Geometric and Topological
Methods in Machine Learning
2021-2022

Computational Topology: Simplicial Complexes and (Persistent) Homology

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

Class outline: taking a step back...

My classes are about

Topological Data Analysis (TDA)

Class outline: taking a step back...

My classes are about

Topological Data Analysis (TDA)

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

Class outline: taking a step back...

My classes are about

Topological Data Analysis (TDA)

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

Question: What is topology?

Class outline: taking a step back...

My classes are about

Topological Data Analysis (TDA)

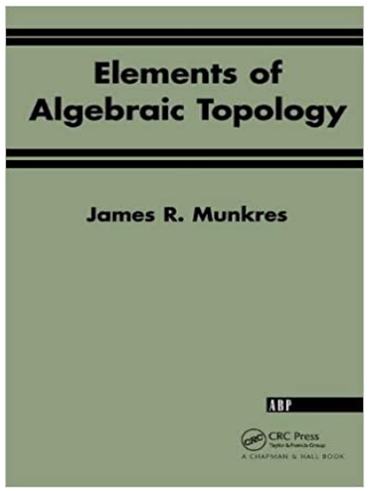
Goal: Study geometric data sets with techniques coming from *topology*.

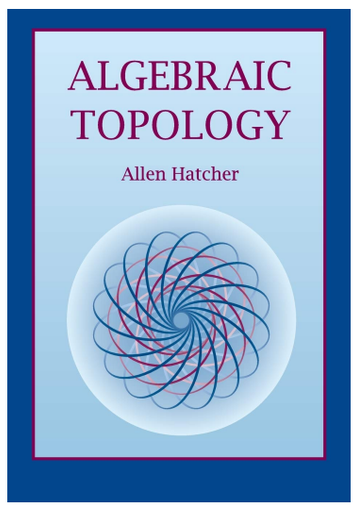
Question: What is topology?

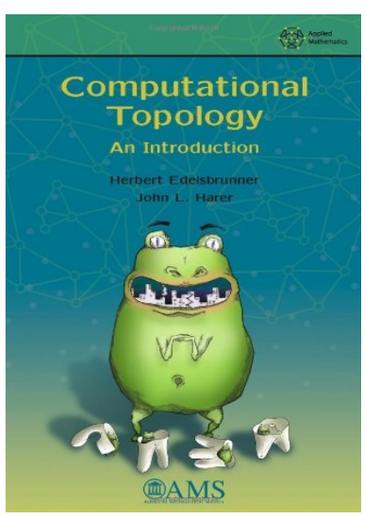
[*Elements of Algebraic Topology*, Munkres, CRC Press, 1984]

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

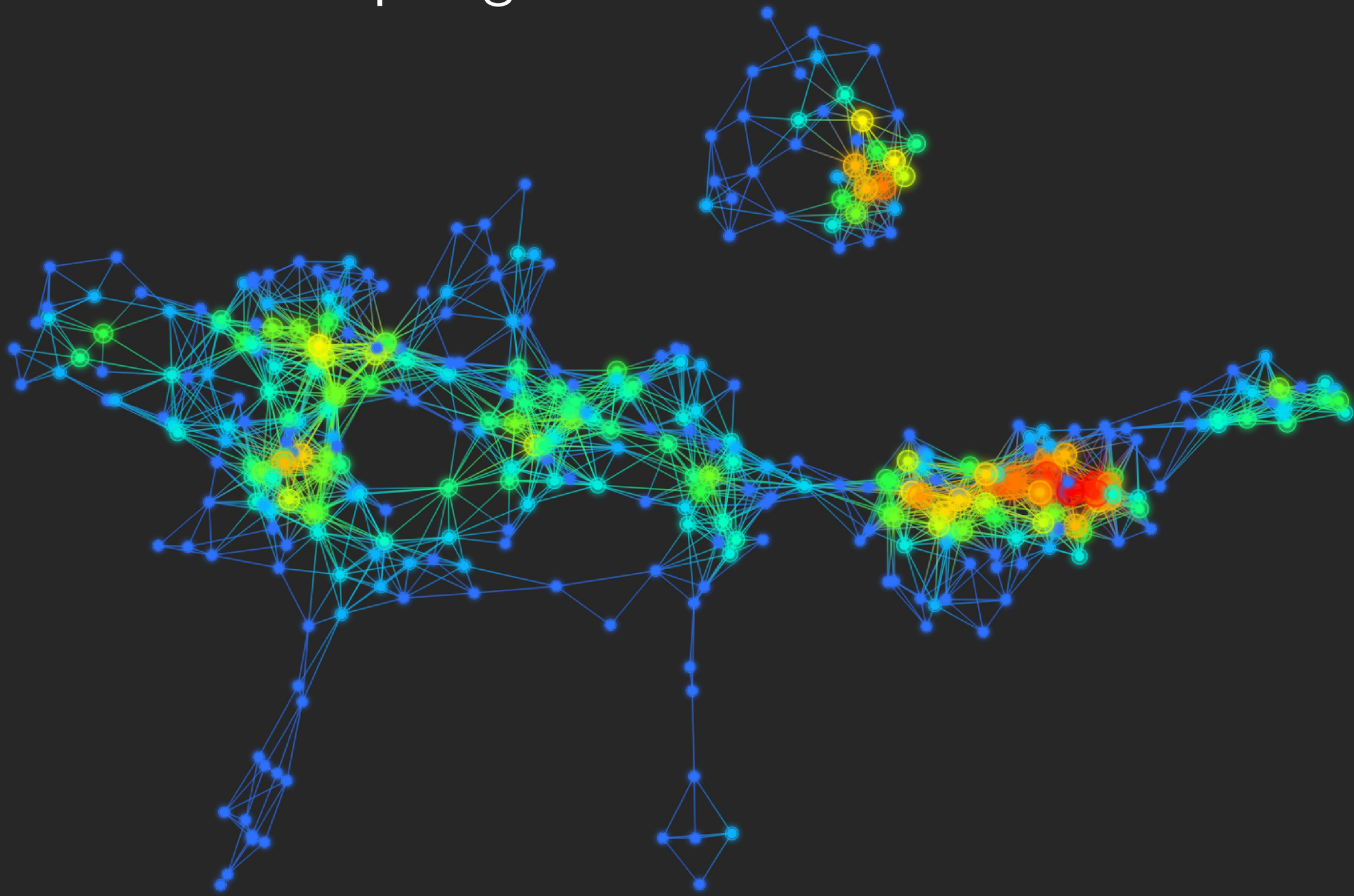
[*Computational Topology: an introduction*, Edelsbrunner, Harer, AMS, 2010]



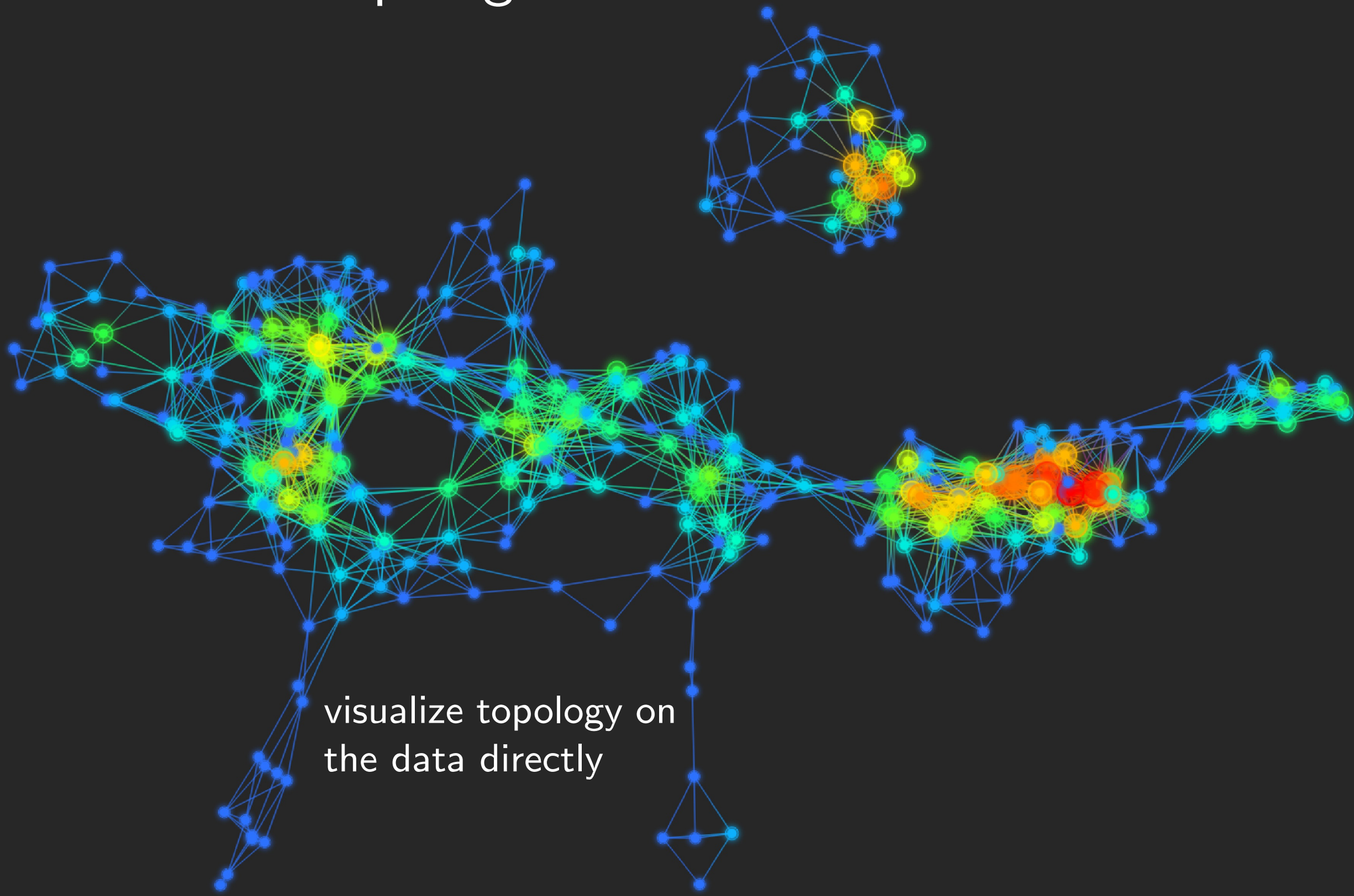




Introduction: topological visualization



Introduction: topological visualization



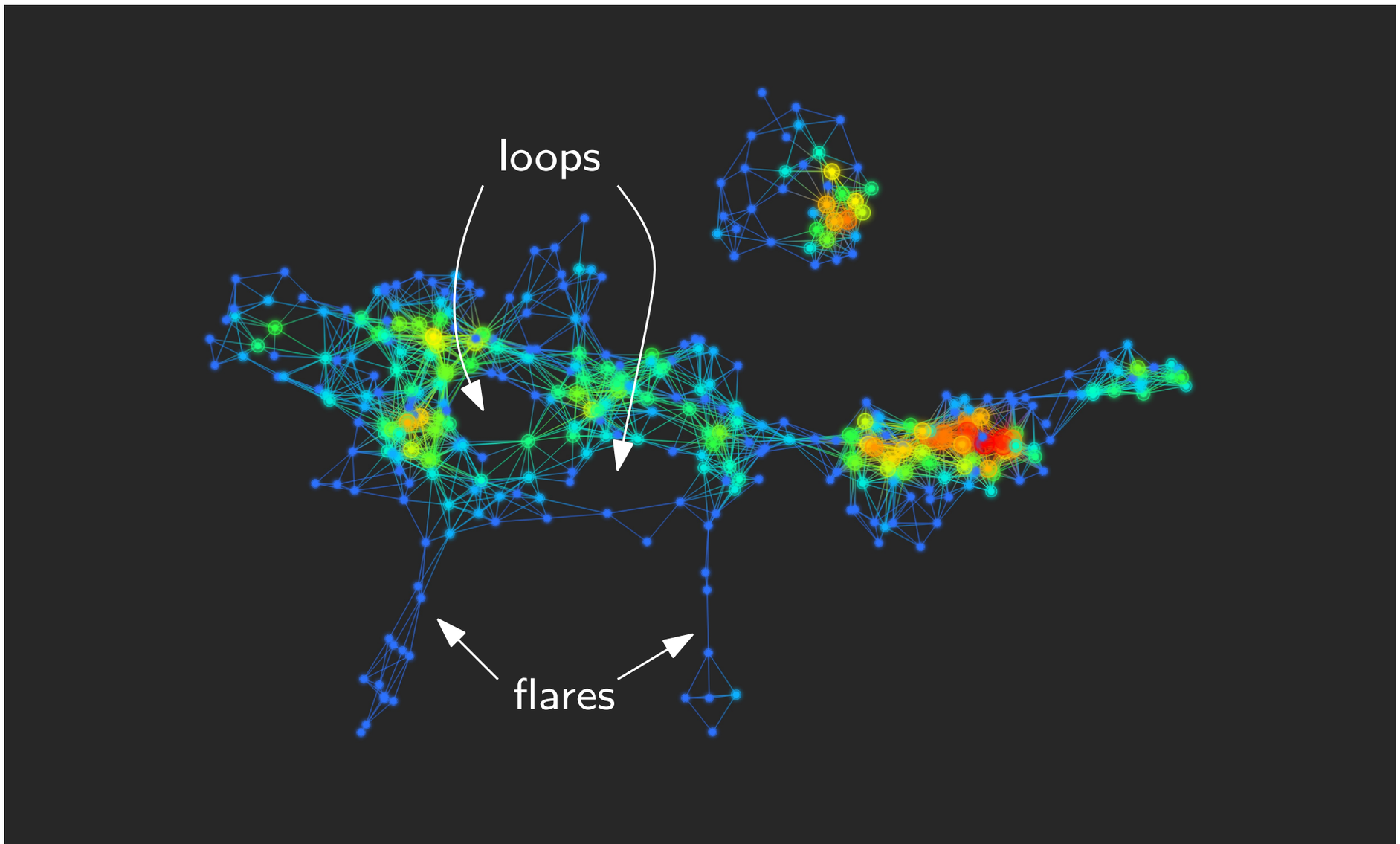
visualize topology on
the data directly

Introduction: topological visualization

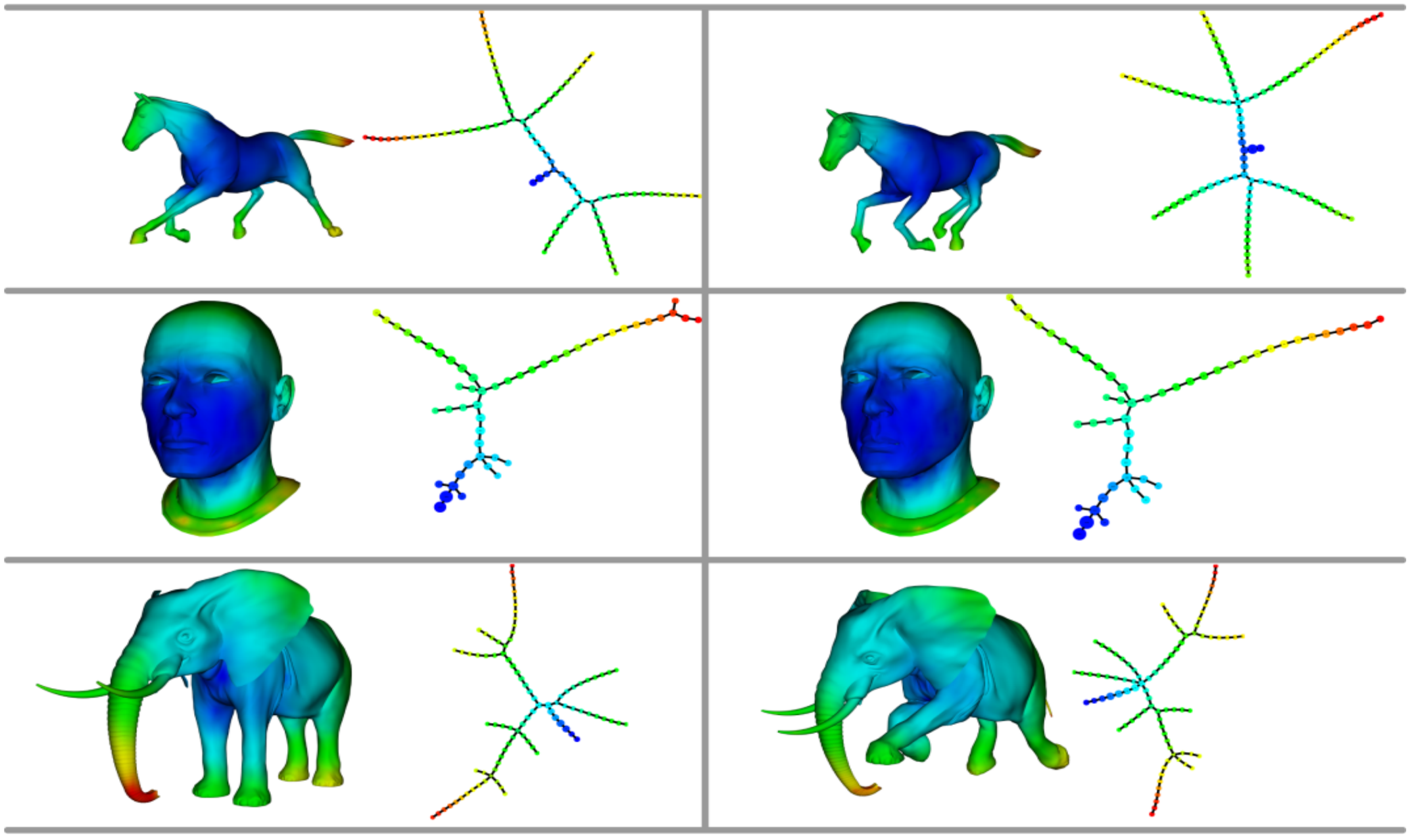
Two types of applications:

- 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]

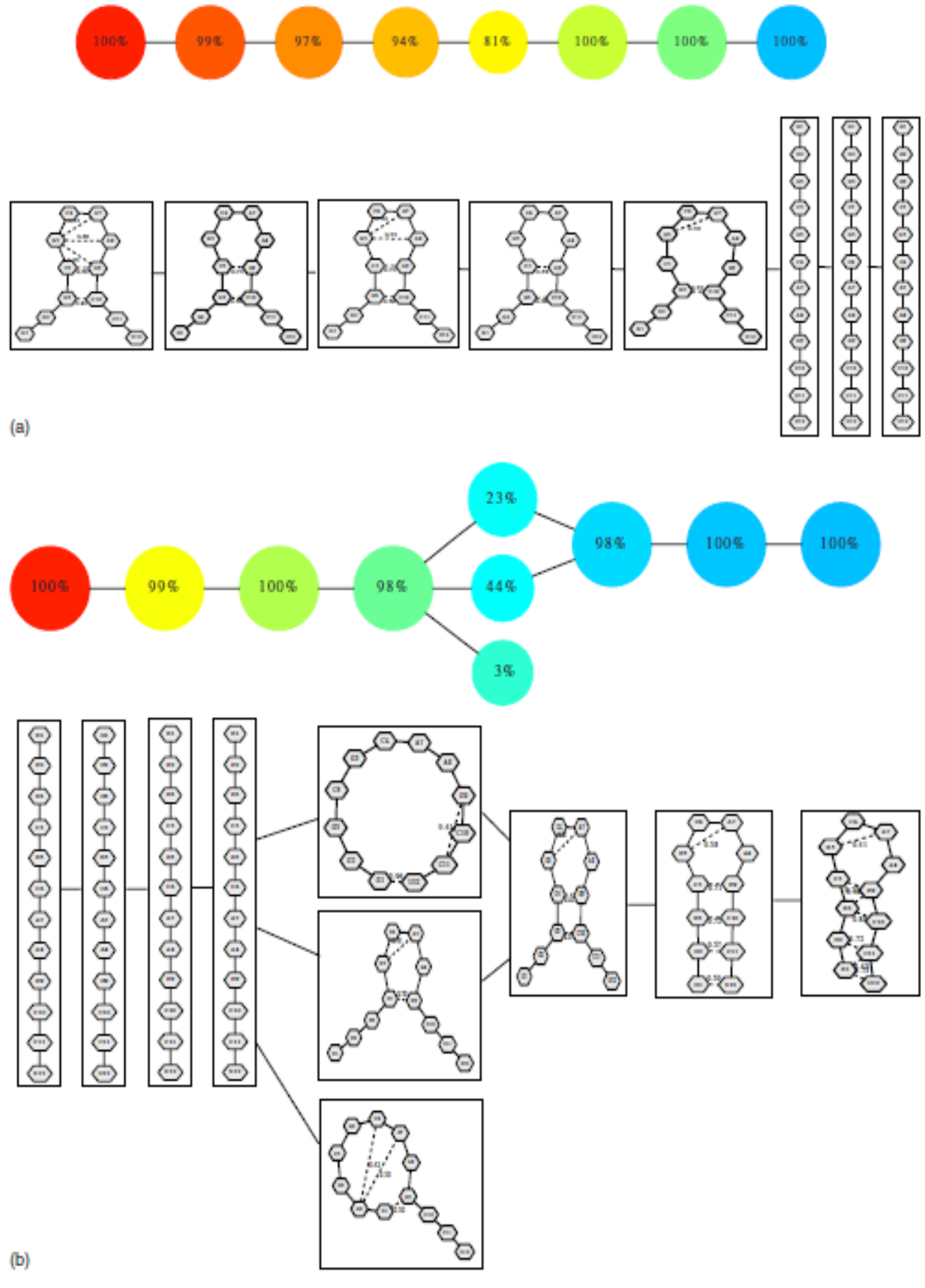
Introduction: topological visualization

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

Data: conformations of molecules.

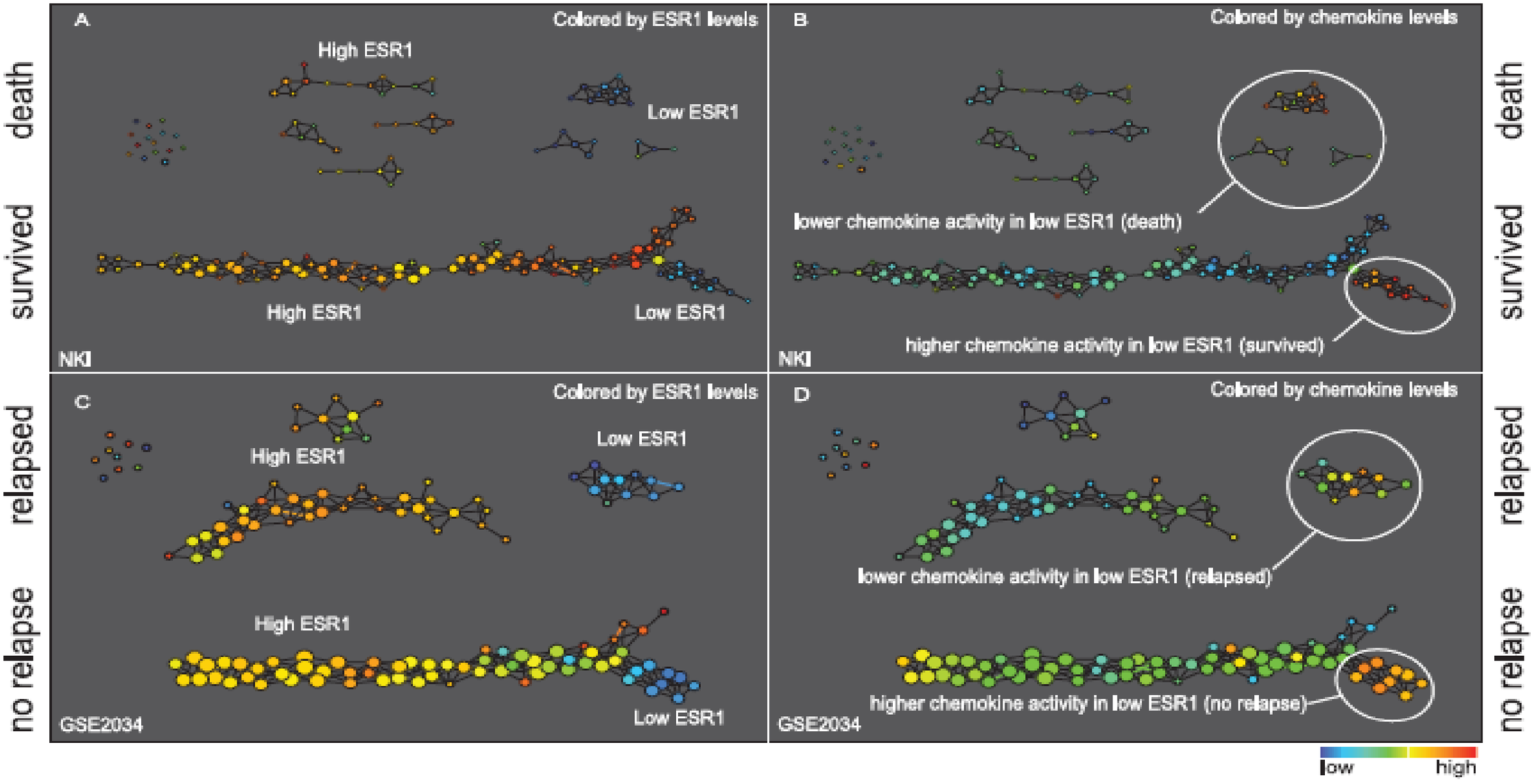
Goal: detect folding pathways.

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

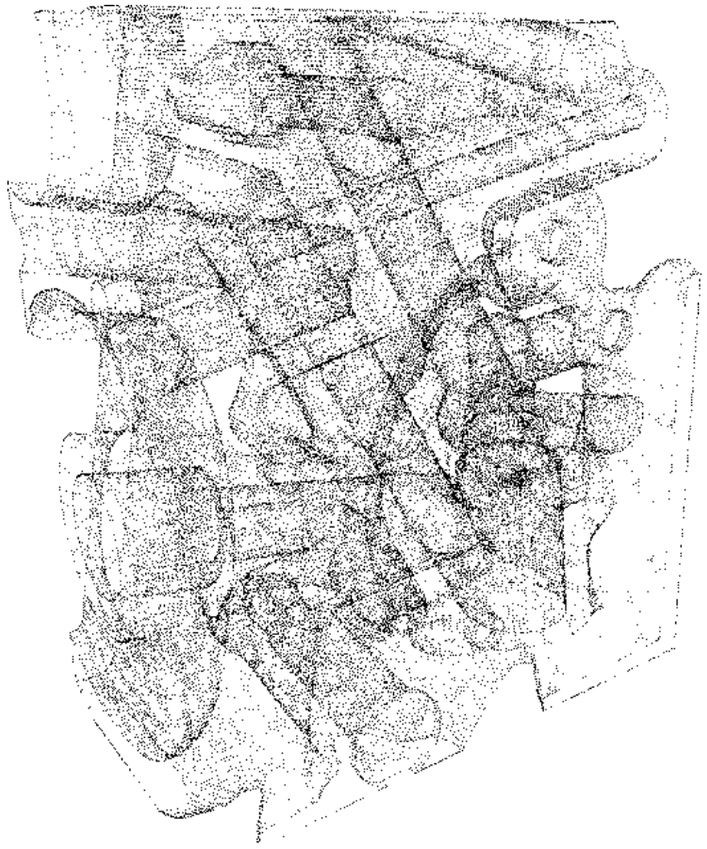
We will see how to build new topological features from data sets...

Introduction: topological descriptors built from data

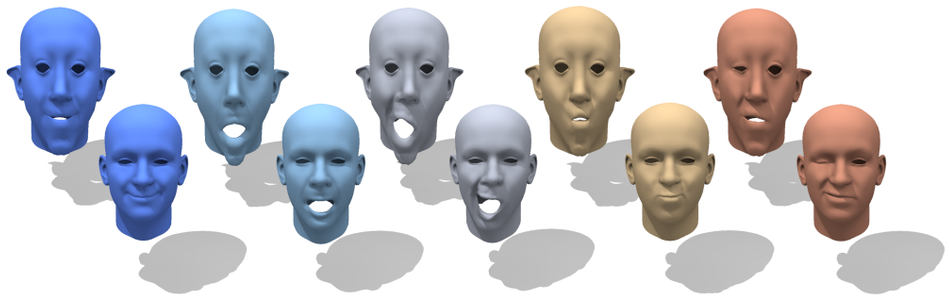
We will see how to build new topological features from data sets...

...but why is that interesting?

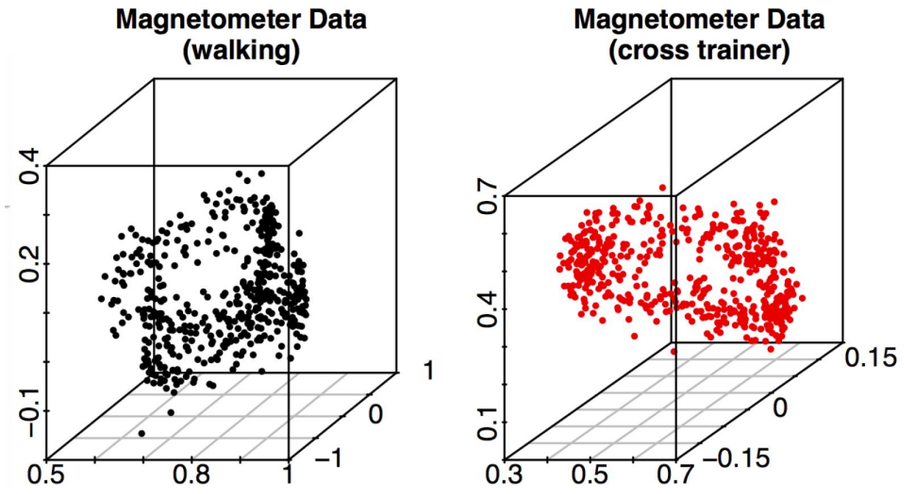
Introduction: topological descriptors built from data



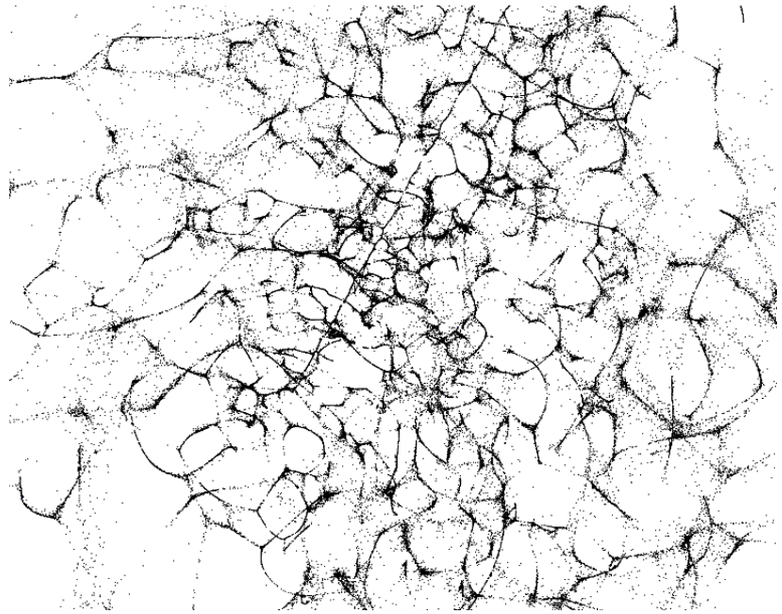
Scans



3D shapes

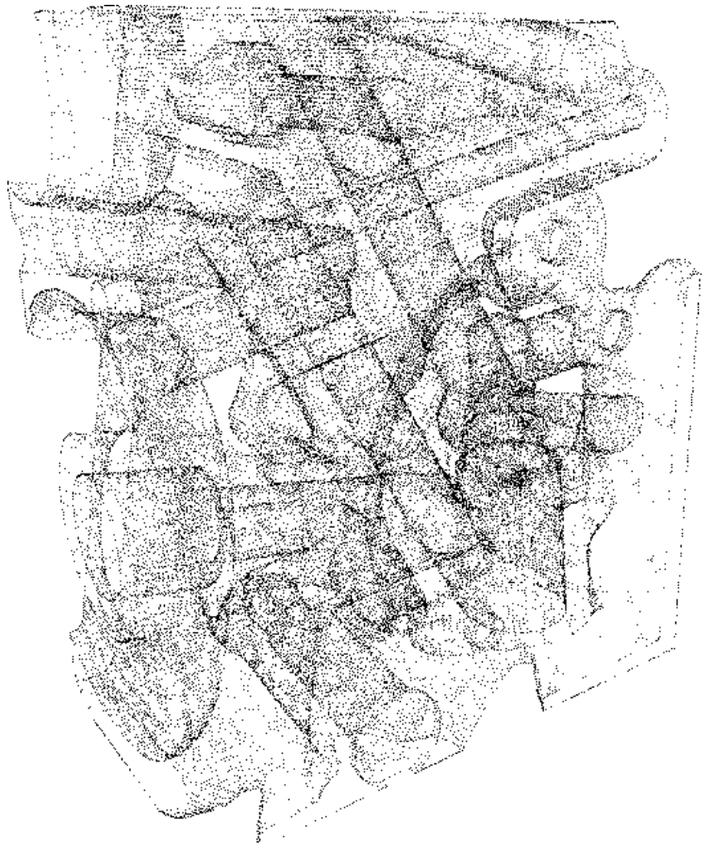


Magnetometer

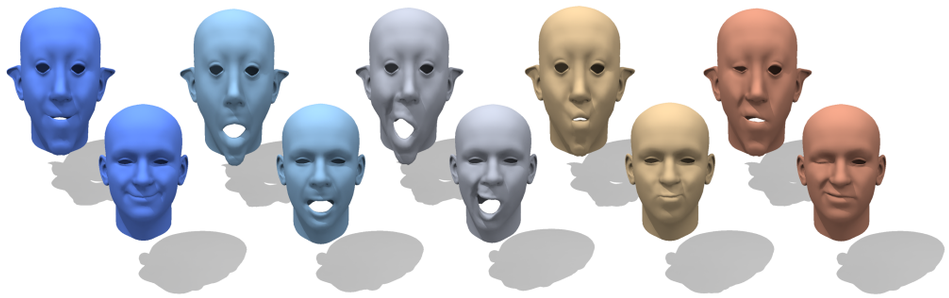


Galaxies

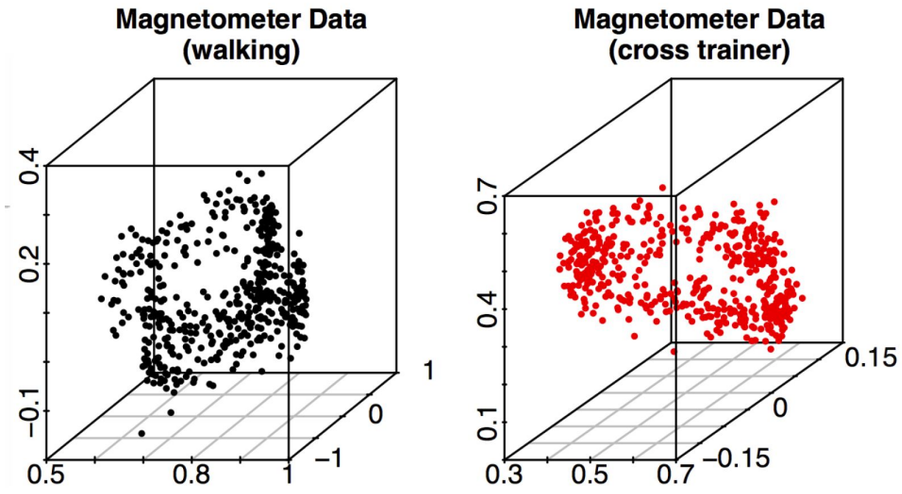
Introduction: topological descriptors built from data



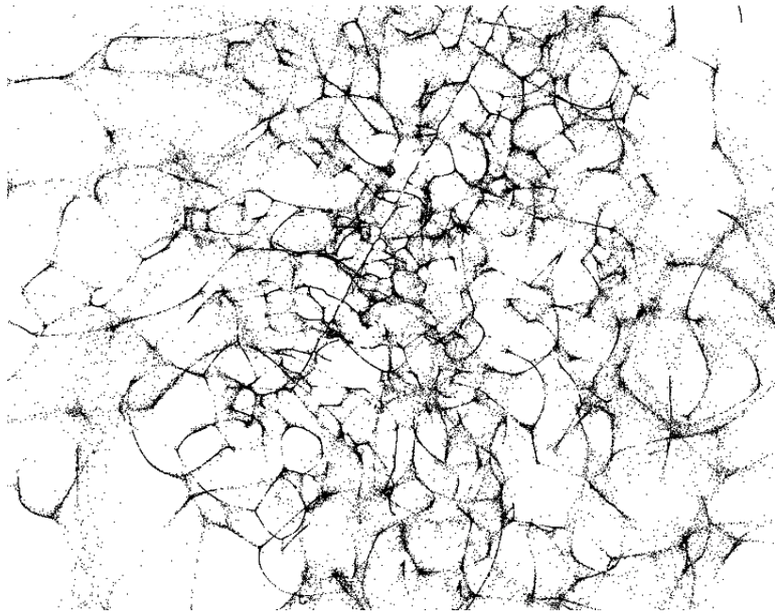
Scans



3D shapes



Magnetometer

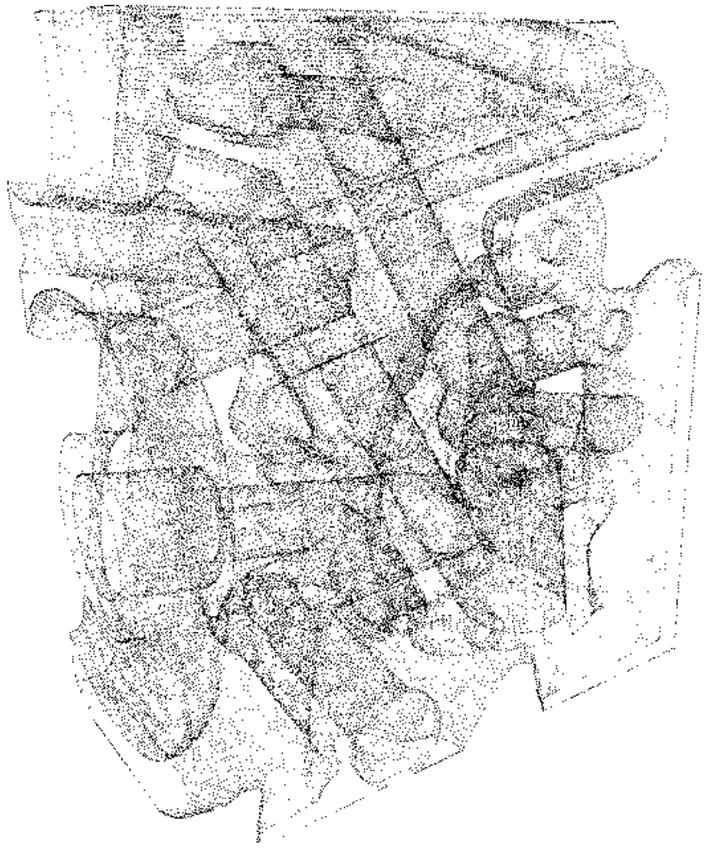


Galaxies

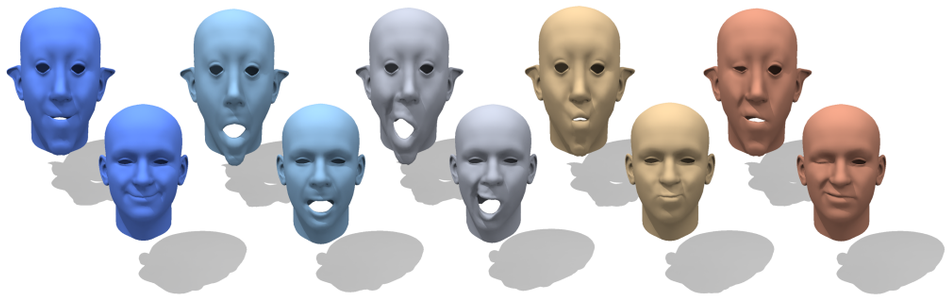
Data often come as (sampling of) metric spaces or sets/spaces endowed with a similarity measure with, possibly complex, topological/geometric structure.

Data carrying geometric information is usually high dimensional.

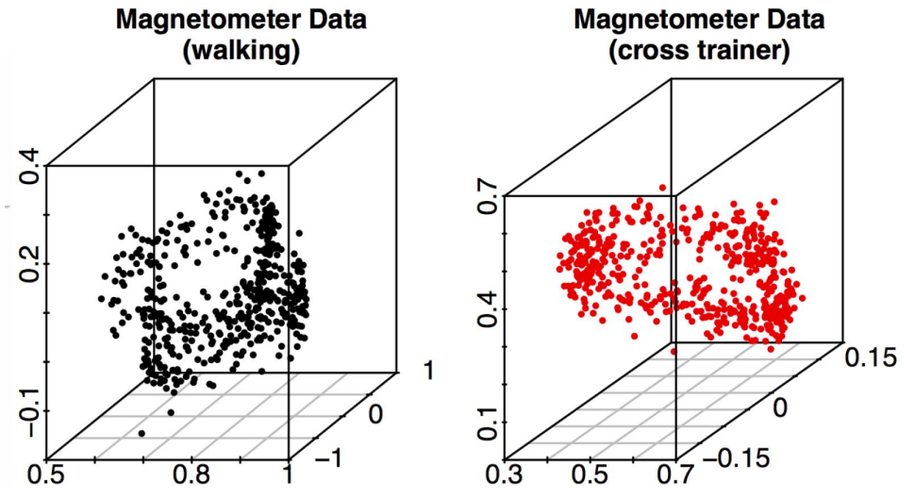
Introduction: topological descriptors built from data



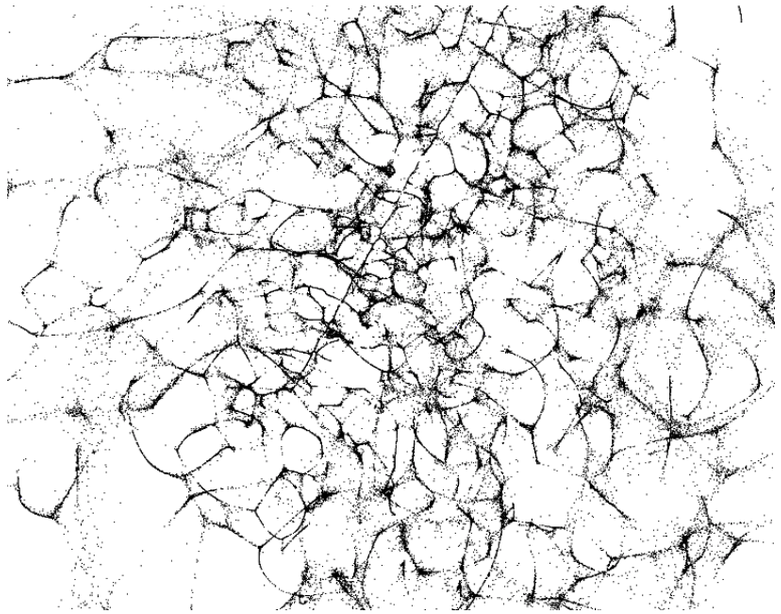
Scans



3D shapes



Magnetometer

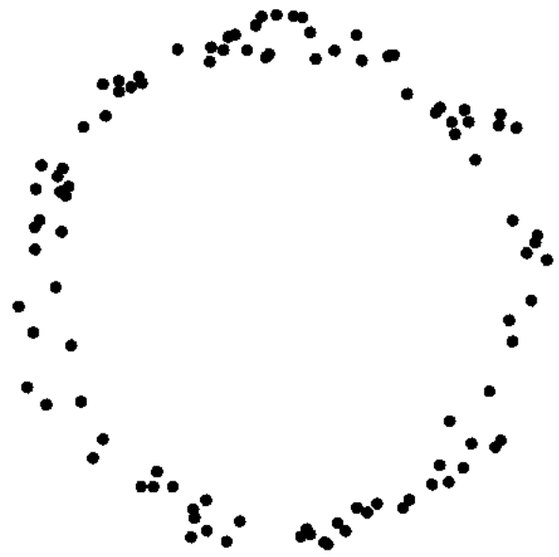


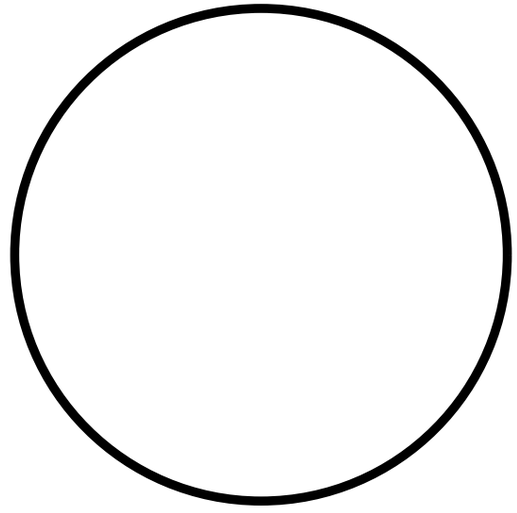
Galaxies

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

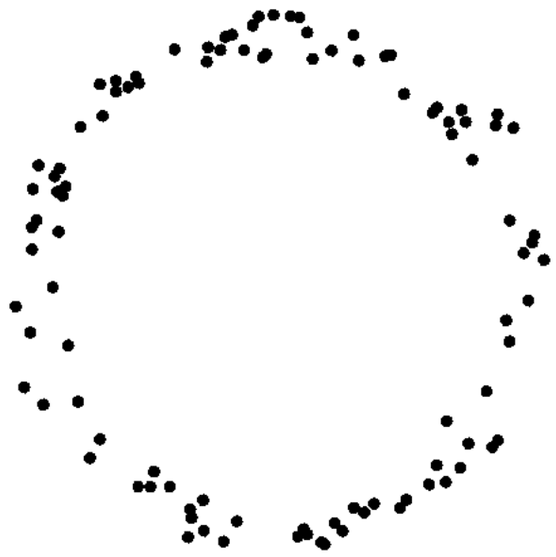
Problem: how to define the *topology* of a data set?

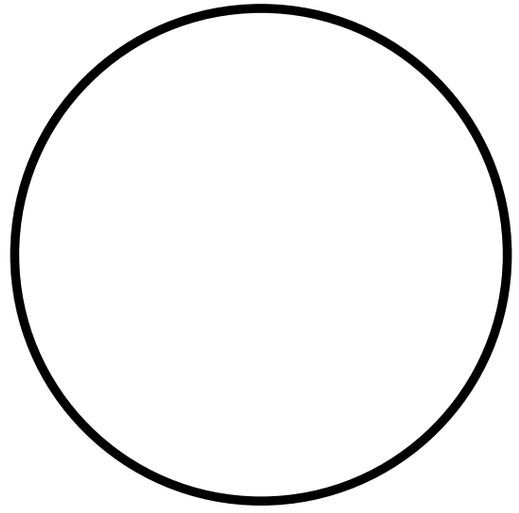




Challenges and advantages

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





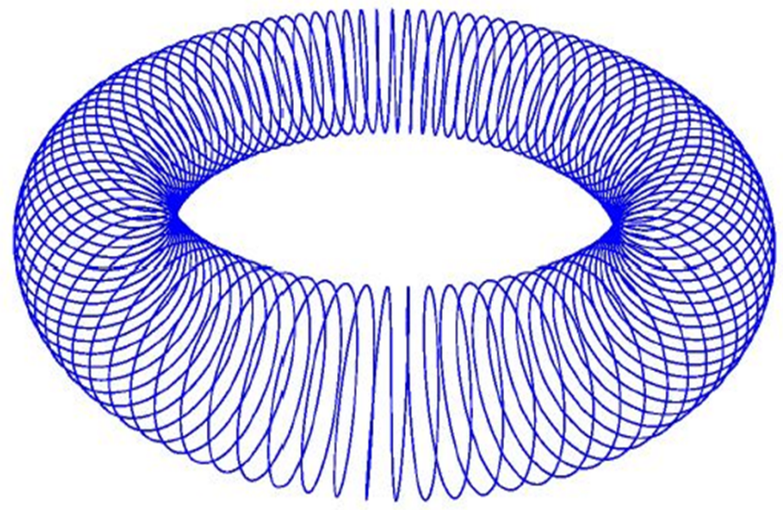
Challenges and goals:

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

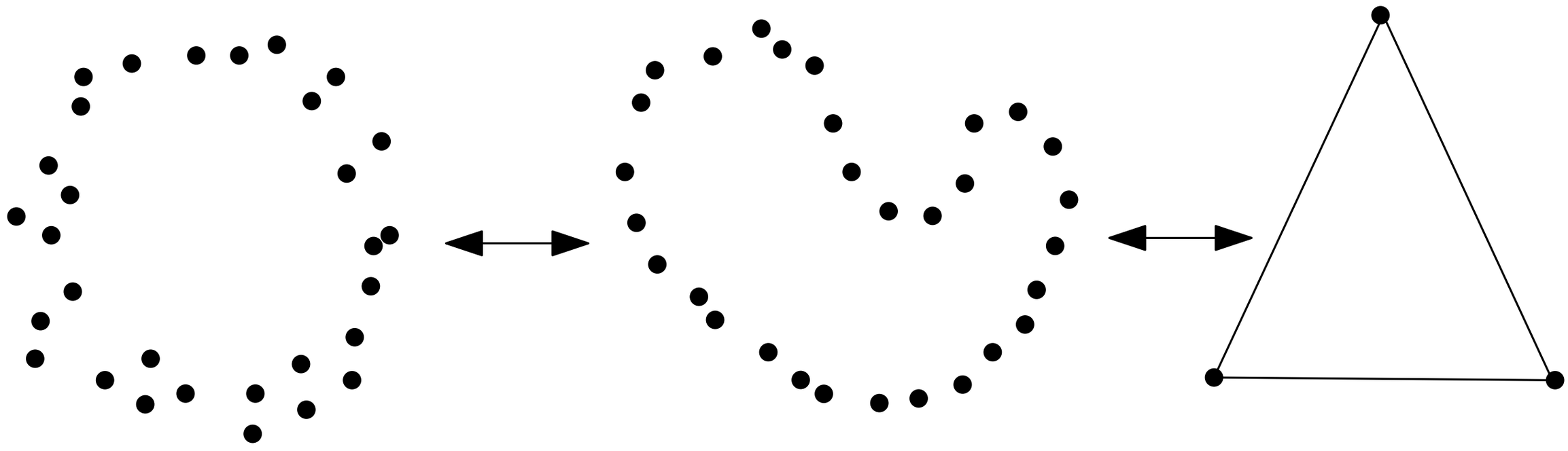
$\rightarrow$ distinguish topological "signal" from noise;

$\rightarrow$ 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:

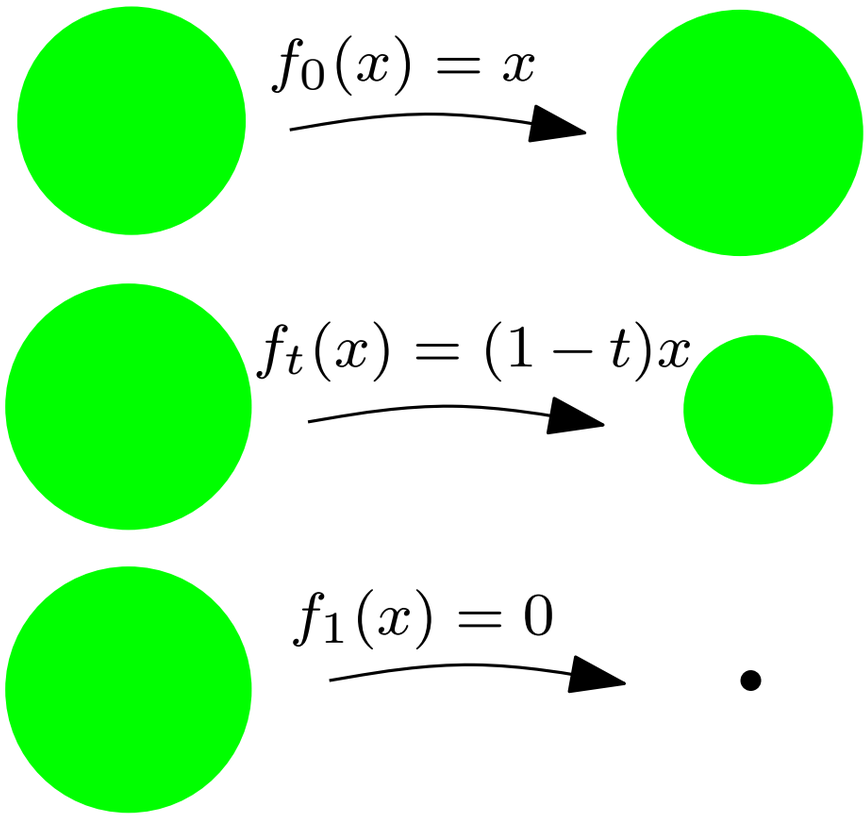
- Two maps $f_0 : X \to Y$ and $f_1 : X \to Y$ are *homotopic* if $\exists$ a continuous map $F : [0,1] \times X \to Y$ s.t. $\forall x \in X$, $F(0,x) = f_0(x)$ and $F_1(1,x) = f_1(x)$. X and Y are homotopy equivalent if $\exists$ continuous maps $f : X \to Y$ and $g : Y \to X$ s.t. $g \circ f$ is homotopic 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.

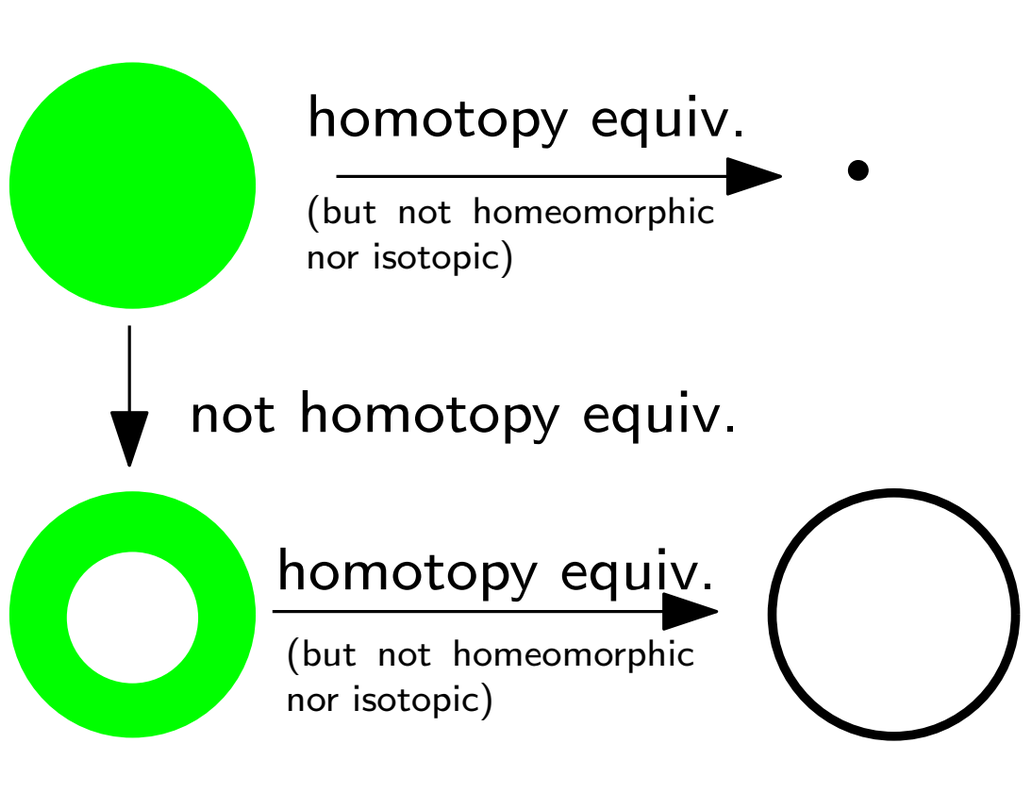
Q: Which notion is stronger/weaker?

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?

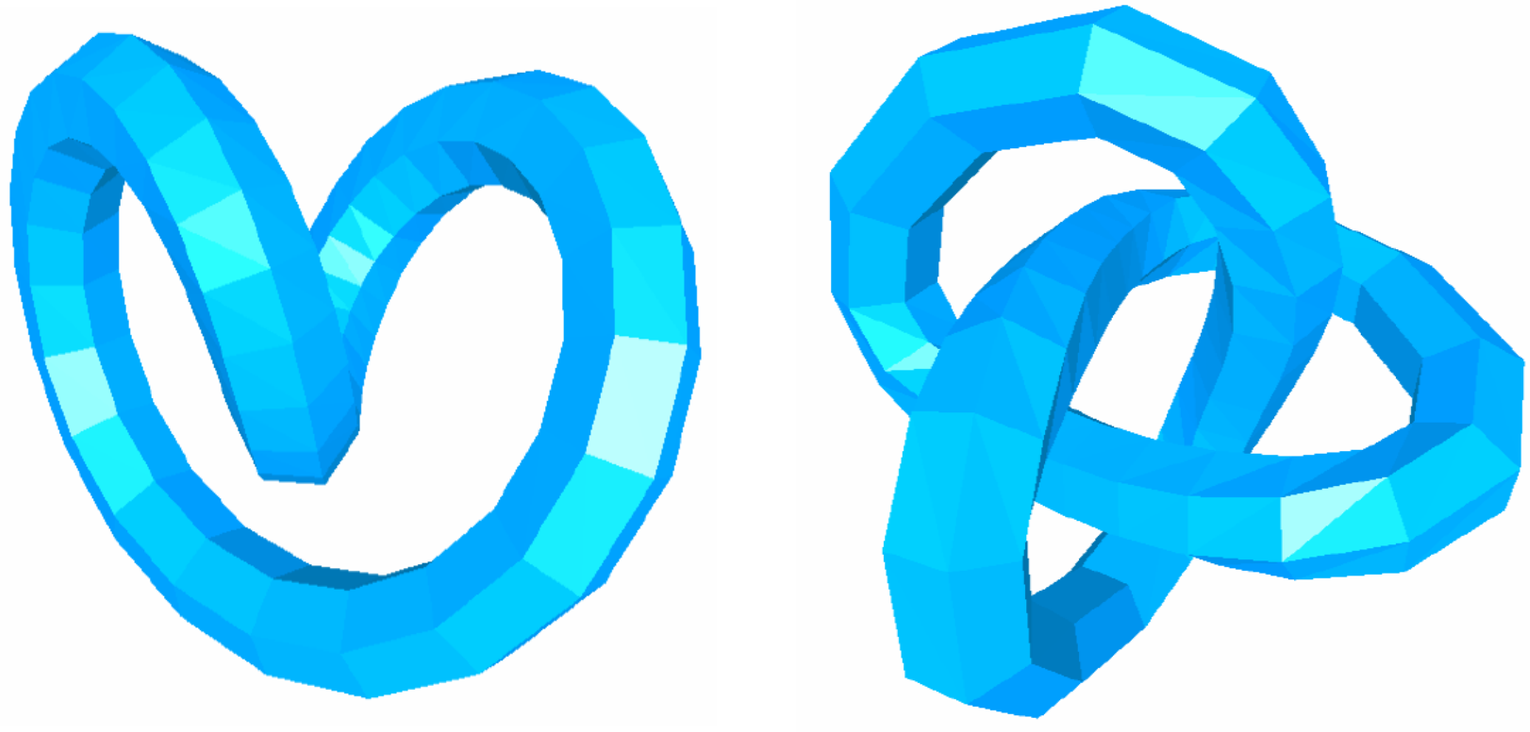
A brief look at topology

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?

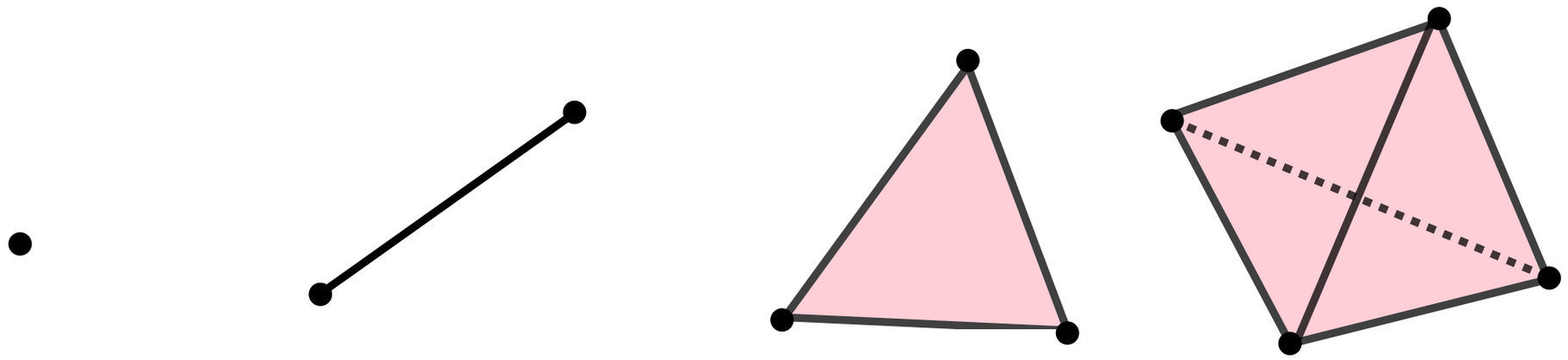
A topological space fit for computation

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

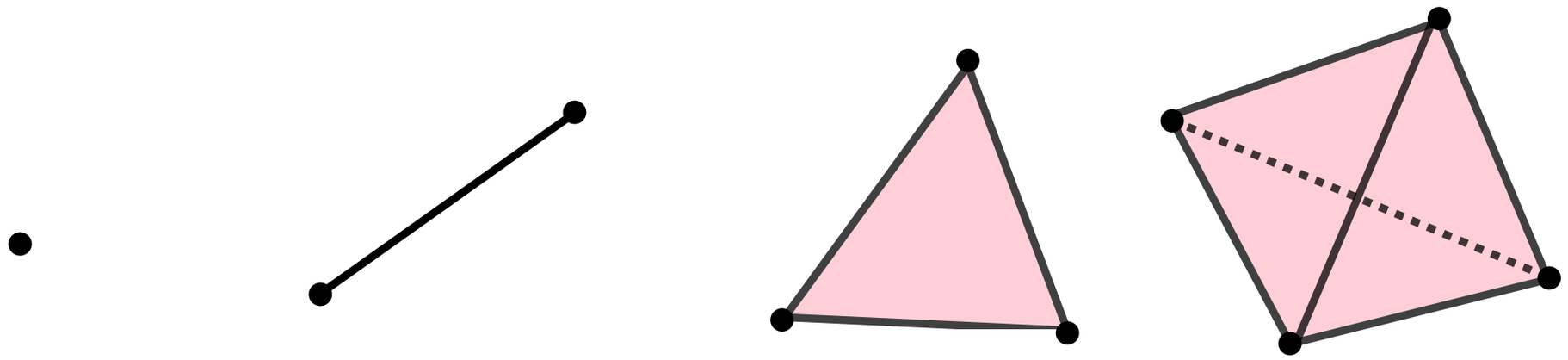
1-simplex:
edge

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.

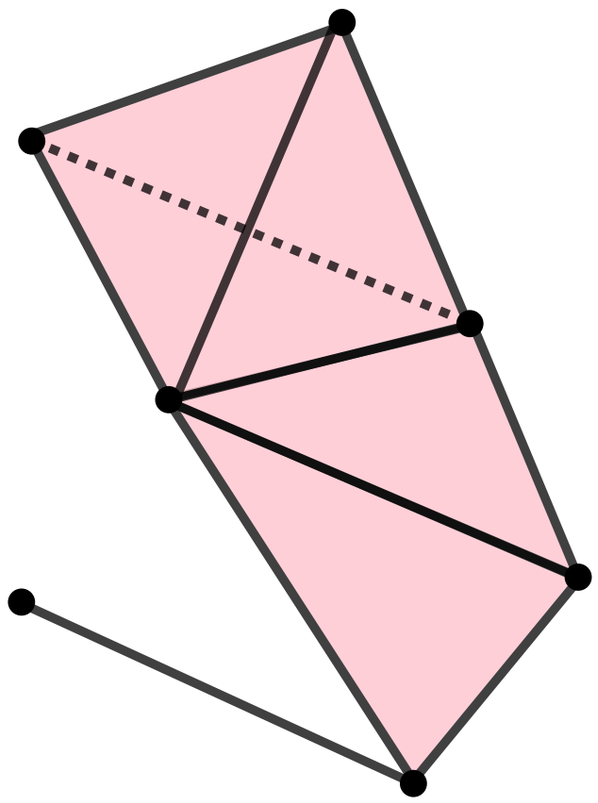
Simplex and simplicial complex

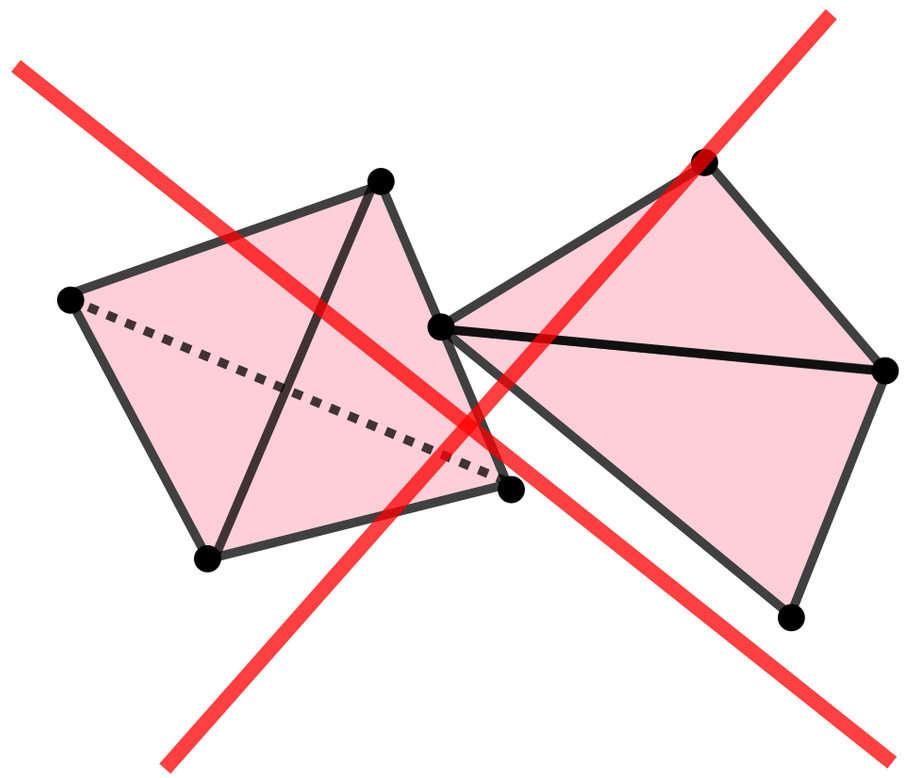
Def: A simplicial complex K in $\mathbb{R}^d$ is a collection of simplices s.t.:

- *(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.

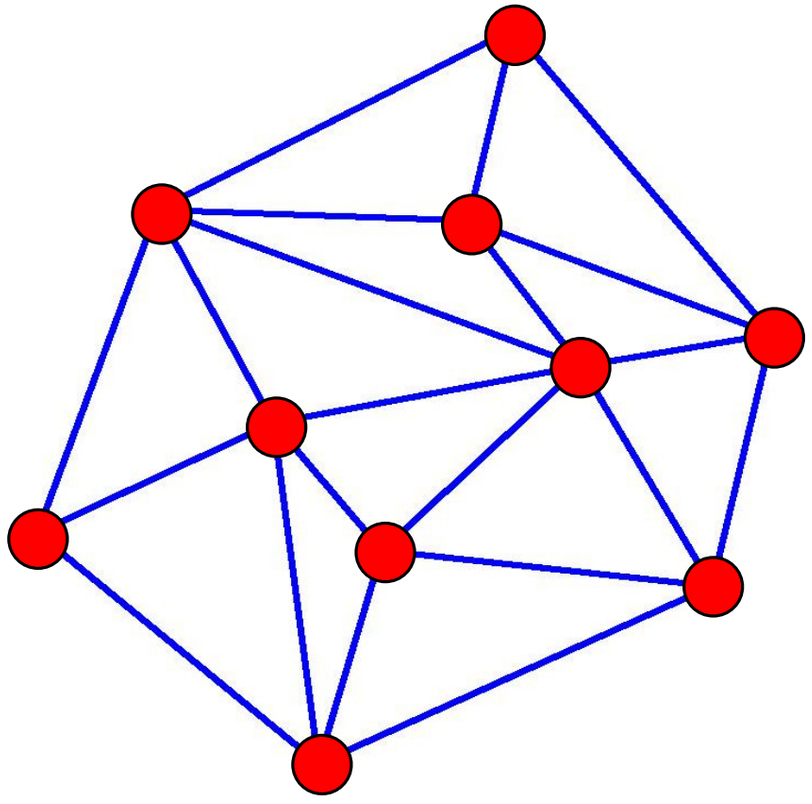
Triangulations

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 $|K| = \mathrm{conv}(P)$.

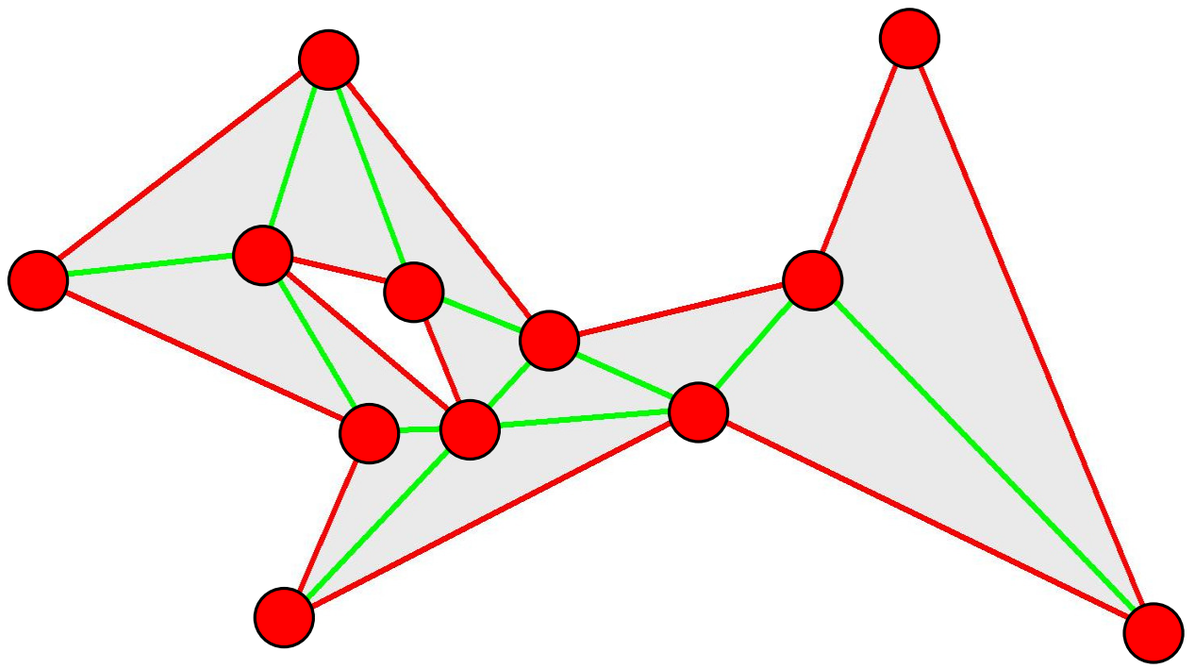


Triangulations

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 $|K| = \Omega$.



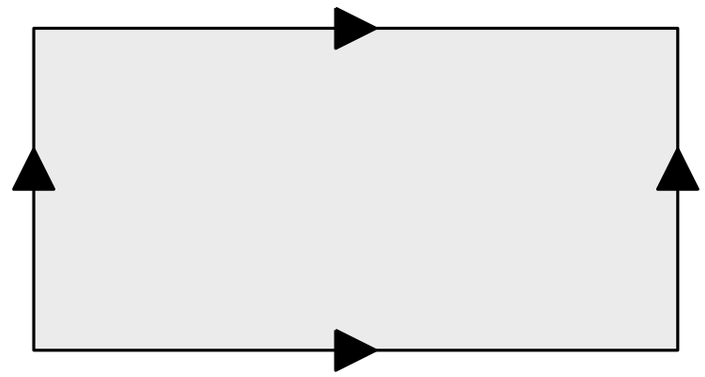
Triangulations

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

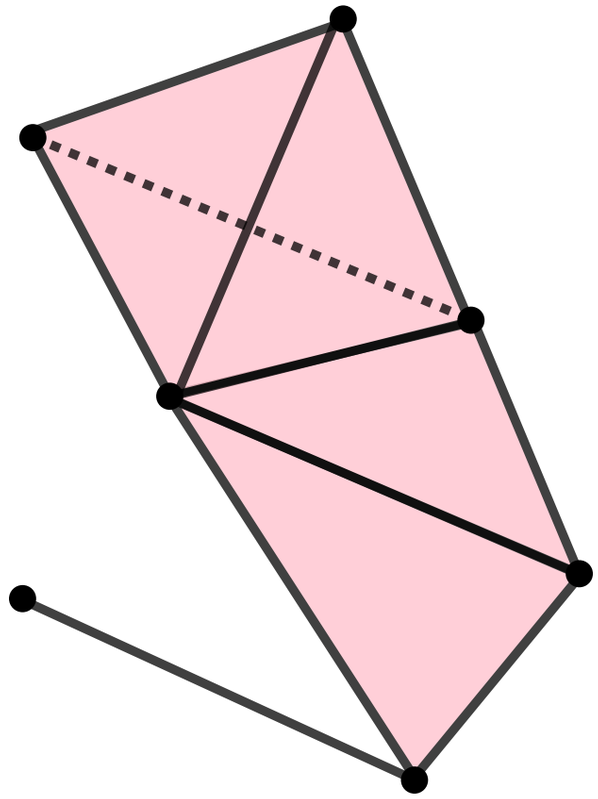


Abstract simplex and simplicial complex

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.

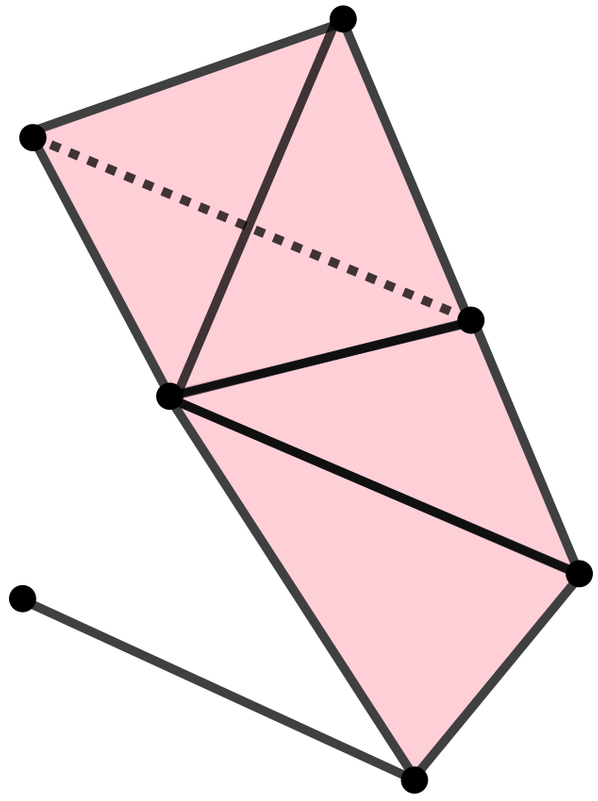


Abstract simplex and simplicial complex

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 $\mathbb{R}^n$.

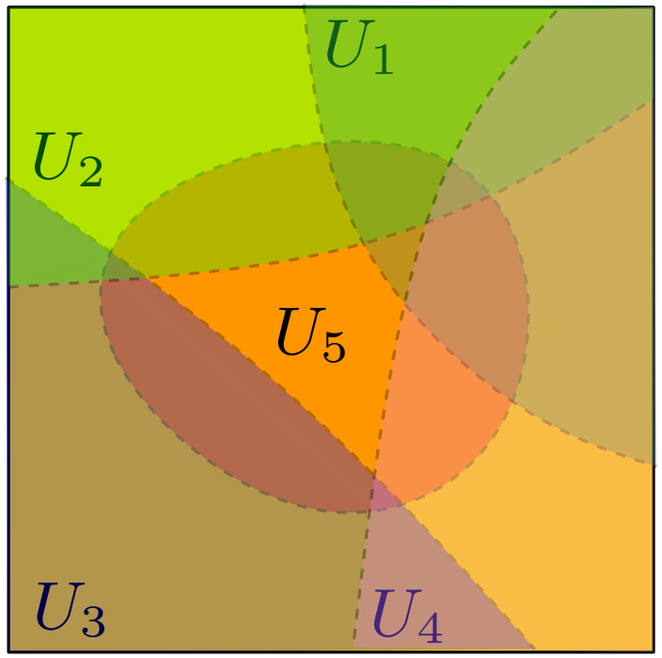
Q: Prove it.

Abstract simplicial complexes and their realizations are *homeomorphic*.

Nerve complex

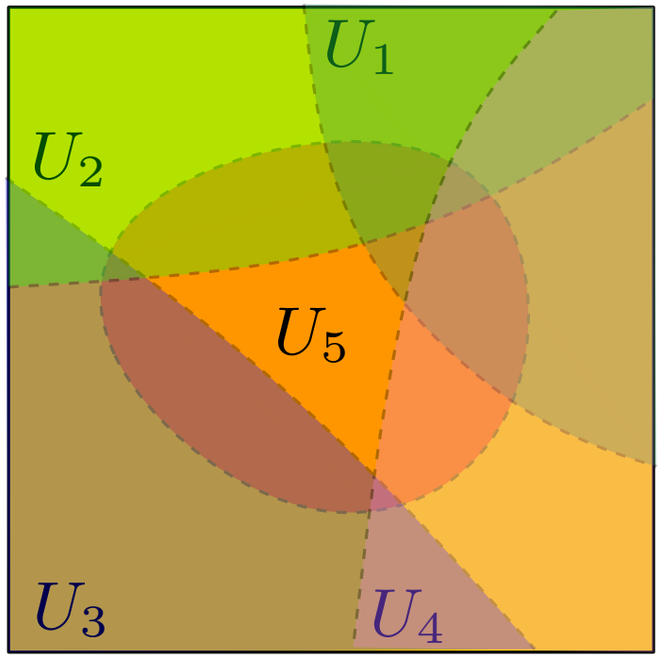
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

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

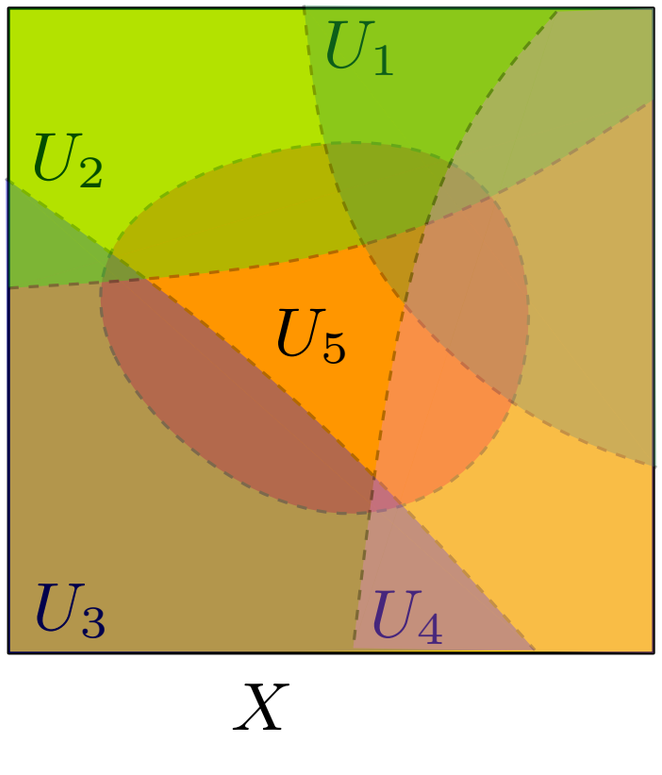


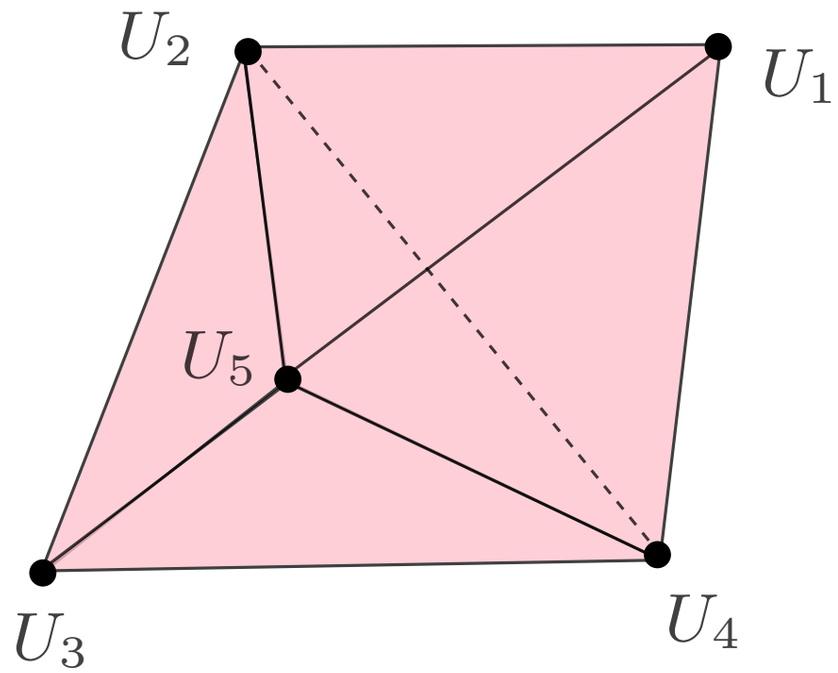
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

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





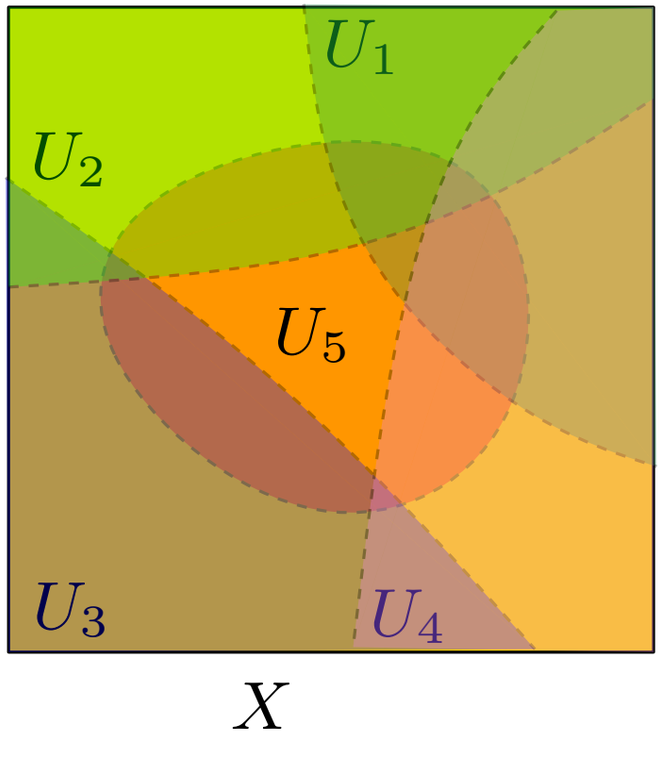
Def: An open cover of a topological space X is a collection $\mathcal{U} = (U_i)_{i \in I}$ of open subsets $U_i \subseteq X$, $i \in I$ where I is a set, such that $X \subseteq \cup_{i \in I} U_i$.

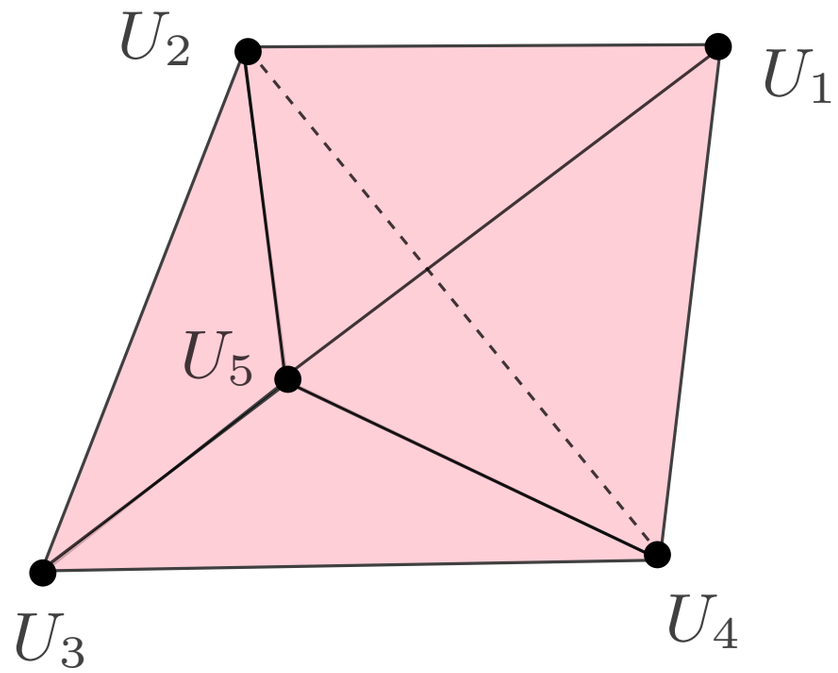
Def: Given a cover of a topological space X, $\mathcal{U} = (U_i)_{i \in I}$, its nerve is the abstract simplicial complex $C(\mathcal{U})$ whose vertex set is $\mathcal{U}$ and s.t.

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

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

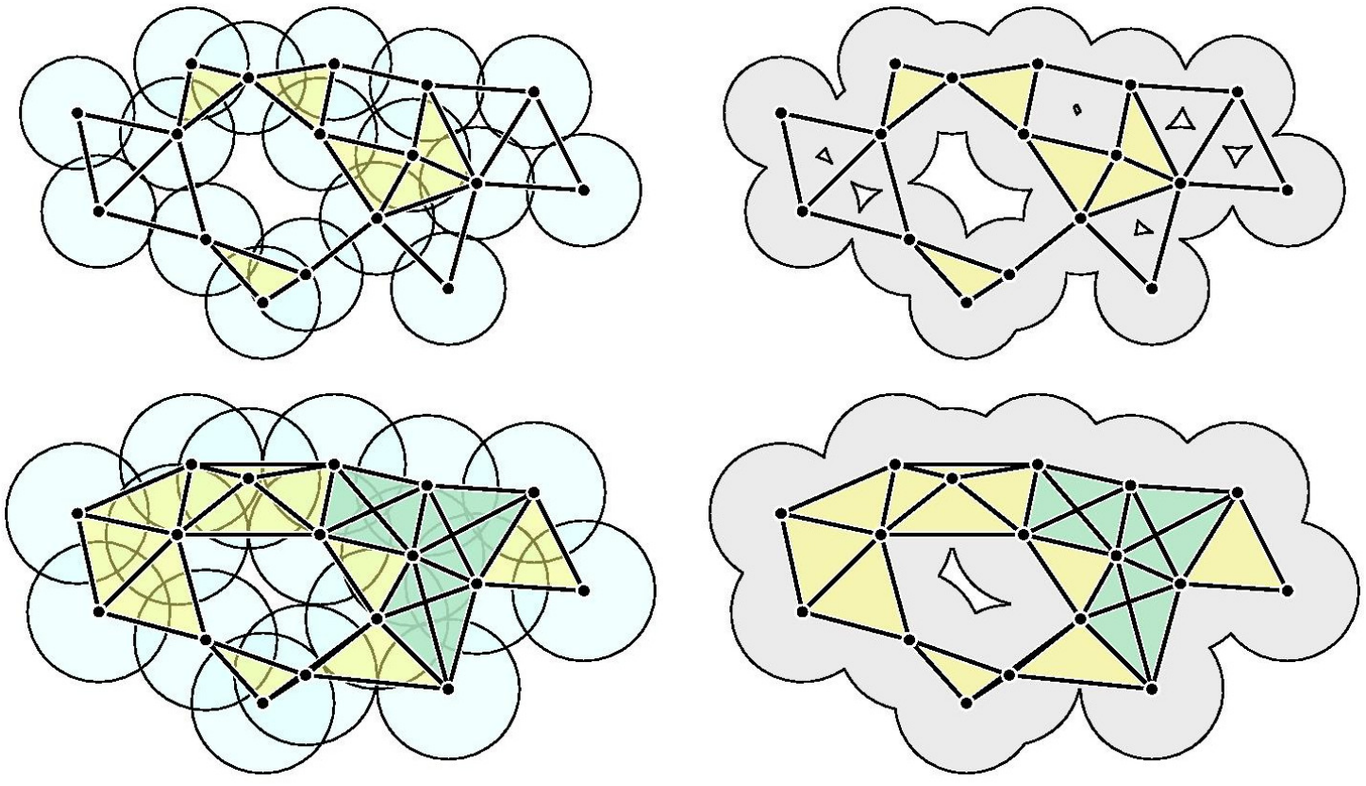
$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in C(P, r) \quad \text{iif} \quad \cap_{j=0}^{k} B(P_{i_j}, r) \neq \emptyset.$$

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.

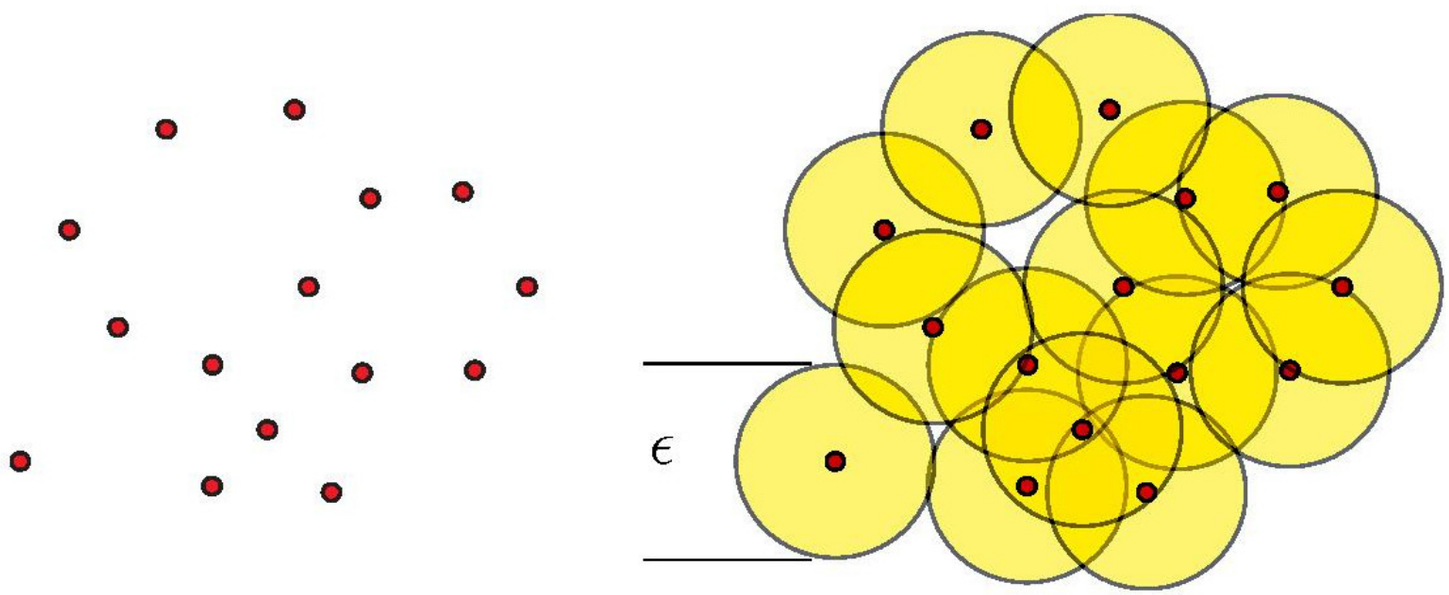
Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Rips complex of radius $r > 0$ is the abstract simplicial complex $R(P, r)$ s.t. $\mathrm{vert}(R(P, r)) = P$ and

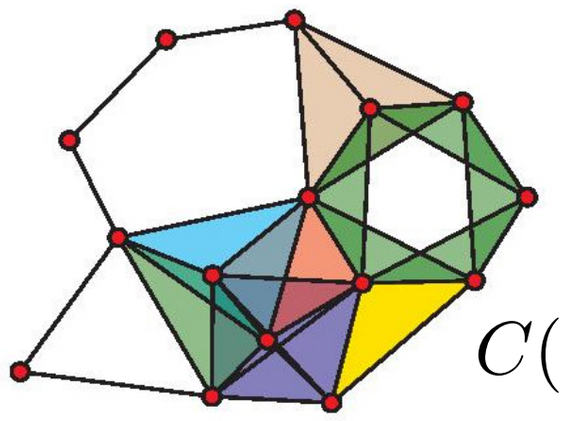
$$\sigma = [P_{i_0}, P_{i_1}, \ldots, P_{i_k}] \in R(P, r) \text{ iif } \|P_{i_j} - P_{i_{j'}}\| \leq 2r, \forall 1 \leq j, j' \leq k.$$

Čech and (Vietoris)-Rips complexes

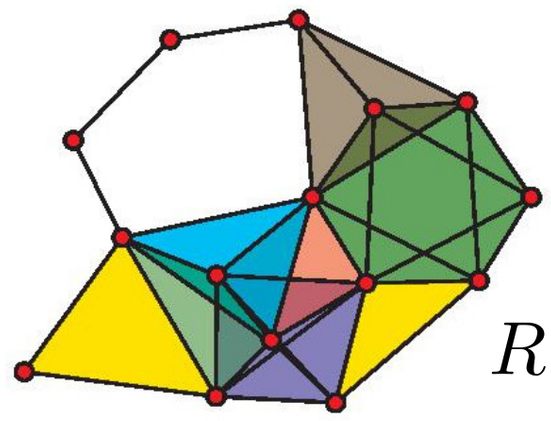
Def: Given a point cloud $P = \{P_1, \ldots, P_n\} \subset \mathbb{R}^d$, its Čech complex of radius $r > 0$ is the abstract simplicial complex $C(P, r)$ s.t. $\text{vert}(C(P, r)) = P$ and

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





$C(P, \epsilon)$



$R(P, \epsilon)$

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]

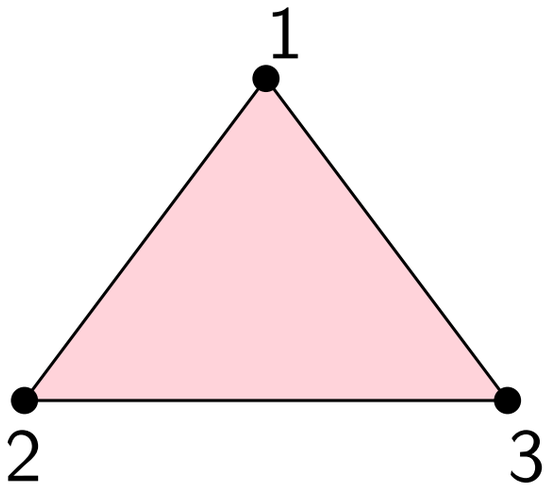
We want to store simplicial complexes with a data structure that allows to perform standard operations (insertion of a simplex, checking if a simplex is present, etc) in a fast and easy way.

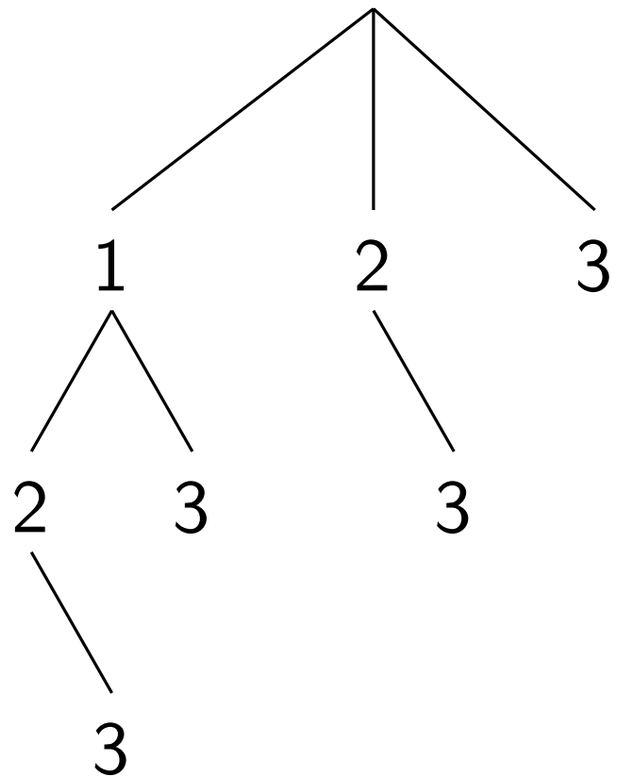
Storing simplicial complexes

[*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)*.



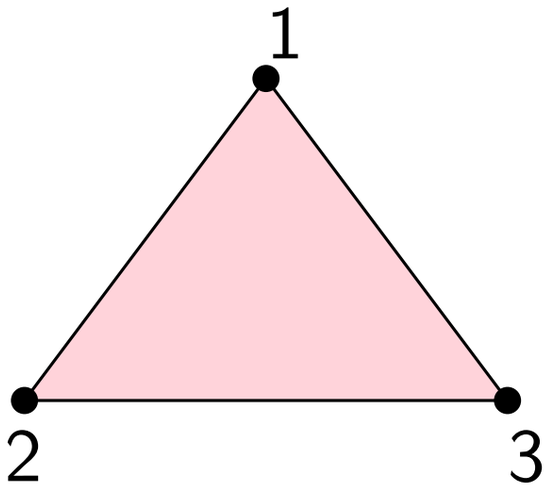


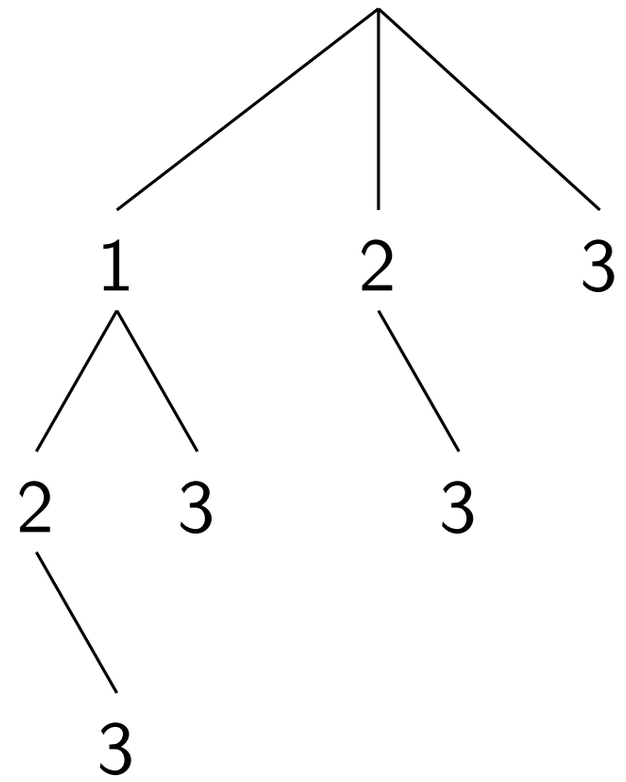
Storing simplicial complexes

[*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*.

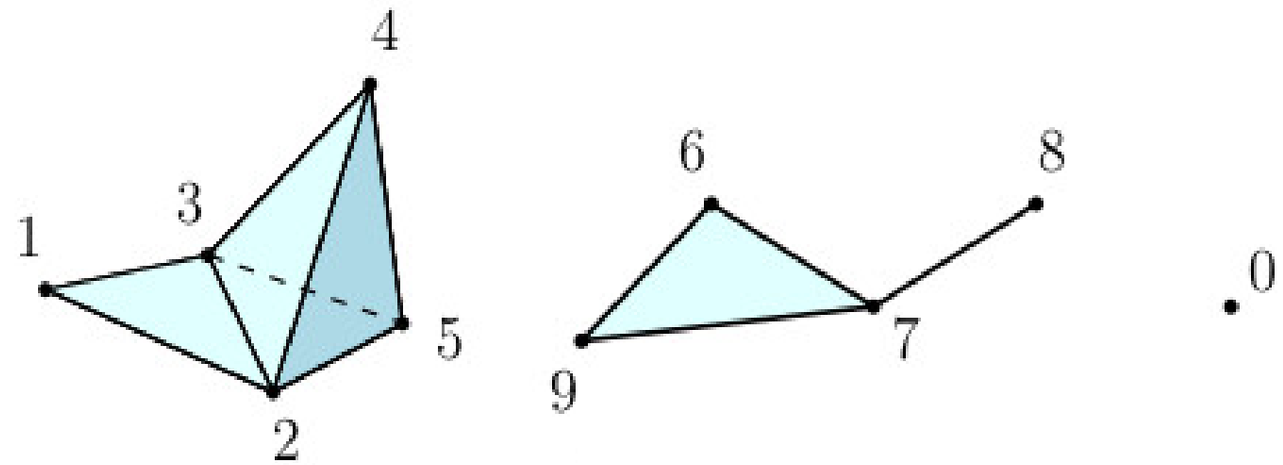
It allows to store all simplices explicitly without storing all adjacency relations, while maintaining low complexity for basic operations.

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

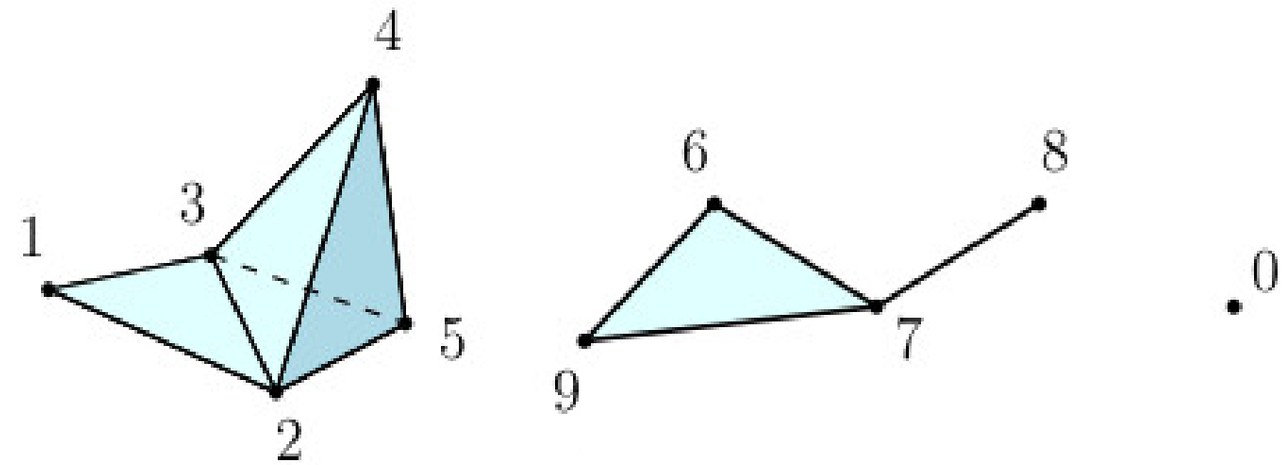


Storing simplicial complexes

[*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

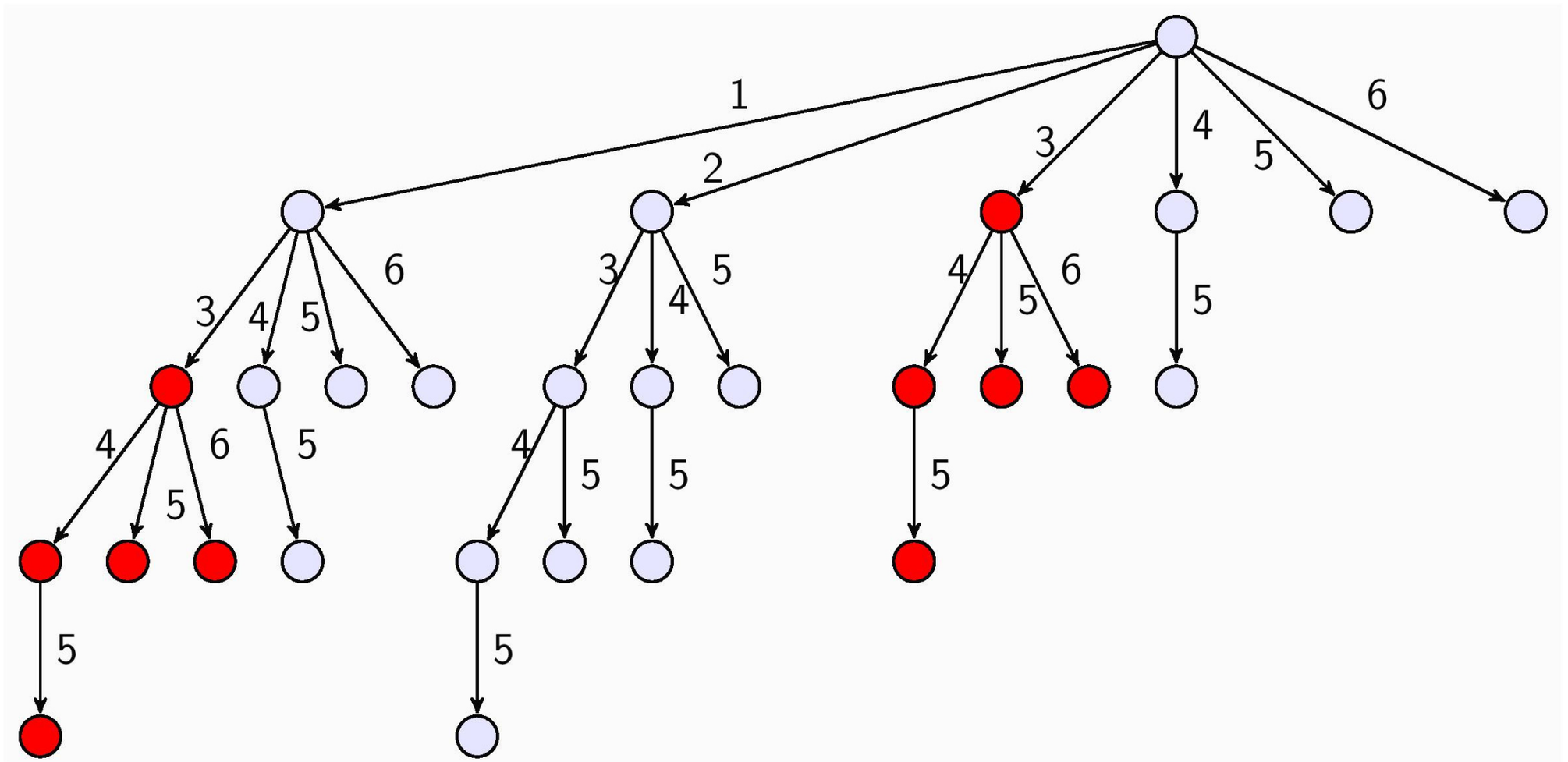
Depth of simplex tree = 1 + dimension of complex

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.