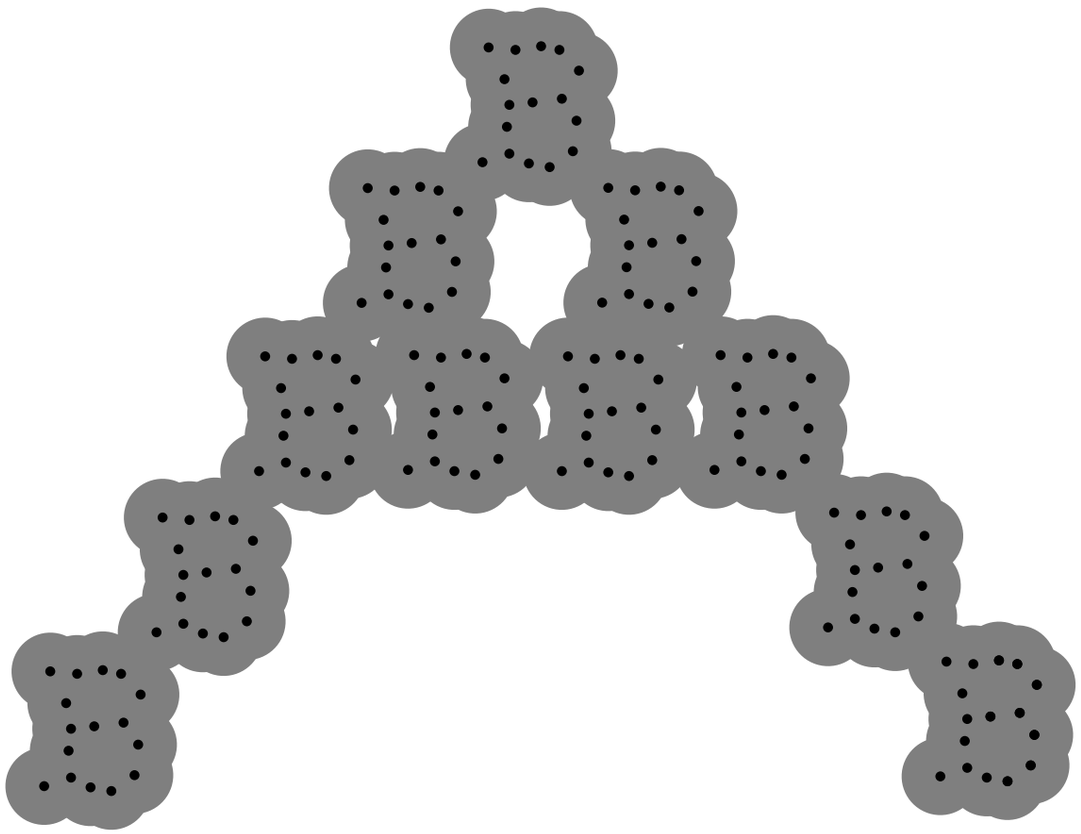


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology

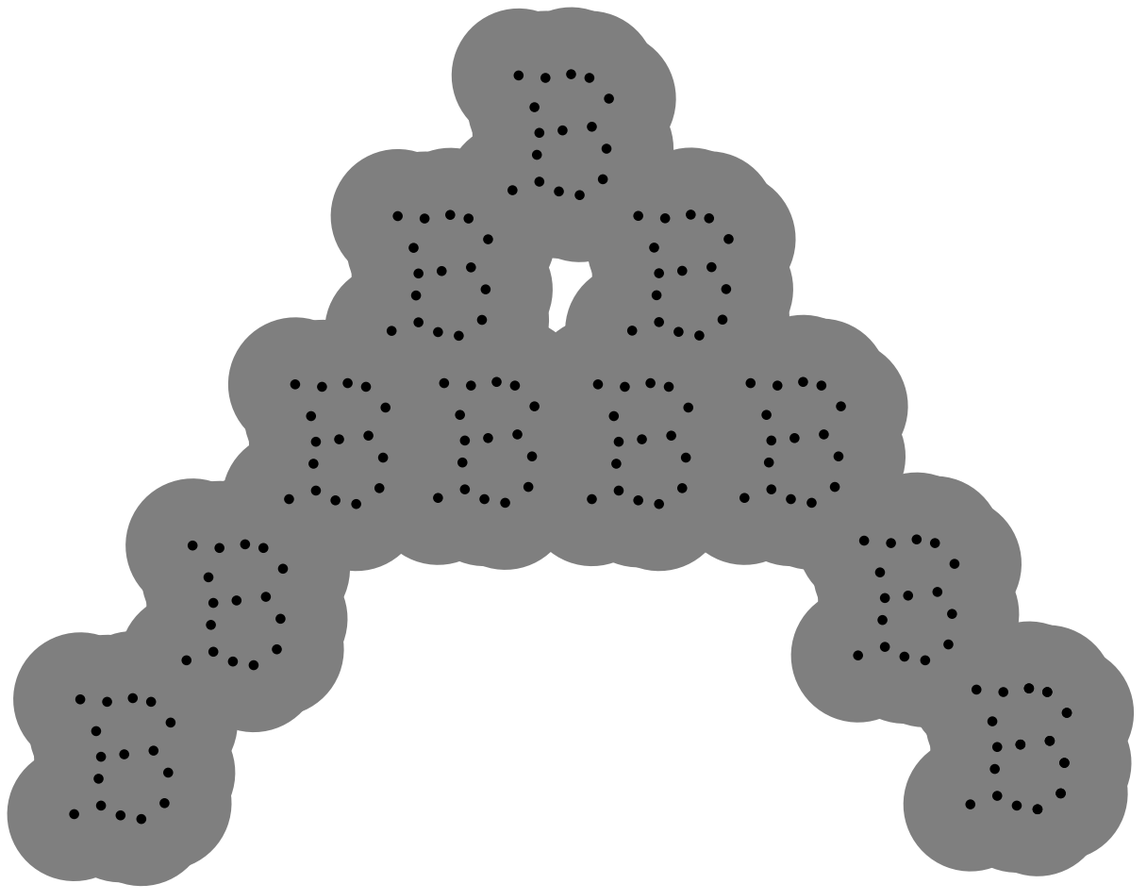


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology

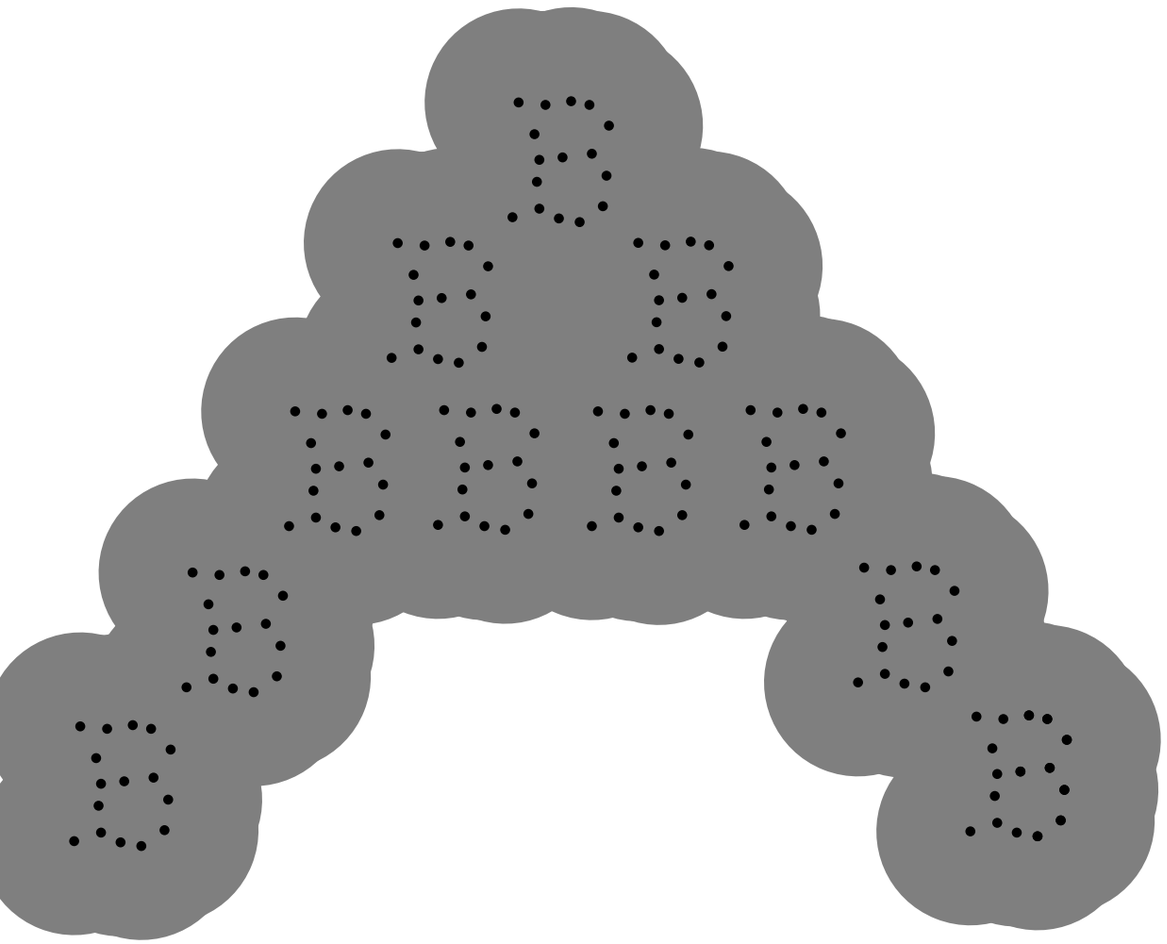


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology

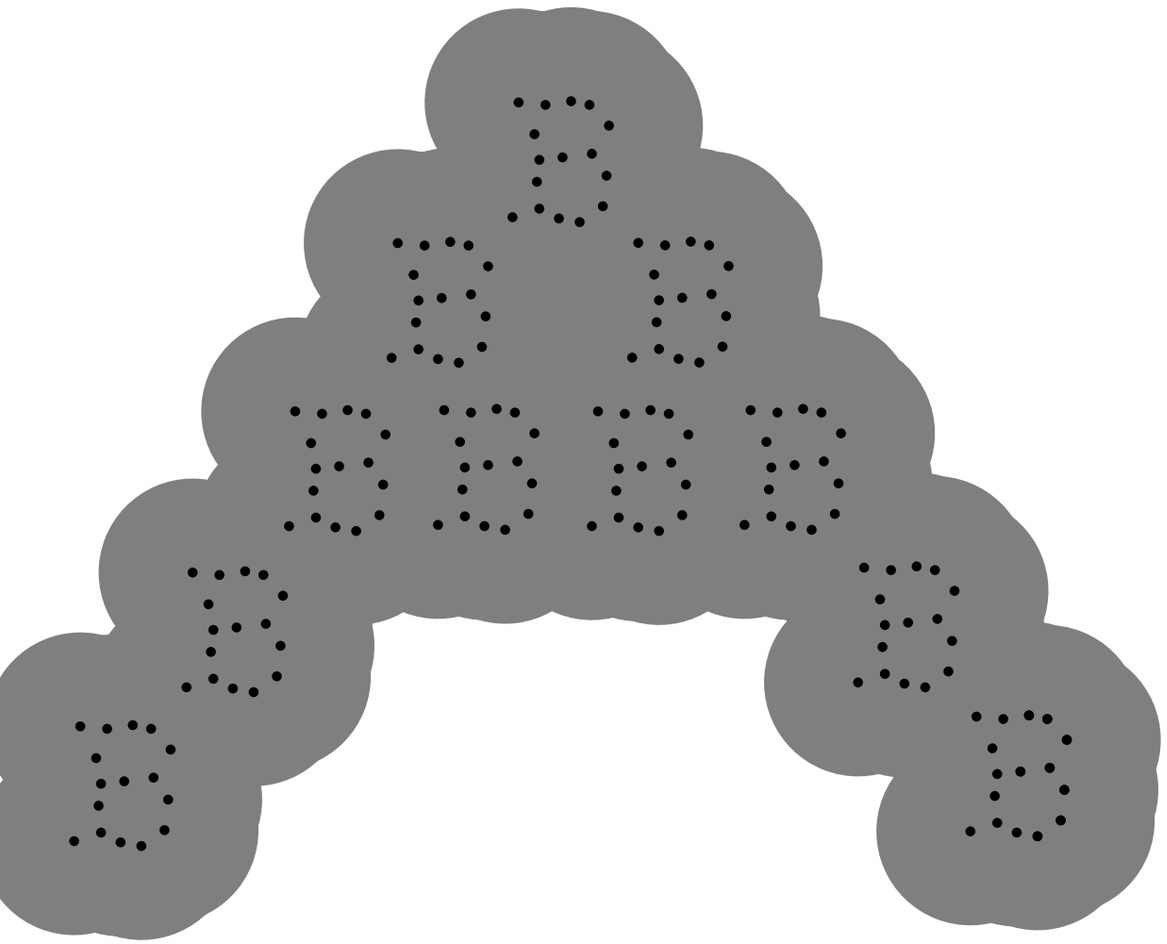


First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

Problems with homology



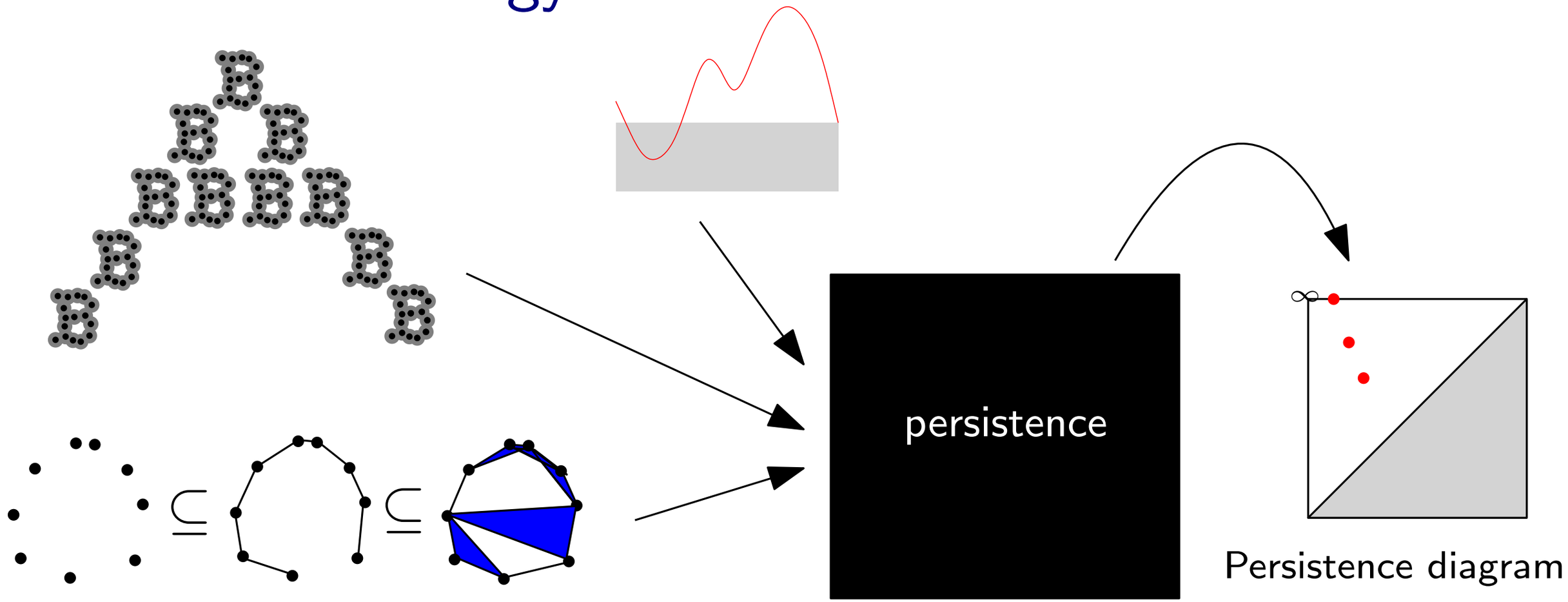
First, the algorithm for computing homology contains much more information than the mere homology of the last complex in the filtration.

Indeed, it contains the homology of *all* the subcomplexes in the filtration.

This is very interesting in the sense that data can be analyzed at *multiple* scales.

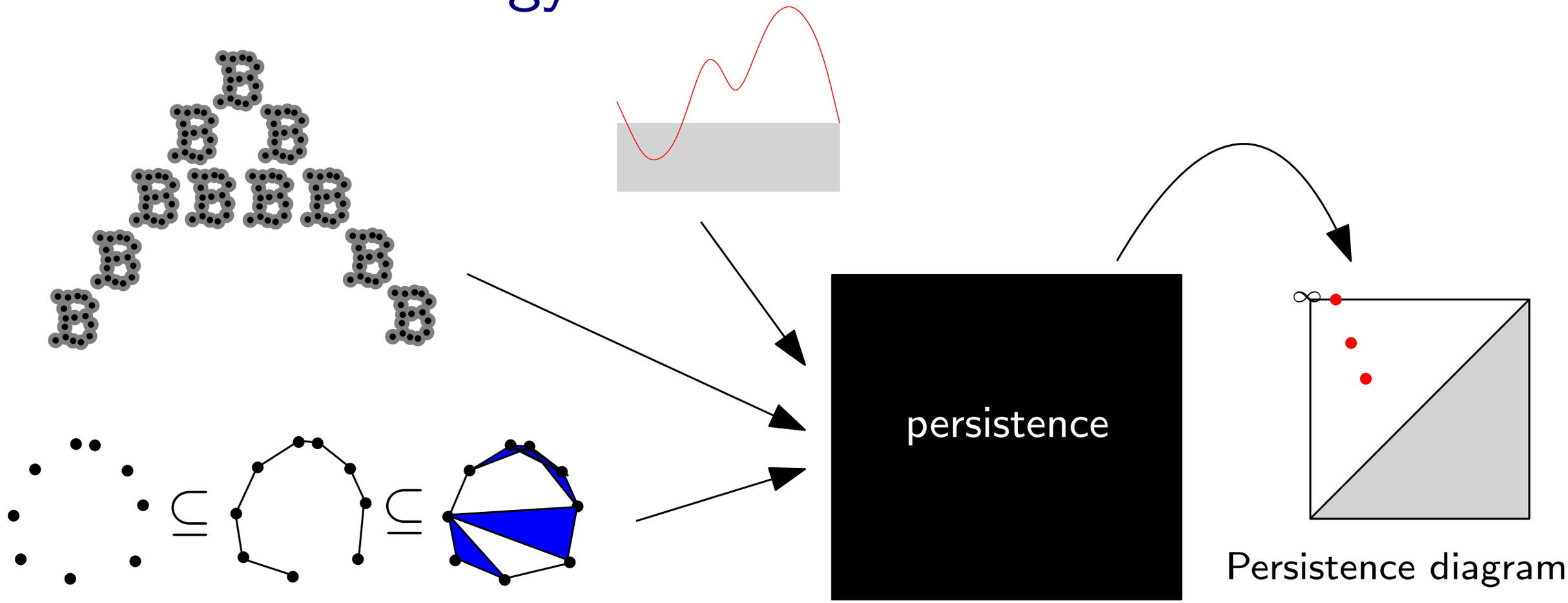
Persistent homology aims at encoding the homology of the complex at *all possible scales* into a compact descriptor.

Persistent homology



What is persistent homology?

Persistent homology

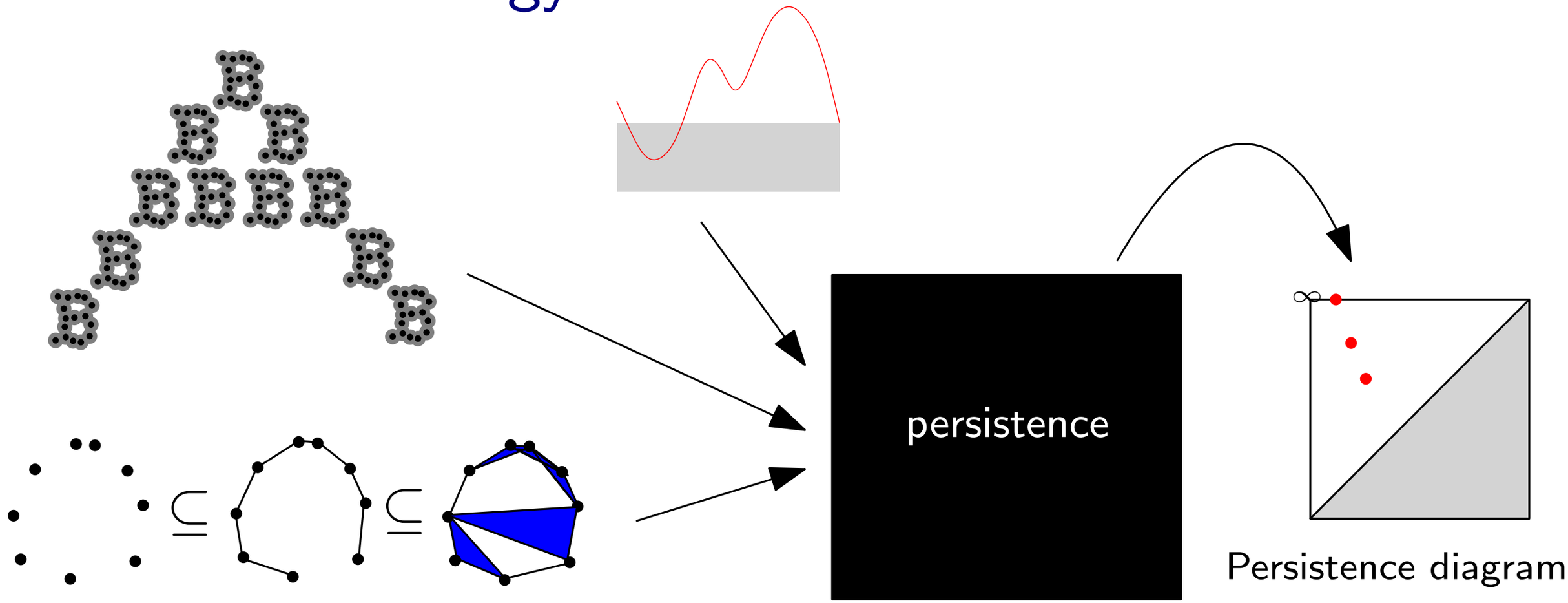


What is persistent homology?

$\rightarrow$ a mathematical framework for encoding the evolution of the homology of filtrations of simplicial complexes (it also works for general filtered spaces).

$\rightarrow$ formalized by H. Edelsbrunner et al. (2002) and G. Carlsson et al. (2005) with wide developments during the last decade.

Persistent homology



What is persistent homology?

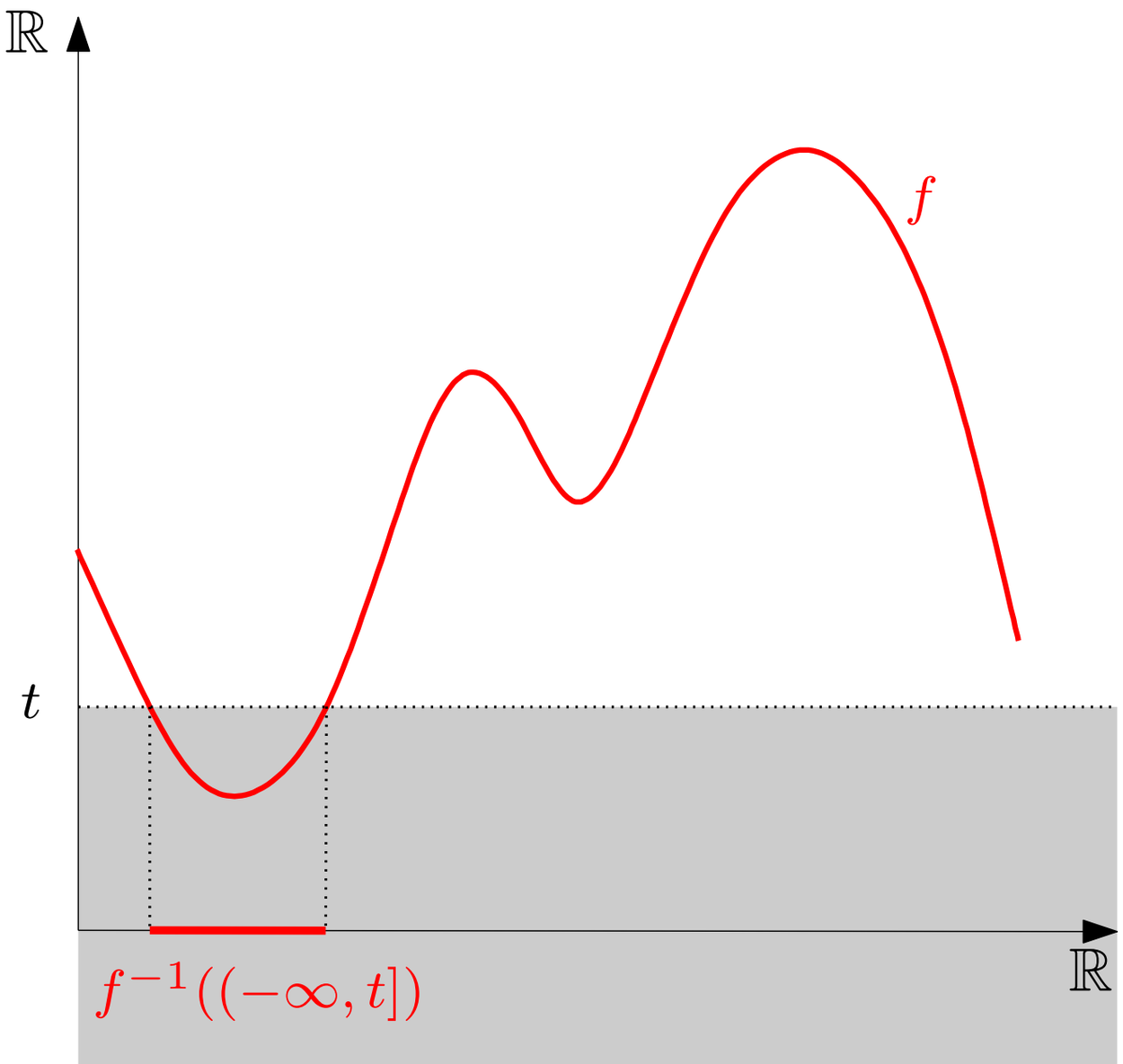
$\rightarrow$ barcodes/persistence diagrams can be efficiently computed.

$\rightarrow$ multiscale topological information.

$\rightarrow$ stability properties.

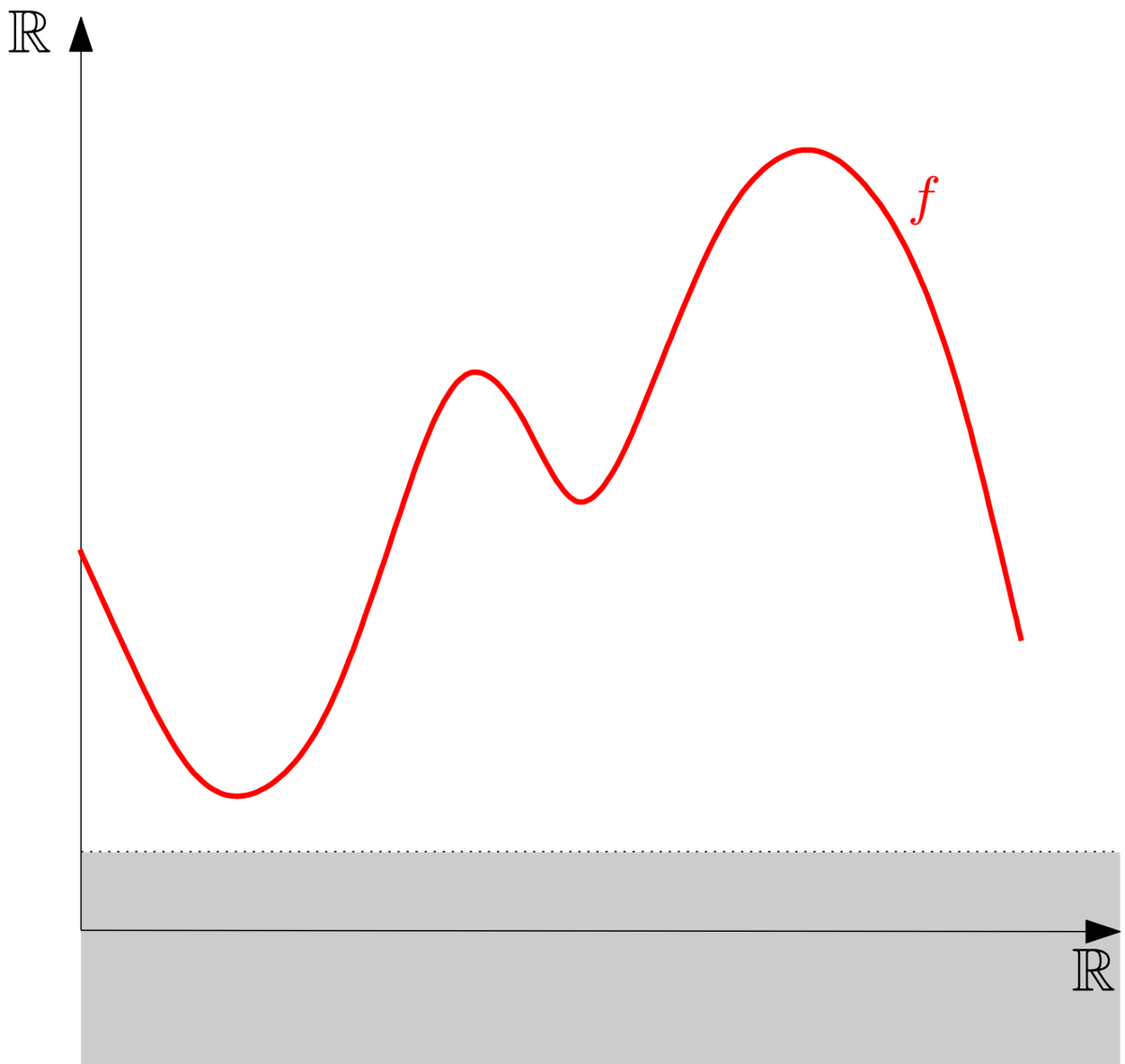
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



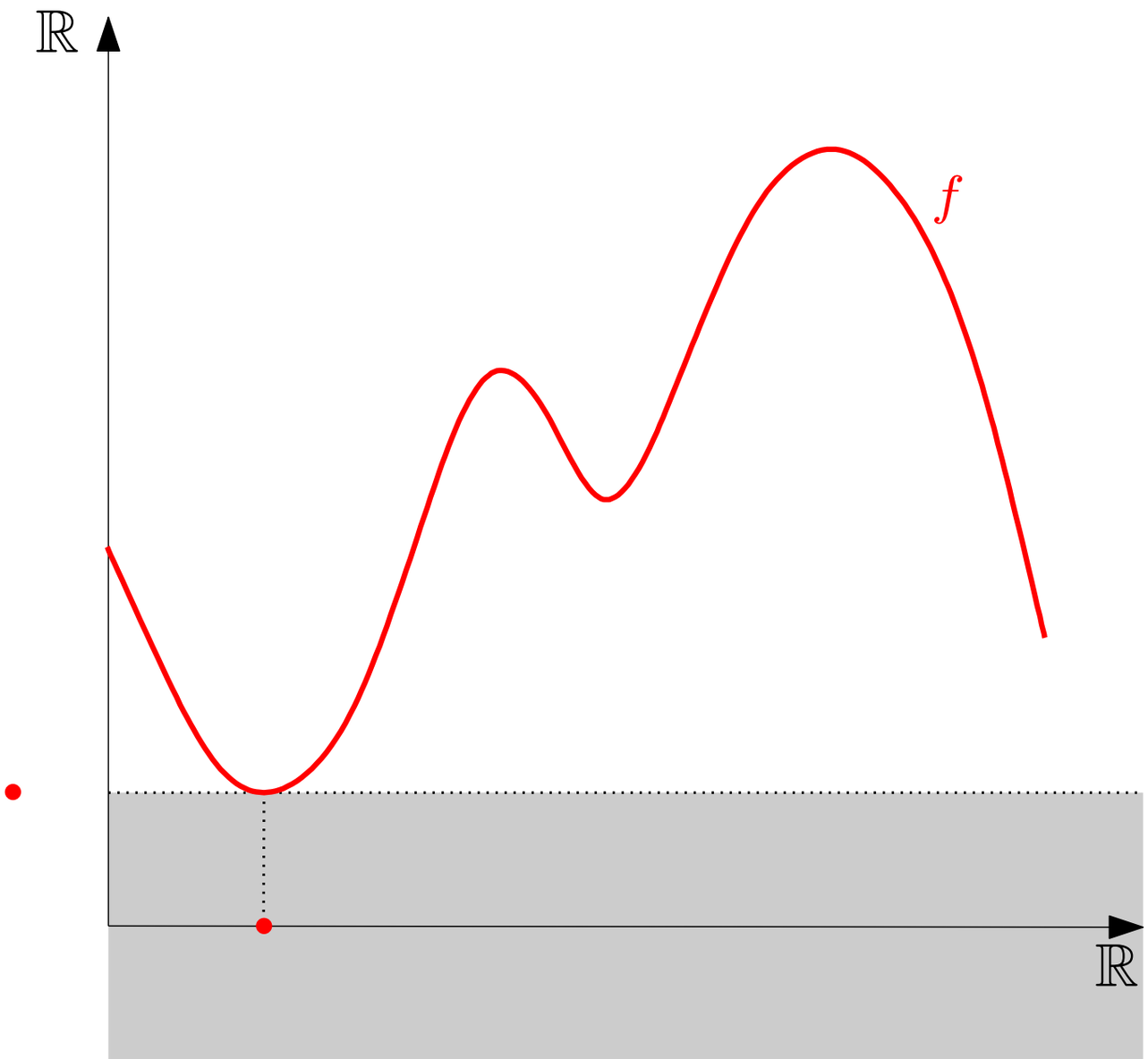
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



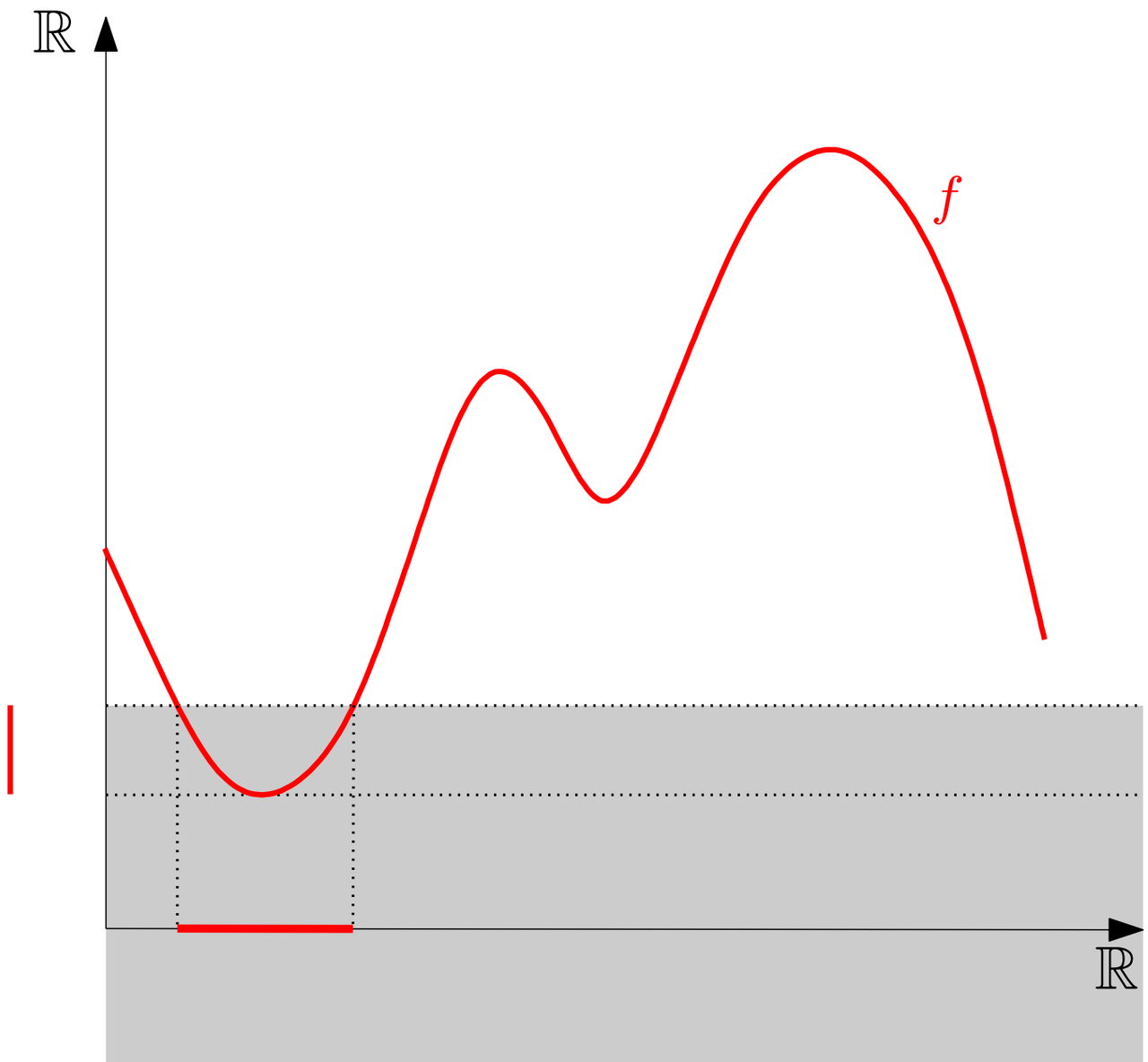
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



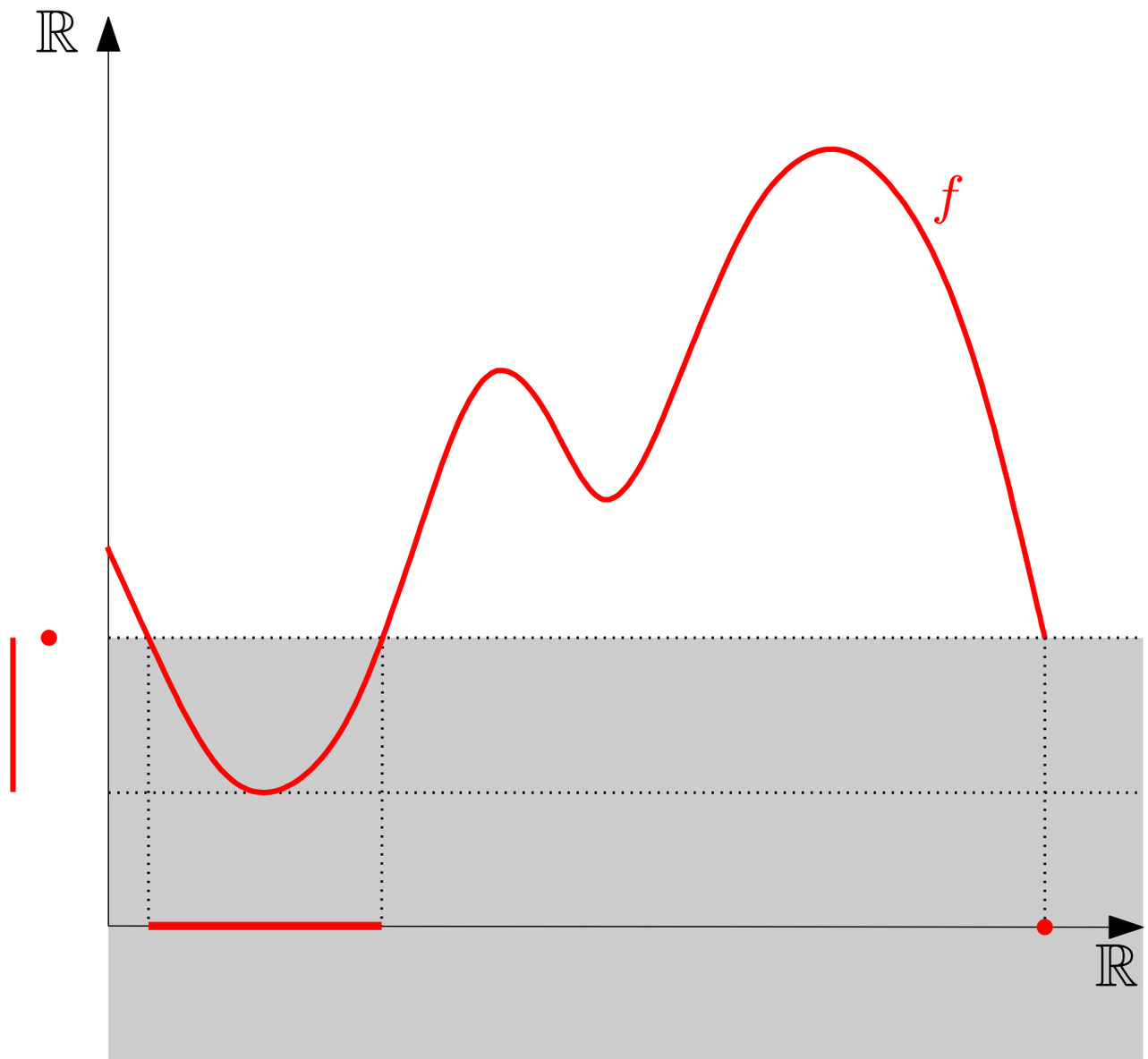
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



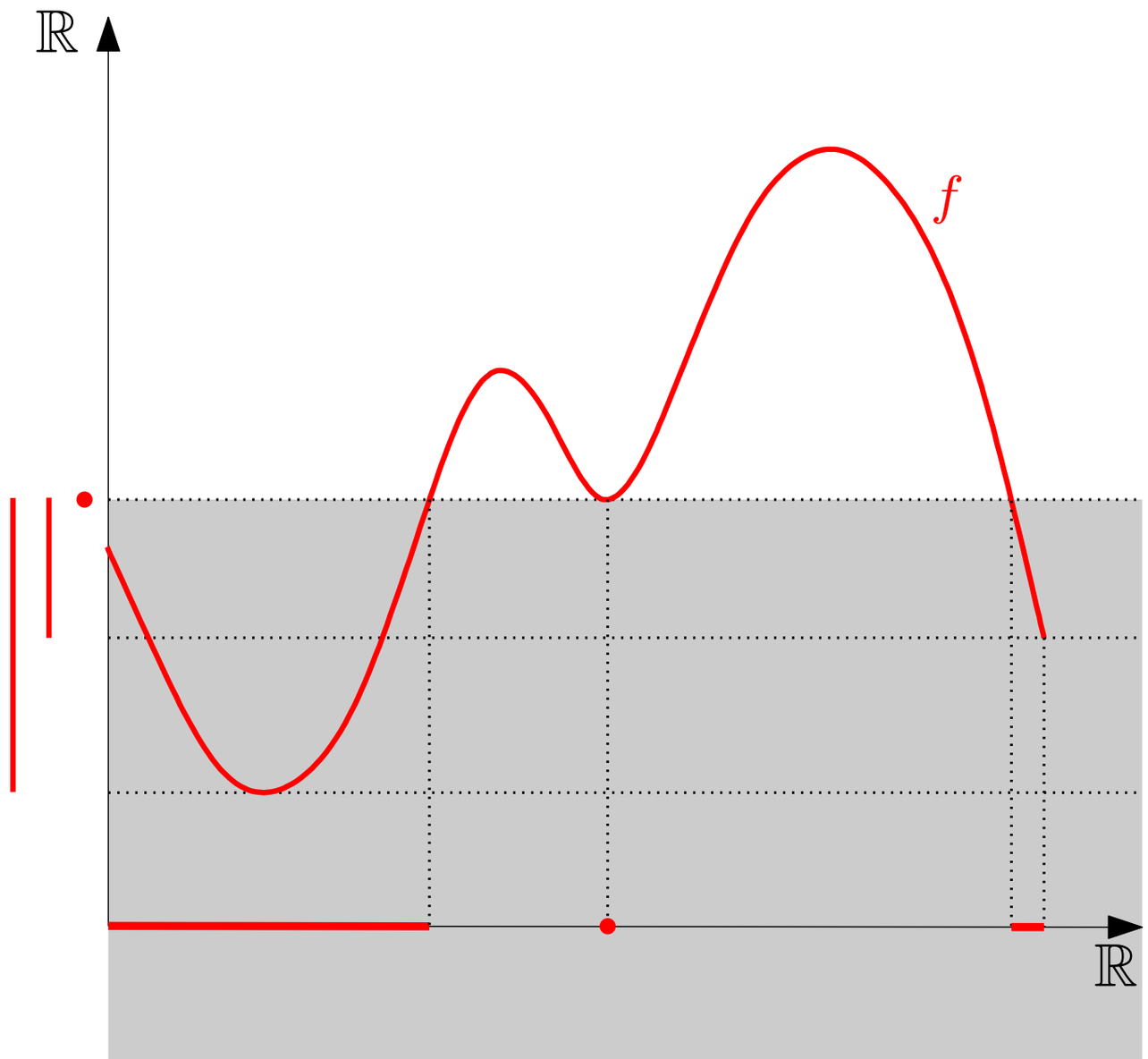
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



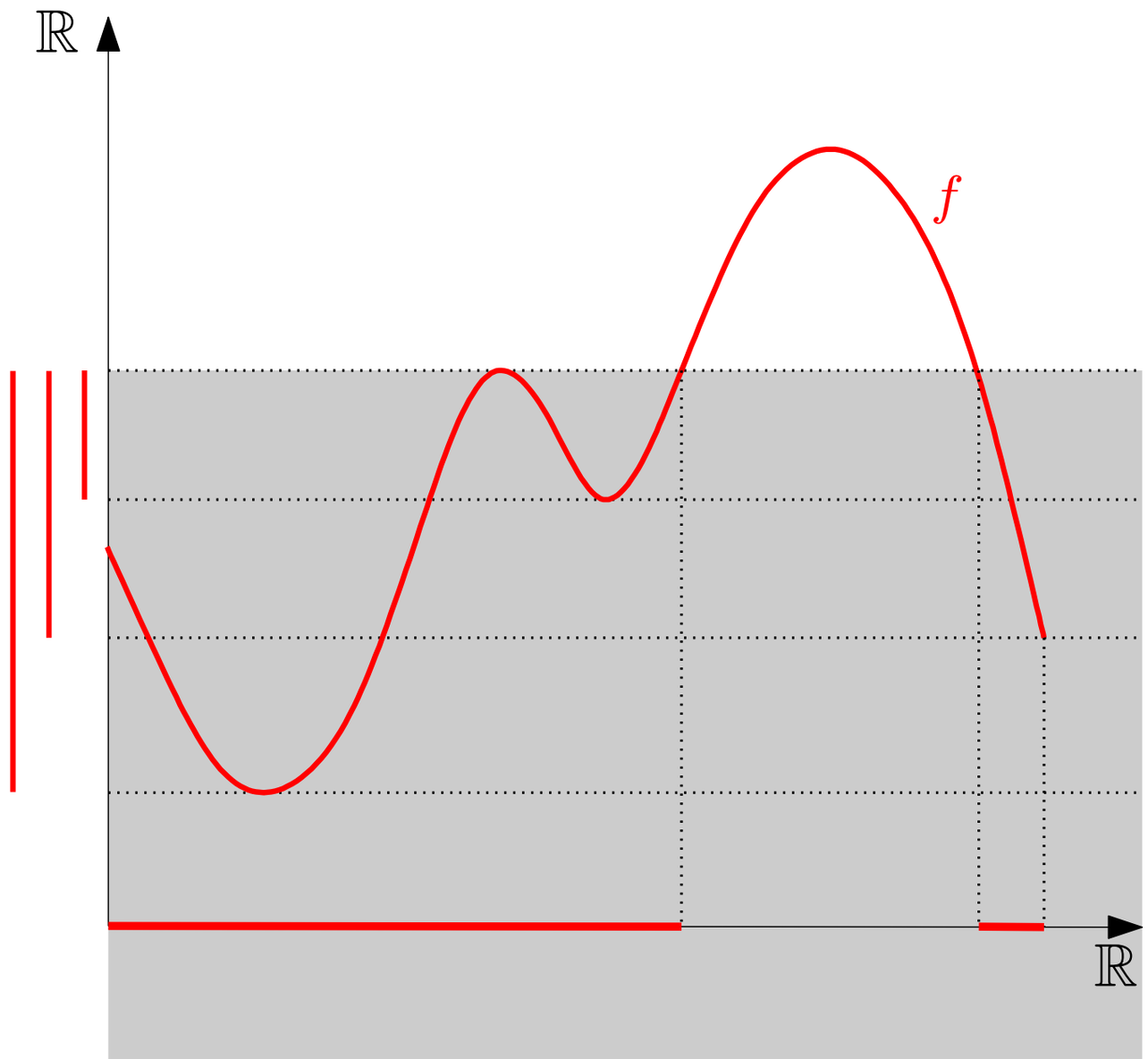
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



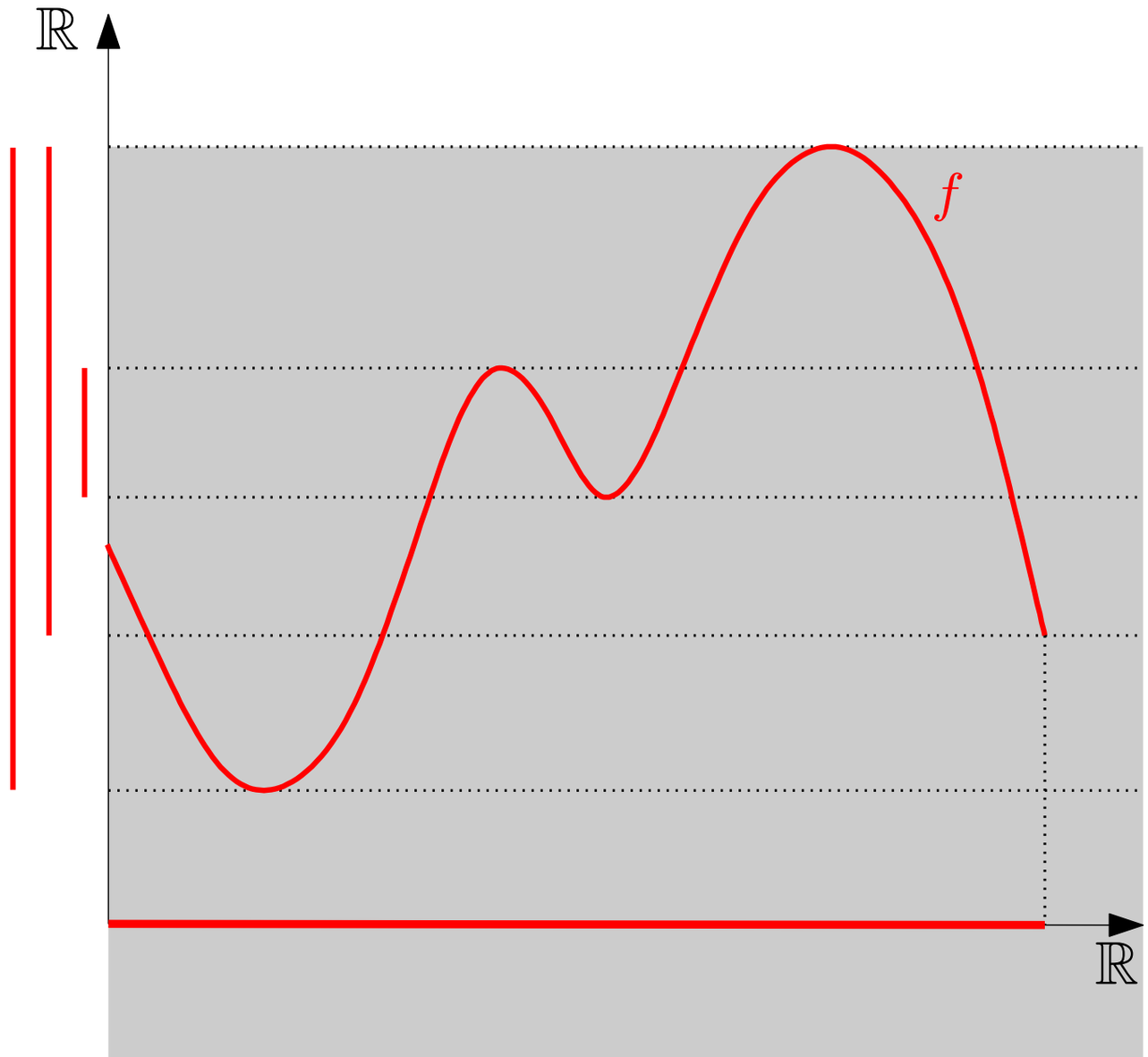
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



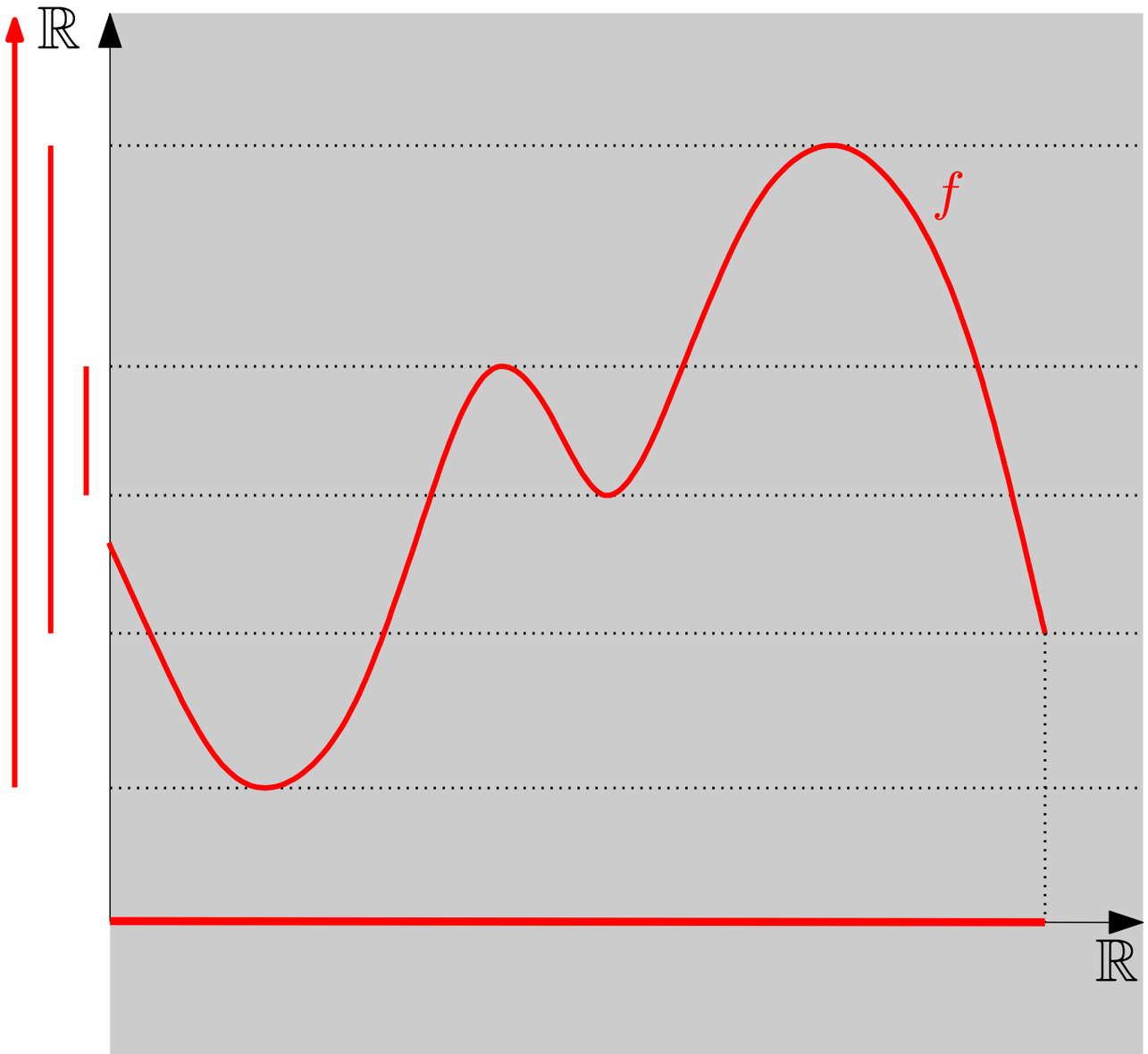
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family



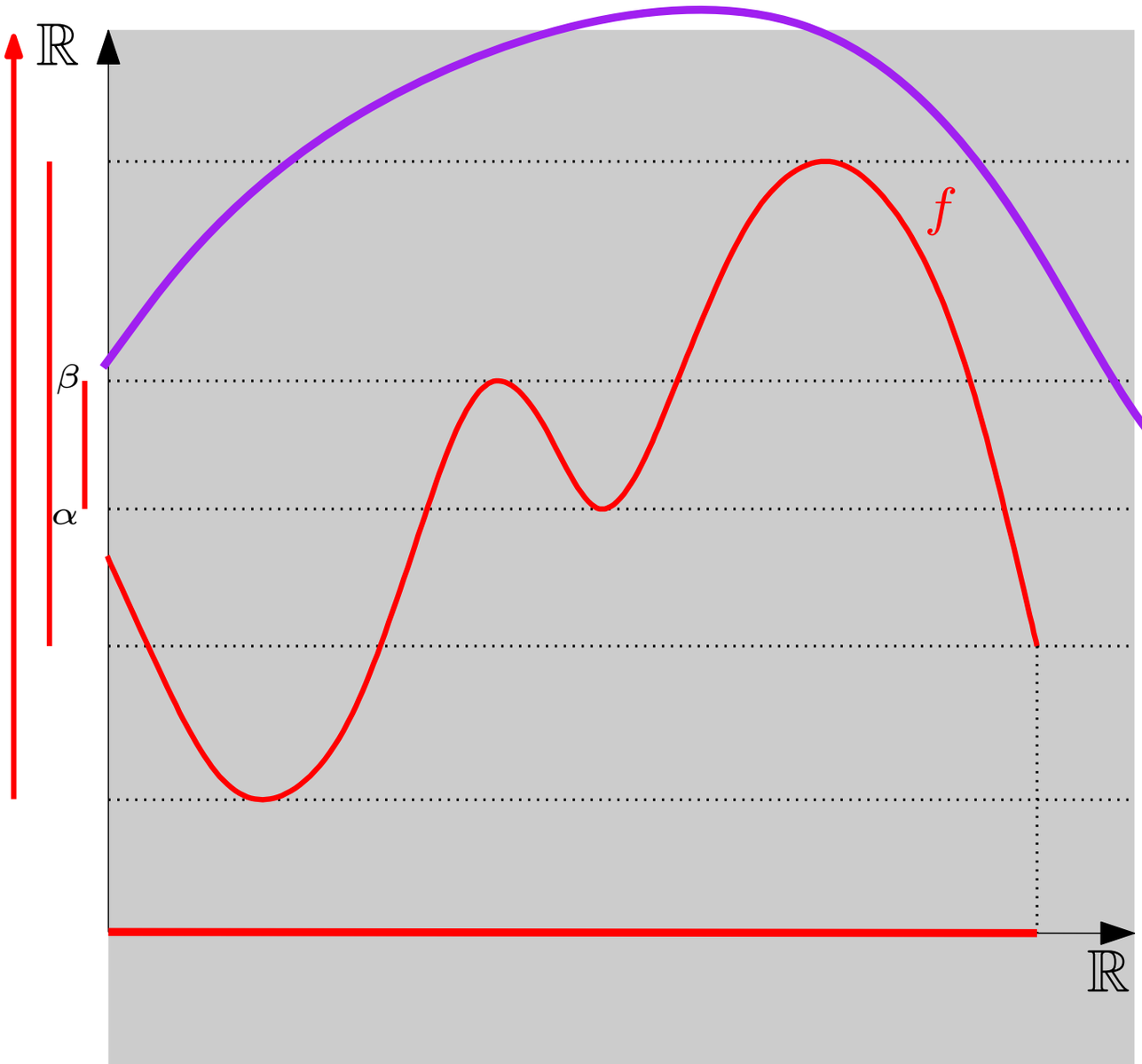
Example: persistence of sublevel sets of function

- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family
- finite set of intervals (barcode) encodes births/deaths of homology classes

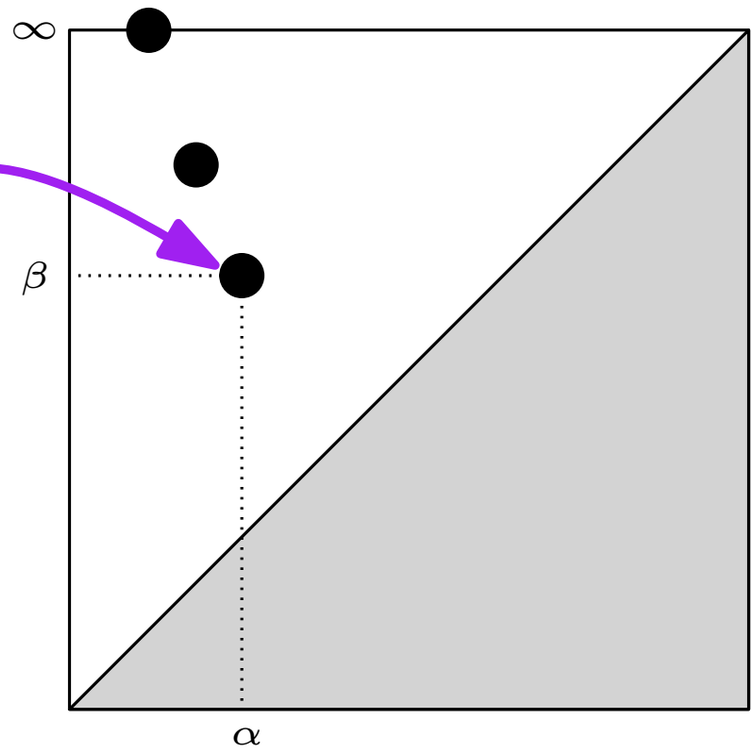


Example: persistence of sublevel sets of function

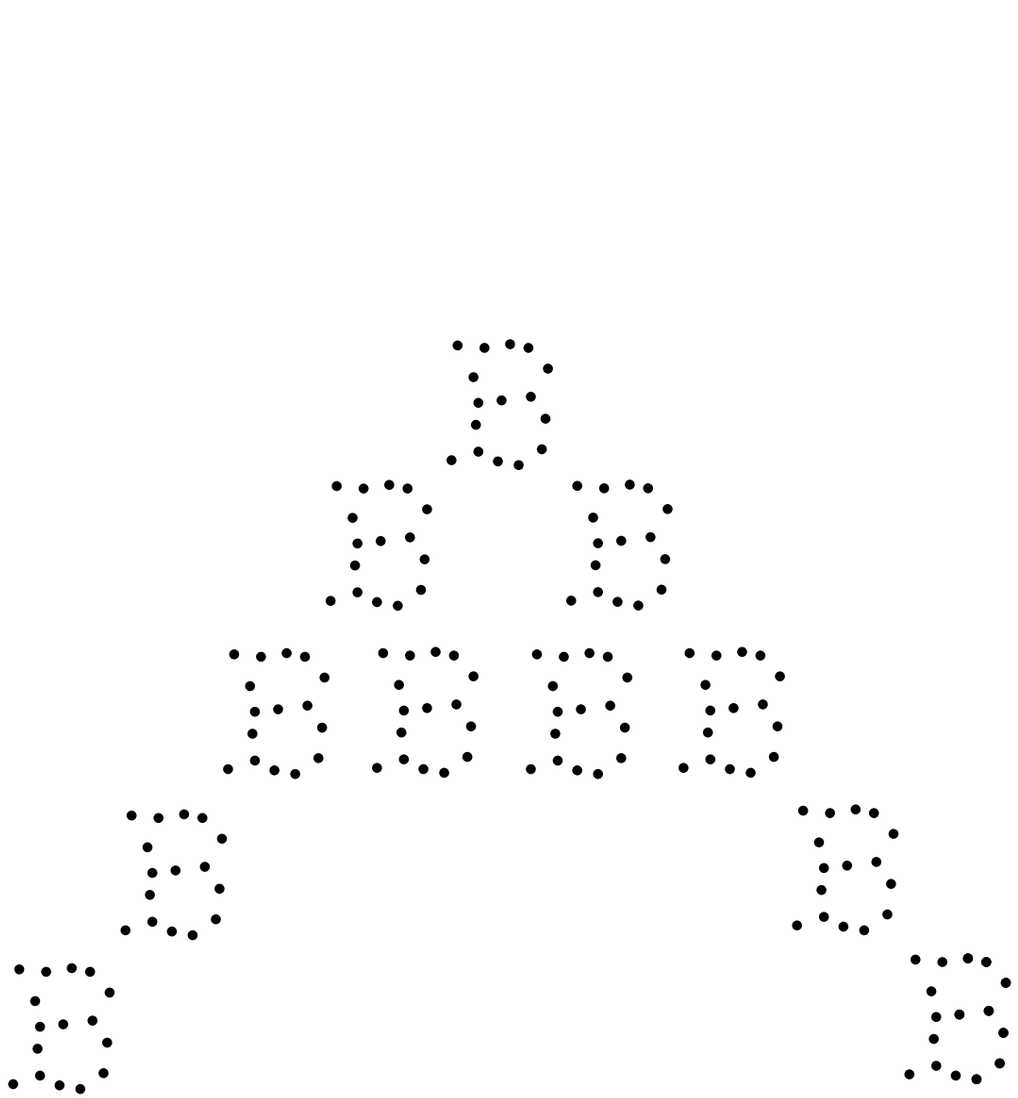
- input: *filtration* = nested family of sublevel-sets $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$
- track the evolution of the topology (homology) throughout the family
- finite set of intervals (barcode) encodes births/deaths of homology classes

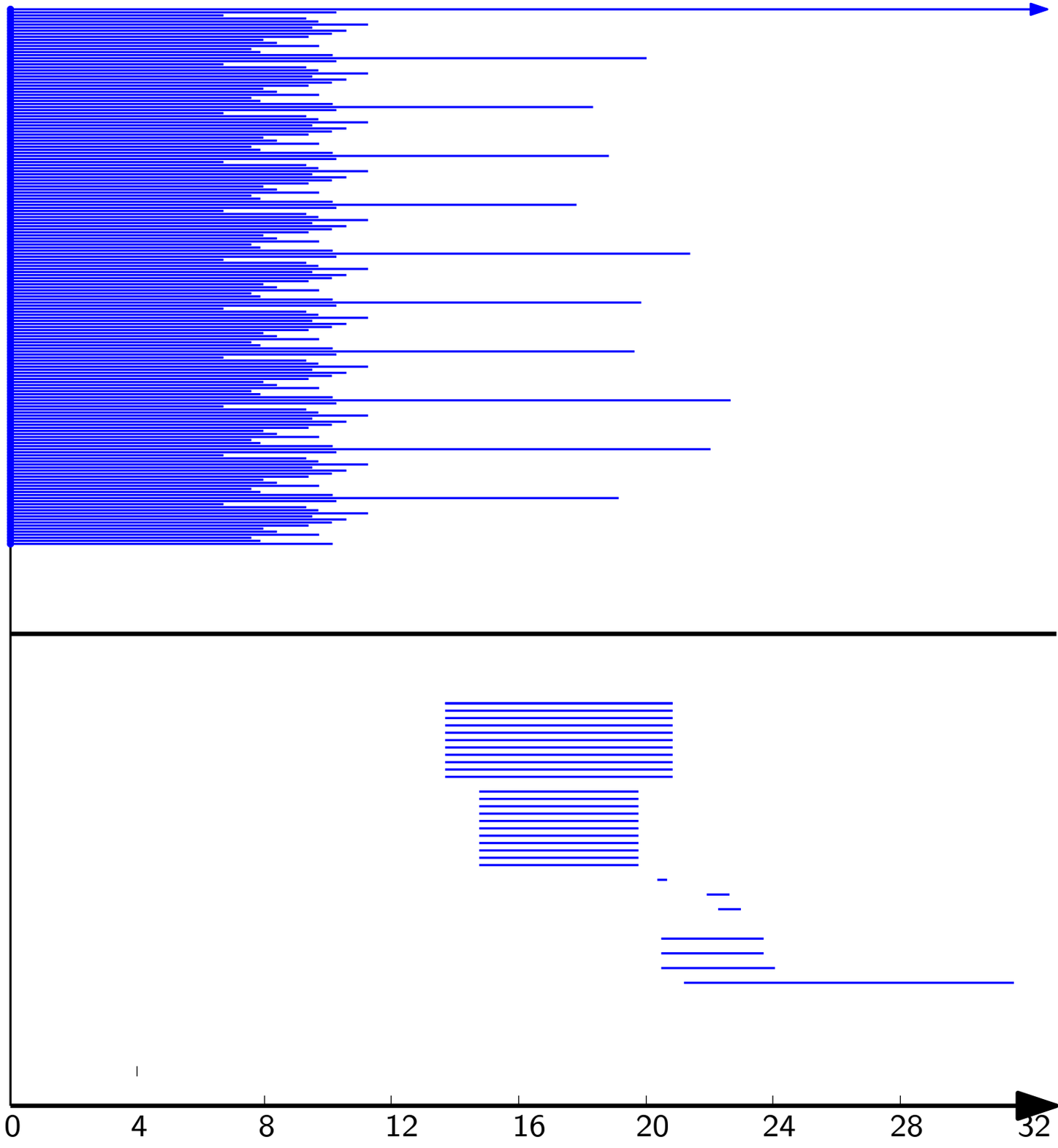


- alternate representation as a (multi-) set of points in the plane (*persistence diagram*).

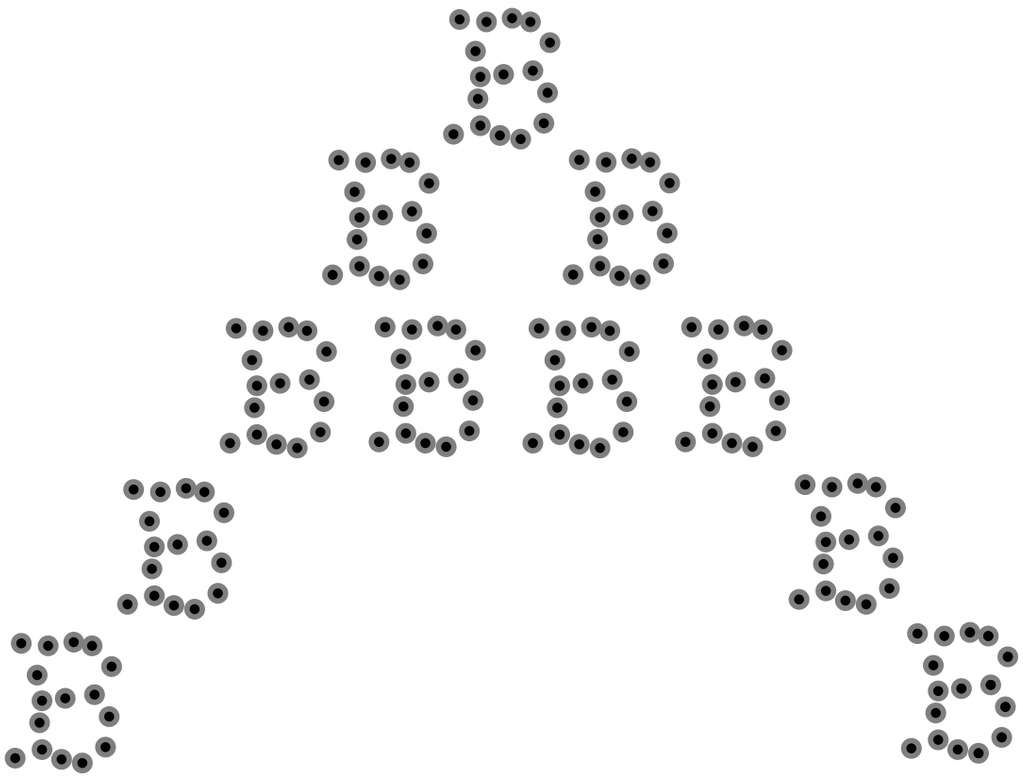


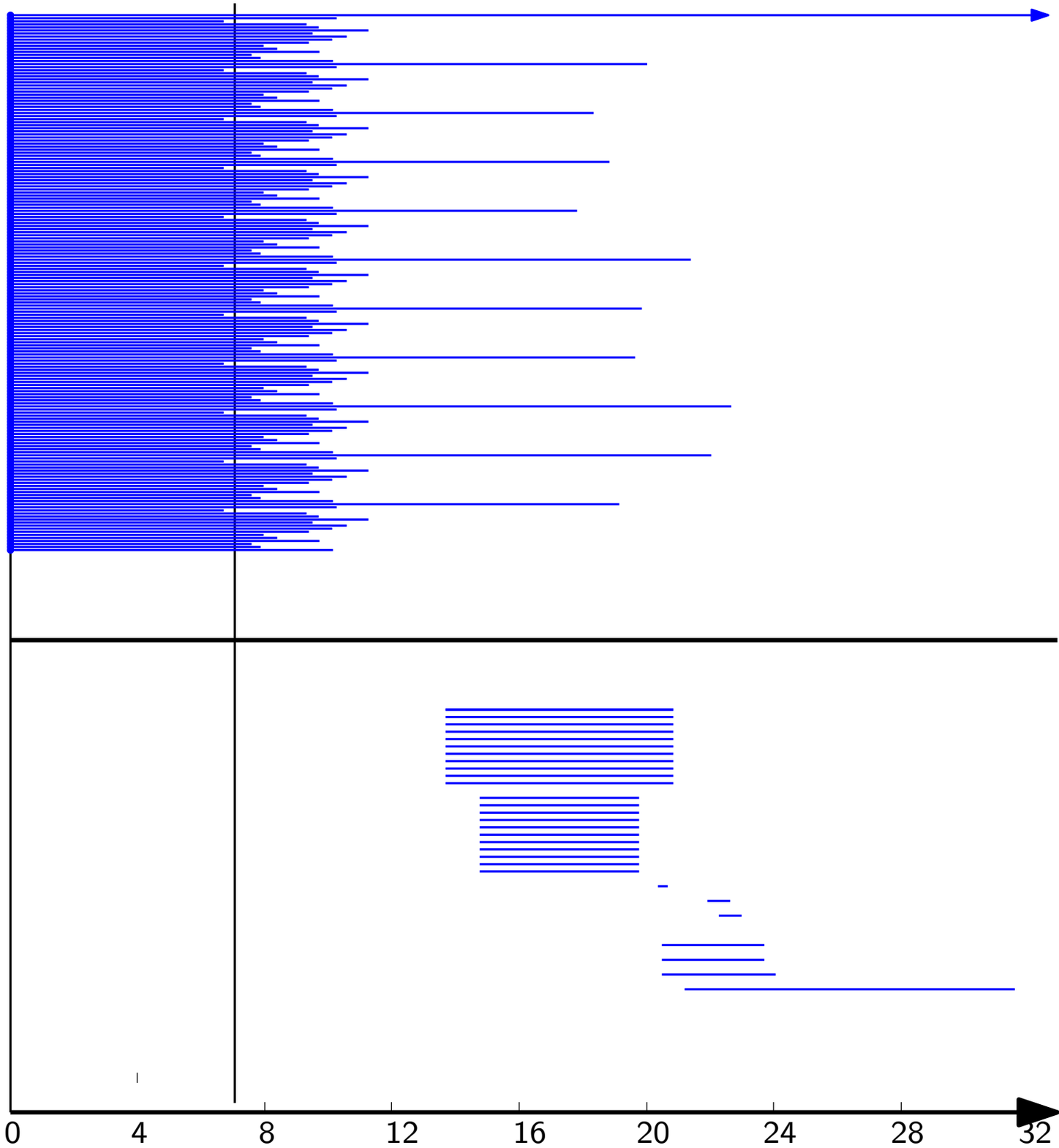
Example: persistence of Čech complexes



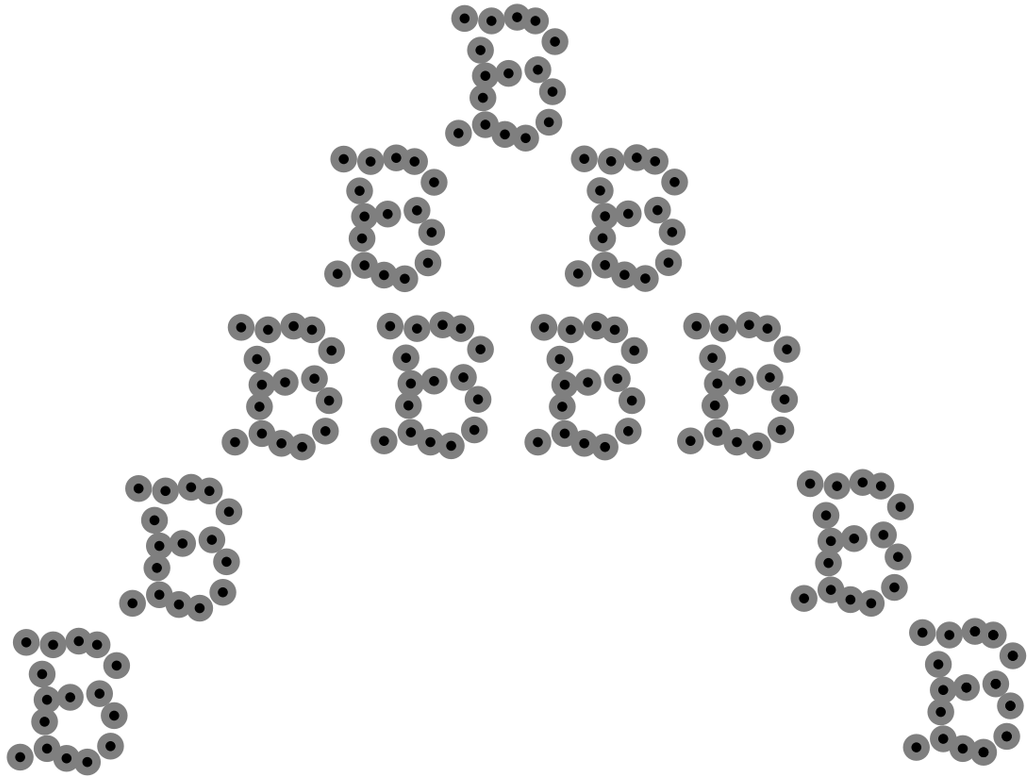


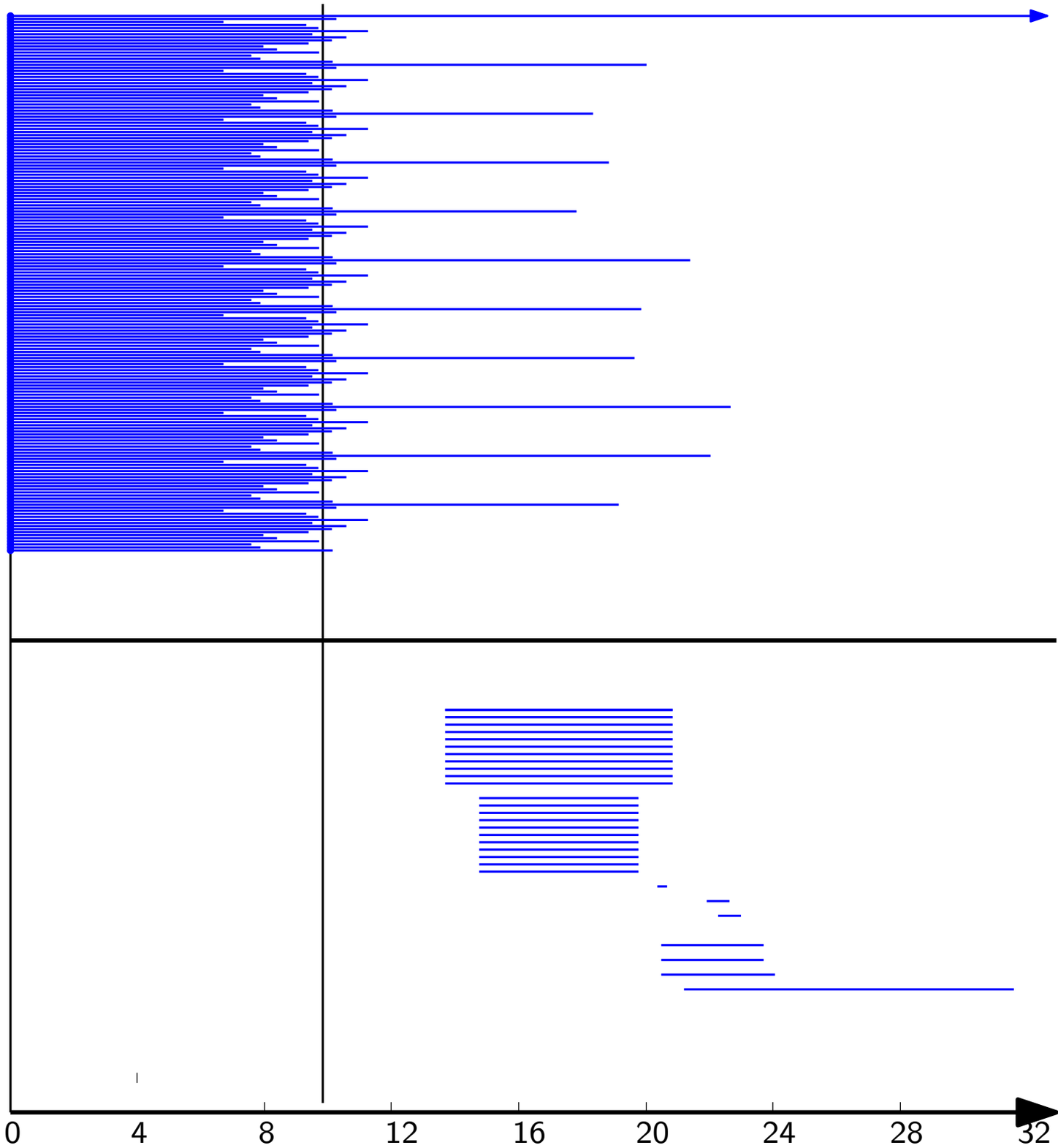
Example: persistence of Čech complexes



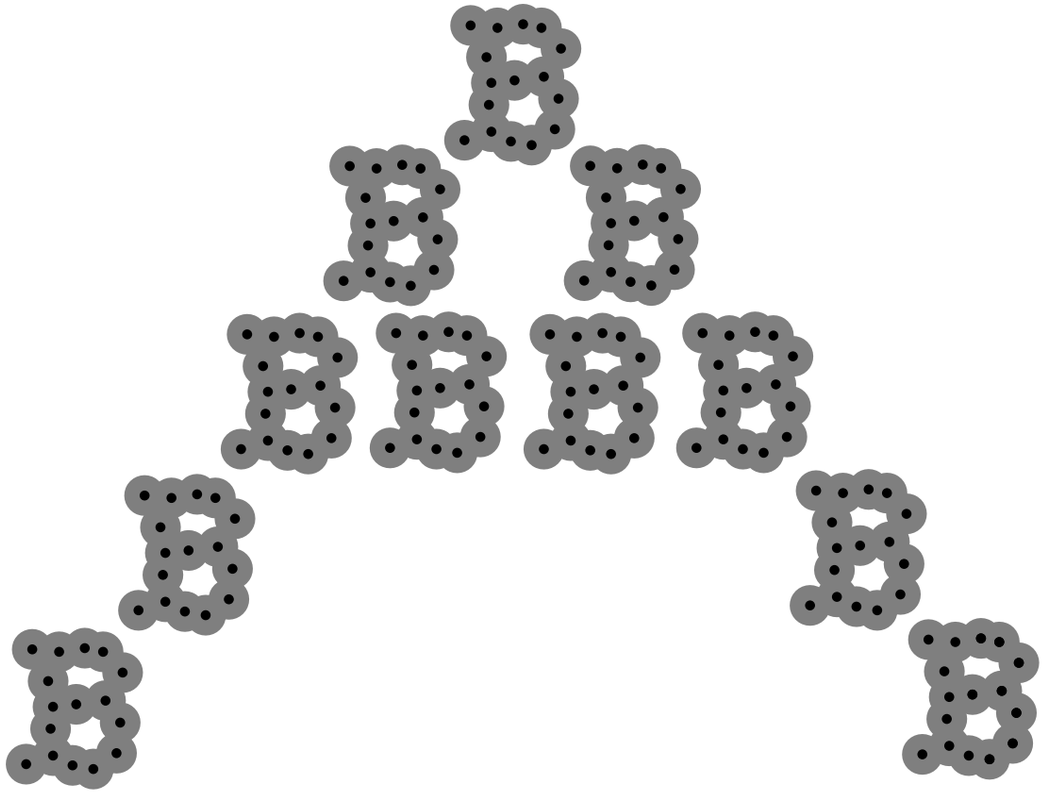


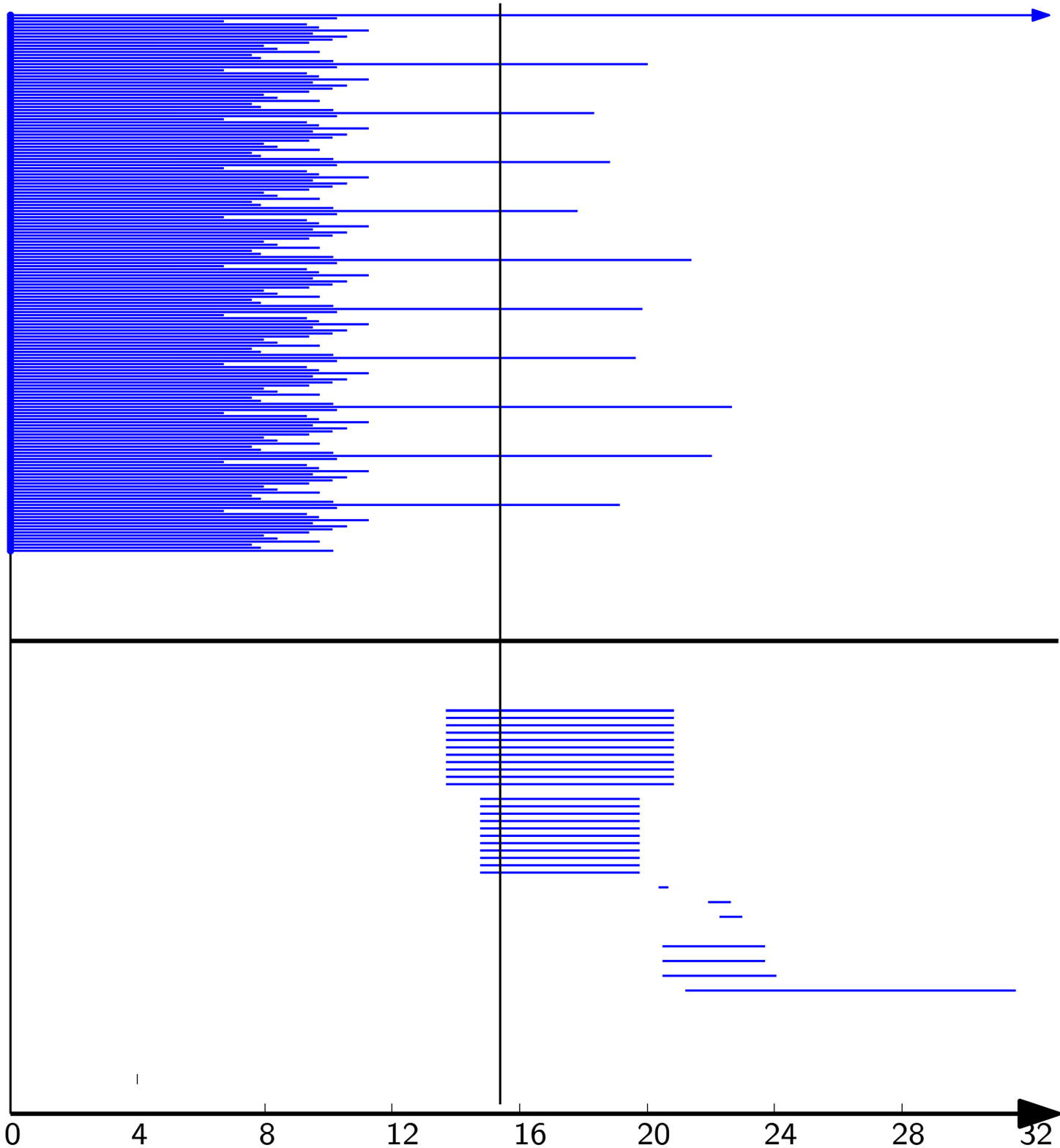
Example: persistence of Čech complexes



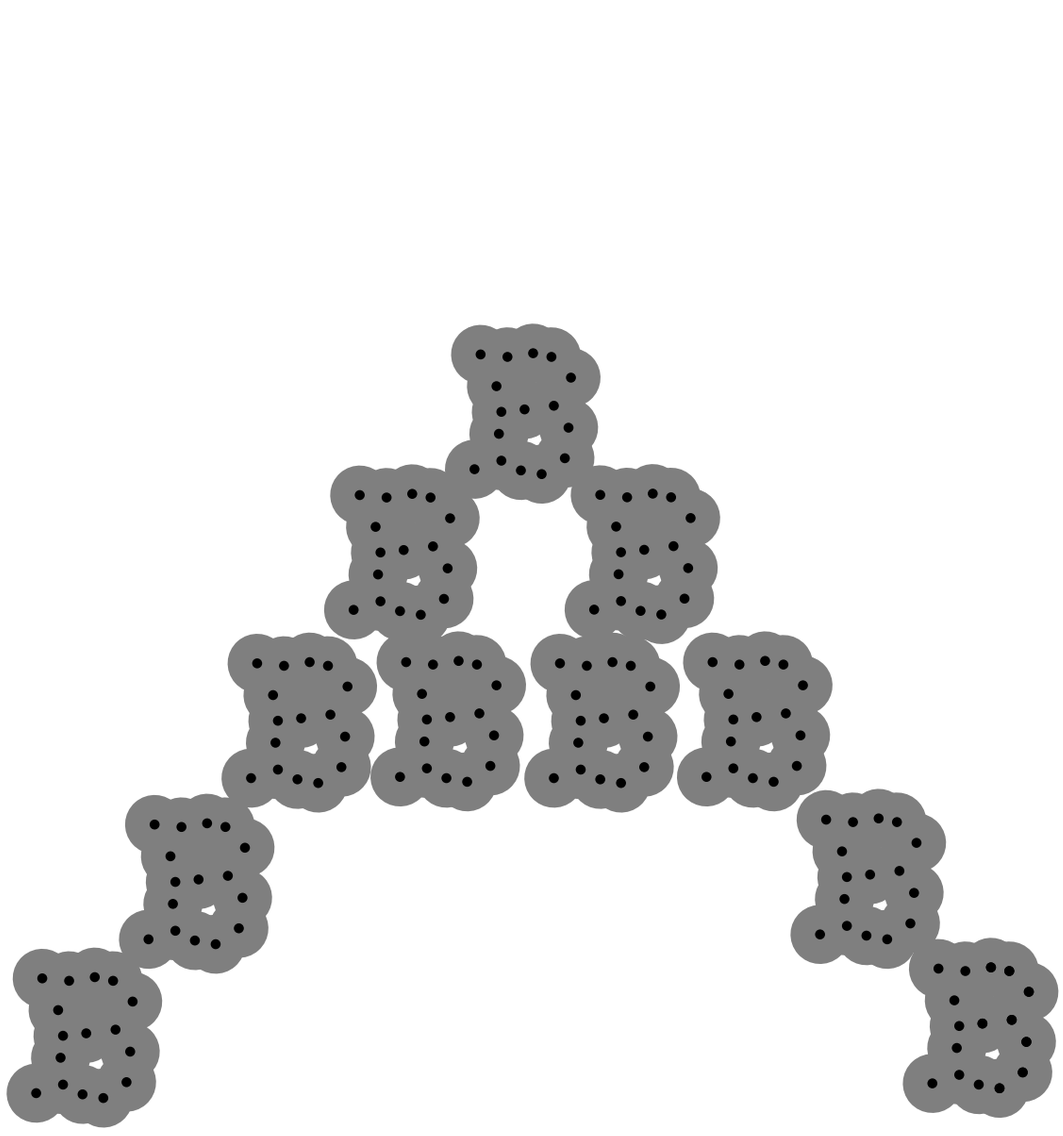


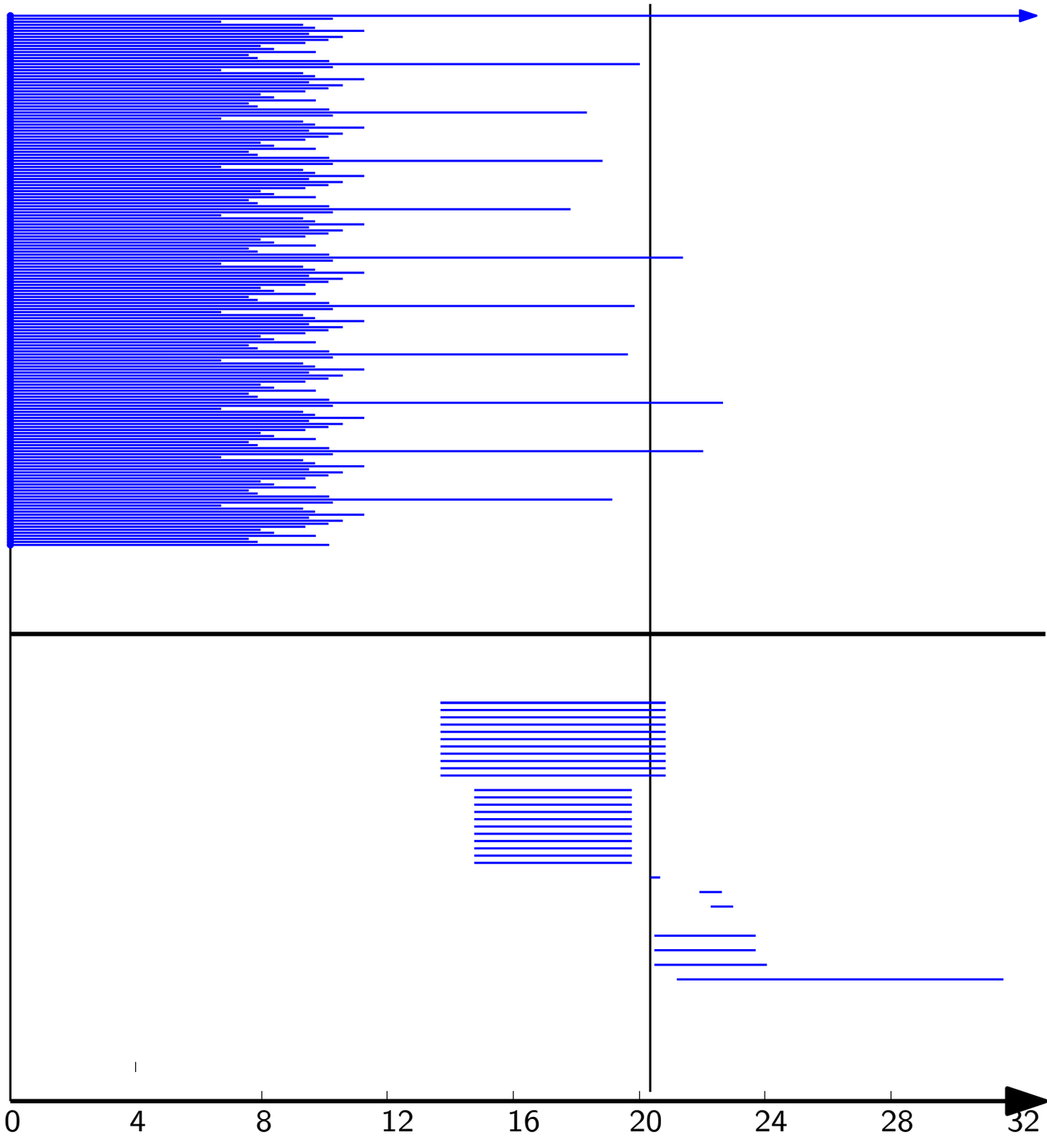
Example: persistence of Čech complexes



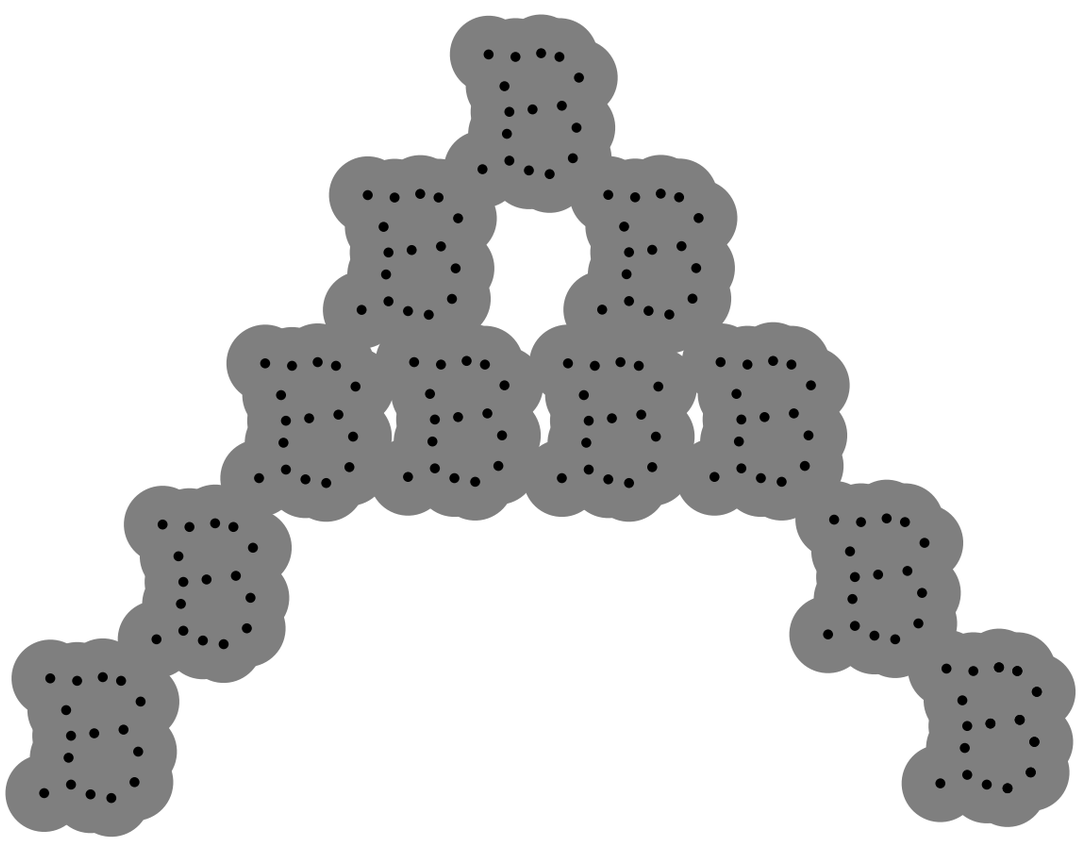


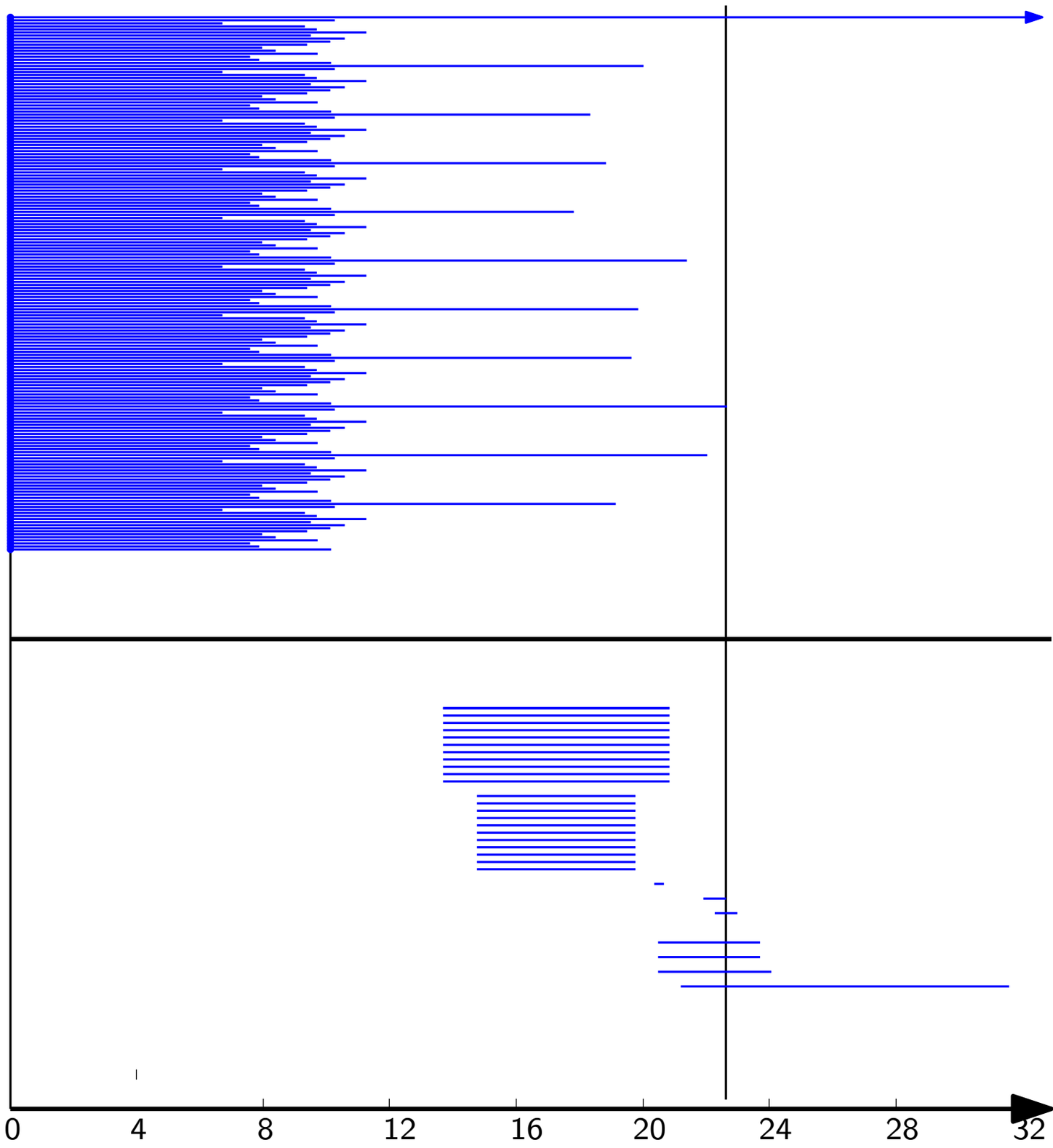
Example: persistence of Čech complexes



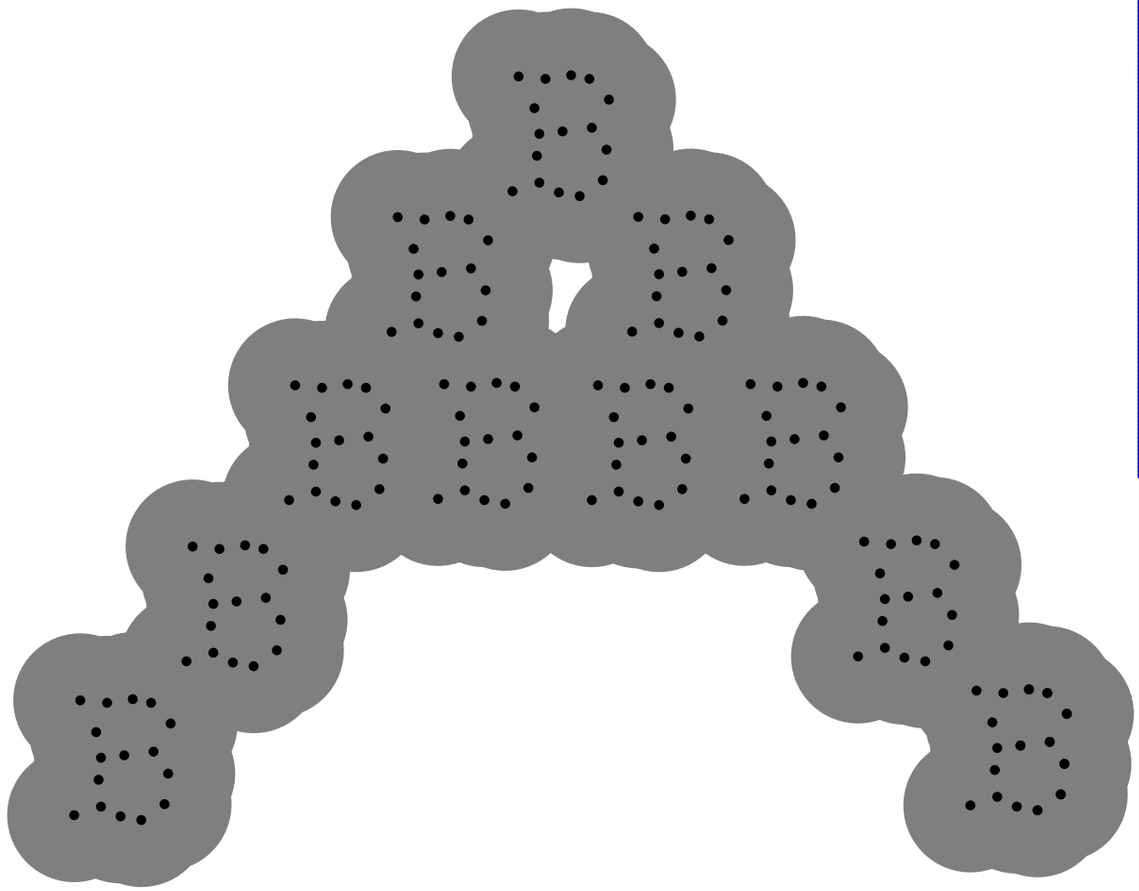


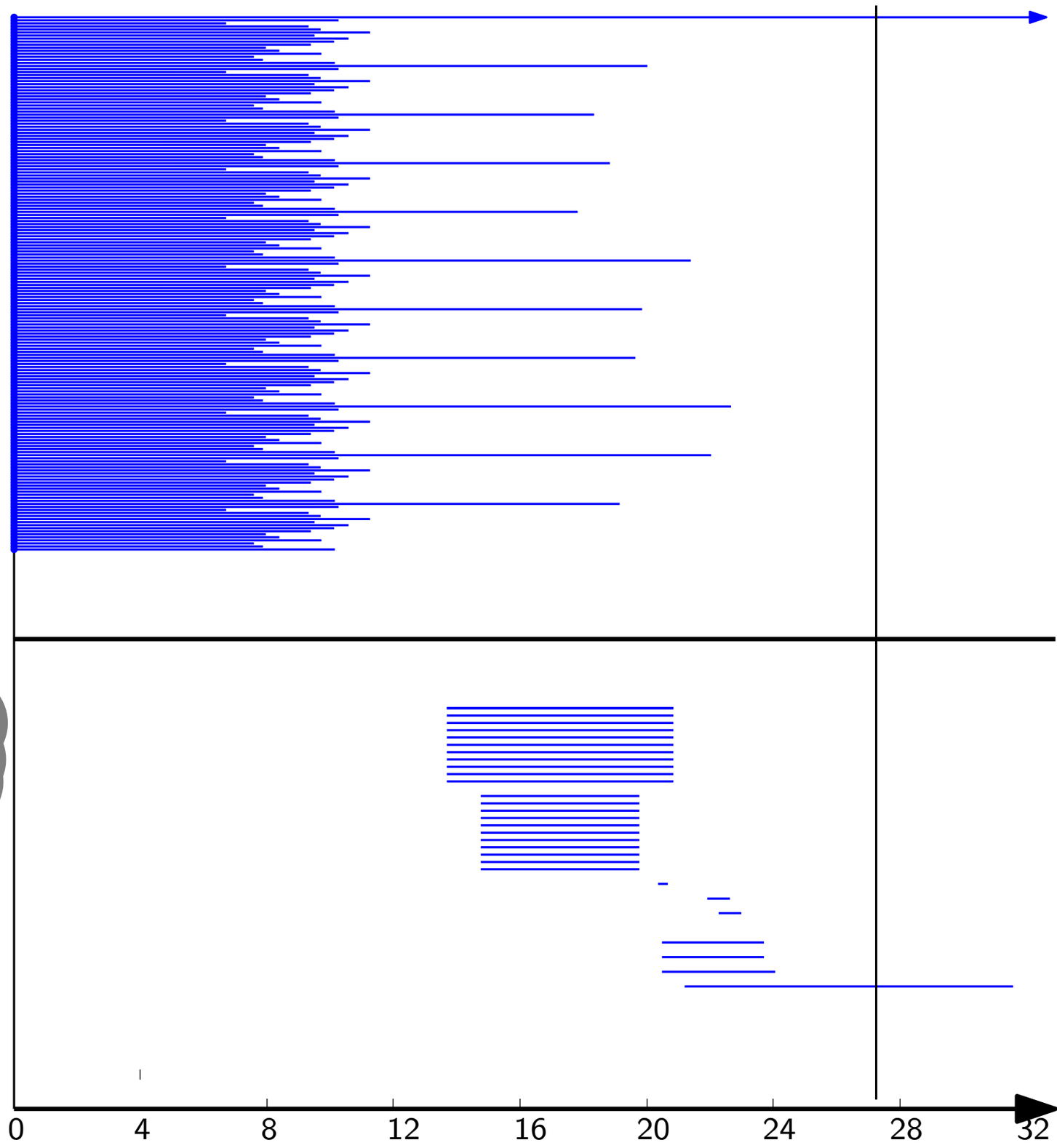
Example: persistence of Čech complexes



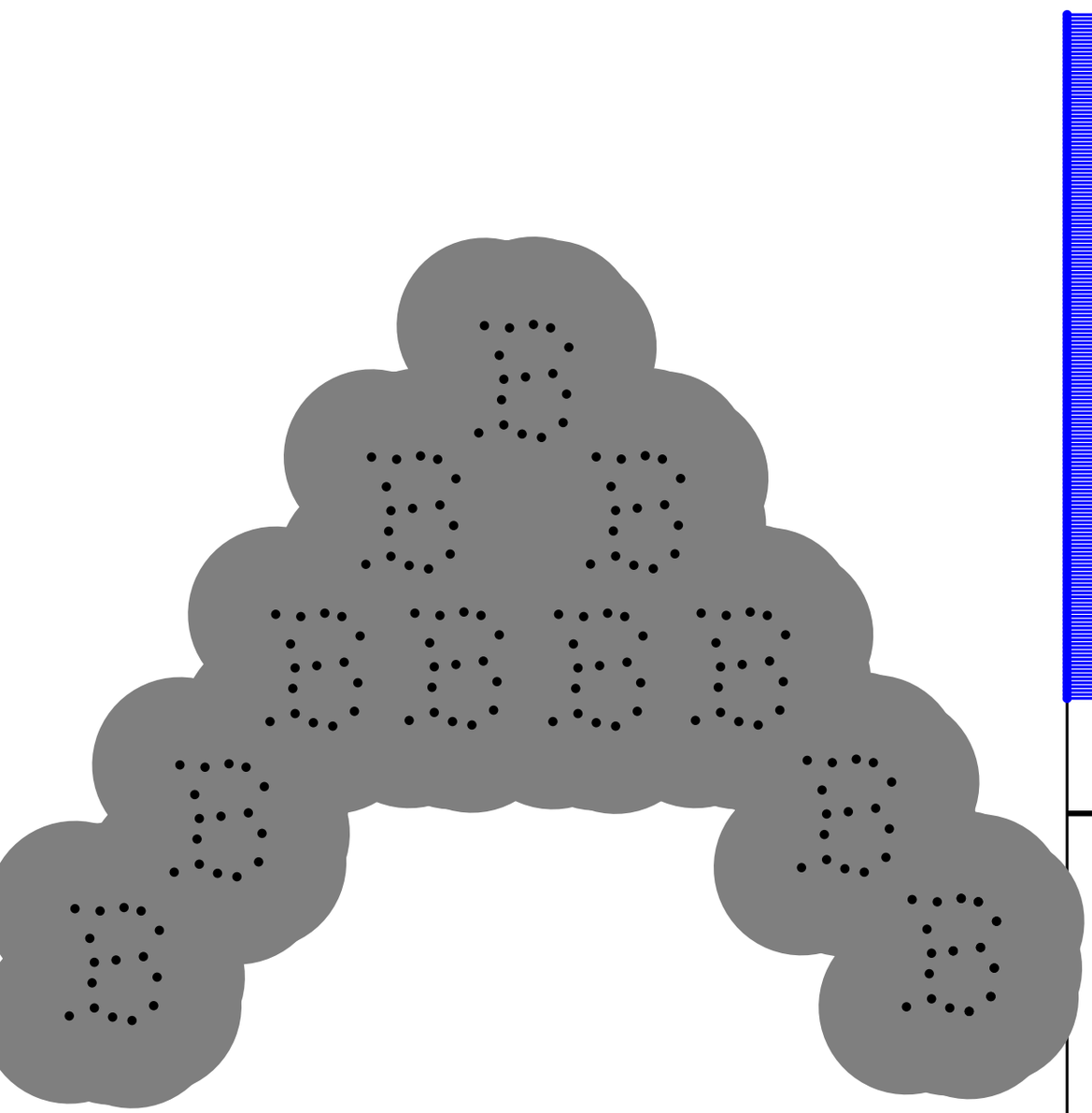


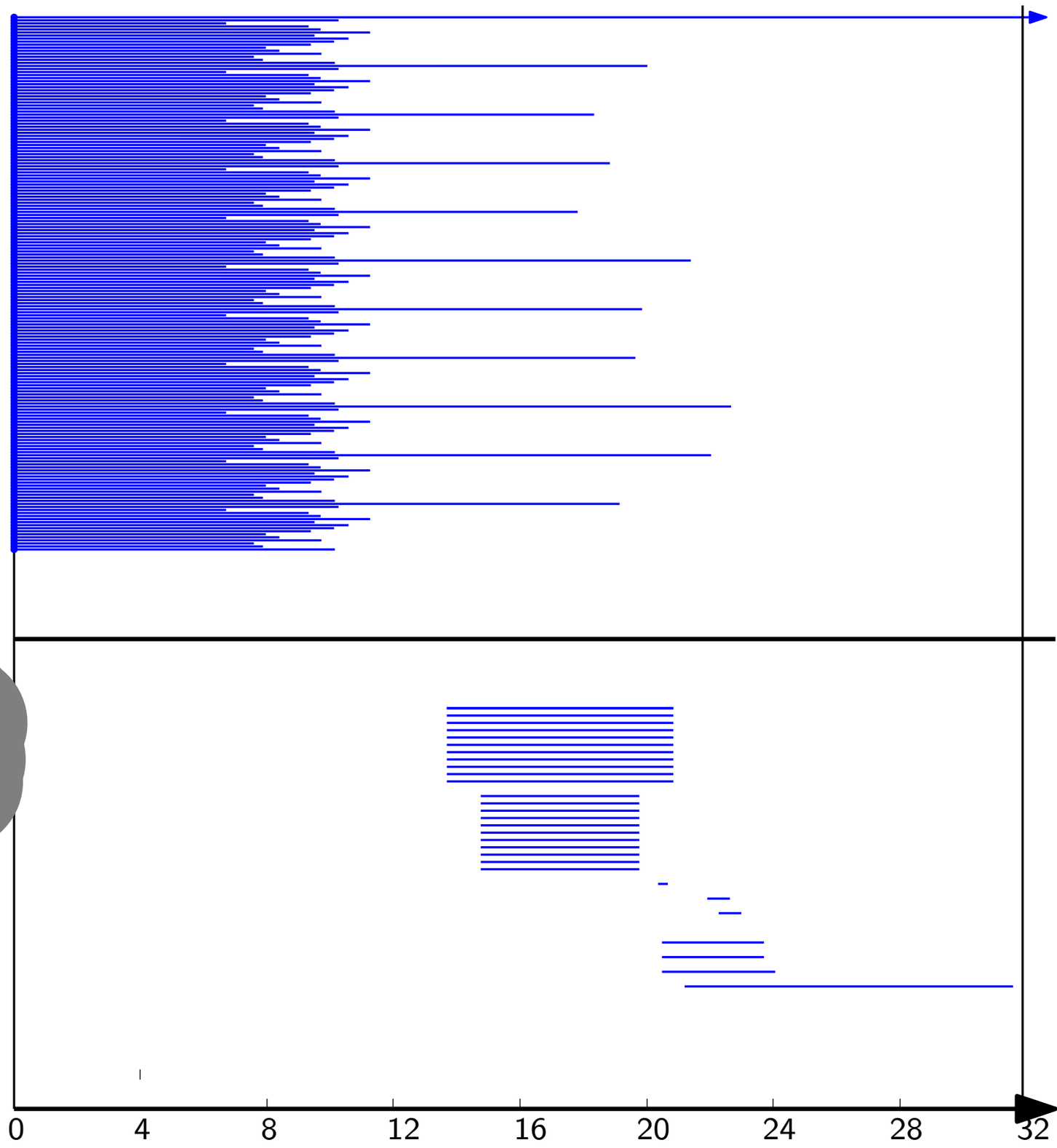
Example: persistence of Čech complexes



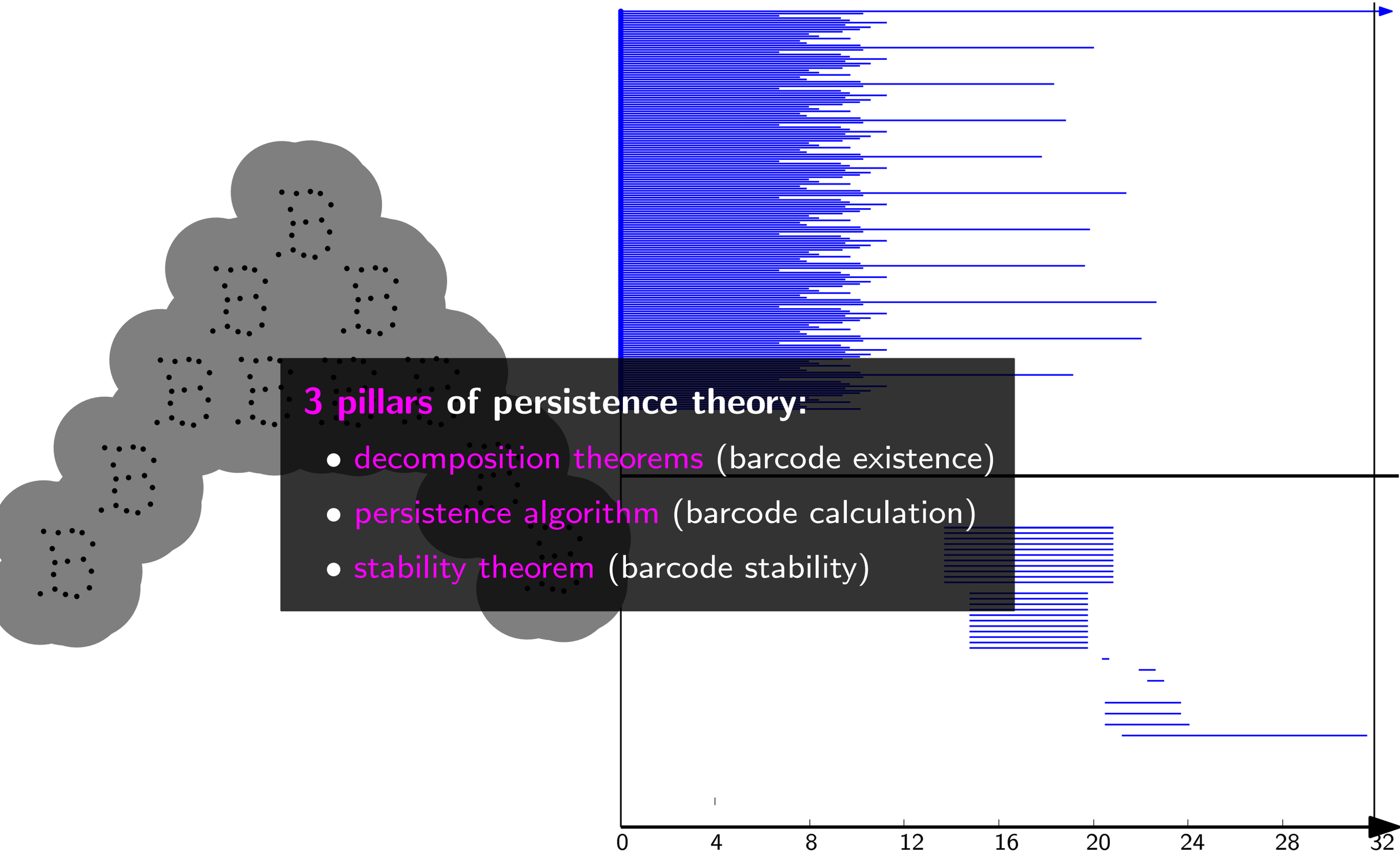


Example: persistence of Čech complexes





Example: persistence of Čech complexes

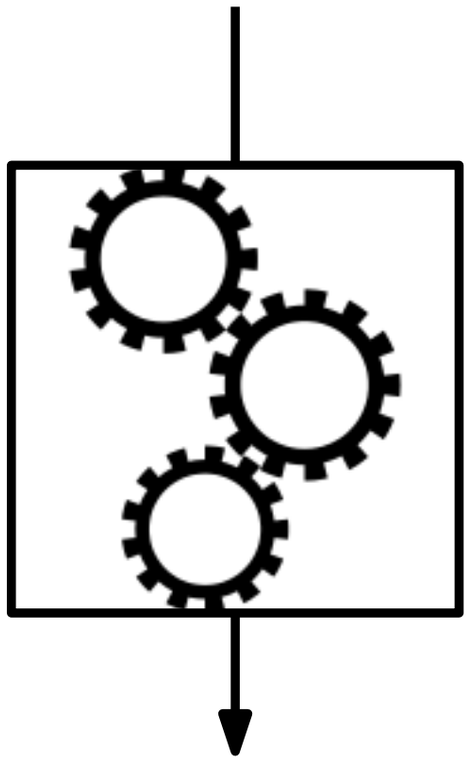


3 pillars of persistence theory:

- decomposition theorems (barcode existence)
- persistence algorithm (barcode calculation)
- stability theorem (barcode stability)

Mathematical foundations

Filtration: $F_1 \subseteq F_2 \subseteq F_3 \subseteq F_4 \subseteq F_5 \cdots$



(homology functor)

topological level

algebraic level

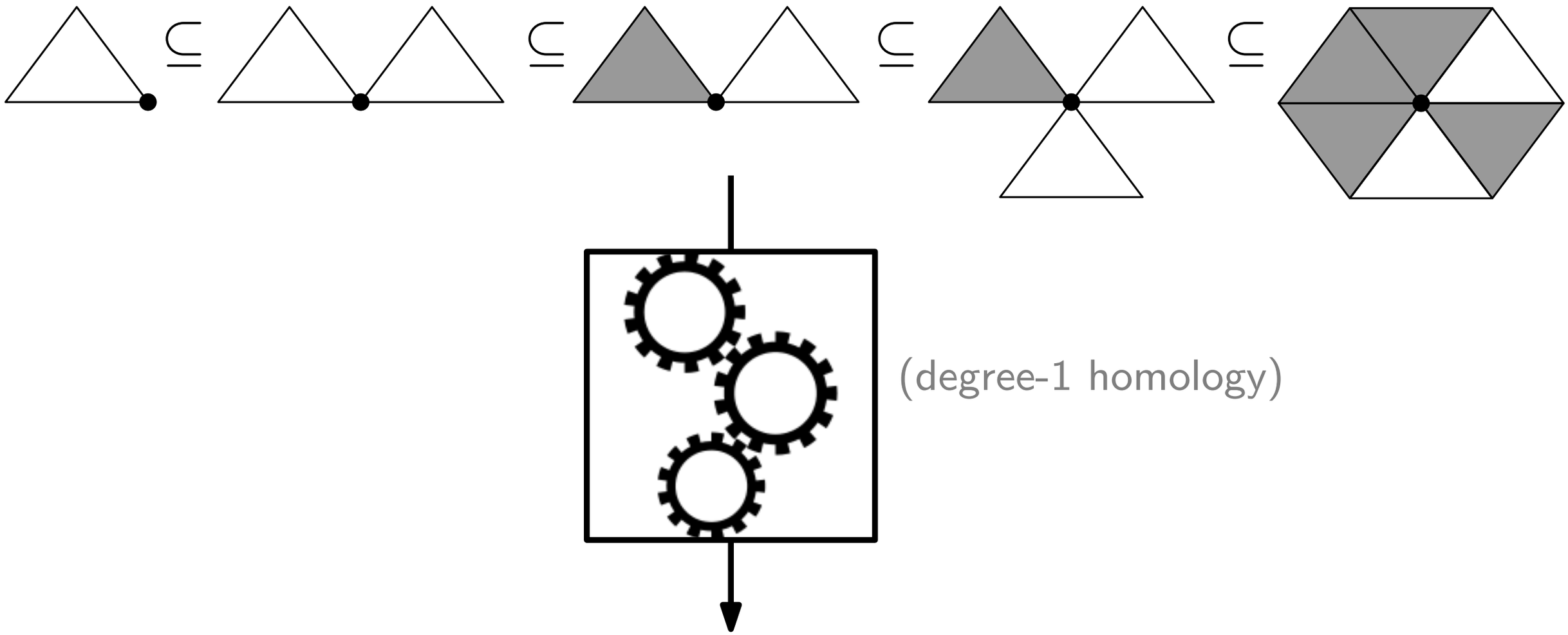
$$H_*(F_1) \to H_*(F_2) \to H_*(F_3) \to H_*(F_4) \to H_*(F_5) \to \cdots$$

Def: A *persistence module* is a sequence of vector spaces connected with linear maps:

$$H_*(F_1) \to H_*(F_2) \to H_*(F_3) \to H_*(F_4) \to \cdots$$

Mathematical foundations

Example:

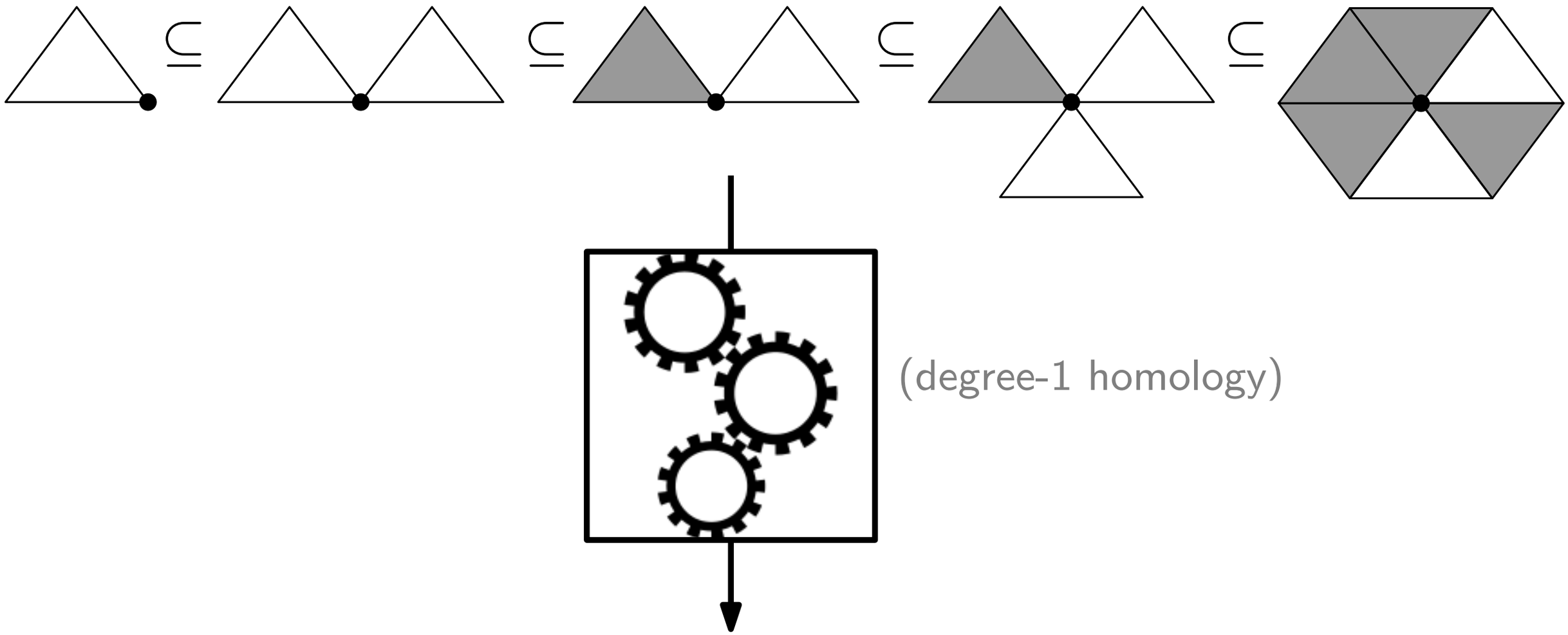


(degree-1 homology)

$$\boldsymbol{k} \xrightarrow{\binom{1}{0}} \boldsymbol{k}^2 \xrightarrow{(0\ 1)} \boldsymbol{k}$$

Mathematical foundations

Example:

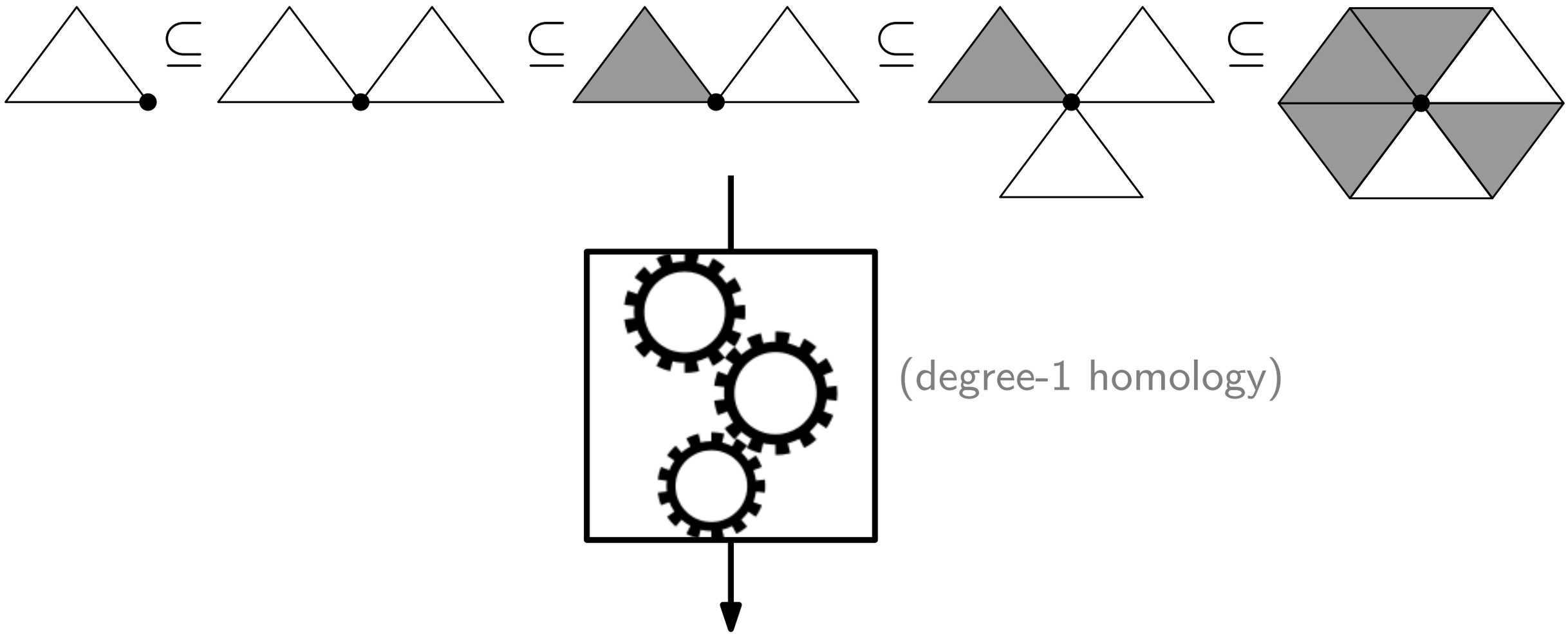


(degree-1 homology)

$$\boldsymbol{k} \xrightarrow{\binom{1}{0}} \boldsymbol{k}^2 \xrightarrow{(0\ 1)} \boldsymbol{k} \qquad \xrightarrow{\binom{0}{1}} \boldsymbol{k}^2$$

Mathematical foundations

Example:



(degree-1 homology)

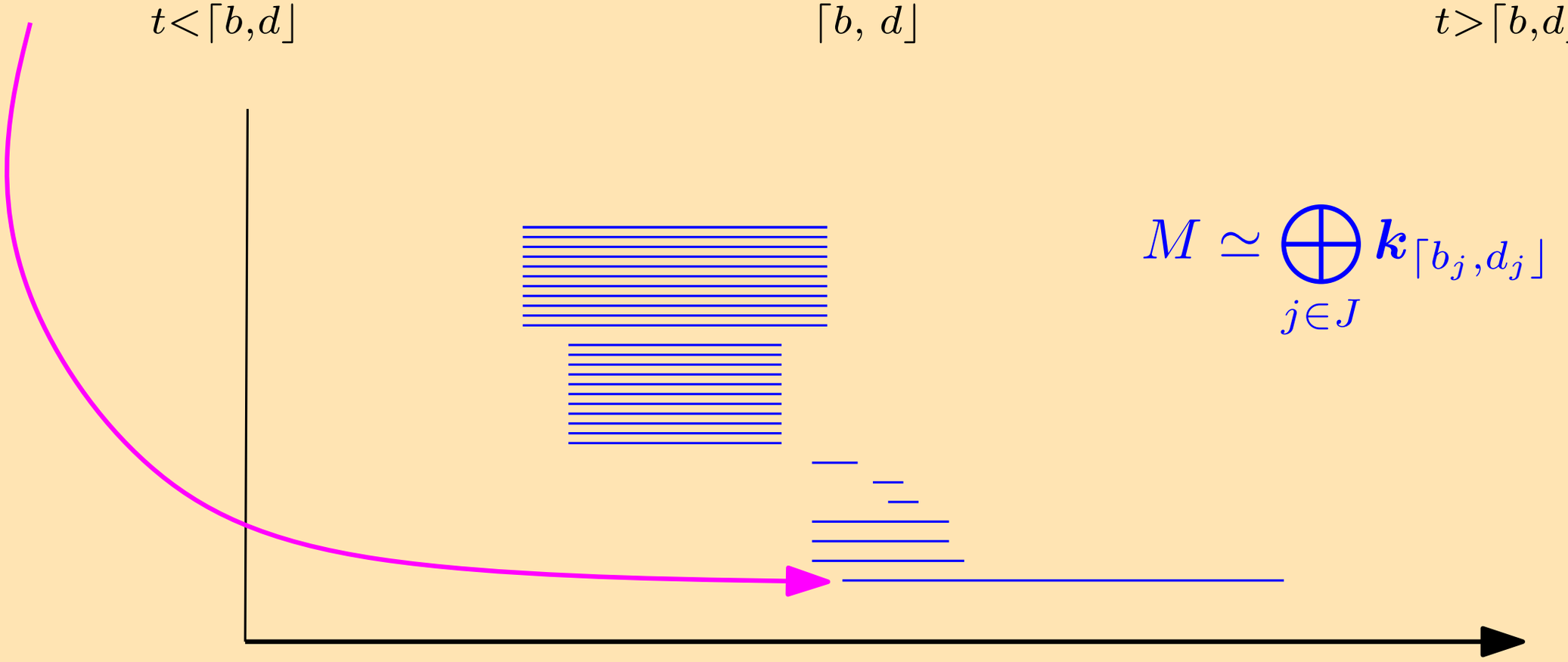
$$\boldsymbol{k} \xrightarrow{\binom{1}{0}} \boldsymbol{k}^2 \xrightarrow{(0\ 1)} \boldsymbol{k} \qquad \xrightarrow{\binom{0}{1}} \boldsymbol{k}^2 \xrightarrow{\binom{1\ 0}{0\ 1}} \boldsymbol{k}^2 \ \cdots$$

Mathematical foundations

[*The structure and stability of persistence modules*, Chazal, de Silva, Glisse, Oudot, Springer, 2016].

Thm: Let M be a persistence module over an index set $T \subseteq \mathbb{R}$. Then, M decomposes as a direct sum of *interval modules* $\boldsymbol{k}_{\lceil b,d \rfloor}$:

$$\underbrace{0 \xrightarrow{0} \cdots \xrightarrow{0} 0}_{t<\lceil b,d \rfloor} \xrightarrow{0} \underbrace{\boldsymbol{k} \xrightarrow{\mathbb{1}} \cdots \xrightarrow{\mathbb{1}} \boldsymbol{k}}_{\lceil b,\, d \rfloor} \xrightarrow{0} \underbrace{0 \xrightarrow{0} \cdots \xrightarrow{0} 0}_{t>\lceil b,d \rfloor}$$

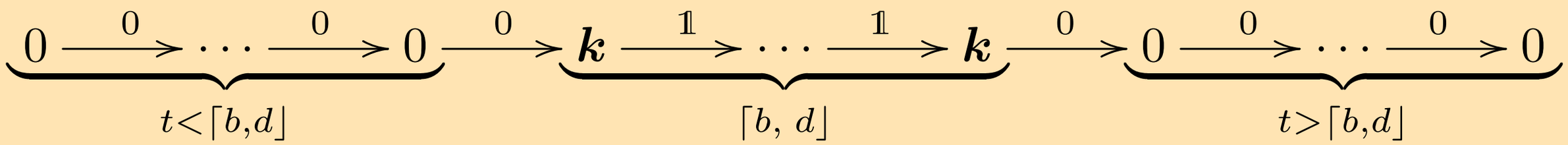


(the barcode is a complete descriptor of the algebraic structure of M)

Mathematical foundations

[*The structure and stability of persistence modules*, Chazal, de Silva, Glisse, Oudot, Springer, 2016].

Thm: Let M be a persistence module over an index set $T \subseteq \mathbb{R}$. Then, M decomposes as a direct sum of *interval modules* $\boldsymbol{k}_{\lceil b,d \rfloor}$:



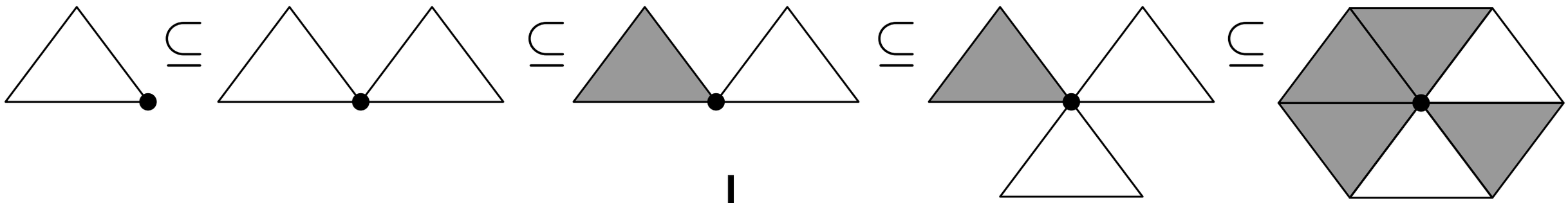
in the following cases:

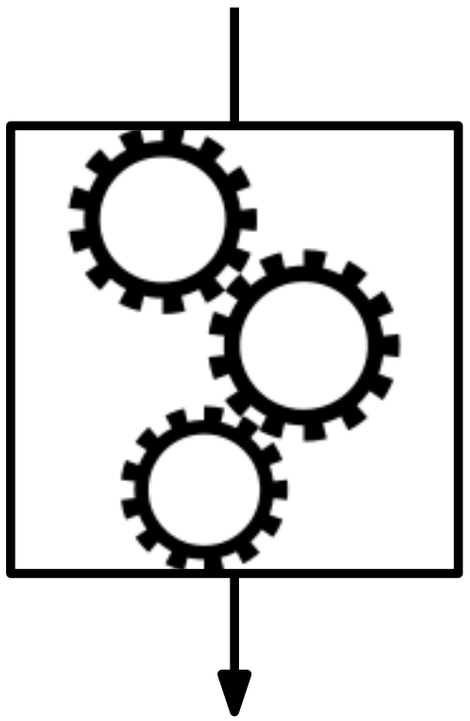
- T is finite,
- M is *pointwise finite-dimensional* (pfd), i.e., every space M_t has finite dimension.

Moreover, when it exists, the decomposition is **unique** up to isomorphism and permutation of the terms [Azumaya 1950].

Mathematical foundations

Example:

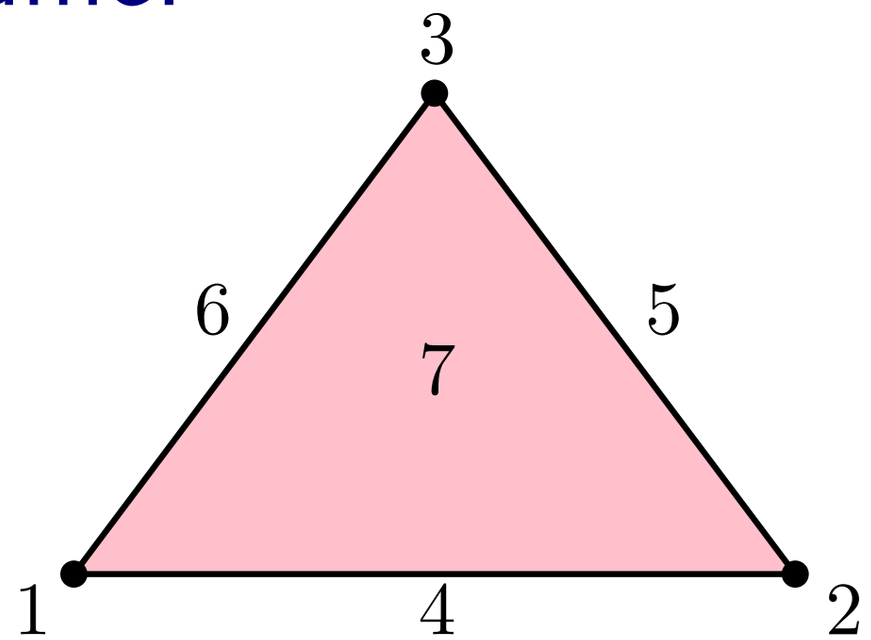




(degree-1 homology)

$$\boldsymbol{k} \xrightarrow{\binom{1}{0}} \boldsymbol{k}^2 \xrightarrow{(0\ 1)} \boldsymbol{k} \qquad \xrightarrow{\binom{0}{1}} \boldsymbol{k}^2 \xrightarrow{\binom{1\ 0}{0\ 1}} \boldsymbol{k}^2 \ \cdots$$

Good news: the algorithm is the same!



Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

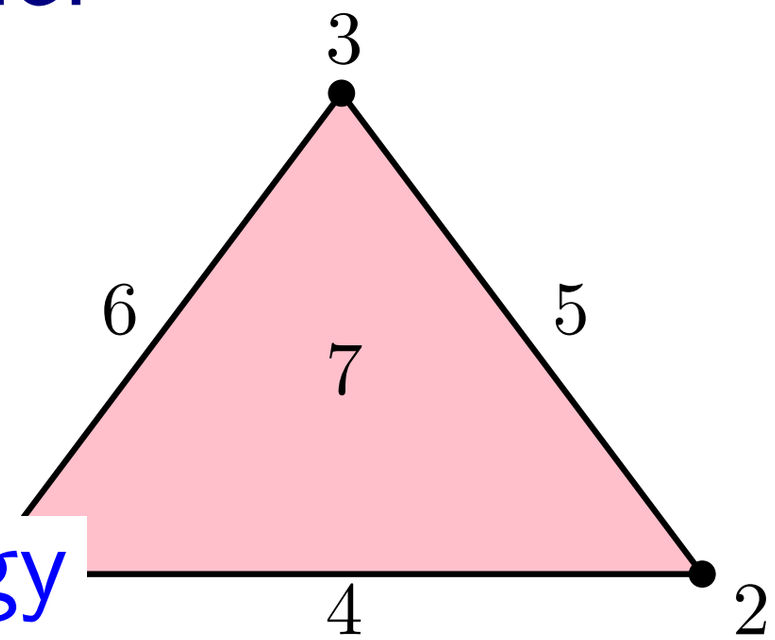
◯ simplex pairs give finite intervals:
$[2,4)$, $[3,5)$, $[6,7)$

□ 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							

Good news: the algorithm is the same!



Input: simplicial filtration

Output: boundary matrix
reduced to column-echelon form

◯ simplex pairs give Persistent homology
$[2,4)$, $[3,5)$, $[6,7)$

□ unpaired simplices give Regular homology

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

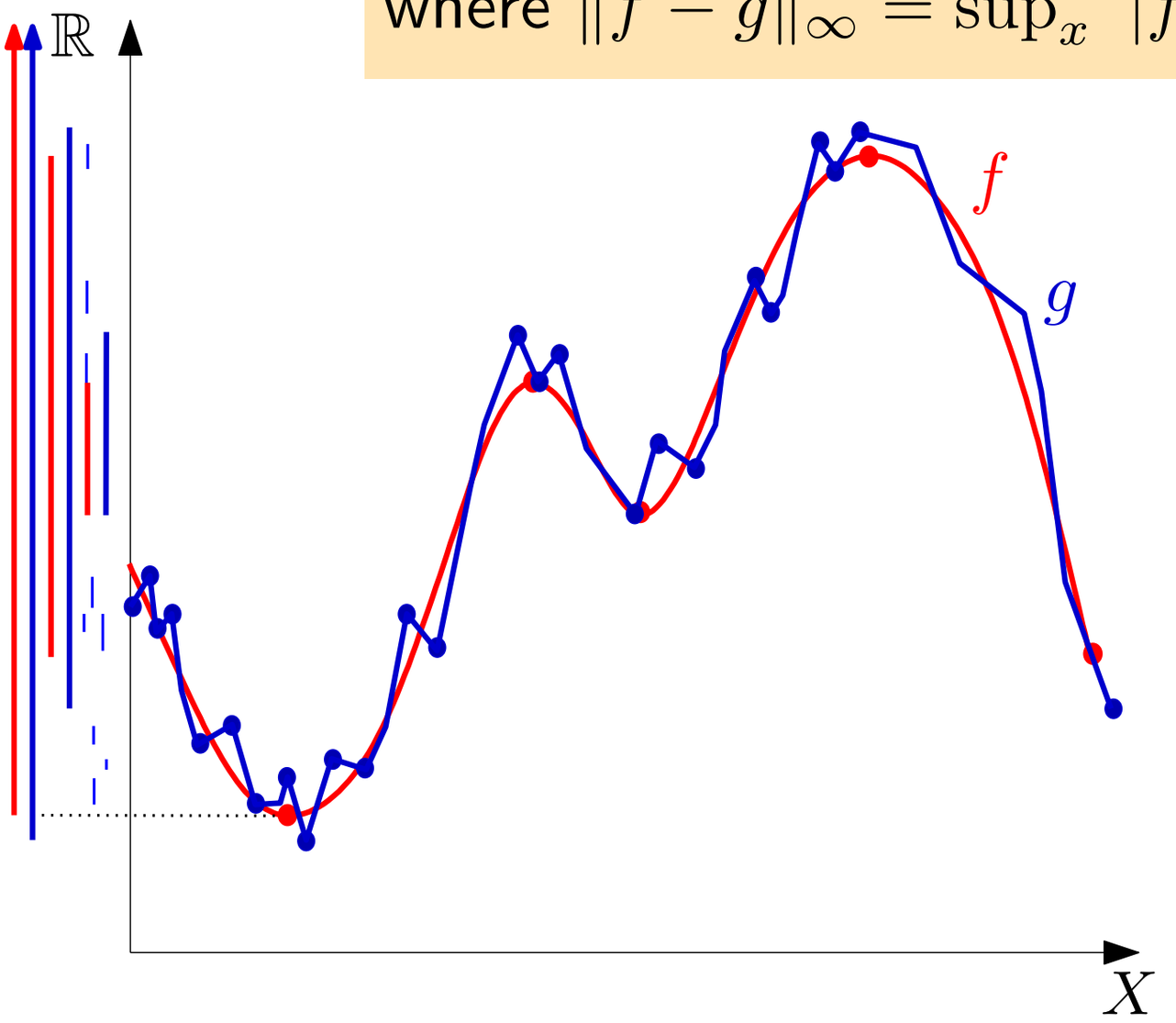
	1	2	3	4	5	6	7
1				$*$			
2				(1)	$*$		
3					(1)		
4							$*$
5							$*$
6							(1)
7							

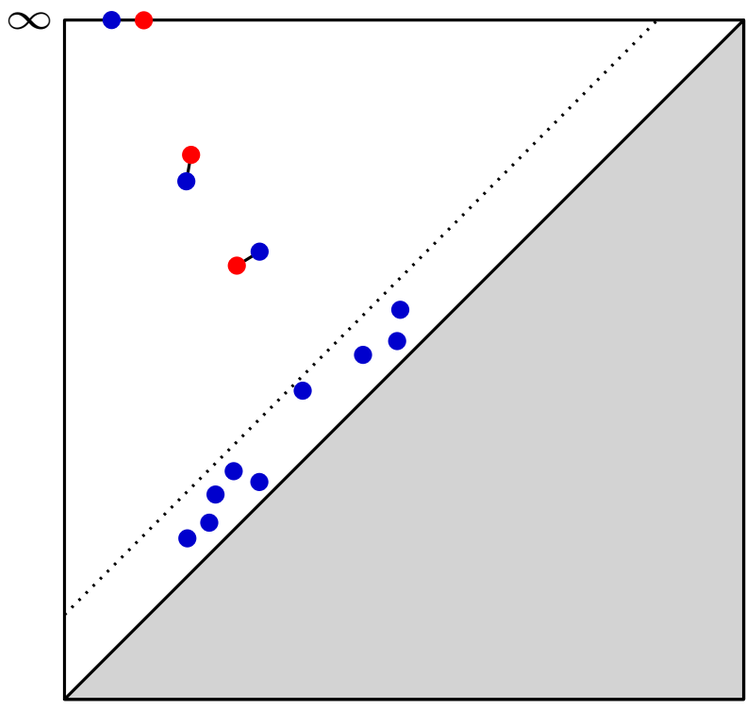
Stability properties

Thm: For any pfd functions $f, g : X \to \mathbb{R}$ and homological dimension k,

$$d_b(D_f, D_g) \leq \|f - g\|_\infty,$$

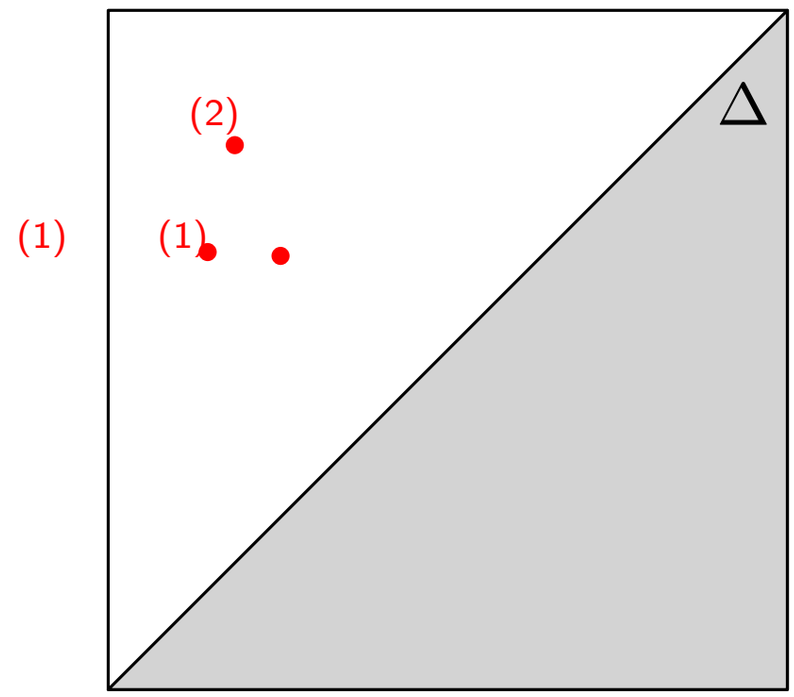
where $\|f - g\|_\infty = \sup_x \ |f(x) - g(x)|$.





Stability properties

Persistence diagram $\equiv$ **finite** multiset in the open half-plane $\Delta \times \mathbb{R}_{>0}$.



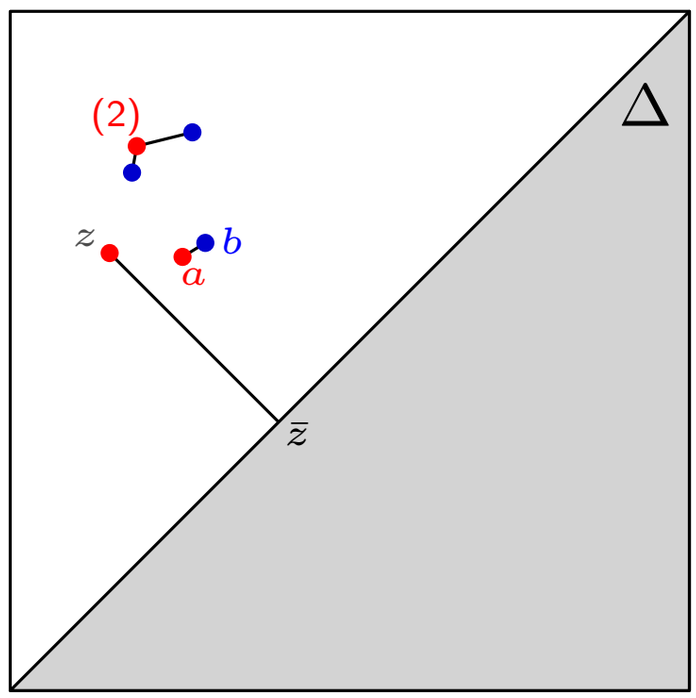
Stability properties

Persistence diagram $\equiv$ **finite** multiset in the open half-plane $\Delta \times \mathbb{R}_{>0}$.

Given a **partial matching** $M : D \leftrightarrow D'$:

- cost of a matched pair $(a, b) \in M$: $c_p(a, b) := \|a - b\|_\infty^p$,
- cost of an unmatched point $c \in A \sqcup B$: $c_p(c) := \|c - \bar{c}\|_\infty^p$,
- **cost of** M:

$$c_p(M) := \left(\sum_{(a,\, b) \text{ matched}} c_p(a, b) \; + \sum_{c \text{ unmatched}} c_p(c) \right)^{1/p}$$



Stability properties

Persistence diagram $\equiv$ **finite** multiset in the open half-plane $\Delta \times \mathbb{R}_{>0}$.

Given a **partial matching** $M : D \leftrightarrow D'$:

- cost of a matched pair $(a, b) \in M$: $c_p(a, b) := \|a - b\|_\infty^p$,
- cost of an unmatched point $c \in A \sqcup B$: $c_p(c) := \|c - \bar{c}\|_\infty^p$,
- **cost of** M:

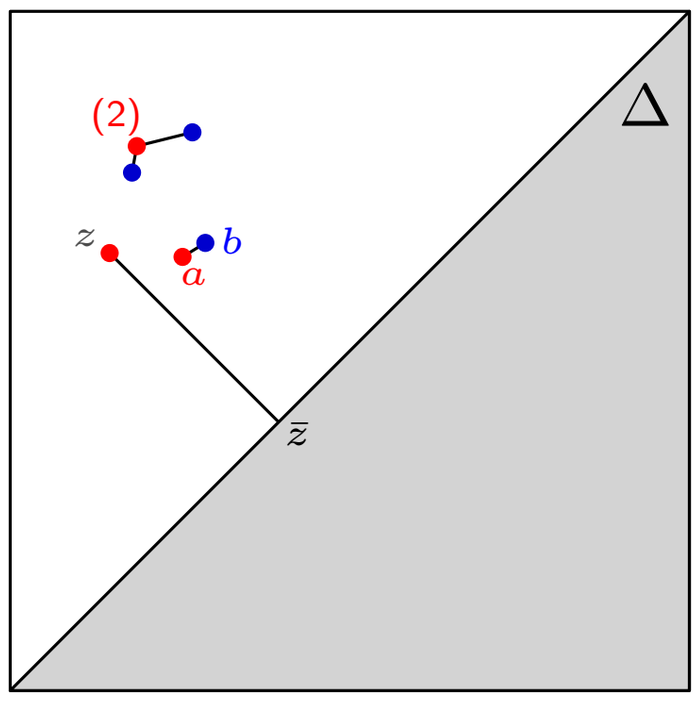
$$c_p(M) := \left(\sum_{(a,\, b) \text{ matched}} c_p(a, b) \; + \sum_{c \text{ unmatched}} c_p(c) \right)^{1/p}$$

Def: p-th diagram distance (extended metric):

$$d_p(D, D') := \inf_{M : D \leftrightarrow D'} c_p(M)$$

Def: bottleneck distance:

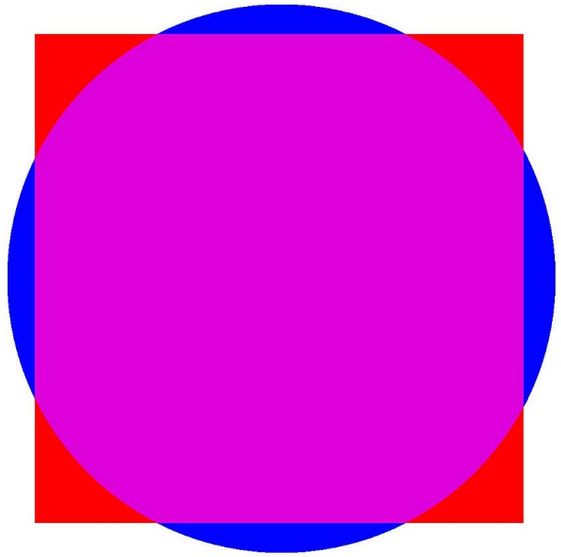
$$d_b(D, D') = d_\infty(D, D') := \lim_{p \to \infty} d_p(D, D')$$

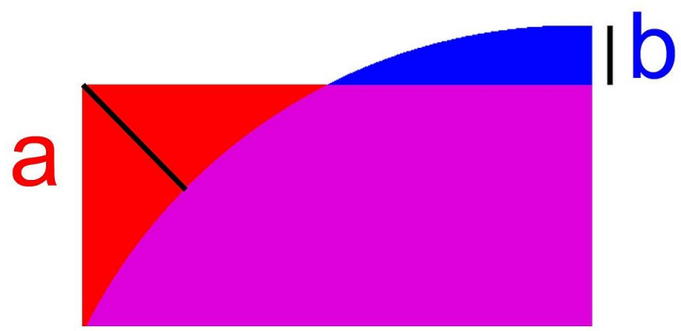


Stability properties for point clouds

Def: The Hausdorff distance between two subspaces X, Y of a common metric space (Z, d) is:

$$d_H(X,Y) = \max\{\sup_{y\in Y} d(y,X), \sup_{x\in X} d(x,Y)\}$$
$$= \max\{\sup_{y\in Y}\inf_{x\in X} d(y,x), \sup_{x\in X}\inf_{y\in Y} d(x,y)\}$$





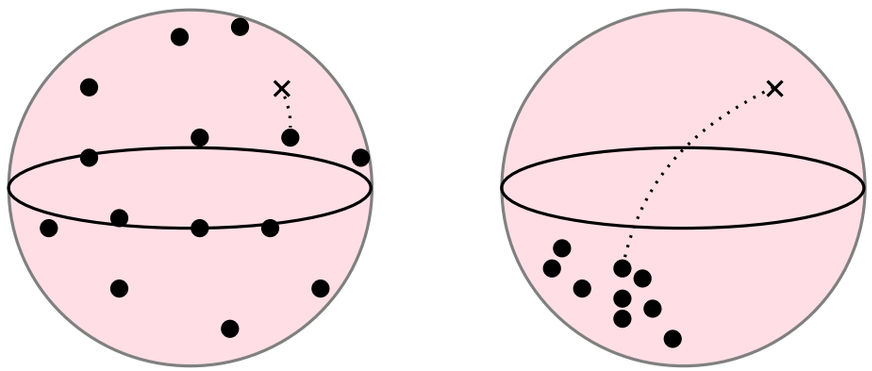
$$d_H(X,Y) = \max\{a,b\}$$

Stability properties for point clouds

Def: The Hausdorff distance between two subspaces X, Y of a common metric space (Z, d) is:

$$d_H(X,Y) = \max\{\sup_{y\in Y} d(y,X), \sup_{x\in X} d(x,Y)\}$$
$$= \max\{\sup_{y\in Y}\inf_{x\in X} d(y,x), \sup_{x\in X}\inf_{y\in Y} d(x,y)\}$$

Ex: Given a sampling $\hat{X}_n \subseteq X$, $d_H(\hat{X}_n, X)$ is a measure of sampling quality (cf class 3 for Mapper parameters).



Q: Show that $d_H(X,Y) = \inf\{\epsilon > 0 : X^\epsilon \subseteq Y \text{ and } Y^\epsilon \subseteq X\}$, where $X^\epsilon = \{z : \exists x \in X \text{ s.t. } d(x,z) \leq \epsilon\}$.

Stability properties for point clouds

Def: The Hausdorff distance between two subspaces X, Y of a common metric space (Z, d) is:

$$d_H(X,Y) = \max\{\sup_{y\in Y} d(y,X), \sup_{x\in X} d(x,Y)\}$$
$$= \max\{\sup_{y\in Y}\inf_{x\in X} d(y,x), \sup_{x\in X}\inf_{y\in Y} d(x,y)\}$$

Def: The Gromov-Hausdorff distance between metric spaces $(X, d_X), (Y, d_Y)$ is the Hausdorff distance of the best common isometric embedding:

$$d_{GH}((X,d_X),(Y,d_Y)) = \inf_\gamma d_H(\gamma(X),\gamma(Y)),$$

where $d(\gamma(x),\gamma(x')) = d_X(x,x')$ and $d(\gamma(y),\gamma(y')) = d_X(y,y')$.

Stability properties for point clouds

Def: The Hausdorff distance between two subspaces X, Y of a common metric space (Z, d) is:

$$d_H(X,Y) = \max\{\sup_{y\in Y} d(y,X), \sup_{x\in X} d(x,Y)\}$$
$$= \max\{\sup_{y\in Y}\inf_{x\in X} d(y,x), \sup_{x\in X}\inf_{y\in Y} d(x,y)\}$$

Def: The Gromov-Hausdorff distance between metric spaces $(X, d_X), (Y, d_Y)$ is metric distortion of the best correspondence:

$$d_{GH}((X,d_X),(Y,d_Y)) = \inf_{\mathcal{C}} \sup_{(x,y),(x',y')\in\mathcal{C}} |d_X(x,x') - d_Y(y,y')|,$$

where $\mathcal{C} \subseteq X \times Y$ s.t. $\forall x$, $\exists y_x \in Y$ s.t. $(x, y_x) \in \mathcal{C}$ (and vice-versa).

Stability properties for point clouds

Def: The Hausdorff distance between two subspaces X, Y of a common metric space (Z, d) is:

$$d_H(X,Y) = \max\{\sup_{y\in Y} d(y,X), \sup_{x\in X} d(x,Y)\}$$
$$= \max\{\sup_{y\in Y}\inf_{x\in X} d(y,x), \sup_{x\in X}\inf_{y\in Y} d(x,y)\}$$

Def: The Gromov-Hausdorff distance between metric spaces $(X, d_X), (Y, d_Y)$ is metric distortion of the best correspondence:

$$d_{GH}((X,d_X),(Y,d_Y)) = \inf_{\mathcal{C}} \sup_{(x,y),(x',y')\in\mathcal{C}} |d_X(x,x') - d_Y(y,y')|,$$

where $\mathcal{C} \subseteq X \times Y$ s.t. $\forall x$, $\exists y_x \in Y$ s.t. $(x, y_x) \in \mathcal{C}$ (and vice-versa).

Thm: If X and Y are common subspaces of a common metric space (Z, d), then

$$d_b(D_{\text{Cech}}(X), D_{\text{Cech}}(Y)) \leq d_H(X,Y).$$

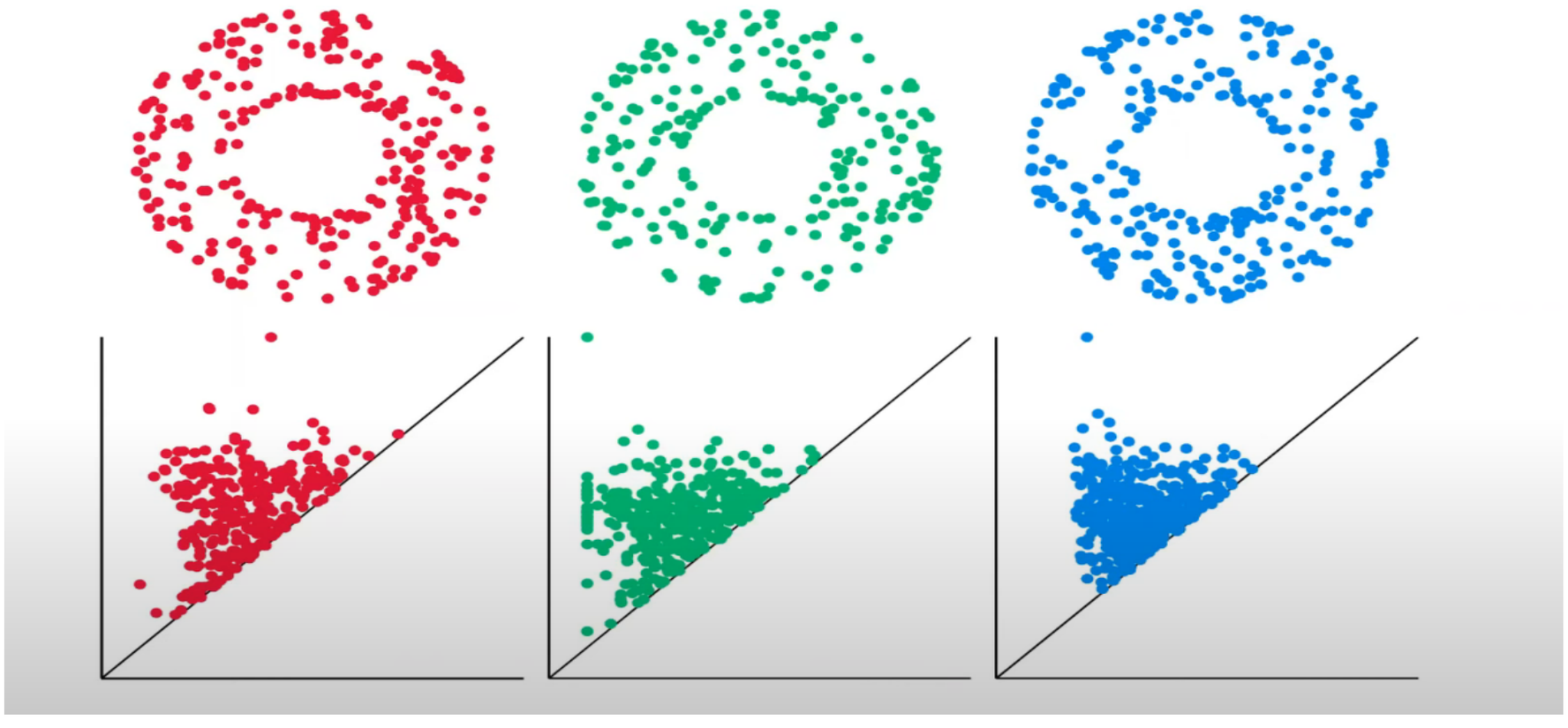
Q: Prove it.

Stability properties for point clouds

[*Persistence stability for geometric complexes*, Chazal, de Silva, Oudot, Geom. Dedicata, 2013].

Thm: If X and Y are pre-compact metric spaces, then

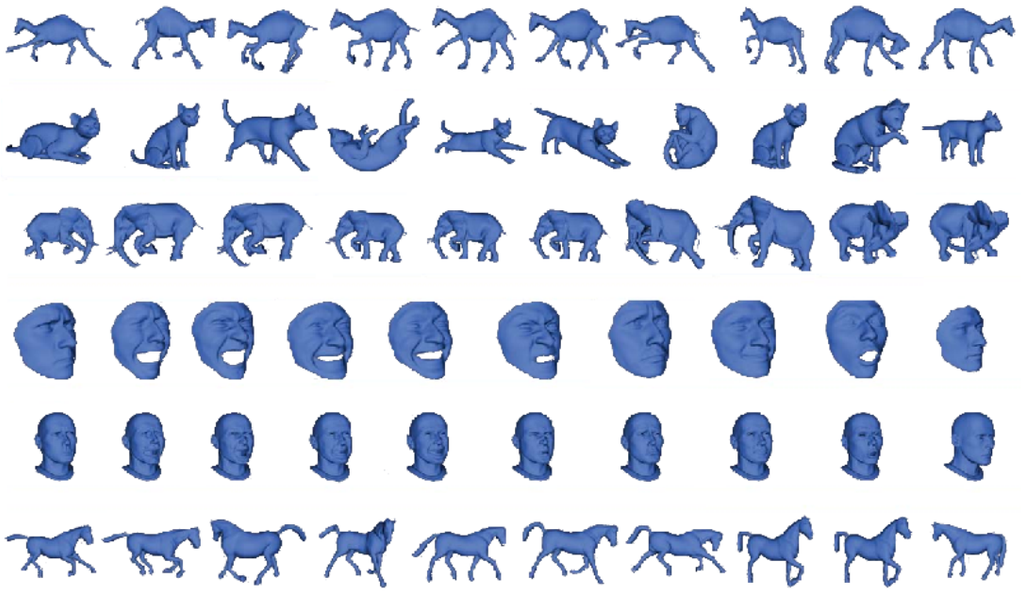
$$d_b(D_{\mathrm{Rips}}(X), D_{\mathrm{Rips}}(Y)) \leq d_{GH}(X, Y).$$

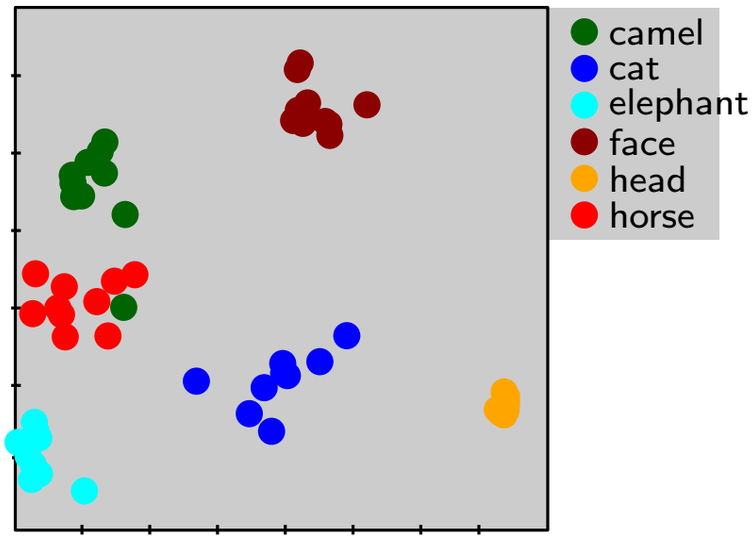


Rem: This result also holds for Čech and other families of filtrations (particular case of a more general theorem).

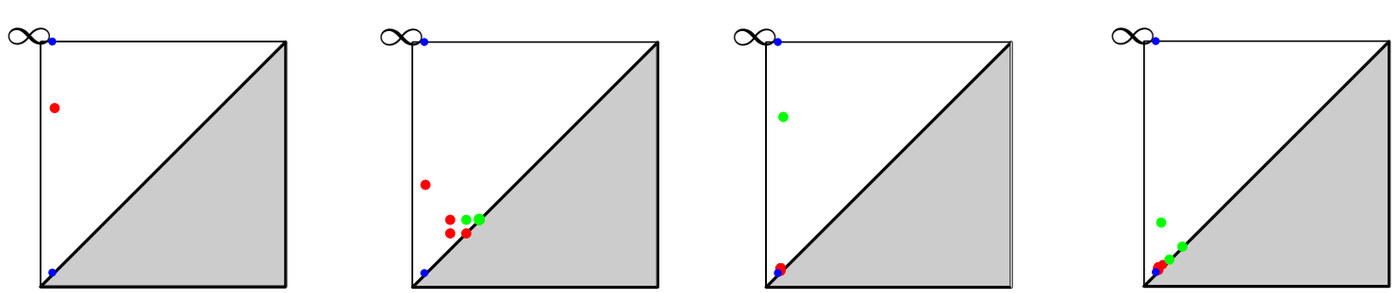
Application: non rigid shape classification

[*Gromov-Hausdorff Stable Signatures for Shapes using Persistence*, Chazal et al., Symp. Geom. Process., 2009]





MDS using bottleneck distance.



- Non rigid shapes in a same class are almost isometric, but computing Gromov-Hausdorff distance between shapes is extremely expensive.
- Compare diagrams of sampled shapes instead of shapes themselves.

Limitations

Thm: If X and Y are pre-compact metric spaces, then

$$d_b(D_{\mathrm{Rips}}(X), D_{\mathrm{Rips}}(Y)) \leq d_{GH}(X, Y).$$

$\rightarrow$ Vietoris-Rips (or Čech, witness) filtrations become quickly prohibitively large as the size of the data increases: $O(|X|^d)$, making the practical computation of persistence almost impossible.

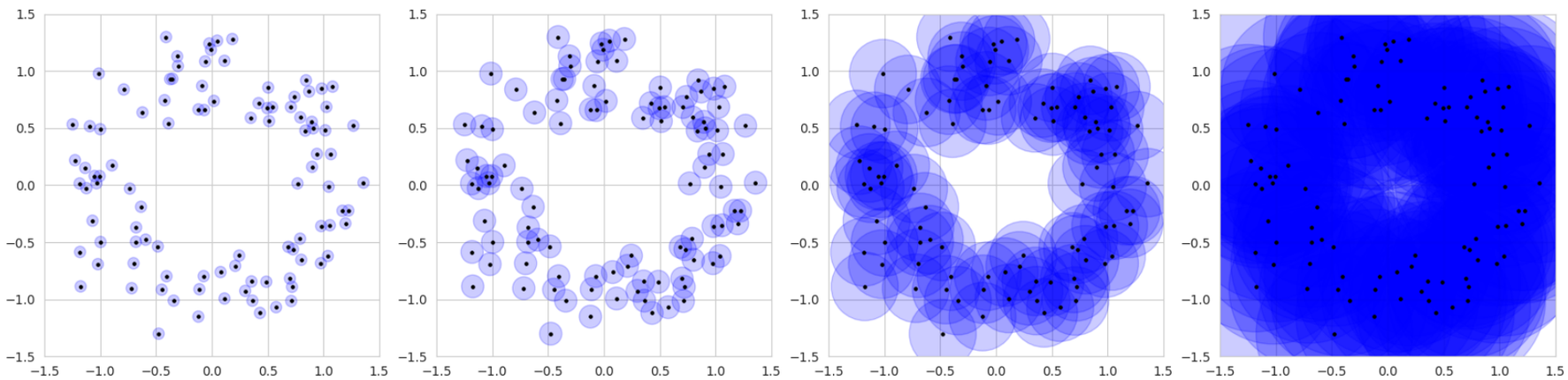
Limitations

Thm: If X and Y are pre-compact metric spaces, then

$$d_b(D_{\mathrm{Rips}}(X), D_{\mathrm{Rips}}(Y)) \leq d_{GH}(X, Y).$$

$\rightarrow$ Vietoris-Rips (or Čech, witness) filtrations become quickly prohibitively large as the size of the data increases: $O(|X|^d)$, making the practical computation of persistence almost impossible.

$\rightarrow$ Persistence diagrams of Vietoris-Rips (as well as Čech, witness,..) filtrations and Gromov-Hausdorff distance are very sensitive to noise and outliers.



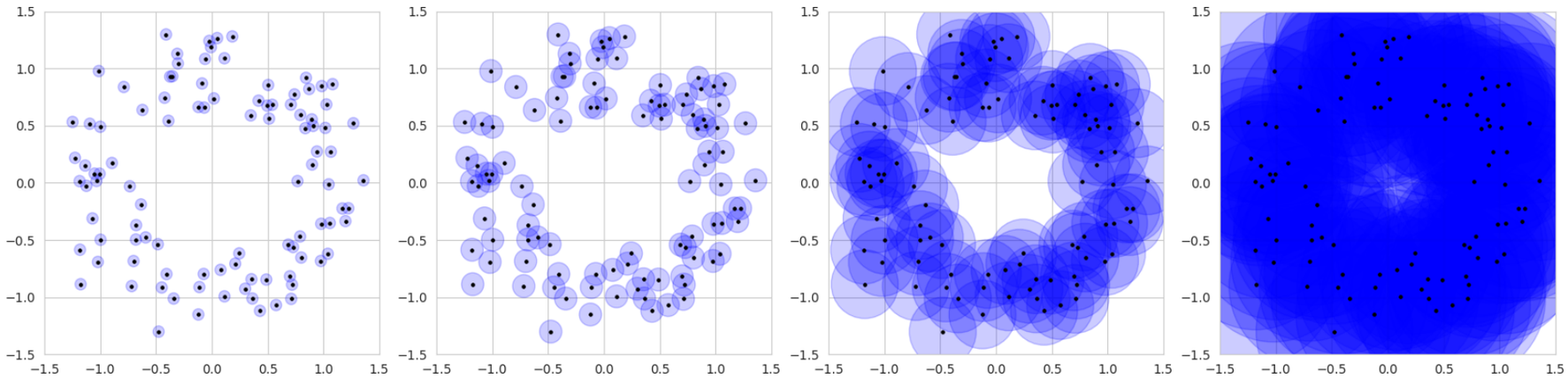
Limitations

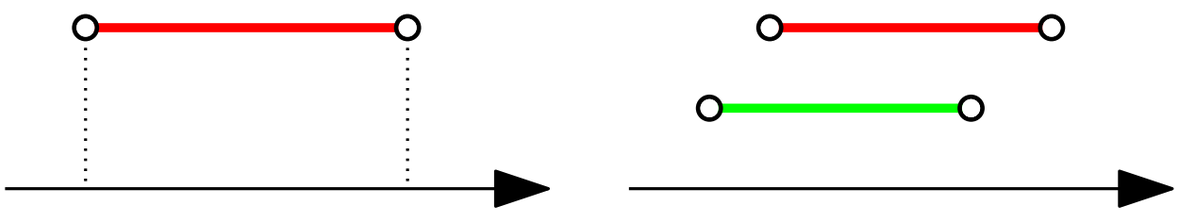
Thm: If X and Y are pre-compact metric spaces, then

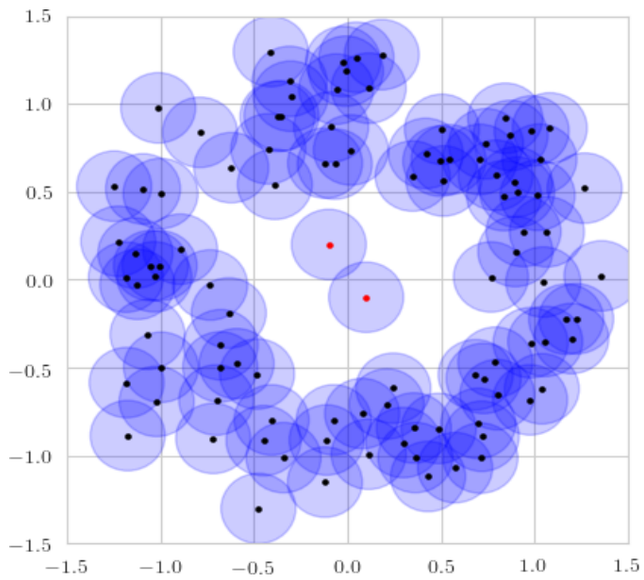
$$d_b(D_{\mathrm{Rips}}(X), D_{\mathrm{Rips}}(Y)) \leq d_{GH}(X, Y).$$

$\rightarrow$ Vietoris-Rips (or Čech, witness) filtrations become quickly prohibitively large as the size of the data increases: $O(|X|^d)$, making the practical computation of persistence almost impossible.

$\rightarrow$ Persistence diagrams of Vietoris-Rips (as well as Čech, witness,..) filtrations and Gromov-Hausdorff distance are very sensitive to noise and outliers.



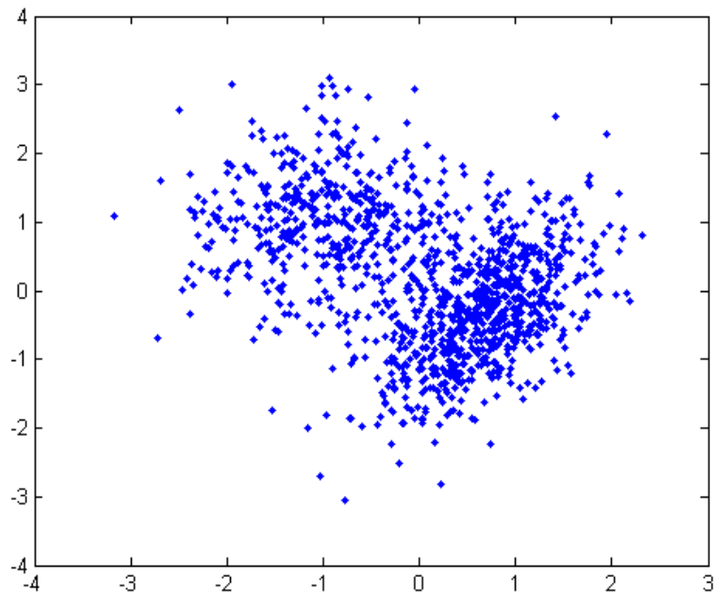


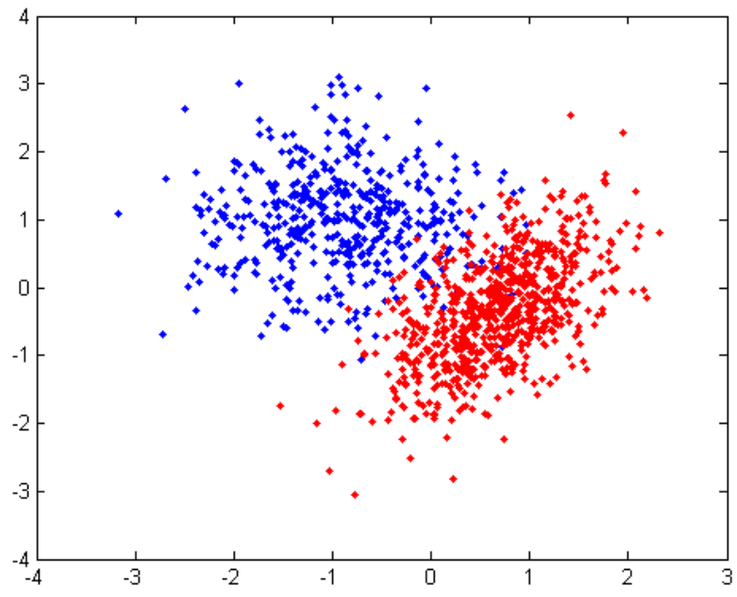


Clustering and 0-dimensional Persistent Homology

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.

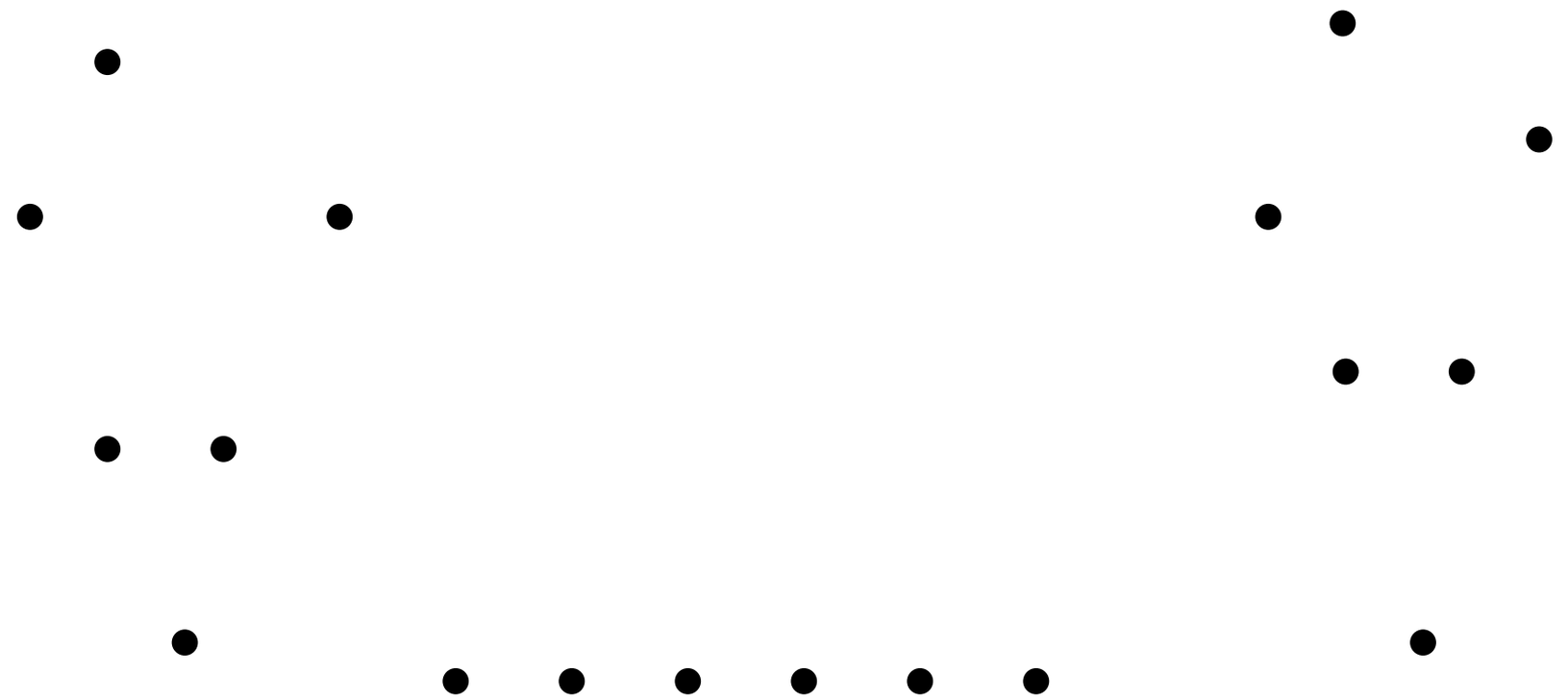




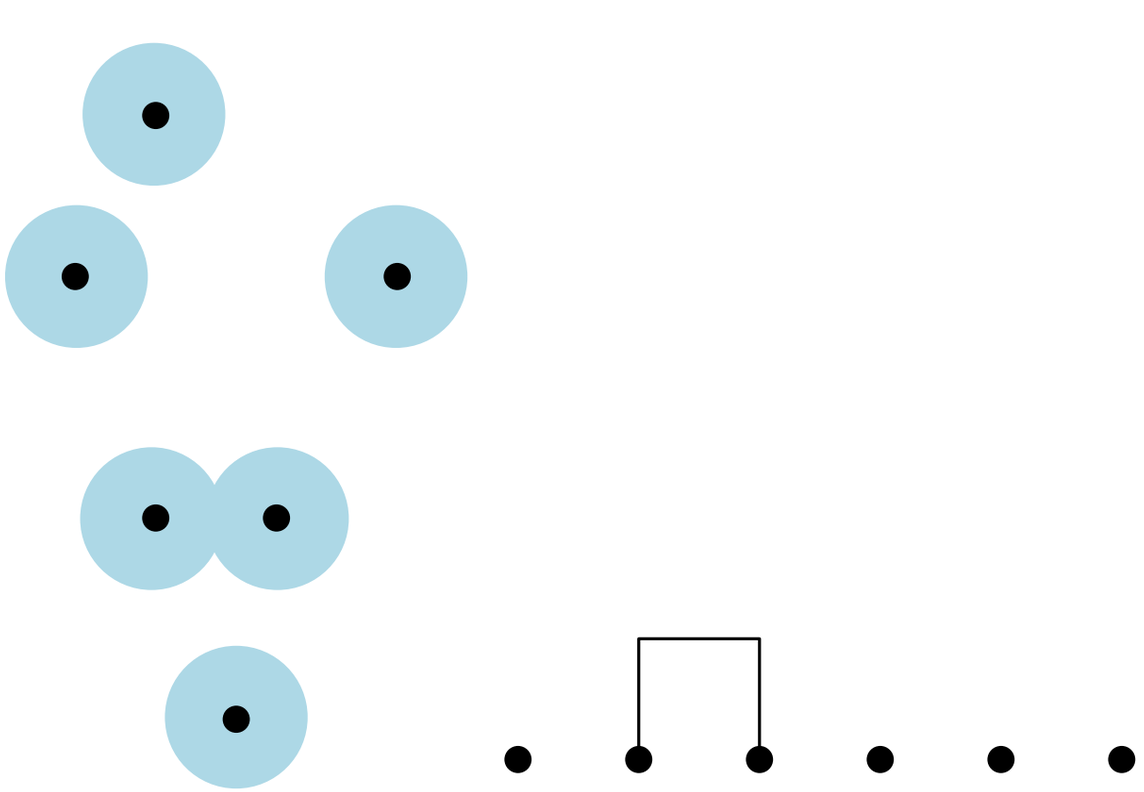
Goal: partition the data into a relevant family of subsets (clusters).

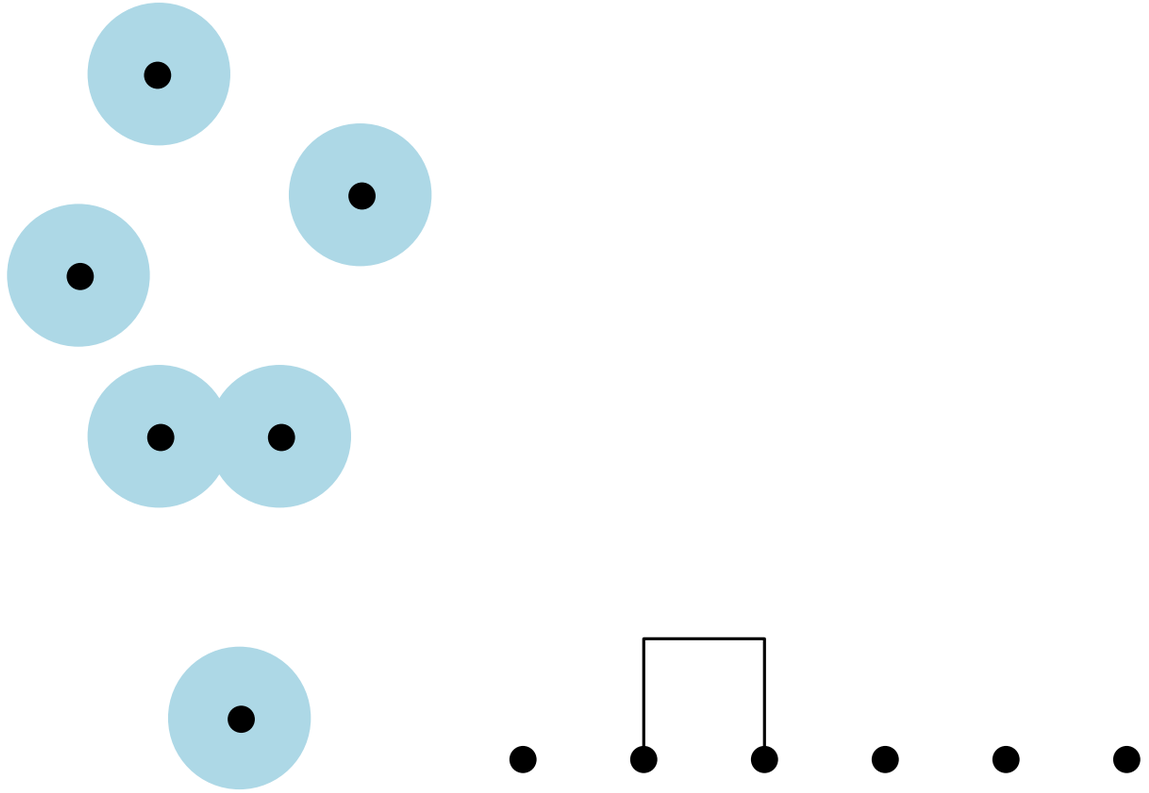
The (in)stability of dendograms

The (in)stability of dendograms

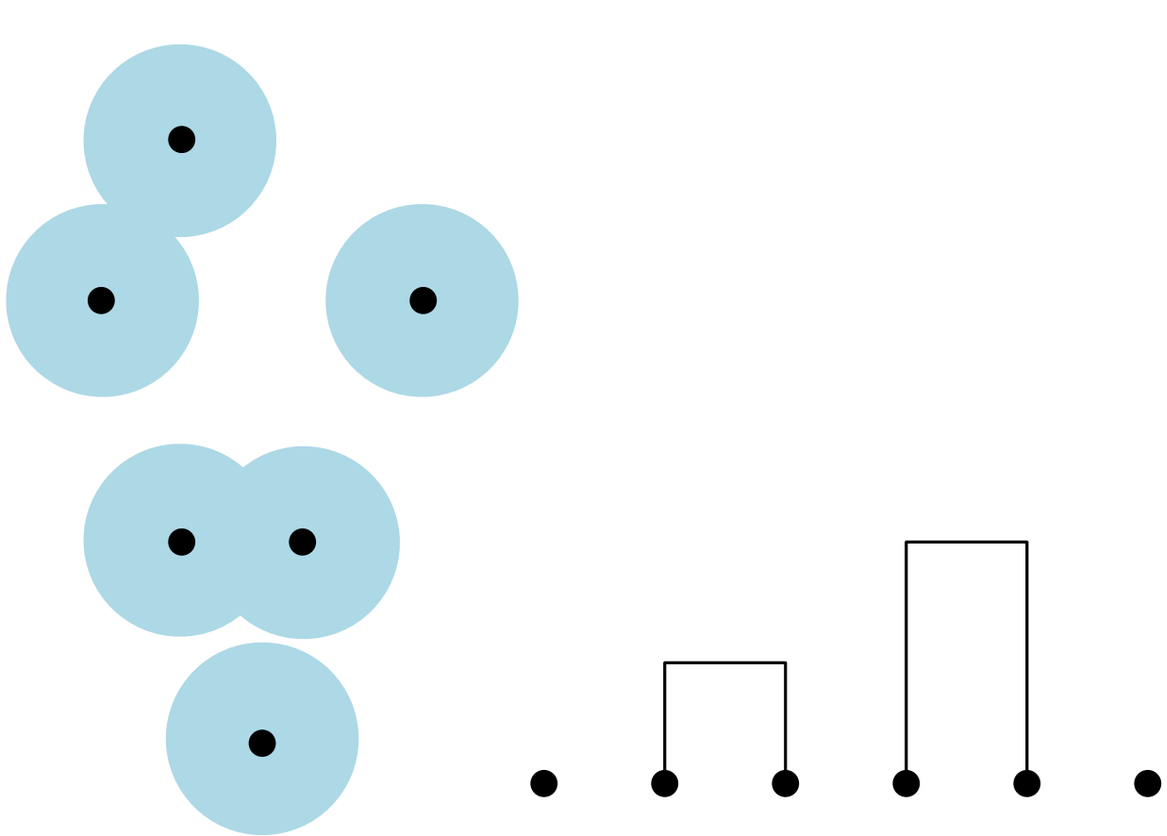


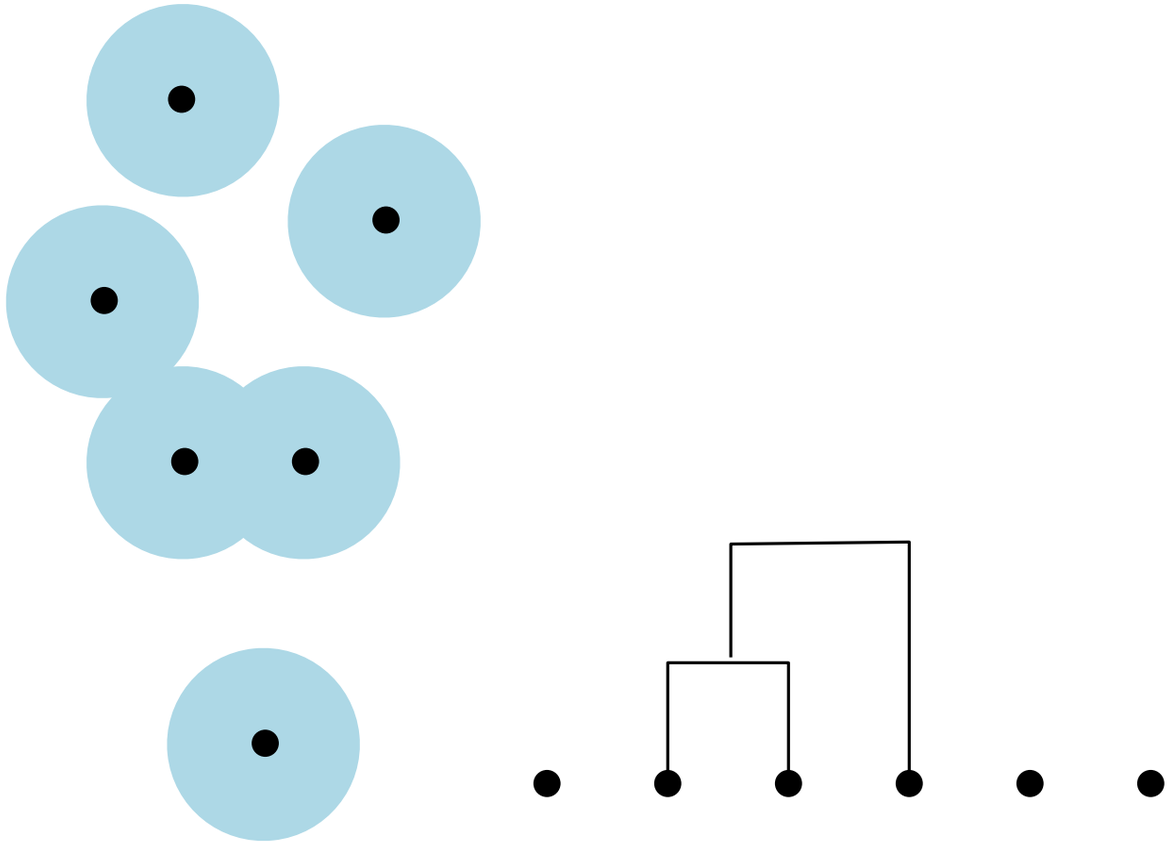
The (in)stability of dendograms



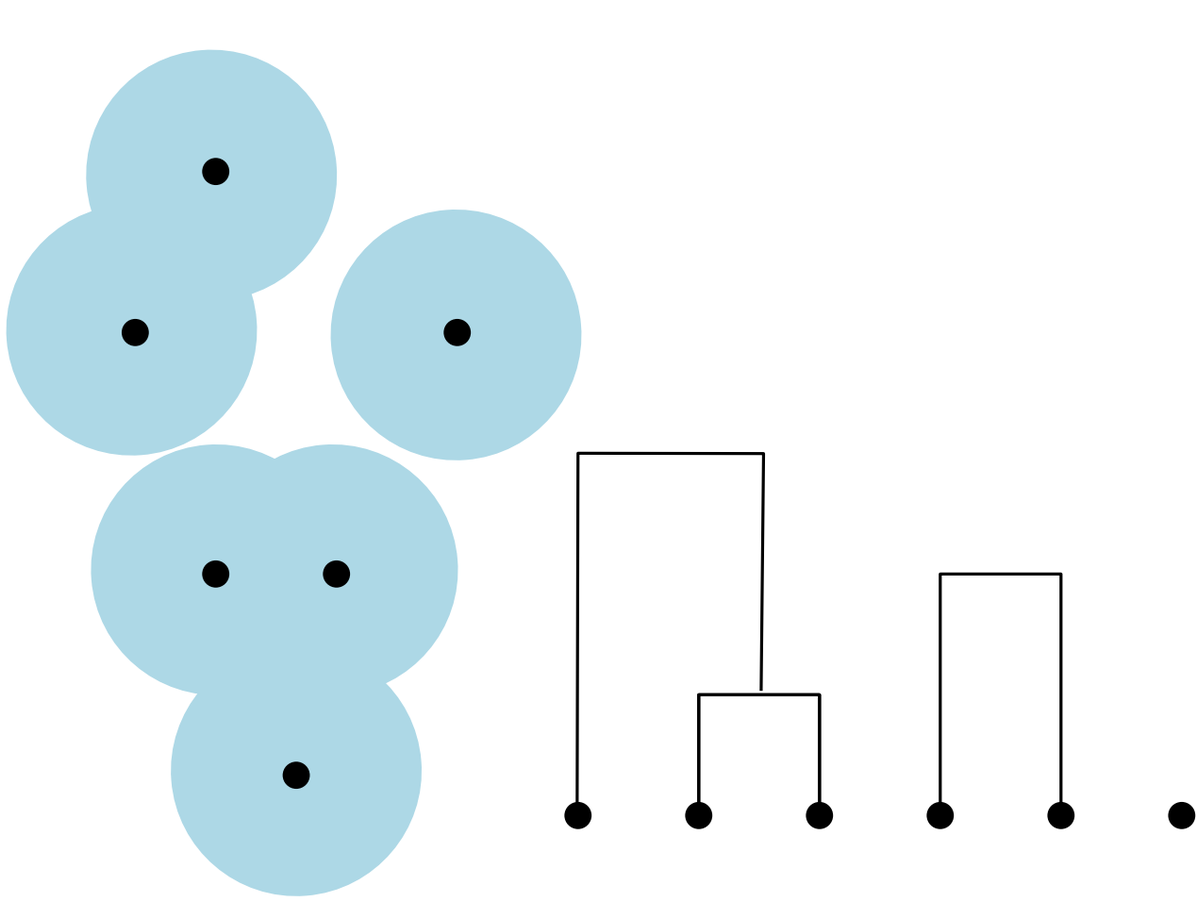


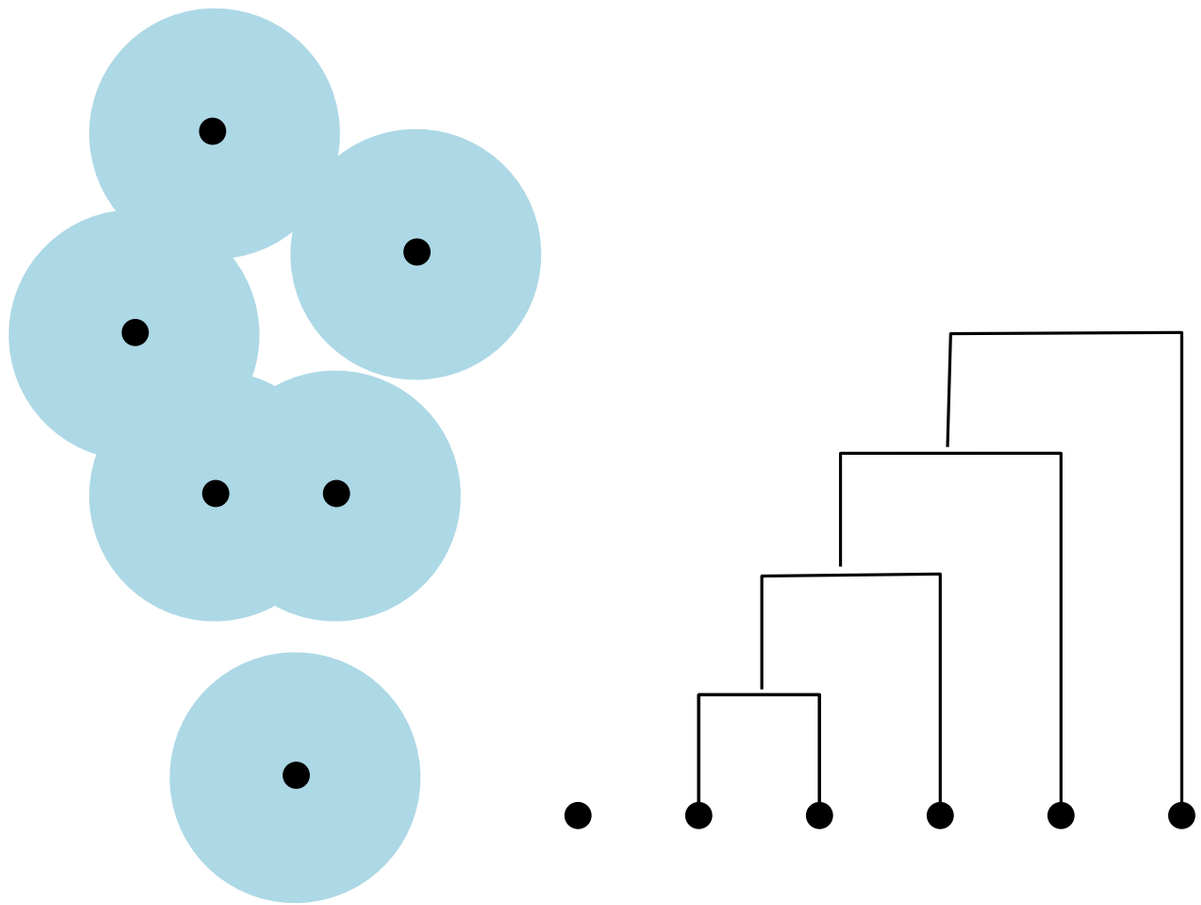
The (in)stability of dendograms



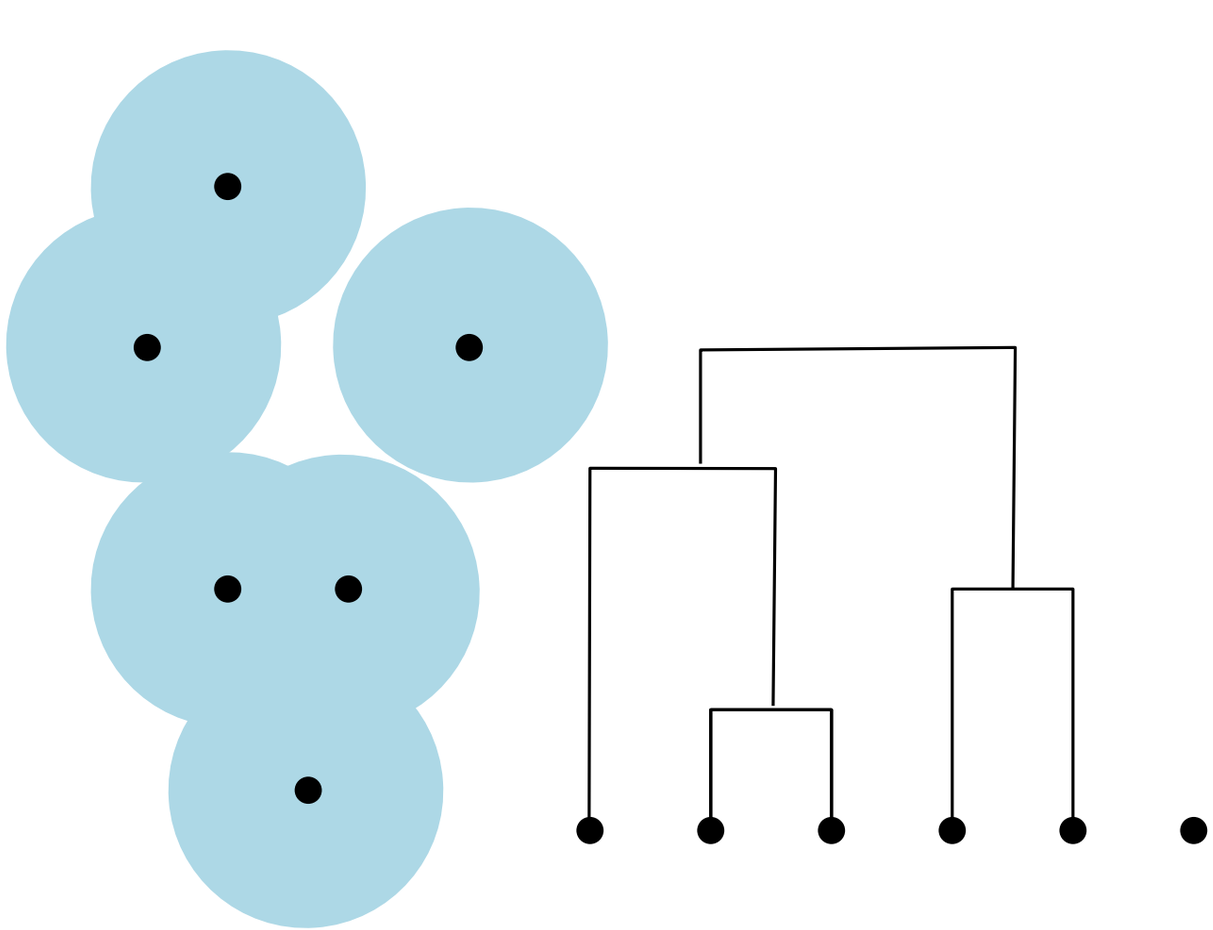


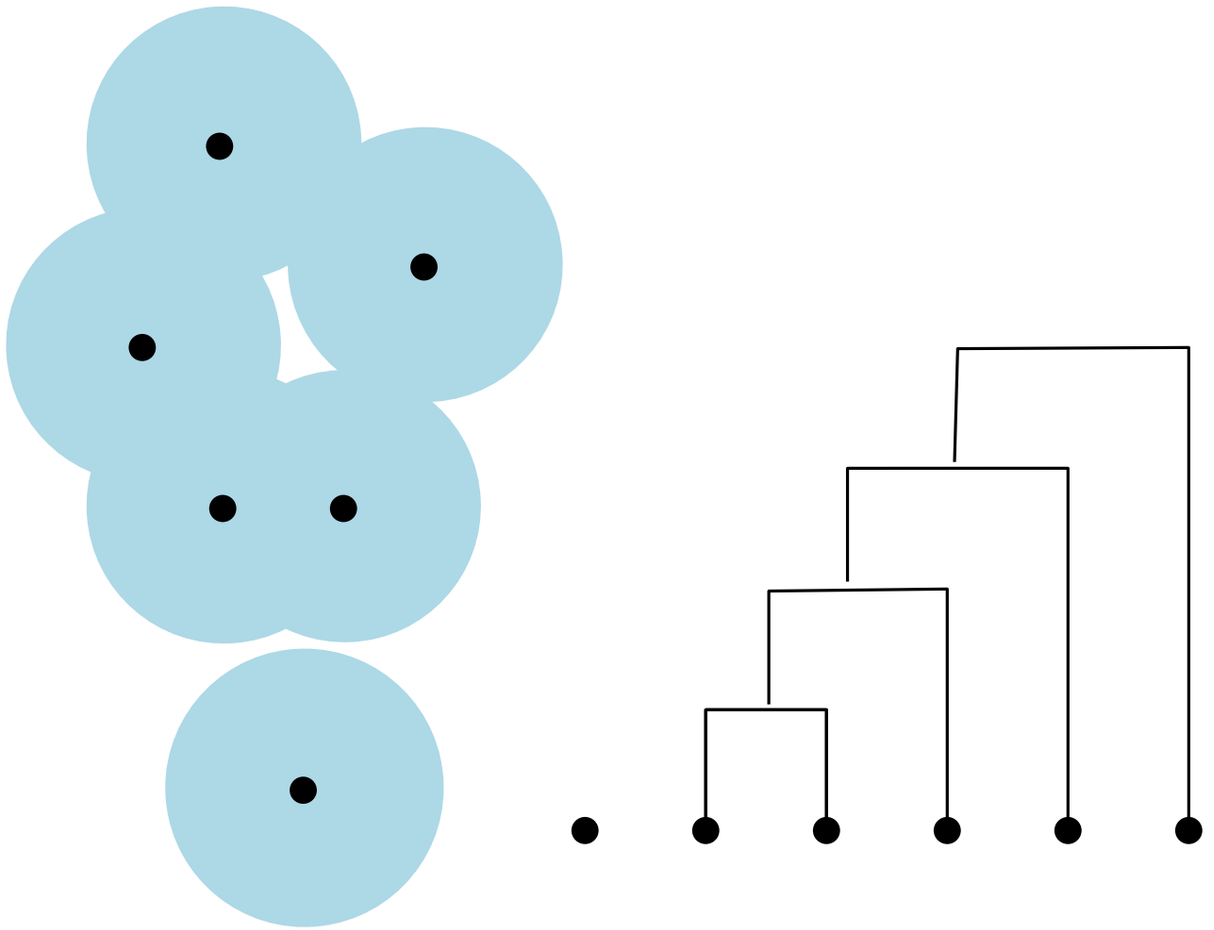
The (in)stability of dendograms



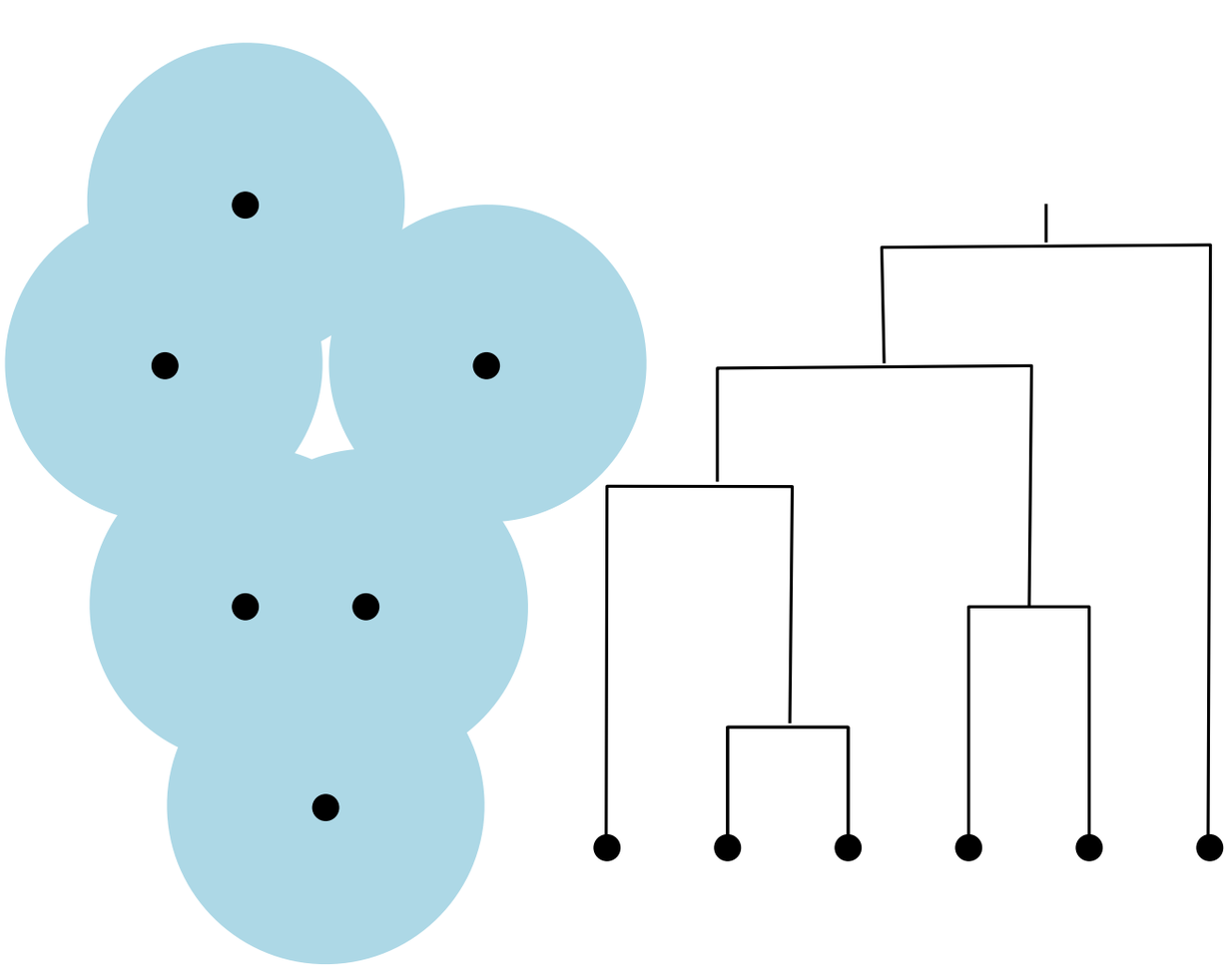


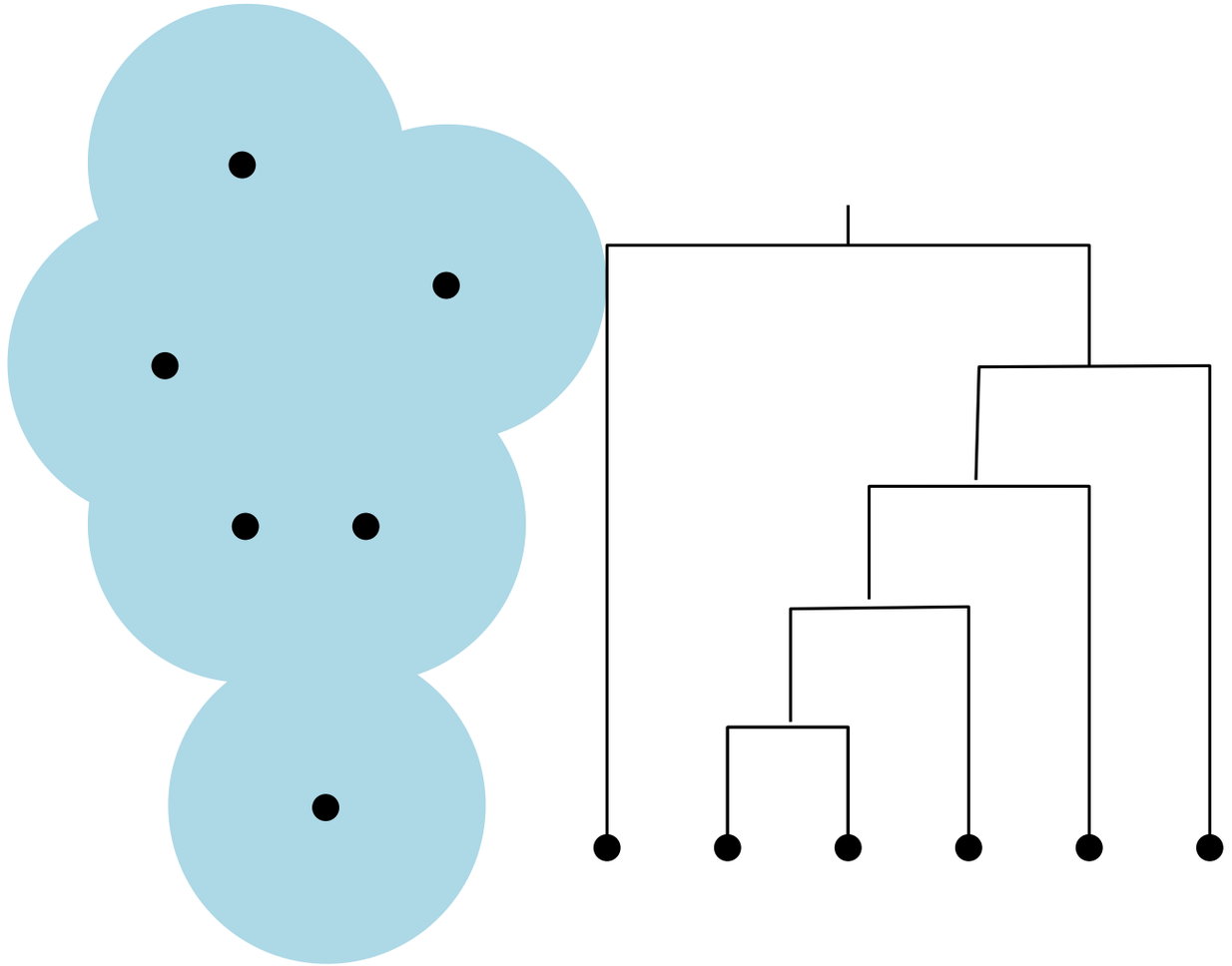
The (in)stability of dendograms





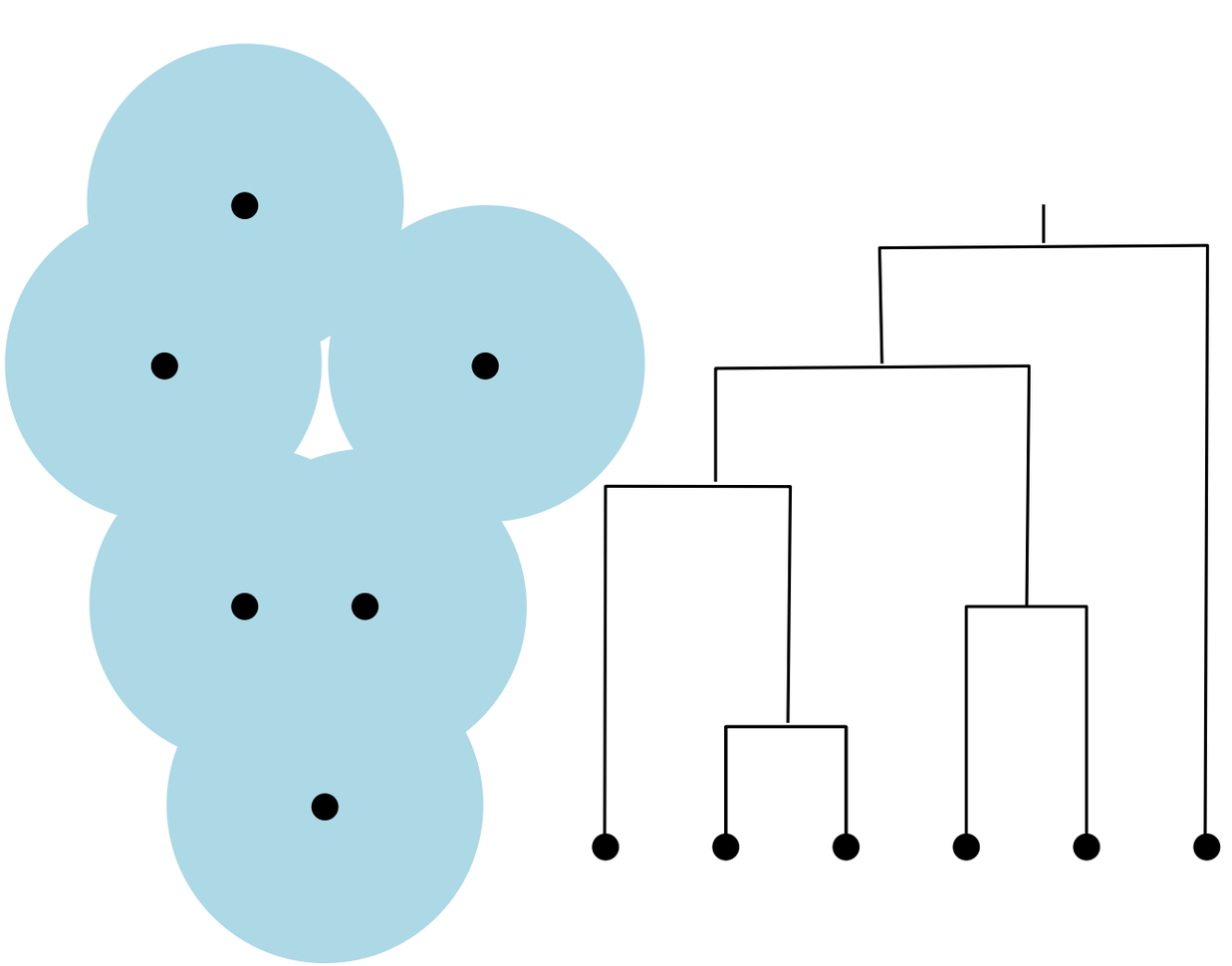
The (in)stability of dendograms

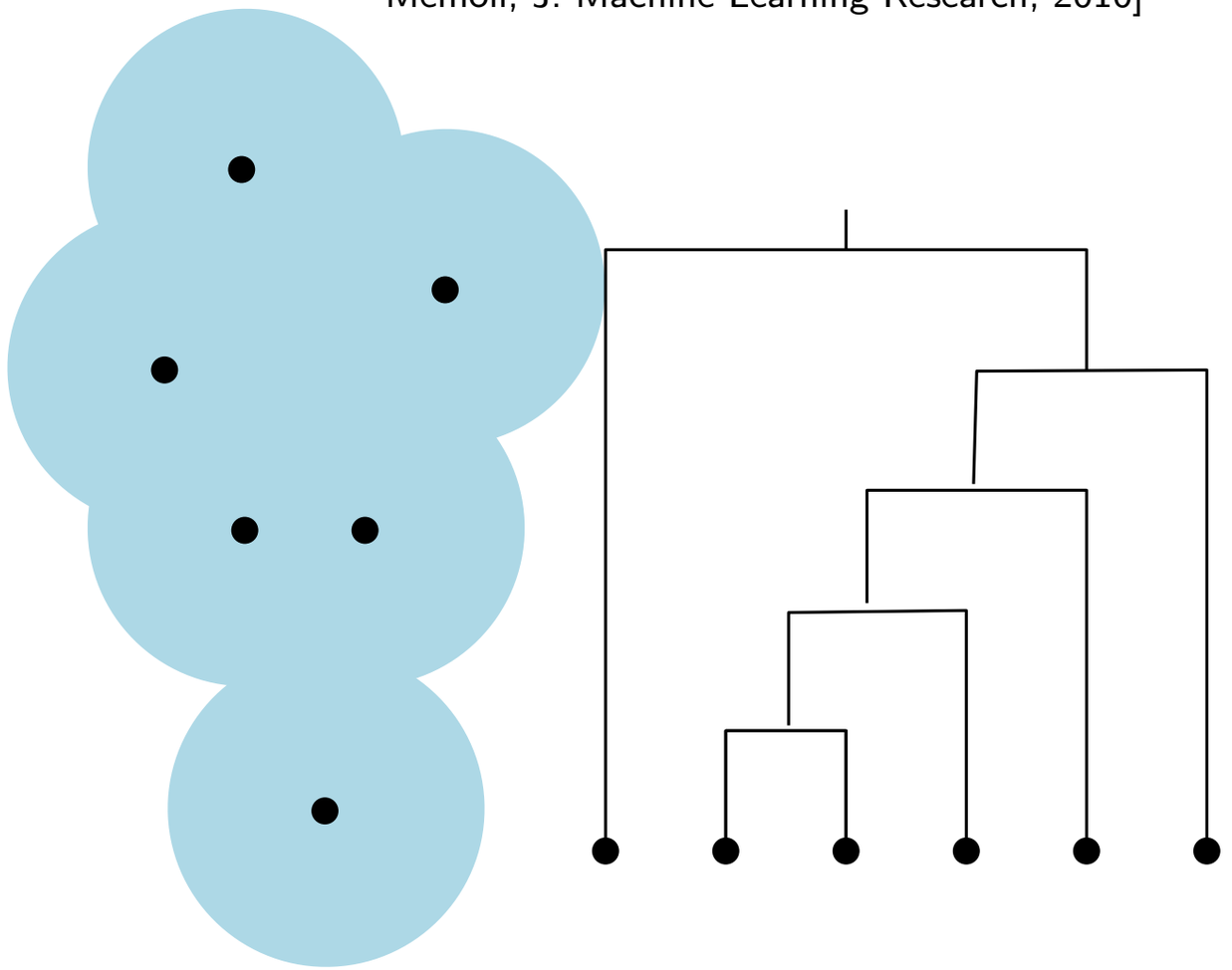




The (in)stability of dendograms

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





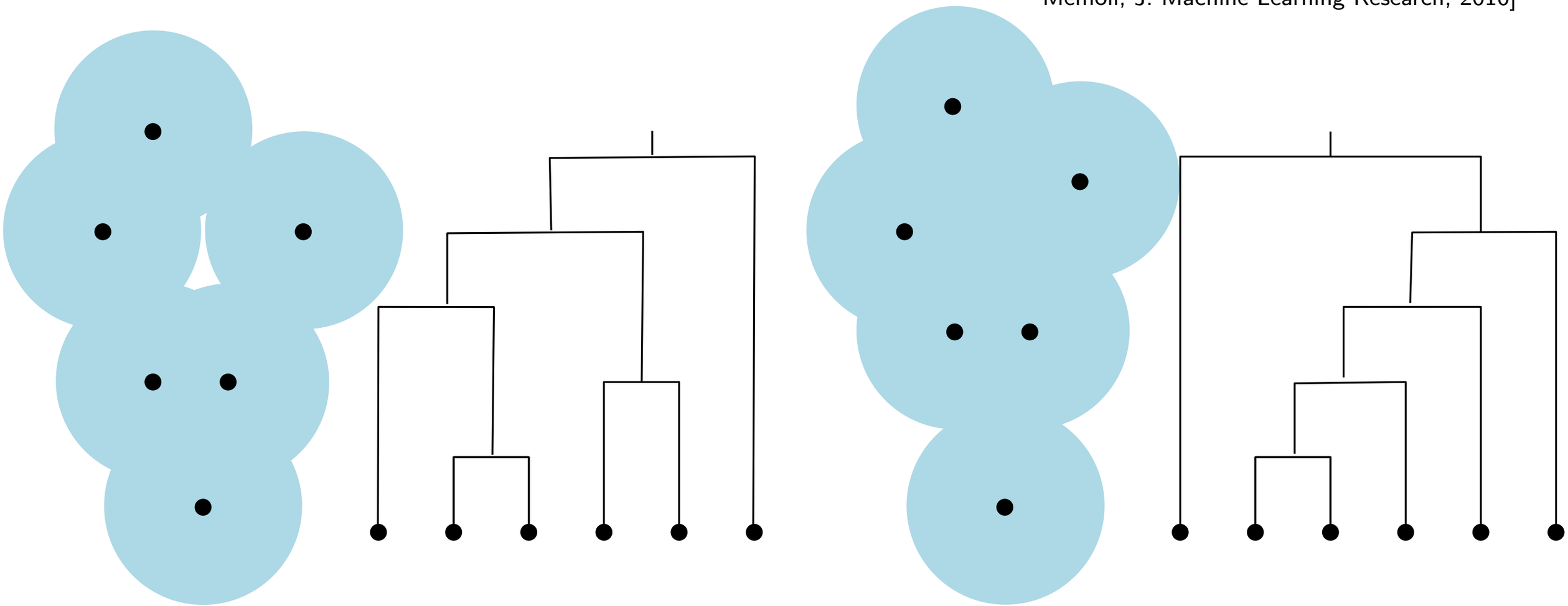
Let $d_{\mathcal{D}}(x, y) =$ height of lowest common ancestor of x, y in dendrogram $\mathcal{D}$.

ultrametric!

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

The (in)stability of dendograms

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



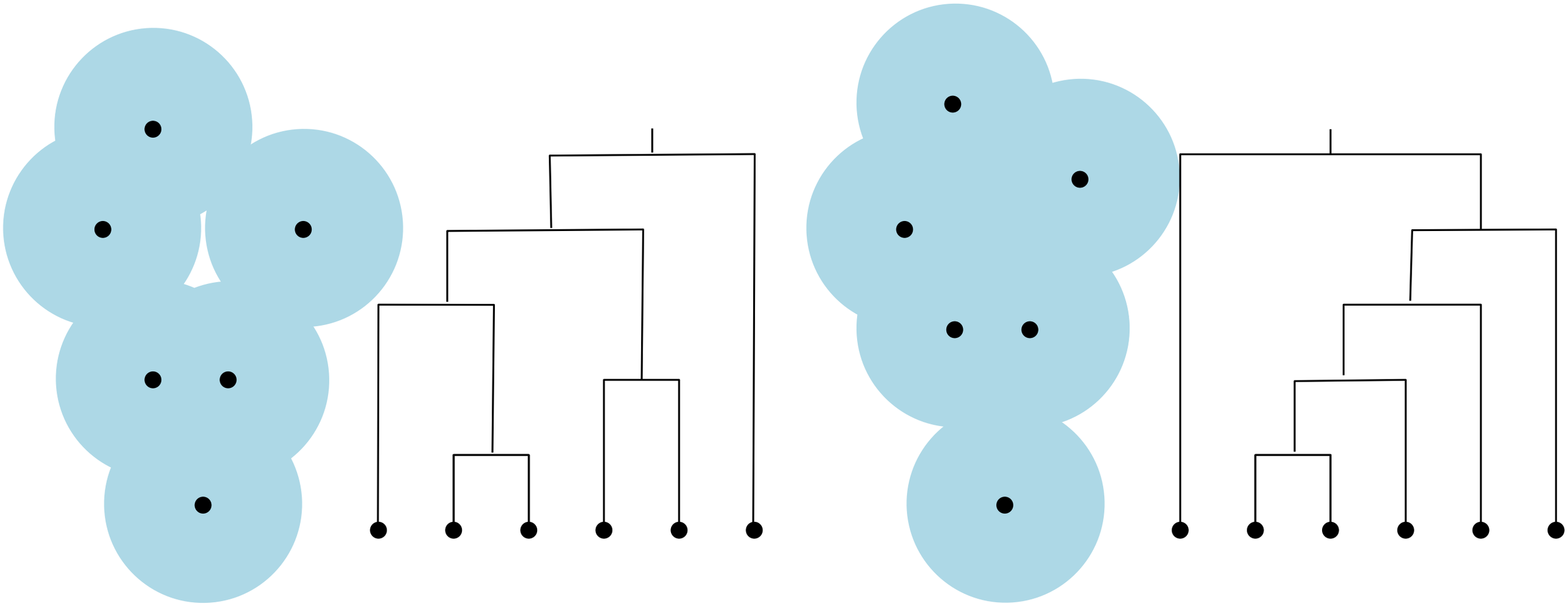
Let $d_{\mathcal{D}}(x, y) =$ height of lowest common ancestor of x, y in dendrogram $\mathcal{D}$.

ultrametric!

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

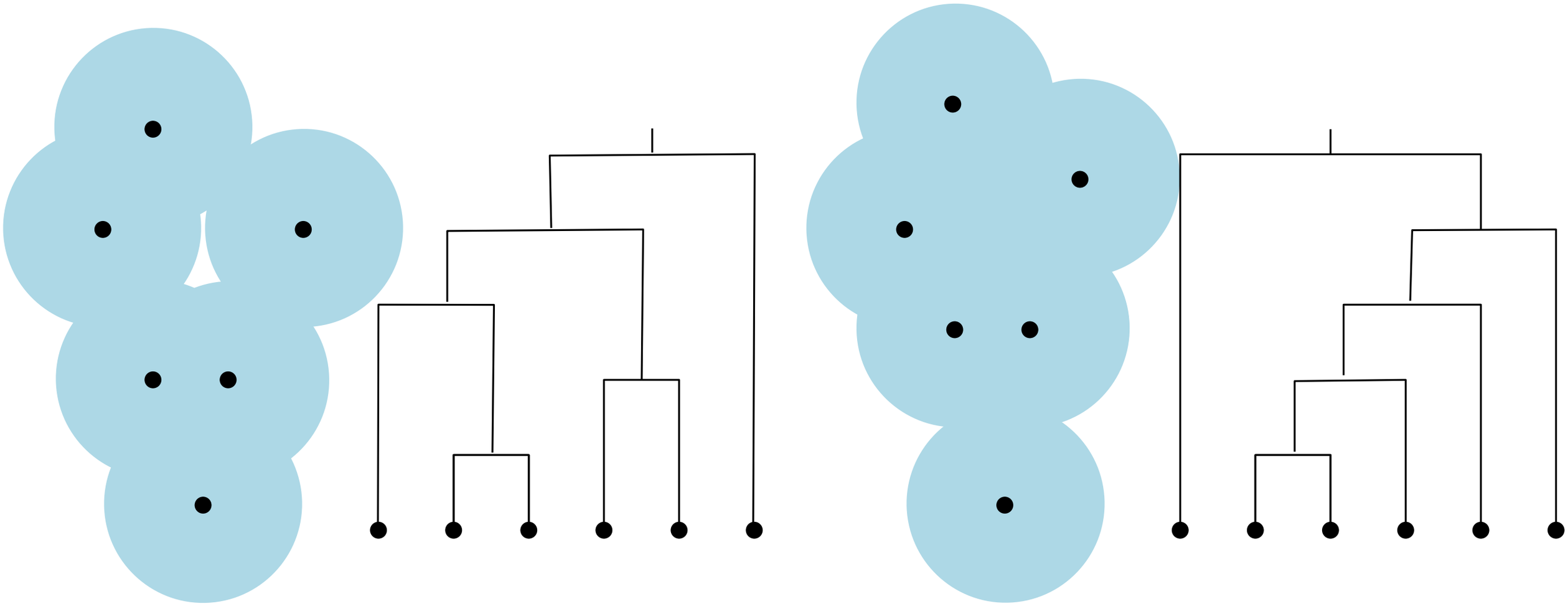
This is actually not true for complete and average clustering!

The (in)stability of dendograms



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

The (in)stability of dendograms

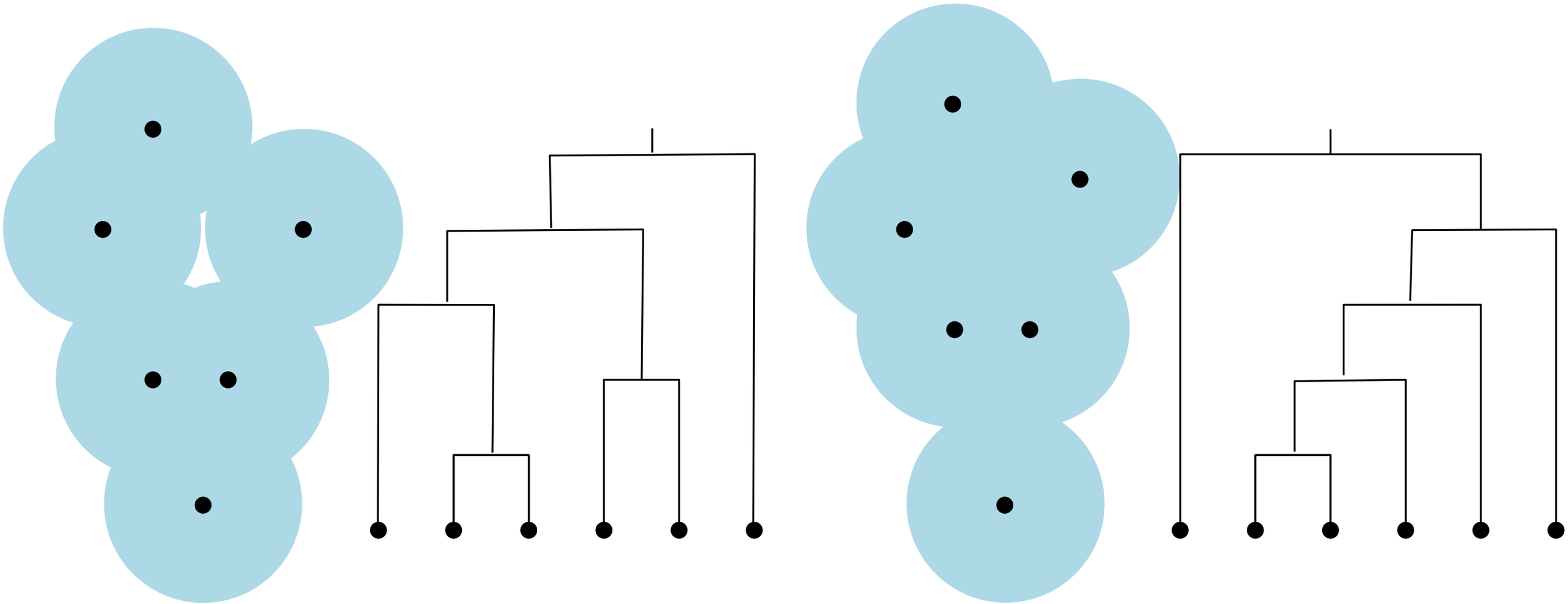


Small perturbations on the input data may lead to wide change in the structure of the trees.

However, the 'merging times' remain stable.

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

The (in)stability of dendograms



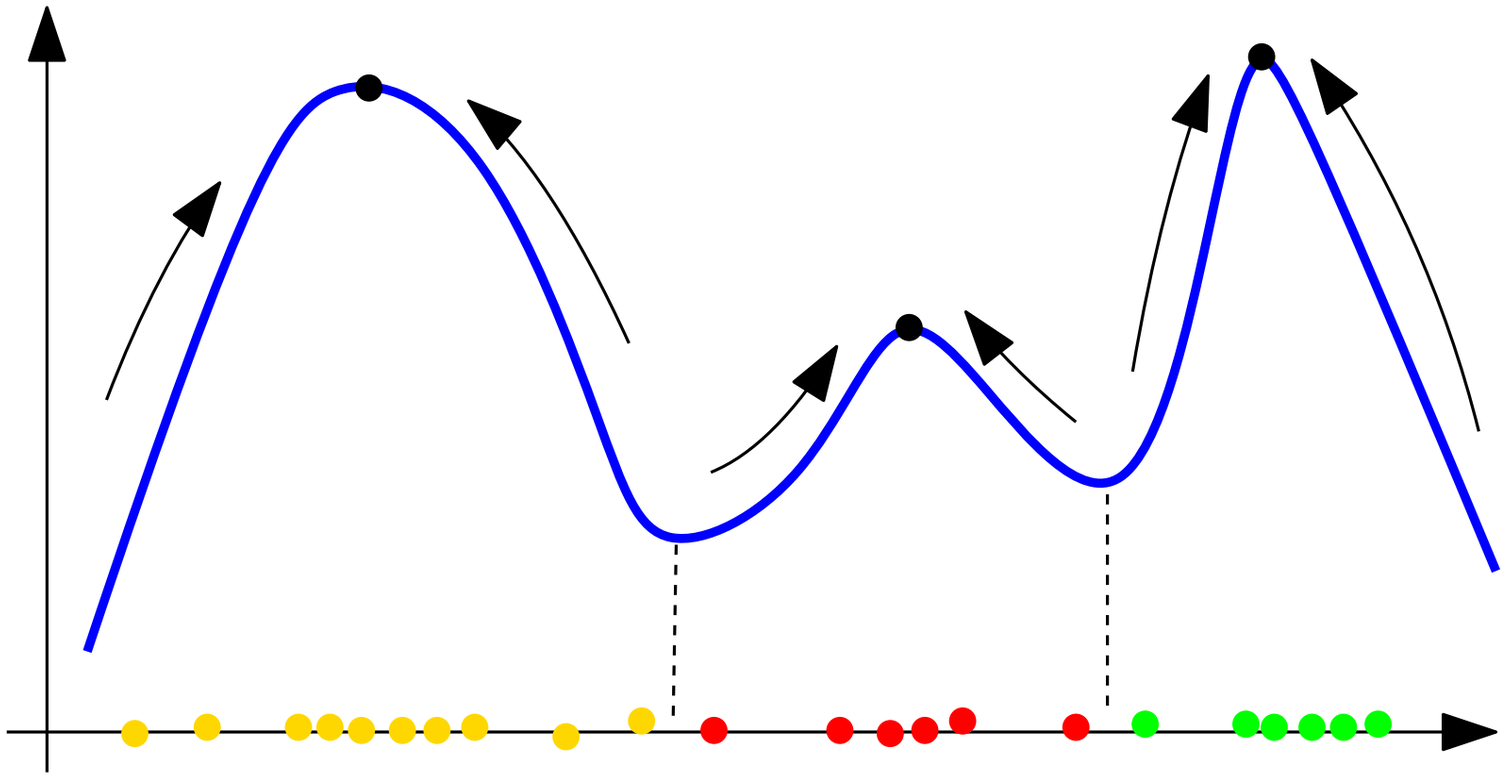
Small perturbations on the input data may lead to wide change in the structure of the trees.

Persistent homology!

However, the 'merging times' remain stable.

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

Mode seeking clustering

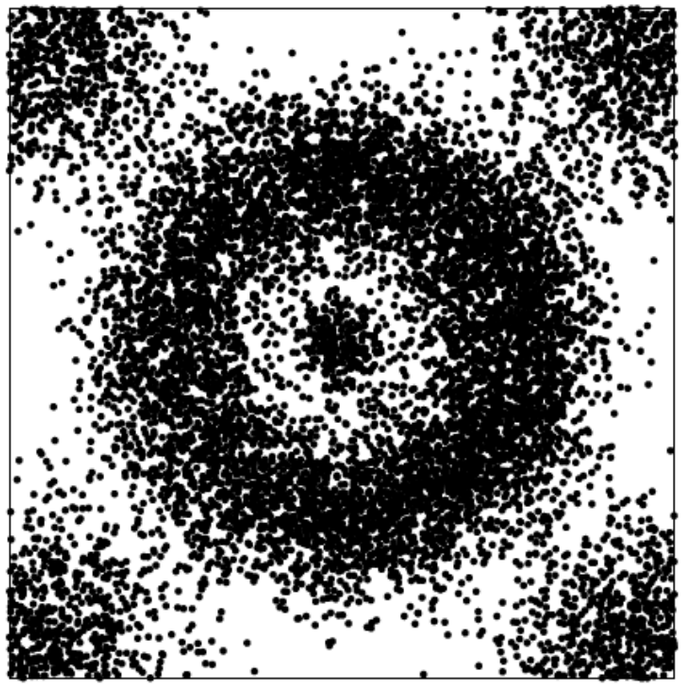


- Data points are sampled according to some (unknown) probability density.
- Clusters = basins of attractions of the density.

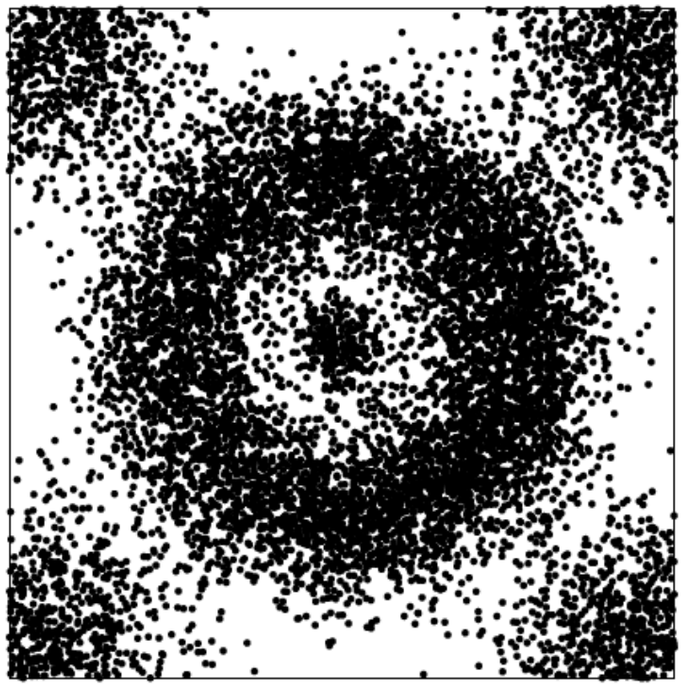
Two approaches:

- **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].

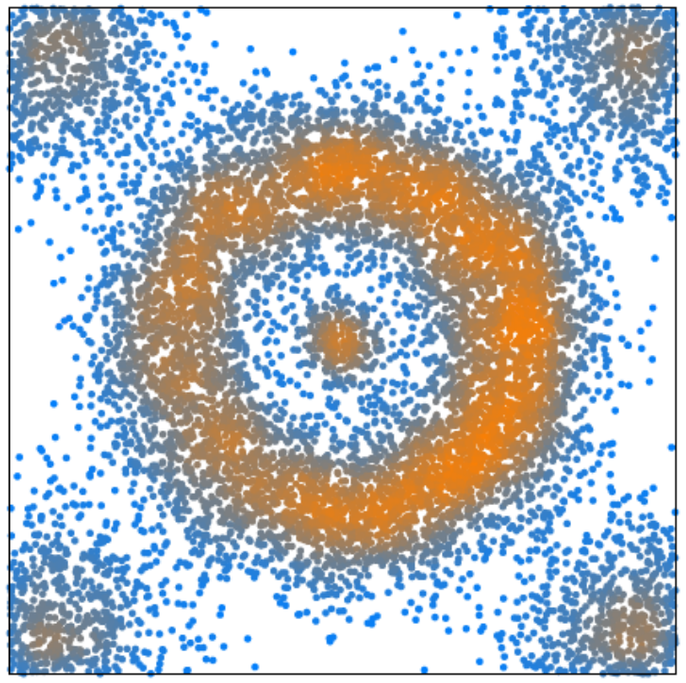
The Koonz, Narendra and Fukunaga algorithm (1976)



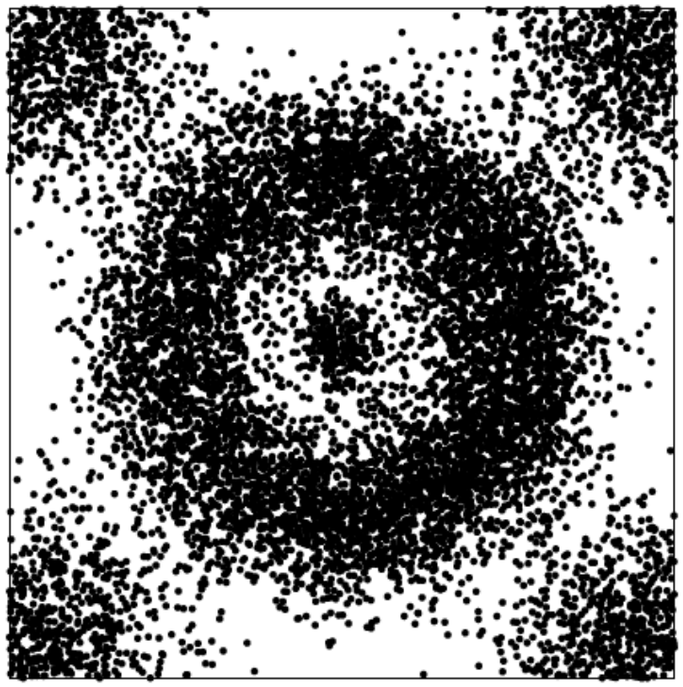
The Koonz, Narendra and Fukunaga algorithm (1976)



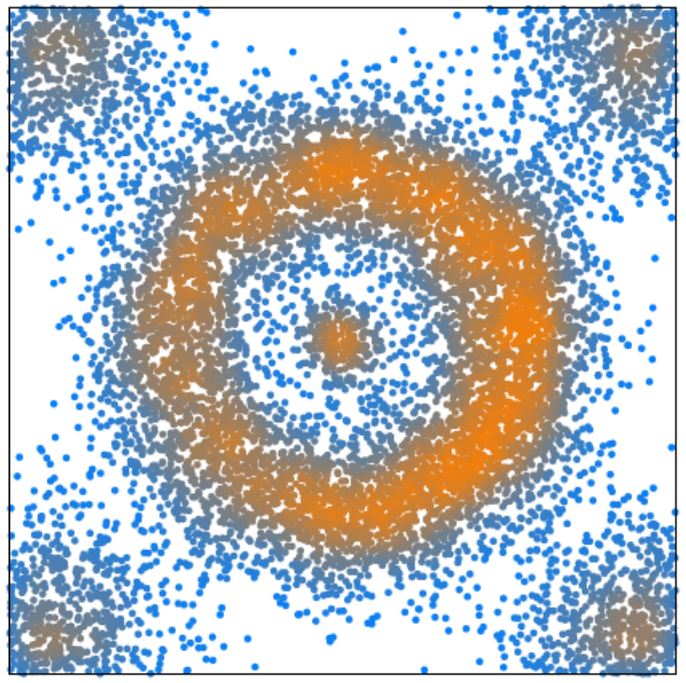
Density estimation



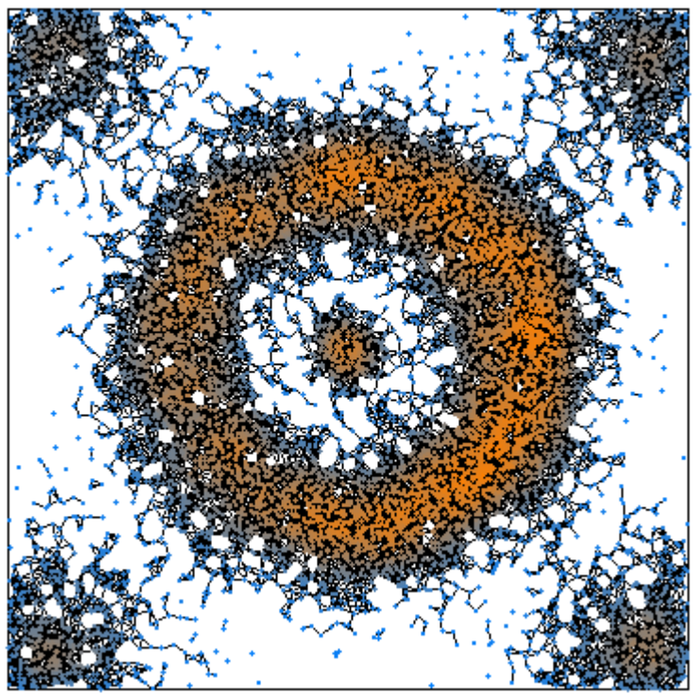
The Koonz, Narendra and Fukunaga algorithm (1976)



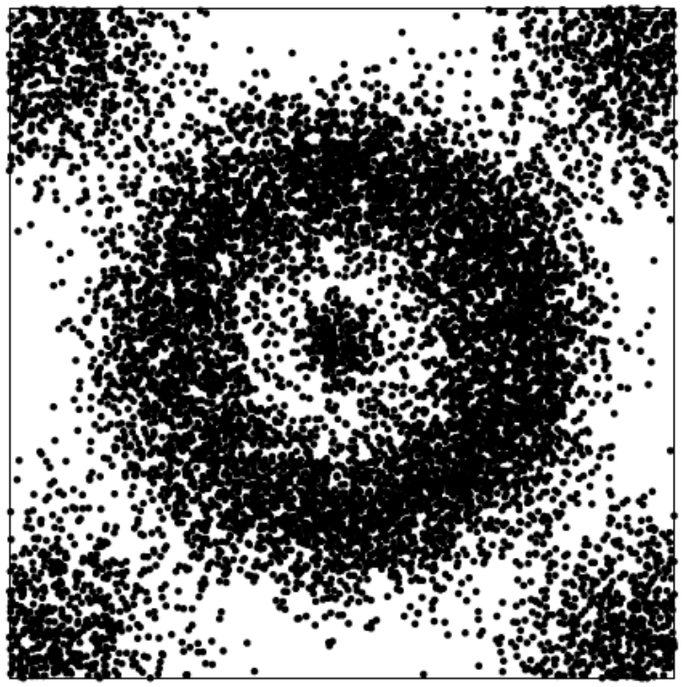
Density estimation



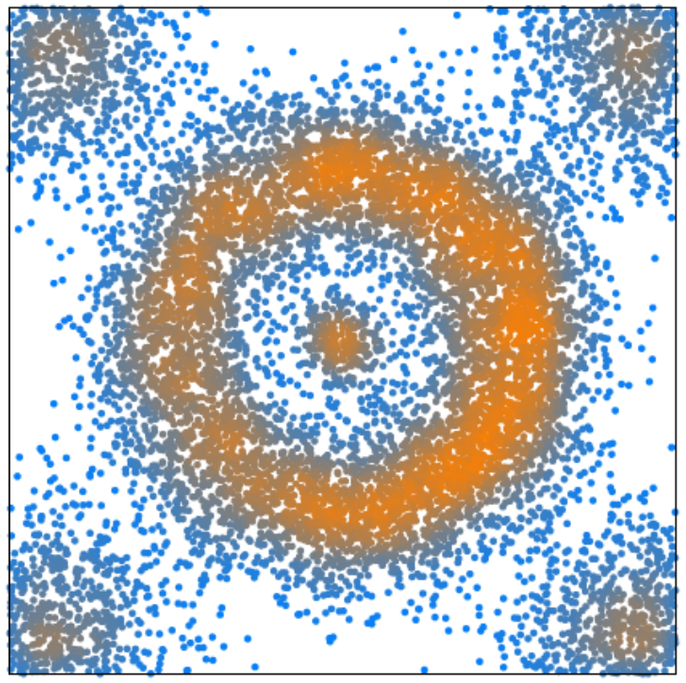
Neighborhood graph



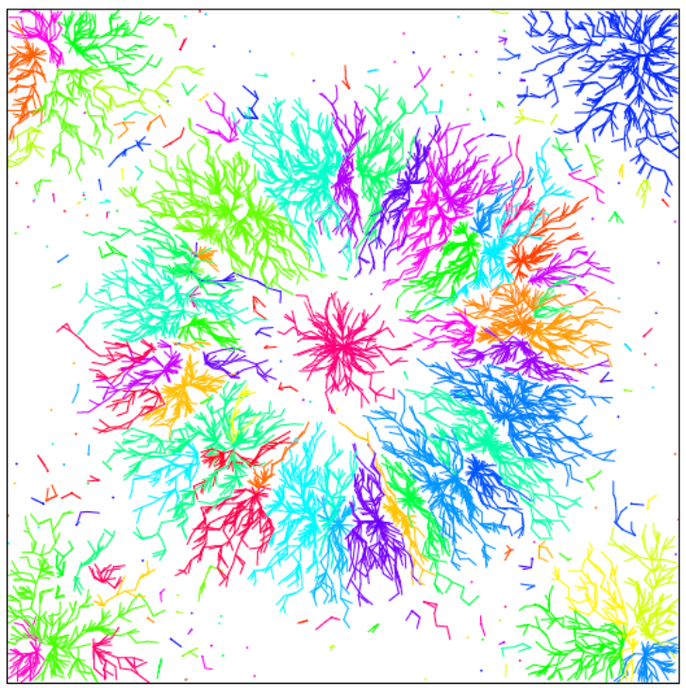
The Koonz, Narendra and Fukunaga algorithm (1976)



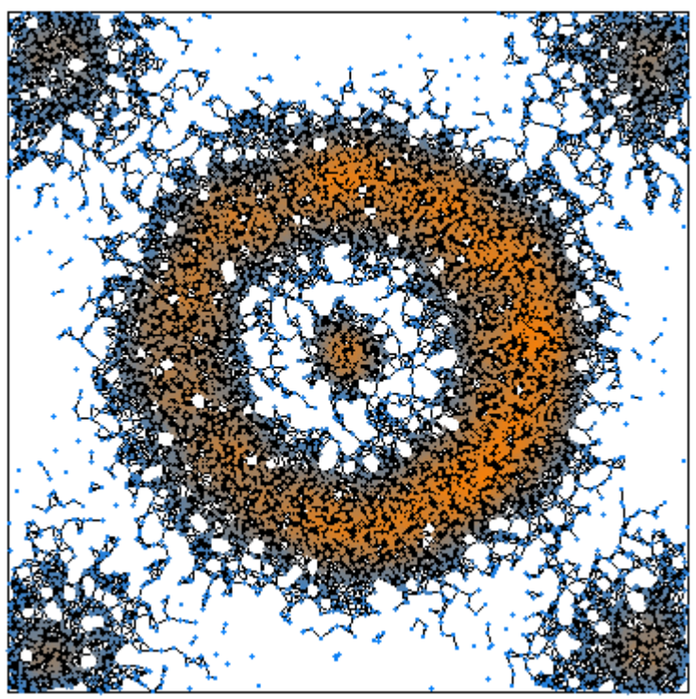
Density estimation



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)

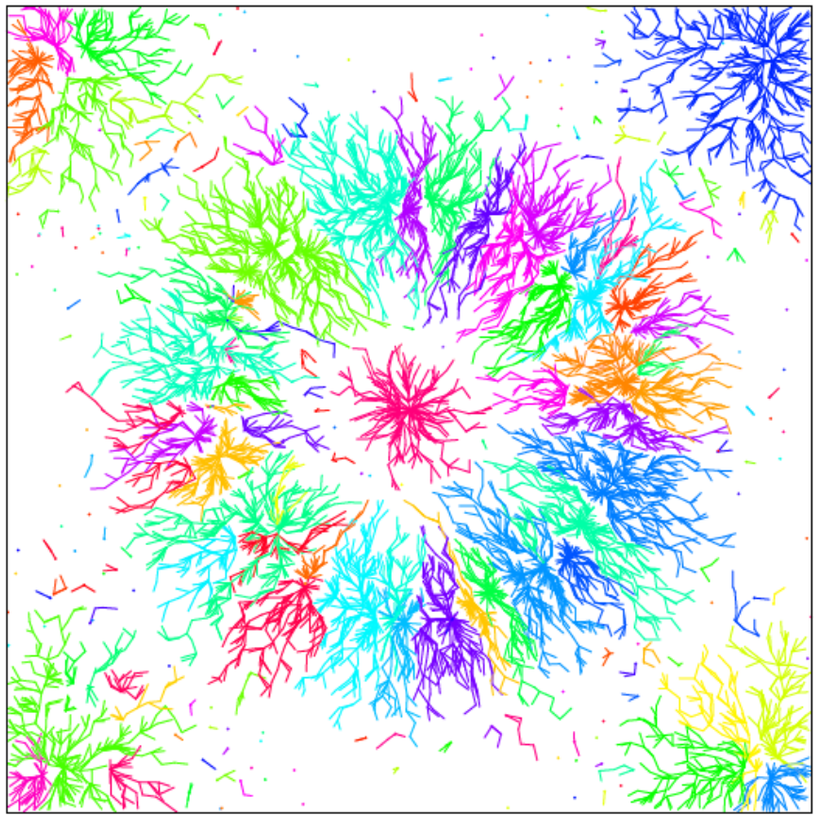
Sort the vertex indices $\{1, 2, \ldots, n\}$ in decreasing order: $\hat{f}(1) \geq \cdots \geq \hat{f}(n)$;
Initialize a union-find data structure $\mathcal{U}$ and two vectors g, r of size n;

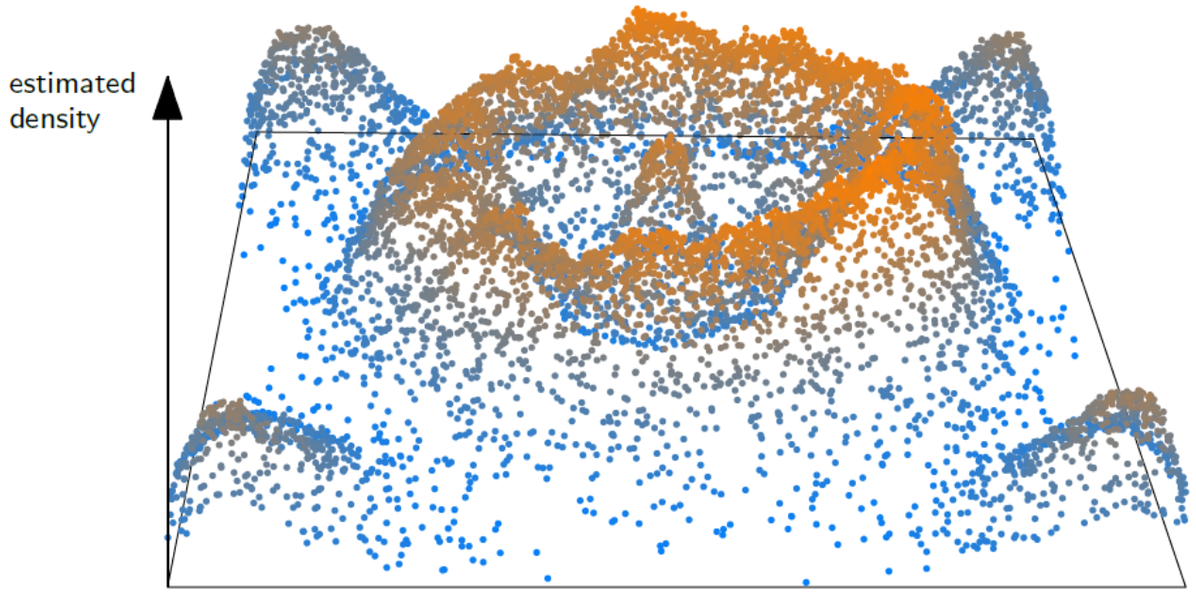
for $i = 1$ to n **do**
  Let N be the set of neighbors of i in G that have indices higher than i;
  if $N = \emptyset$
    Create a new entry e in $\mathcal{U}$ and attach vertex i to it: $\mathcal{U}$.`MakeSet`(i);
    $r(e) \leftarrow i$ *// $r(e)$ stores the root vertex associated with the entry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

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

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

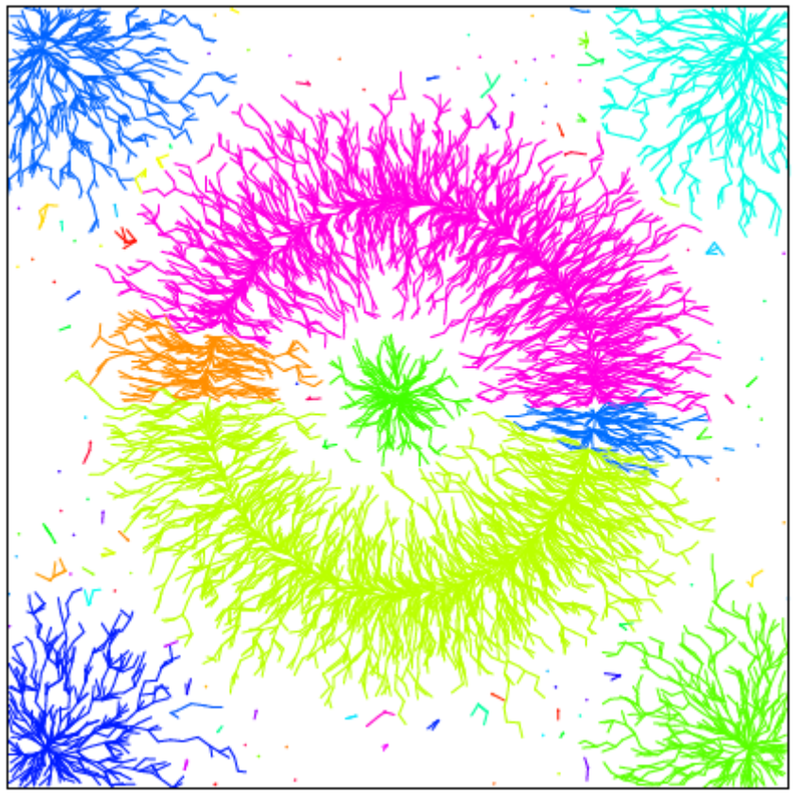


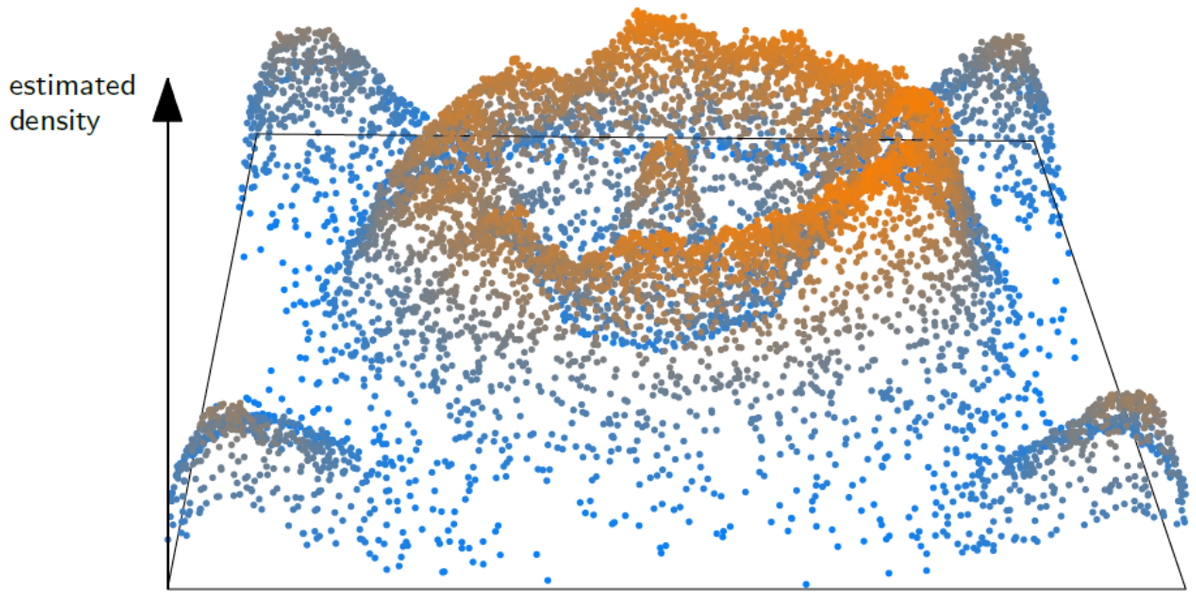


As many clusters as local maxima of density estimate $\rightarrow$ sensitivity to noise!

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:





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

The choice of the neighborhood graph results in wide changes in the output.

The Koonz, Narendra and Fukunaga algorithm (1976)

Drawbacks:

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

The choice of the neighborhood graph results in wide changes in the output.

Approaches to overcome these issues:

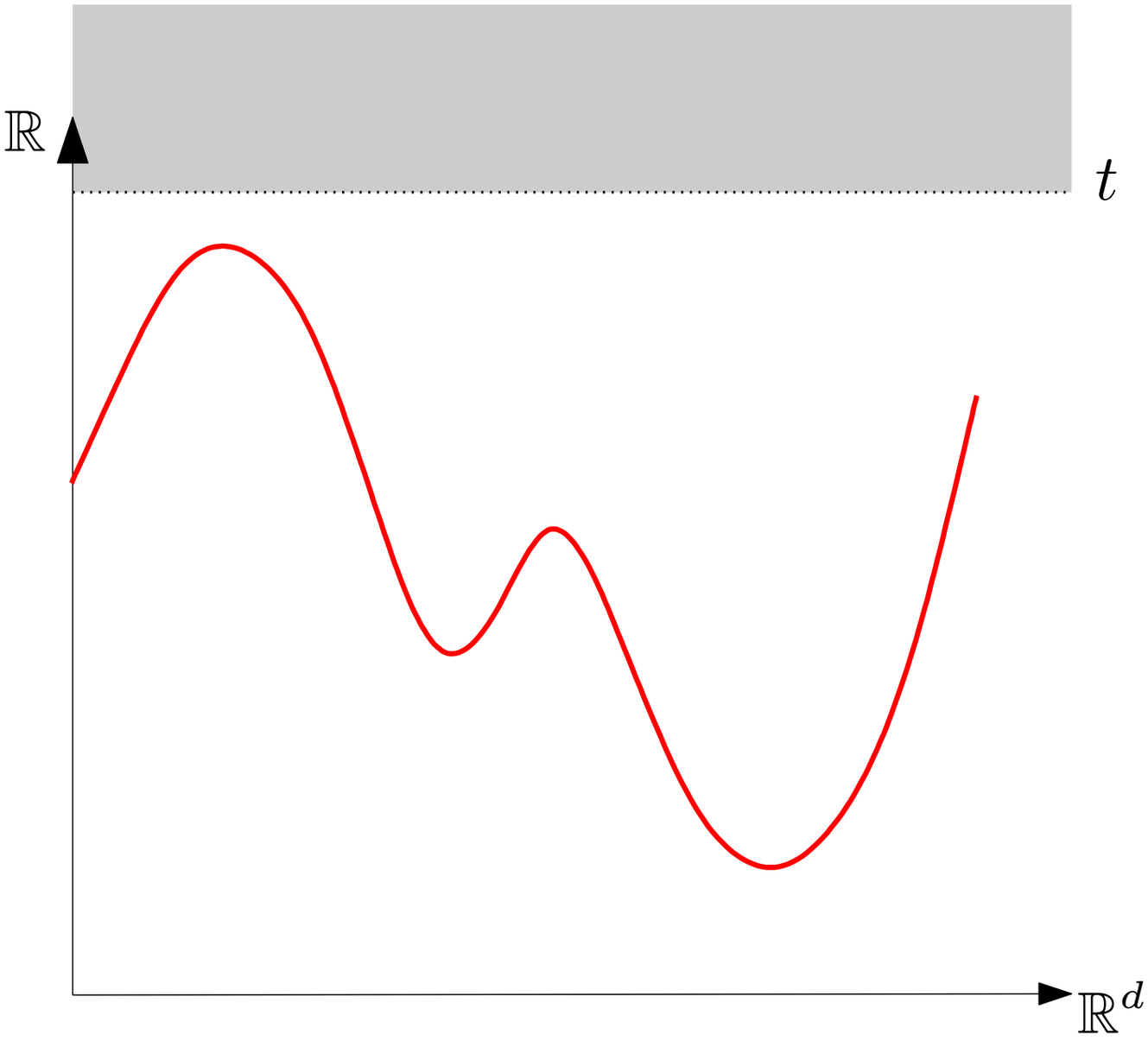
Smooth out the density estimate (e.g. mean-shift)... But how to choose the smoothing parameter?

Merge clusters with 0-dimensional persistent homology!

0-dimensional PH of density

Given a probability density f:

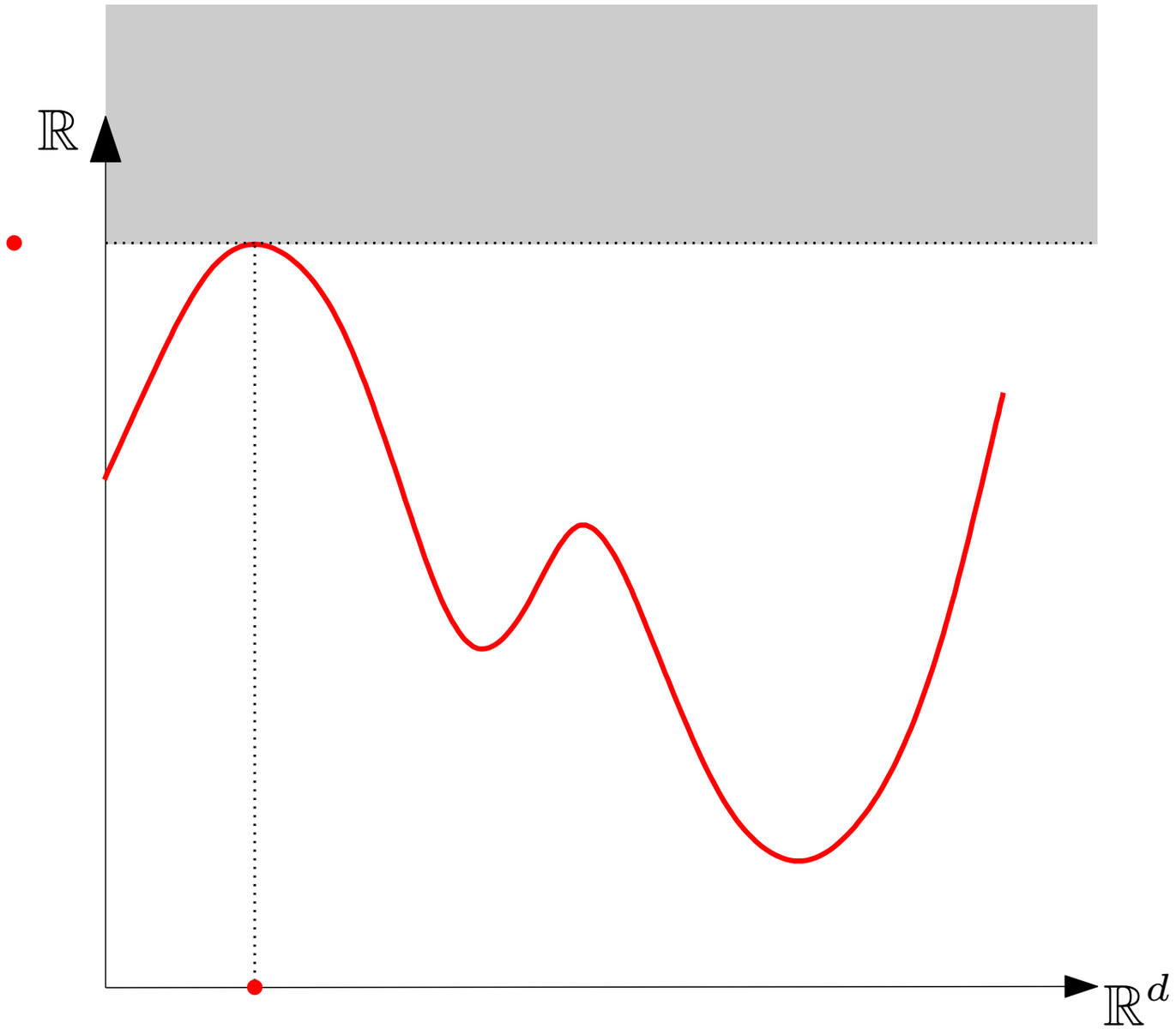
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

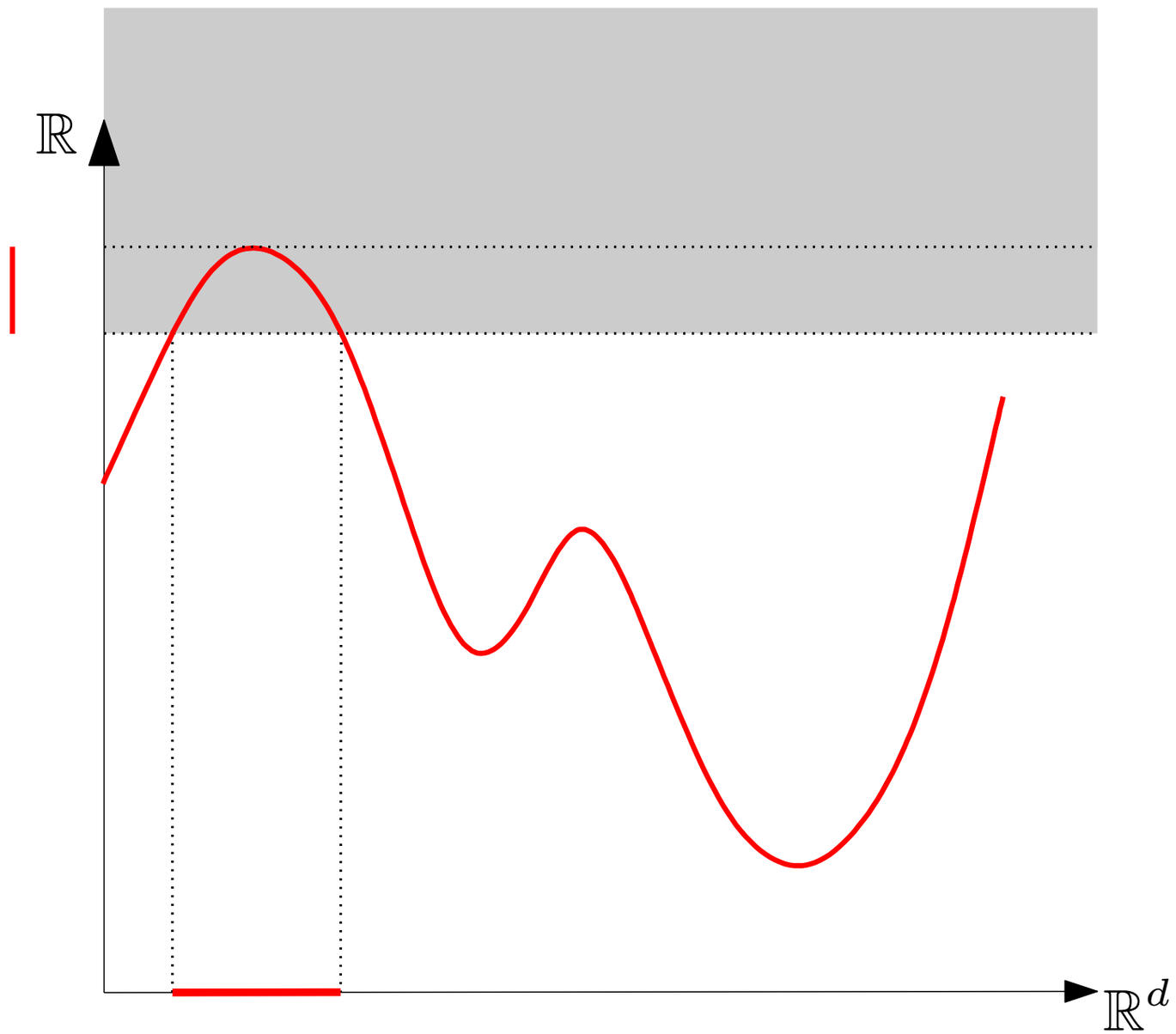
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

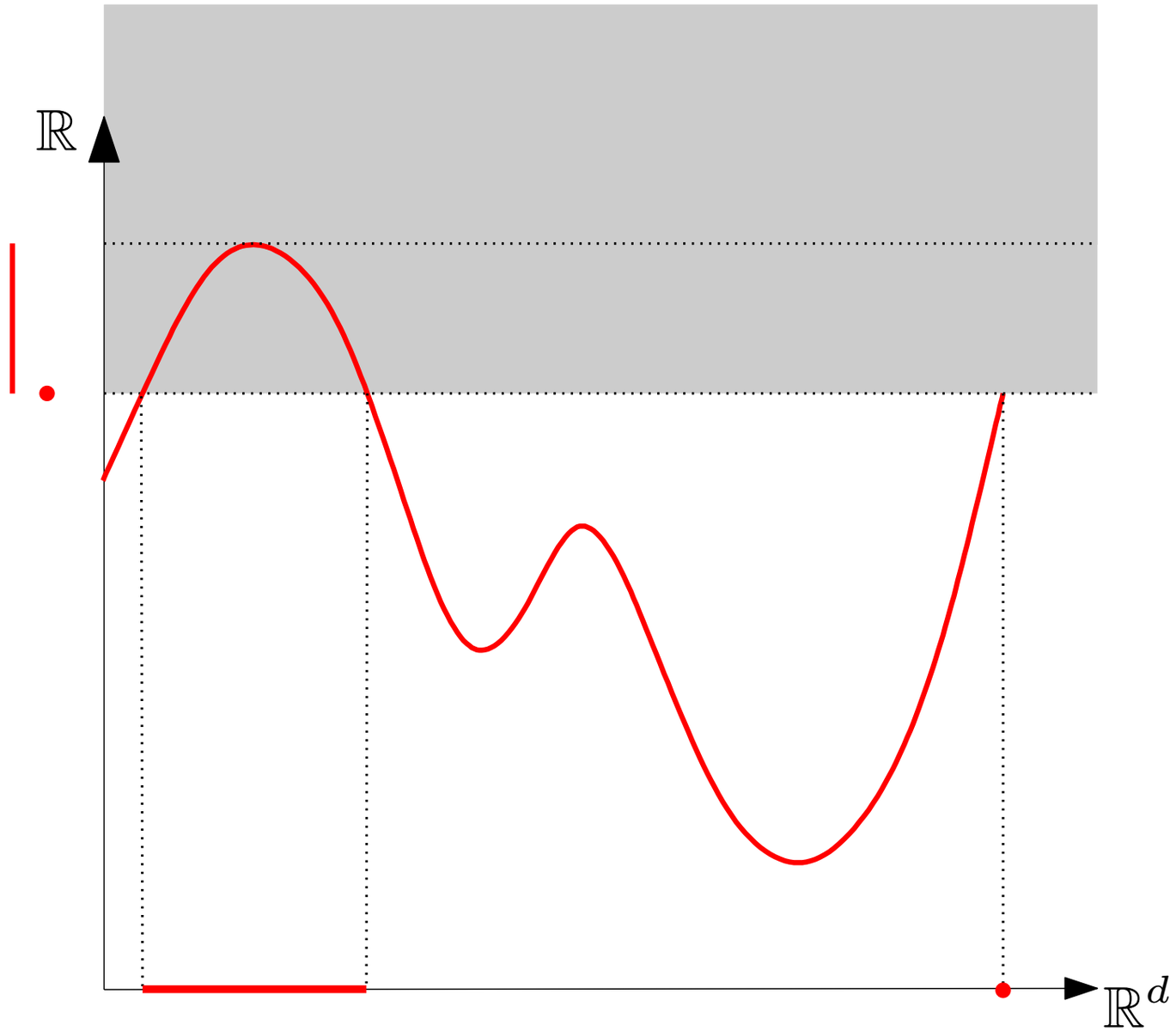
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

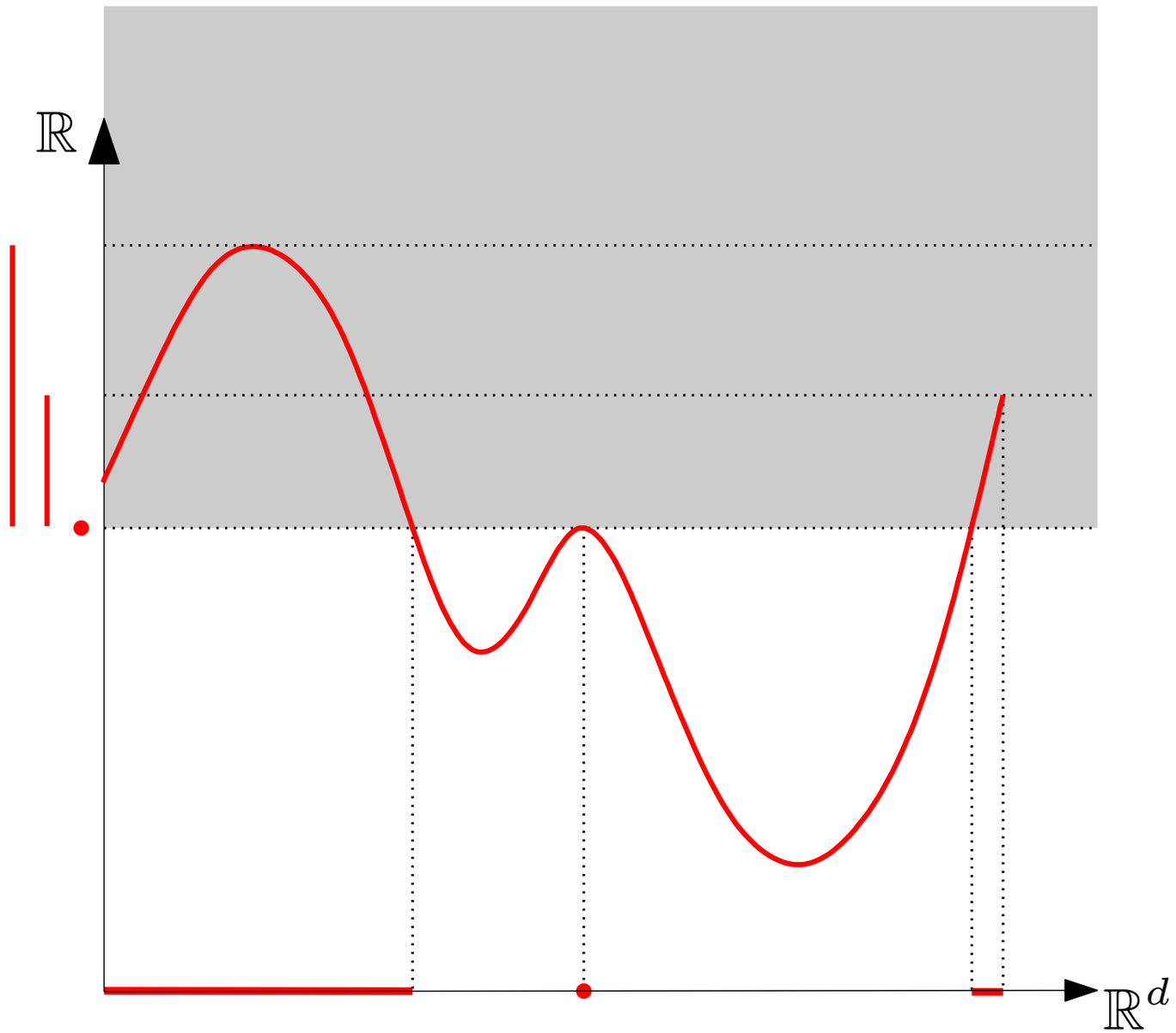
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

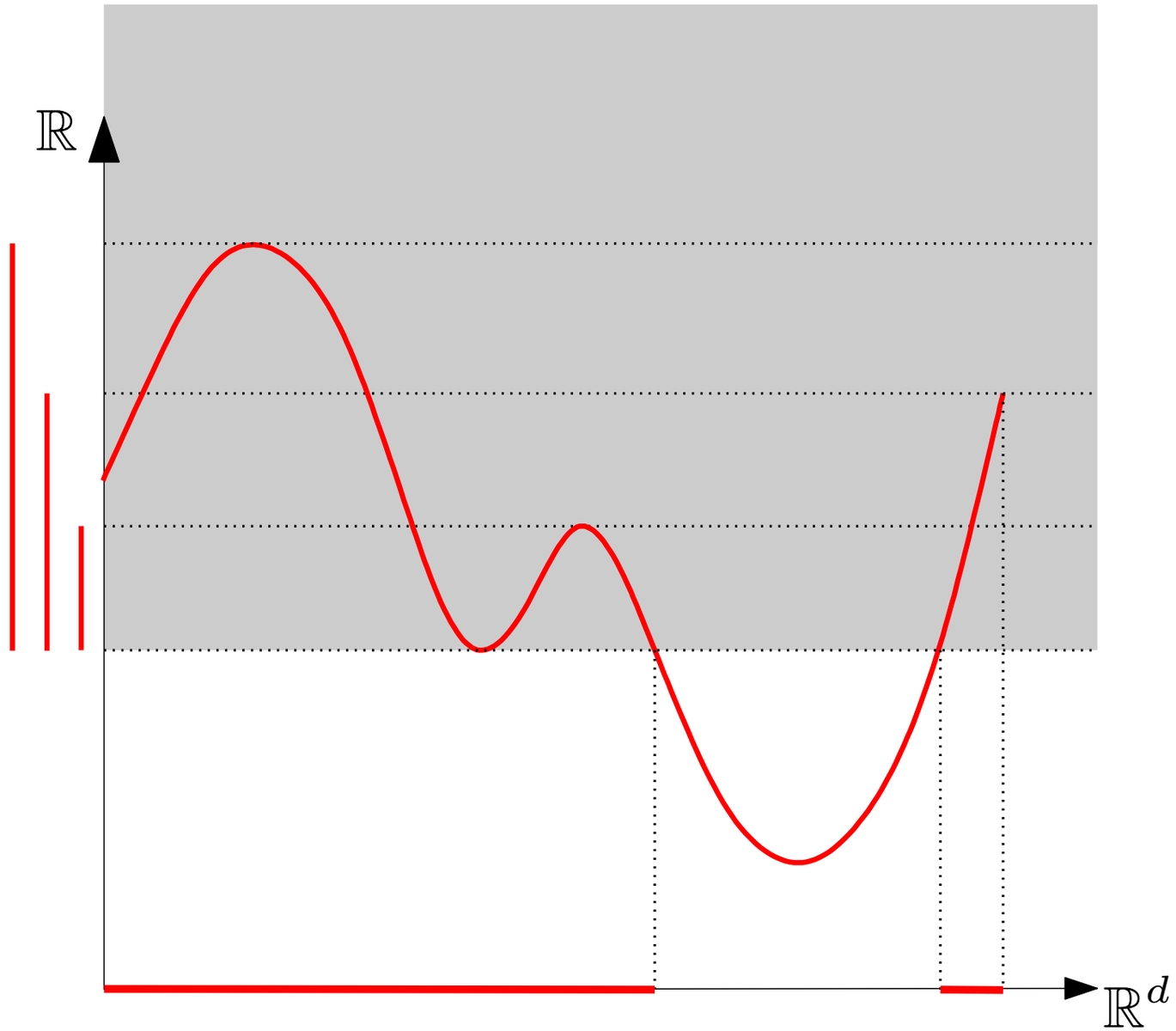
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

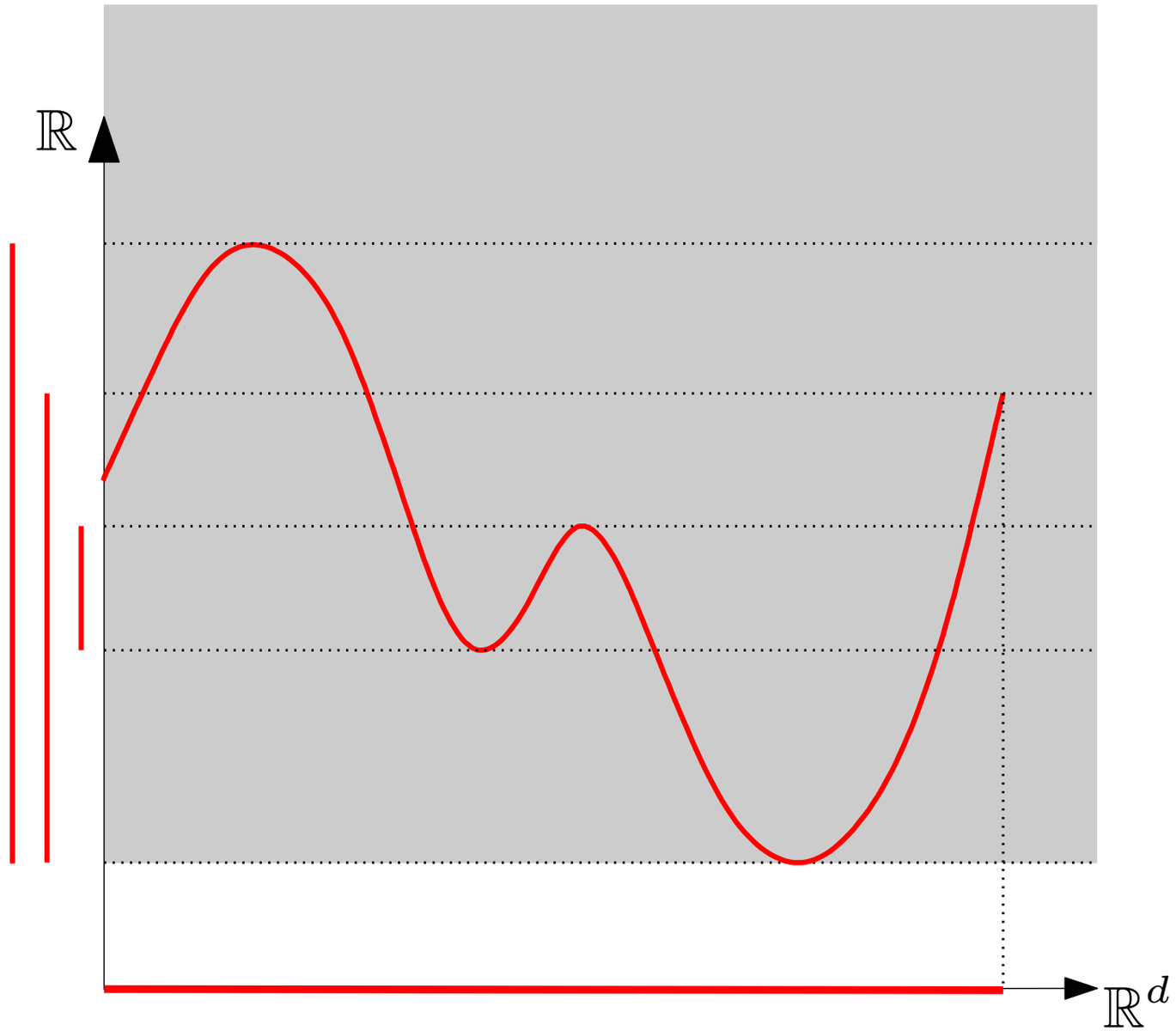
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

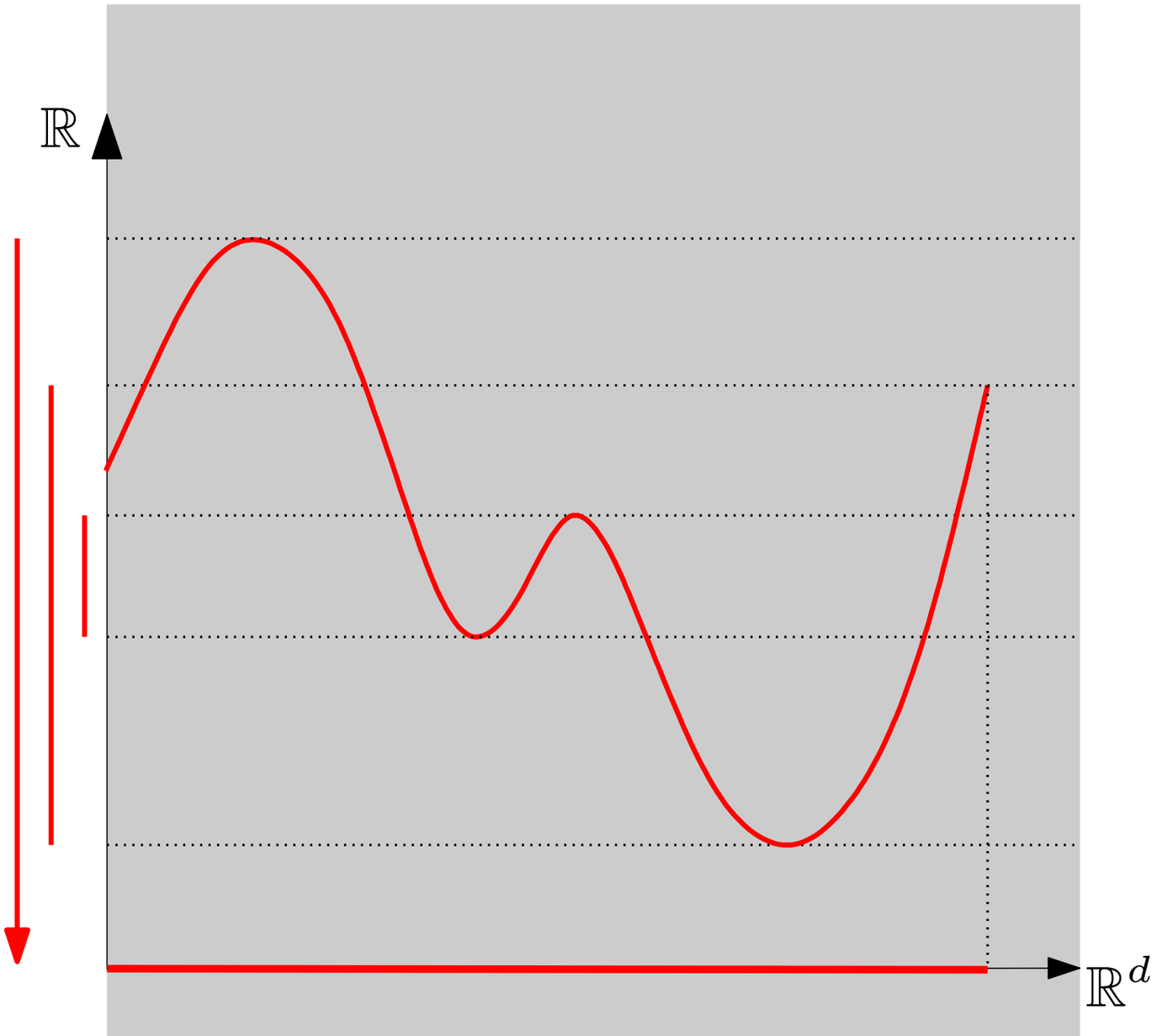
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

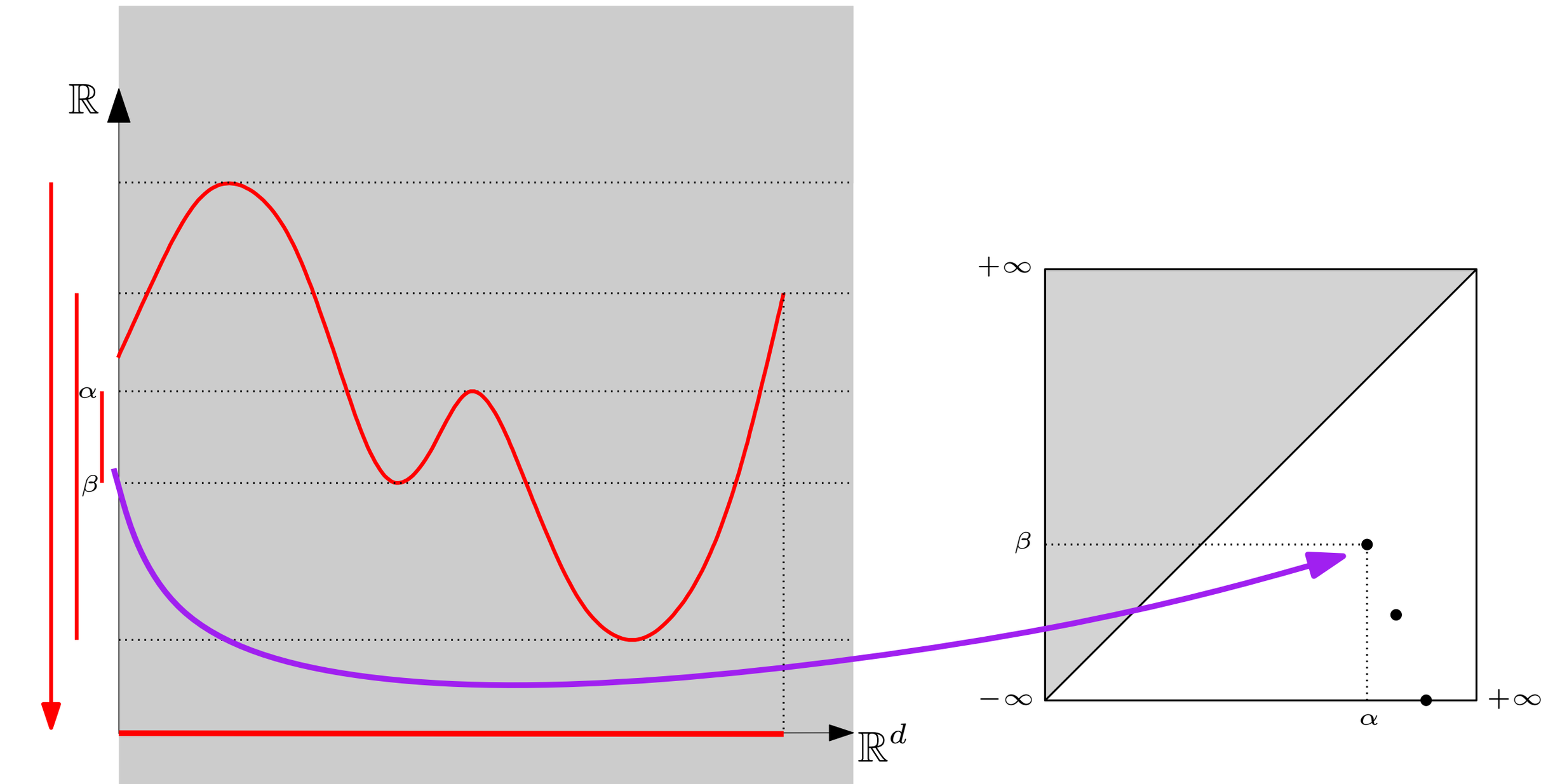
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way



0-dimensional PH of density

Given a probability density f:

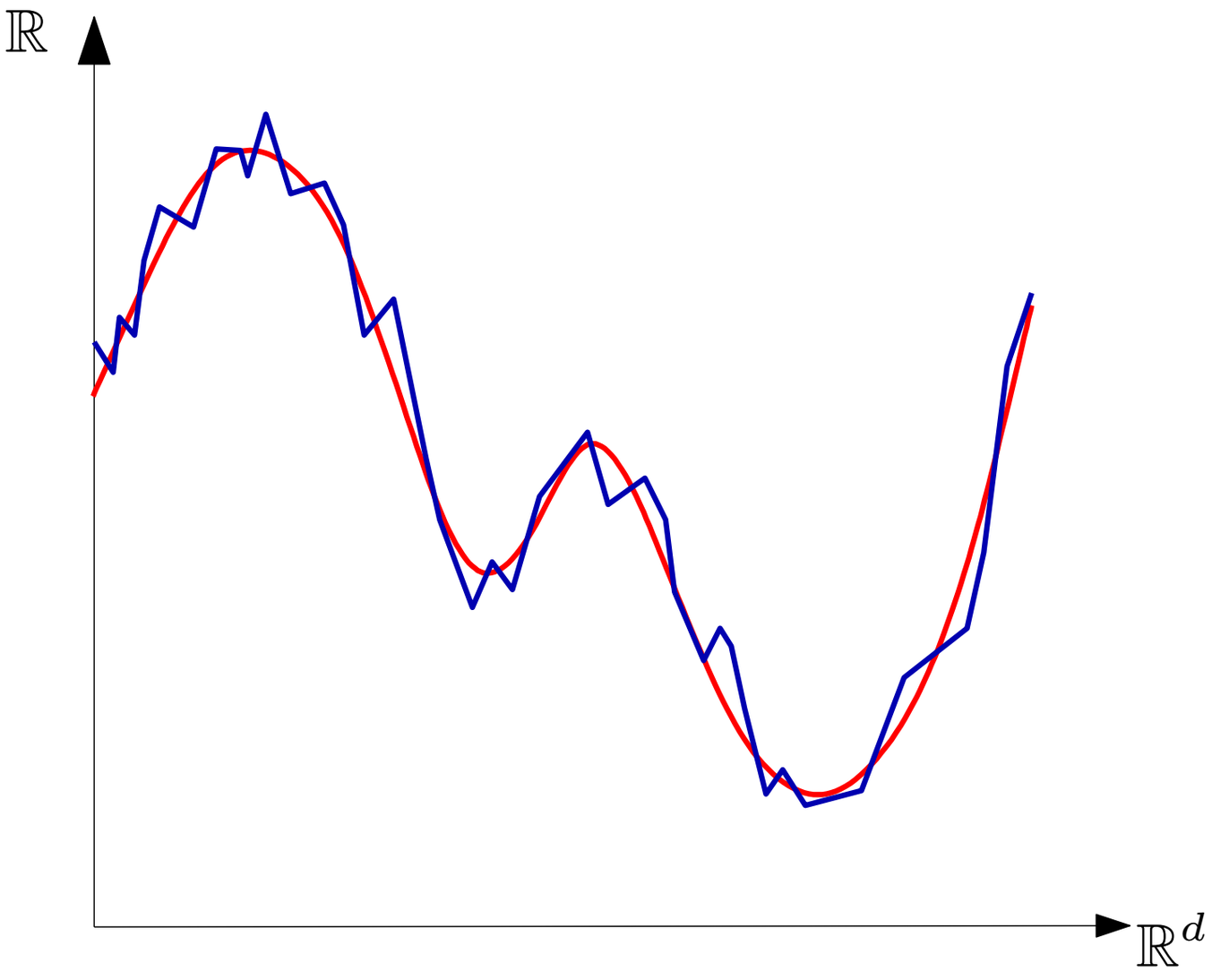
- Consider the superlevel-sets filtration $f^{-1}([t, +\infty))$ for t from $+\infty$ to $-\infty$, instead of the sublevel-sets filtration.
- Persistence is defined in the same way

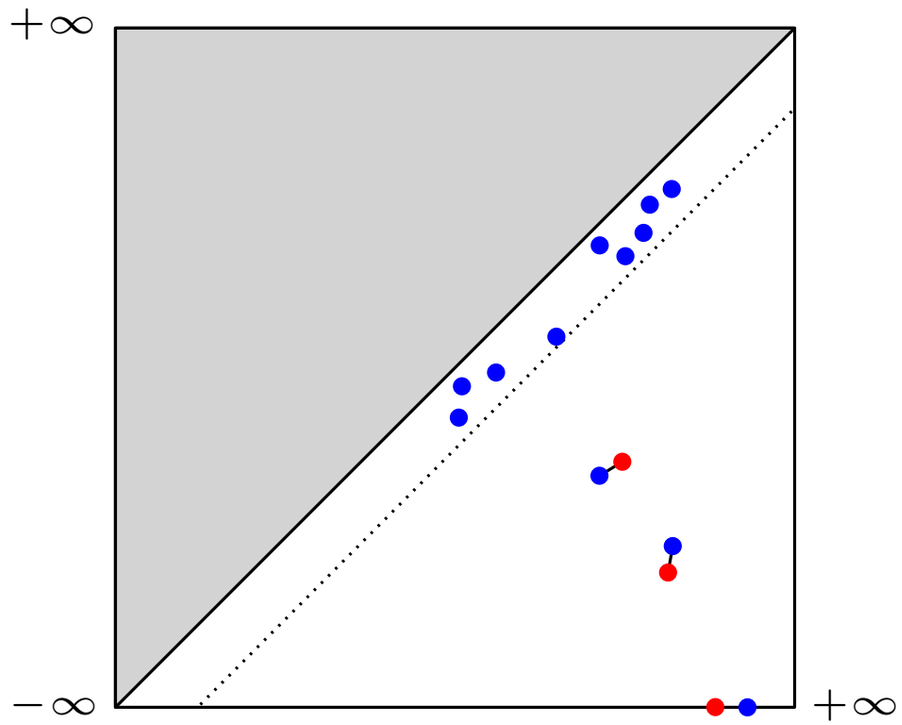


0-dimensional PH of density

Given an estimator $\hat{f}$:

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



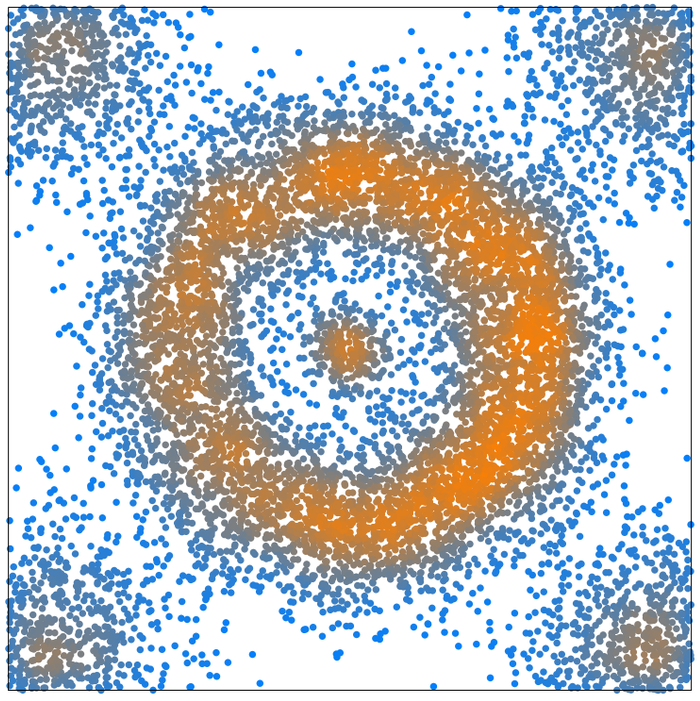


Persistence-based clustering

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

- Density estimator $\hat{f}$ defines an order on the point cloud

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



Persistence-based clustering

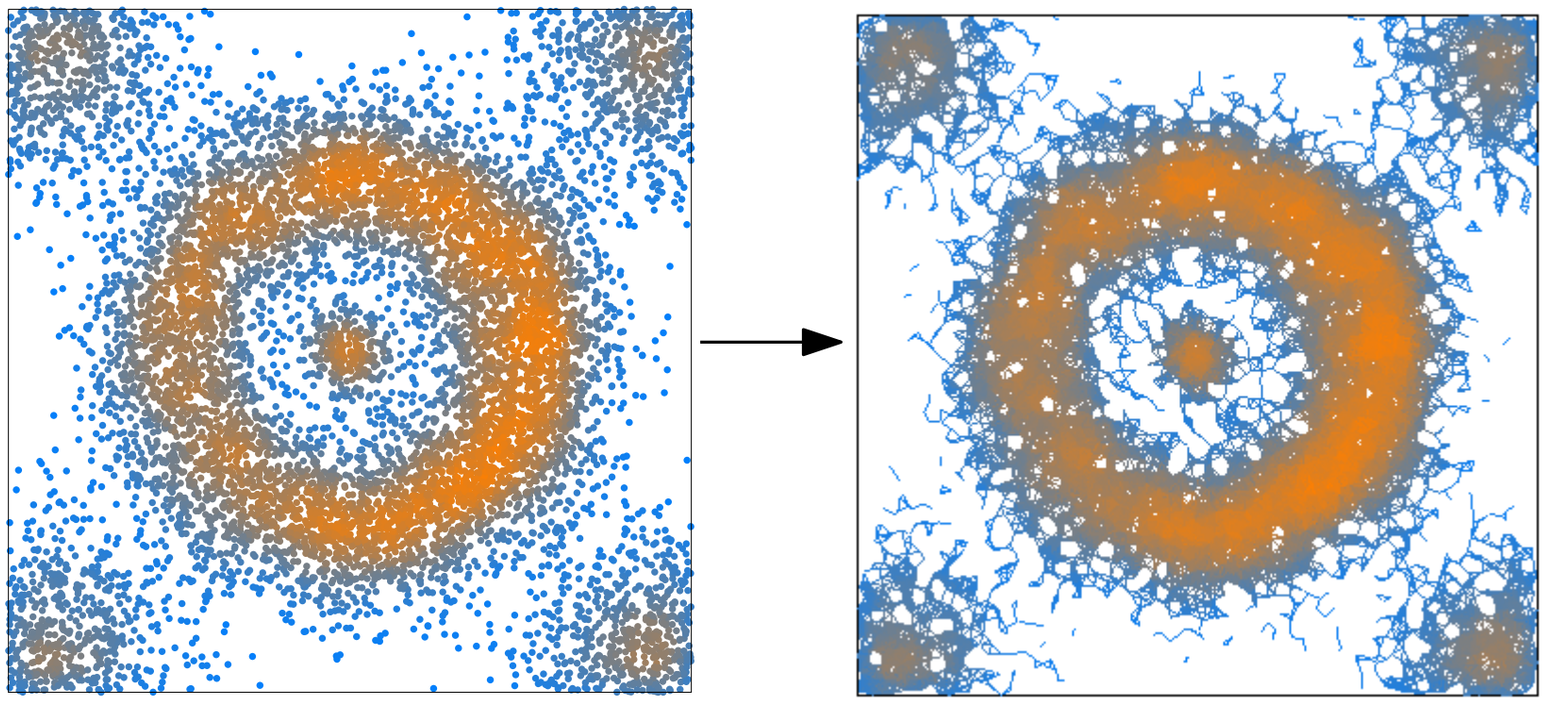
[*Persistence-Based Clustering in Riemannian Manifolds*, Chazal, Oudot, Skraba, Guibas, J. ACM, 2013]

- Density estimator $\hat{f}$ defines an order on the point cloud

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

- Extend order to the graph edges $\rightarrow$ *upper-star filtration*

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



Persistence-based clustering

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

- Density estimator $\hat{f}$ defines an order on the point cloud

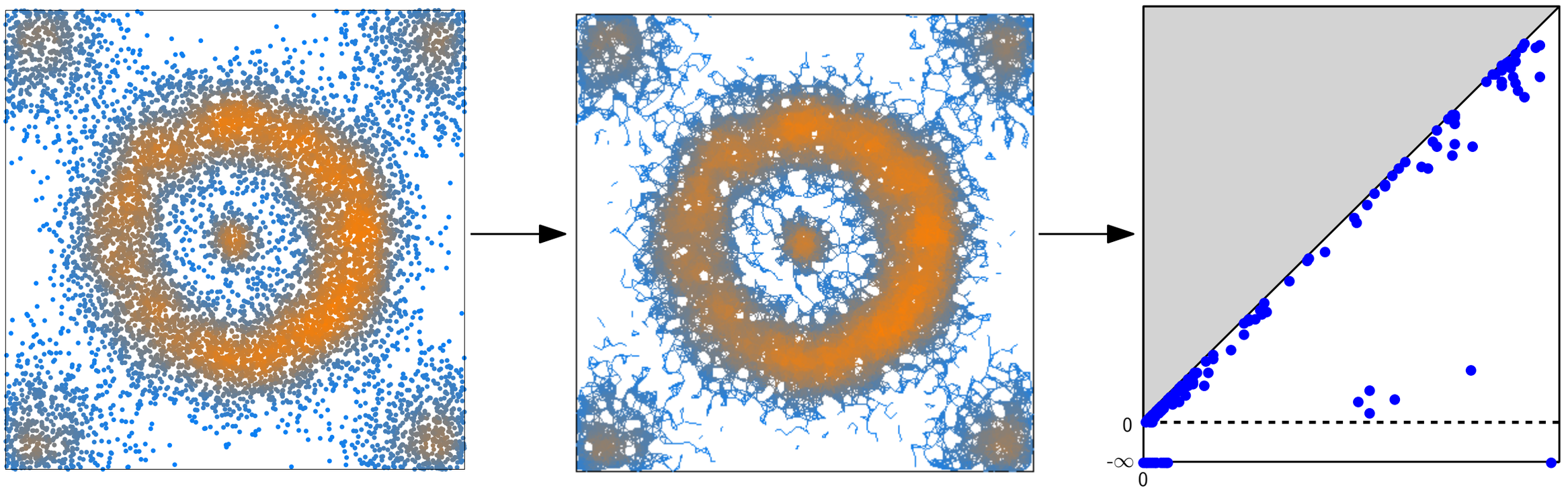
  (sort data points by **decreasing** estimated density values)

- Extend order to the graph edges $\rightarrow$ *upper-star filtration*

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

- Compute the 0-dimensional persistence diagram of this filtration

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



Estimating the correct number of clusters

- Density estimator $\hat{f}$ defines an order on the point cloud

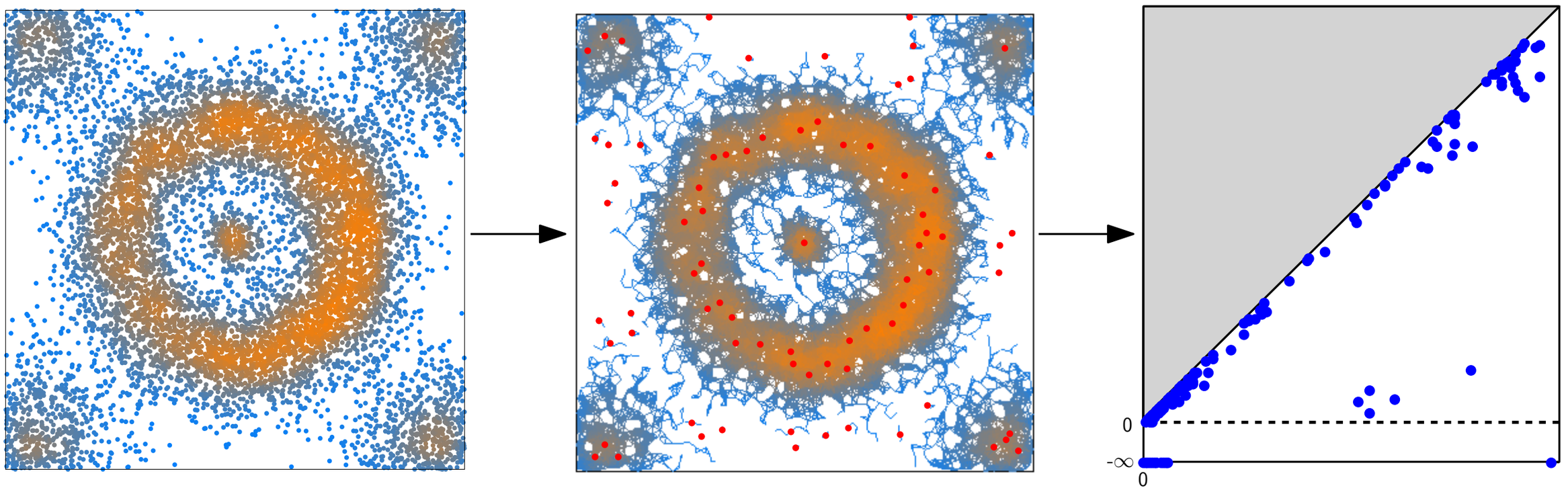
  (sort data points by **decreasing** estimated density values)

- Extend order to the graph edges $\rightarrow$ *upper-star filtration*

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

- Compute the 0-dimensional persistence diagram of this filtration

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



Estimating the correct number of clusters

- Density estimator $\hat{f}$ defines an order on the point cloud

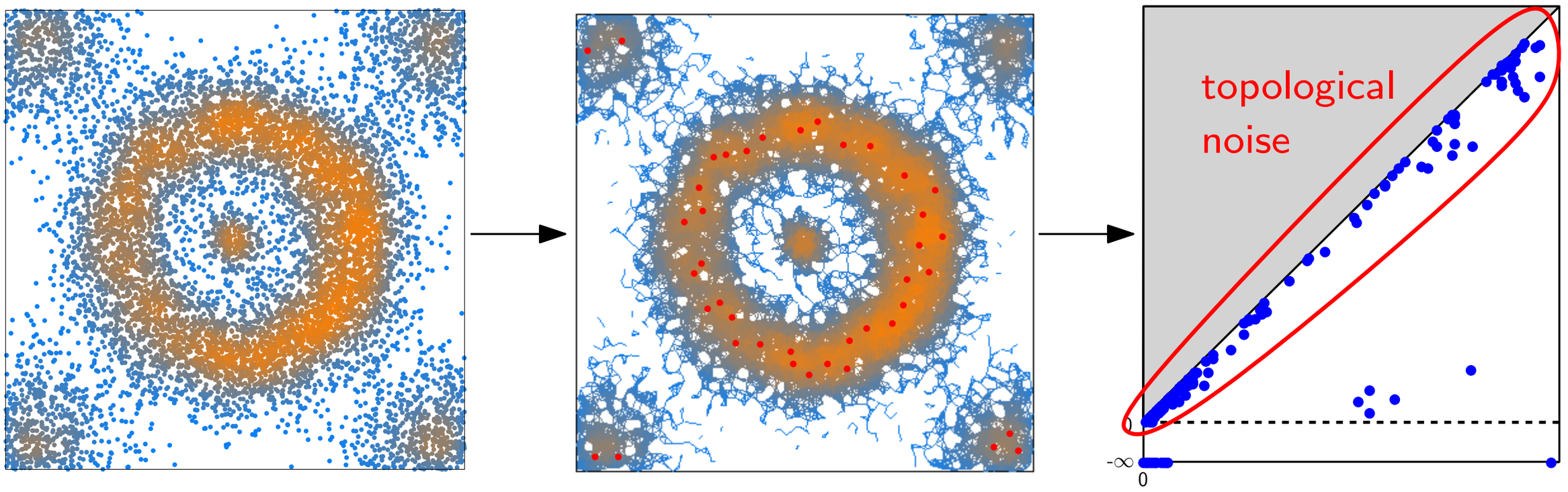
  (sort data points by **decreasing** estimated density values)

- Extend order to the graph edges $\rightarrow$ *upper-star filtration*

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

- Compute the 0-dimensional persistence diagram of this filtration

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



Estimating the correct number of clusters

- Density estimator $\hat{f}$ defines an order on the point cloud

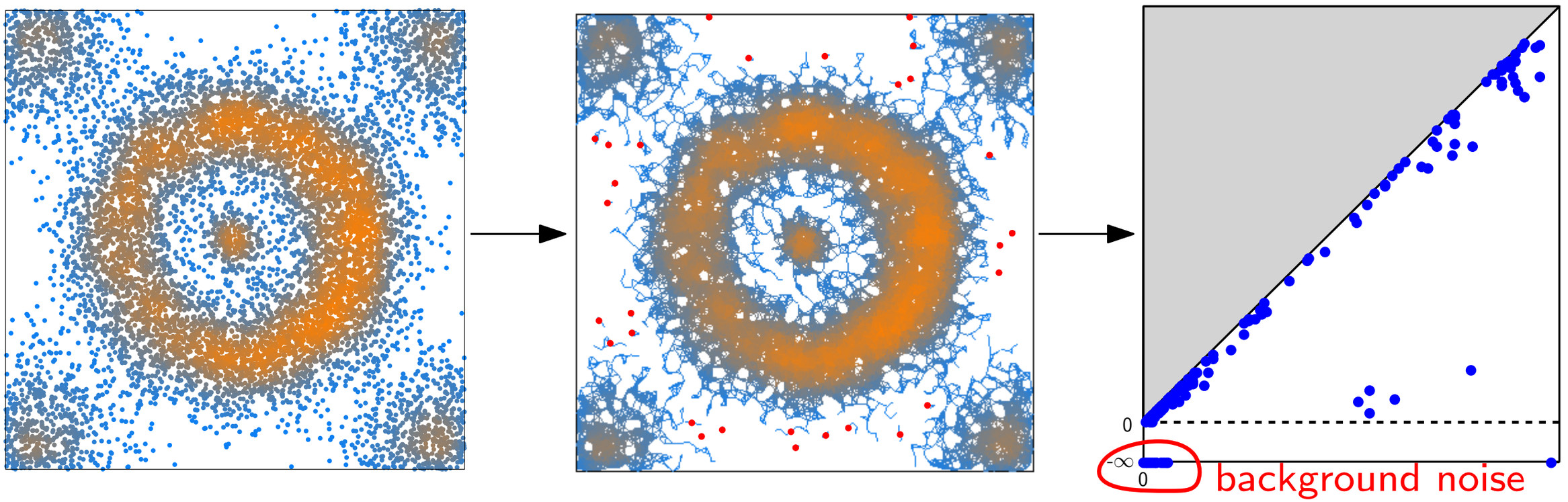
  (sort data points by **decreasing** estimated density values)

- Extend order to the graph edges $\rightarrow$ *upper-star filtration*

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

- Compute the 0-dimensional persistence diagram of this filtration

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



Estimating the correct number of clusters

- Density estimator $\hat{f}$ defines an order on the point cloud

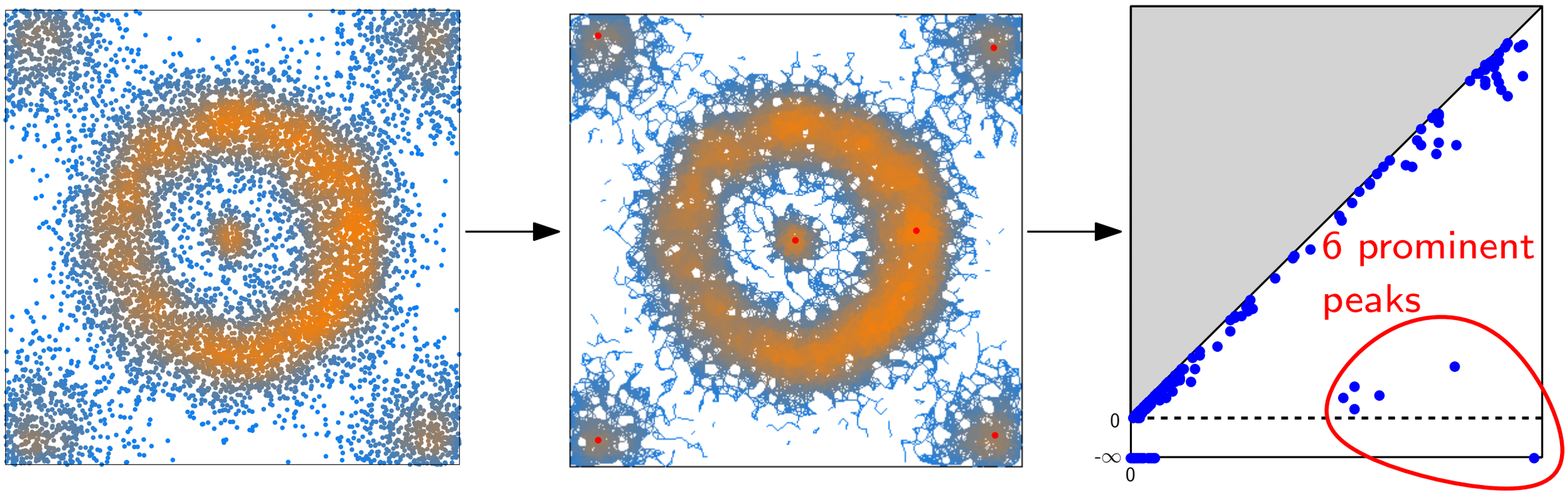
  (sort data points by **decreasing** estimated density values)

- Extend order to the graph edges $\rightarrow$ *upper-star filtration*

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

- Compute the 0-dimensional persistence diagram of this filtration

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



Estimating the correct number of clusters

- Density estimator $\hat{f}$ defines an order on the point cloud

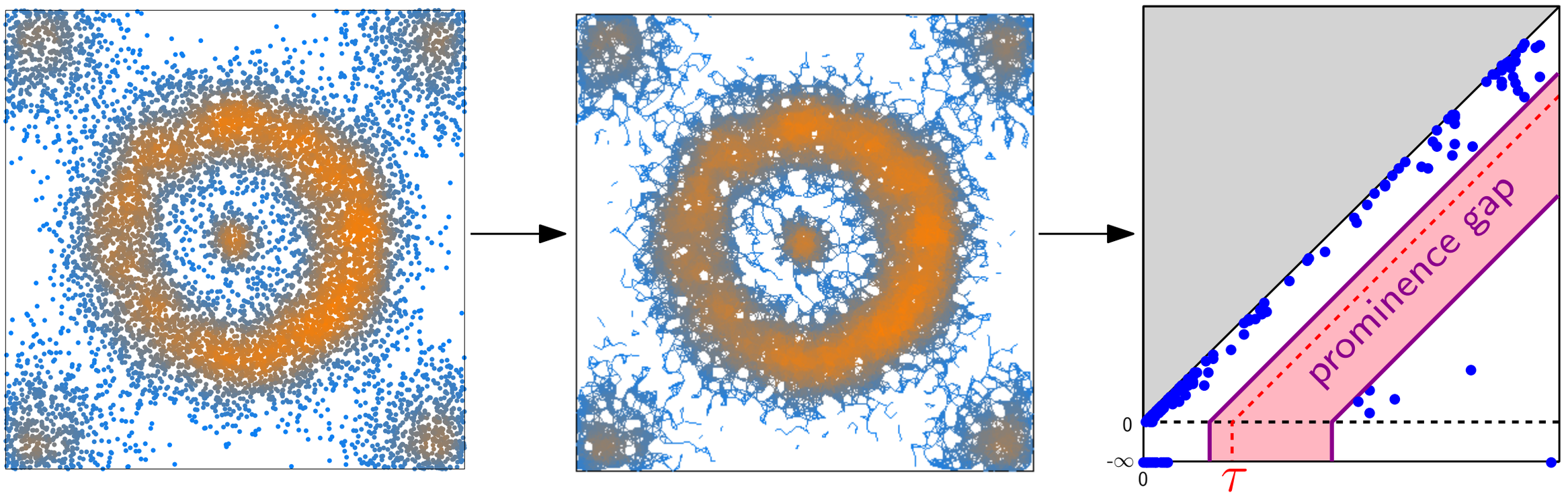
  (sort data points by **decreasing** estimated density values)

- Extend order to the graph edges $\rightarrow$ *upper-star filtration*

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

- Compute the 0-dimensional persistence diagram of this filtration

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



Estimating the correct number of clusters

Hypotheses:

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

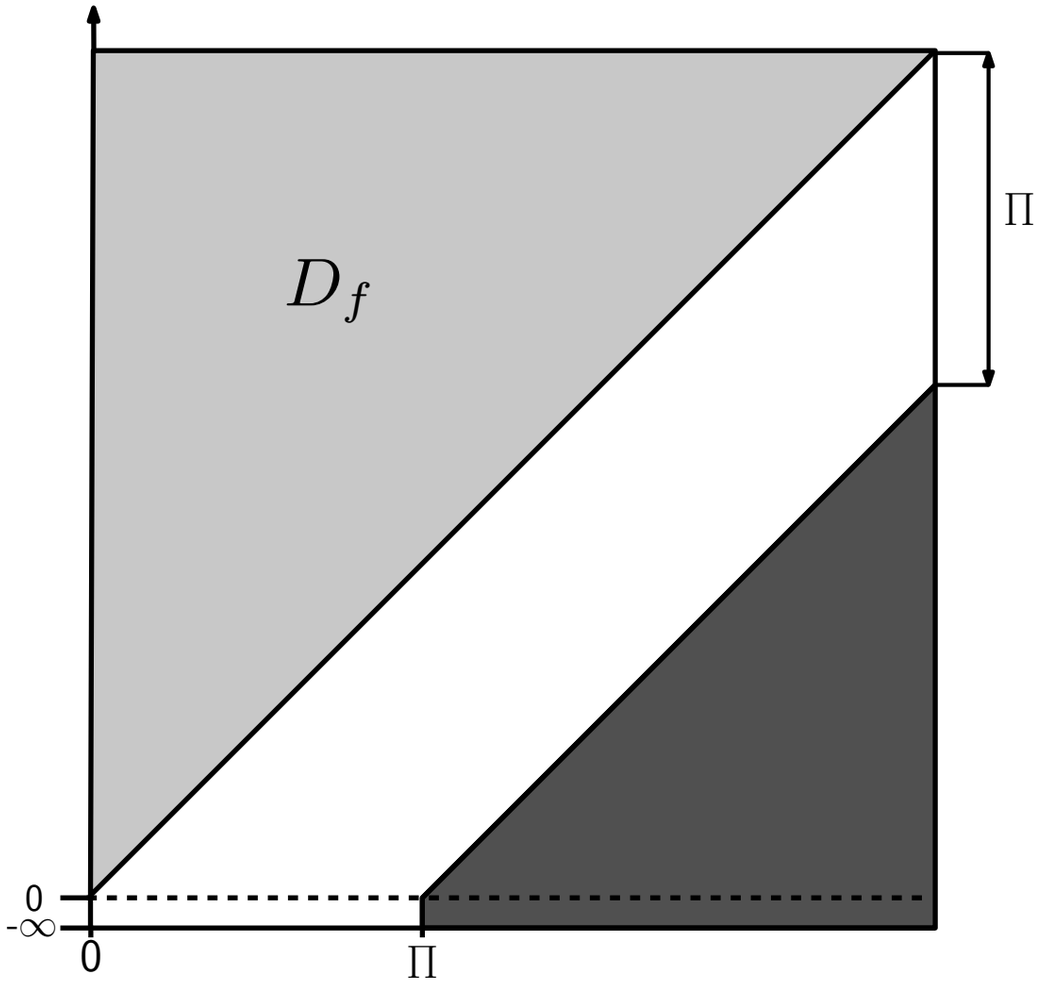
Note: Π is the prominence of the least prominent peak of f

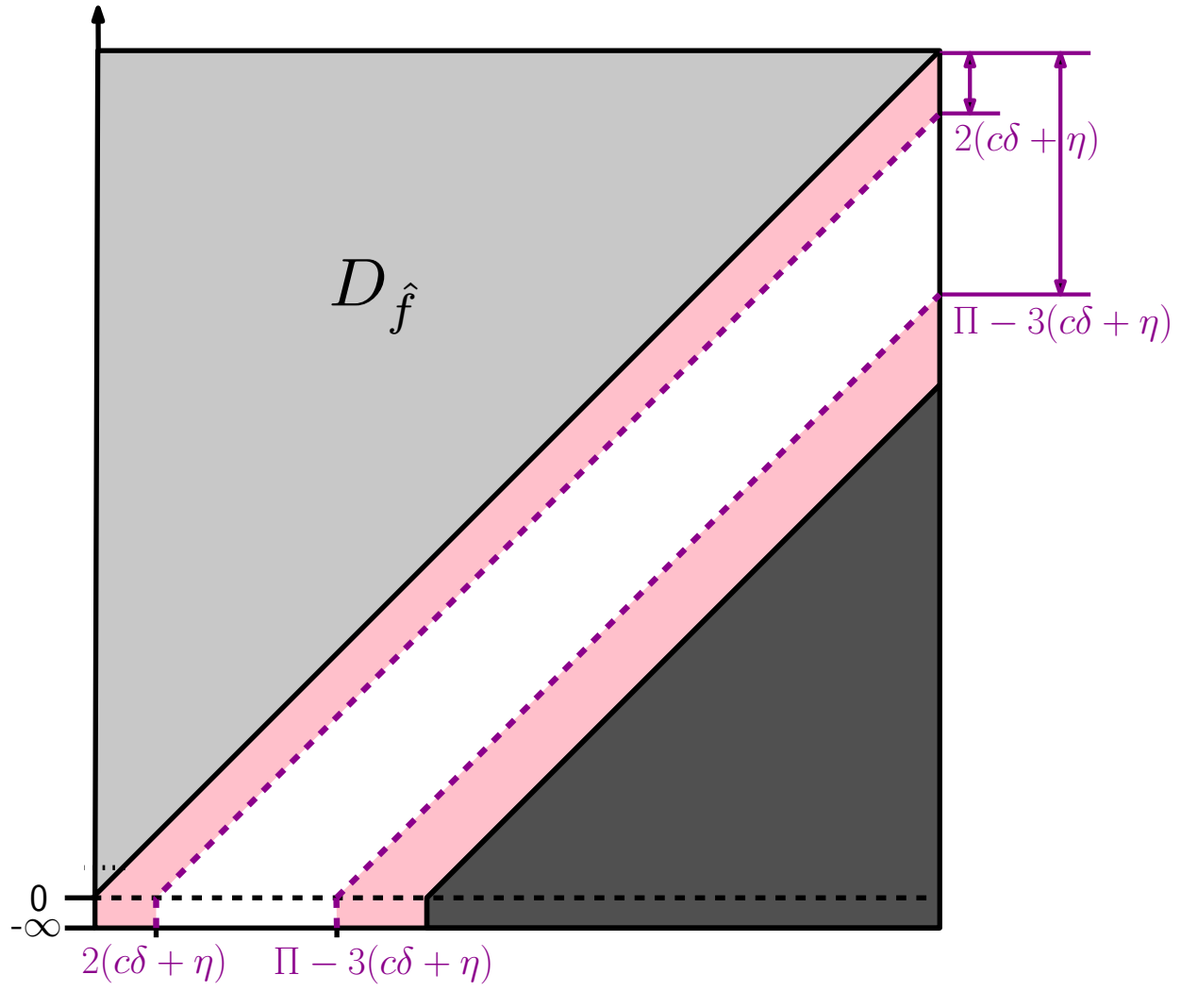
Conclusion:

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)

Estimating the correct number of clusters





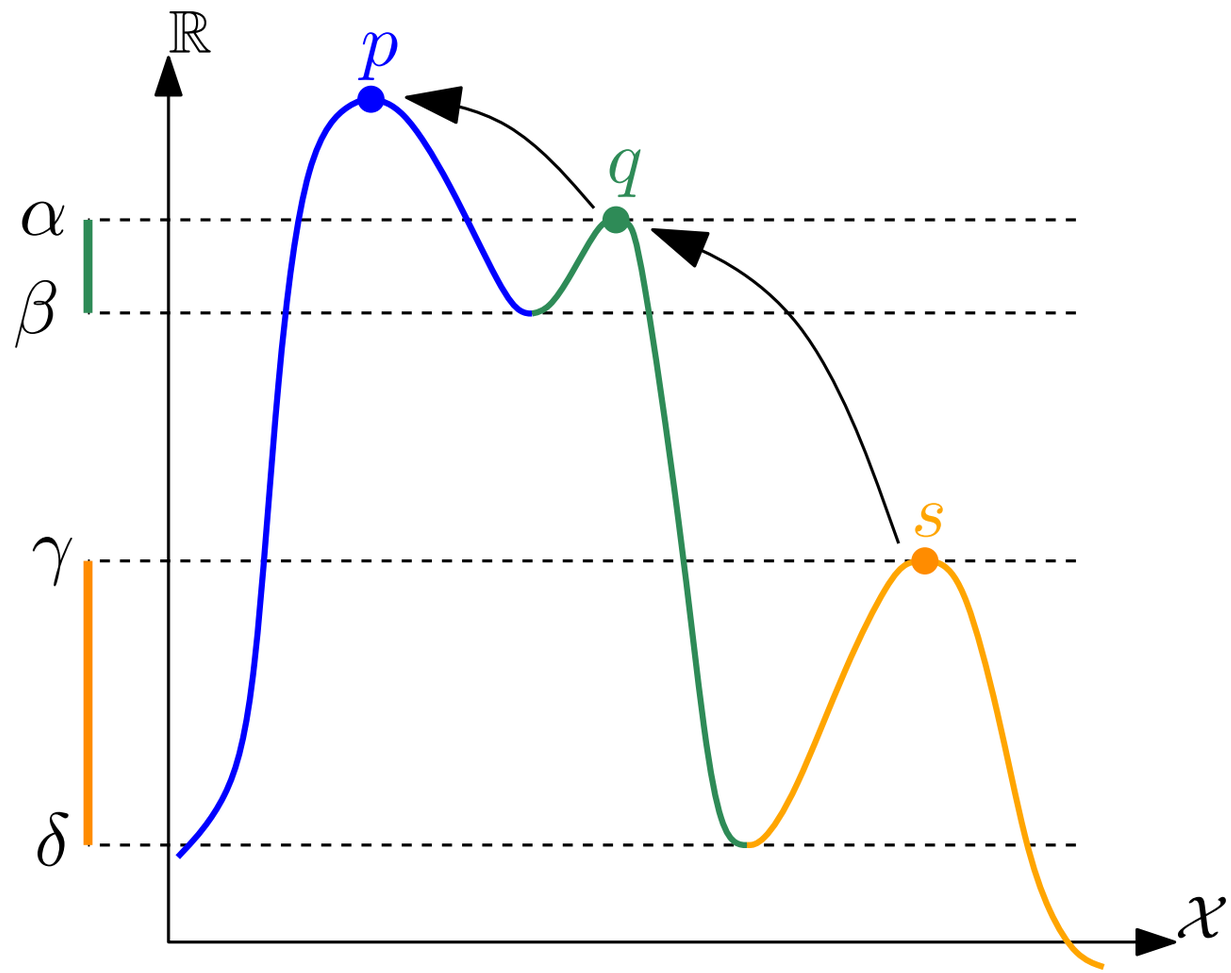
Conclusion:

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)

Merging clusters

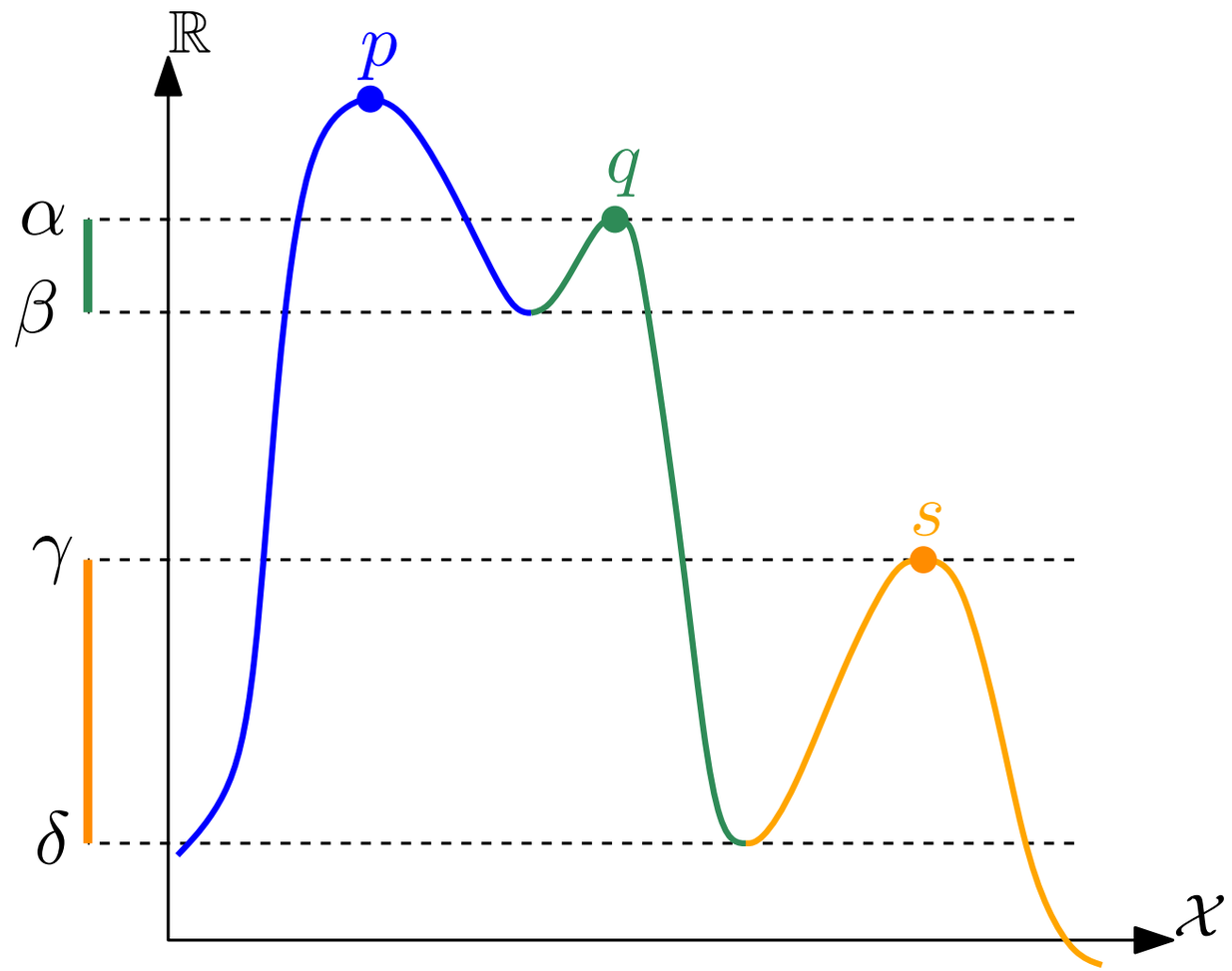
- degree-0 persistence algo. builds a hierarchy of the peaks of $\hat{f}$ (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)



Merging clusters

- degree-0 persistence algo. builds a hierarchy of the peaks of $\hat{f}$ (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)
- given a fixed threshold $\tau \geq 0$, only merge those clusters of prominence $< \tau$

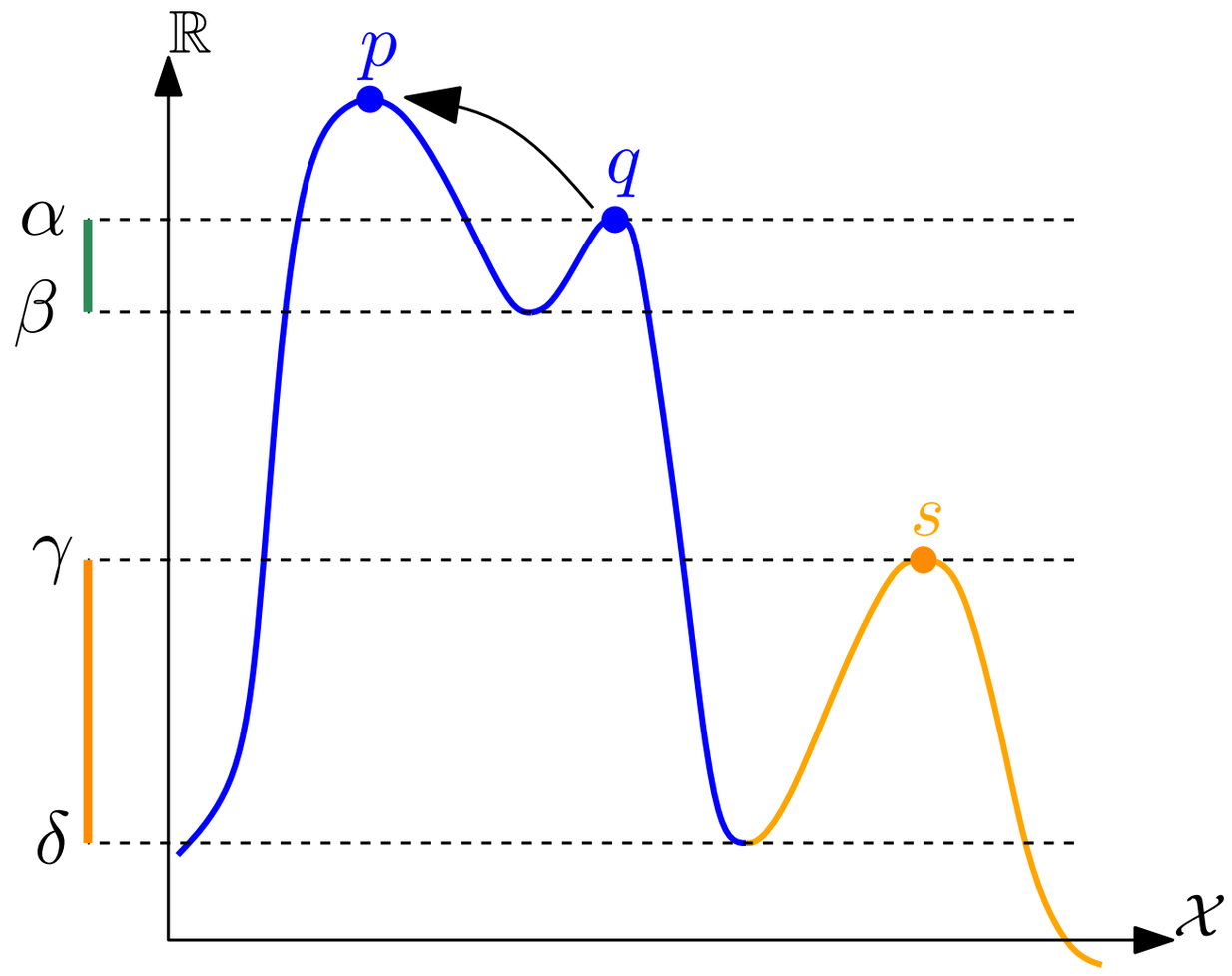
$$0 \leq \tau \leq \alpha - \beta$$



Merging clusters

- degree-0 persistence algo. builds a hierarchy of the peaks of $\hat{f}$ (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)
- given a fixed threshold $\tau \geq 0$, only merge those clusters of prominence $< \tau$

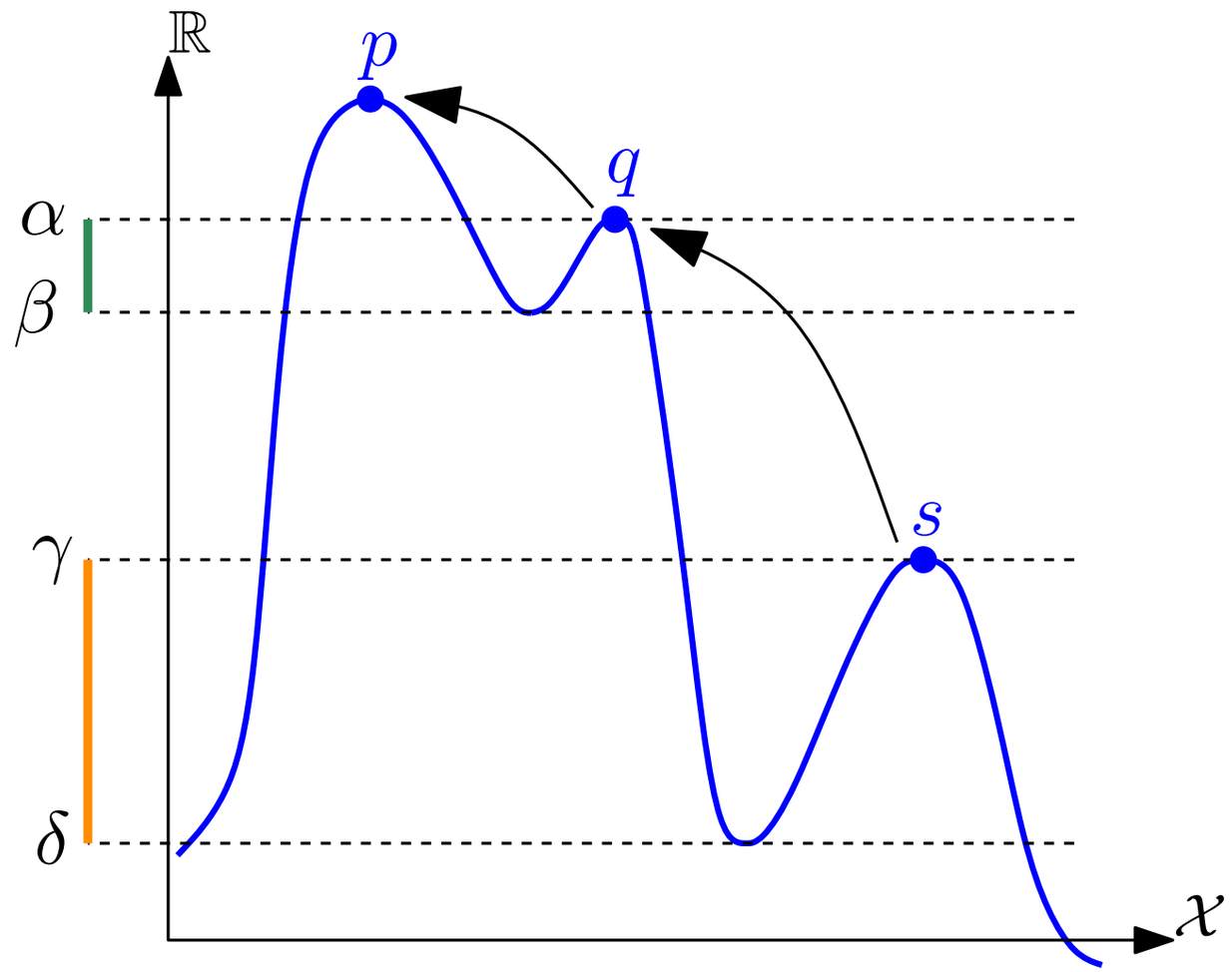
$$\alpha - \beta < \tau \leq \gamma - \delta$$



Merging clusters

- degree-0 persistence algo. builds a hierarchy of the peaks of $\hat{f}$ (merge tree)
- merge clusters according to the hierarchy (merge each cluster into its parent)
- given a fixed threshold $\tau \geq 0$, only merge those clusters of prominence $< \tau$

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



Pseudo-code

Input: simple graph G with n vertices, n-dimensional vector $\hat{f}$, real parameter $\tau \geq 0$.

Sort the vertex indices $\{1, 2, \ldots, n\}$ so that $\hat{f}(1) \geq \cdots \geq \hat{f}(n)$;
Initialize a union-find data structure $\mathcal{U}$ and two vectors g, r of size n;

for $i = 1$ to n **do**
    Let $\mathcal{N}$ be the set of neighbors of i in G that have indices lower than i;
    if $\mathcal{N} = \emptyset$ *// vertex i is a peak of $\hat{f}$ within G*
        Create a new entry e in $\mathcal{U}$ and attach vertex i to it: $\mathcal{U}.\texttt{MakeSet}(i)$;
        $r(e) \leftarrow i$ *// $r(e)$ stores the root vertex associated with the entry e*
    else *// vertex i is not a peak of $\hat{f}$ within G*
        $g(i) \leftarrow \operatorname{argmax}_{j \in \mathcal{N}} \hat{f}(j)$ *// $g(i)$ stores the approximate gradient at vertex i*
        $e_i \leftarrow \mathcal{U}.\texttt{Find}(g(i))$;
        Attach vertex i to the entry e_i: $\mathcal{U}.\texttt{Union}(i, e_i)$;
    for $j \in \mathcal{N}$ **do**
        $e \leftarrow \mathcal{U}.\texttt{Find}(j)$;
        if $e \neq e_i$ and $\min\{\hat{f}(r(e)),\ \hat{f}(r(e_i))\} < \hat{f}(i) + \tau$
            $\mathcal{U}.\texttt{Union}(e,\ e_i)$;
            $r(e \cup e_i) \leftarrow \operatorname{argmax}_{\{r(e),\ r(e_i)\}} \hat{f}$;
            $e_i \leftarrow e \cup e_i$;

graph-based hill-climbing (1976)

cluster merges with persistence (2013)

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

Complexity

Given a neighborhood graph with n vertices (with density values) and m edges:

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

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

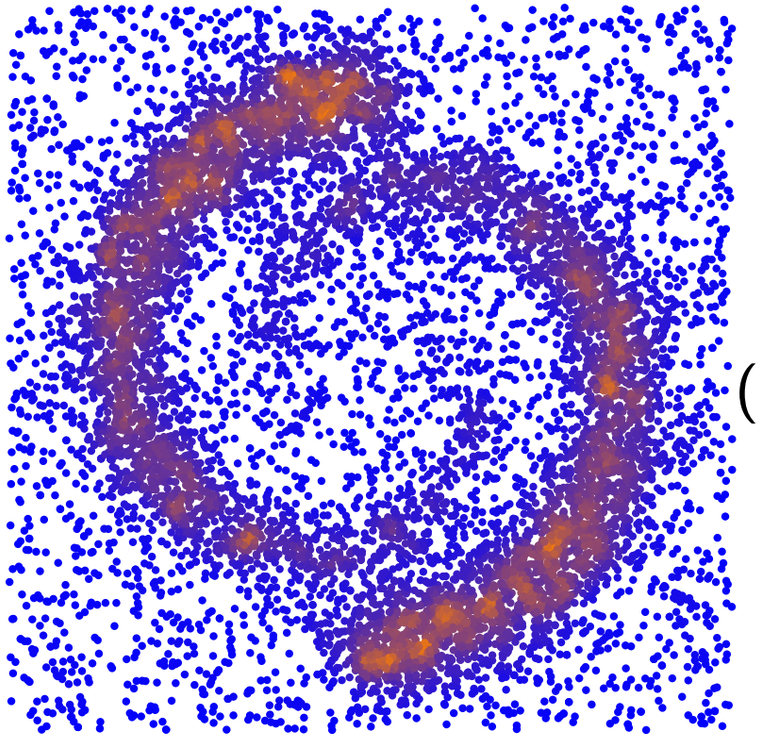
$\rightarrow$ Running time: $O(n \log n + (n + m)\alpha(n))$

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

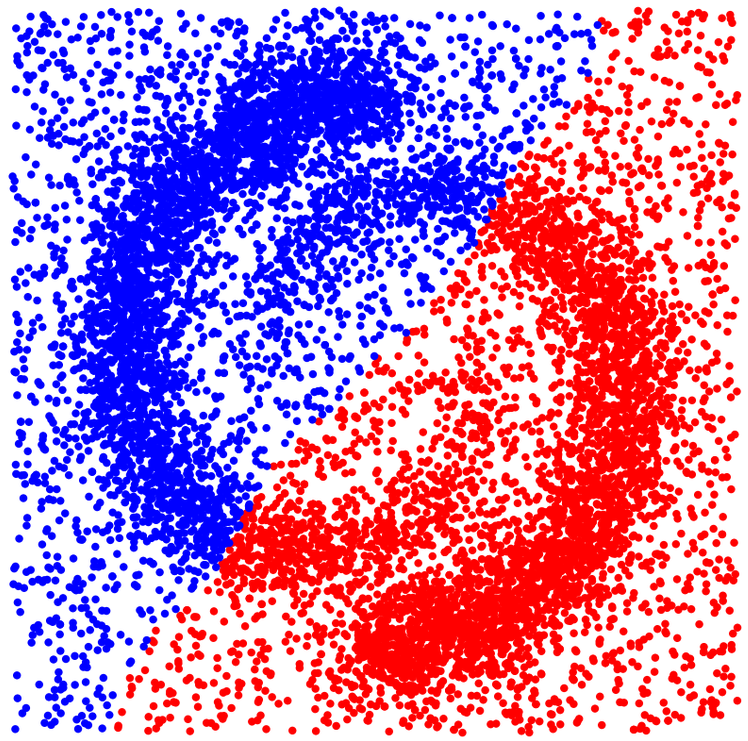
$\rightarrow$ Main memory usage: $O(n)$

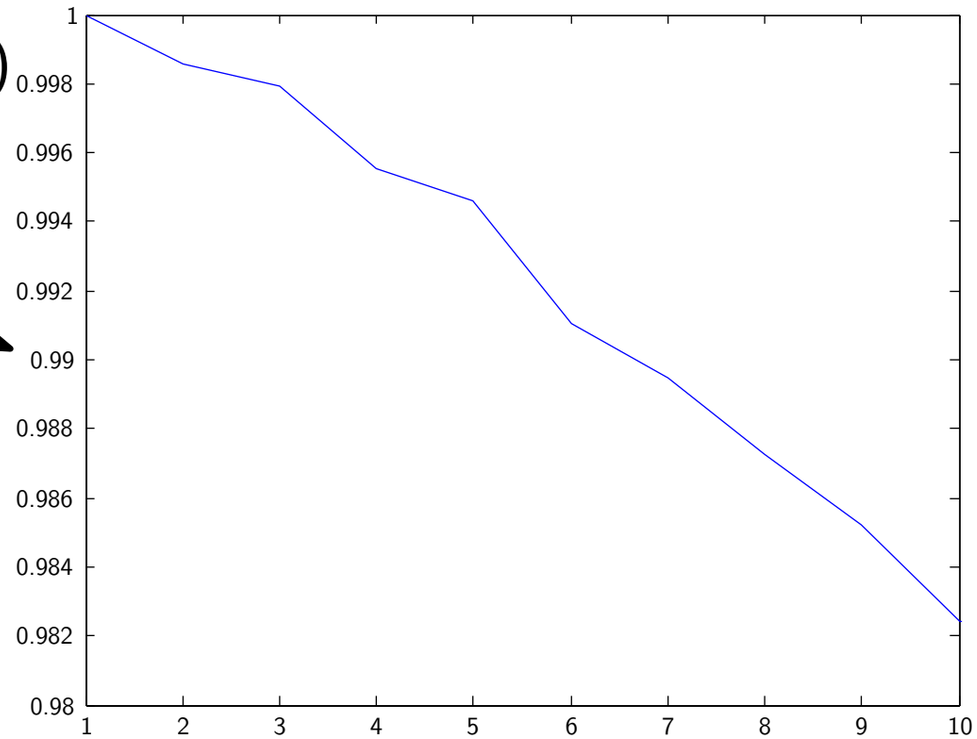
Experimental results

Synthetic Data



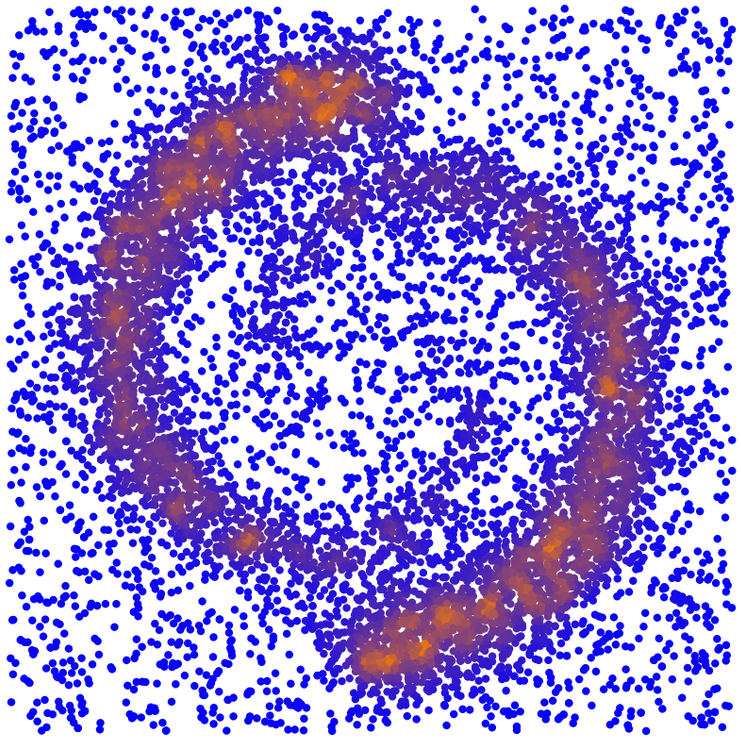
Spectral clustering
(k-means in eigenspace)





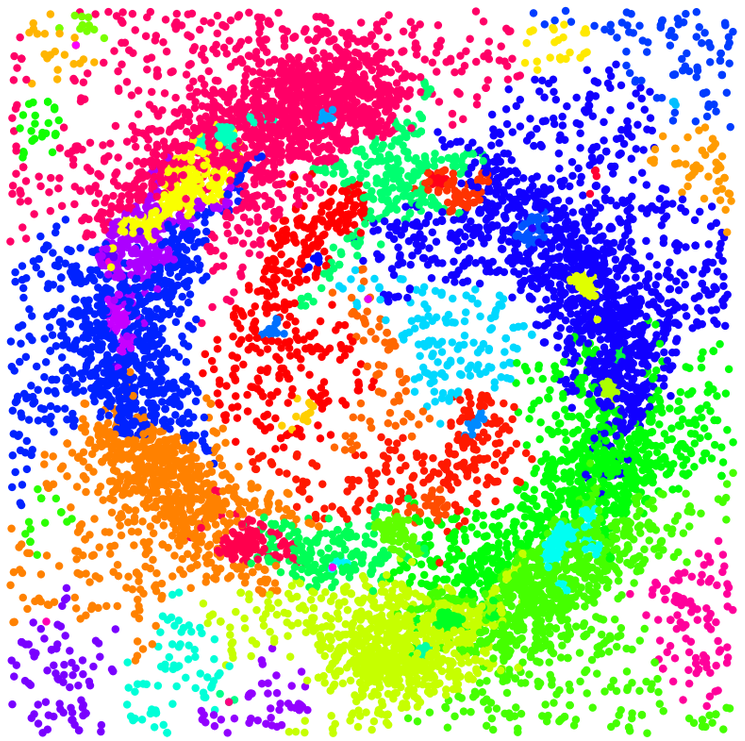
Experimental results

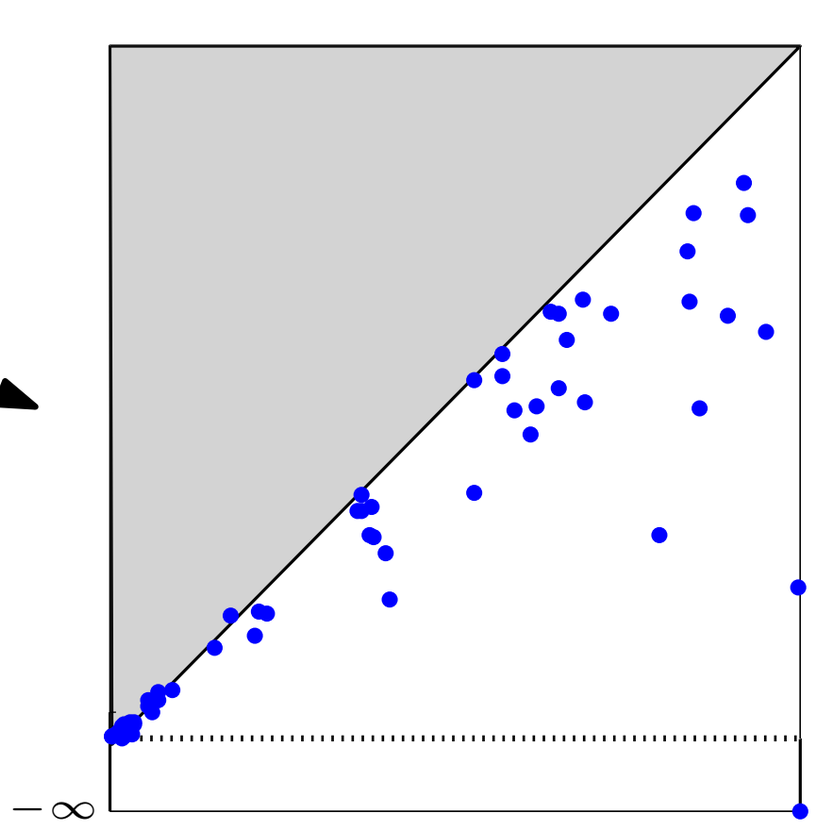
Synthetic Data



$\tau = 0$

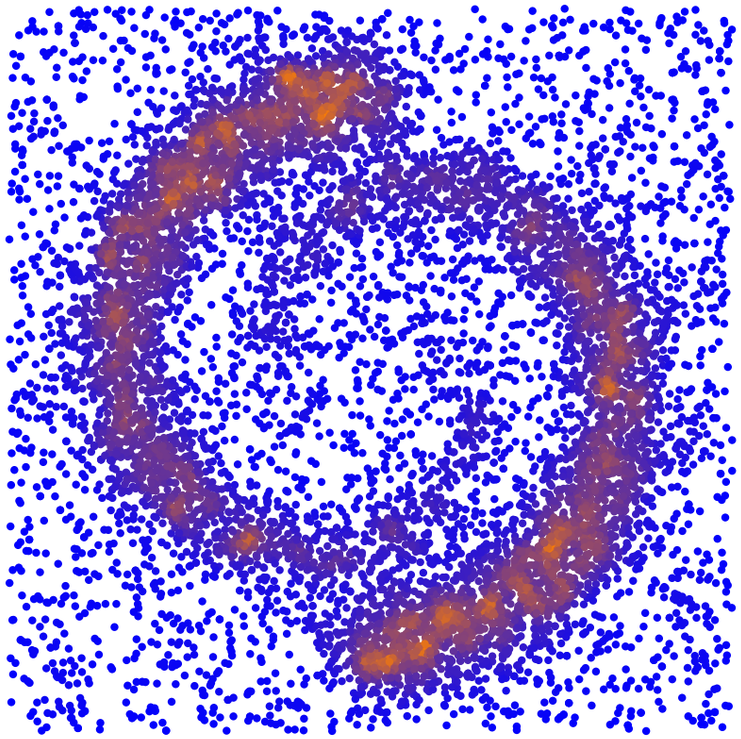
ToMATo



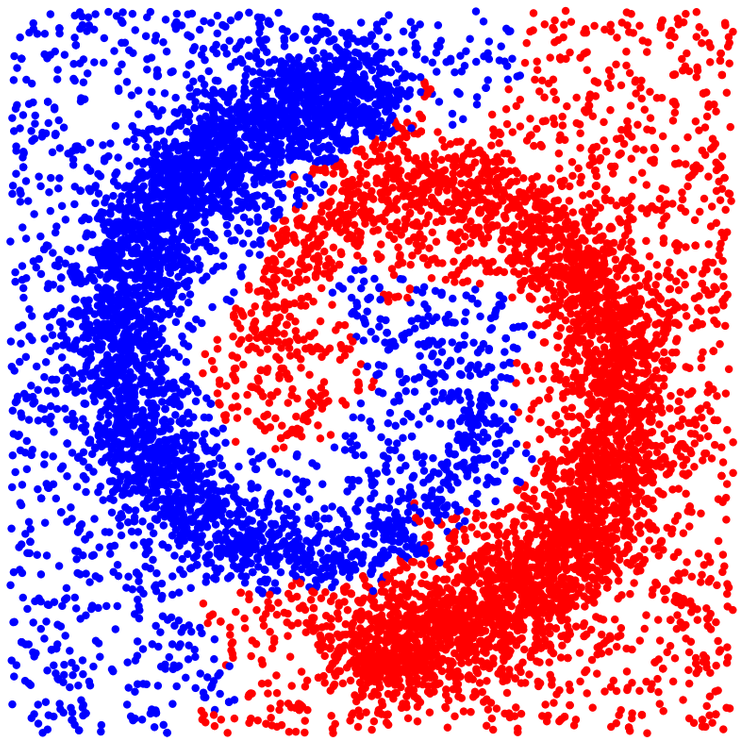


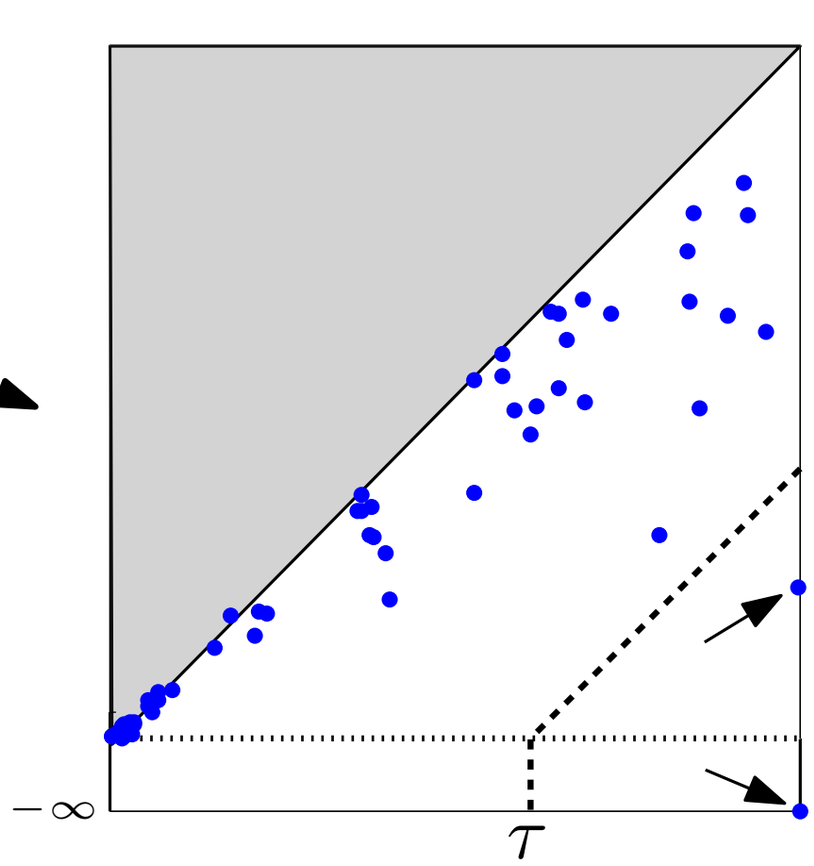
Experimental results

Synthetic Data



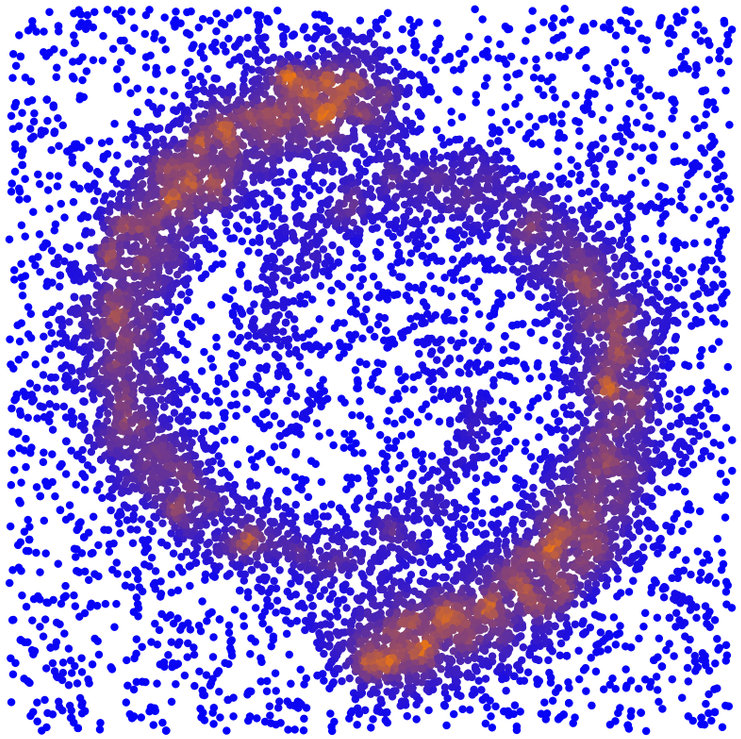
ToMATo

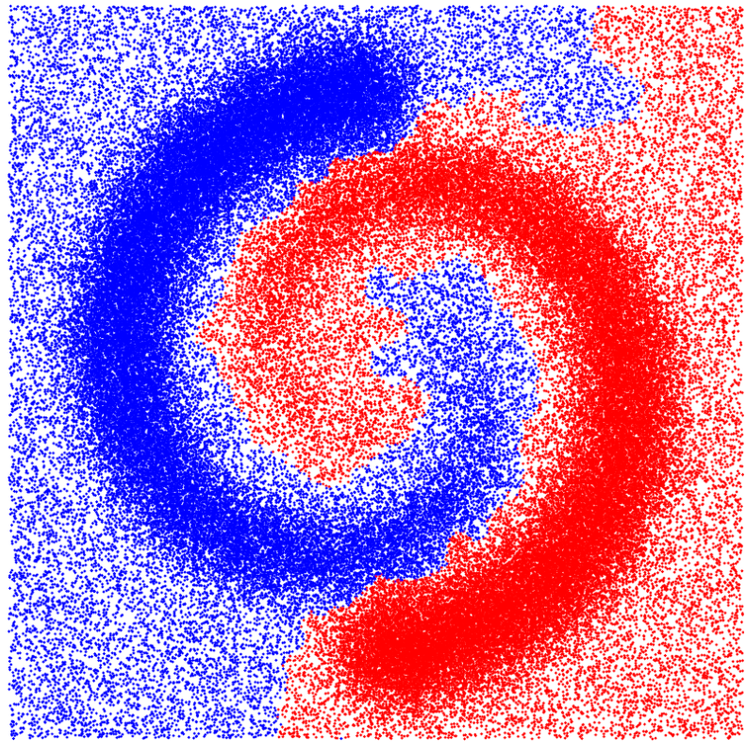


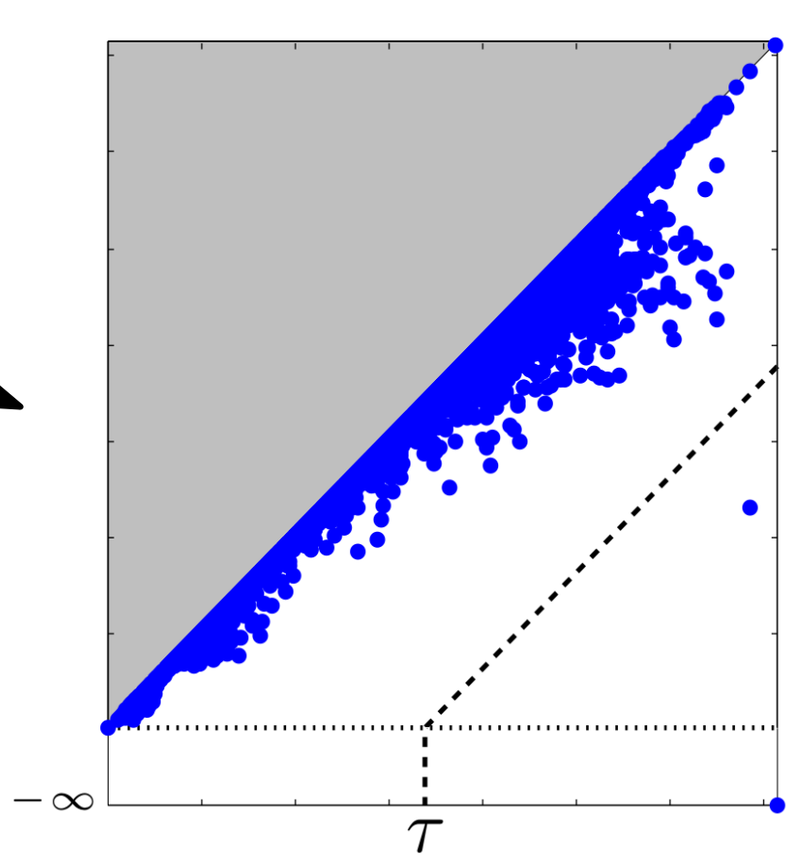


Experimental results

Synthetic Data





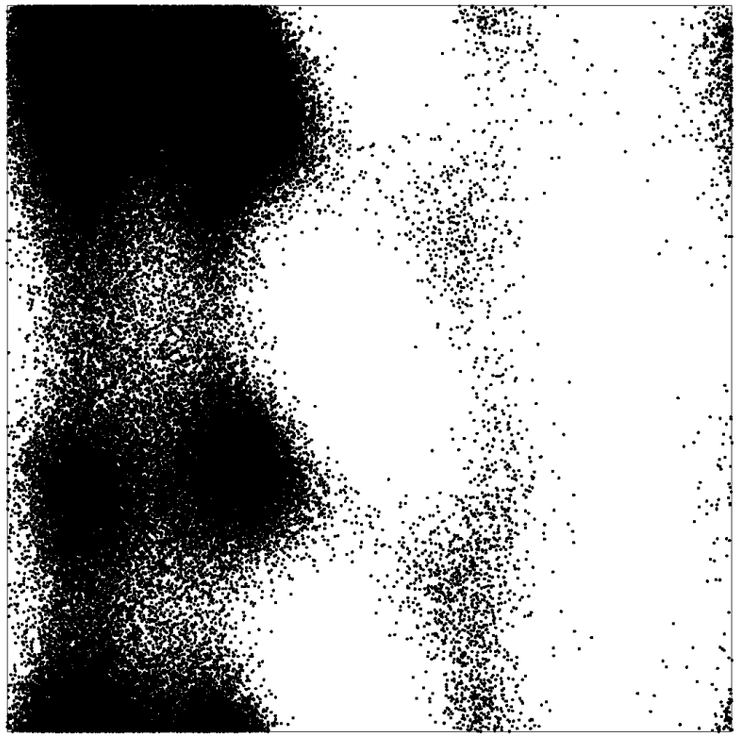


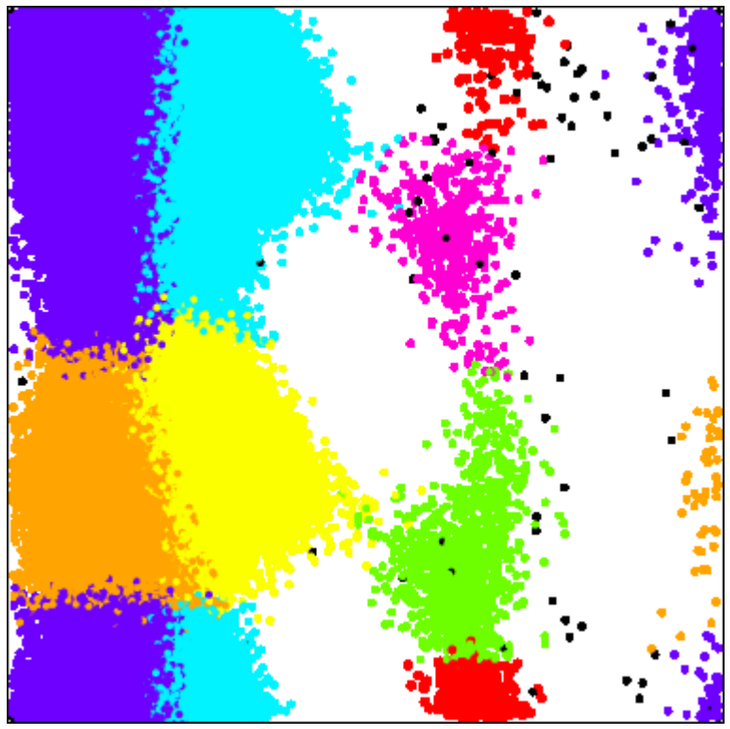
Experimental results

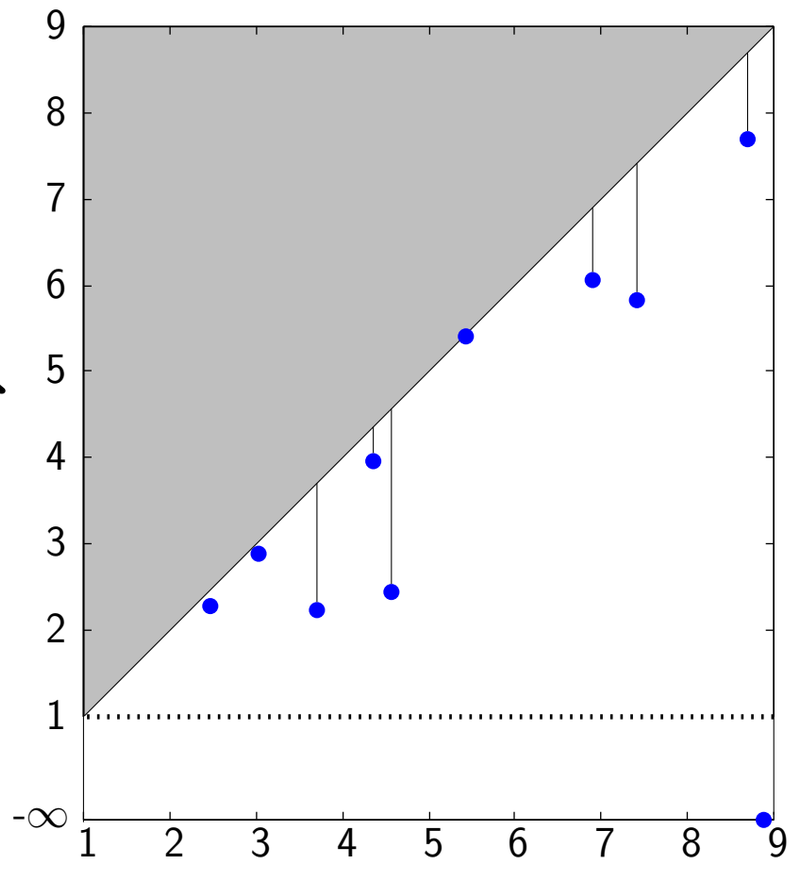
Biological Data

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

RMSD distance (non-Euclidean)







Common belief: 6 metastable states

PD shows anywhere between 4 and 7 clusters

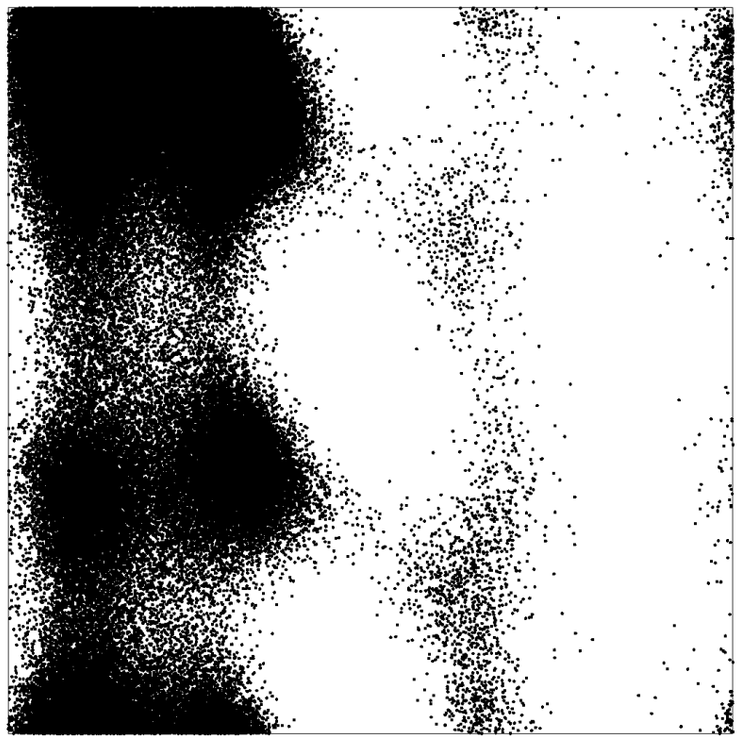
Experimental results

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

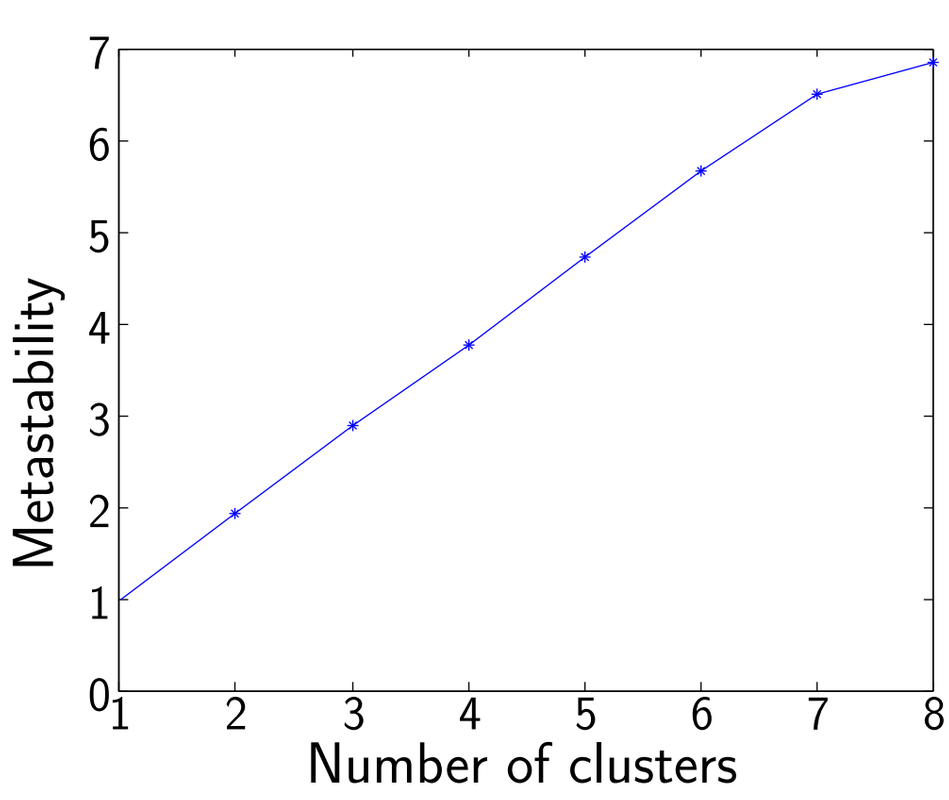
Biological Data

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

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

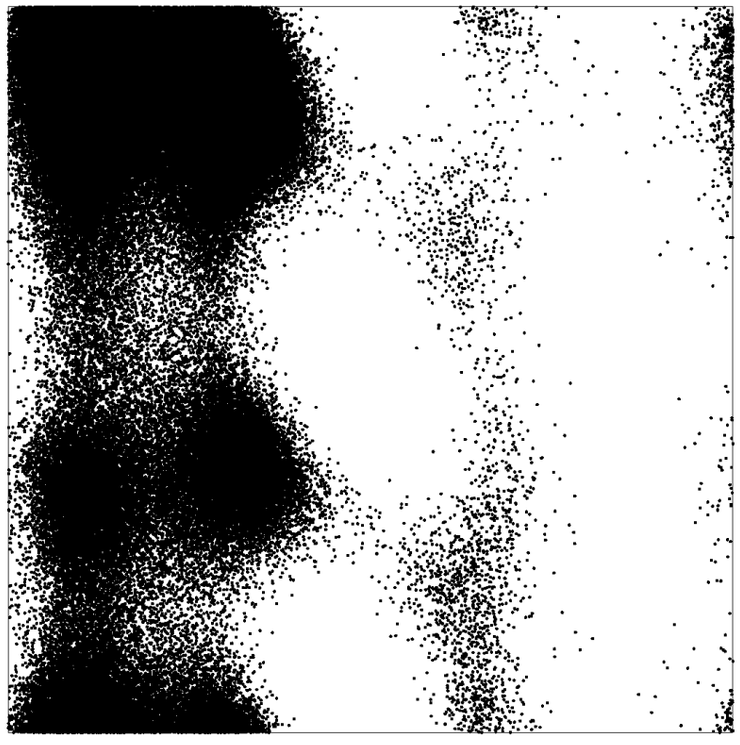
Experimental results

[*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 ($\mathbb{R}^{21}$)

RMSD distance (non-Euclidean)



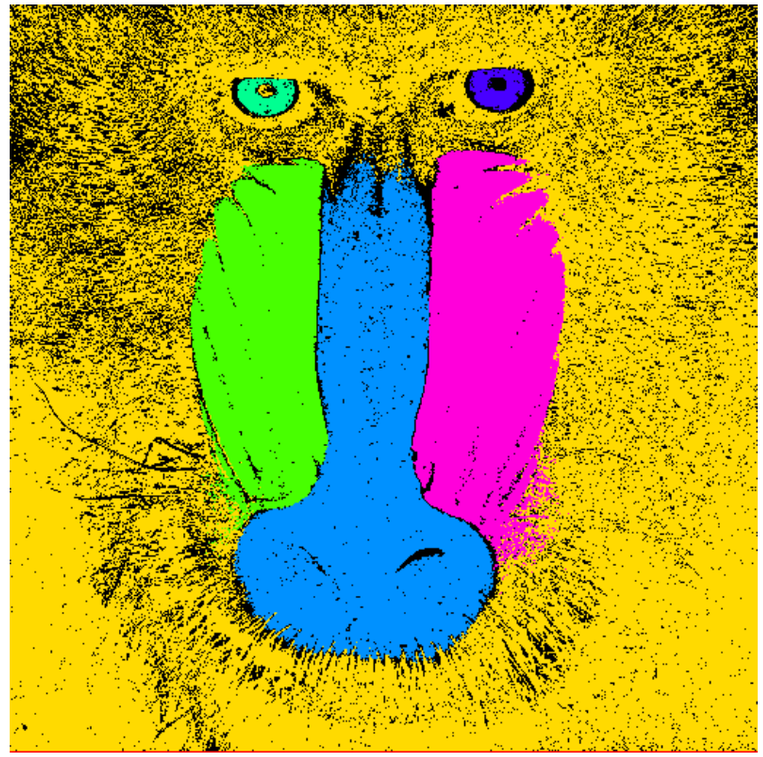
Note: Spectral Clustering takes a week of tweaking, while ToMATo runs out-of-the-box in a few minutes

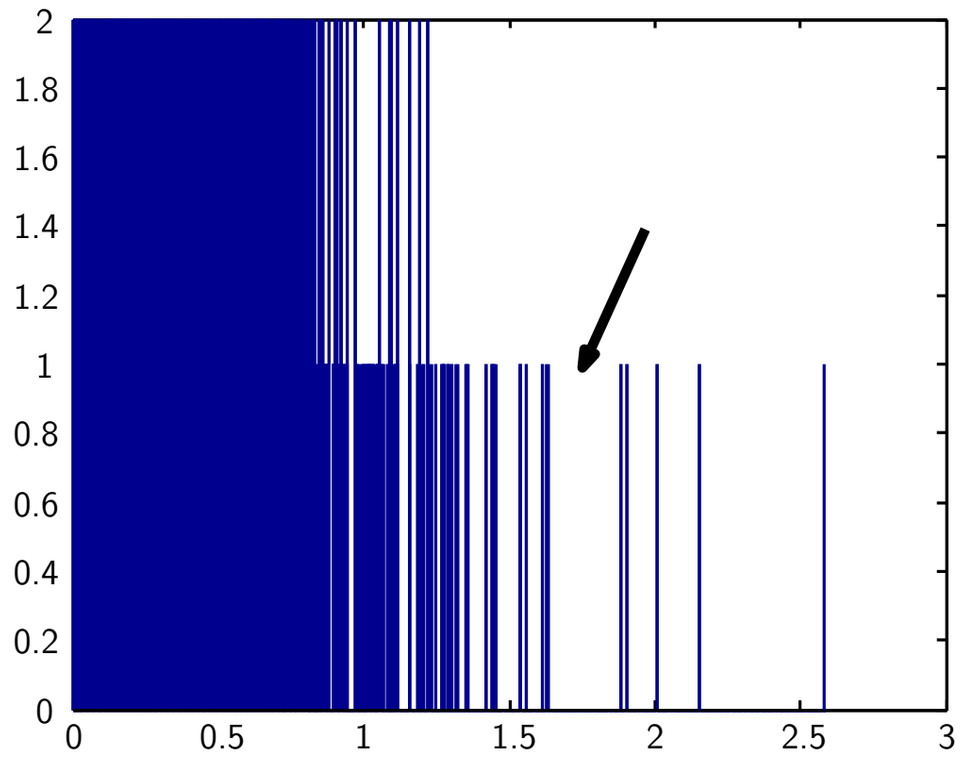
Experimental results

Image Segmentation

Density is estimated in 3D color space (Luv)

Neighborhood graph is built in image domain



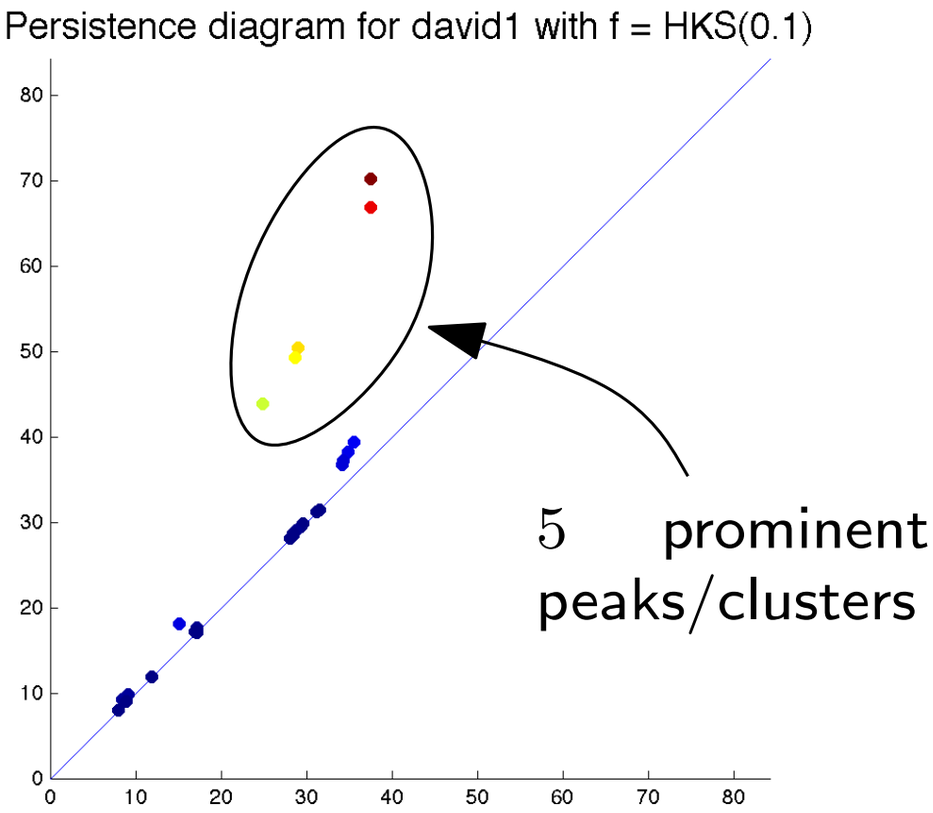


Distribution of prominences does not usually show a clear unique gap

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

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

Application to non-rigid shape segmentation

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

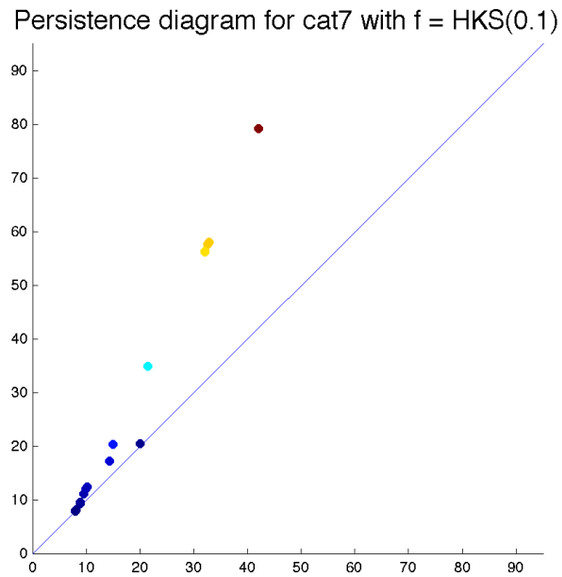
Problem: some part of clusters are unstable $\rightarrow$ dirty segments

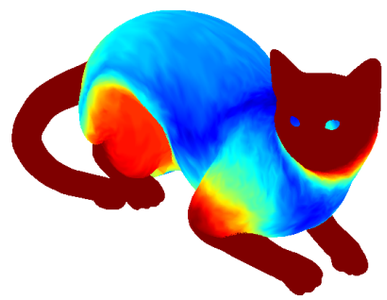
Idea:

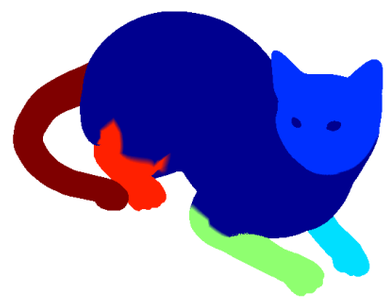
- Run the persistence based algorithm several times on random perturbations of f (size bounded by the "persistence" gap).
- Partial stability of clusters allows to establish correspondences between clusters across the different runs $\rightarrow$ for any $x \in X$, a vector giving the probability for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

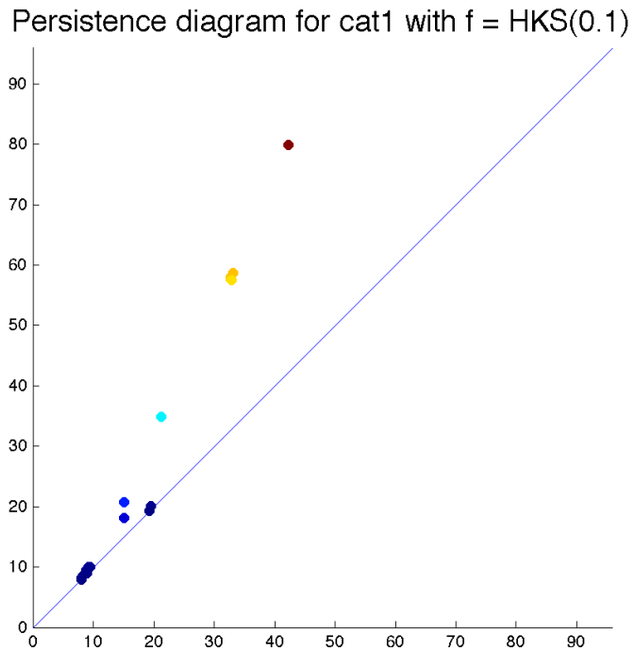
Application to non-rigid shape segmentation

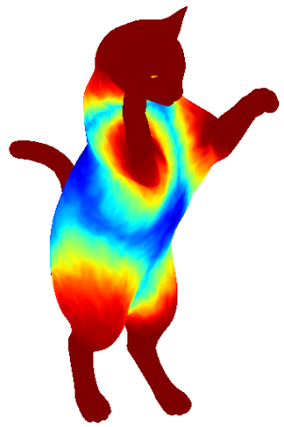
[*Persistence-Based Segmentation of Deformable Shapes*, Skraba, Ovsjanikov, Chazal, Guibas, Proc. CVPR 2010]









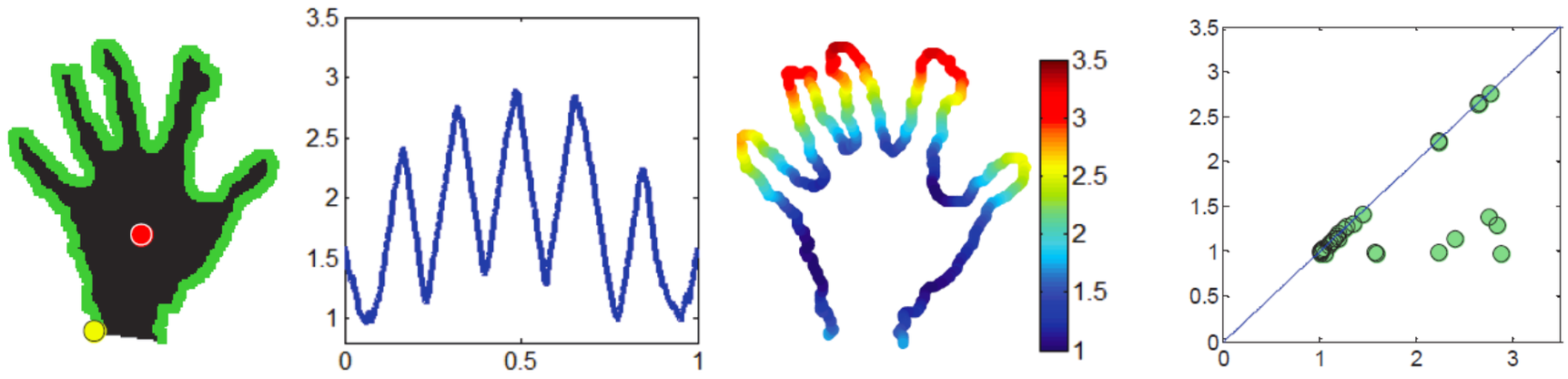


Other applications: classification, object recognition

Examples:

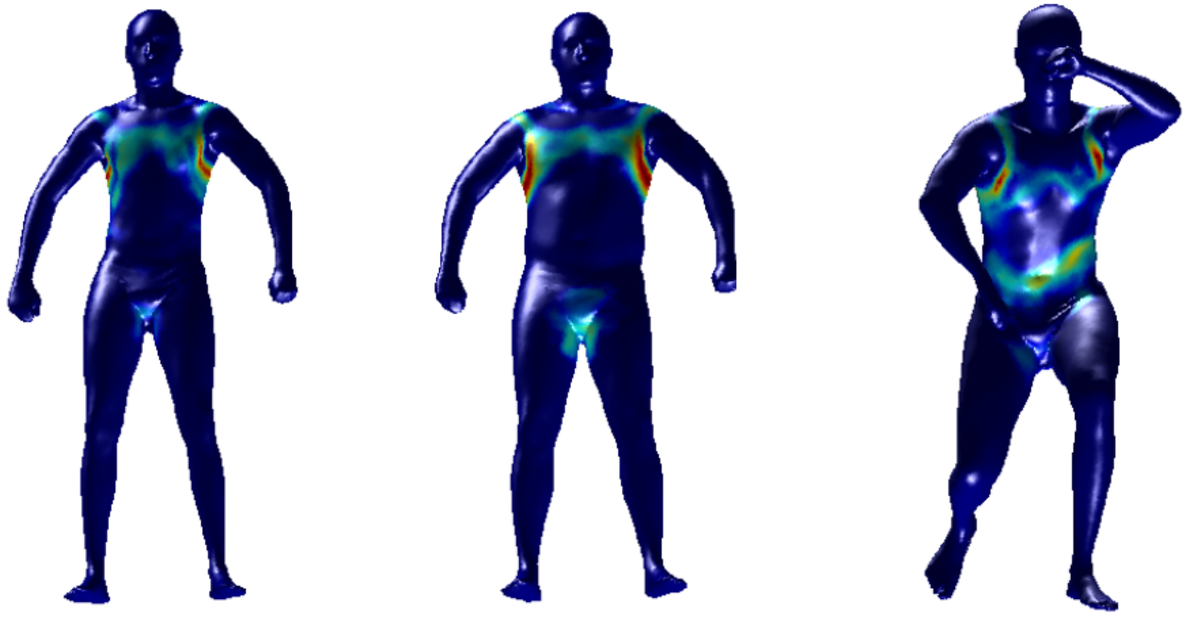
- Hand gesture recognition

[*Persistence-based structural recognition*, Li, Ovsjanikov, Chazal, Proc. CVPR, 2014]



- Persistence-based pooling for shape recognition

[*Persistence-based Pooling for Shape Pose Recognition*, Bonis, Ovsjanikov, Oudot, Chazal, 2015]



Thanks!

Projects:
https://drive.google.com/drive/folders/1oUpXXv-NWdcqDBOHjHv7jDvVaCObTJA2?usp=sharing

Gudhi:
https://gudhi.inria.fr/python/latest/index.html

The Distance To Measure (DTM)

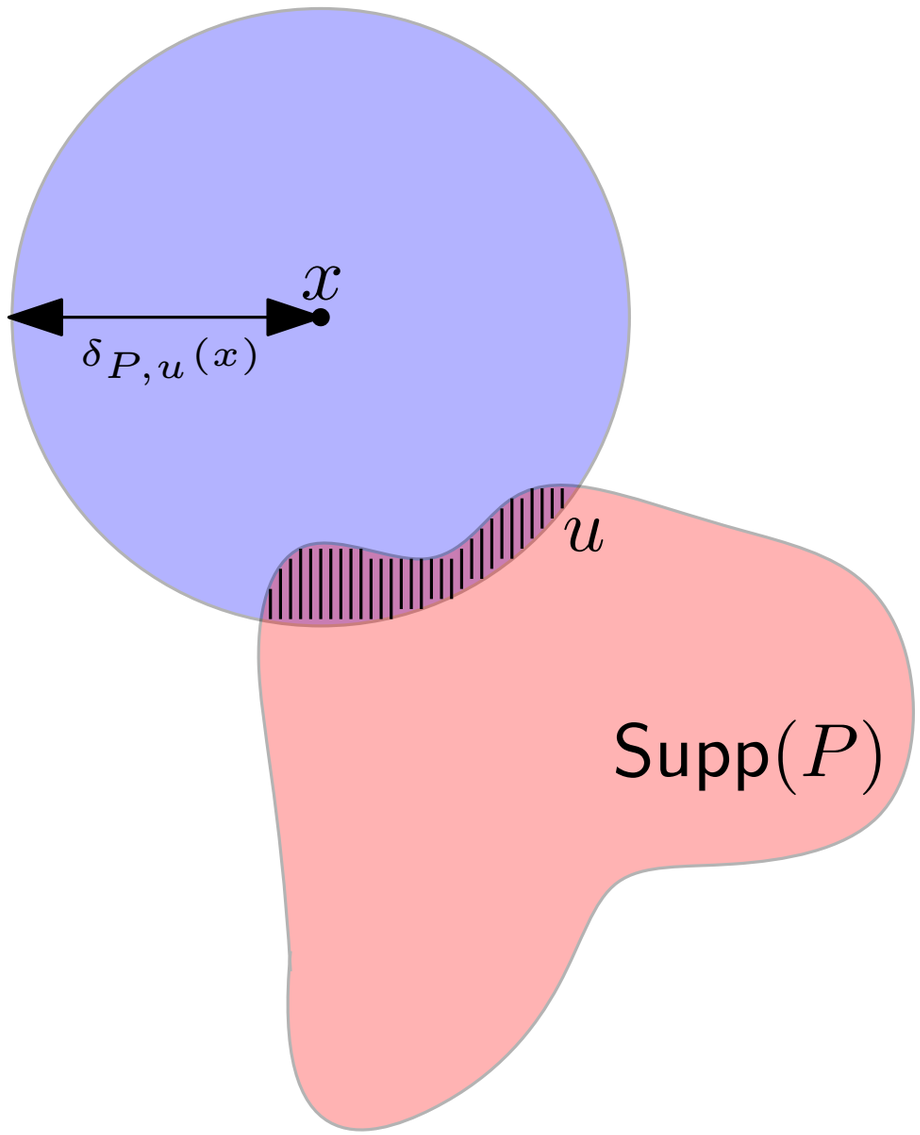
[*Geometric inference for probability measures*, Chazal, Cohen-Steiner, Mérigot, Found. Comput. Math., 2011]

The Distance To Measure (DTM)

[*Geometric inference for probability measures*, Chazal, Cohen-Steiner, Mérigot, Found. Comput. Math., 2011]

Preliminary distance function to a measure P: let $u \in]0,1[$ be a positive mass, and P a probability measure on $\mathbb{R}^d$:

$$\delta_{P,u}(x) = \inf\{r > 0 : P(B(x,r)) \geq u\}$$

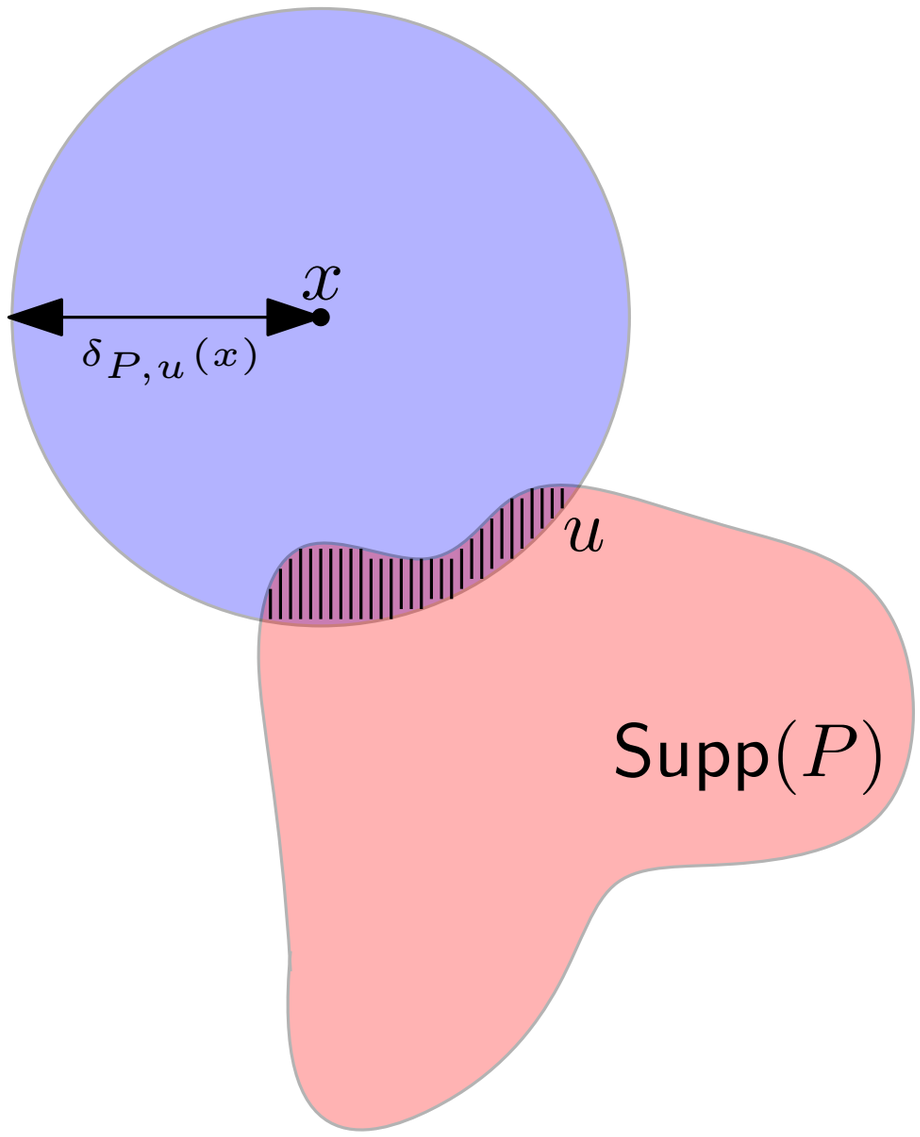


The Distance To Measure (DTM)

[*Geometric inference for probability measures*, Chazal, Cohen-Steiner, Mérigot, Found. Comput. Math., 2011]

Preliminary distance function to a measure P: let $u \in]0,1[$ be a positive mass, and P a probability measure on $\mathbb{R}^d$:

$$\delta_{P,u}(x) = \inf\{r > 0 : P(B(x,r)) \geq u\}$$



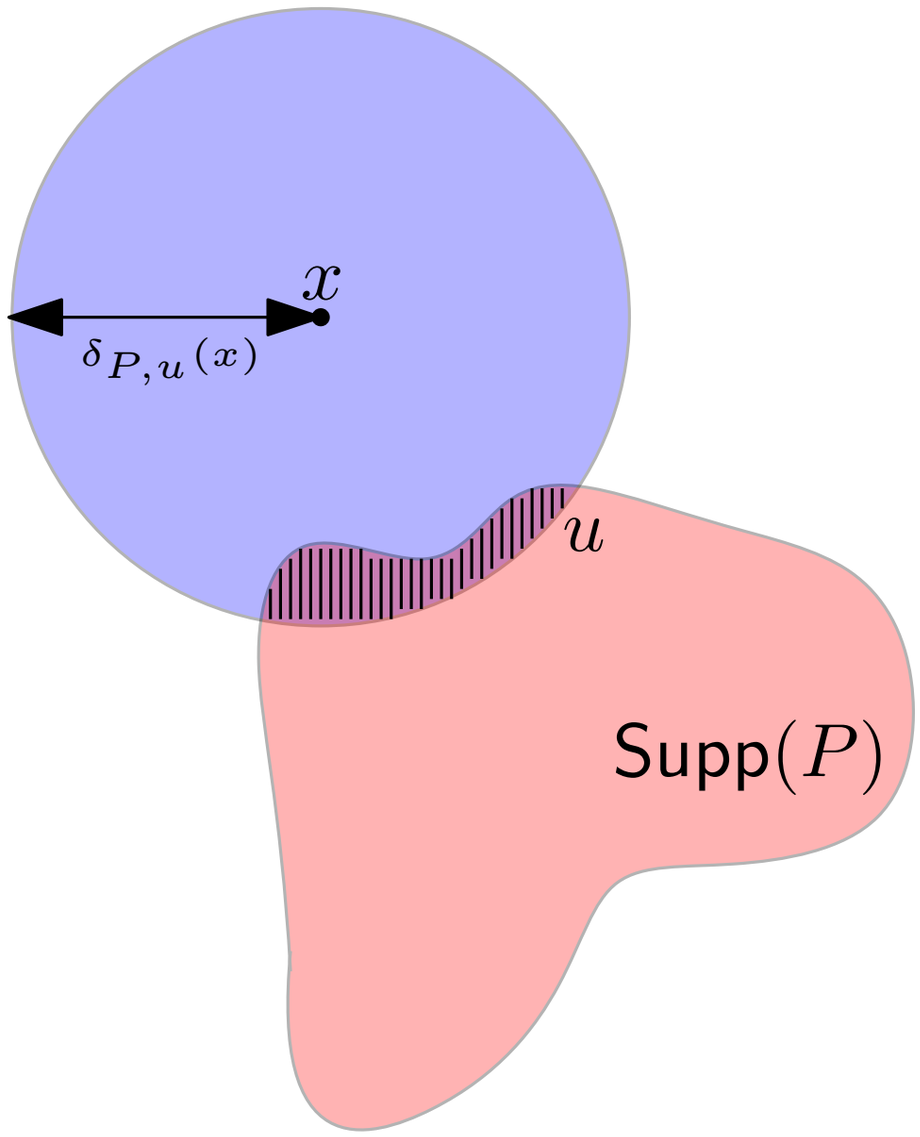
$\delta_{P,u}$ is the smallest distance needed to capture a mass of at least u.

The Distance To Measure (DTM)

[*Geometric inference for probability measures*, Chazal, Cohen-Steiner, Mérigot, Found. Comput. Math., 2011]

Preliminary distance function to a measure P: let $u \in]0,1[$ be a positive mass, and P a probability measure on $\mathbb{R}^d$:

$$\delta_{P,u}(x) = \inf\{r > 0 : P(B(x,r)) \geq u\}$$



$\delta_{P,u}$ is the smallest distance needed to capture a mass of at least u.

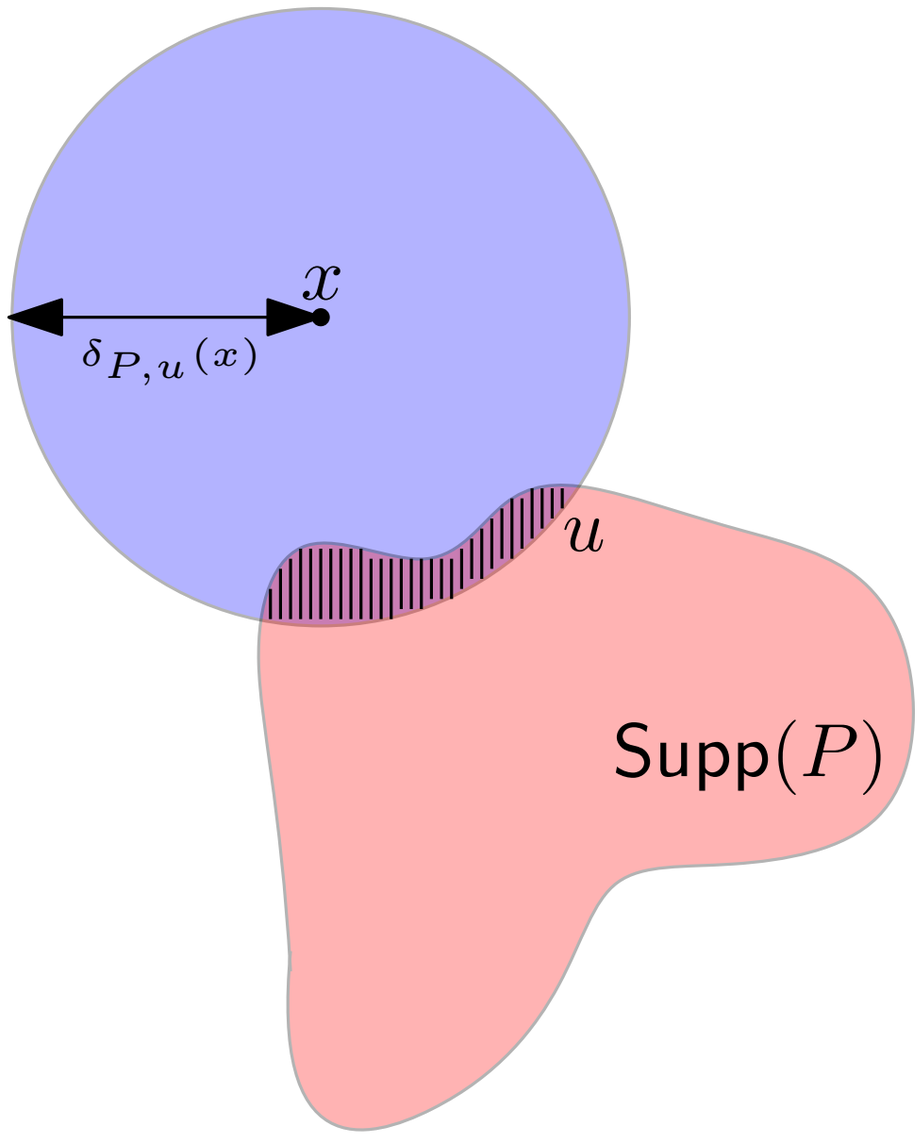
$\delta_{P,u}$ is the quantile function at u of the r.v. $\|x - X\|$ where $X \sim P$.

The Distance To Measure (DTM)

[*Geometric inference for probability measures*, Chazal, Cohen-Steiner, Mérigot, Found. Comput. Math., 2011]

Preliminary distance function to a measure P: let $u \in]0,1[$ be a positive mass, and P a probability measure on $\mathbb{R}^d$:

$$\delta_{P,u}(x) = \inf\{r > 0 : P(B(x,r)) \geq u\}$$



Def: Given a probability measure P on $\mathbb{R}^d$ and $m > 0$, the distance function to the measure P (DTM) is defined by

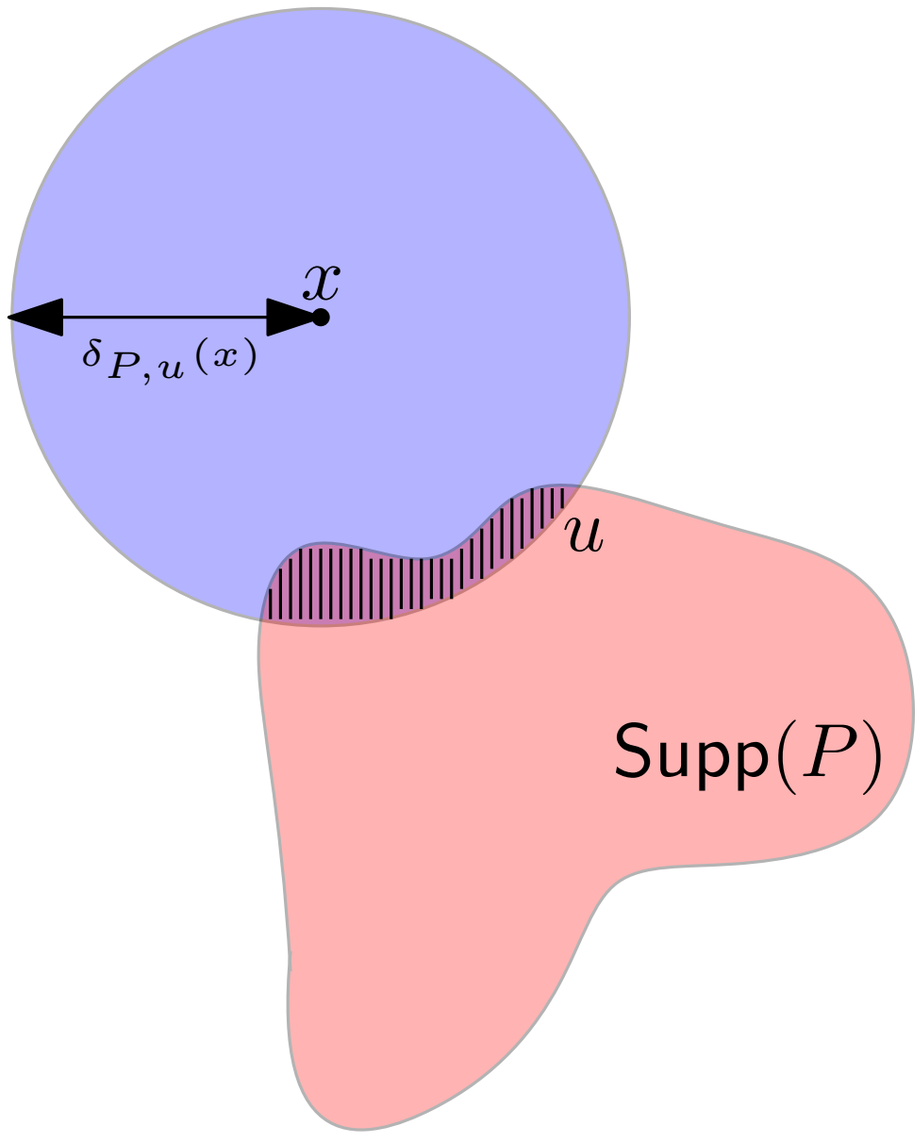
$$d_{P,m} : x \in \mathbb{R} \mapsto \left(\frac{1}{m}\int_0^m \delta_{P,u}^2(x)\mathrm{d}u\right)^{1/2}$$

The Distance To Measure (DTM)

[*Geometric inference for probability measures*, Chazal, Cohen-Steiner, Mérigot, Found. Comput. Math., 2011]

Preliminary distance function to a measure P: let $u \in]0,1[$ be a positive mass, and P a probability measure on $\mathbb{R}^d$:

$$\delta_{P,u}(x) = \inf\{r > 0 : P(B(x,r)) \geq u\}$$



Def: Given a probability measure P on $\mathbb{R}^d$ and $m > 0$, the distance function to the measure P (DTM) is defined by

$$d_{P,m} : x \in \mathbb{R} \mapsto \left(\frac{1}{m}\int_0^m \delta_{P,u}^2(x)\mathrm{d}u\right)^{1/2}$$

The DTM is robust, i.e., stable under Wasserstein perturbations:

$$\|d_{P,m} - d_{Q,m}\|_\infty \leq \frac{1}{\sqrt{m}} W_2(P,Q)$$

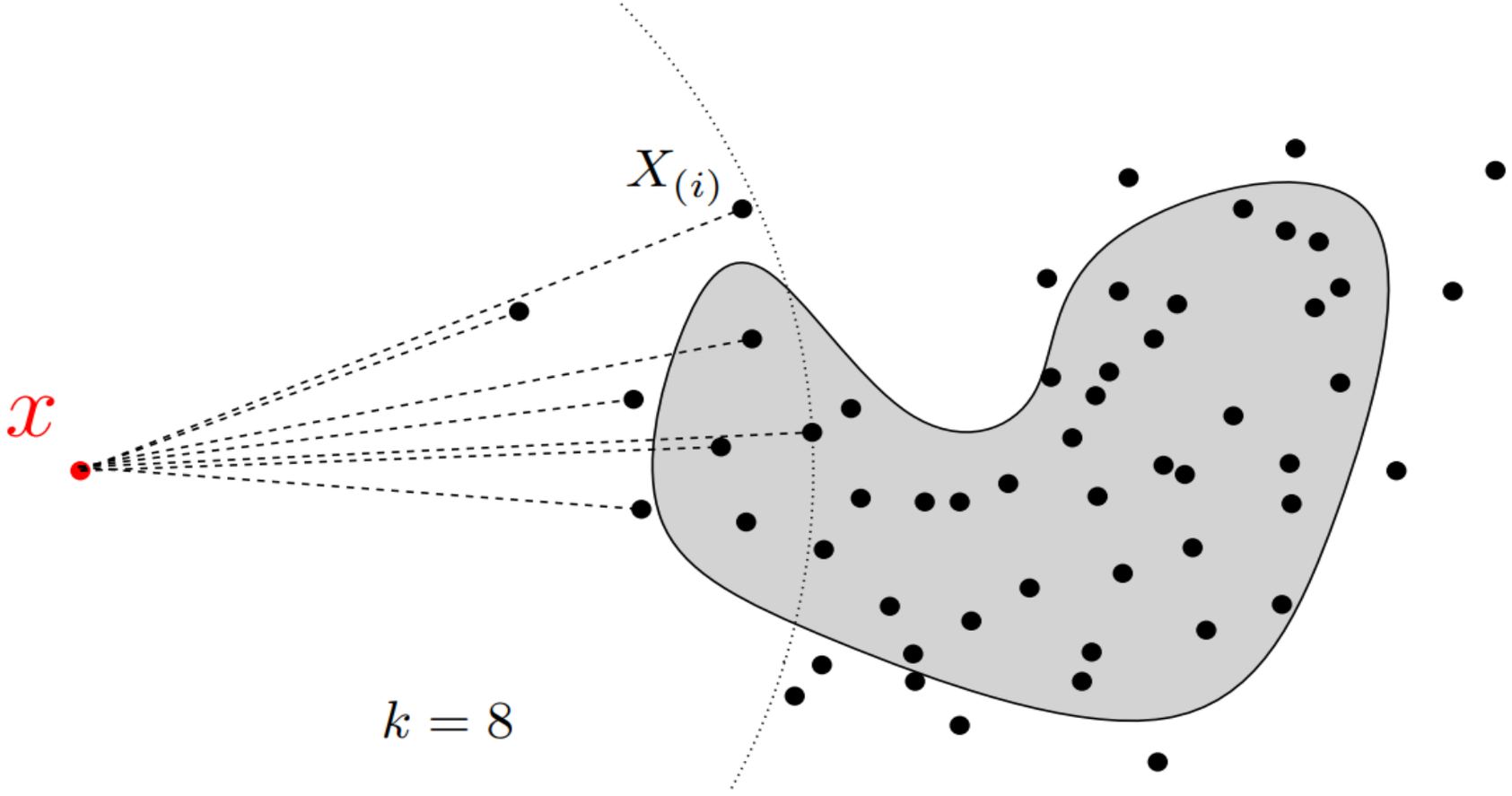
The Distance To Measure (DTM)

[*Geometric inference for probability measures*, Chazal, Cohen-Steiner, Mérigot, Found. Comput. Math., 2011]

Def: Let $X_1, \ldots, X_n$ sampled according to P and let P_n be the empirical measure. Then

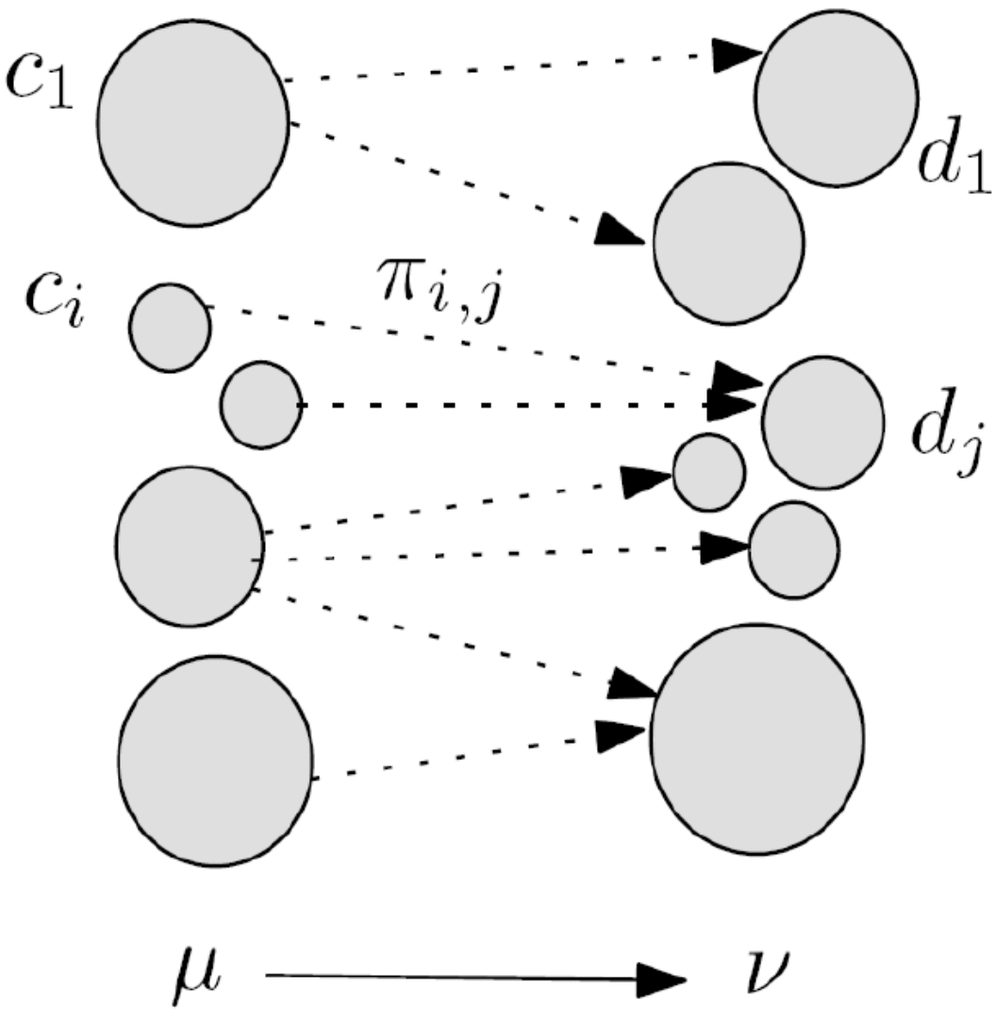
$$d_{P_n,k/n}(x) = \frac{1}{k} \sum_{i=1}^{k} \|x - X_{(i)}\|^2,$$

where $\|X_{(1)} - x\| \leq \|X_{(2)} - x\| \leq \cdots \leq \|X_{(k)} - x\| \leq \cdots \leq \|X_{(n)} - x\|$.



The Wasserstein distance

Let (X, d) be a metric space and let μ, ν be probability measures on X with finite p-moments ($p \geq 1$). The Wasserstein distance $W_p(\mu, \nu)$ quantifies the optimal cost of pushing μ onto ν, the cost of moving a small mass dx from x to y being $d(x, y)^p dx$.

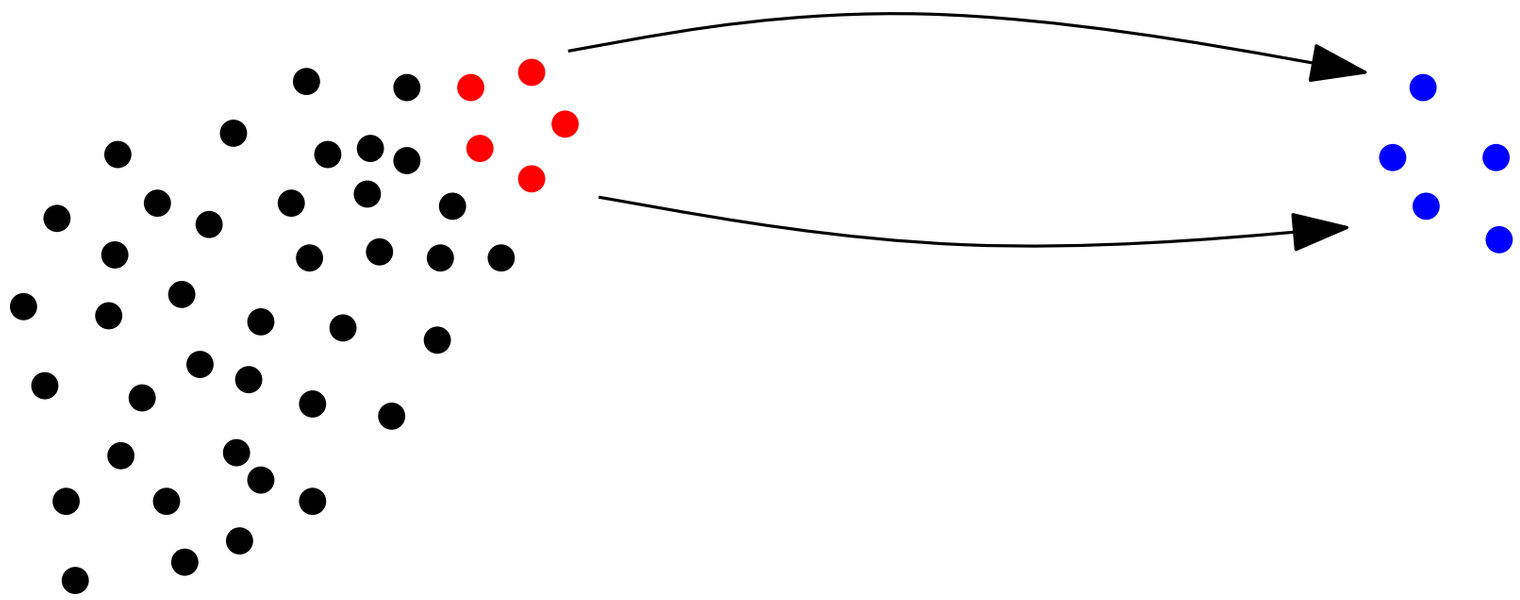


- Transport plan: Π a probability measure on $X \times X$ s.t. $\Pi(A \times \mathbb{R}^d) = \mu(A)$ and $\Pi(\mathbb{R}^d \times B) = \nu(B)$ for any borelian sets $A, B \subseteq X$.
- Cost of a transport plan:

$$C(\Pi) = \left(\int_{X \times X} d(x, y)^p d\Pi(x, y) \right)^{\frac{1}{p}}$$

- $W_p(\mu, \nu) = \inf_\Pi C(\Pi)$.

The Wasserstein distance



Ex: If $P = \{p_1, \ldots, p_n\}$ is a point cloud, and $P' = \{p_1, \ldots, p_{n-k-1}, o_1, \ldots, o_k\}$ with $d(o_i, P) = R$, then

$$d_H(P, P') \geq R \qquad \text{but} \qquad W_2(\mu_P, \mu_{P'}) \leq \sqrt{\frac{k}{n}}(R + \operatorname{diam}(P))$$

DTM-based filtrations

[*DTM-based filtrations*, Anai et al., Symp. Comp. Geom., 2019]

Def: Let V be a point cloud (in a metric space). The DTM-based complex $W(V)$ is the filtered simplicial complex indexed by $\mathbb{R}$ whose vertex set is V and whose other simplices are defined with

$$\sigma = [p_0, p_1 \ldots, p_k] \in W(V, \alpha) \iff \cap_{i=0}^{k} B(p_i, r_{p_i}(\alpha)) \neq \emptyset$$

where $r_p(\alpha) = 0$ if $\alpha \leq d_{P_n,k/n}(p)$ and $|\alpha^q - d_{P_n,k/n}(p)^q|^{1/q}$ otherwise.

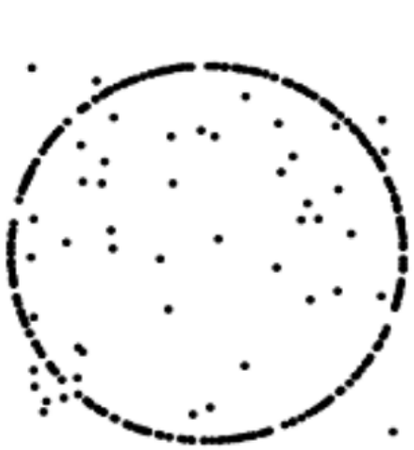
DTM-based filtrations

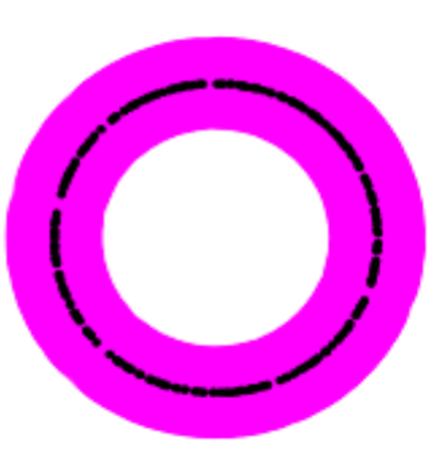
[*DTM-based filtrations*, Anai et al., Symp. Comp. Geom., 2019]

Def: Let V be a point cloud (in a metric space). The DTM-based complex $W(V)$ is the filtered simplicial complex indexed by $\mathbb{R}$ whose vertex set is V and whose other simplices are defined with

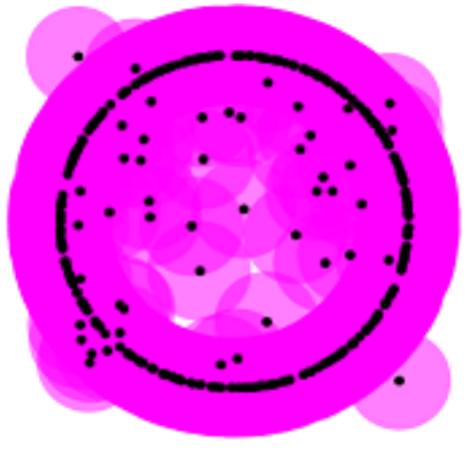
$$\sigma = [p_0, p_1 \ldots, p_k] \in W(V, \alpha) \iff \cap_{i=0}^{k} B(p_i, r_{p_i}(\alpha)) \neq \emptyset$$

where $r_p(\alpha) = 0$ if $\alpha \leq d_{P_n,k/n}(p)$ and $|\alpha^q - d_{P_n,k/n}(p)^q|^{1/q}$ otherwise.

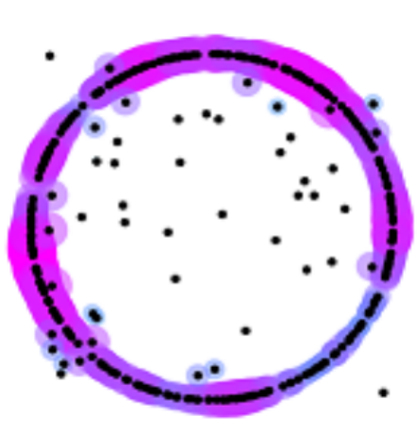




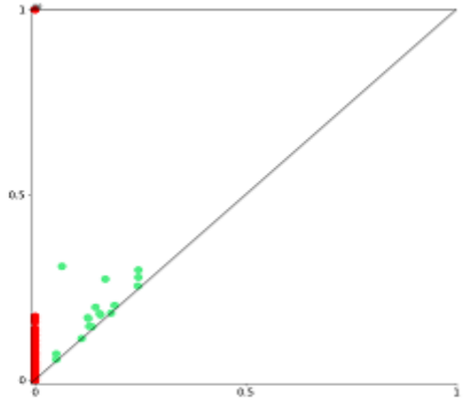
Rips

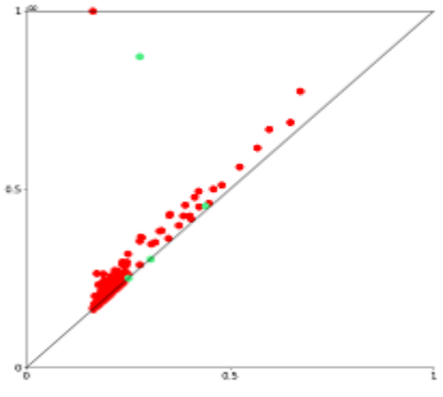


Rips



DTM-based





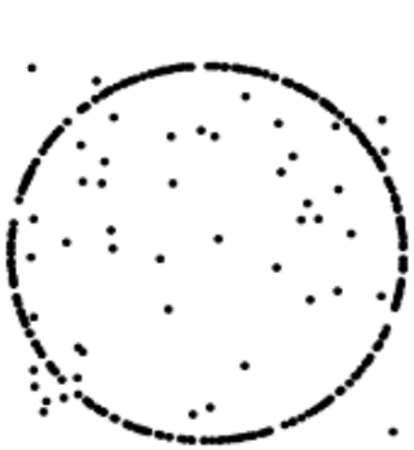
DTM-based filtrations

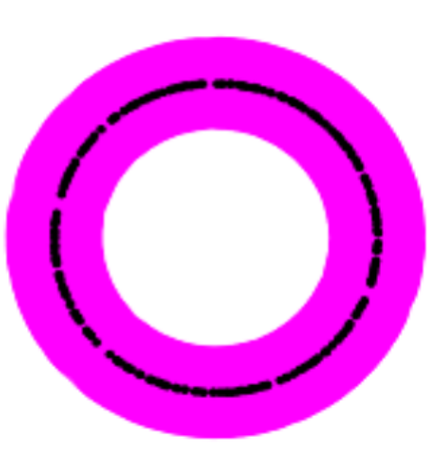
[*DTM-based filtrations*, Anai et al., Symp. Comp. Geom., 2019]

Def: Let V be a point cloud (in a metric space). The DTM-based complex $W(V)$ is the filtered simplicial complex indexed by $\mathbb{R}$ whose vertex set is V and whose other simplices are defined with

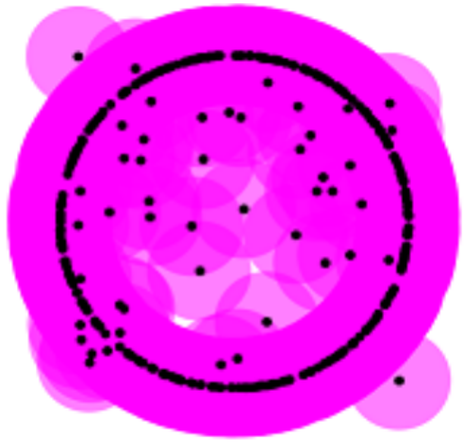
$$\sigma = [p_0, p_1 \ldots, p_k] \in W(V, \alpha) \iff \cap_{i=0}^{k} B(p_i, r_{p_i}(\alpha)) \neq \emptyset$$

where $r_p(\alpha) = 0$ if $\alpha \leq d_{P_n,k/n}(p)$ and $|\alpha^q - d_{P_n,k/n}(p)^q|^{1/q}$ otherwise.

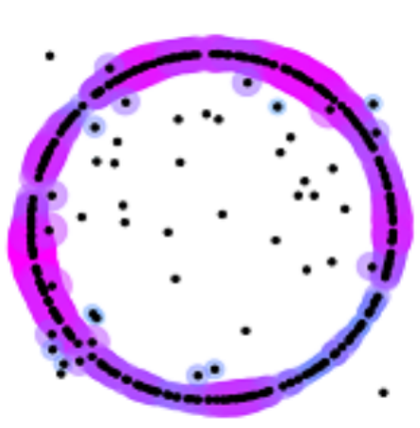




Rips



Rips



DTM-based

Thm: $d_b(W(X), W(Y)) \leq \sqrt{\frac{n}{k}} W_2(X, Y) + 2^{1/q} d_H(X, Y)$.