

Computation of Persistent Homology

Part 2: Efficient Computation of Vietoris–Rips Persistence

Ulrich Bauer

TUM

August 14, 2018

Tutorial on Multiparameter Persistence, Computation, and Applications
Institute for Mathematics and its Applications

Vietoris–Rips persistent homology

Vietoris–Rips filtrations

Consider a finite metric space (X, d) .

The *Vietoris–Rips complex* is the simplicial complex

$$\text{Rips}_t(X) = \{S \subseteq X \mid \text{diam } S \leq t\}$$

- 1-skeleton: all edges with pairwise distance $\leq t$
- all possible higher simplices (flag complex)

Vietoris–Rips filtrations

Consider a finite metric space (X, d) .

The *Vietoris–Rips complex* is the simplicial complex

$$\text{Rips}_t(X) = \{S \subseteq X \mid \text{diam } S \leq t\}$$

- 1-skeleton: all edges with pairwise distance $\leq t$
- all possible higher simplices (flag complex)

Goal:

- compute persistence barcodes for $H_d(\text{Rips}_t(X))$
(in dimensions $0 \leq d \leq k$)

Demo: Ripser

Example data set:

- 192 points on \mathbb{S}^2
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Demo: Ripser

Example data set:

- 192 points on \mathbb{S}^2
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Comparison with other software:

- javaplex: 3200 seconds, 12 GB
- Dionysus: 533 seconds, 3.4 GB
- GUDHI: 75 seconds, 2.9 GB
- DIPHA: 50 seconds, 6 GB
- Eirene: 12 seconds, 1.5 GB

Demo: Ripser

Example data set:

- 192 points on \mathbb{S}^2
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Comparison with other software:

- javaplex: 3200 seconds, 12 GB
- Dionysus: 533 seconds, 3.4 GB
- GUDHI: 75 seconds, 2.9 GB
- DIPHA: 50 seconds, 6 GB
- Eirene: 12 seconds, 1.5 GB

Ripser: 1.2 seconds, 152 MB

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies
- support for
 - coefficients in a prime field \mathbb{F}_p
 - sparse distance matrices for distance threshold
- open source (<http://ripser.org>)
 - released in July 2016
- online version (<http://live.ripser.org>)
 - launched in August 2016
- most efficient software for Vietoris–Rips persistence
 - computes H^2 barcode for 50 000 random points on a torus in 136 seconds / 9 GB (using distance threshold)
- 2016 ATMCS Best New Software Award (jointly with RIVET)

Design goals

Goals for previous persistence software projects:

- PHAT [B, Kerber, Reininghaus, Wagner 2013]:
fast persistence computation (matrix reduction only),
comparing different algorithms and data structures
- DIPHA [B, Kerber, Reininghaus 2014]:
distributed persistence computation, based on spectral
sequence algorithm

Design goals

Goals for previous persistence software projects:

- PHAT [B, Kerber, Reininghaus, Wagner 2013]:
fast persistence computation (matrix reduction only),
comparing different algorithms and data structures
- DIPHA [B, Kerber, Reininghaus 2014]:
distributed persistence computation, based on spectral
sequence algorithm

Goals for Ripser:

- Use as little memory as possible
- Be reasonable about computation time

The four special ingredients

Main improvements to the standard reduction algorithm:

- Clearing inessential columns [Chen, Kerber 2011]
- Computing cohomology [de Silva et al. 2011]
- Implicit matrix reduction
- Apparent and emergent pairs

The four special ingredients

Main improvements to the standard reduction algorithm:

- Clearing inessential columns [Chen, Kerber 2011]
- Computing cohomology [de Silva et al. 2011]
- Implicit matrix reduction
- Apparent and emergent pairs

Lessons from PHAT:

- Clearing and cohomology yield considerable speedup,
- but only when *both* are used in conjunction!

Filtrations and refinements

Simplexwise filtrations

Call a filtration $(K_i)_{i \in I}$ of simplicial complexes (I totally ordered)

- *essential* if $i \neq j$ implies $K_i \neq K_j$,
- *simplexwise* if for all $i \in I$ with $K_i \neq \emptyset$ there is some index $j < i \in I$ and some simplex $\sigma \in K$ such that $K_i \setminus K_j = \{\sigma\}$.

Simplexwise filtrations

Call a filtration $(K_i)_{i \in I}$ of simplicial complexes (I totally ordered)

- *essential* if $i \neq j$ implies $K_i \neq K_j$,
- *simplexwise* if for all $i \in I$ with $K_i \neq \emptyset$ there is some index $j < i \in I$ and some simplex $\sigma \in K$ such that $K_i \setminus K_j = \{\sigma\}$.

Note:

- These properties are natural assumptions for computation.
- The *Vietoris–Rips filtration* (indexed by \mathbb{R}) is not simplexwise (and not essential).
- To compute Vietoris–Rips persistence, we will reindex and refine.

Reindexing and refinement

A filtration $K : I \rightarrow \mathbf{Simp}$ is a *reindexing* of another filtration $F : R \rightarrow \mathbf{Simp}$ if $F = K \circ r$ for some monotonic map $r : R \rightarrow I$.

- If r is injective, we call K a *refinement* of F ;
- if r is surjective, we call K a *reduction* of F .

Reindexing and refinement

A filtration $K : I \rightarrow \mathbf{Simp}$ is a *reindexing* of another filtration $F : R \rightarrow \mathbf{Simp}$ if $F = K \circ r$ for some monotonic map $r : R \rightarrow I$.

- If r is injective, we call K a *refinement* of F ;
- if r is surjective, we call K a *reduction* of F .

The Vietoris–Rips filtration can be reindexed:

- reduced to an essential filtration (over the set of pairwise distances), and further
- refined to a simplexwise filtration

Reindexing and refinement

A filtration $K : I \rightarrow \mathbf{Simp}$ is a *reindexing* of another filtration $F : R \rightarrow \mathbf{Simp}$ if $F = K \circ r$ for some monotonic map $r : R \rightarrow I$.

- If r is injective, we call K a *refinement* of F ;
- if r is surjective, we call K a *reduction* of F .

The Vietoris–Rips filtration can be reindexed:

- reduced to an essential filtration (over the set of pairwise distances), and further
- refined to a simplexwise filtration

Ripser use the *lexicographic refinement*: simplices ordered by

- diameter, then
- dimension, then
- (decreasing) lexicographic order of vertices $\{v_{i_k}, \dots, v_{i_0}\}$ (induced by fixed total order $v_0 < \dots < v_{n-1}$ on vertices)

Enumerating (co)faces in lexicographic order

There is an order-preserving bijection

$$\begin{array}{ccccccc} [v_2, v_1, v_0] & [v_3, v_1, v_0] & [v_3, v_1, v_0] & [v_3, v_2, v_0] & \cdots & [v_{n-3}, v_{n-2}, v_{n-1}] \\ \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow \\ 0 & 1 & 2 & 3 & \cdots & \binom{n}{k+1} - 1 \end{array}$$

from k -simplices (as ordered tuples, ordered lexicographically) to natural numbers (called *combinatorial number system*).

- One direction is given by

$$[v_{i_k}, \dots, v_{i_0}] \mapsto \sum_{j=0}^k \binom{i_j}{j+1}$$

- The other direction can also be computed quickly
- Using this bijection, faces and cofaces can be efficiently enumerated in (decreasing) lexicographic order

Persistence barcodes through reindexing

Consider a filtration $F : \mathbb{R} \rightarrow \mathbf{Simp}$ with reindexing $F = K \circ r$, $K : I \rightarrow \mathbf{Simp}$, $r : \mathbb{R} \rightarrow [n]$. The persistence barcode of K_\bullet determines the persistence barcode of F_\bullet :

$$B(H_*(F_\bullet)) = \{r^{-1}(I) \neq \emptyset \mid I \in B(H_*(K_\bullet))\}.$$

Matrix reduction

Computing homology

Computing homology $H_* = Z_*/B_*$:

- compute basis for boundaries $B_* = \text{im } \partial_*$
- extend to basis for cycles $Z_* = \ker \partial_*$
- new (non-boundary) basis cycles generate quotient Z_*/B_*

Computing homology

Computing homology $H_* = Z_*/B_*$:

- compute basis for boundaries $B_* = \text{im } \partial_*$
- extend to basis for cycles $Z_* = \ker \partial_*$
- new (non-boundary) basis cycles generate quotient Z_*/B_*

Computing *persistent* homology $H_* = Z_*/B_*$ (for a simplexwise filtration $K_i \subseteq K$):

- compute *filtered* basis for boundaries $B_* = \text{im } \partial_*$
- extend to basis for cycles $Z_* = \ker \partial_*$
- all basis cycles generate *persistent* homology

Persistence by matrix reduction

Given:

- D : matrix of boundary ∂_d for a simplexwise filtration $(K_i)_i$ (for canonical basis in filtration order, indexed by $I \times J$)

Wanted:

- persistence barcode of homology $H_d(K_i; \mathbb{F})$ for some (prime) field \mathbb{F} , in dimensions $d = 0, \dots, k$

Notation:

- m_j : j th column of M , m_{ij} : entry in i th row and j th column
- Pivot $m_j = \min\{i \in I : m_{kj} = 0 \text{ for all } k > i\}$.

Persistence by matrix reduction

Given:

- D : matrix of boundary ∂_d for a simplexwise filtration $(K_i)_i$ (for canonical basis in filtration order, indexed by $I \times J$)

Wanted:

- persistence barcode of homology $H_d(K_i; \mathbb{F})$ for some (prime) field \mathbb{F} , in dimensions $d = 0, \dots, k$

Notation:

- m_j : j th column of M , m_{ij} : entry in i th row and j th column
- Pivot $m_j = \min\{i \in I : m_{kj} = 0 \text{ for all } k > i\}$.

Computation: barcode is obtained by *matrix reduction* of D

- $R = D \cdot V$ reduced (non-zero columns have distinct pivots)
- V is regular upper triangular

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$I_b = \{i : r_i = 0\}$ *birth* indices ,

$I_d = \{j : r_j \neq 0\}$ *death* indices,

$I_e = I_b \setminus \text{Pivots } R$ *essential* indices

(note: $i = \text{Pivot } r_j$ implies $r_i = 0$, thus $\text{Pivots } R \subseteq I_b$). Then

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$\begin{array}{ll} I_b = \{i : r_i = 0\} & \text{birth indices,} \\ I_d = \{j : r_j \neq 0\} & \text{death indices,} \\ I_e = I_b \setminus \text{Pivots } R & \text{essential indices} \end{array}$$

(note: $i = \text{Pivot } r_j$ implies $r_i = 0$, thus $\text{Pivots } R \subseteq I_b$). Then

$$\Sigma_B = \{r_j \mid j \in I_d\} \quad \text{is a basis of } B_*,$$

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$\begin{array}{ll} I_b = \{i : r_i = 0\} & \text{birth indices,} \\ I_d = \{j : r_j \neq 0\} & \text{death indices,} \\ I_e = I_b \setminus \text{Pivots } R & \text{essential indices} \end{array}$$

(note: $i = \text{Pivot } r_j$ implies $r_i = 0$, thus $\text{Pivots } R \subseteq I_b$). Then

$$\begin{array}{ll} \Sigma_B = \{r_j \mid j \in I_d\} & \text{is a basis of } B_*, \\ \Sigma_Z = \Sigma_B \cup \{v_i \mid i \in I_e\} & \text{is a compatible basis of } Z_*. \end{array}$$

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$I_b = \{i : r_i = 0\}$ *birth* indices ,

$I_d = \{j : r_j \neq 0\}$ *death* indices,

$I_e = I_b \setminus \text{Pivots } R$ *essential* indices

(note: $i = \text{Pivot } r_j$ implies $r_i = 0$, thus $\text{Pivots } R \subseteq I_b$). Then

$\Sigma_B = \{r_j \mid j \in I_d\}$ is a basis of B_* ,

$\Sigma_Z = \Sigma_B \cup \{v_i \mid i \in I_e\}$ is a *compatible* basis of Z_* .

- Persistent homology is generated by the basis cycles Σ_Z .

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$I_b = \{i : r_i = 0\}$ *birth* indices ,

$I_d = \{j : r_j \neq 0\}$ *death* indices,

$I_e = I_b \setminus \text{Pivots } R$ *essential* indices

(note: $i = \text{Pivot } r_j$ implies $r_i = 0$, thus $\text{Pivots } R \subseteq I_b$). Then

$\Sigma_B = \{r_j \mid j \in I_d\}$ is a basis of B_* ,

$\Sigma_Z = \Sigma_B \cup \{v_i \mid i \in I_e\}$ is a *compatible* basis of Z_* .

- Persistent homology is generated by the basis cycles Σ_Z .
- Persistence intervals: $\{[i, j) \mid i = \text{Pivot } r_j\} \cup \{[i, \infty) \mid i \in I_e\}$

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$I_b = \{i : r_i = 0\}$ *birth* indices ,

$I_d = \{j : r_j \neq 0\}$ *death* indices,

$I_e = I_b \setminus \text{Pivots } R$ *essential* indices

(note: $i = \text{Pivot } r_j$ implies $r_i = 0$, thus $\text{Pivots } R \subseteq I_b$). Then

$\Sigma_B = \{r_j \mid j \in I_d\}$ is a basis of B_* ,

$\Sigma_Z = \Sigma_B \cup \{v_i \mid i \in I_e\}$ is a *compatible* basis of Z_* .

- Persistent homology is generated by the basis cycles Σ_Z .
- Persistence intervals: $\{[i, j) \mid i = \text{Pivot } r_j\} \cup \{[i, \infty) \mid i \in I_e\}$
- Matrix V not used for barcode

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$I_b = \{i : r_i = 0\}$ *birth* indices ,

$I_d = \{j : r_j \neq 0\}$ *death* indices,

$I_e = I_b \setminus \text{Pivots } R$ *essential* indices

(note: $i = \text{Pivot } r_j$ implies $r_i = 0$, thus $\text{Pivots } R \subseteq I_b$). Then

$\Sigma_B = \{r_j \mid j \in I_d\}$ is a basis of B_* ,

$\Sigma_Z = \Sigma_B \cup \{v_i \mid i \in I_e\}$ is a *compatible* basis of Z_* .

- Persistent homology is generated by the basis cycles Σ_Z .
- Persistence intervals: $\{[i, j) \mid i = \text{Pivot } r_j\} \cup \{[i, \infty) \mid i \in I_e\}$
- Matrix V not used for barcode
- Columns with indices $\text{Pivots } R$ not used at all

Matrix reduction algorithm

Require: $D: I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V : regular upper triangular,

P : persistence pairs

$R := D$

$V := \text{Id}(J \times J)$

for $j \in J$ in increasing order **do**

while $\exists k < j$ with $i := \text{Pivot } r_k = \text{Pivot } r_j$ **do**

$$\lambda = \frac{r_{ij}}{r_{ik}}$$

$$r_j := r_j - \lambda \cdot r_k$$

▷ eliminate pivot entry r_{ij}

$$v_j := v_j - \lambda \cdot v_k$$

if $i := \text{Pivot } r_j \neq 0$ **then**

 append (i, j) to P

return R, V

Clearing

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential birth indices
- Use the fact that $i = \text{Pivot } r_j$ implies $r_i = 0$
- Reduce boundary matrices of $\partial_d : C_d \rightarrow C_{d-1}$ in decreasing dimension $d = k + 1, \dots, 1$
- Whenever $i = \text{Pivot } r_j$ (in matrix for ∂_d)
 - Set r_i to 0 (in matrix for ∂_{d-1})

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential birth indices
- Use the fact that $i = \text{Pivot } r_j$ implies $r_i = 0$
- Reduce boundary matrices of $\partial_d : C_d \rightarrow C_{d-1}$ in decreasing dimension $d = k + 1, \dots, 1$
- Whenever $i = \text{Pivot } r_j$ (in matrix for ∂_d)
 - Set r_i to 0 (in matrix for ∂_{d-1})
 - Set v_i to r_j (if V is needed)

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential birth indices
- Use the fact that $i = \text{Pivot } r_j$ implies $r_i = 0$
- Reduce boundary matrices of $\partial_d : C_d \rightarrow C_{d-1}$ in decreasing dimension $d = k + 1, \dots, 1$
- Whenever $i = \text{Pivot } r_j$ (in matrix for ∂_d)
 - Set r_i to 0 (in matrix for ∂_{d-1})
 - Set v_i to r_j (if V is needed)
- Still yields $R = D \cdot V$ reduced, V regular upper triangular

Note:

- reducing *birth* columns typically harder than death columns: $O(j^2)$ (birth) vs. $O((j - i)^2)$ (death)
- with clearing: need only reduce *essential* columns

Counting homology column reductions

Consider $k + 1$ -skeleton of $n - 1$ -simplex (Rips filtration)

Number of columns in boundary matrix:

$$\underbrace{\sum_{d=1}^{k+1} \binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=1}^{k+1} \binom{n-1}{d}}_{\text{death}} + \underbrace{\sum_{d=1}^{k+1} \binom{n-1}{d+1}}_{\text{birth}}$$

Counting homology column reductions

Consider $k + 1$ -skeleton of $n - 1$ -simplex (Rips filtration)

Number of columns in boundary matrix:

$$\begin{aligned} \underbrace{\sum_{d=1}^{k+1} \binom{n}{d+1}}_{\text{total}} &= \underbrace{\sum_{d=1}^{k+1} \binom{n-1}{d}}_{\text{death}} + \underbrace{\sum_{d=1}^{k+1} \binom{n-1}{d+1}}_{\text{birth}} \\ &= \underbrace{\sum_{d=1}^{k+1} \binom{n-1}{d}}_{\text{death}} + \underbrace{\binom{n-1}{k+2}}_{\text{essential}} + \underbrace{\sum_{d=1}^k \binom{n-1}{d+1}}_{\text{cleared}} \end{aligned}$$

Counting homology column reductions

Consider $k + 1$ -skeleton of $n - 1$ -simplex (Rips filtration)

Number of columns in boundary matrix:

$$\begin{aligned} \sum_{d=1}^{k+1} \underbrace{\binom{n}{d+1}}_{\text{total}} &= \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\text{death}} + \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d+1}}_{\text{birth}} \\ &= \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\text{death}} + \underbrace{\binom{n-1}{k+2}}_{\text{essential}} + \underbrace{\sum_{d=1}^k \binom{n-1}{d+1}}_{\text{cleared}} \end{aligned}$$

$$k = 2, n = 192: \quad 56\,050\,096 = 1161\,471 + 53\,727\,345 + 1161\,280$$

- Clearing didn't help much!

Cohomology

Persistent (co)homology

How is persistent cohomology related to persistent homology?

- Cohomology (over \mathbb{F}) is vector space dual to homology:

$$H^p(K) \cong H_p(K)^* = \text{Hom}(H_p(K), \mathbb{F}).$$

Persistent (co)homology

How is persistent cohomology related to persistent homology?

- Cohomology (over \mathbb{F}) is vector space dual to homology:

$$H^p(K) \cong H_p(K)^* = \text{Hom}(H_p(K), \mathbb{F}).$$

- Duality preserves ranks of linear maps (in finite dimensions): for $f : V \rightarrow W$ with dual $f^* : W^* \rightarrow V^*$,

$$\text{rank } f^* = \text{rank } f.$$

Persistent (co)homology

How is persistent cohomology related to persistent homology?

- Cohomology (over \mathbb{F}) is vector space dual to homology:

$$H^p(K) \cong H_p(K)^* = \text{Hom}(H_p(K), \mathbb{F}).$$

- Duality preserves ranks of linear maps (in finite dimensions): for $f : V \rightarrow W$ with dual $f^* : W^* \rightarrow V^*$,

$$\text{rank } f^* = \text{rank } f.$$

- The persistence barcode of a persistence module is uniquely determined by the ranks of the internal maps.

Persistent (co)homology

How is persistent cohomology related to persistent homology?

- Cohomology (over \mathbb{F}) is vector space dual to homology:

$$H^p(K) \cong H_p(K)^* = \text{Hom}(H_p(K), \mathbb{F}).$$

- Duality preserves ranks of linear maps (in finite dimensions): for $f : V \rightarrow W$ with dual $f^* : W^* \rightarrow V^*$,

$$\text{rank } f^* = \text{rank } f.$$

- The persistence barcode of a persistence module is uniquely determined by the ranks of the internal maps.

Thus, persistent homology and persistent cohomology have the same barcode [de Silva et al 2011].

Persistent (relative) homology

How is persistent relative homology related to persistent homology?

- The short exact sequence of filtered chain complexes (with coefficients in \mathbb{F})

$$0 \rightarrow C_p(K_\bullet) \rightarrow C_p(K) \rightarrow C_p(K, K_\bullet) \rightarrow 0$$

induces a long exact sequence in persistent homology

$$\cdots \rightarrow H_{p+1}(K, K_\bullet) \rightarrow H_p(K_\bullet) \rightarrow H_p(K) \rightarrow H_p(K, K_\bullet) \rightarrow \cdots$$

and analogously for persistent cohomology.

Persistent (relative) homology barcodes

Consider the long exact sequence

$$\cdots \rightarrow H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \rightarrow \cdots$$

We have:

- $B(\ker(i)) = \{[b, d] \in B(H_p(K_\bullet))\} = B(\text{im}(\partial))$

Persistent (relative) homology barcodes

Consider the long exact sequence

$$\cdots \rightarrow H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \rightarrow \cdots$$

We have:

- $B(\ker(i)) = \{[b, d] \in B(H_p(K_\bullet))\} = B(\text{im}(\partial))$
- $B(\text{im}(i)) = \{[b, \infty] \in B(H_p(K_\bullet))\}$

Persistent (relative) homology barcodes

Consider the long exact sequence

$$\cdots \rightarrow H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \rightarrow \cdots$$

We have:

- $B(\ker(i)) = \{[b, d] \in B(H_p(K_\bullet))\} = B(\text{im}(\partial))$
- $B(\text{im}(i)) = \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $H_p(K_\bullet) \cong \text{im}(\partial) \oplus \text{im}(i)$

Persistent (relative) homology barcodes

Consider the long exact sequence

$$\cdots \rightarrow H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \rightarrow \cdots$$

We have:

- $B(\ker(i)) = \{[b, d] \in B(H_p(K_\bullet))\} = B(\text{im}(\partial))$
- $B(\text{im}(i)) = \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $H_p(K_\bullet) \cong \text{im}(\partial) \oplus \text{im}(i)$
- $B(\text{coker}(i)) = \{(-\infty, b) \mid [b, \infty) \in B(H_p(K_\bullet))\} = B(\text{im}(r))$

Persistent (relative) homology barcodes

Consider the long exact sequence

$$\cdots \rightarrow H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \rightarrow \cdots$$

We have:

- $B(\ker(i)) = \{[b, d] \in B(H_p(K_\bullet))\} = B(\text{im}(\partial))$
- $B(\text{im}(i)) = \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $H_p(K_\bullet) \cong \text{im}(\partial) \oplus \text{im}(i)$
- $B(\text{coker}(i)) = \{(-\infty, b) \mid [b, \infty) \in B(H_p(K_\bullet))\} = B(\text{im}(r))$
- $H_{p+1}(K, K_\bullet) \cong \text{im}(\partial) \oplus \text{im}(r)$

Persistent (relative) homology barcodes

Consider the long exact sequence

$$\dots \rightarrow H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \rightarrow \dots$$

We have:

- $B(\ker(i)) = \{[b, d) \in B(H_p(K_\bullet))\} = B(\text{im}(\partial))$
- $B(\text{im}(i)) = \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $H_p(K_\bullet) \cong \text{im}(\partial) \oplus \text{im}(i)$
- $B(\text{coker}(i)) = \{(-\infty, b) \mid [b, \infty) \in B(H_p(K_\bullet))\} = B(\text{im}(r))$
- $H_{p+1}(K, K_\bullet) \cong \text{im}(\partial) \oplus \text{im}(r)$

Thus, the *absolute* and *relative* persistence barcodes determine each other [de Silva et al 2011; Schmahl 2018].

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0 (avoiding the initial overhead)

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0 (avoiding the initial overhead)
- Persistence in dimension 0 has special algorithms (Kruskal's algorithm for MST, union-find data structure)

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0 (avoiding the initial overhead)
- Persistence in dimension 0 has special algorithms (Kruskal's algorithm for MST, union-find data structure)
- Persistent cohomology arises from a *reverse* filtration

$$C^*(K_0) \leftarrow C^*(K_1) \leftarrow \dots \leftarrow C^*(K_n)$$

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0 (avoiding the initial overhead)
- Persistence in dimension 0 has special algorithms (Kruskal's algorithm for MST, union-find data structure)
- Persistent cohomology arises from a *reverse* filtration

$$C^*(K_0) \leftarrow C^*(K_1) \leftarrow \dots \leftarrow C^*(K_n)$$

- Persistent *relative* cohomology arises from a filtration

$$C^*(K, K_0) \leftarrow C^*(K, K_1) \leftarrow \dots \leftarrow C^*(K, K_n)$$

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0 (avoiding the initial overhead)
- Persistence in dimension 0 has special algorithms (Kruskal's algorithm for MST, union-find data structure)
- Persistent cohomology arises from a *reverse* filtration

$$C^*(K_0) \leftarrow C^*(K_1) \leftarrow \dots \leftarrow C^*(K_n)$$

- Persistent *relative* cohomology arises from a filtration

$$C^*(K, K_0) \leftarrow C^*(K, K_1) \leftarrow \dots \leftarrow C^*(K, K_n)$$

- Can be computed by reduction of coboundary matrix: transpose of boundary matrix, with reversed index orders

Counting cohomology column reductions

Consider K : $k + 1$ -skeleton of $n - 1$ -simplex, Rips filtration

Number of columns in coboundary matrix:

$$\underbrace{\sum_{d=0}^k \binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=0}^k \binom{n-1}{d+1}}_{\text{death}} + \underbrace{\sum_{d=0}^k \binom{n-1}{d}}_{\text{birth}}$$

Counting cohomology column reductions

Consider K : $k + 1$ -skeleton of $n - 1$ -simplex, Rips filtration

Number of columns in coboundary matrix:

$$\begin{aligned} \underbrace{\sum_{d=0}^k \binom{n}{d+1}}_{\text{total}} &= \underbrace{\sum_{d=0}^k \binom{n-1}{d+1}}_{\text{death}} + \underbrace{\sum_{d=0}^k \binom{n-1}{d}}_{\text{birth}} \\ &= \underbrace{\sum_{d=0}^k \binom{n-1}{d+1}}_{\text{death}} + \underbrace{\binom{n-1}{0}}_{\text{essential}} + \underbrace{\sum_{d=1}^k \binom{n-1}{d}}_{\text{cleared}} \end{aligned}$$

Counting cohomology column reductions

Consider K : $k + 1$ -skeleton of $n - 1$ -simplex, Rips filtration

Number of columns in coboundary matrix:

$$\begin{aligned} \underbrace{\sum_{d=0}^k \binom{n}{d+1}}_{\text{total}} &= \underbrace{\sum_{d=0}^k \binom{n-1}{d+1}}_{\text{death}} + \underbrace{\sum_{d=0}^k \binom{n-1}{d}}_{\text{birth}} \\ &= \underbrace{\sum_{d=0}^k \binom{n-1}{d+1}}_{\text{death}} + \underbrace{\binom{n-1}{0}}_{\text{essential}} + \underbrace{\sum_{d=1}^k \binom{n-1}{d}}_{\text{cleared}} \end{aligned}$$

$$k = 2, n = 192: \quad 1179\,808 = 1161\,471 + 1 + 18\,336$$

Observations

For a typical input:

- V has very few off-diagonal entries
- most death columns of D are already reduced

Observations

For a typical input:

- V has very few off-diagonal entries
- most death columns of D are already reduced

Previous example ($k = 2, n = 192$):

- Only $79 + 42 + 1 = 122$ out of $192 + 18\,145 + 1\,143\,135 = 1\,161\,471$ columns are actually modified in matrix reduction

Implicit matrix reduction

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
 - Explicitly generated and stored in memory

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
 - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
 - transform D into R by column operations

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
 - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
 - transform D into R by column operations

Approach for Ripser:

- Boundary matrix D for lexicographically ordered basis
 - Implicitly defined and recomputed when needed

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
 - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
 - transform D into R by column operations

Approach for Ripser:

- Boundary matrix D for lexicographically ordered basis
 - Implicitly defined and recomputed when needed
- Matrix reduction in Ripser: store only coefficient matrix V
 - recompute previous columns of $R = D \cdot V$ when needed
 - Typically, V is much sparser and smaller than R
 - Also used in Dionysus, Gudhi

Implicit boundary matrix reduction algorithm

Require: $D: I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V : regular upper triangular,
 P : persistence pairs

for $j \in J$ in increasing order **do**

$v_j := e_j, r_j := d_j$

while $\exists k < j$ with $i := \text{Pivot } r_j$ and $(i, k) \in P$ **do**

$\lambda = r_{ij}$

$r_j := r_j - \lambda \cdot D \cdot v_k$

▷ eliminate pivot entry r_{ij}

$v_j := v_j - \lambda \cdot v_k$

if $i := \text{Pivot } r_j \neq 0$ **then**

$v_j = r_{ij}^{-1} \cdot v_j$

▷ make pivot entry $r_{ij} = 1$

append (i, j) to P

return V

Implicit boundary matrix reduction algorithm

Require: $D: I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V : regular upper triangular,
 P : persistence pairs

for $j \in J$ in increasing order **do**

$v_j := e_j, r_j := d_j$

while $\exists k < j$ with $i := \text{Pivot } r_j$ and $(i, k) \in P$ **do**

$\lambda = r_{ij}$

$r_j := r_j - \lambda \cdot D \cdot v_k$

▷ eliminate pivot entry r_{ij}

$v_j := v_j - \lambda \cdot v_k$

if $i := \text{Pivot } r_j \neq 0$ **then**

$v_j = r_{ij}^{-1} \cdot v_j$

▷ make pivot entry $r_{ij} = 1$

append (i, j) to P

return V

Implicit boundary matrix reduction algorithm

Require: $D: I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V : regular upper triangular,
 P : persistence pairs

for $j \in J$ in increasing order **do**

$v_j := e_j, r_j := d_j$

while $\exists k < j$ with $i := \text{Pivot } r_j$ and $(i, k) \in P$ **do**

$\lambda = r_{ij}$

$r_j := r_j - \lambda \cdot D \cdot v_k$

▷ eliminate pivot entry r_{ij}

$v_j := v_j - \lambda \cdot v_k$

if $i := \text{Pivot } r_j \neq 0$ **then**

$v_j = r_{ij}^{-1} \cdot v_j$

▷ make pivot entry $r_{ij} = 1$

append (i, j) to P

return V

Implicit boundary matrix reduction algorithm

Require: $D: I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V : regular upper triangular,
 P : persistence pairs

for $j \in J$ in increasing order **do**

$v_j := e_j, r_j := d_j$

while $\exists k < j$ with $i := \text{Pivot } r_j$ and $(i, k) \in P$ **do**

$\lambda = r_{ij}$

$r_j := r_j - \lambda \cdot D \cdot v_k$

▷ eliminate pivot entry r_{ij}

$v_j := v_j - \lambda \cdot v_k$

if $i := \text{Pivot } r_j \neq 0$ **then**

$v_j = r_{ij}^{-1} \cdot v_j$

▷ make pivot entry $r_{ij} = 1$

append (i, j) to P

return V

Implementation details

Current (working) columns v_j, r_j :

- priority queue (heap), comparison-based
- representing a linear combination of row basis elements
- elements: tuples consisting of
 - coefficient
 - row index
 - diameter of corresponding simplex
- pivot entry lazily evaluated (when extracting top element)

Previous (finalized) columns v_k ($k < j$):

- sparse matrix data structure

Apparent and emergent pairs

Apparent pairs

Definition

In a simplexwise filtration, (σ, τ) is an *apparent* pair if

- σ is the youngest face of τ
- τ is the oldest coface of σ

Apparent pairs

Definition

In a simplexwise filtration, (σ, τ) is an *apparent* pair if

- σ is the youngest face of τ
- τ is the oldest coface of σ

Lemma

The apparent pairs are persistence pairs.

Lemma

The apparent pairs form a discrete gradient.

- Generalizes a construction proposed by [Kahle 2011] for the study of random Rips filtrations
- Apparent pairs also appear (independently) in Eirene [Henselmann 2016] and in [Mendoza-Smith, Tanner 2017]

Emergent persistent pairs

A persistence pair (σ, τ) for a simplexwise filtration is

- an *emergent face pair* if σ is the youngest proper face of τ ,
- an *emergent coface pair* if τ is the oldest proper coface of σ .

Emergent persistent pairs

A persistence pair (σ, τ) for a simplexwise filtration is

- an *emergent face pair* if σ is the youngest proper face of τ ,
- an *emergent coface pair* if τ is the oldest proper coface of σ .

Lemma

Consider the lexicographically refined Rips filtration. Assume that

- τ is the lexicographically minimal proper coface of σ with $\text{diam}(\tau) = \text{diam}(\sigma)$, and
- τ is not already in a persistence pair (ρ, τ) with $\rho > \sigma$.

Then (σ, τ) is a 0-persistence emergent coface pair.

Emergent persistent pairs

A persistence pair (σ, τ) for a simplexwise filtration is

- an *emergent face pair* if σ is the youngest proper face of τ ,
- an *emergent coface pair* if τ is the oldest proper coface of σ .

Lemma

Consider the lexicographically refined Rips filtration. Assume that

- τ is the lexicographically minimal proper coface of σ with $\text{diam}(\tau) = \text{diam}(\sigma)$, and
- τ is not already in a persistence pair (ρ, τ) with $\rho > \sigma$.

Then (σ, τ) is a 0-persistence emergent coface pair.

- Includes all apparent pairs with persistence 0
- Can be identified *without* enumerating all cofaces of σ (shortcut for computation)

Speedup from emergent pairs shortcut

Previous example ($k = 2, n = 192$):

- Only $36\,672 + 164\,214 + 3\,392\,039 = 3\,592\,925$ out of $36\,672 + 3\,447\,550 + 216\,052\,515 = 219\,536\,737$ nonzero entries of the coboundary matrix are actually visited

Speedup from emergent pairs shortcut

Previous example ($k = 2, n = 192$):

- Only $36\,672 + 164\,214 + 3\,392\,039 = 3\,592\,925$ out of $36\,672 + 3\,447\,550 + 216\,052\,515 = 219\,536\,737$ nonzero entries of the coboundary matrix are actually visited

Using implicit reduction (boundary matrix columns may be revisited multiple times):

- Only $155\,474 + 253\,134 + 7\,500\,332 = 7\,908\,940$ out of $155\,474 + 3\,536\,470 + 220\,160\,808 = 223\,852\,752$ nonzero entries are actually visited

Speedup: 1.2 s vs. 5.6 s (factor 4.7)

Ripser Live: users from 567 different cities

