

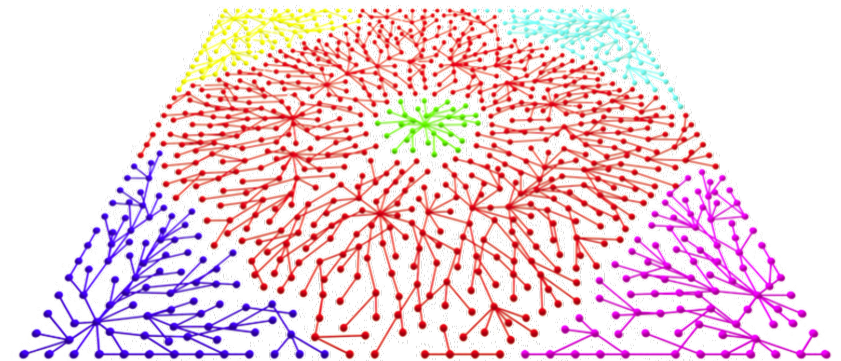
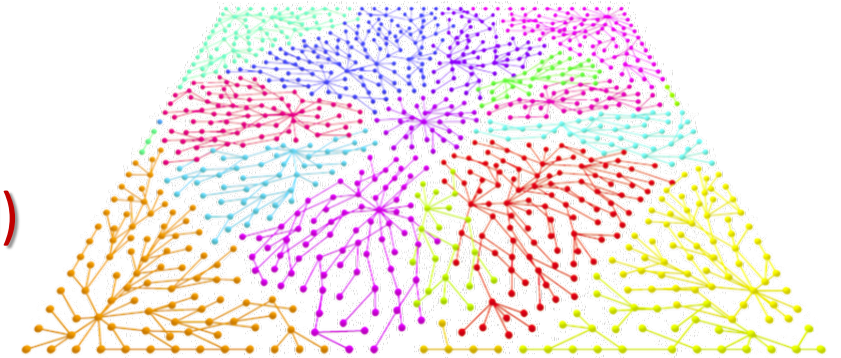
CS233, CME251: Geometric and Topological Data Analysis

Leonidas Guibas
Computer Science Department
Stanford University

Guest lecture: Samir Chowdhury (Stanford)



Lecture 8
21 April 2021

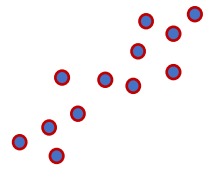


Last Time: Introduction to Computational Topology

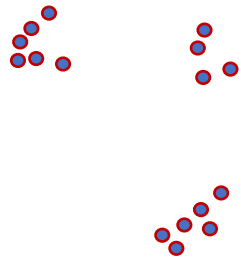
Topological Data Analysis (TDA)

The “Shape of Data”

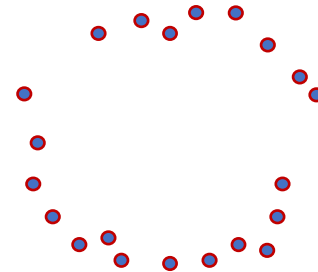
- TDA Studies Sampled Spaces



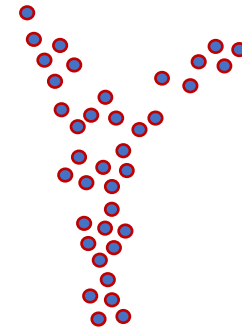
Regression



Cluster



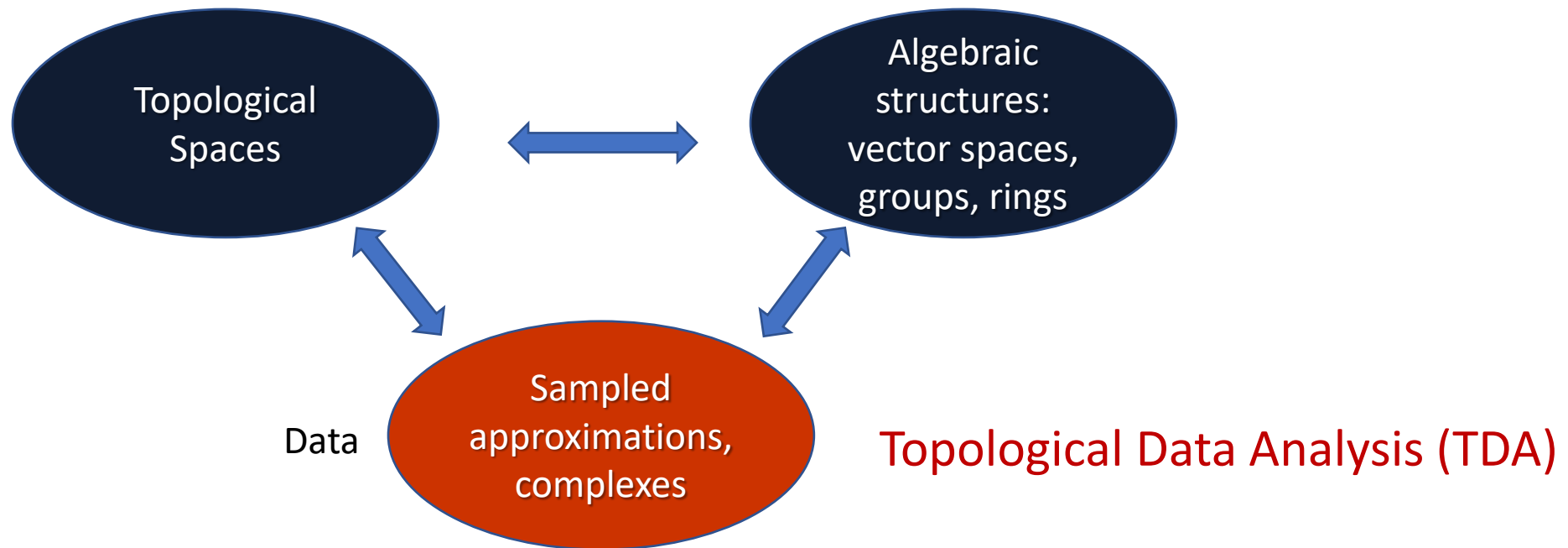
Loop



Flared

Algebraic Topology and Topology Inference

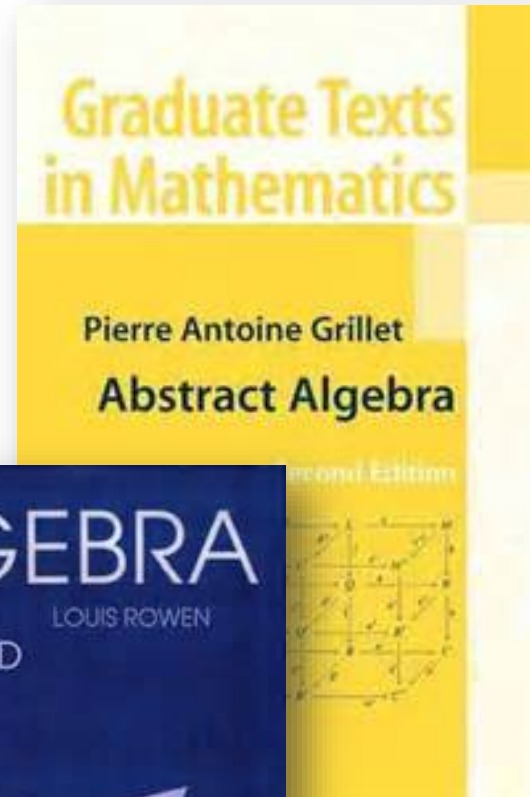
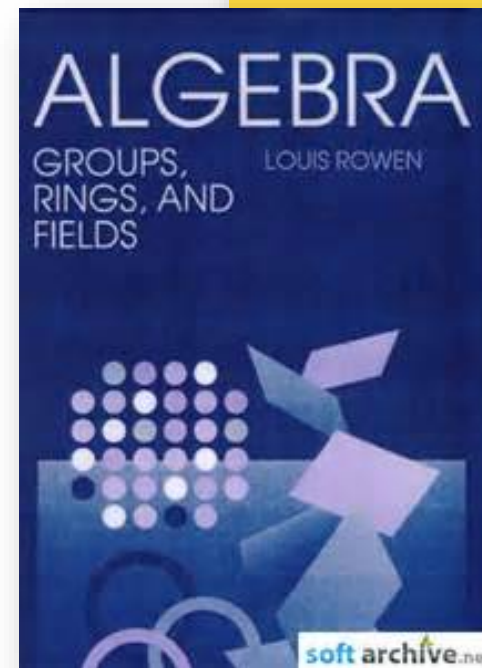
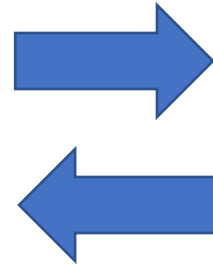
- Unlike geometry, topology studies the **global** structure of spaces
- Getting to this structure via **algebra** algebraic topology



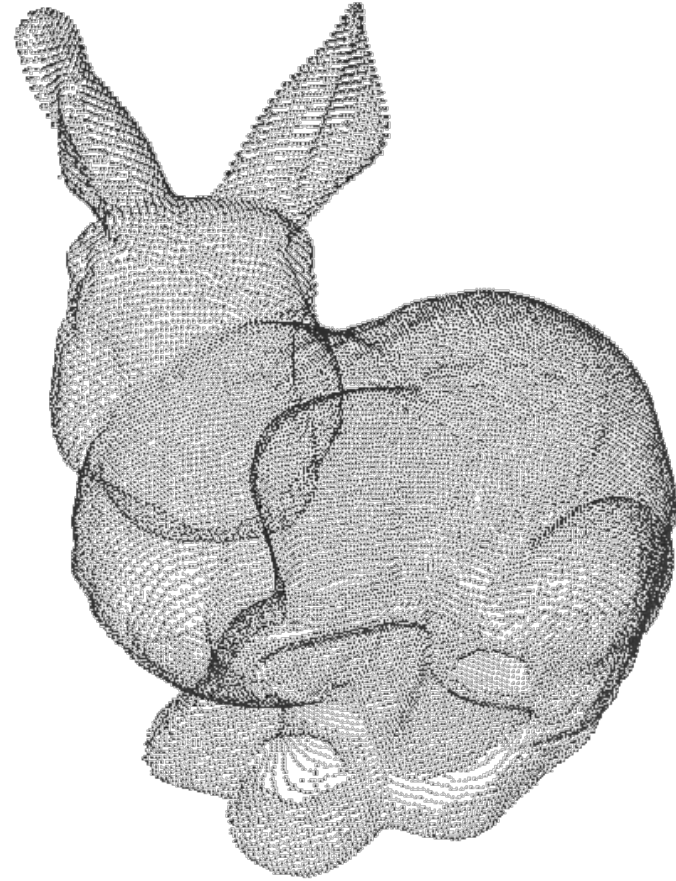
From Data to Algebraic Objects



Algebraic entities as
data descriptors



Point Clouds Have No Higher-D Topology

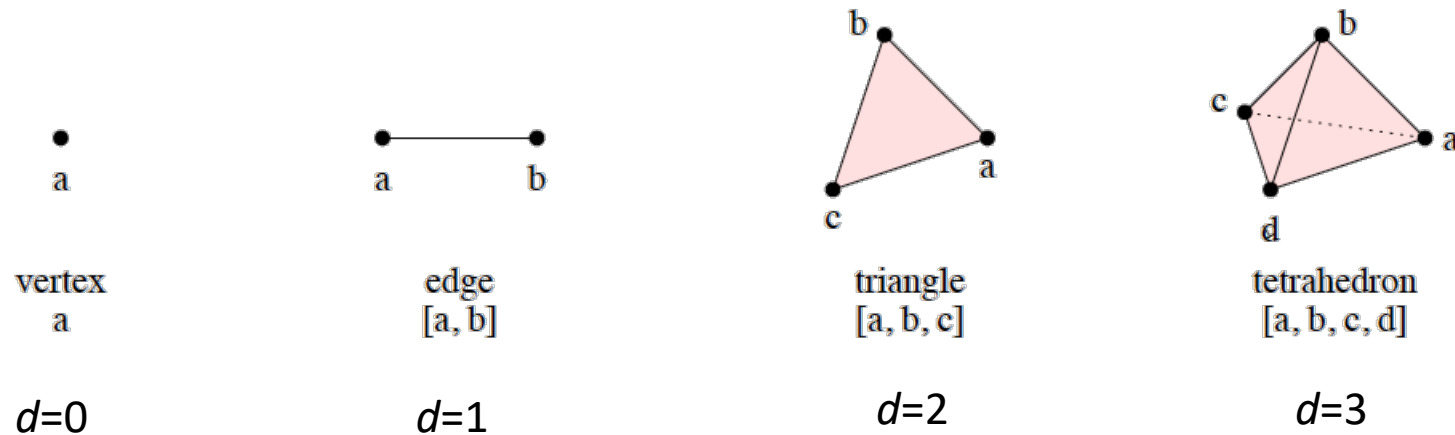


Connect the points into simplicial complexes

Simplicial homology

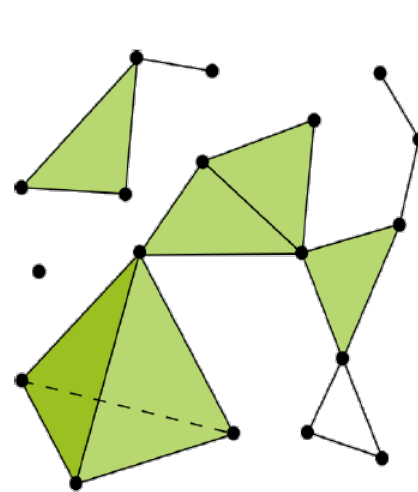
A Simplex

- A **k -simplex** is the convex hull of $k + 1$ affinely independent points $S = \{v_0, v_1, \dots, v_k\}$. The points in S are the **vertices** of the simplex.
- A k -simplex is a k -dimensional subspace of \mathbb{R}^d , $\dim \sigma = k$.

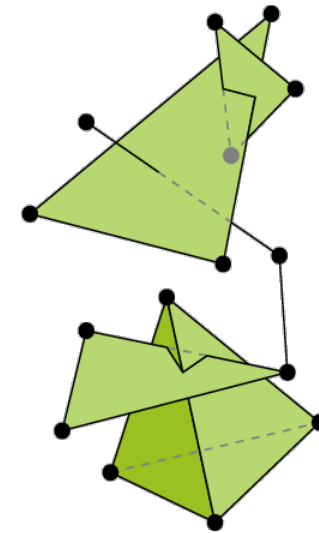


Simplicial Complexes

- A **simplicial complex** K is a finite set of simplices such that
 1. $\sigma \in K, \tau \leq \sigma \Rightarrow \tau \in K$,
 2. $\sigma, \sigma' \in K \Rightarrow \sigma \cap \sigma' \leq \sigma, \sigma'$ or $\sigma \cap \sigma' = \emptyset$.
- The **dimension** of K is $\dim K = \max\{\dim \sigma \mid \sigma \in K\}$.
- The **vertices** of K are the zero-simplices in K .
- A simplex is **principal** if it has no proper coface in K .

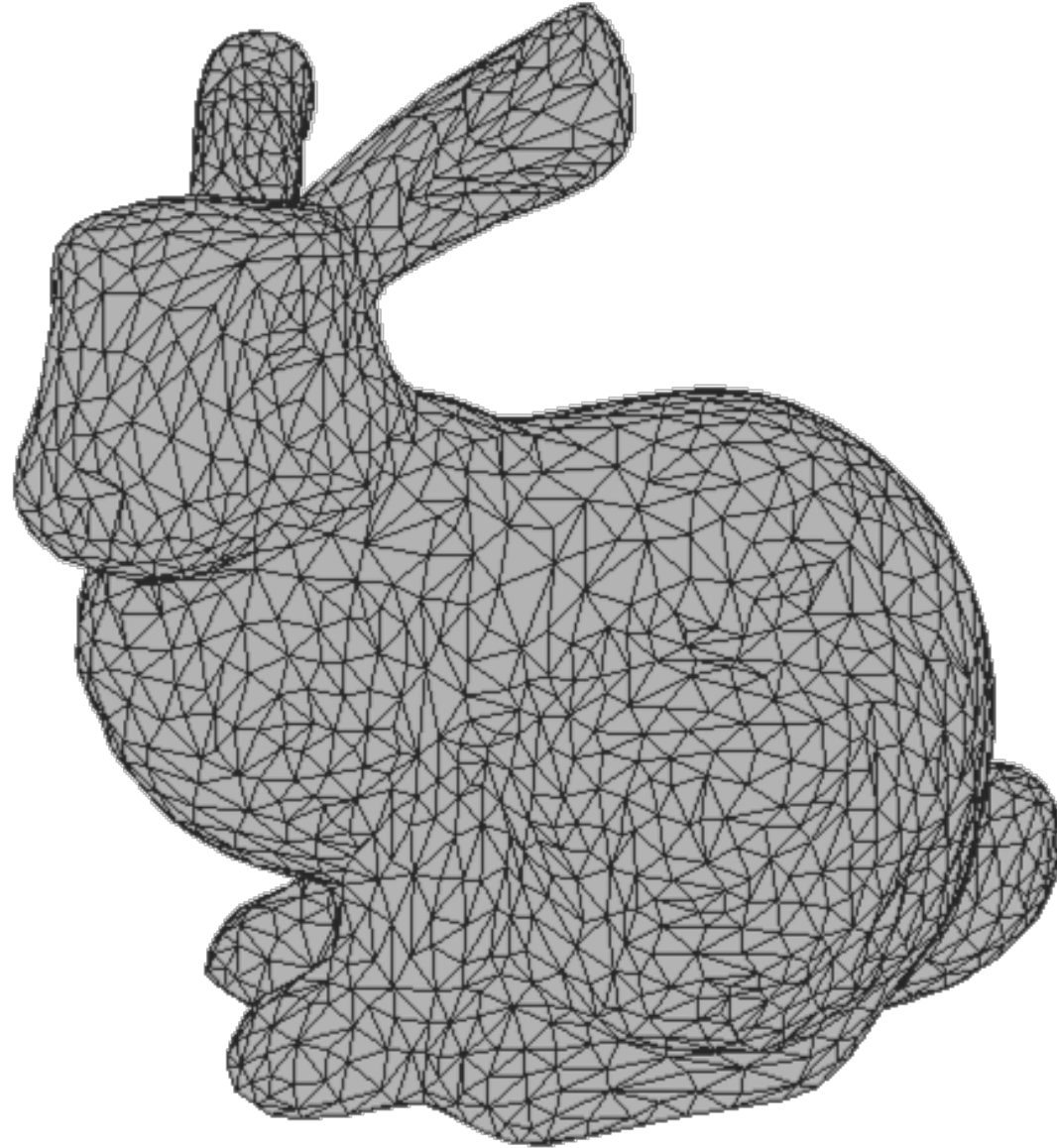


(left) an example

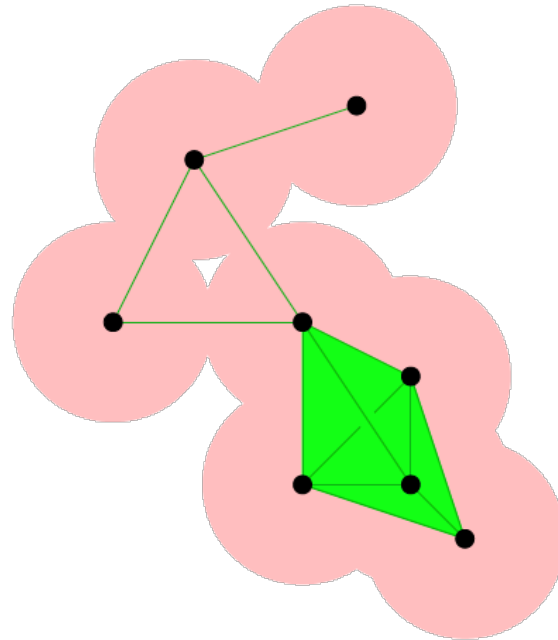


(right) a non example

Meshes are 2-D Complexes

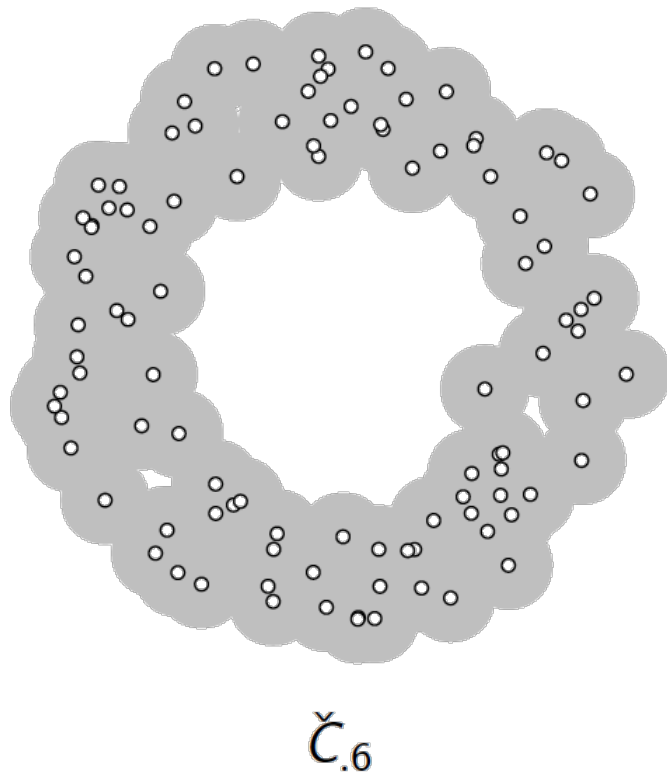


Čech Complex



- $C_\epsilon(M) = \{\text{conv} T \mid T \subseteq M, \bigcap_{m_i \in T} B_\epsilon(m_i) \neq \emptyset\}.$
- $\sum_{k=0}^m \binom{m}{k} = 2^{m+1} - 1$
- $C_\epsilon(M) \simeq \tilde{M}$

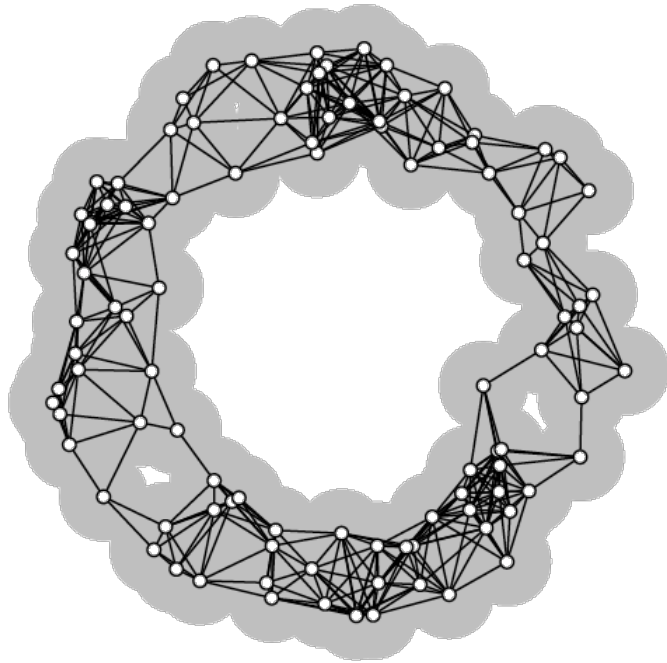
Čech Complex



The Čech complex \check{C}_ϵ encodes the intersection pattern of M_ϵ : Encode:

Points as *vertices*
(0-cells)

Čech Complex



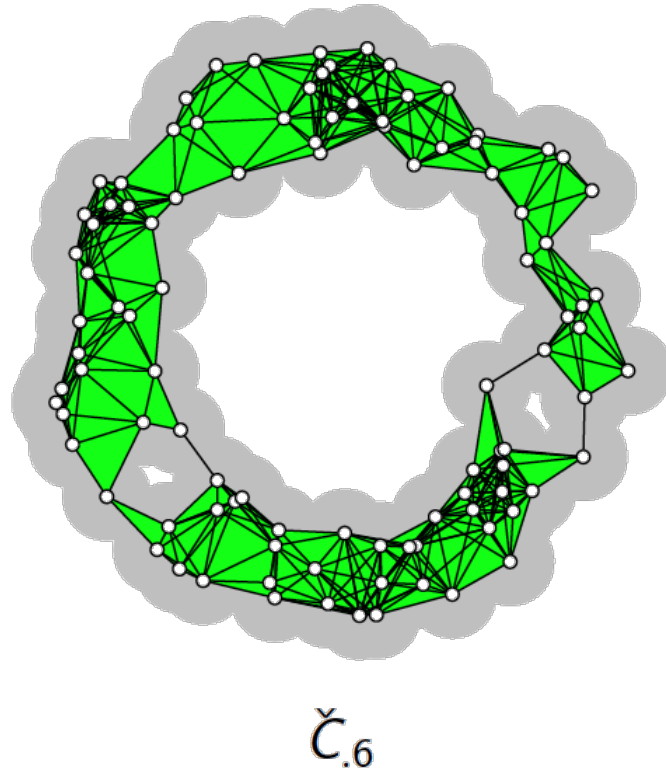
$\check{C}_{\epsilon,6}$

The Čech complex \check{C}_{ϵ} encodes the intersection pattern of M_{ϵ} : Encode:

Points as *vertices*
(0-cells)

Pairwise intersections
as *edges* (1-cells)

Čech Complex



The Čech complex \check{C}_ϵ encodes the intersection pattern of M_ϵ : Encode:

Points as *vertices*
(0-cells)

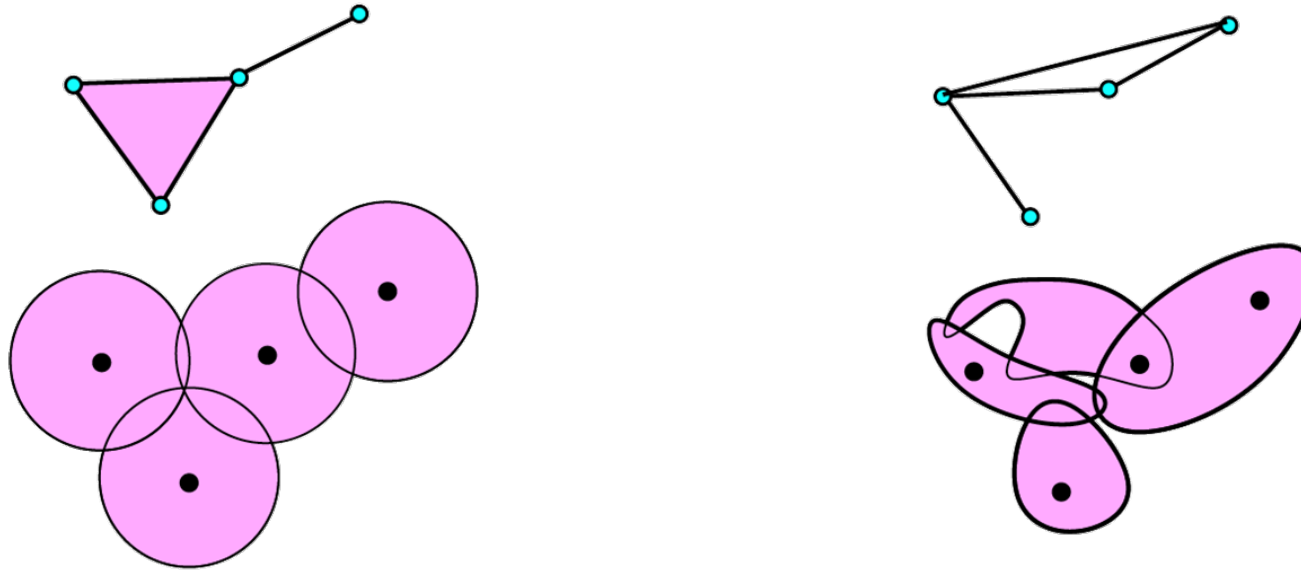
Pairwise intersections
as *edges* (1-cells)

Threeway intersections
as *triangles* (2-cells)

k -way intersections as
($k+1$)-cells

Can be hard to compute ...

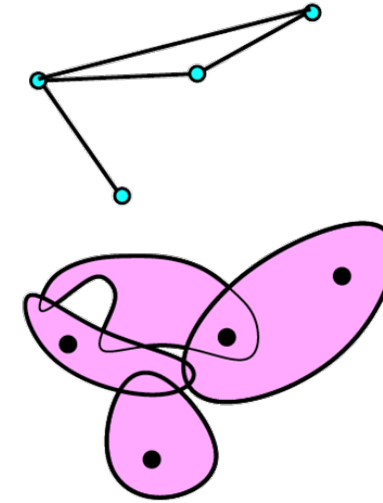
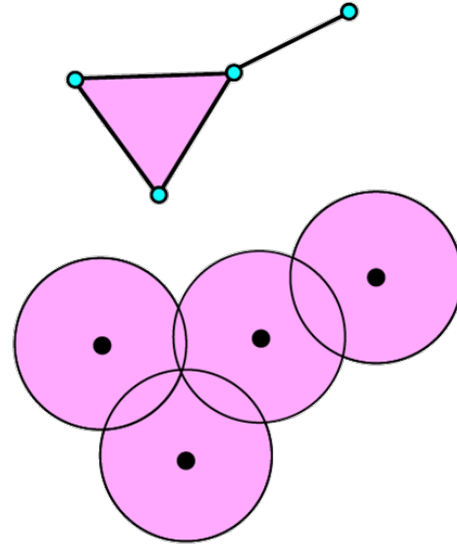
General Čech Complex = Nerve



- Let $\mathcal{U} = (U_i)_{i \in I}$ be a covering of a topological space X by open sets:
 $X = \cup_{i \in I} U_i$.
- The Čech complex $C(\mathcal{U})$ associated to the covering \mathcal{U} is the simplicial complex defined by:
 - the vertex set of $C(\mathcal{U})$ is the set of the open sets U_i
 - $[U_{i_0}, \dots, U_{i_k}]$ is a k -simplex in $C(\mathcal{U})$ iff $\cap_{j=0}^k U_{i_j} \neq \emptyset$.

General Čech Complex

Lemma (Nerve Lemma, Leray '45)
 \check{C}_ϵ is topologically equivalent to M_ϵ .



Nerve theorem (Leray): If all the intersections between opens in \mathcal{U} are either empty or contractible then $C(\mathcal{U})$ and $X = \cup_{i \in I} U_i$ are homotopy equivalent.

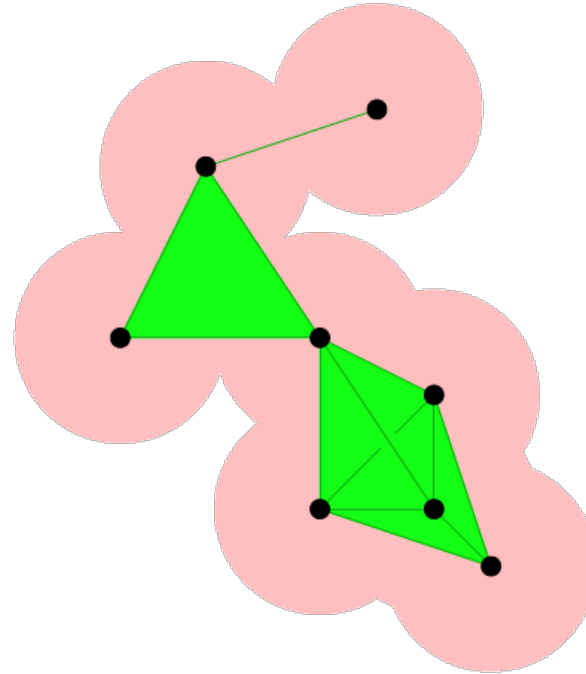
\Rightarrow The combinatorics of the covering (a simplicial complex) carries the topology of the space.

Warning: even when the open sets are euclidean balls, the computation of the Čech complex is a very difficult task!

Rips-Vietoris Complex

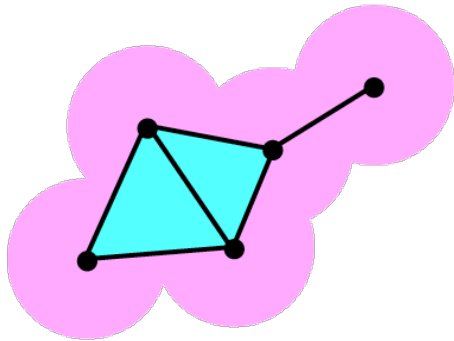
The “poor man’s” alternative
to the Čech

This is a common complex
For computations

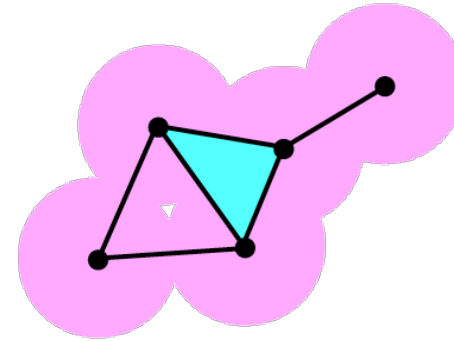


- $R_\epsilon(M) = \{\text{conv } T \mid T \subseteq M, d(m_i, m_j) < \epsilon, m_i, m_j \in T\}.$
- Still $O\left(\binom{m}{k}\right)$ for the k th skeleton
- Need $(k + 1)$ st skeleton for computing H_k

Rips vs. Čech



Rips vs Čech



Let $L = \{p_0, \dots, p_n\}$ be a (finite) point cloud (in a metric space).

The **Rips complex** $\mathcal{R}^\alpha(L)$: for $p_0, \dots, p_k \in L$,

$$\sigma = [p_0 p_1 \dots p_k] \in \mathcal{R}^\alpha(L) \text{ iff } \forall i, j \in \{0, \dots, k\}, d(p_i, p_j) \leq \alpha$$

- Easy to compute and fully determined by its 1-skeleton
- Rips-Čech interleaving: for any $\alpha > 0$,

$$\mathcal{C}^{\frac{\alpha}{2}}(L) \subseteq \mathcal{R}^\alpha(L) \subseteq \mathcal{C}^\alpha(L) \subseteq \mathcal{R}^{2\alpha}(L) \subseteq \dots$$

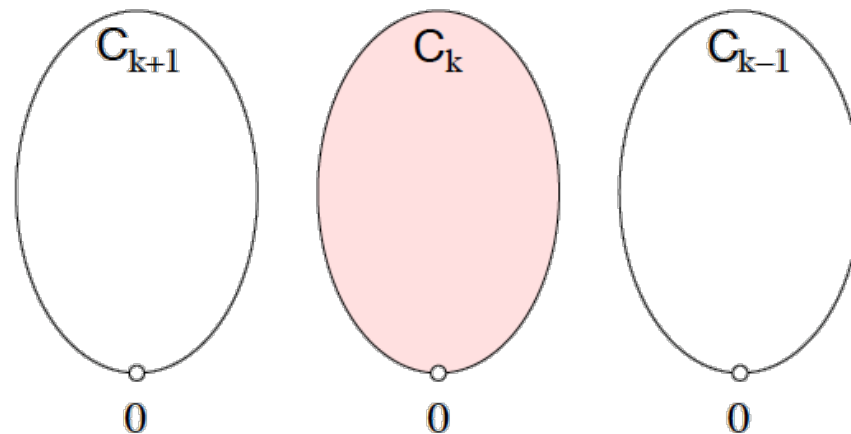
Groups

- A **group** $\langle G, * \rangle$ is a set G , together with a binary operation $*$ on G , such that the following axioms are satisfied:
 - (a) $*$ is associative.
 - (b) G has an **identity** e element for $*$ such that $e * x = x * e = x$ for all $x \in G$.
 - (c) any element a has an **inverse** a' with respect to the operation $*$, i.e. $\forall a \in G, \exists a' \in G$ such that $a' * a = a * a' = e$.
- If G is finite, the **order** of G is $|G|$.
- We often omit the operation and refer to G as the group.
- $\langle \mathbb{Z}, + \rangle, \langle \mathbb{R}, \cdot \rangle, \langle \mathbb{R}, + \rangle$, are all groups.
- A group G is **abelian** if its binary operation $*$ is commutative.

Chain Groups

Other coefficient
fields/rings also OK

- Simplicial complex K
- **k -chain**: $c = \sum_i n_i [\sigma_i]$, $n_i \in \mathbb{Z}$, $\sigma_i \in K$ (like a path)
- $[\sigma] = -[\tau]$ if $\sigma = \tau$ and σ and τ have different orientations.
- The **k th chain group \mathbf{C}_k** of K is the free abelian group on its set of oriented k -simplices
- $\text{rank } \mathbf{C}_k = ?$



We take linear combinations of
simplices of a given dimension k

Boundary Operator

- The boundary operator $\partial_k : \mathbf{C}_k \rightarrow \mathbf{C}_{k-1}$ is a homomorphism defined linearly on a chain c by its action on any simplex

$$\sigma = [v_0, v_1, \dots, v_k] \in c,$$

$$\partial_k \sigma = \sum_i (-1)^i [v_0, v_1, \dots, \hat{v}_i, \dots, v_k],$$

where \hat{v}_i indicates that v_i is deleted from the sequence.

- $\partial_1[a, b] = b - a.$
- $\partial_2[a, b, c] = [b, c] - [a, c] + [a, b] = [b, c] + [c, a] + [a, b].$
- $\partial_3[a, b, c, d] = [b, c, d] - [a, c, d] + [a, b, d] - [a, b, c].$

Boundary Theorem

- (Theorem) $\partial_{k-1}\partial_k = 0$, for all k .

- Proof:

$$\partial_{k-1}\partial_k[v_0, v_1, \dots, v_k] =$$

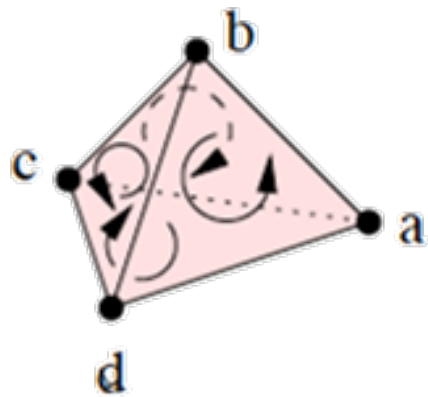
$$= \partial_{k-1} \sum_i (-1)^i [v_0, v_1, \dots, \hat{v}_i, \dots, v_k]$$

$$= \sum_{j < i} (-1)^i (-1)^j [v_0, \dots, \hat{v}_j, \dots, \hat{v}_i, \dots, v_k]$$

$$+ \sum_{j > i} (-1)^i (-1)^{j-1} [v_0, \dots, \hat{v}_i, \dots, \hat{v}_j, \dots, v_k]$$

$$= 0,$$

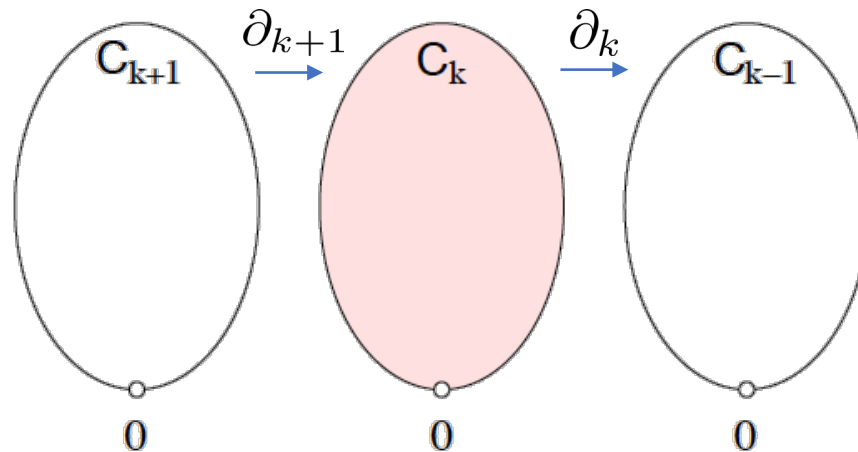
as switching i and j in the second sum negates the first sum.



Chain Complex

- The boundary operator connects the chain groups into a **chain complex** C_* :

$$\dots \rightarrow C_{k+1} \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \rightarrow \dots$$

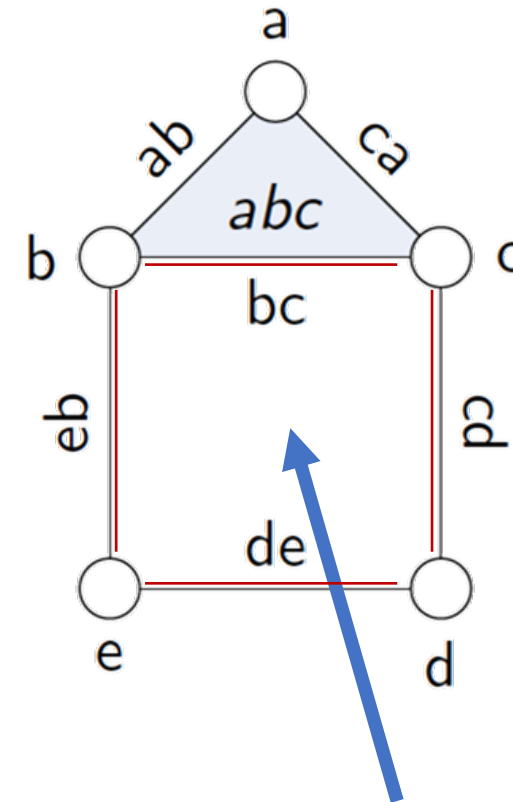
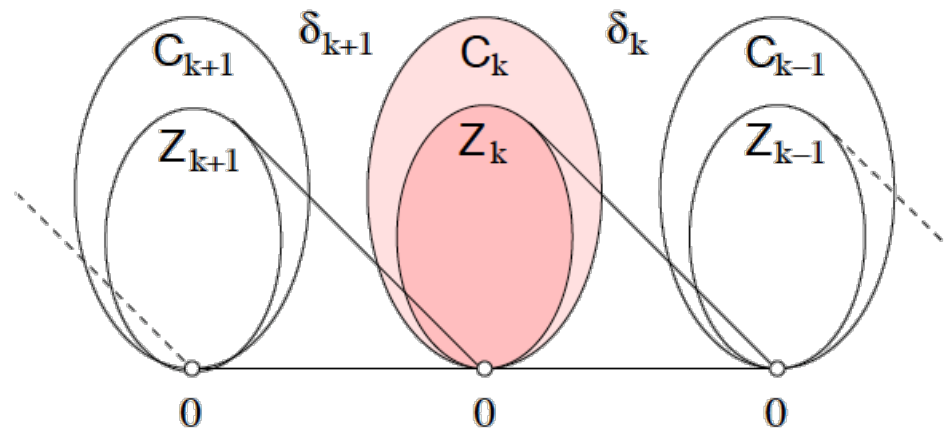


Cycle Group

- Let c be a k -chain
- If it has no boundary, it is a k -cycle (zycle?)
- $\partial_k c = \emptyset$, so $c \in \ker \partial_k$
- The k th cycle group is

$$Z_k = \ker \partial_k = \{c \in C_k \mid \partial_k c = \emptyset\}.$$

$$\partial = \delta$$



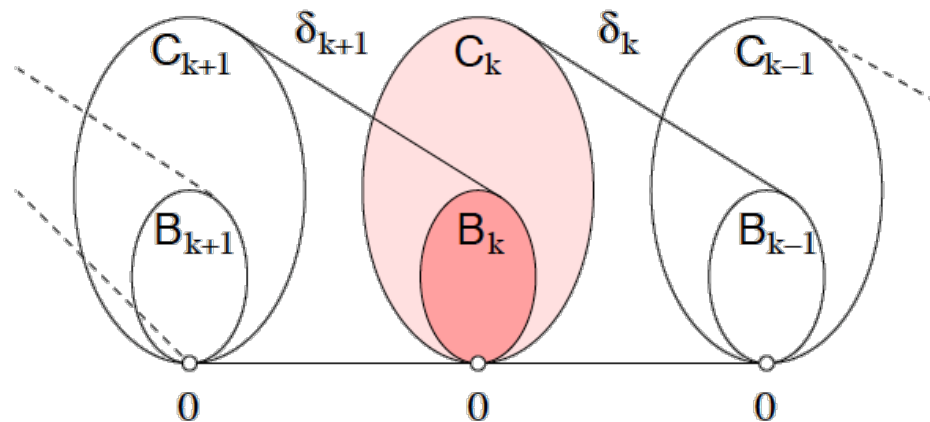
$$\begin{aligned} \partial(bc + cd + de + eb) \\ = c - b + d - c + e - d + b - e = 0. \end{aligned}$$

Boundary Group

- Let b be a k -chain
- If b is a boundary of something, it is a k -boundary.
- The k th boundary group is

$$\mathbf{B}_k = \text{im } \partial_{k+1} = \{c \in \mathbf{C}_k \mid \exists d \in \mathbf{C}_{k+1} : c = \partial_{k+1}d\}.$$

$$\partial = \delta$$



Boundaries are Cycles!

- Let b be a k -boundary.
- Then, $\exists c \in \mathbf{C}_{k+1}$, such that $b = \partial_{k+1}c$.
- What is the boundary of b ?

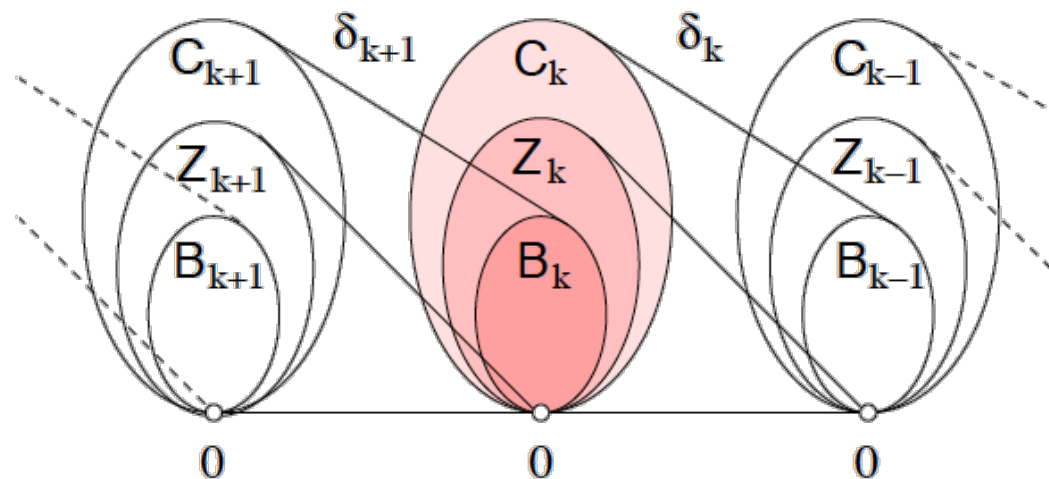
$$\partial_k b = \partial_k \partial_{k+1} c = \emptyset,$$

$$\bullet \mathbf{B}_k \subseteq \mathbf{Z}_k \subseteq \mathbf{C}_k$$

by the boundary theorem.

- That is, every boundary is a cycle!

$$\partial = \delta$$



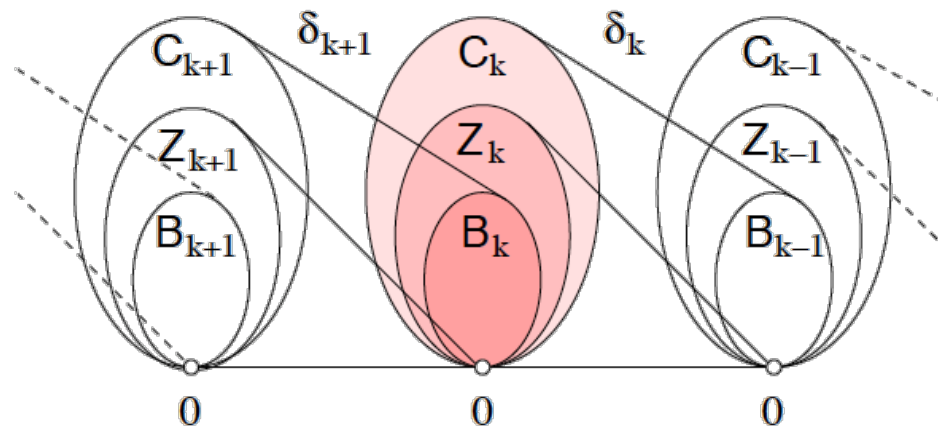
Nesting behavior!

Simplicial Homology

- The k th homology group is

$$H_k = Z_k / B_k = \ker \partial_k / \text{im } \partial_{k+1}.$$

- If $z_1 = z_2 + B_k$, $z_1, z_2 \in Z_k$, we say z_1 and z_2 are **homologous**
- $z_1 \sim z_2$.



Homology Groups

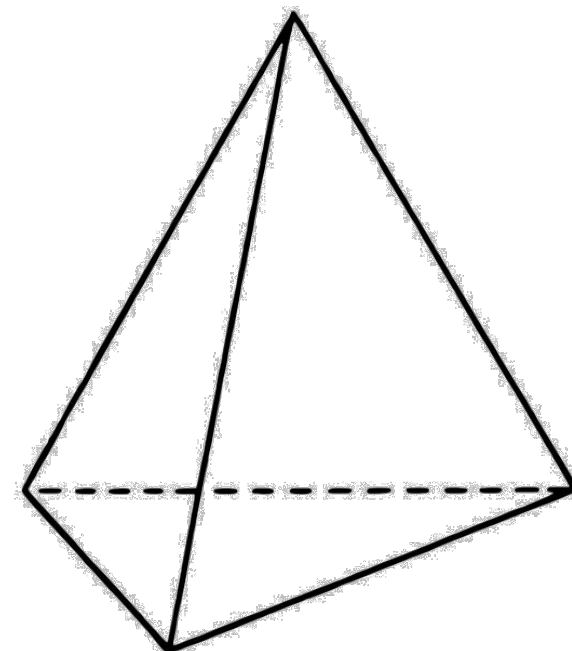
- Homology groups are finitely generated abelian.
- (Theorem) Every finitely generated abelian group is isomorphic to product of cyclic groups of the form

$$\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \dots \times \mathbb{Z}_{m_r} \times \boxed{\mathbb{Z} \times \mathbb{Z} \times \dots \times \mathbb{Z}}$$

- The k th Betti number β_k of a simplicial complex K is $\beta_k = \beta(\mathbf{H}_k)$, the rank of the free part of \mathbf{H}_k .
- Torsion coefficients
 - Alexander Duality:
 - β_0 measures the number of components of the complex.
 - β_1 is the rank of a basis for the tunnels.
 - β_2 counts the number of voids in the complex.

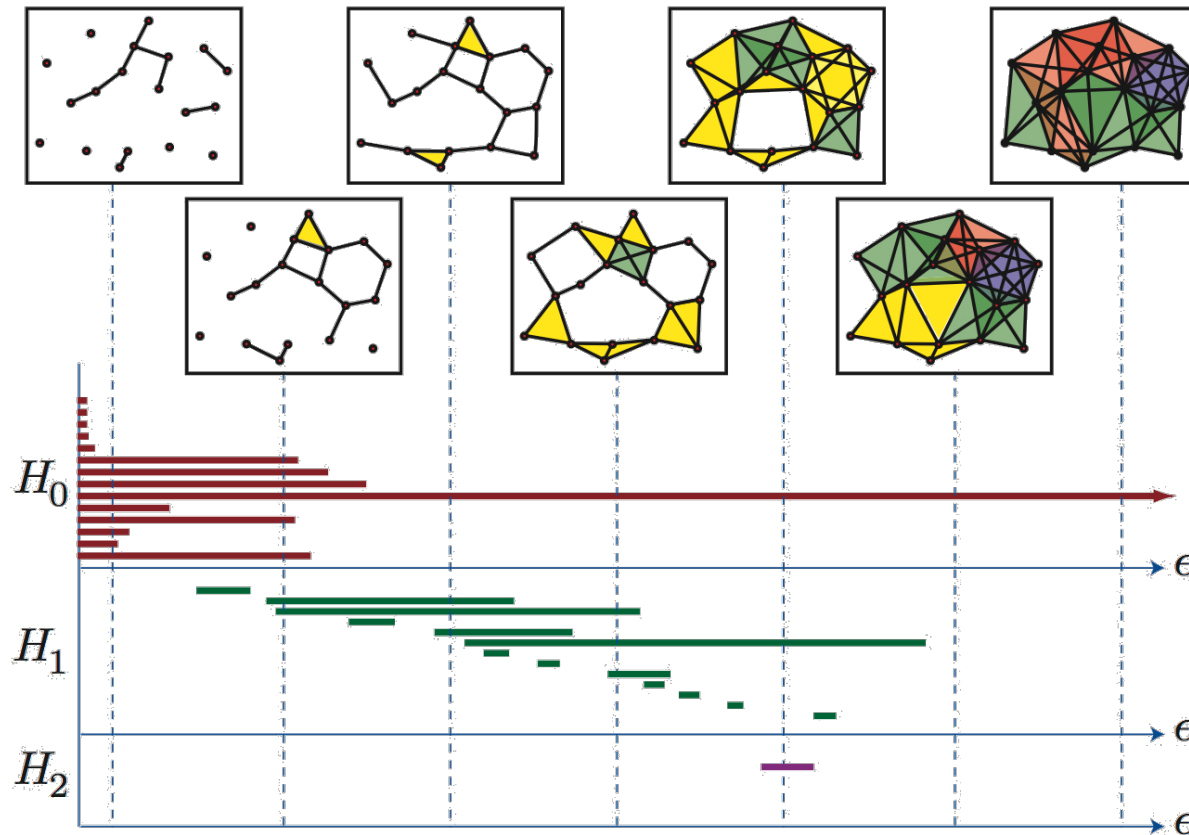
Betti Numbers β_i

- Ranks of the free part of homology groups H_i
- β_0 counts the number of connected components
- β_1 counts the number of independent loops
- β_2 counts the number of independent voids
- ...

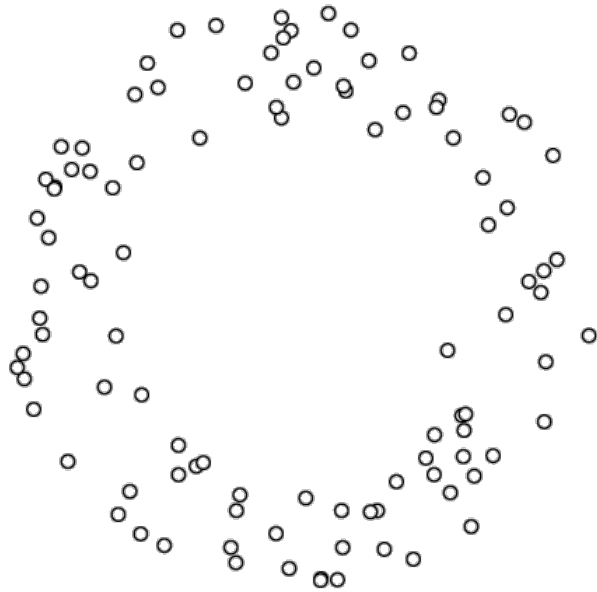


Topology is fundamentally a tool for classification

Persistent Homology

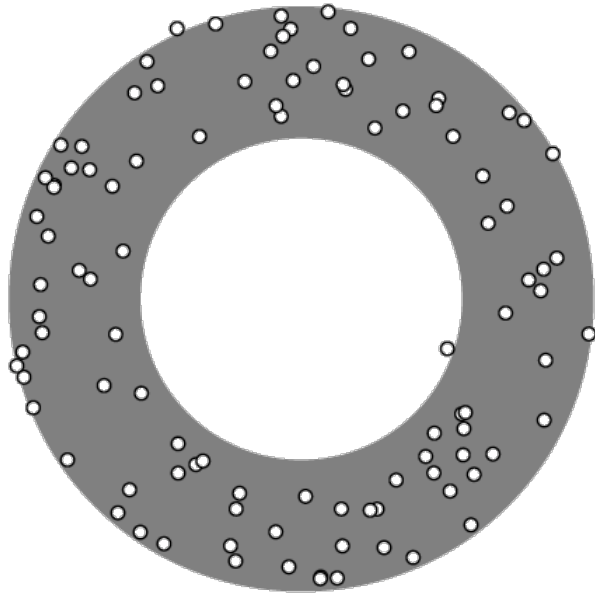


Sampled Data Has “Shape”



2-dimensional
Approximates annulus

Sampled Data Has “Shape”



2-dimensional

Approximates annulus

Topological features of annulus:

1 component ($\beta_0 = 1$)

1 loop ($\beta_1 = 1$)

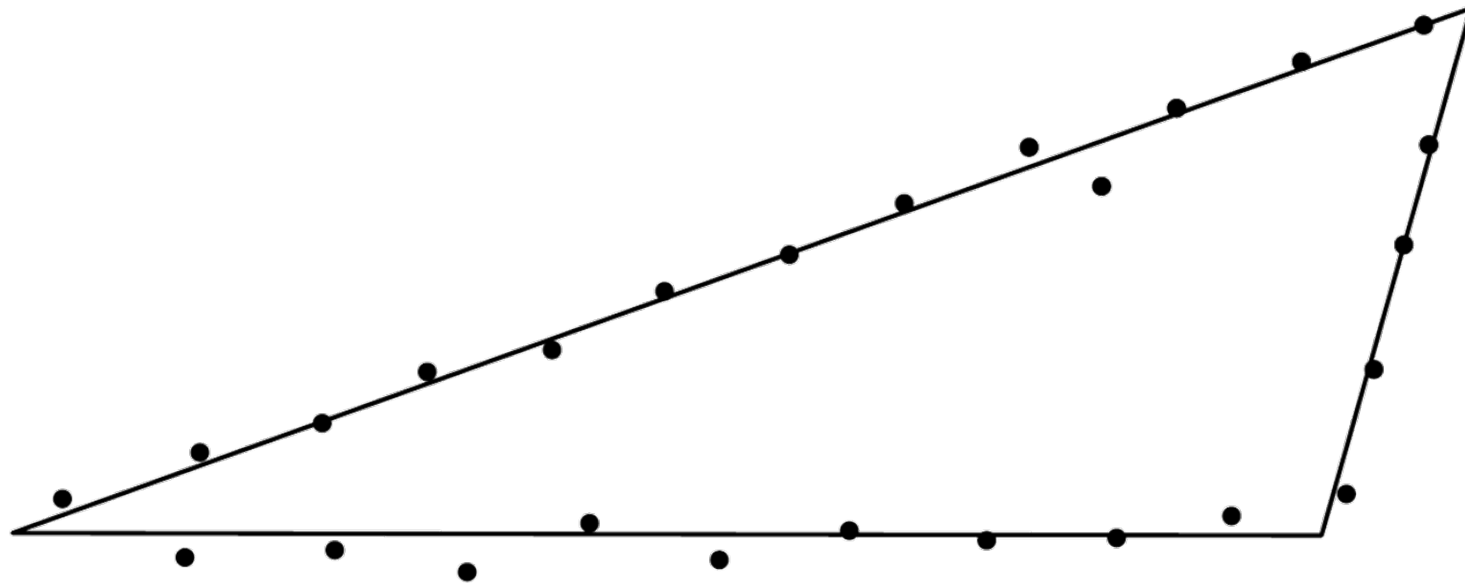
Goal: Recover *topology* of annulus from point cloud

We do so by building various complexes on the point cloud

Filtrations

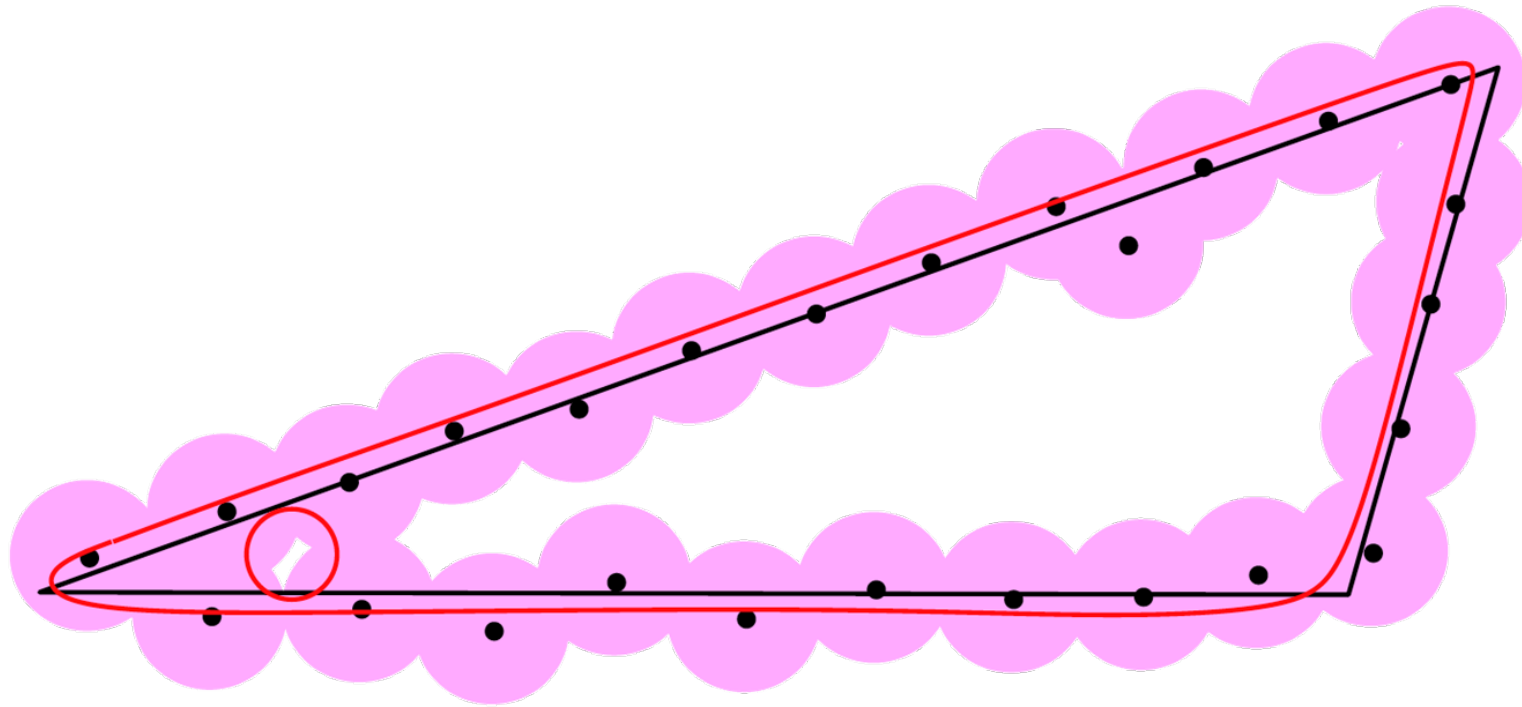
How to Choose ϵ ?

- How to determine the topology of the underlying space from a point cloud approximation?



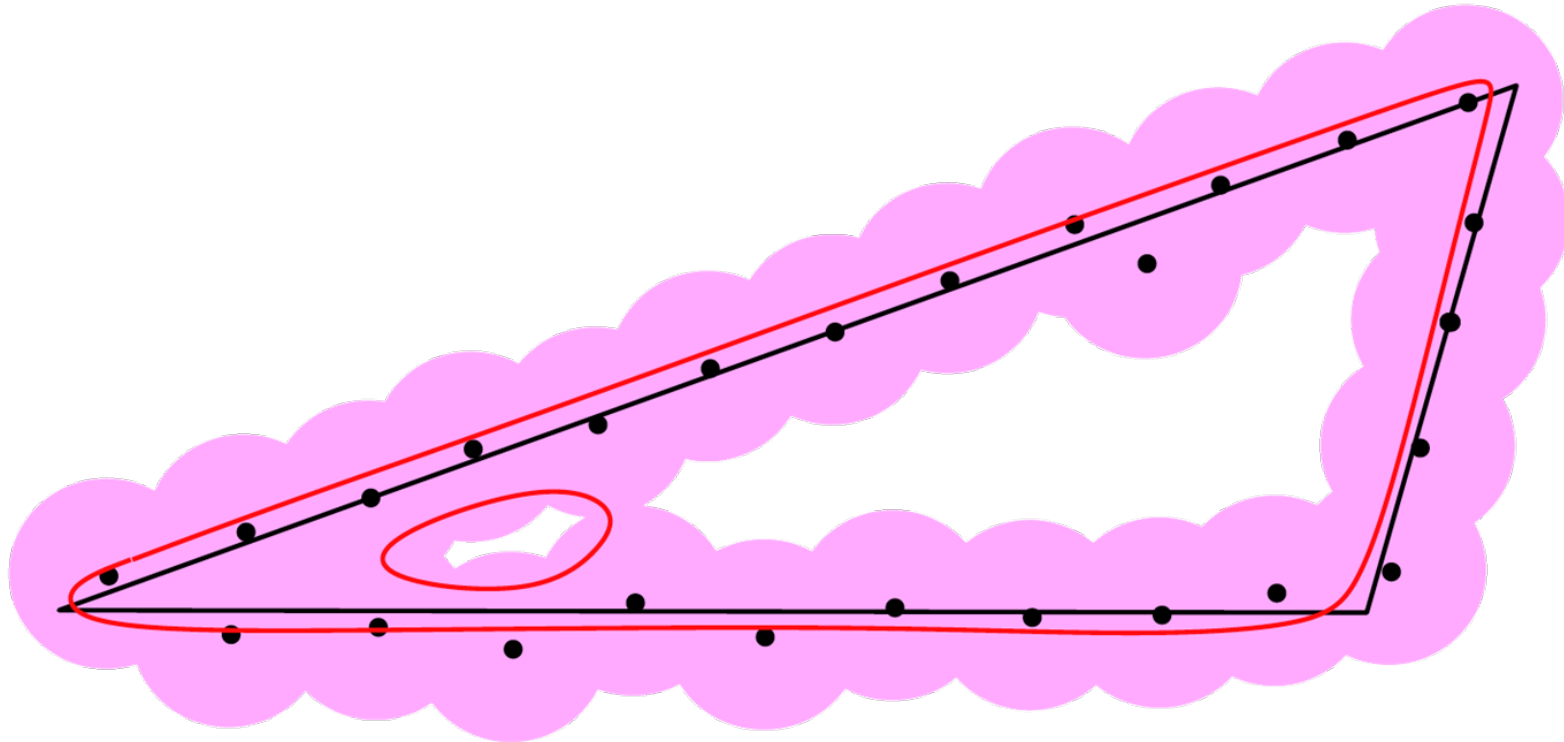
How to Choose ϵ ?

- How to determine the topology of the underlying space from a point cloud approximation?



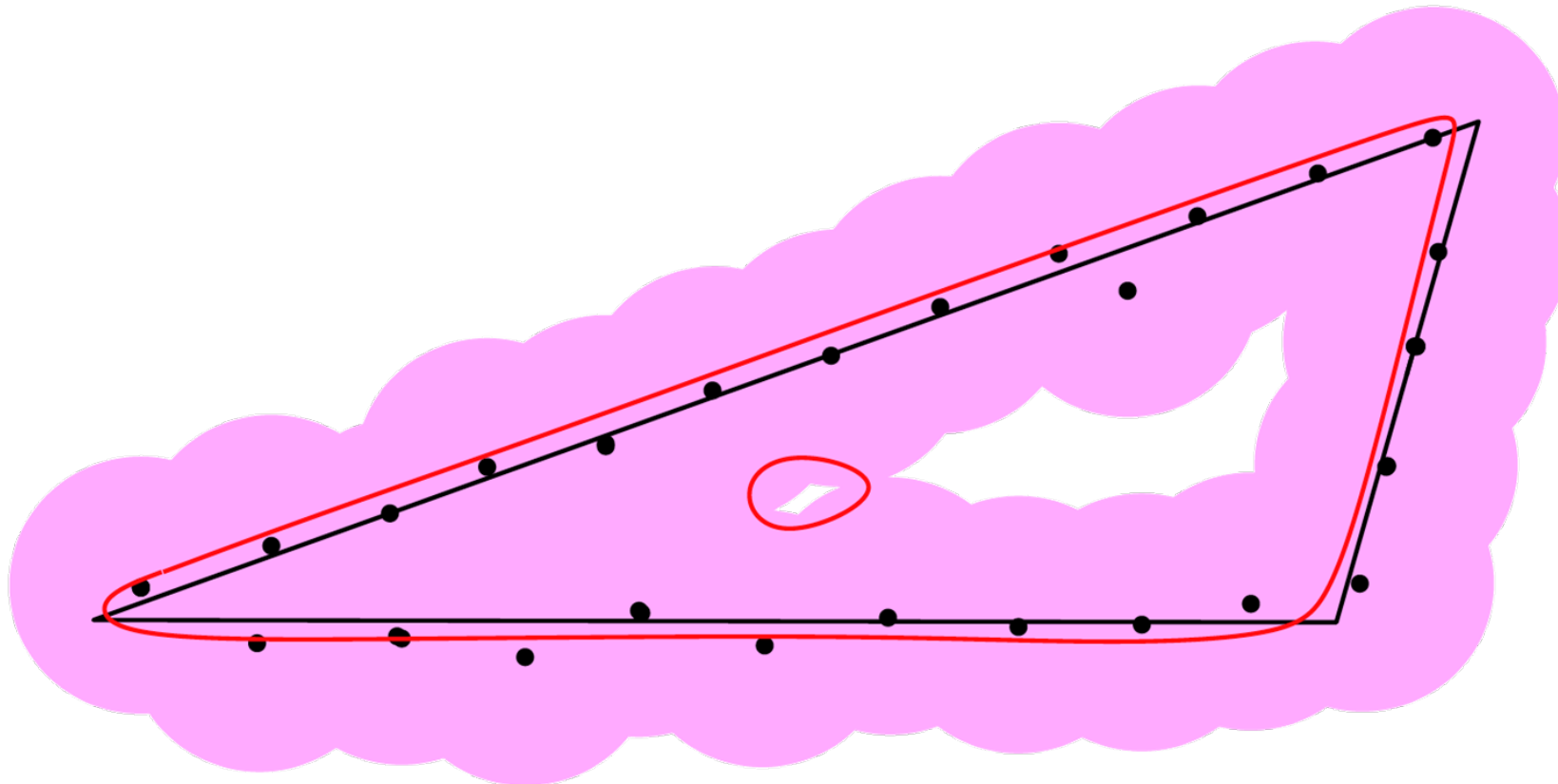
How to Choose ϵ ?

- How to determine the topology of the underlying space from a point cloud approximation?

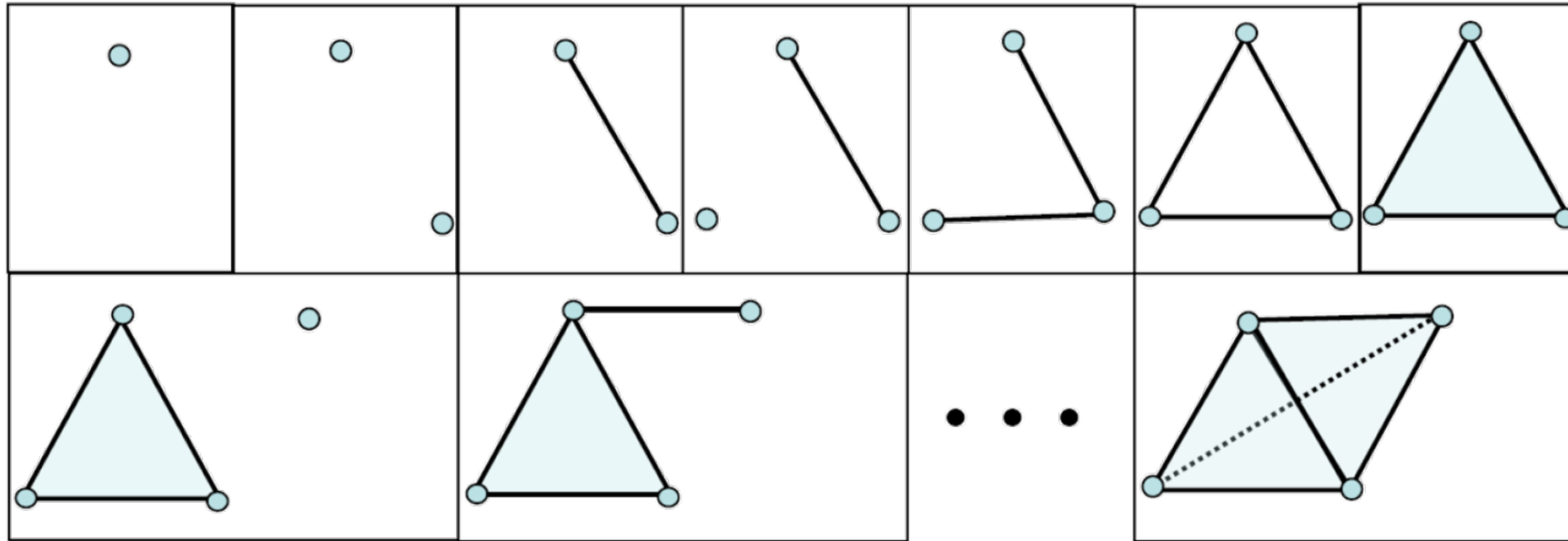


How to Choose ϵ ?

- How to determine the topology of the underlying space from a point cloud approximation?



Filtrations



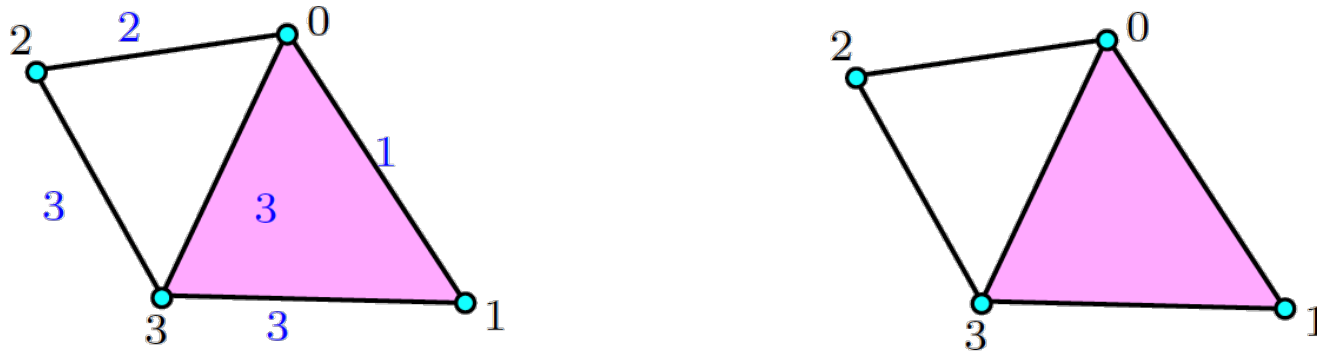
A **filtration** of a (finite) simplicial complex K is a sequence of subcomplexes such that

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

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

Sub-simplices of a simplex must be added before the simplex!

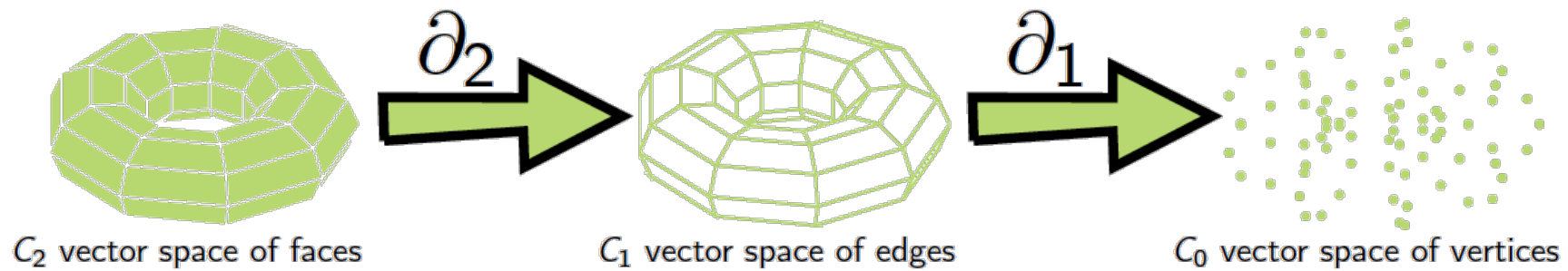
The Sub-Level Set Filtration



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

Persistent Homology:
Do not choose an ϵ !

Standard Homology



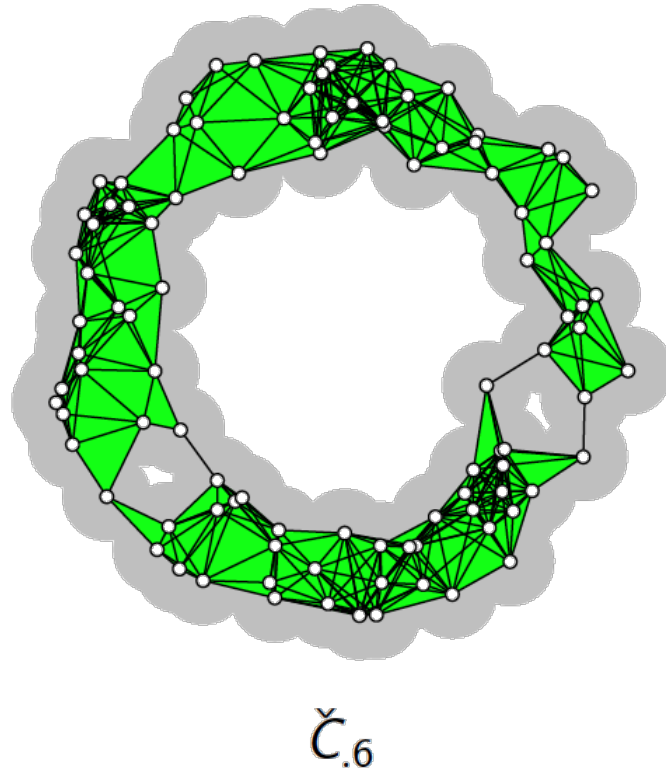
Take the linear extension of the boundary operator:

$$\partial_d([v_0, \dots, v_d]) = \sum_{i=0}^d (-1)^i [v_0, \dots, \hat{v}_i, \dots, v_d]$$

Fact: $\partial_{d-1} \circ \partial_d \equiv 0 \Rightarrow \text{Im } \partial_d \subseteq \text{ker } \partial_{d-1}$

Definition: $H_d(K) = \text{ker } \partial_d / \text{Im } \partial_{d+1}$

Čech Complex



The Čech complex \check{C}_ϵ encodes the intersection pattern of M_ϵ : Encode:

Points as *vertices*
(0-cells)

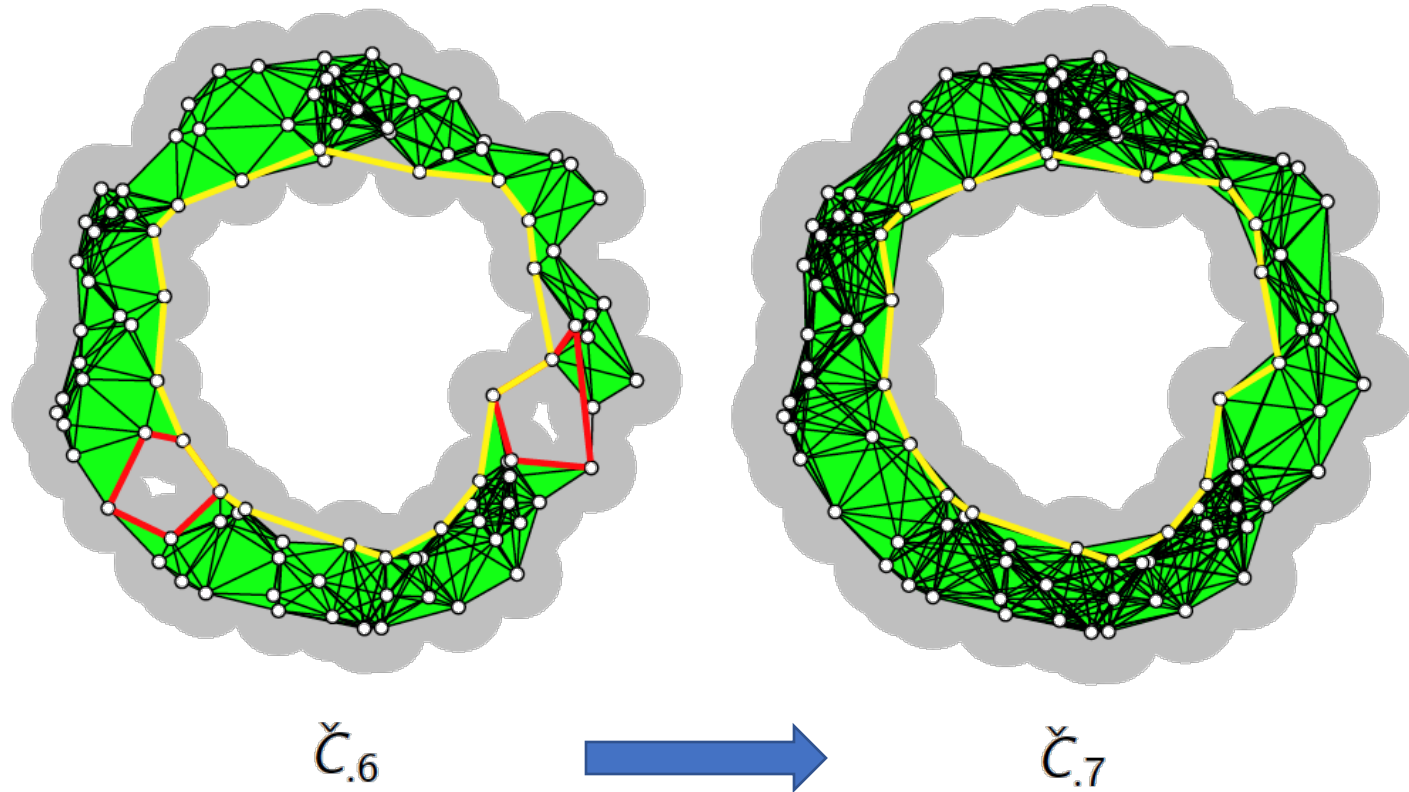
Pairwise intersections
as *edges* (1-cells)

Threeway intersections
as *triangles* (2-cells)

k -way intersections as
($k+1$)-cells

Can be hard to compute ...

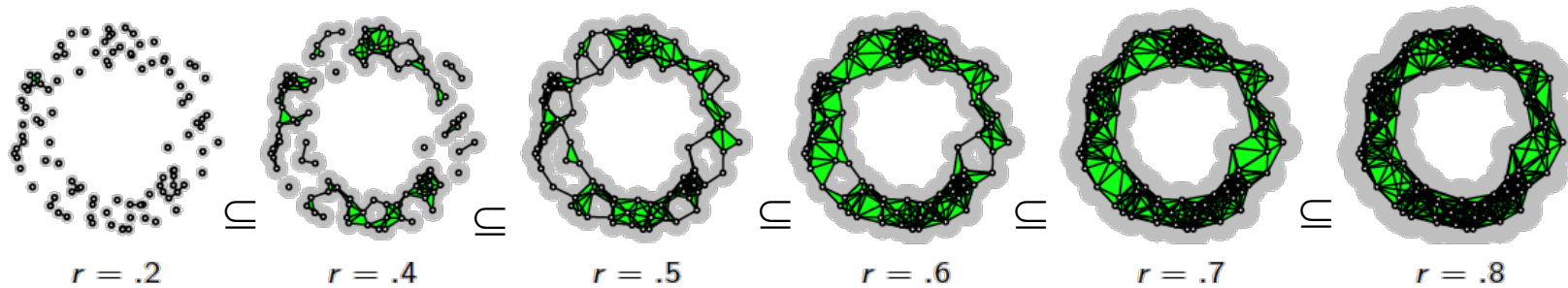
We Can Track Topological Features in a Filtration



Functoriality allows us to systematically track holes over time!

The inclusion map among the complexes translates to a homomorphism between the homology groups

Persistent Homology is Functorial Homology

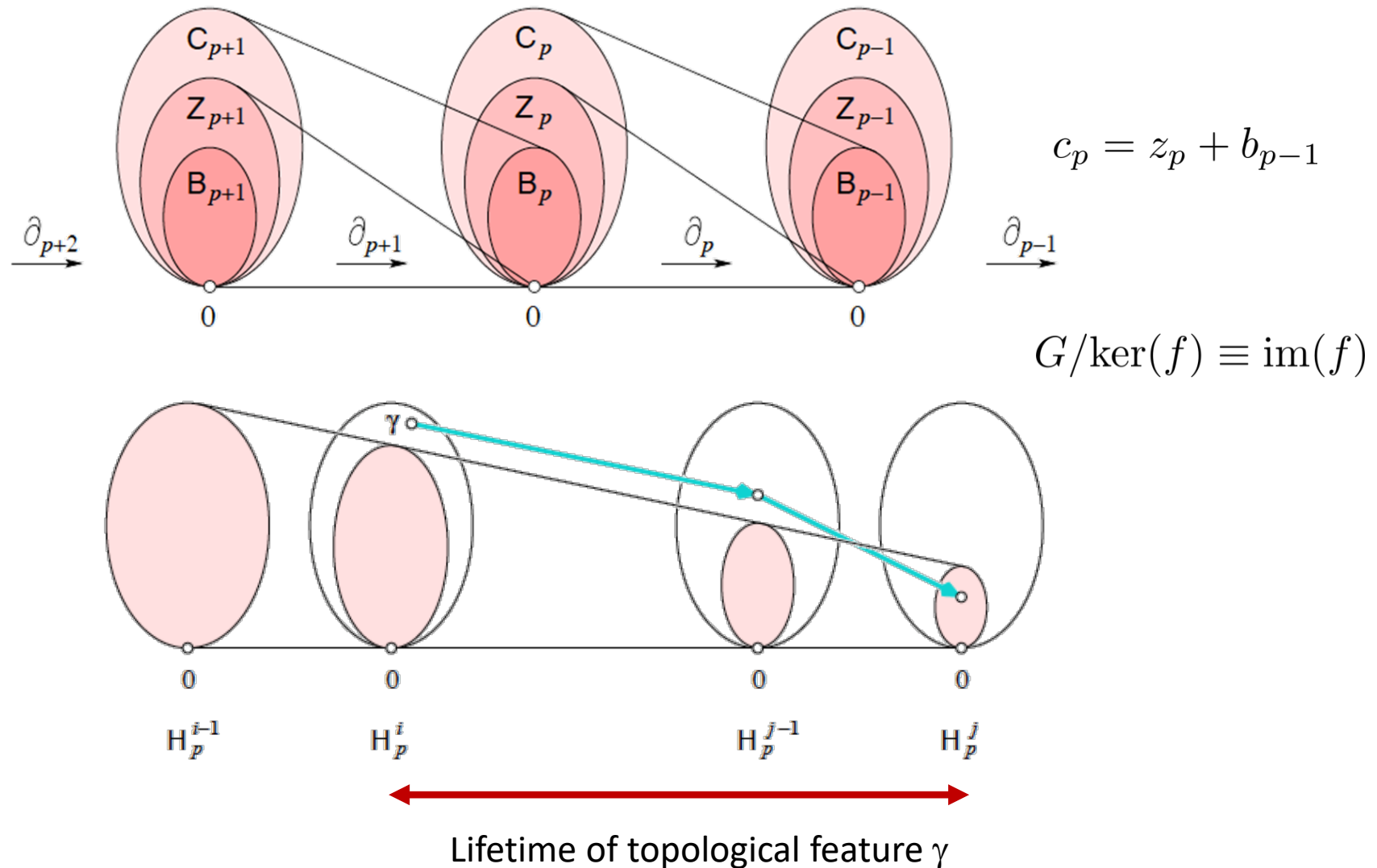


$$H_d(\check{C}_*) = \bigoplus_{\epsilon} H_d(\check{C}_\epsilon)$$

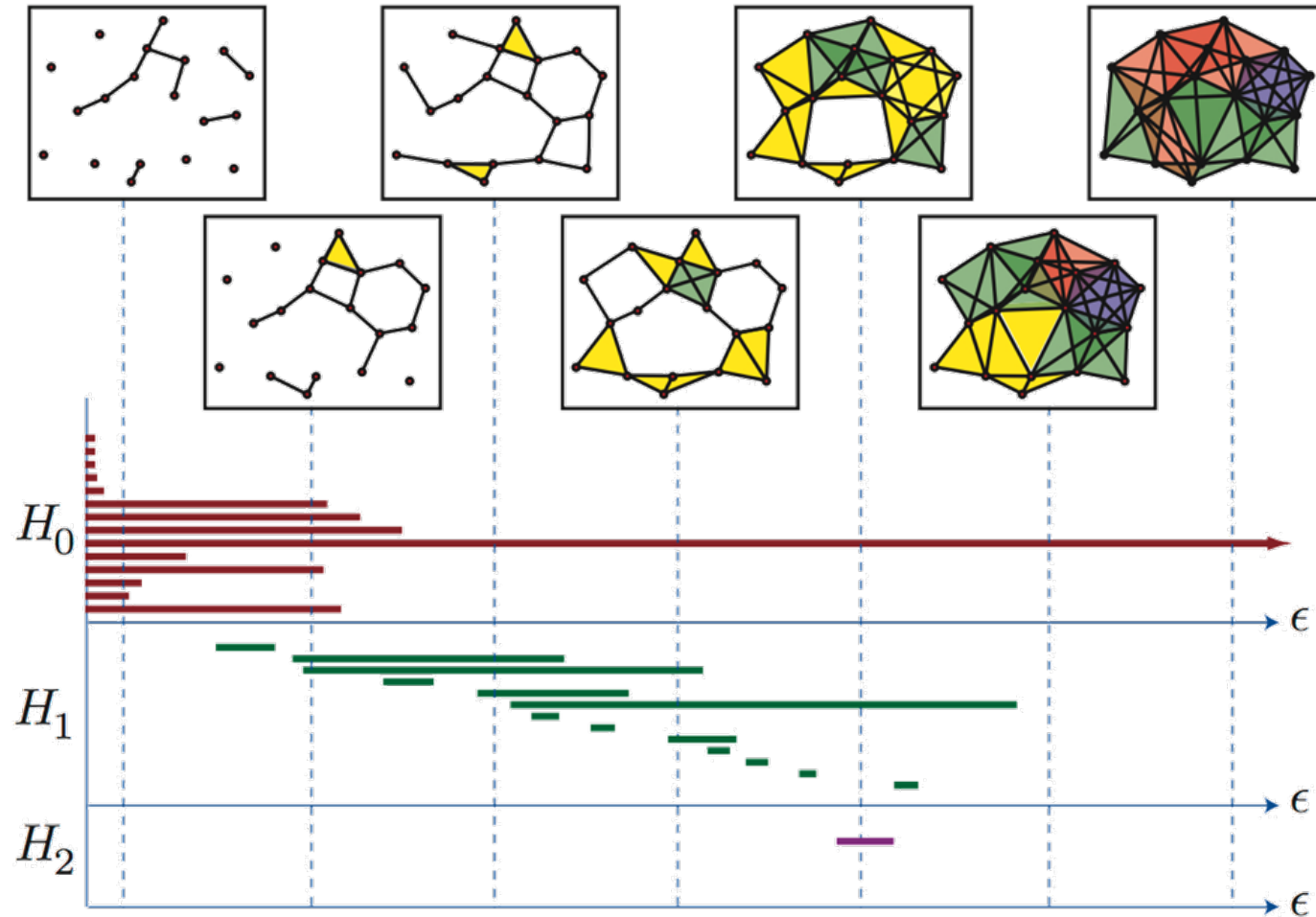
Homology of the entire filtration

Homomorphisms at the homology level allow us to track homology classes – i.e., topological features

Persistence of Homology Classes

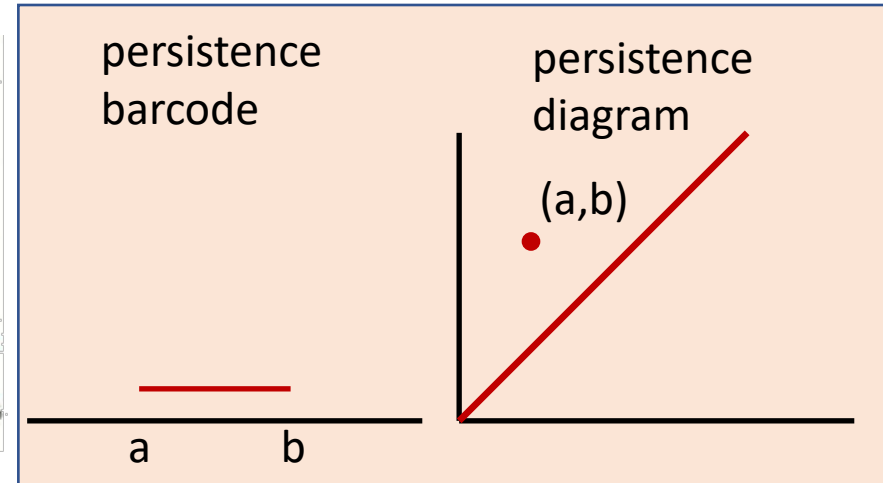
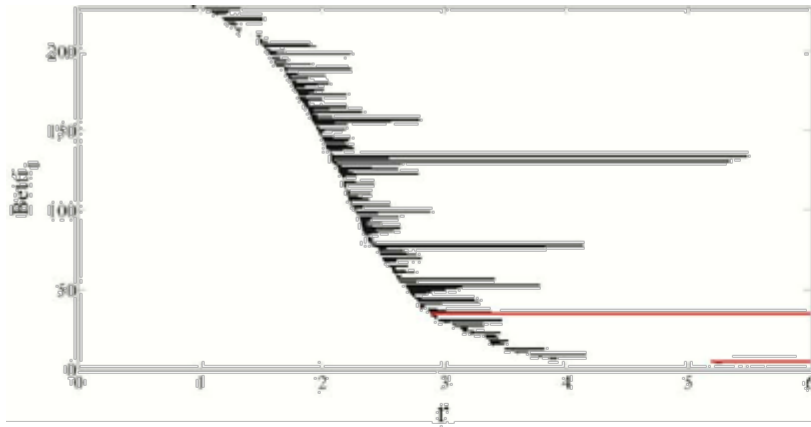


Barcodes are the Lifetimes of Topological Features

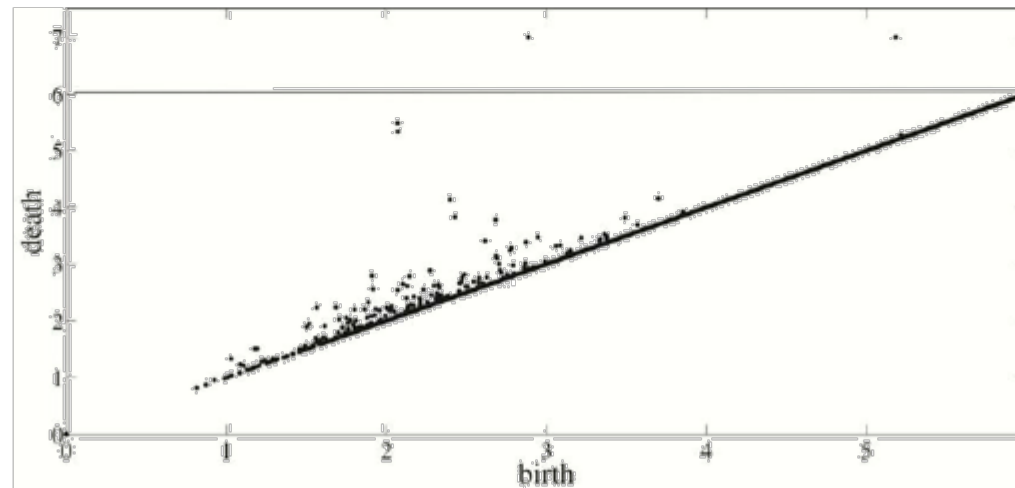


Barcodes are the output of persistent homology

Another View: Persistence Diagrams



long barcodes =
points away from
the diagonal =
robust features



Short barcodes =
points near
the diagonal =
noise

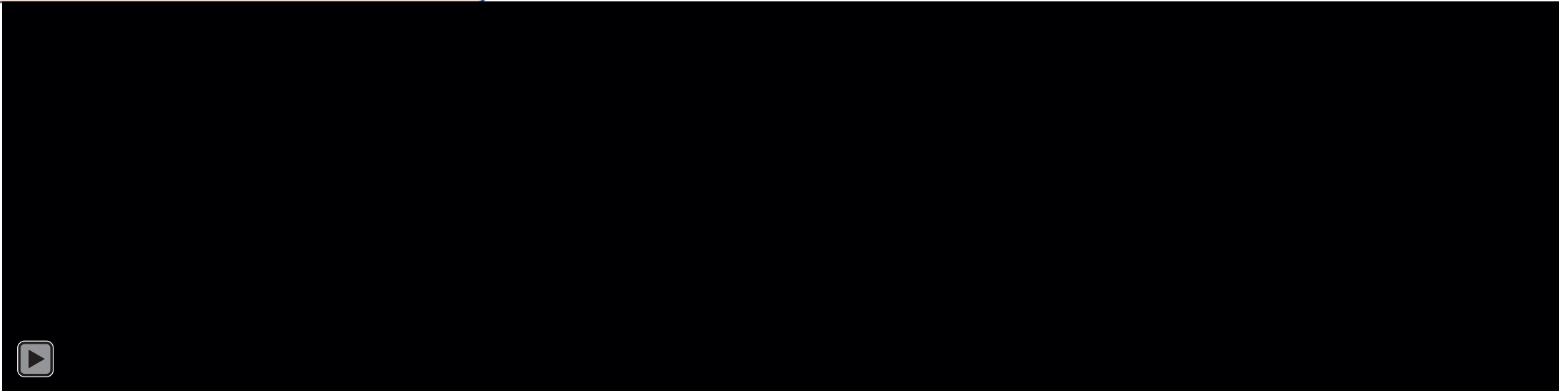
Map 1-D intervals to points in 2-D

Today: Persistent Homology

Today: Persistent Homology

(3) Visualization/input to ML pipeline

Youtube “Persistent homology on a noisy torus”



$$\emptyset \subseteq \Sigma^{(1)} \subseteq \Sigma^{(2)} \subseteq \dots \subseteq \Sigma^{(n-1)} \subseteq \Sigma^{(n)}$$

(1) Topology

Functoriality of homology

$$0 \rightarrow H_p(\Sigma^{(1)}) \rightarrow H_p(\Sigma^{(2)}) \rightarrow \dots \rightarrow H_p(\Sigma^{(n-1)}) \rightarrow H_p(\Sigma^{(n)})$$

(2) Algebra

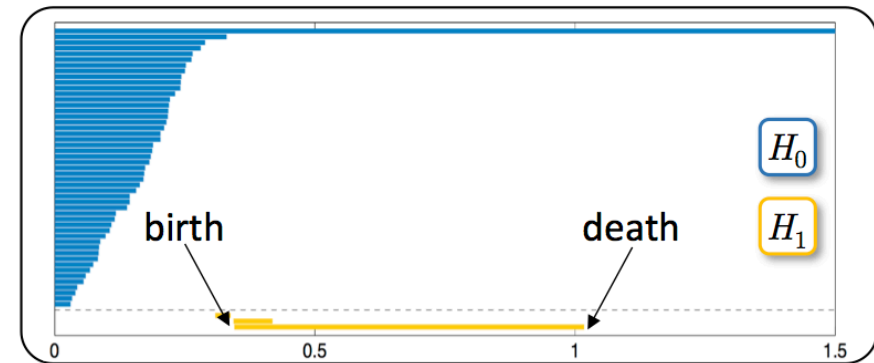
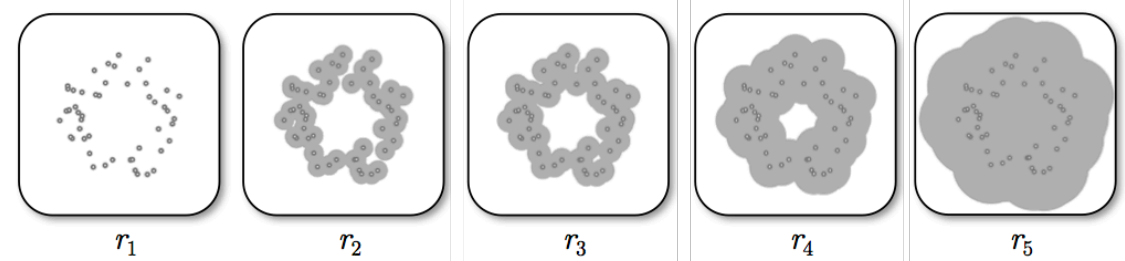
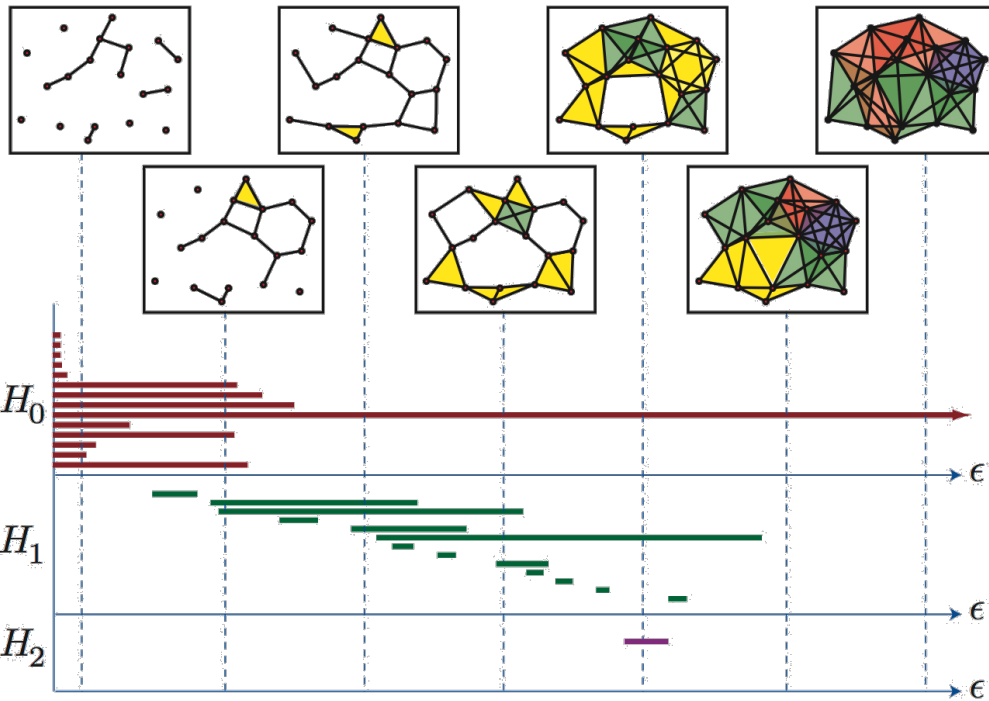
Pipeline

Dataset:
- Euclidean point cloud data
- Metric spaces
- Triangulated shapes
- Graphs, matrices...

Filtered simplicial complex
• Possible to have both inclusions and deletions (dynamic data)

Vector spaces with linear maps: a **persistent vector space**

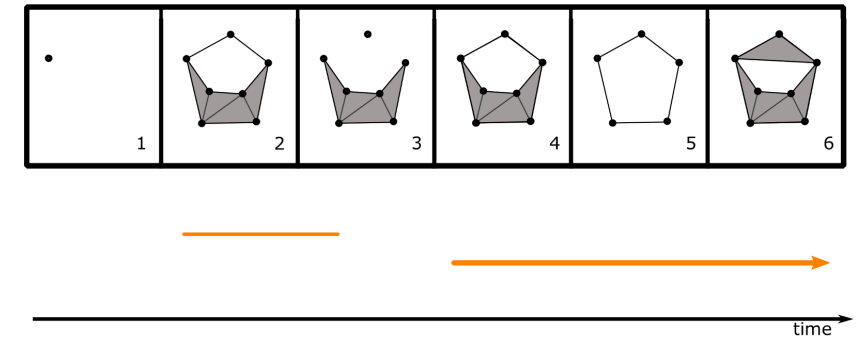
Visual summary: a **persistence diagram** or **persistence barcode**
• Postprocessing available for ML-friendly output



Plan

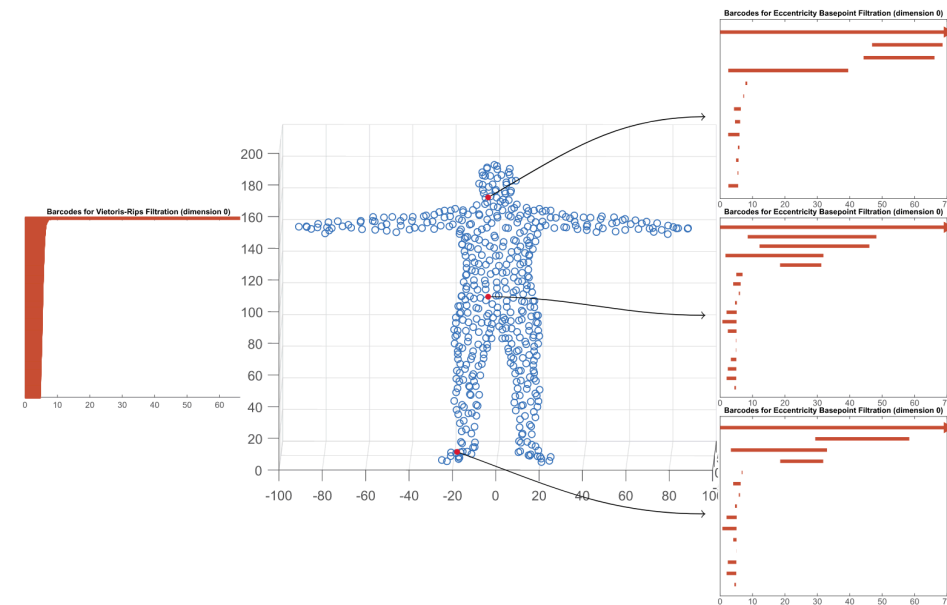
Today:

- Computing homology; the persistence algorithm
- Demo using Ripser and Javaplex
- Mathematical theory of persistence; Zigzag persistence
- Sublevel set persistence; Demo using Dionysus
- Bottleneck distance and Gromov-Hausdorff stability



Next time:

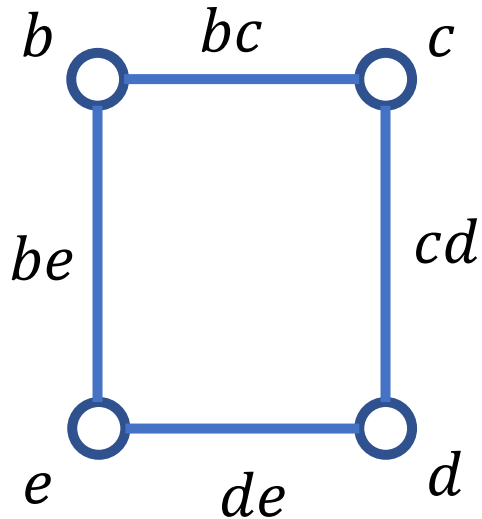
- 0-dimensional persistence and single linkage clustering
- The Mapper algorithm
- Applications of TDA:
 - Clustering, time series analysis, topological loss for neural networks, applications to empirical science



Background: Computing Homology

Recall: Simplicial Complexes \rightarrow Vector Spaces

- ◆ Σ a simplicial complex, Σ_0 the 0-simplices (vertices), Σ_1 the 1-simplices (edges), Σ_2 the 2-simplices (triangles), ...
- ◆ \mathbb{F} a field, typically $\mathbb{Z}/2\mathbb{Z} = \{0,1\}$
- ◆ $C_p(\Sigma)$, the vector space of p -chains
 - Intuitively, just allow addition and scalar multiplication (using \mathbb{F}) of simplices!
 - Linear combinations of p -simplices will be called p -chains

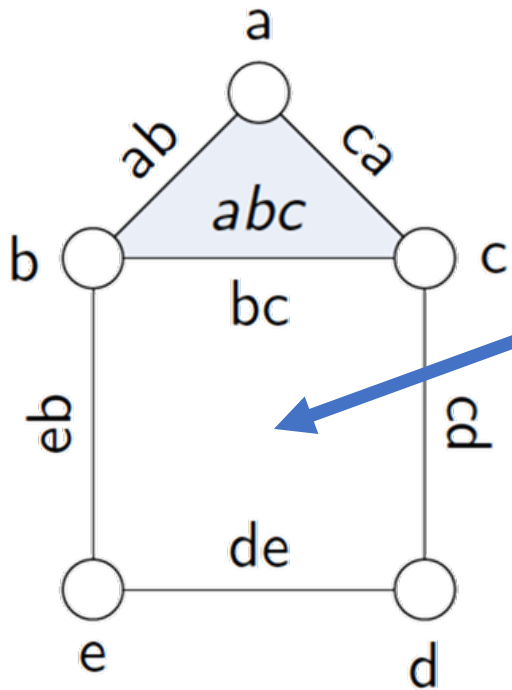


- ◆ Intuitively, $bc + cd + de + be$ traces out a loop bounding a box.

Recall: the Boundary Map ∂

- ♦ Alternating sum (hat denotes omission):

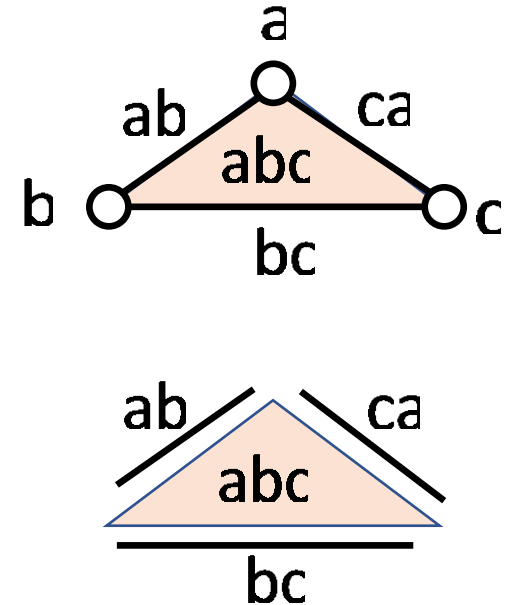
$$\partial[x_0, \dots, x_n] = \sum_{i=0}^n (-1)^i [x_0, \dots, \hat{x}_i, \dots, x_n]$$



$$\begin{aligned} \partial(bc + cd + de + eb) \\ = c - b + d - c + e - d + b - e = 0. \end{aligned}$$

- ♦ Key identity: $\partial^2 = 0$.

$$\begin{aligned} \partial^2(abc) &= \partial(bc - ac + ab) \\ &= c - b - c + a + b - a = 0. \end{aligned}$$



- ♦ $\partial = 0$ on linear combinations that are boundaries. Intuitively, $\partial^2 = 0$ says that the boundary of a simplex really is a boundary!

Recall: the Chain Complex

- We now have a sequence of vector spaces and linear maps:

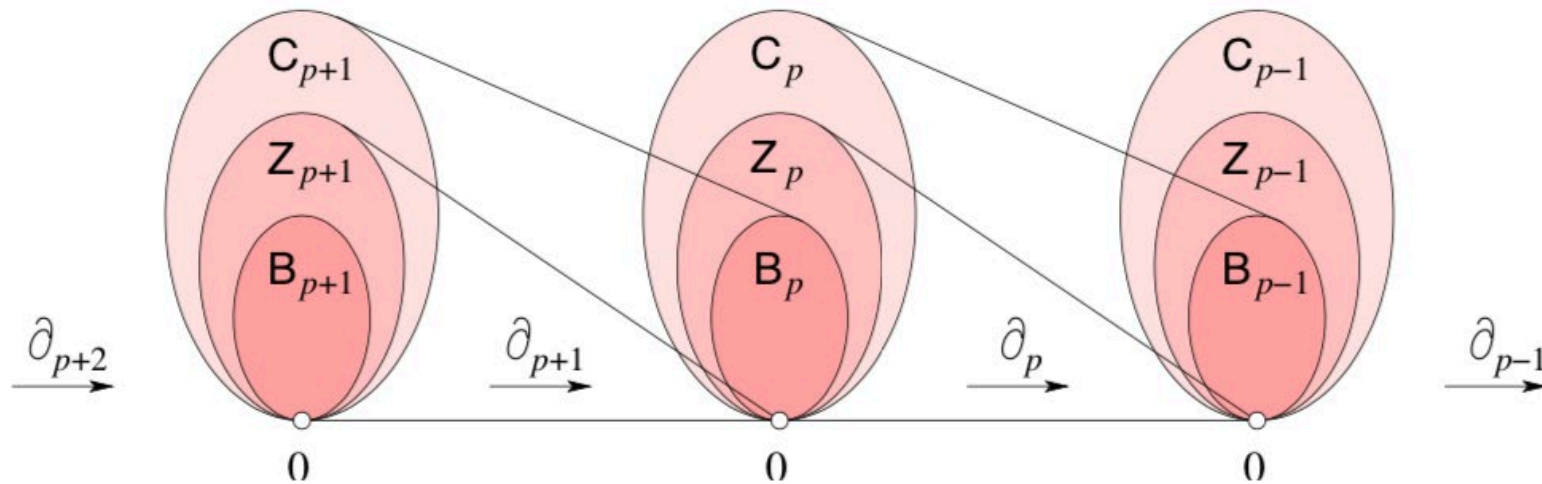
$$\dots \xrightarrow{\partial_{p+2}} C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \dots$$

- Notation: $Z_p := \ker \partial_p = \{v \in C_p : \partial_p(v) = 0\}$ the p -cycles, $B_p := \text{im } \partial_{p+1} \subseteq C_p$ the p -boundaries.

p th homology vector space:

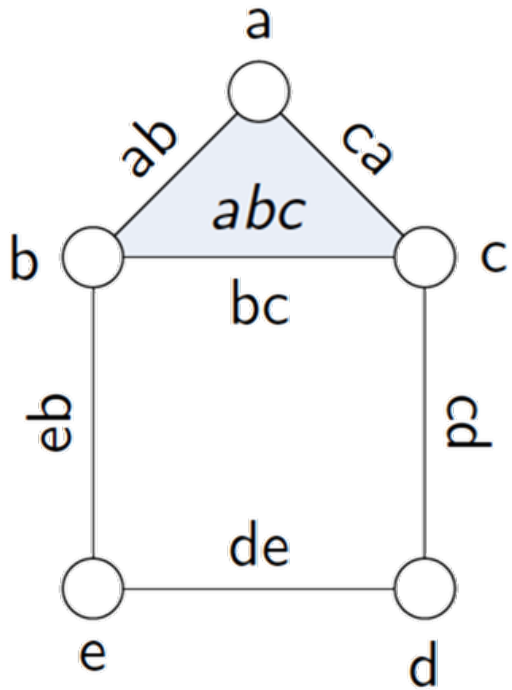
$$H_p := \frac{\ker \partial_p}{\text{im } \partial_{p+1}}$$

Intuition: p -dimensional connectivity



Setting up the Linear Algebra

◆ $\partial_p: C_p \rightarrow C_{p-1}$ is a linear map between vector spaces, so we'll use matrices



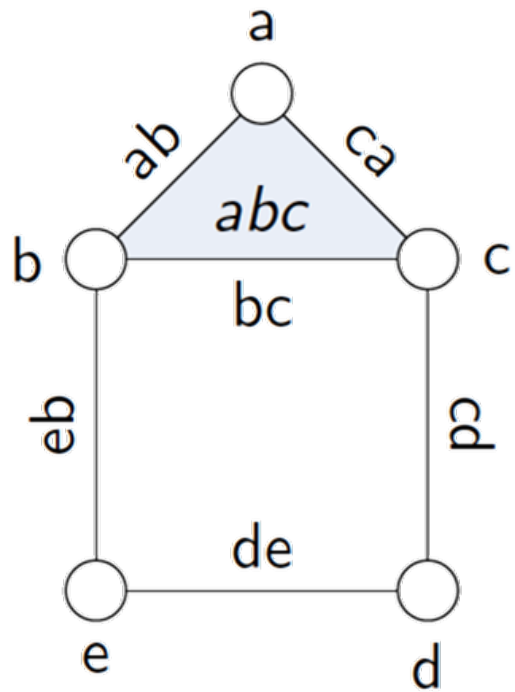
$$\begin{array}{c}
 ab \quad ca \quad bc \quad cd \quad de \quad eb \\
 a \\
 b \\
 c \\
 d \\
 e
 \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1
 \end{bmatrix}$$

$\underbrace{\hspace{15em}}_{\partial_1}$

$$\begin{array}{c}
 abc \\
 ab \\
 ca \\
 bc \\
 cd \\
 de \\
 eb
 \end{array}
 \begin{bmatrix}
 1 \\
 1 \\
 1 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

$\underbrace{\hspace{5em}}_{\partial_2}$

Setting up the Linear Algebra



$$\begin{array}{c}
 ab \quad ca \quad bc \quad cd \quad de \quad eb \\
 a \\
 b \\
 c \\
 d \\
 e
 \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1
 \end{bmatrix}$$

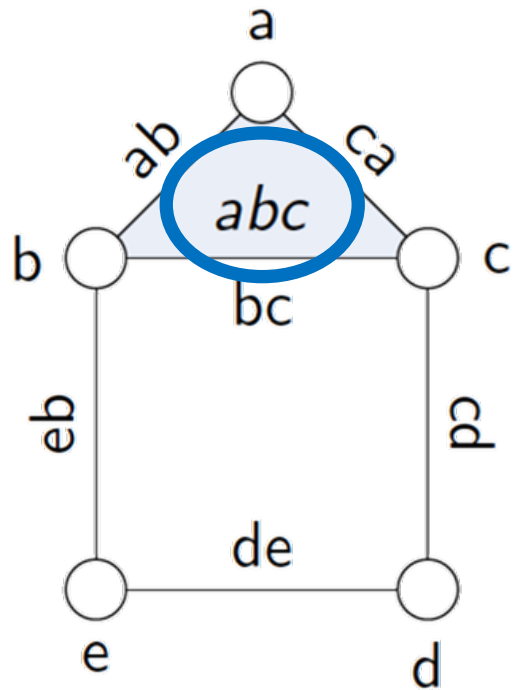
$\underbrace{\hspace{15em}}_{\partial_1}$

$$\begin{array}{c}
 abc \\
 ab \\
 ca \\
 bc \\
 cd \\
 de \\
 eb
 \end{array}
 \begin{bmatrix}
 1 \\
 1 \\
 1 \\
 0 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

$\underbrace{\hspace{10em}}_{\partial_2}$

◆ E. g. $H_1 = \frac{\ker \partial_1}{\text{im } \partial_2}$, so we need to extract bases for these vector spaces

Setting up the linear algebra



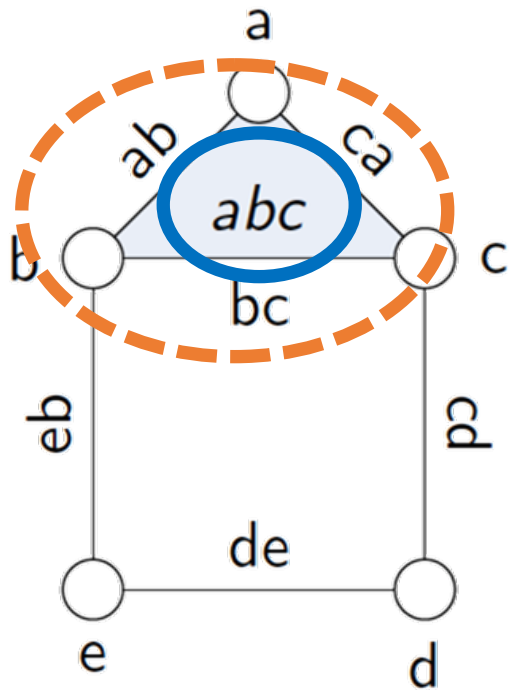
$$\begin{array}{c}
 ab \quad ca \quad bc \quad cd \quad de \quad eb \\
 a \\
 b \\
 c \\
 d \\
 e
 \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1
 \end{bmatrix}$$

∂_1

$$\begin{array}{c}
 abc \\
 ab \\
 ca \\
 bc \\
 cd \\
 de \\
 eb
 \end{array}
 \begin{bmatrix}
 1 \\
 1 \\
 1 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

∂_2

Setting up the Linear Algebra



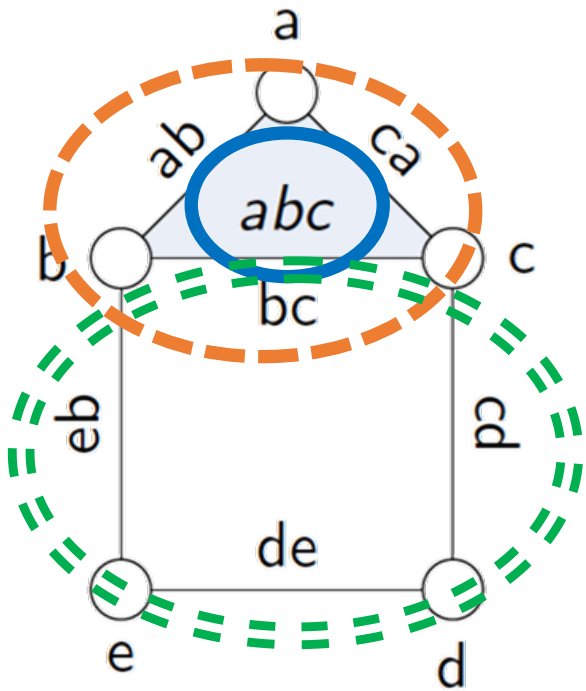
$$\begin{array}{c}
 \text{ab} \quad \text{ca} \quad \text{bc} \quad \text{cd} \quad \text{de} \quad \text{eb} \\
 \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1
 \end{bmatrix}
 \end{array}$$

∂_1

$$\begin{array}{c}
 \text{abc} \\
 \text{ab} \\
 \text{ca} \\
 \text{bc} \\
 \text{cd} \\
 \text{de} \\
 \text{eb}
 \end{array}
 \begin{bmatrix}
 1 \\
 1 \\
 1 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

∂_2

Setting up the Linear Algebra



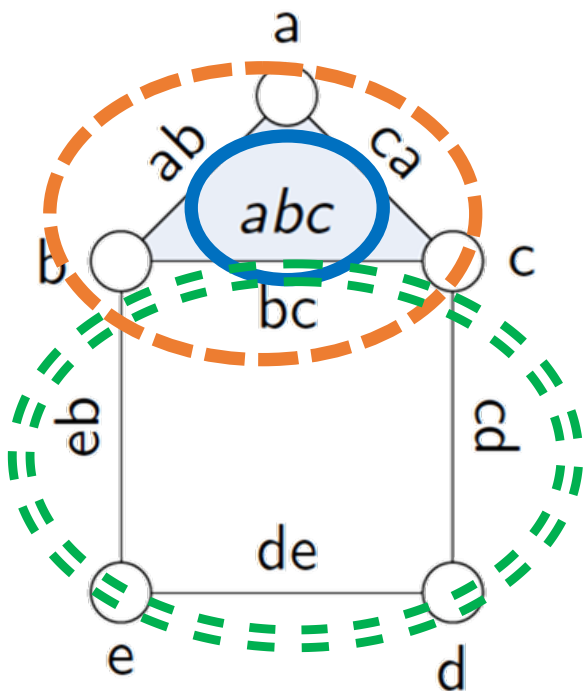
$$\begin{array}{c}
 \text{ab} \quad \text{ca} \quad \text{bc} \quad \text{cd} \quad \text{de} \quad \text{eb} \\
 \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1
 \end{bmatrix}
 \end{array}$$

∂_1

$$\begin{array}{c}
 \text{abc} \\
 \text{ab} \\
 \text{ca} \\
 \text{bc} \\
 \text{cd} \\
 \text{de} \\
 \text{eb}
 \end{array}
 \begin{bmatrix}
 1 \\
 1 \\
 1 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

∂_2

Setting up the Linear Algebra



$$\begin{array}{c}
 ab \quad ca \quad bc \quad cd \quad de \quad eb \\
 \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1
 \end{bmatrix}
 \end{array}$$

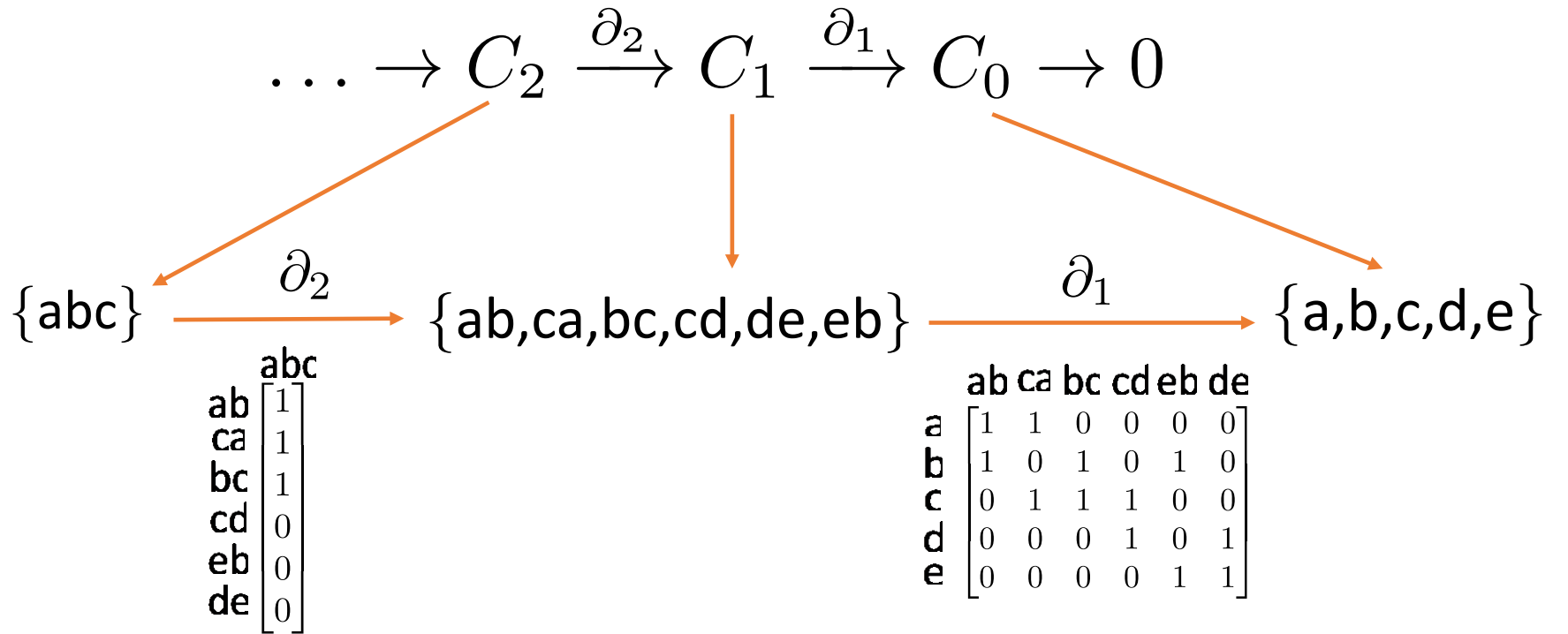
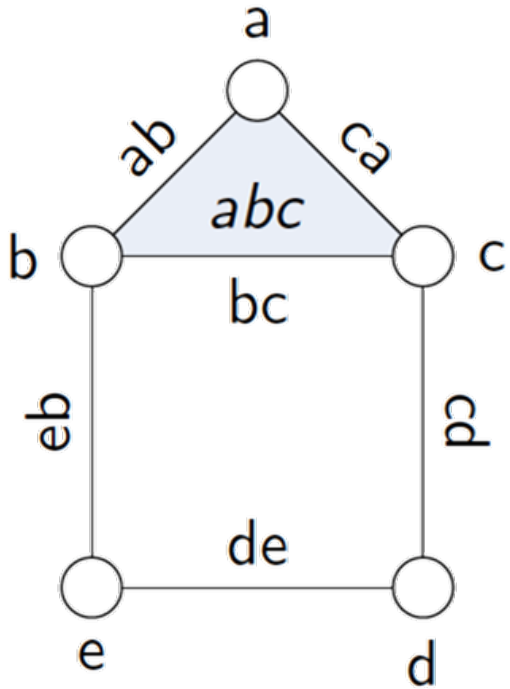
∂_1

$$\begin{array}{c}
 abc \\
 ab \\
 ca \\
 bc \\
 cd \\
 de \\
 eb \\
 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
 \end{array}$$

∂_2

- ◆ Need $\mathbb{Z}/2\mathbb{Z}$ arithmetic
- ◆ Column operations

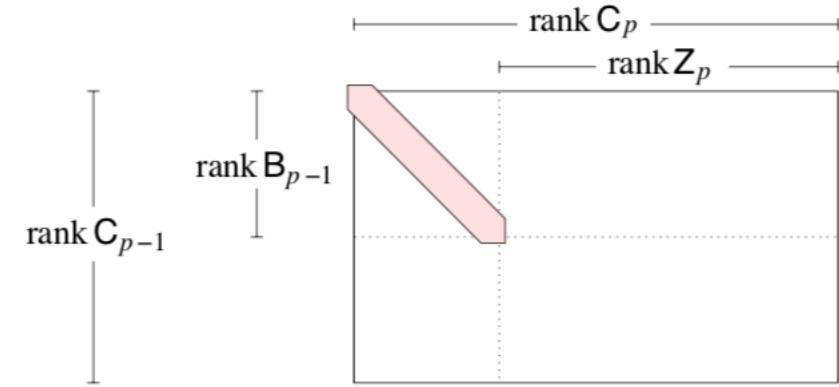
Summary So Far



Smith Normal Form

Standard algorithm for homology:

- ◆ Use row/column operations (Gaussian elimination) on boundary matrices to get form where initial segment of diagonal is 1, everything else 0
- ◆ Read off basis for $\ker \partial_p$ (nullspace) and $\text{im } \partial_{p+1}$ (range)
- ◆ Diagonal form means we have 1-1 correspondence, know which cycles are boundaries
- ◆ We'll see a slightly different method that only uses column operations and goes through unchanged for *persistent* homology



Column operations in $\mathbb{Z}/2\mathbb{Z}$

Arithmetic in $\mathbb{Z}/2\mathbb{Z}$ is very easy:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Reduction Algorithm for Homology*

*Cohen-Steiner, Edelsbrunner, Morozov - Vines and Vineyards by Updating Persistence in Linear Time

To get H_p , we need: (1) a basis for $\ker \partial_p$, and (2) a sub-basis for $\text{im } \partial_{p+1}$.

	ab	ca	bc	cd	de	eb	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	0	1	0
c	0	1	1	1	0	0	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0

R

Notation: $\text{low}_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, \text{low}_R(j') = \text{low}_R(j)$ do
add column j' to column j

Reduction Algorithm for Homology

	<i>ab</i>	<i>ca</i>	<i>bc</i>	<i>cd</i>	<i>de</i>	<i>eb</i>	<i>abc</i>
<i>a</i>	1	1	0	0	0	0	0
<i>b</i>	1	0	1	0	0	1	0
<i>c</i>	0	1	1	1	0	0	0
<i>d</i>	0	0	0	1	1	0	0
<i>e</i>	0	0	0	0	1	1	0
<i>ab</i>	0	0	0	0	0	0	1
<i>ca</i>	0	0	0	0	0	0	1
<i>bc</i>	0	0	0	0	0	0	1
<i>cd</i>	0	0	0	0	0	0	0
<i>de</i>	0	0	0	0	0	0	0
<i>eb</i>	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

	<i>ab</i>	<i>ca</i>	<i>bc</i>	<i>cd</i>	<i>de</i>	<i>eb</i>	<i>abc</i>
<i>a</i>	1	1	0	0	0	0	0
<i>b</i>	1	0	1	0	0	1	0
<i>c</i>	0	1	1	1	0	0	0
<i>d</i>	0	0	0	1	1	0	0
<i>e</i>	0	0	0	0	1	1	0
<i>ab</i>	0	0	0	0	0	0	1
<i>ca</i>	0	0	0	0	0	0	1
<i>bc</i>	0	0	0	0	0	0	1
<i>cd</i>	0	0	0	0	0	0	0
<i>de</i>	0	0	0	0	0	0	0
<i>eb</i>	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

	<i>ab</i>	<i>ca</i>	<i>bc</i>	<i>cd</i>	<i>de</i>	<i>eb</i>	<i>abc</i>
<i>a</i>	1	1	0	0	0	0	0
<i>b</i>	1	0	1	0	0	1	0
<i>c</i>	0	1	1	1	0	0	0
<i>d</i>	0	0	0	1	1	0	0
<i>e</i>	0	0	0	0	1	1	0
<i>ab</i>	0	0	0	0	0	0	1
<i>ca</i>	0	0	0	0	0	0	1
<i>bc</i>	0	0	0	0	0	0	1
<i>cd</i>	0	0	0	0	0	0	0
<i>de</i>	0	0	0	0	0	0	0
<i>eb</i>	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

	ab	ca	$bc + ca$	cd	de	eb	abc
a	1	1	1	0	0	0	0
b	1	0	1	0	0	1	0
c	0	1	0	1	0	0	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	0
$bc + ca$	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0

R

Reminder: to keep bases compatible, column operations require corresponding row operation!

E.g. $T: V \rightarrow W$ linear map, $\{v_1, v_2\}, \{w_1, w_2\}$ bases,
 $T(v_1) = a_{11}w_1 + a_{21}w_2$. Changing codomain basis by
 setting $w_1 \leftarrow w_1, w_2 \leftarrow w_2 + w_1$ means we have:

$$\begin{aligned} T(v_1) &= a_{11}w_1 + a_{21}(w_2 + w_1) - a_{21}w_1 \\ &= (a_{11} - a_{21})w_1 + a_{21}(w_2 + w_1). \end{aligned}$$

I.e. subtract row 2 from row 1 (notice the “reflected” operation). In $\mathbb{Z}/2\mathbb{Z}$, this is just addition!

Reduction Algorithm for Homology

	<i>ab</i>	<i>ca</i>	<i>bc + ca + ab</i>	<i>cd</i>	<i>de</i>	<i>eb</i>	<i>abc</i>
<i>a</i>	1	1	0	0	0	0	0
<i>b</i>	1	0	0	0	0	1	0
<i>c</i>	0	1	0	1	0	0	0
<i>d</i>	0	0	0	1	1	0	0
<i>e</i>	0	0	0	0	1	1	0
<i>ab</i>	0	0	0	0	0	0	0
<i>ca</i>	0	0	0	0	0	0	0
<i>bc + ca + ab</i>	0	0	0	0	0	0	1
<i>cd</i>	0	0	0	0	0	0	0
<i>de</i>	0	0	0	0	0	0	0
<i>eb</i>	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

$$\begin{array}{c}
 a \\
 b \\
 c \\
 d \\
 e \\
 ab \\
 ca \\
 bc + ca + ab \\
 cd \\
 de \\
 eb
 \end{array}
 \begin{array}{c}
 ab \\
 ca \\
 bc + ca + ab \\
 cd \\
 de \\
 eb \\
 abc
 \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}$$

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

	ab	ca	$bc + ca + ab$	cd	de	eb	abc
a	1	1	0	0	0	0	0
b	1	0	0	0	0	1	0
c	0	1	0	1	0	0	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

$$\begin{array}{c}
 a \\
 b \\
 c \\
 d \\
 e \\
 ab \\
 ca \\
 bc + ca + ab \\
 cd \\
 de \\
 eb
 \end{array}
 \begin{array}{c}
 ab \\
 ca \\
 bc + ca + ab \\
 cd \\
 de \\
 eb \\
 abc
 \end{array}
 \begin{bmatrix}
 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}$$

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

	ab	ca	$bc + ca + ab$	cd	de	$eb + de$	abc
a	1	1	0	0	0	0	0
b	1	0	0	0	0	1	0
c	0	1	0	1	0	0	0
d	0	0	0	1	1	1	0
e	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
$eb + de$	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd$	abc
a	1	1	0	0	0	0	0
b	1	0	0	0	0	1	0
c	0	1	0	1	0	1	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
$eb + de + cd$	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd$	abc
a	1	1	0	0	0	0	0
b	1	0	0	0	0	1	0
c	0	1	0	1	0	1	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
$eb + de + cd$	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do
 add column j' to column j

Reduction Algorithm for Homology

	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca$	abc
a	1	1	0	0	0	1	0
b	1	0	0	0	0	1	0
c	0	1	0	1	0	0	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
$eb + de + cd + ca$	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do
 add column j' to column j

Reduction Algorithm for Homology

	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca + ab$	abc
a	1	1	0	0	0	0	0
b	1	0	0	0	0	0	0
c	0	1	0	1	0	0	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
$eb + de + cd + ca + ab$	0	0	0	0	0	0	0

R

Notation: $low_R(j)$ = row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D

Initialize $R := D$

For column $j = 1, 2, \dots, n$

while $\exists j' < j, low_R(j') = low_R(j)$ do

add column j' to column j

Reduction Algorithm for Homology

$$\begin{array}{l}
 a \\
 b \\
 c \\
 d \\
 e \\
 ab \\
 ca \\
 bc + ca + ab \\
 cd \\
 de \\
 eb + de + cd + ca + ab \\
 + ca + ab
 \end{array}
 \begin{array}{c}
 ab \\
 ca \\
 bc + ca + ab \\
 cd \\
 de \\
 eb + de + cd + ca + ab \\
 abc
 \end{array}
 \begin{array}{c}
 \left[\begin{array}{ccccccc}
 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array} \right]
 \end{array}$$

R

Reduction Algorithm for Homology

	a	b	c	d	e	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca + ab$	abc
a	0	0	0	0	0	1	1	0	0	0	0	0
b	0	0	0	0	0	1	0	0	0	0	0	0
c	0	0	0	0	0	0	1	0	1	0	0	0
d	0	0	0	0	0	0	0	0	1	1	0	0
e	0	0	0	0	0	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0	0	0	0	0	0
$eb + de + cd$	0	0	0	0	0	0	0	0	0	0	0	0
$+ ca + ab$	0	0	0	0	0	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

Reduction Algorithm for Homology

	a	b	c	d	e	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca + ab$	abc
a	0	0	0	0	0	1	1	0	0	0	0	0
b	0	0	0	0	0	1	0	0	0	0	0	0
c	0	0	0	0	0	0	1	0	1	0	0	0
d	0	0	0	0	0	0	0	0	1	1	0	0
e	0	0	0	0	0	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0	0	0	0	0	0
$eb + de + cd + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value

Reduction Algorithm for Homology

	a	b	c	d	e	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca + ab$	abc
a	0	0	0	0	0	1	1	0	0	0	0	0
b	0	0	0	0	0	1	0	0	0	0	0	0
c	0	0	0	0	0	0	1	0	1	0	0	0
d	0	0	0	0	0	0	0	0	1	1	0	0
e	0	0	0	0	0	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0	0	0	0	0	0
$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0	0	0	0	0	0
$eb + de + cd + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$

Reduction Algorithm for Homology

		1	2	3	4	5			8		11		
		a	b	c	d	e	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca + ab$	abc
1	a	0	0	0	0	0	1	1	0	0	0	0	0
2	b	0	0	0	0	0	1	0	0	0	0	0	0
3	c	0	0	0	0	0	0	1	0	1	0	0	0
4	d	0	0	0	0	0	0	0	0	1	1	0	0
5	e	0	0	0	0	0	0	0	0	0	1	0	0
	ab	0	0	0	0	0	0	0	0	0	0	0	0
	ca	0	0	0	0	0	0	0	0	0	0	0	0
8	$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	1
	cd	0	0	0	0	0	0	0	0	0	0	0	0
	de	0	0	0	0	0	0	0	0	0	0	0	0
	$eb + de + cd$	0	0	0	0	0	0	0	0	0	0	0	0
11	$+ ca + ab$	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$
- (Proposition) Indices i such that column i is zero but $i \neq low_R(j)$ for any j correspond exactly to the cycles that are not boundaries!
- Thus we get a basis for homology in each dimension.

Reduction Algorithm for Homology

	1	2	3	4	5			8		11	
1	a					a					
2	b					ab					
3	c					ca					
4	d										
5	e										
	ab										
	ca										
8	$bc + ca + ab$										
	cd										
	de										
	$eb + de + cd$										
11	$+ ca + ab$										
	abc										

Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$
- (Proposition) Indices i such that column i is zero but $i \neq low_R(j)$ for any j correspond exactly to the cycles that are not boundaries!
- Thus we get a basis for homology in each dimension.

Reduction Algorithm for Homology

$$H_0 \cong \mathbb{F}[a]$$

$$H_1 \cong \mathbb{F}[eb + de + cd + ca + ab]$$

		1	2	3	4	5			8		11		
		a	b	c	d	e	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca + ab$	abc
1	a	0	0	0	0	0	1	1	0	0	0	0	0
2	b	0	0	0	0	0	1	0	0	0	0	0	0
3	c	0	0	0	0	0	0	1	0	1	0	0	0
4	d	0	0	0	0	0	0	0	0	1	1	0	0
5	e	0	0	0	0	0	0	0	0	0	1	0	0
	ab	0	0	0	0	0	0	0	0	0	0	0	0
	ca	0	0	0	0	0	0	0	0	0	0	0	0
8	$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	1
	cd	0	0	0	0	0	0	0	0	0	0	0	0
	de	0	0	0	0	0	0	0	0	0	0	0	0
	$eb + de + cd$	0	0	0	0	0	0	0	0	0	0	0	0
11	$+ ca + ab$	0	0	0	0	0	0	0	0	0	0	0	0
	abc	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

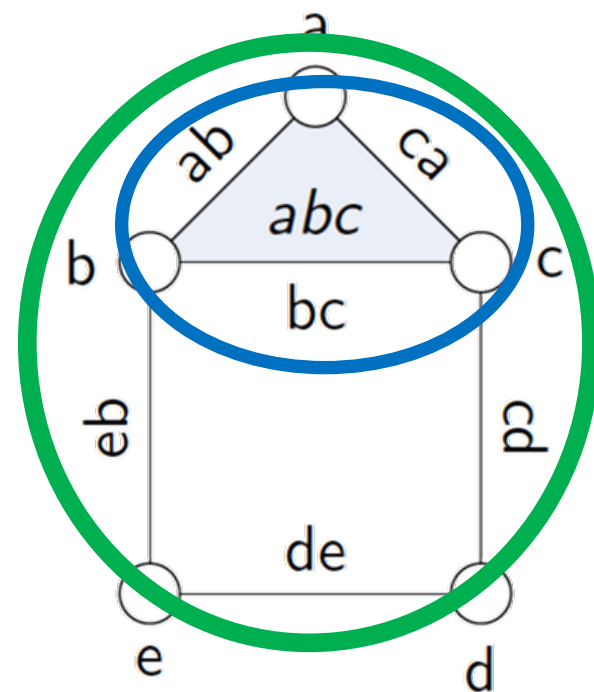
- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$
- (Proposition) Indices i such that column i is zero but $i \neq low_R(j)$ for any j correspond exactly to the cycles that are not boundaries!
- Thus we get a basis for homology in each dimension.

Reduction Algorithm for Homology

$$H_0 \cong \mathbb{F}[a]$$

$$H_1 \cong \mathbb{F}[eb + de + cd + ca + ab]$$

	1	2	3	4	5			8		11		
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>ab</i>	<i>ca</i>	<i>bc + ca + ab</i>	<i>cd</i>	<i>de</i>	<i>eb + de + cd + ca + ab</i>	<i>abc</i>
1	<i>a</i>	0	0	0	0	1	1	0	0	0	0	0
2	<i>b</i>	0	0	0	0	1	0	0	0	0	0	0
3	<i>c</i>	0	0	0	0	0	1	0	1	0	0	0
4	<i>d</i>	0	0	0	0	0	0	0	1	1	0	0
5	<i>e</i>	0	0	0	0	0	0	0	0	1	0	0
	<i>ab</i>	0	0	0	0	0	0	0	0	0	0	0
	<i>ca</i>	0	0	0	0	0	0	0	0	0	0	0
8	<i>bc + ca + ab</i>	0	0	0	0	0	0	0	0	0	0	1
	<i>cd</i>	0	0	0	0	0	0	0	0	0	0	0
	<i>de</i>	0	0	0	0	0	0	0	0	0	0	0
	<i>eb + de + cd</i>	0	0	0	0	0	0	0	0	0	0	0
11	<i>+ ca + ab</i>	0	0	0	0	0	0	0	0	0	0	0
	<i>abc</i>	0	0	0	0	0	0	0	0	0	0	0

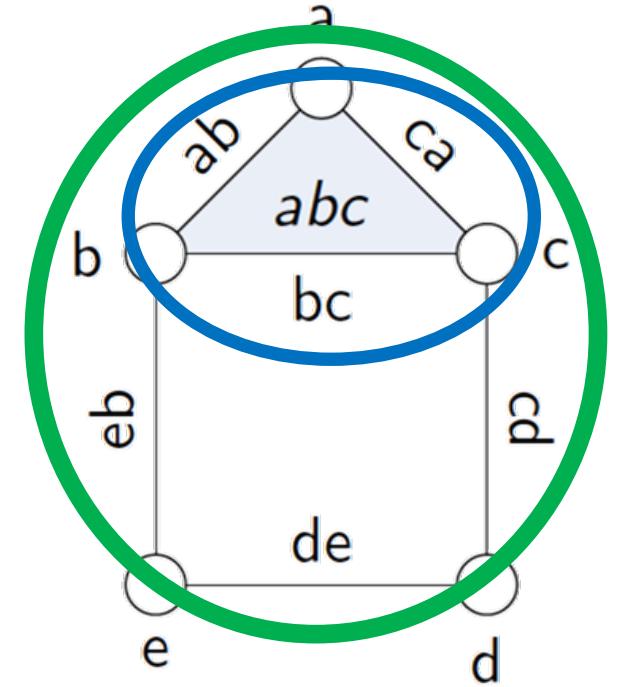


Basis Selection

$$H_0 \cong \mathbb{F}[a]$$

$$H_1 \cong \mathbb{F}[eb + de + cd + ca + ab]$$

	1	2	3	4	5			8		11		
	a	b	c	d	e	ab	ca	$bc + ca + ab$	cd	de	$eb + de + cd + ca + ab$	abc
1	a	0	0	0	0	1	1	0	0	0	0	0
2	b	0	0	0	0	1	0	0	0	0	0	0
3	c	0	0	0	0	0	1	0	1	0	0	0
4	d	0	0	0	0	0	0	0	1	1	0	0
5	e	0	0	0	0	0	0	0	0	1	0	0
	ab	0	0	0	0	0	0	0	0	0	0	0
	ca	0	0	0	0	0	0	0	0	0	0	0
8	$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	1
	cd	0	0	0	0	0	0	0	0	0	0	0
	de	0	0	0	0	0	0	0	0	0	0	0
	$eb + de + cd$	0	0	0	0	0	0	0	0	0	0	0
11	$+ ca + ab$	0	0	0	0	0	0	0	0	0	0	0
	abc	0	0	0	0	0	0	0	0	0	0	0



Caveat: homology involves an equivalence class, so we get “representative cycles”. E.g. we got $eb + de + cd + ca + ab$ instead of $eb + de + cd + bc$.

Operating at the algebraic level gives up some control over geometric correspondence.

Persistent homology

Reduction Algorithm for Persistent Homology

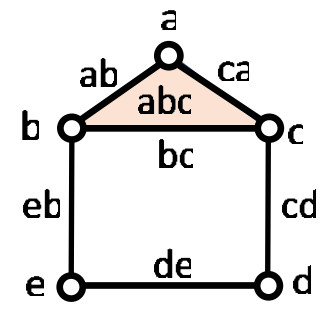
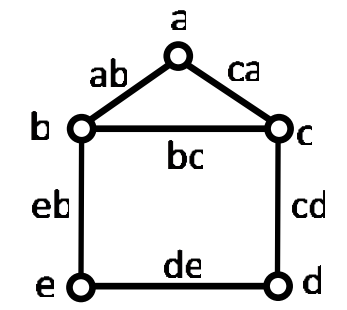
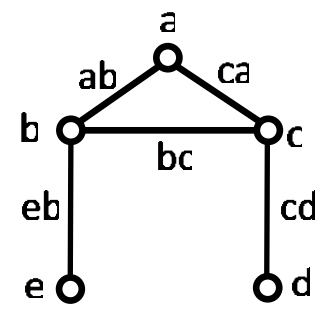
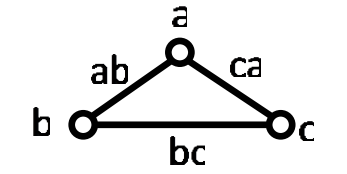
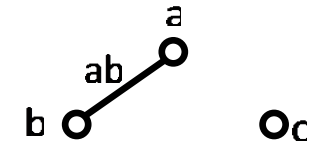
Ulrich Bauer, *Ripser: efficient computation of Vietoris–Rips persistence barcodes*

Persistence algorithm:

- ◆ When setting up boundary matrices/overall incidence matrix, order basis vectors according to order of appearance of simplices
- ◆ Apply reduction algorithm from before
- ◆ Read off **persistence pairs** $\{(i, j) \mid i = \text{low}_R(j) \neq 0\}$ and **essential indices** $\{i \mid R[:, i] = 0, i \neq \text{low}_R(j) \text{ for any } j\}$
- ◆ Persistence pairs track representative cycles that are **born** at index i and **die** at index j
- ◆ Essential indices track cycles that are born at index i and live forever
- ◆ Complexity: Gaussian elimination, so essentially **cubic** in the number of simplices (state-of-the-art: matrix multiplication time)

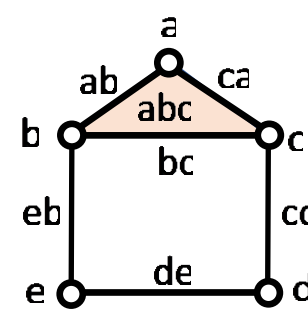
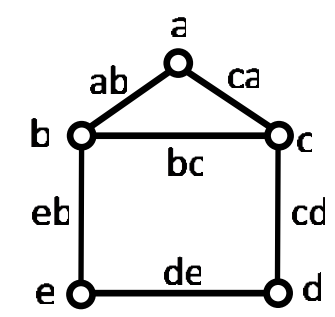
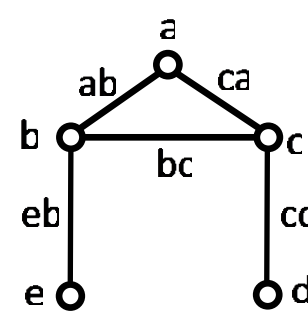
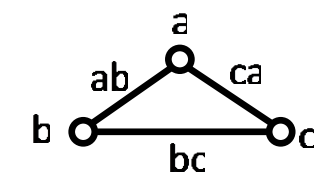
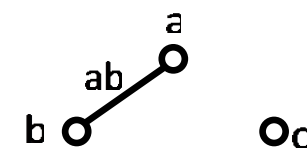
Back to the House Example

Entry time:	1	1	1	1	1	1	2	2	3	3	4	5
Index:	1	2	3	4	5	6	7	8	9	10	11	12
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>ab</i>	<i>ca</i>	<i>bc</i>	<i>cd</i>	<i>eb</i>	<i>de</i>	<i>abc</i>
<i>a</i>	0	0	0	0	0	1	1	0	0	0	0	0
<i>b</i>	0	0	0	0	0	1	0	1	0	1	0	0
<i>c</i>	0	0	0	0	0	0	1	1	1	0	0	0
<i>d</i>	0	0	0	0	0	0	0	0	1	0	1	0
<i>e</i>	0	0	0	0	0	0	0	0	0	1	1	0
<i>ab</i>	0	0	0	0	0	0	0	0	0	0	0	1
<i>ca</i>	0	0	0	0	0	0	0	0	0	0	0	1
<i>bc</i>	0	0	0	0	0	0	0	0	0	0	0	1
<i>cd</i>	0	0	0	0	0	0	0	0	0	0	0	0
<i>eb</i>	0	0	0	0	0	0	0	0	0	0	0	0
<i>de</i>	0	0	0	0	0	0	0	0	0	0	0	0
<i>abc</i>	0	0	0	0	0	0	0	0	0	0	0	0



Back to the House Example

	1	2	3	4	5	<i>ab</i>	<i>ca</i>	8	11	<i>de + eb + cd + ca + ab</i>	<i>abc</i>
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>			<i>bc + ca + ab</i>	<i>cd</i>	<i>eb</i>	
1	0	0	0	0	0	1	1	0	0	0	0
2	0	0	0	0	0	1	0	0	0	1	0
3	0	0	0	0	0	0	1	0	1	0	0
4	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	1	0
<i>ab</i>	0	0	0	0	0	0	0	0	0	0	0
<i>ca</i>	0	0	0	0	0	0	0	0	0	0	0
8 <i>bc + ca + ab</i>	0	0	0	0	0	0	0	0	0	0	1
<i>cd</i>	0	0	0	0	0	0	0	0	0	0	0
<i>eb</i>	0	0	0	0	0	0	0	0	0	0	0
11 <i>de + eb + cd</i>	0	0	0	0	0	0	0	0	0	0	0
+ <i>ca + ab</i>	0	0	0	0	0	0	0	0	0	0	0
<i>abc</i>	0	0	0	0	0	0	0	0	0	0	0



Back to the House Example

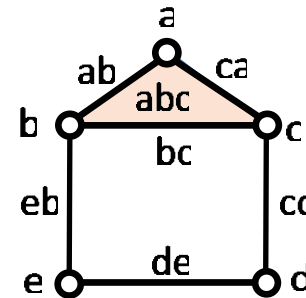
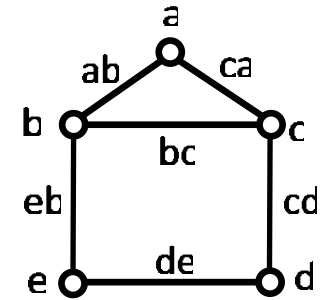
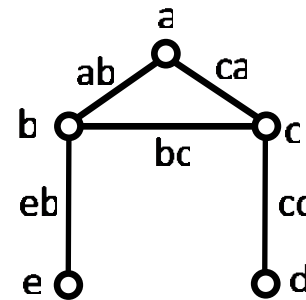
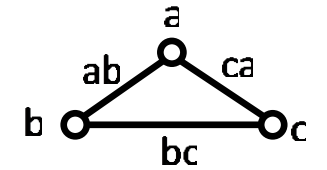
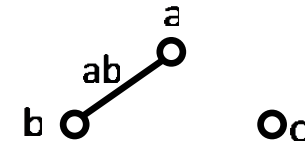
Persistence pairs: $\{(2,6), (3,7), (4,9), (5,10), (8,12)\}$
 (column indices)

Lifetimes (dim 0): $\{[1,1), [1,2), [1,3), [1,3)\}$
 (time indices)

Lifetimes (dim 1): $\{[2,5)\}$
 (time indices)

Essential indices: $\{1, 11\}$
 (column indices)

Lifetimes: $\{[1, \infty)$ (connected component), $[4, \infty)$ (loop) $\}$
 (time indices)

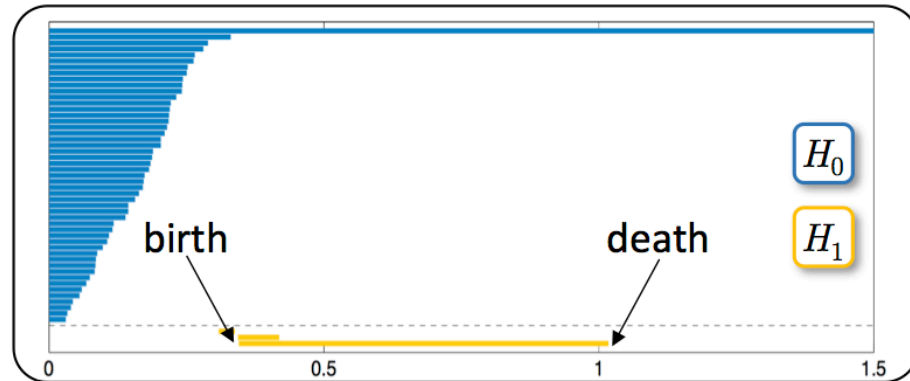
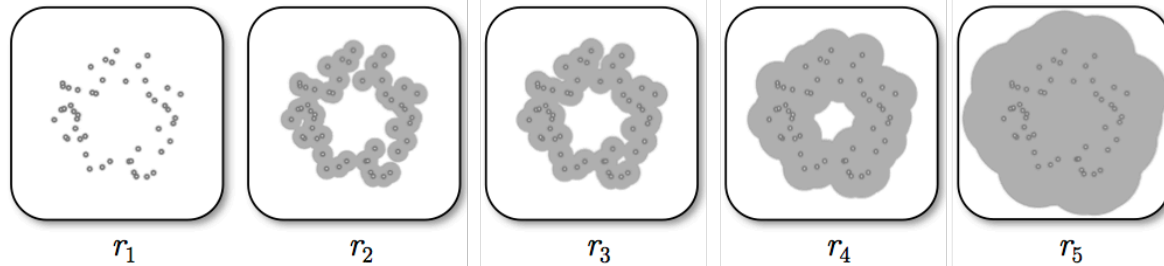


Simplices are “Creators” or “Destroyers”

- ◆ Every simplex either creates a homology class or destroys one
- ◆ Specifically, a k -simplex either creates a k -dimensional feature or destroys a $(k - 1)$ -dimensional feature

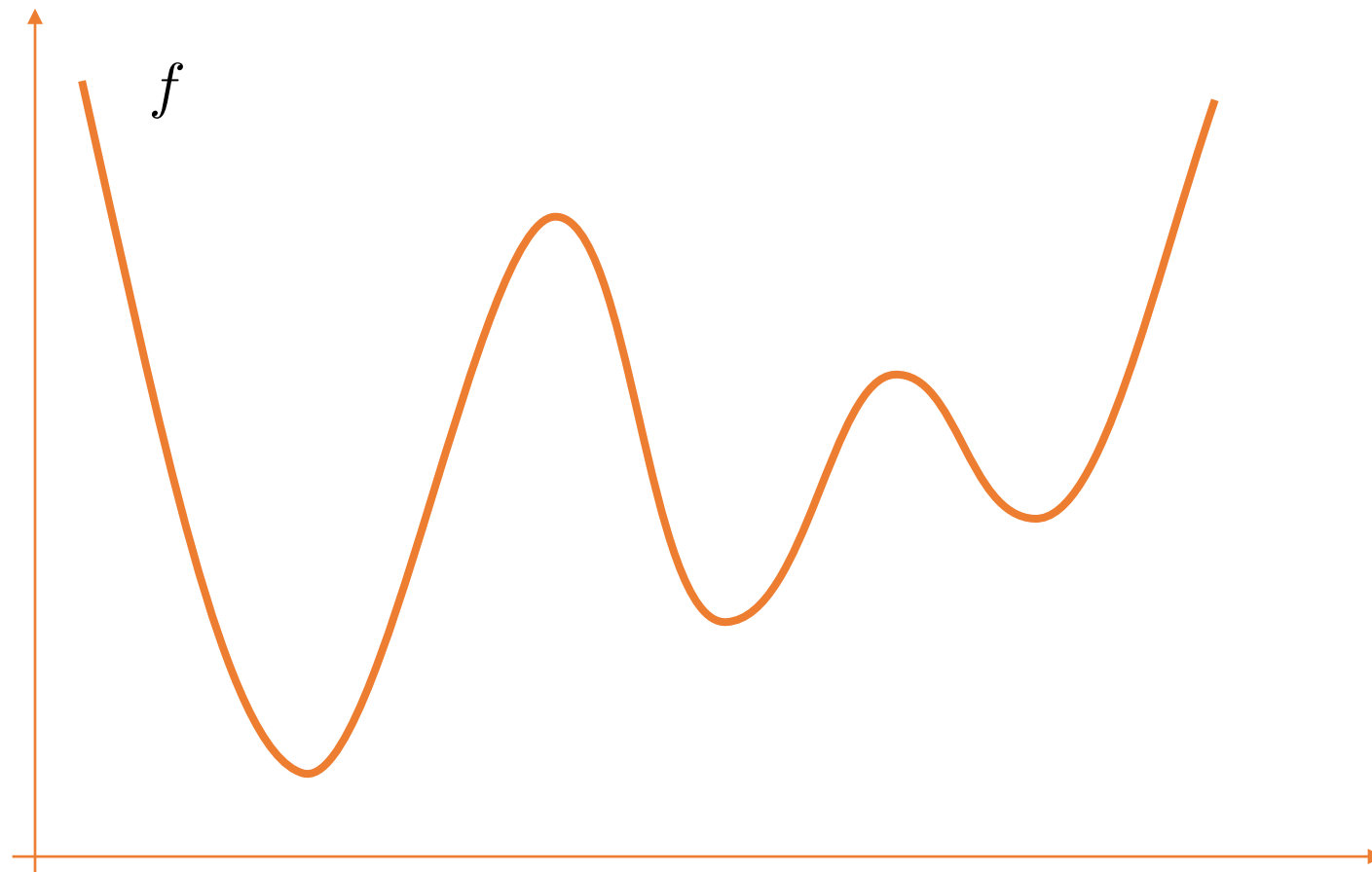
Sublevel Set Persistence

- For general point cloud or metric data, we build filtered simplicial complexes using the metric as a filter

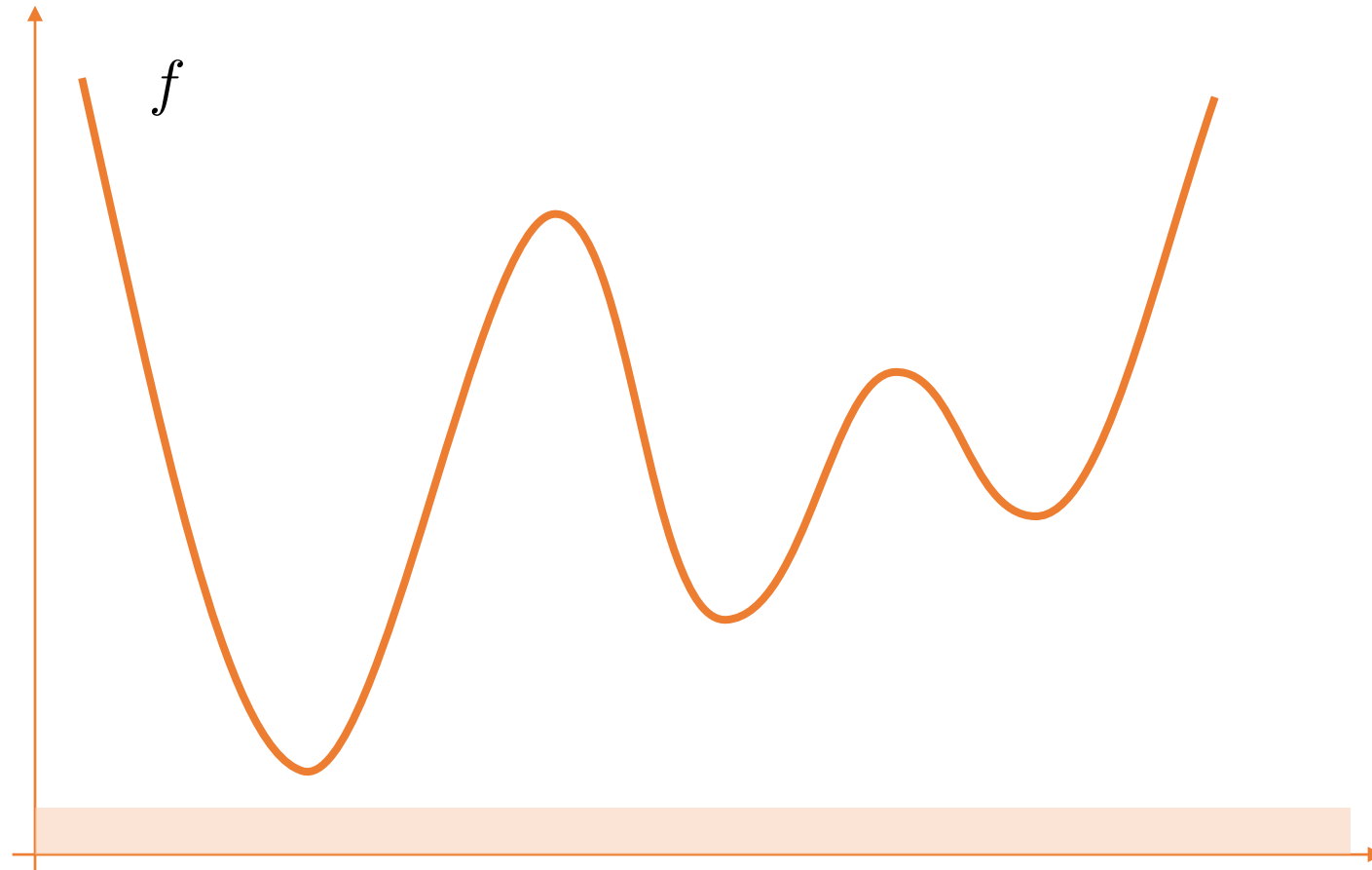


- When given a function $f: \mathcal{M} \rightarrow \mathbb{R}$ where \mathcal{M} is a triangulated space (i.e. a simplicial complex at the outset), we can use f as a filter

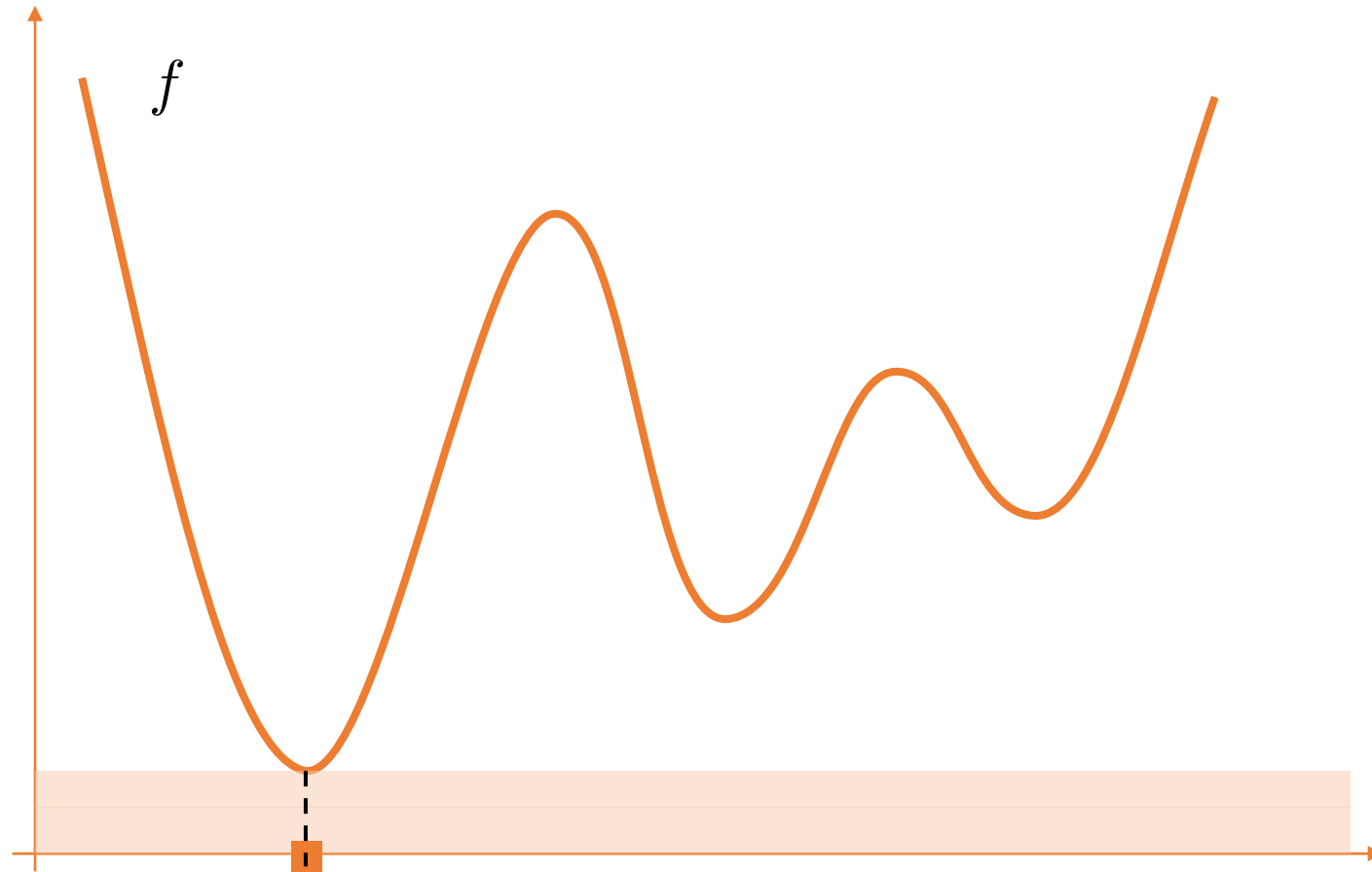
Sublevel Set Persistence



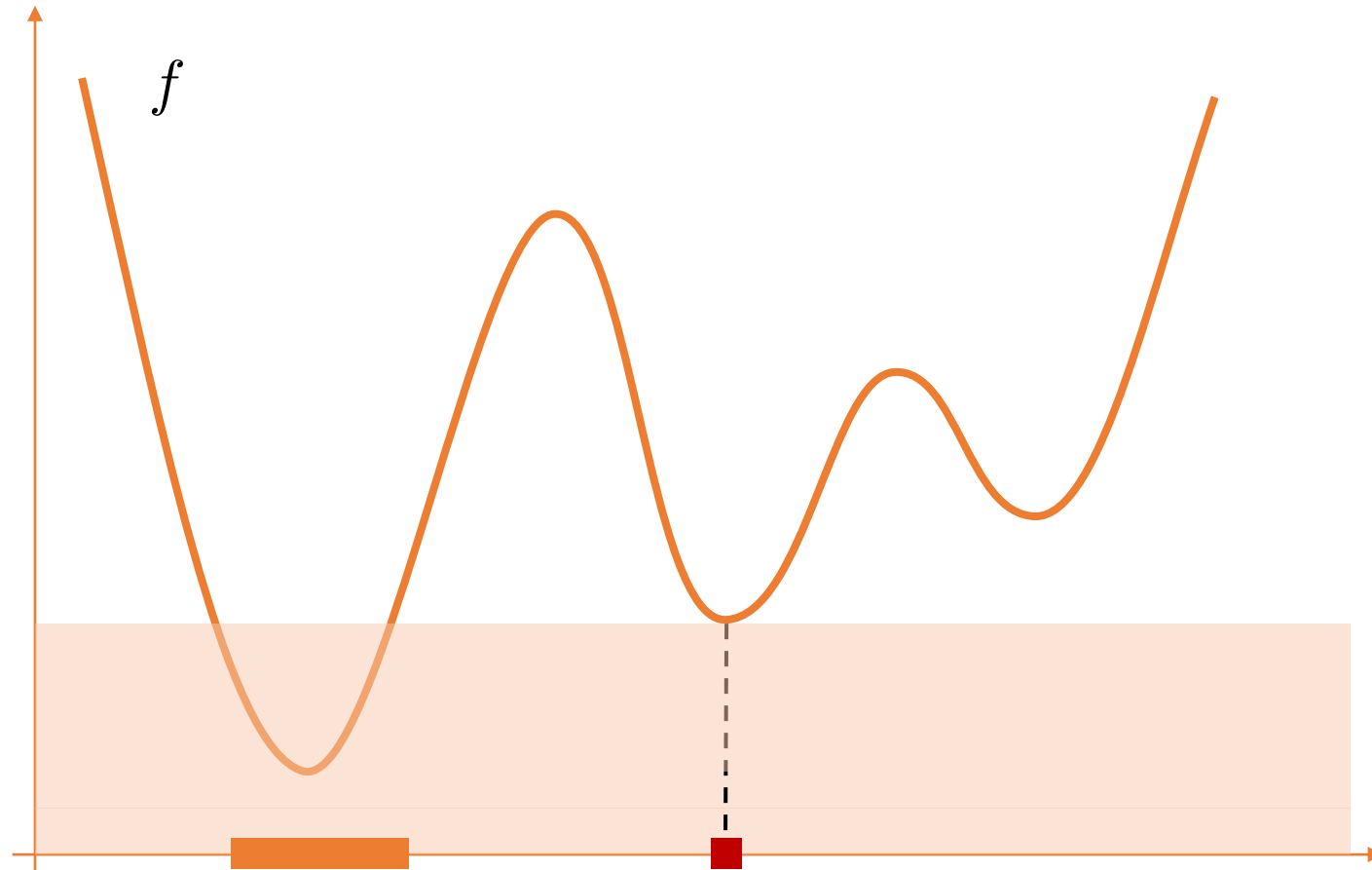
Sublevel Set Persistence



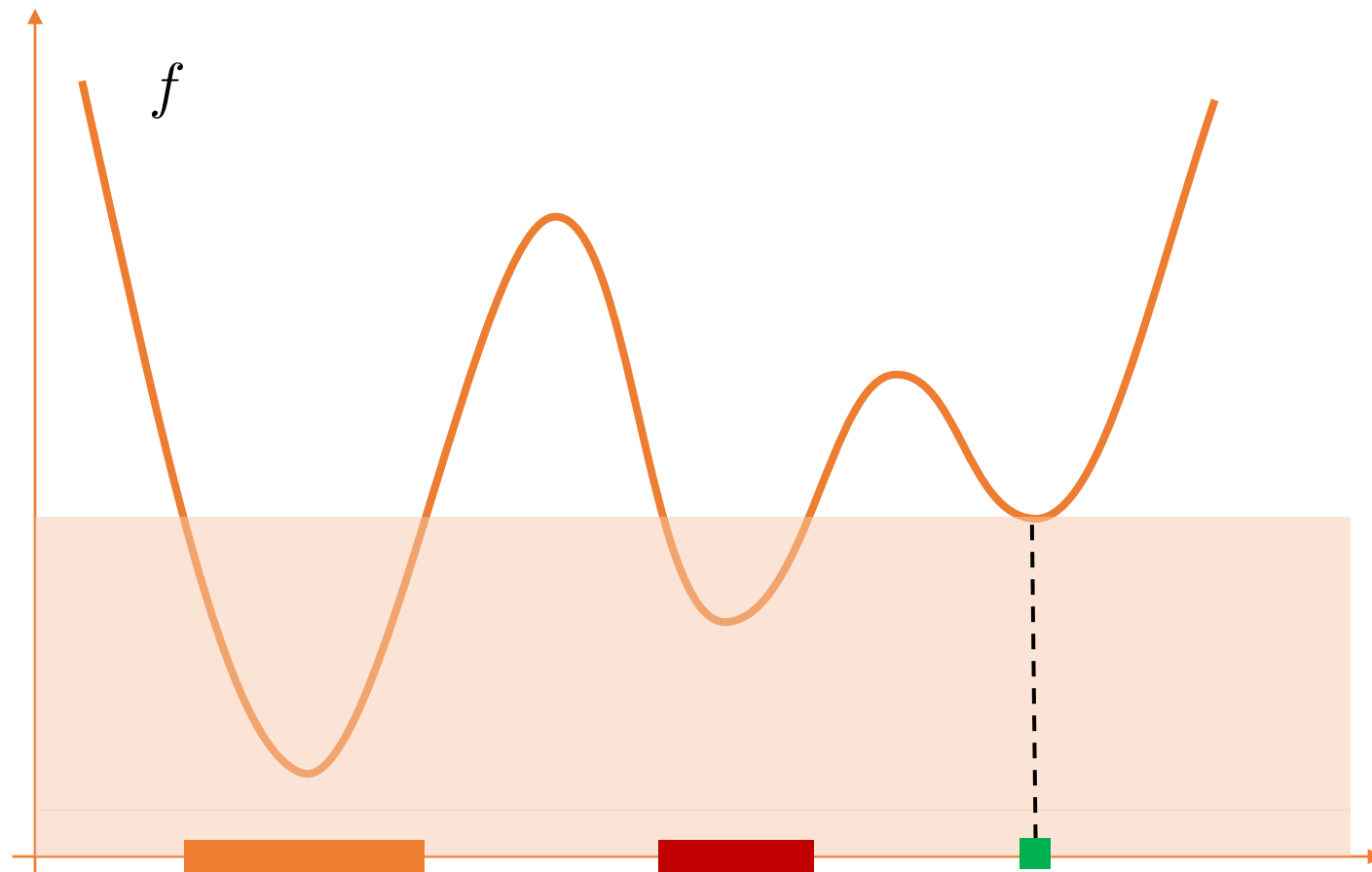
Sublevel Set Persistence



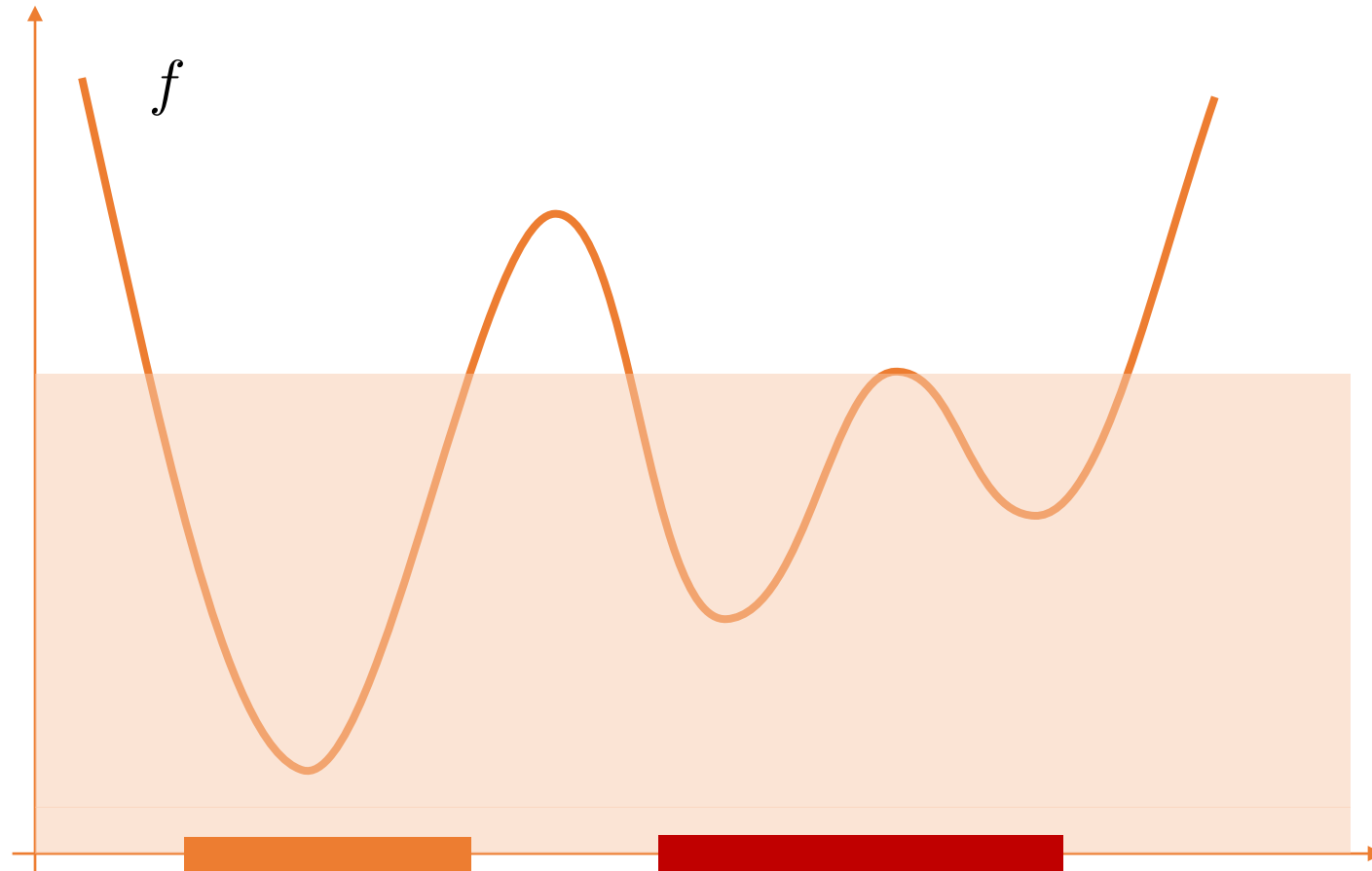
Sublevel Set Persistence



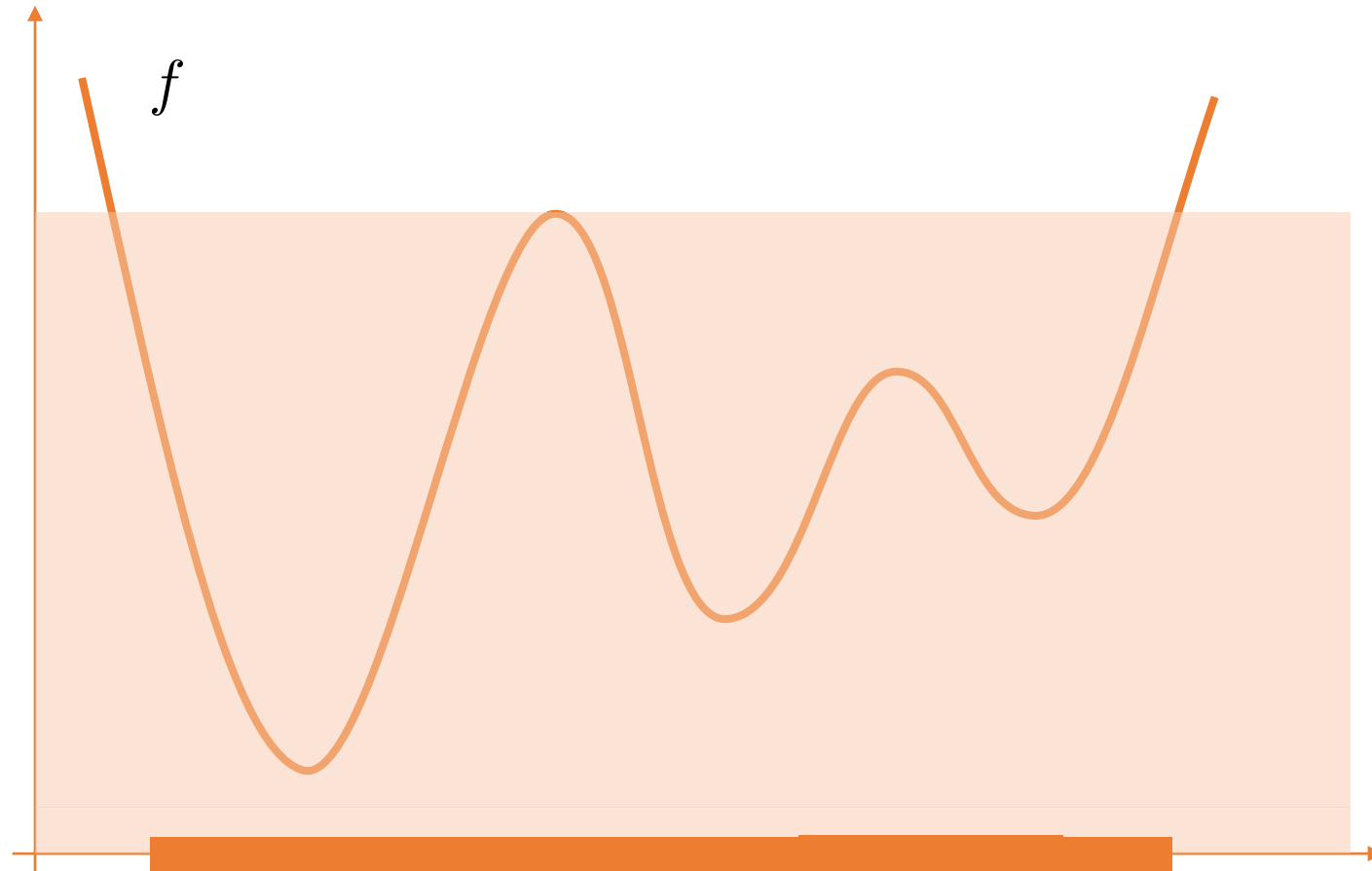
Sublevel Set Persistence



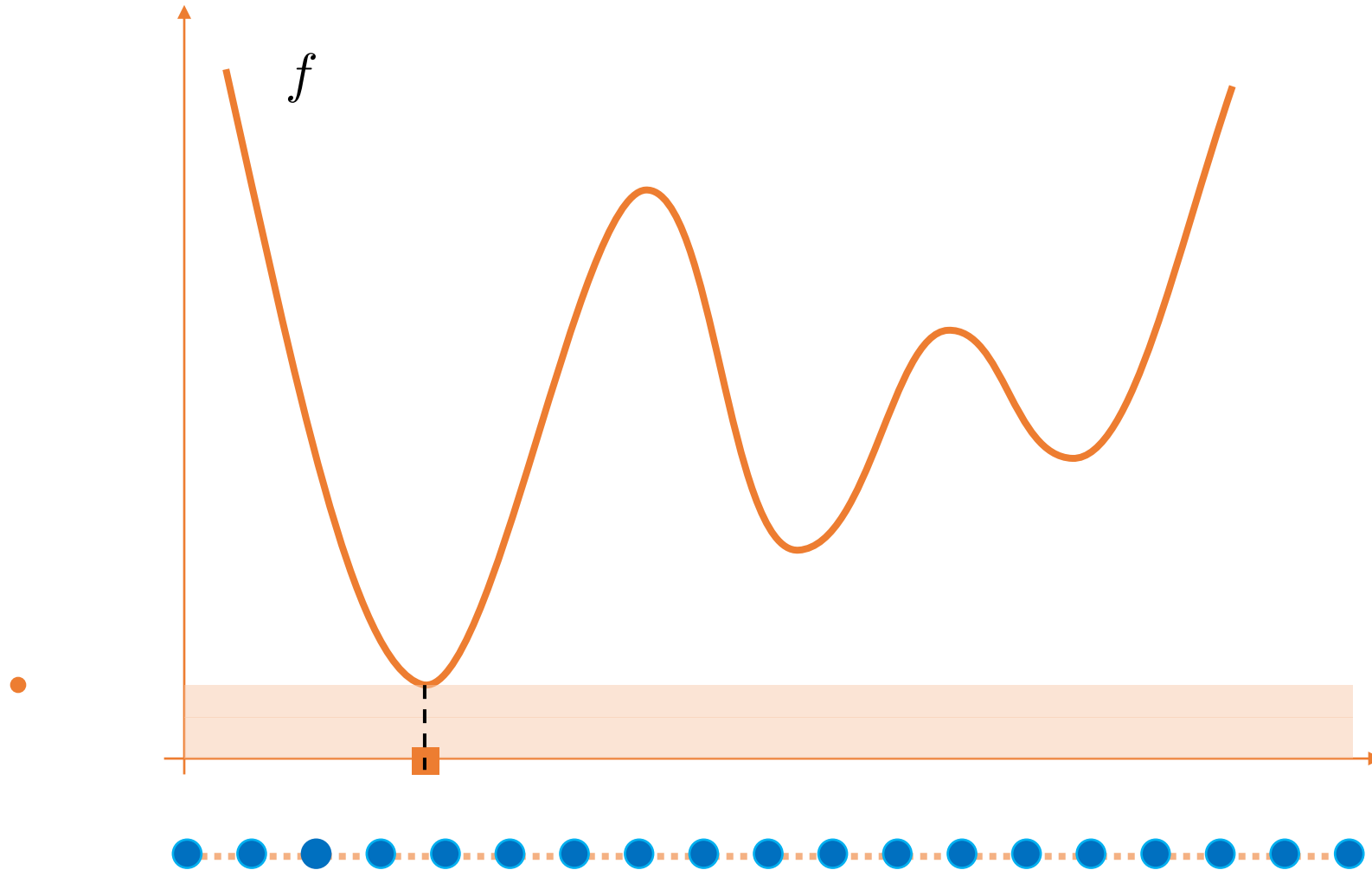
Sublevel Set Persistence



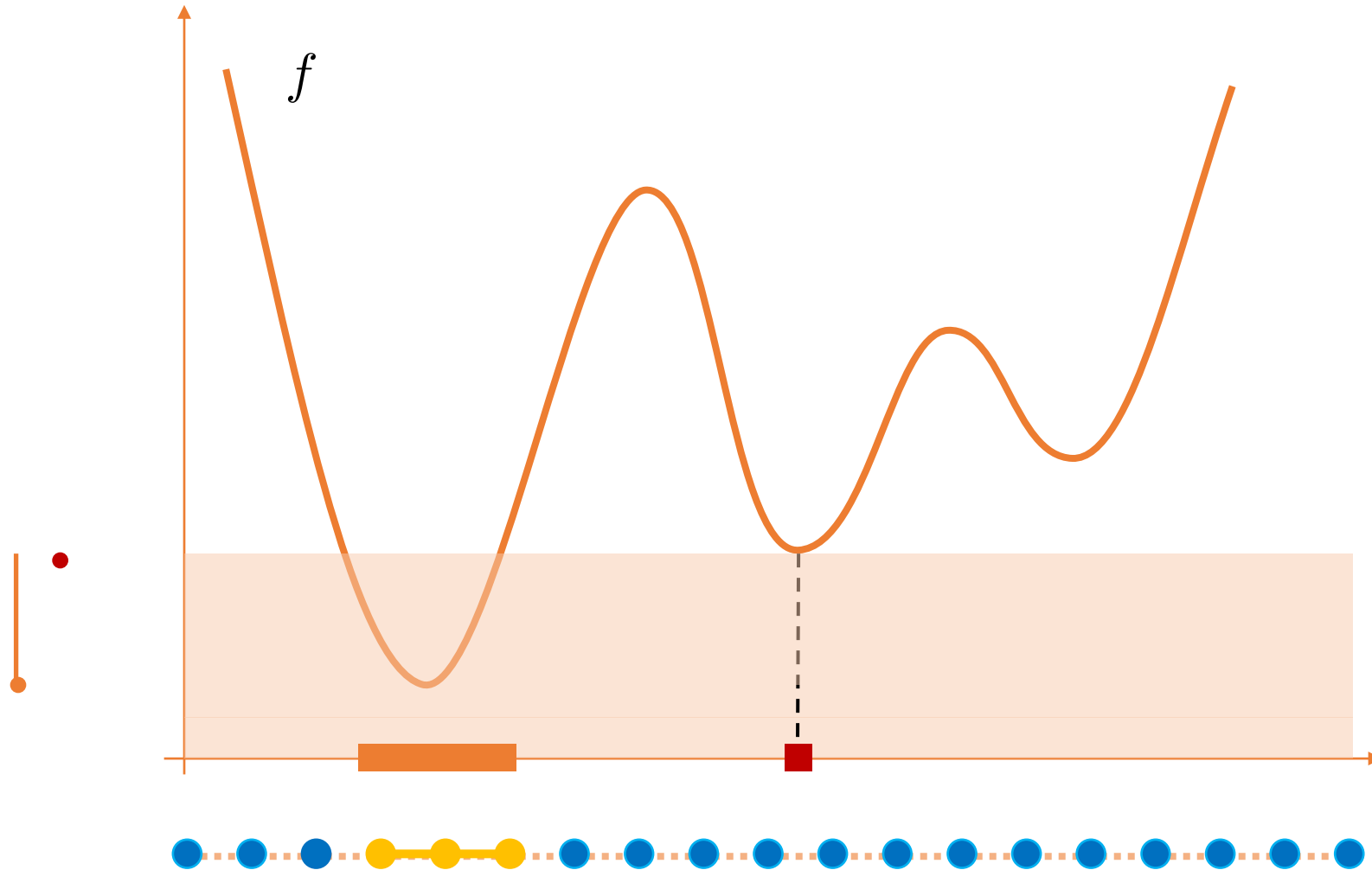
Sublevel Set Persistence



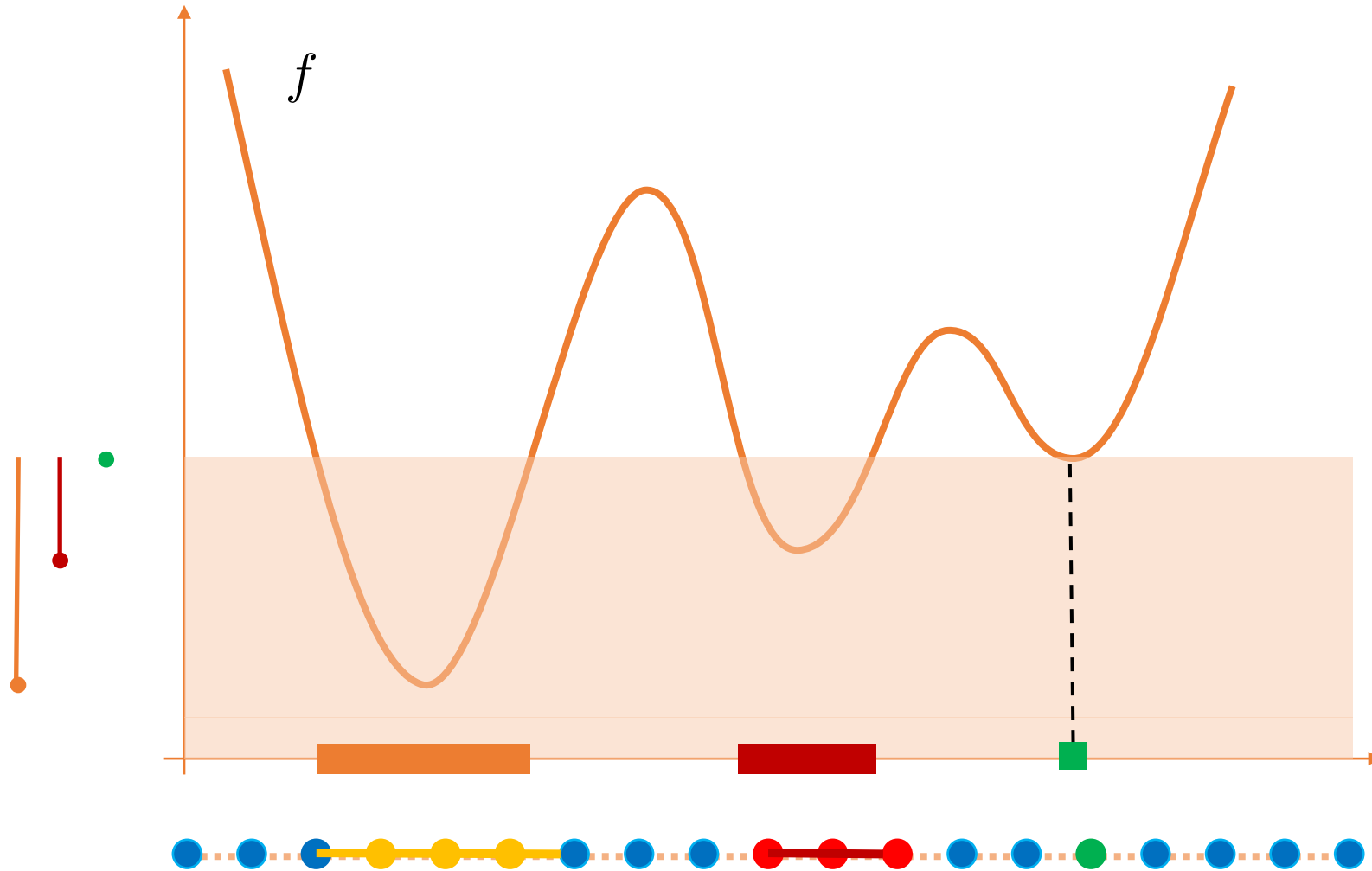
Sublevel Set Persistence



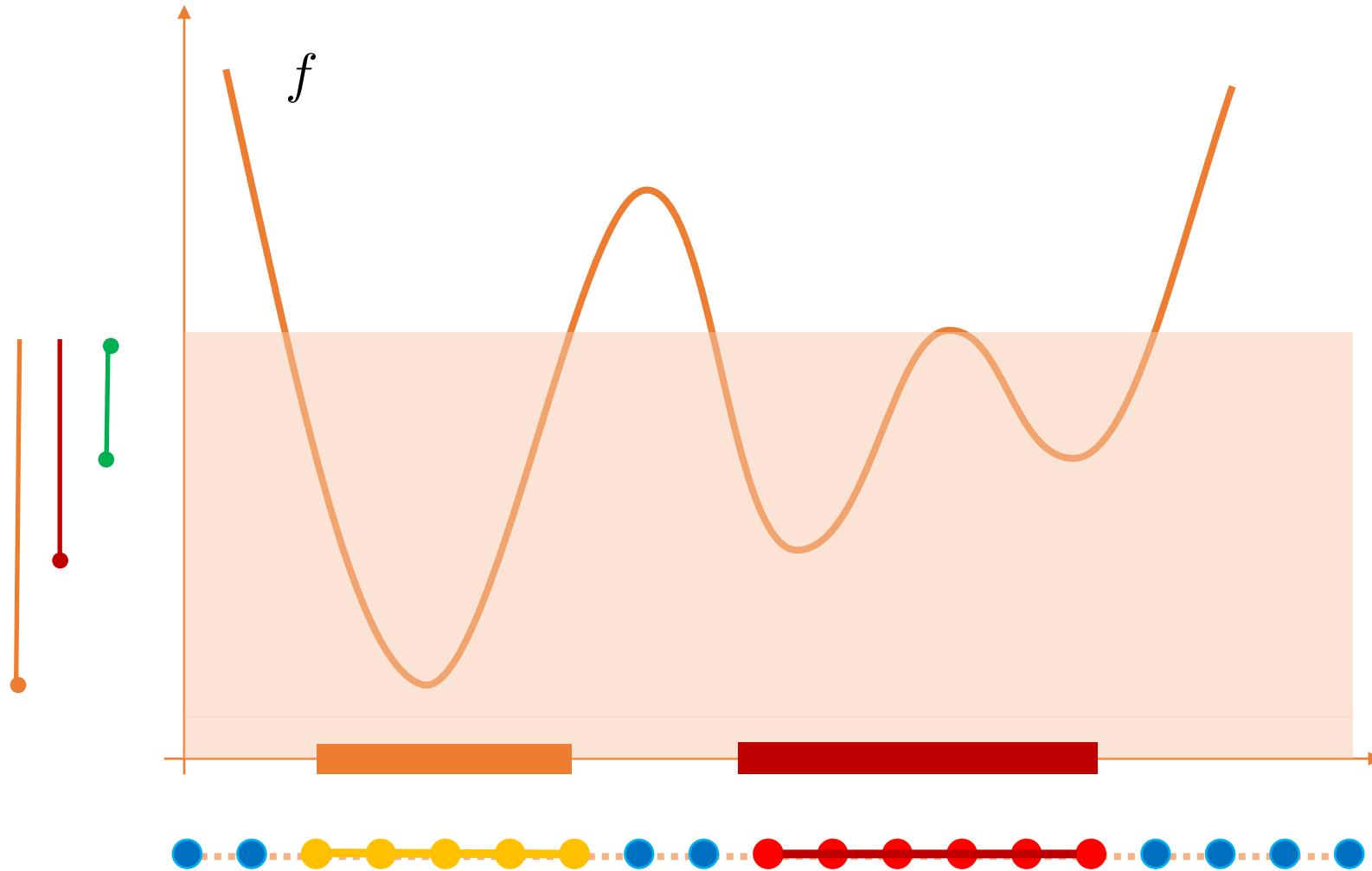
Sublevel Set Persistence



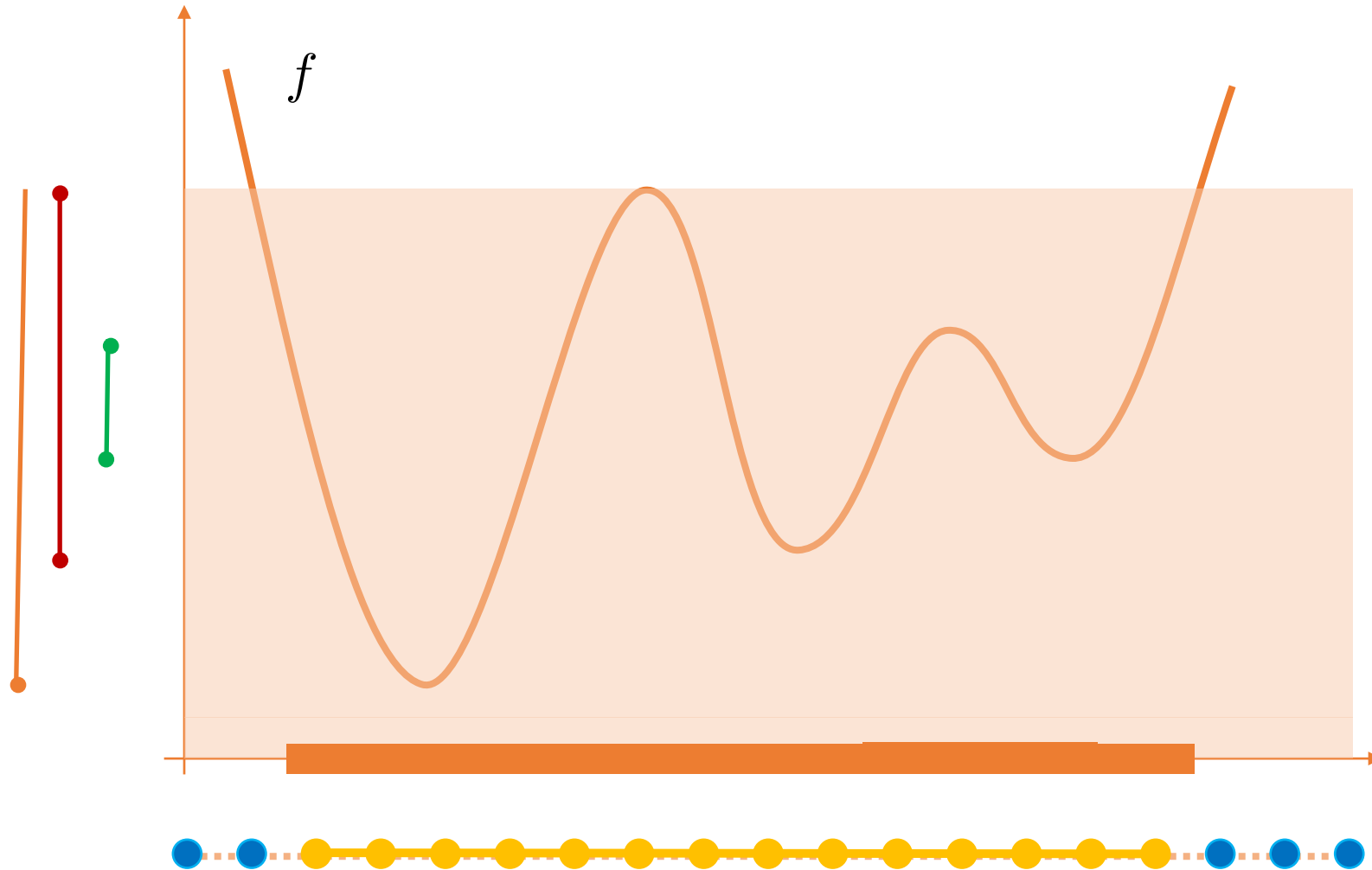
Sublevel Set Persistence



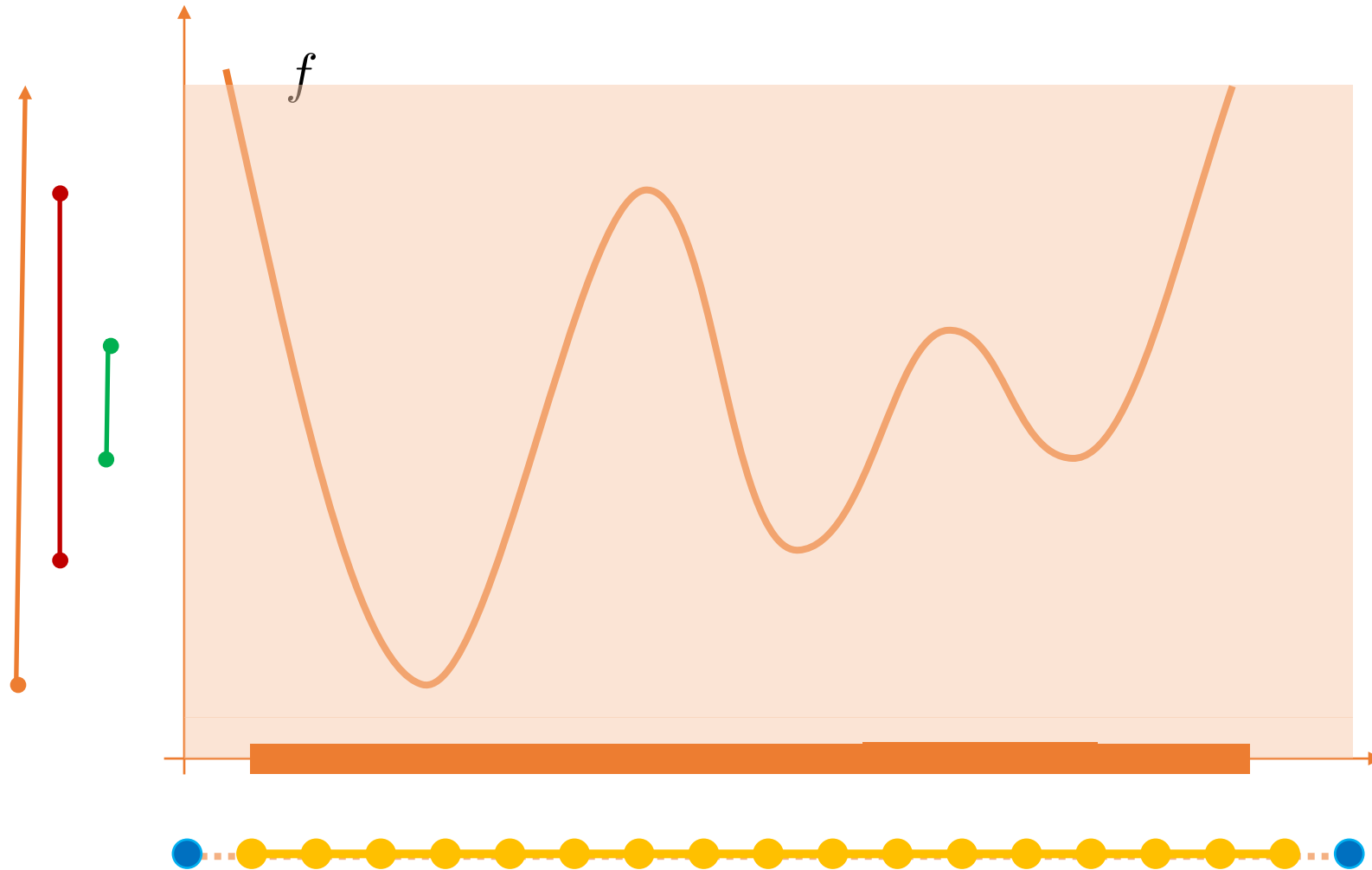
Sublevel Set Persistence



Sublevel Set Persistence



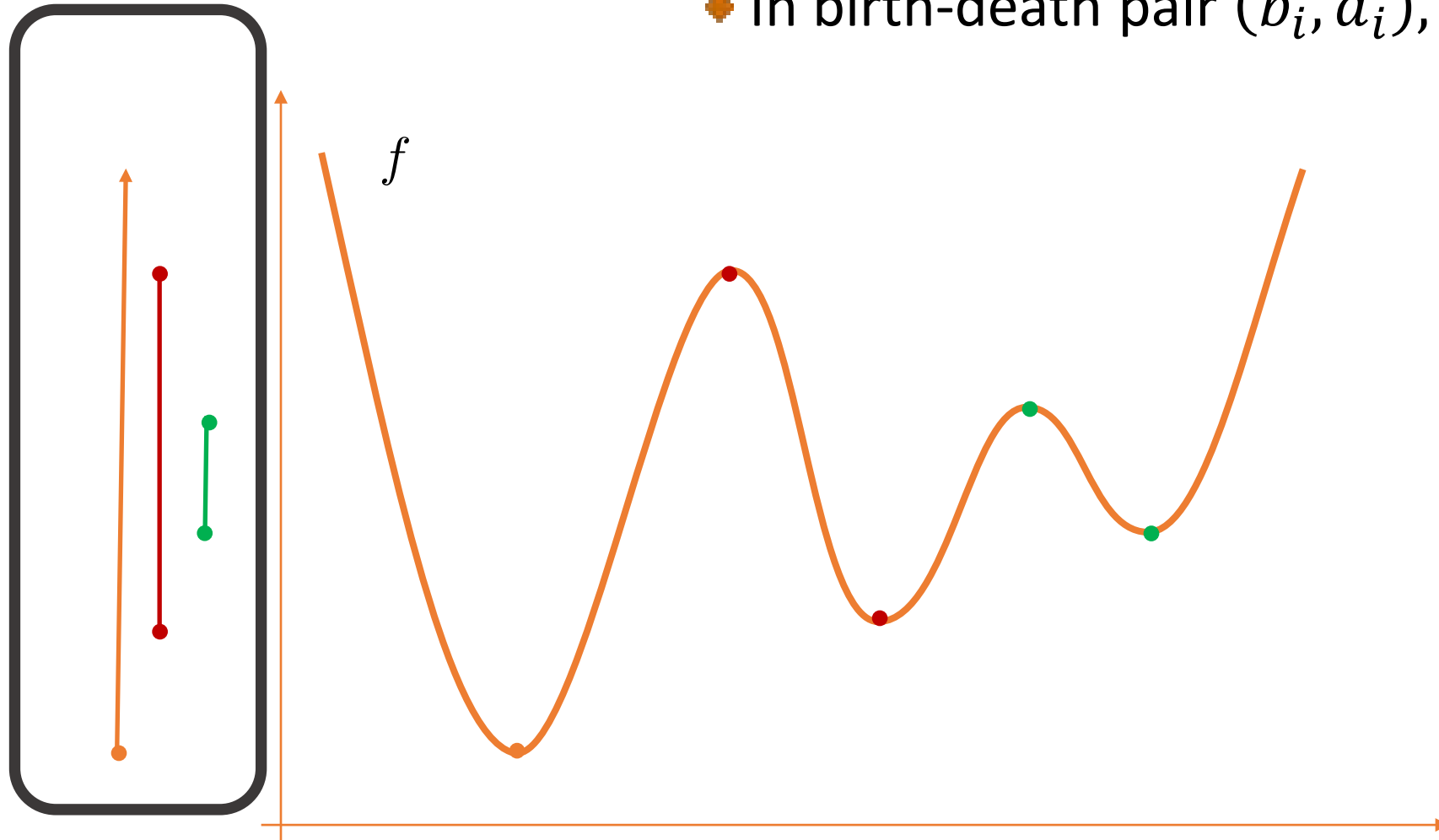
Sublevel Set Persistence



Persistence Barcodes and Diagrams

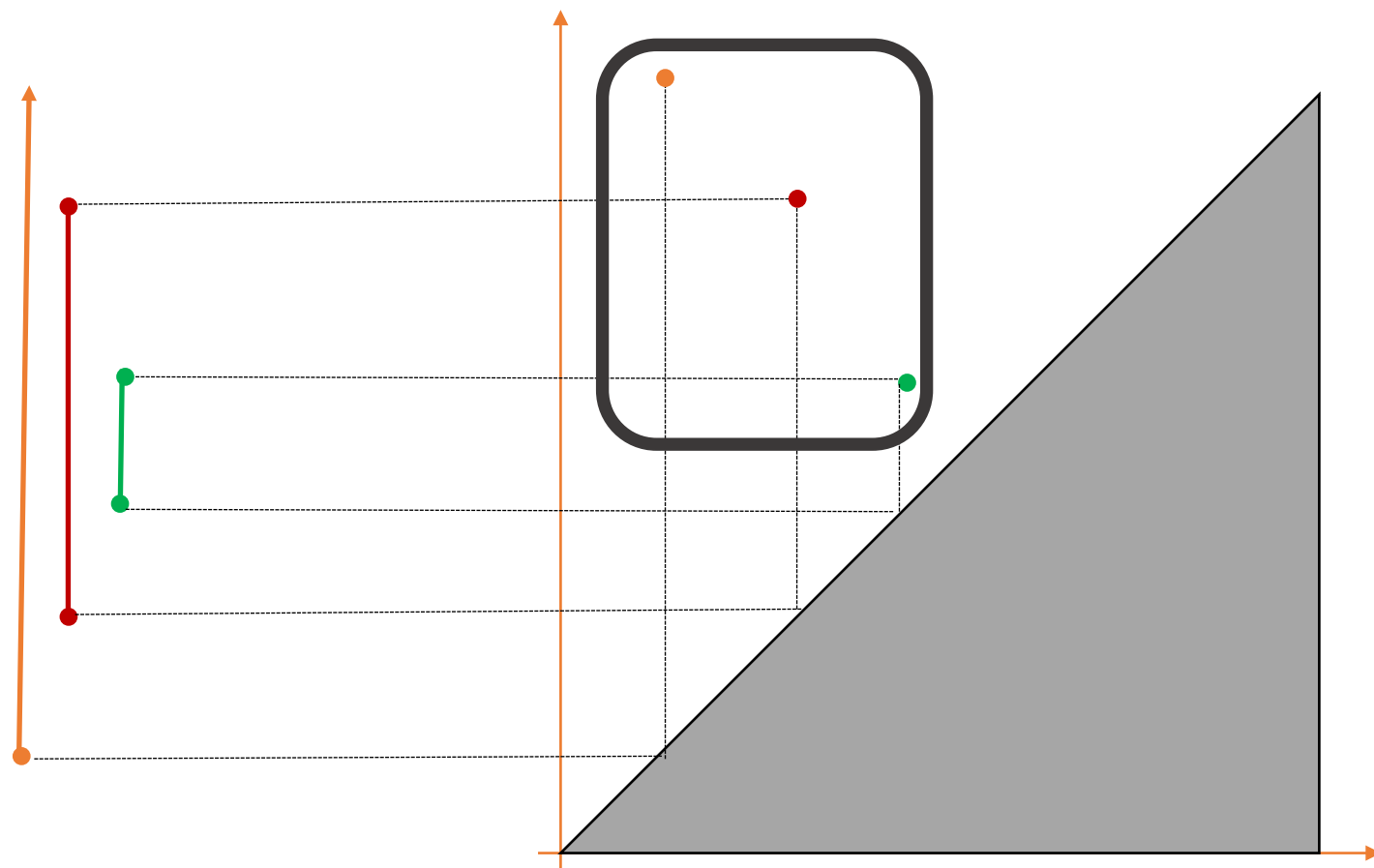
Persistence Pairing Yields a Barcode

◆ In birth-death pair (b_i, d_i) , we have $d_i \geq b_i$

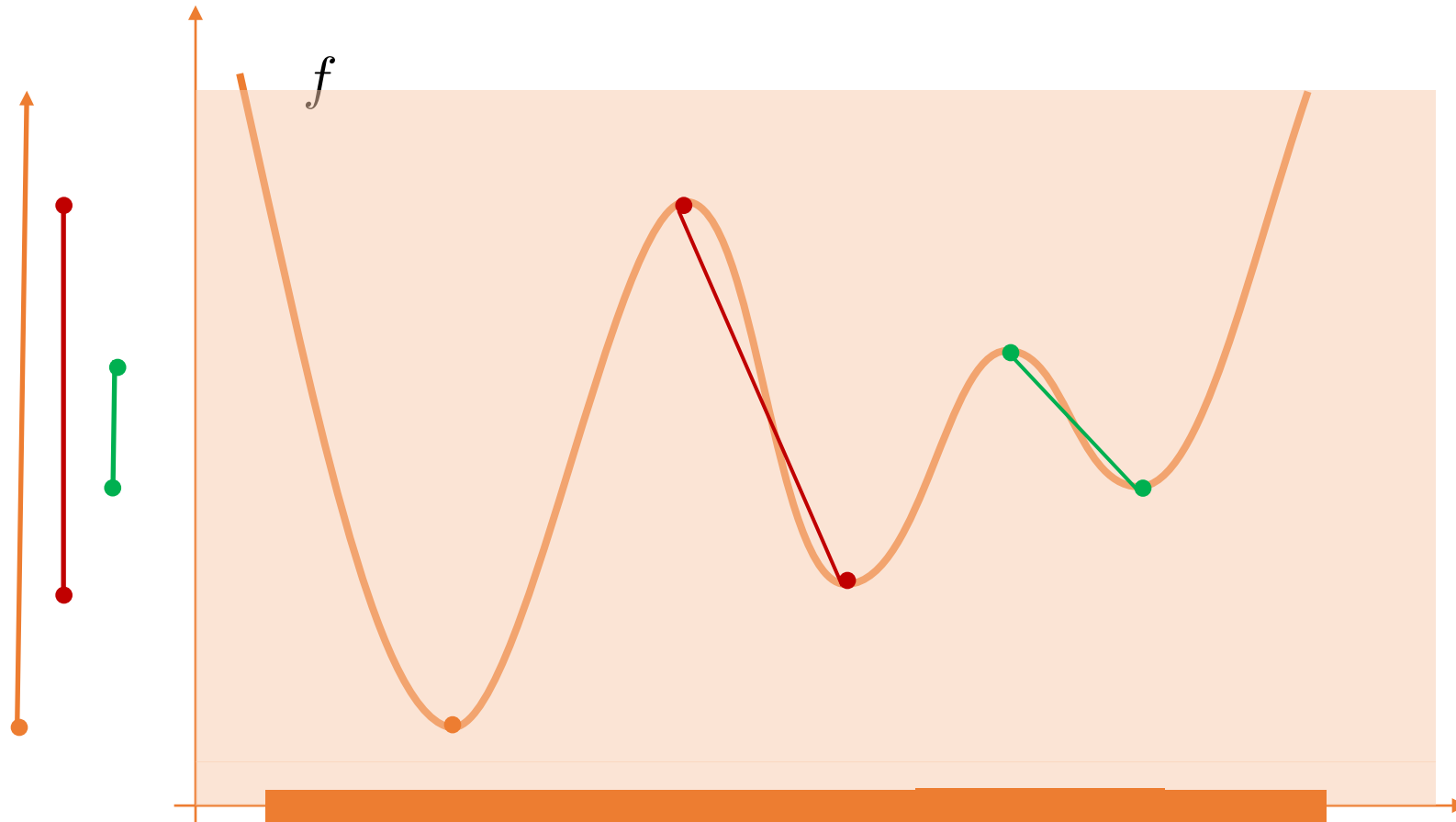


Equivalently, a Diagram

- ◆ Pairs (b_i, d_i) are represented as points on upper-half plane
- ◆ Often (but not always), short bars correspond to “topological noise”



Elder Rule



Demo – Ripser and Javaplex

Mathematical theory

A Deeper Look at the Algorithm

- ◆ Why the restriction on directions of column operations?
- ◆ Why is persistent homology different from computing standard homology a bunch of times in parallel?

$$0 \rightarrow H_p(\Sigma^{(1)}) \rightarrow H_p(\Sigma^{(2)}) \rightarrow \dots \rightarrow H_p(\Sigma^{(n-1)}) \rightarrow H_p(\Sigma^{(n)})$$

A Deeper Look at the Algorithm

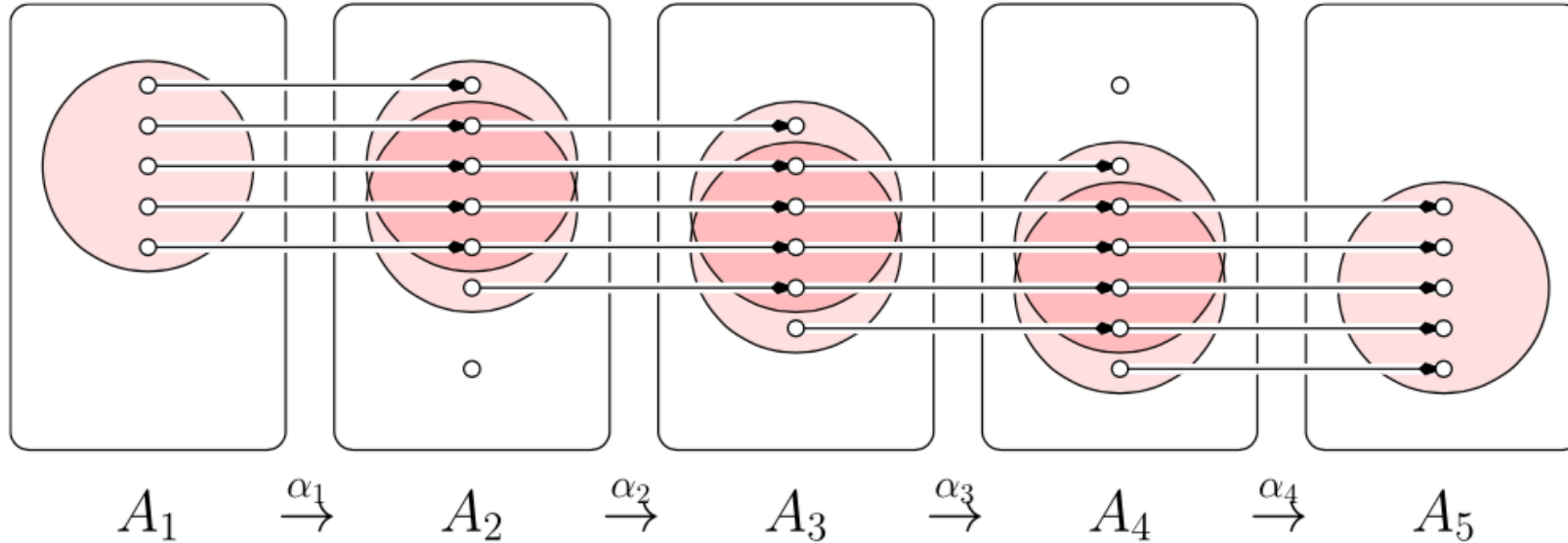
- ◆ Why the restriction on directions of column operations?
- ◆ Why is persistent homology different from computing standard homology a bunch of times in parallel?

$$0 \rightarrow H_p(\Sigma^{(1)}) \rightarrow H_p(\Sigma^{(2)}) \rightarrow \dots \rightarrow H_p(\Sigma^{(n-1)}) \rightarrow H_p(\Sigma^{(n)})$$

Obtaining bases for the individual vector spaces is **not sufficient**. To encode linear maps, we need **compatible** bases.

A Deeper Look at the Algorithm

Figure from Edelsbrunner, Jablonski, Mrozek, *The persistent homology of a self-map*



- ◆ Left-to-right ordering of chain complex basis vectors + left-to-right column operations gets us to compatible bases for homology
- ◆ “Existence and uniqueness” comes from commutative algebra

A Deeper Look at the Algorithm

Carlsson, Zomorodian, *Computing Persistent Homology*

$$\begin{array}{ccccccc} \partial_3 \downarrow & & \partial_3 \downarrow & & \partial_3 \downarrow & & \\ \mathbf{C}_2^0 & \xrightarrow{f^0} & \mathbf{C}_2^1 & \xrightarrow{f^1} & \mathbf{C}_2^2 & \xrightarrow{f^2} & \dots \\ \partial_2 \downarrow & & \partial_2 \downarrow & & \partial_2 \downarrow & & \\ \mathbf{C}_1^0 & \xrightarrow{f^0} & \mathbf{C}_1^1 & \xrightarrow{f^1} & \mathbf{C}_1^2 & \xrightarrow{f^2} & \dots \\ \partial_1 \downarrow & & \partial_1 \downarrow & & \partial_1 \downarrow & & \\ \mathbf{C}_0^0 & \xrightarrow{f^0} & \mathbf{C}_0^1 & \xrightarrow{f^1} & \mathbf{C}_0^2 & \xrightarrow{f^2} & \dots \end{array}$$

tl;dr: (1) Massage all the vector spaces into a “familiar structure” (*equivalence of categories*), (2) apply known classification results for the familiar structure

- ◆ Rephrased PH computation: stack all these vector spaces together into a single structure, find a compatible basis for the full structure in one shot

- ◆ (Carlsson, Zomorodian ‘05) The sequence

$$V_1 \xrightarrow{f_1} V_2 \xrightarrow{f_2} \dots \rightarrow V_{n-1} \xrightarrow{f_{n-1}} V_n$$

can be viewed as a module (=generalized vector space) over a polynomial ring $\mathbb{F}[t] = \{\sum_{i \geq 0} a_i t^i \mid a_i \in \mathbb{F}\}$

- ◆ powers of t encode the filtration order
- ◆ formal equivalence to modules enables import of standard algebra results

PH as Applied Representation Theory

Carlsson, de Silva, Zigzag Persistence

◆ $\mathbb{V} := V_1 \xrightarrow{f_1} V_2 \xrightarrow{f_2} \dots \rightarrow V_{n-1} \xrightarrow{f_{n-1}} V_n = \cdot \rightarrow \cdot \rightarrow \dots \rightarrow \cdot \rightarrow \cdot$ + choice of vector spaces and linear maps

◆ This is a **quiver representation**. Given equal-length quiver reps \mathbb{W}, \mathbb{X} , say $\mathbb{V} = \mathbb{W} \oplus \mathbb{X}$ if $V_i = W_i \oplus X_i$ and $f_i(W_i) \subseteq W_{i+1}, f_i(X_i) \subseteq X_{i+1}$ for each i

Theorem (Gabriel '72): Any \mathbb{V} as above can be written $\mathbb{V} = \bigoplus_i \mathbb{I}(b_i, d_i)$, where

$$\mathbb{I}(b_i, d_i) = 0 \rightarrow 0 \rightarrow \dots 0 \rightarrow \mathbb{F} \rightarrow \mathbb{F} \rightarrow \mathbb{F} \rightarrow \dots \rightarrow \mathbb{F} \rightarrow 0 \rightarrow 0 \rightarrow 0$$

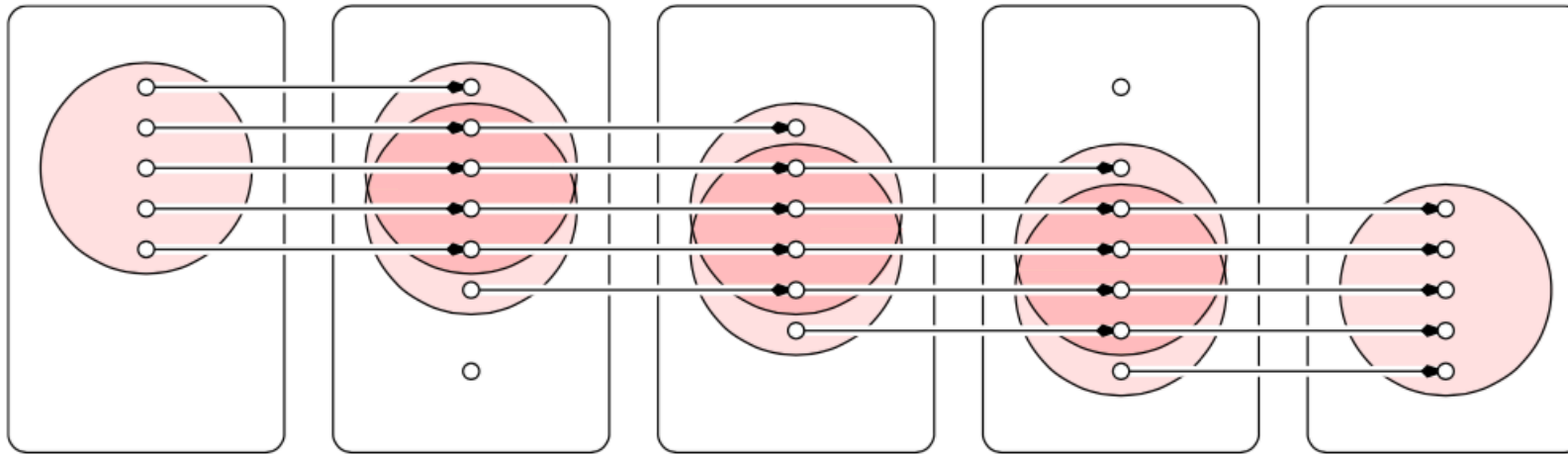
↑
↑
↑
↑

Index 1
Index b_i
Index d_i
Index n

Theorem (Krull-Remak-Schmidt): Such a decomposition is unique up to relabeling.

PH as Applied Representation Theory

Carlsson, de Silva, *Zigzag Persistence*



Theorem (Gabriel '72): Any \mathbb{V} as above can be written $\mathbb{V} = \bigoplus_i \mathbb{I}(b_i, d_i)$, where

$$\mathbb{I}(b_i, d_i) = 0 \rightarrow 0 \rightarrow \cdots 0 \rightarrow \mathbb{F} \rightarrow \mathbb{F} \rightarrow \mathbb{F} \rightarrow \cdots \rightarrow \mathbb{F} \rightarrow 0 \rightarrow 0 \rightarrow 0$$

\uparrow
Index 1
 \uparrow
Index b_i
 \uparrow
Index d_i
 \uparrow
Index n

Theorem (Krull-Remak-Schmidt): Such a decomposition is unique up to relabeling.

◆ Gabriel's theorem does not need all the arrows to point in the same direction!

- Standard persistence:

$$V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V_n$$

- Zigzag persistence:

$$V_1 \xleftrightarrow{p_1} V_2 \xleftrightarrow{p_2} \cdots \xleftrightarrow{p_{n-1}} V_n.$$

- Each p_i can be forward or backward

◆ Application: inclusion not needed, can compare/correlate features across incomparable spaces

- E.g. Union sequence: $X_1 \rightarrow X_1 \cup X_2 \leftarrow X_2 \rightarrow \cdots \leftarrow X_n$

Demo - Dionysus

Theory part II: Stability

Stability of Persistent Homology

- ◆ Persistent homology outputs are robust to small perturbations of data

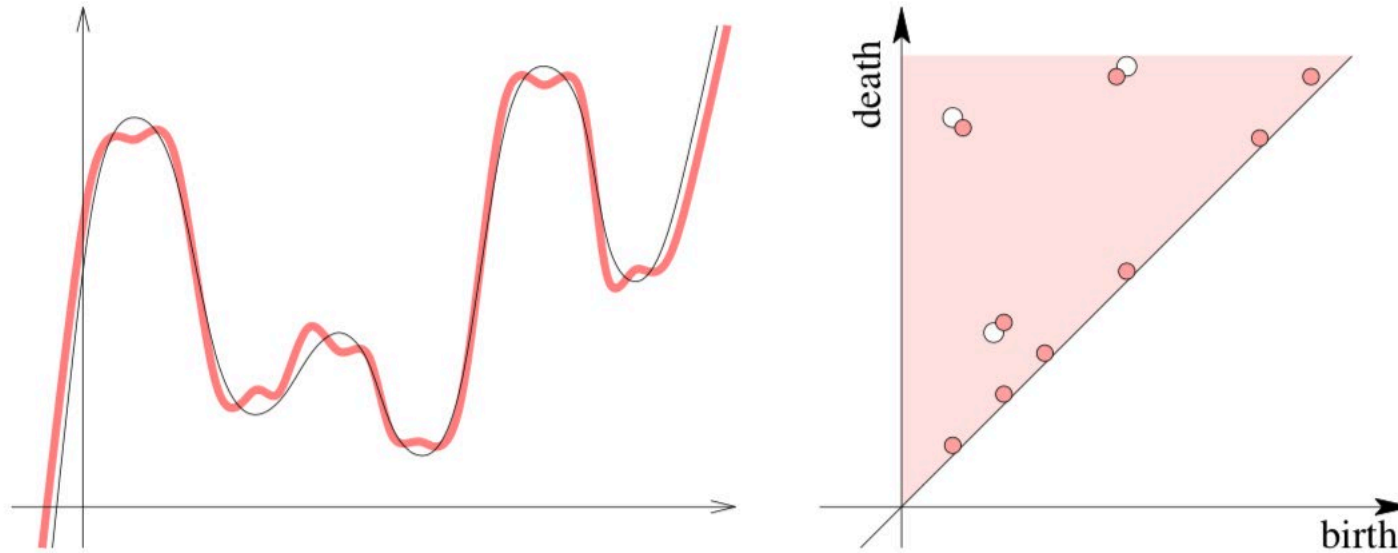


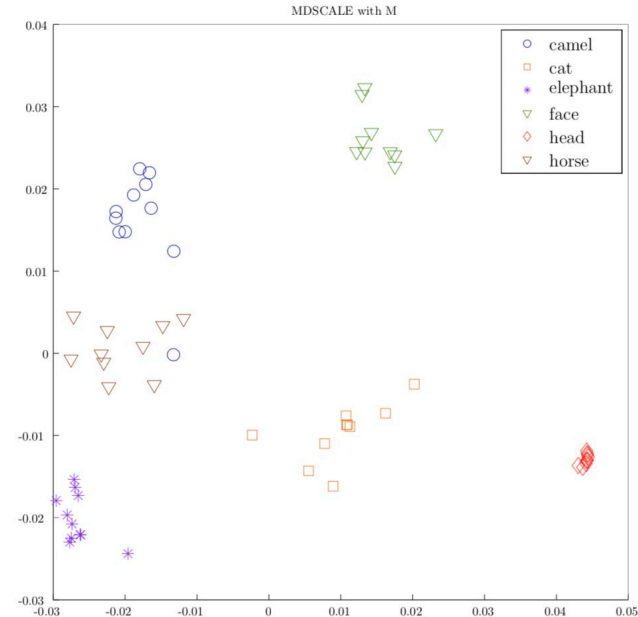
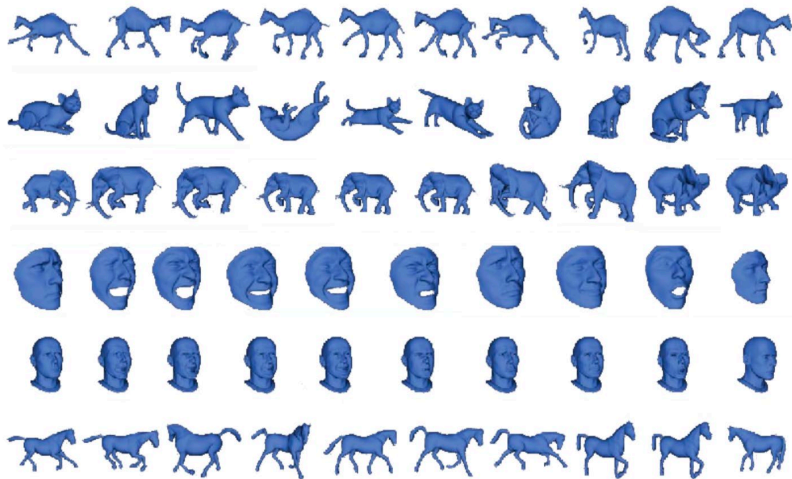
Figure from Edelsbrunner, Harer,
Computational Topology

- ◆ Theorem (Cohen-Steiner, Edelsbrunner, Harer): \mathcal{M} a triangulable space, $f, g: \mathcal{M} \rightarrow \mathbb{R}$ “nice”. Then,
$$d_B(D_k(f), D_k(g)) \leq \|f - g\|_\infty.$$

Here D_k is the persistence diagram in dimension k , $\|\cdot\|_\infty$ is the max-norm, and d_B is the bottleneck distance (TBD)

Stability of Persistent Homology

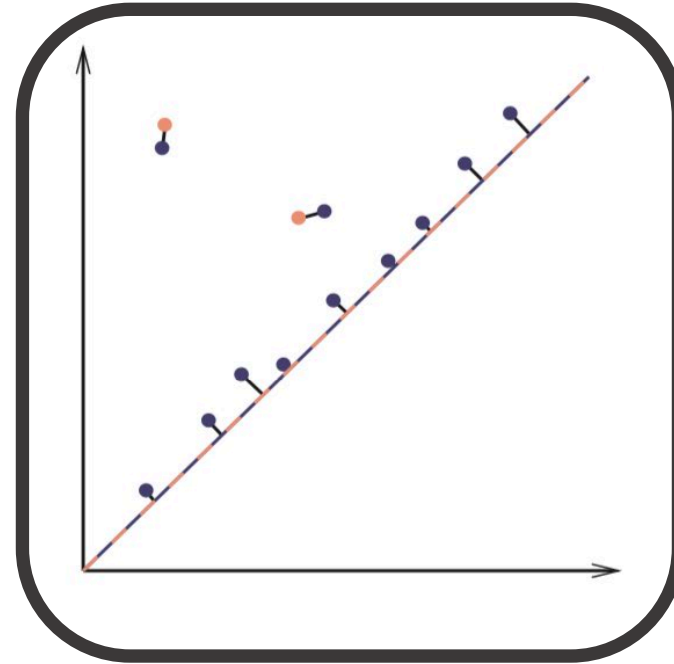
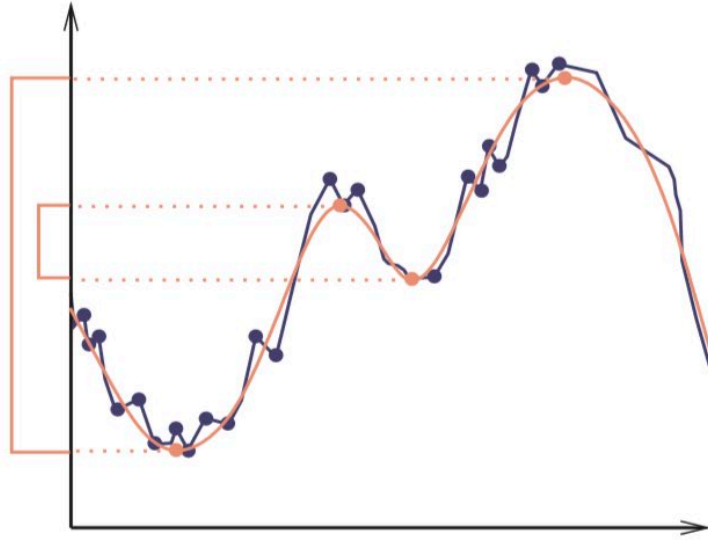
- ◆ We also have stability when building filtered complexes from metric data



- ◆ Theorem (Chazal, Cohen-Steiner, Guibas, Mémoli, Oudot): $(X, d_X), (Y, d_Y)$ finite metric spaces. Then,
$$d_B(D_k(VR(X)), D_k(VR(Y))) \leq 2d_{GH}(X, Y).$$

Here d_{GH} is the Gromov-Hausdorff distance between metric spaces (TBD)

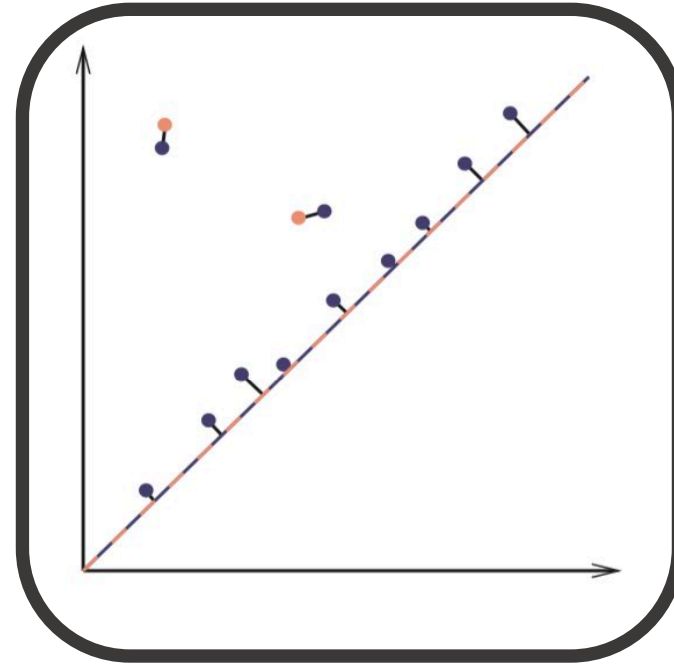
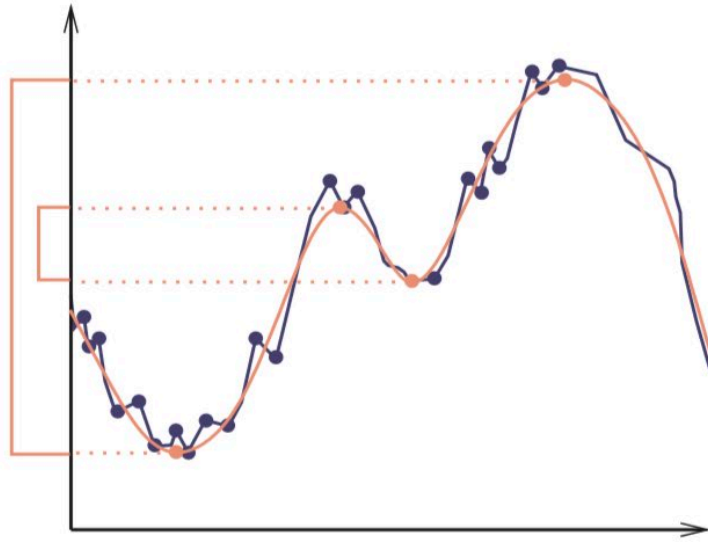
The Bottleneck Distance



Method for comparing persistence diagrams:

- ◆ Choose a bijection, matching points to the diagonal if needed
- ◆ Compute the l^∞ distance between each pair of matched points (max-norm)
- ◆ Take worst case matching cost
- ◆ Optimize over all possible bijections

The Bottleneck Distance



A, B two persistence diagrams.

$$d_B(A, B) := \inf \left\{ \sup_{x \in A \cup \Delta} \|x - \phi(x)\|_\infty : \phi : A \cup \Delta \rightarrow B \cup \Delta \text{ a bijection} \right\}$$

Gromov-Hausdorff Stability

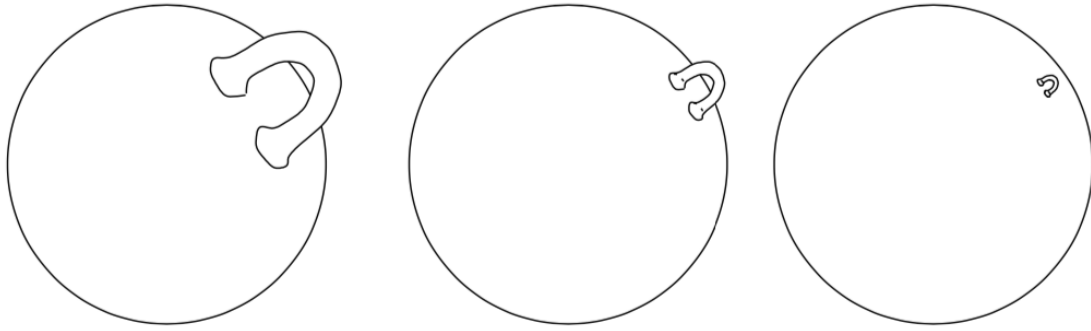


Figure from Burago, Burago, Ivanov, *A course in metric geometry*

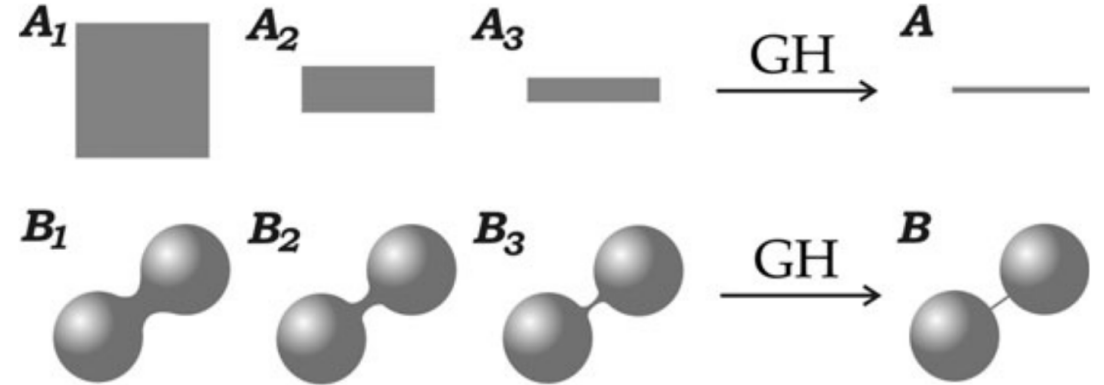


Figure from Sormani, *How Riemannian manifolds converge*

The space of metric spaces is a metric space under the Gromov-Hausdorff distance

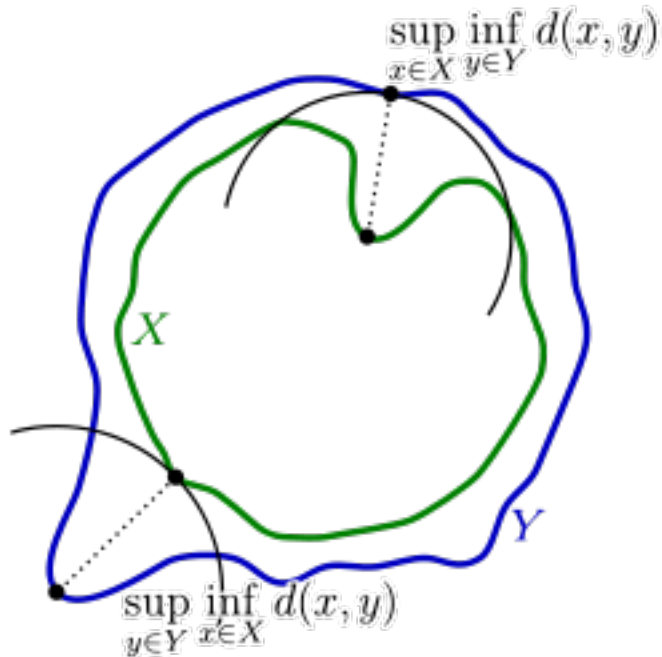
- ◆ Used to study convergence of spaces
- ◆ Popularized by Gromov in the context of geometric group theory

Gromov-Hausdorff stability

X, Y subsets of a finite metric space (Z, d)

Hausdorff distance:

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

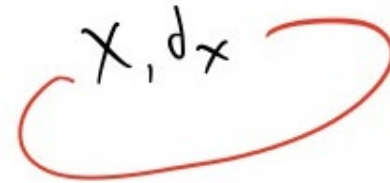


$(X, d_X), (Y, d_Y)$ finite metric spaces

Gromov-Hausdorff distance:

$$d_{GH}(X, Y) = \inf_{Z, \phi_X, \phi_Y} d_H^Z(\phi_X(X), \phi_Y(Y)), \text{ where}$$

ϕ_X, ϕ_Y are isometric embeddings

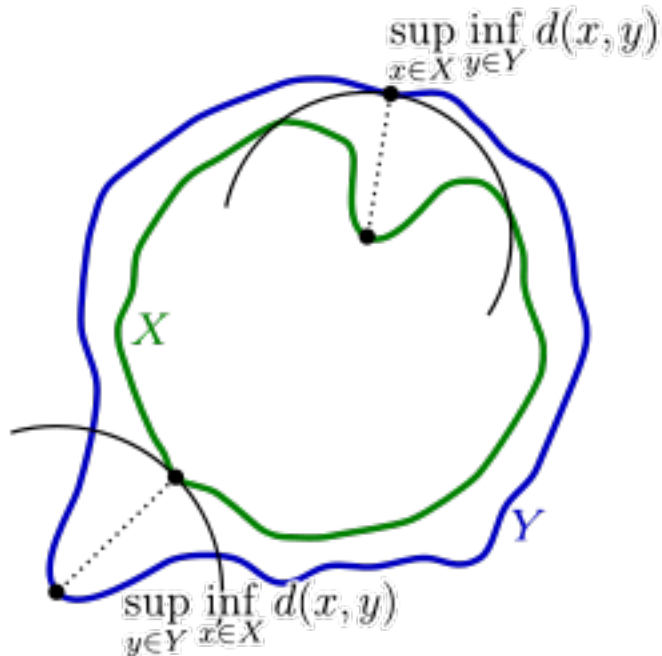


Gromov-Hausdorff Stability

X, Y subsets of a finite metric space (Z, d)

Hausdorff distance:

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

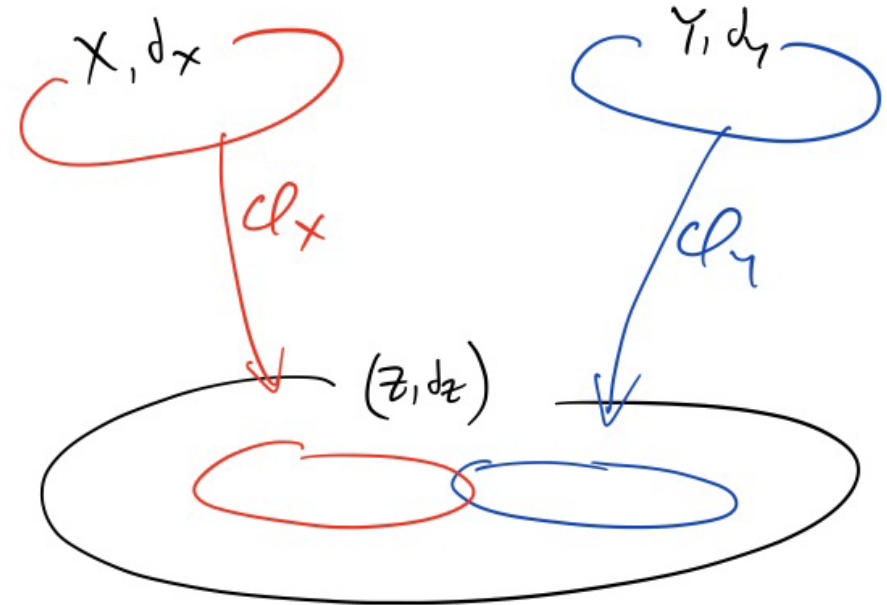


$(X, d_X), (Y, d_Y)$ finite metric spaces

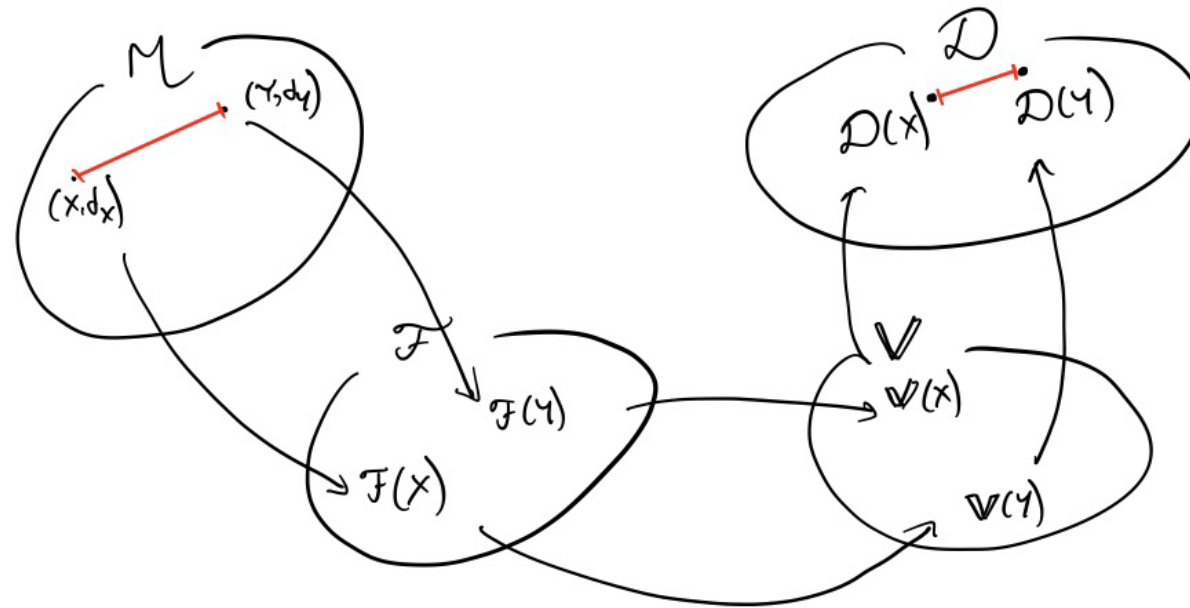
Gromov-Hausdorff distance:

$$d_{GH}(X, Y) = \inf_{Z, \phi_X, \phi_Y} d_H^Z(\phi_X(X), \phi_Y(Y)), \text{ where}$$

ϕ_X, ϕ_Y are isometric embeddings



Gromov-Hausdorff Stability



\mathcal{M} = finite metric spaces
 \mathcal{F} = filtered simplicial complexes
 \mathbb{V} = persistent vector spaces
 \mathbb{D} = persistence diagrams

- Theorem (Chazal, Cohen-Steiner, Guibas, Mémoli, Oudot): $(X, d_X), (Y, d_Y)$ finite metric spaces. Then,

$$d_B(D_k(VR(X)), D_k(VR(Y))) \leq 2d_{GH}(X, Y).$$

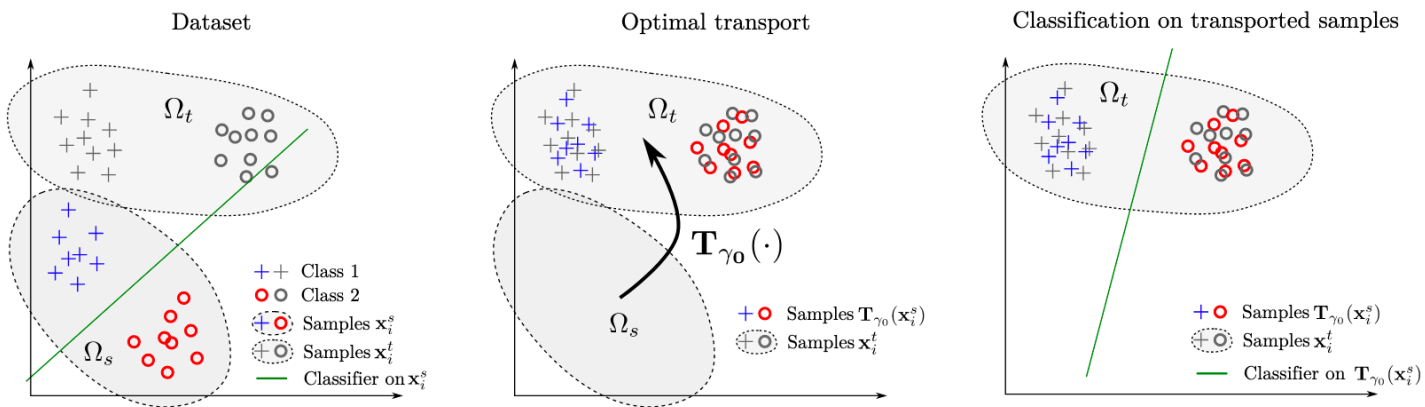
$O(n^3)$ with Hungarian algorithm,
 $O(n^{3/2})$ state-of-the-art (Kerber, Morozov,
 Nigmatov 2017)

NP-hard

Connections

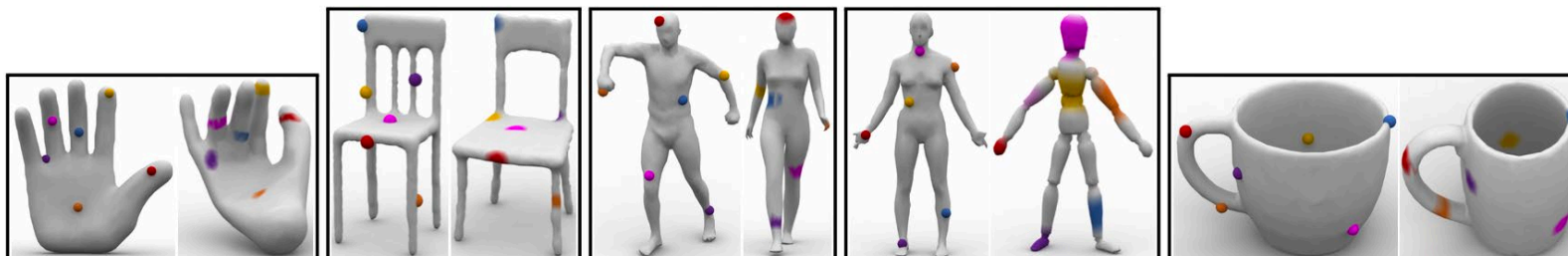
◆ Wasserstein distance: a probabilistic analog of Hausdorff distance

- Popularized in the study of *optimal transport*
- Used in Wasserstein-GANs, domain adaptation, transfer learning



◆ Gromov-Wasserstein distance: probabilistic analog of Gromov-Hausdorff

- Arises as convex relaxation (loss remains nonconvex, but optimization is over a convex set)
- Useful for obtaining correspondences across shapes



Solomon, Peyré, Kim, Sra, *Entropic metric alignment for correspondence problems*. SIGGRAPH'16

Software Packages

- ◆ **Javaplex** - <http://appliedtopology.github.io/javaplex/>
- ◆ **Ripser** - <https://github.com/Ripser/ripser>
 - **Ripser.py** - <https://github.com/scikit-tda/ripser.py>
 - Ripser live - <https://live.ripser.org/>
- ◆ **Ripser++** - <https://github.com/simonzhang00/ripser-plusplus>
- ◆ **Dionysus 2** - <https://mrzv.org/software/dionysus2/>
- ◆ **Gudhi** - <https://gudhi.inria.fr/python/latest/>
- ◆ **DIPHA** - <https://github.com/DIPHA/dipha>
- ◆ **PHAT** - <https://bitbucket.org/phat-code/phat>
- ◆ **Eirene** - <https://github.com/Eetion/Eirene.jl>
- ◆ **BATS** - <https://bnels.github.io/BATS.py/#/>

That's All

