A SUFFICIENT THEORETICAL CONSTRUCTION OF A PERSISTENT HOMOLOGY PROGRAM

ROSALIND W. FINLEY

Abstract. This paper contains a theoretical construction of a simple persistent homology program. This paper does not introduce original concepts; it reflects the author's understanding of the referenced works.

1. Introduction

The structure of this paper is that of a general persistent homology program. We outline the chapters:

1. Introduction
This chapter is outlining the paper.

2. Input and Order
Input parameters vary among persistent homology programs and generally include a metric space, a simplicial complex type, filtration parameters, and a coefficient group. In chapter 2 we give input argument assumptions for this paper then proceed to the first steps in a program's internal operations: construct the simplicial complex and filtration determined by the given input and totally order the complex.

3. Homology
We define a homology class $[c]$, an algebraic object that represents a topological feature in the filtration from chapter 2. A combinatorial object called a persistence pair measures such a feature with respect to the filtration.

4. Persistence Pairs
We define persistence pairs. The chapter concludes with the definition of pair groups, algebraic representations of persistence pairs.

5. Boundary Matrices
Given the totally ordered complex from chapter 2, we construct boundary matrix $[D]$ and define a reduced form of $[D]$ that identifies a unique persistence pair for each class $[c]$. A subcollection of these persistence pairs is the program output. The remaining chapters 6-9 discuss methods for reducing $[D]$.

6. The Standard Algorithm
We define a naive algorithm for reducing $[D]$.

7. Clearing
We define a reduction operation that leverages the fact that $[D]^2 = 0$.

8. The Twist Algorithm
We define a modification of the Standard algorithm that incorporates clearing.

9. Cohomology
We define the filtration coboundary matrix $[D]_\perp$ then introduce a broader class of reduction algorithms for use in a persistent homology program. This paper does not define cohomology.

Date: 2024.

2. Input and Order

§i.) Let the following program input be given: an ordered, $\mathbb{N}$-indexed, nonempty, finite set $X = \{x_0, \ldots, x_v\} \subset \mathbb{R}^n$ with the Euclidean metric d; a complex type of Vietoris-Rips with finite $\mathbb{N}$-indexed diameter set $\{\epsilon_i\}_{i=0}^{d}$ where $\epsilon_i \in \mathbb{R}_{\geq 0}$; coefficient field $\mathbb{Z}_2$. Argument restrictions, such as limits on cardinality $|X|$ and dimension n, are imposed explicitly by implementation or implicitly by computing and memory limits.

Let $VR_\epsilon \subseteq P(X)$ denote a Vietoris-Rips simplicial complex on X of diameter ϵ whose p-simplices are the $p+1$ point subsets $\sigma_p \subseteq X$ such that $d(x, y) < \epsilon$ for all $x, y \in \sigma_p$ where $p \in \mathbb{N}$ and $\epsilon \in \mathbb{R}_{\geq 0}$.

Let functor $f : (\{\epsilon_i\}_{i=0}^{d}, \leq) \to \mathbf{SC}$ given by $\epsilon_i \mapsto VR_{\epsilon_i}$ be an ϵ_i-indexed filtration. Since X is finite, the range of f contains $2^{|X|}$ or fewer distinct complexes $VR_{\epsilon_i} \subset \mathbf{SC}$. For each distinct VR_{ϵ_i}, let ϵ_{min} denote the minimum ϵ_i in the preimage $f^{-1}(VR_{\epsilon_i})$. The collection of ϵ_{min} forms a finite $\mathbb{N}$-indexed set $I = \{\epsilon_i\}_{i=0}^{m}$ where $\epsilon_i < \epsilon_j \iff i < j$. Then $f|_I$ is an injective functor that gives the finite essential Vietoris-Rips filtration

$$VR_{\epsilon_0} \subset VR_{\epsilon_1} \subset \cdots \subset VR_{\epsilon_{m\text{-}1}} \subset VR_{\epsilon_m}.$$

This completes the construction of the simplicial complex VR_{ϵ_m} and the filtration $f|_I$ uniquely determined by the input arguments.

We will derive a total order on VR_{ϵ_m} implicitly, by refining $f|_I$ to a simplexwise filtration f_σ.

Bauer [17] writes that Ripser orders simplices by diameter, then by dimension, and then by reverse colexicographic vertex order. Bauer uses the combinatorial number system to obtain the latter.

Note that $\{VR_{\epsilon_0}, VR_{\epsilon_1} \setminus VR_{\epsilon_0}, \ldots, VR_{\epsilon_m} \setminus VR_{\epsilon_{m\text{-}1}}\}$ orders simplices $\sigma_p \in VR_{\epsilon_m}$ by ascending diameter ϵ_i.

We derive order by dimension in section §ii. We adopt Bauer's combinatorial method to derive a reverse colexicographic vertex order in section §iii. The result is a simplexwise filtration f_σ without an ordered $\mathbb{N}$-index. Section §iv contains a nonessential observation. In section §v we derive an ordered $\mathbb{N}$-index for f_σ.

§ii.) Let $VR_{\epsilon_{\text{-}1}} = \emptyset$. Sort the set $VR_{\epsilon_i} \setminus VR_{\epsilon_{i\text{-}1}}$ of ϵ_i-diameter simplices by ascending dimension, for each $i \in \{0, \ldots, m\}$:

Let $D_i = \{p_{0^{(i)}}, p_{1^{(i)}}, p_{2^{(i)}}, \ldots, p_{d^{(i)}\text{-}1}, p_{d^{(i)}}\} \subset \{0, \ldots, v\}$ denote the ascending set of dimensions $p_{t^{(i)}}$ such that there exists a $p_{t^{(i)}}$-simplex $\sigma_{p_{t^{(i)}}} \in VR_{\epsilon_i} \setminus VR_{\epsilon_{i\text{-}1}}$. Let $\{\sigma_{p_{t^{(i)}}}\}$ denote the collection of $p_{t^{(i)}}$-simplices in $VR_{\epsilon_i} \setminus VR_{\epsilon_{i\text{-}1}}$. Let $VR_{\epsilon_{i\text{-}1}, p_{t^{(i)}}} = VR_{\epsilon_{i\text{-}1}} \bigcup \cup_{k^{(i)}=0^{(i)}}^{t^{(i)}} \{\sigma_{p_{k^{(i)}}}\}$ for each $t^{(i)} \in \{0^{(i)}, \ldots, d^{(i)}\} \subset \mathbb{N}$. Note that $d^{(i)}$ depends on i and may vary among D_i. We obtain a refinement of $f|_I$:

$$\begin{aligned}
\{\sigma_{p_{0^{(0)}}}\} \subset \cup_{k^{(0)}=0^{(0)}}^{1^{(0)}} \{\sigma_{p_{k^{(0)}}}\} \subset \cup_{k^{(0)}=0^{(0)}}^{2^{(0)}} \{\sigma_{p_{k^{(0)}}}\} \subset \cdots \subset \cup_{k^{(0)}=0^{(0)}}^{d^{(0)}\text{-}1} \{\sigma_{p_{k^{(0)}}}\} \subset \cup_{k^{(0)}=0^{(0)}}^{d^{(0)}} \{\sigma_{p_{k^{(0)}}}\} \\
= VR_{\epsilon_0} \subset VR_{\epsilon_0, p_{0^{(1)}}} \subset VR_{\epsilon_0, p_{1^{(1)}}} \subset VR_{\epsilon_0, p_{2^{(1)}}} \subset \cdots \subset VR_{\epsilon_0, p_{d^{(1)}\text{-}1}} \subset VR_{\epsilon_0, p_{d^{(1)}}} \\
= VR_{\epsilon_1} \subset VR_{\epsilon_1, p_{0^{(2)}}} \subset VR_{\epsilon_1, p_{1^{(2)}}} \subset VR_{\epsilon_1, p_{2^{(2)}}} \subset \cdots \subset VR_{\epsilon_1, p_{d^{(2)}\text{-}1}} \subset VR_{\epsilon_1, p_{d^{(2)}}} \\
\vdots \\
= VR_{\epsilon_{m\text{-}1}} \subset VR_{\epsilon_{m\text{-}1}, p_{0^{(m)}}} \subset VR_{\epsilon_{m\text{-}1}, p_{1^{(m)}}} \subset VR_{\epsilon_{m\text{-}1}, p_{2^{(m)}}} \subset \cdots \subset VR_{\epsilon_{m\text{-}1}, p_{d^{(m)}\text{-}1}} \subset VR_{\epsilon_{m\text{-}1}, p_{d^{(m)}}} \\
= VR_{\epsilon_m}.
\end{aligned}$$

Denote this filtration by $f|_{I_p}$. Observe that each complex $VR_{\epsilon_{i\text{-}1}, p_{t^{(i)}}}$ in $f|_{I_p}$ admits new simplices $\sigma_{p_{t^{(i)}}}$ of dimension $p_{t^{(i)}}$ exclusively, for each fixed pair i, t. Therefore, this completes the simplicial order by dimension, for each diameter ϵ_i.

§iii.) Derive a total order on the collection $\{\sigma_{p_{t^{(i)}}}\}$, for each fixed pair i, t:

Note that each $p_{t^{(i)}}$-simplex $\sigma_{p_{t^{(i)}}}$ is a $(p_{t^{(i)}}+1)$-point subset $\{x_{y_0}, \ldots, x_{y_{p_{t^{(i)}}}}\} \subset \{x_0, \ldots, x_v\} = X$ with the order inherited from X.

The map $\zeta : \{\sigma_{p_{t^{(i)}}}\} \to \mathbb{N}^{p_{t^{(i)}}+1}$ given by $\zeta(\sigma_{p_{t^{(i)}}}) = \{y_{p_{t^{(i)}}}, \ldots, y_0\}$ uniquely identifies each $p_{t^{(i)}}$-simplex with its descending index tuple $\{y_{p_{t^{(i)}}}, \ldots, y_0\} \subset \{v, \ldots, 0\}$.

Impose the dictionary order on the collection of tuples $\{\{y_{p_{t^{(i)}}}, \ldots, y_0\}\} \subset \mathbb{N}^{p_{t^{(i)}}+1}$.

Then the map $\iota : \{\{y_{p_{t^{(i)}}}, \dots, y_0\}\} \to \mathbb{N}$ given by $\iota(\{y_{p_{t^{(i)}}}, \dots, y_0\}) = \sum_{a=0}^{p_{t^{(i)}}} \binom{y_a}{a+1}$ where $\binom{y_a}{a+1} = 0$ if $y_a < a+1$ is an order-preserving injection from index tuples to natural numbers.

Let $m = \max\{n : n \in \operatorname{im}\iota\}$. Note that $m \leq \binom{v}{p_{t^{(i)}}+1} + \binom{v-1}{p_{t^{(i)}}} + \binom{v-2}{p_{t^{(i)}}-1} + \cdots + \binom{v-p_{t^{(i)}}}{1} = \binom{v+1}{p_{t^{(i)}}+1} - 1 \in \mathbb{N}$. Then there exists a subset $\mathcal{L} \subseteq \{0, \dots, m\} \subset \mathbb{N}$ such that the map $\iota \circ \zeta : \{\sigma_{p_{t^{(i)}}}\} \to \mathcal{L}$ is an order-preserving bijection between $p_{t^{(i)}}$-simplices and natural numbers. Thus $\mathcal{L}$ is an ordered index set for $\{\sigma_{p_{t^{(i)}}}\}$ with index map $\zeta^{-1} \circ \iota^{-1} : \mathcal{L} \to \{\sigma_{p_{t^{(i)}}}\}$. We define this map algorithmically with C++ pseudocode functions:

Observe that:

$$
\begin{aligned}
y_{p_{t^{(i)}}} &= \max\Big\{c \in \{0, \dots, v\} : \tbinom{c}{p_{t^{(i)}}+1} \leq n\Big\},\\
y_{p_{t^{(i)}}-1} &= \max\Big\{c \in \{0, \dots, y_{p_{t^{(i)}}} - 1\} : \tbinom{c}{p_{t^{(i)}}} \leq n - \tbinom{y_{p_{t^{(i)}}}}{p_{t^{(i)}}+1}\Big\},\\
y_{p_{t^{(i)}}-2} &= \max\Big\{c \in \{0, \dots, y_{p_{t^{(i)}}-1} - 1\} : \tbinom{c}{p_{t^{(i)}}-1} \leq n - \tbinom{y_{p_{t^{(i)}}}}{p_{t^{(i)}}+1} - \tbinom{y_{p_{t^{(i)}}-1}}{p_{t^{(i)}}}\Big\},\\
&\vdots\\
y_1 &= \max\Big\{c \in \{0, \dots, y_2 - 1\} : \tbinom{c}{2} \leq n - \tbinom{y_{p_{t^{(i)}}}}{p_{t^{(i)}}+1} - \tbinom{y_{p_{t^{(i)}}-1}}{p_{t^{(i)}}} - \cdots - \tbinom{y_3}{4} - \tbinom{y_2}{3}\Big\},\\
y_0 &= \max\Big\{c \in \{0, \dots, y_1 - 1\} : \tbinom{c}{1} \leq n - \tbinom{y_{p_{t^{(i)}}}}{p_{t^{(i)}}+1} - \tbinom{y_{p_{t^{(i)}}-1}}{p_{t^{(i)}}} - \cdots - \tbinom{y_2}{3} - \tbinom{y_1}{2}\Big\}.
\end{aligned}
$$

Then the C++ pseudocode implementation of ι^{-1} is given by:

```
int* ι⁻¹(int n ∈ ℒ, int p_{t^(i)}) {
   int Y[p_{t^(i)}+1];  //array to hold descending index tuple
   int c = v;
   int sum = 0;  //sum of binomial coefficient terms to subtract from n
   for (int d = p_{t^(i)}; d ≥ 0; d--) {
     while (C(c, d+1) > n − sum) {
       c − −;}
     Y[d] = c;
     c = Y[d] − 1;
     sum += C(Y[d], d+1);}
   return Y;},
```

and the C++ pseudocode implementation of ζ^{-1} is given by:

```
int* ζ⁻¹(int Y[], p_{t^(i)}) {
   SC σ_{p_{t^(i)}};
   for (int c = p_{t^(i)}; c ≥ 0; c--) {
     σ_{p_{t^(i)}}[c] = x_{Y[c]};  //suppose SC is an array type object
   return σ_{p_{t^(i)}};}.
```

This completes the total ordering of $\{\sigma_{p_{t^{(i)}}}\}$, thus inducing a simplexwise refinement of $f|_{I_p}$.

To denote this refinement, suppose that $VR_{\epsilon_{i-1}, p_{t^{(i)}}}$ admits $N^{(i,t)} \in \mathbb{N}_{\geq 1}$ new simplices, $\{\sigma^{(1)}_{p_{t^{(i)}}}, \dots, \sigma^{(N^{(i,t)})}_{p_{t^{(i)}}}\}$, for each fixed pair i, t.

Then each complex $VR_{\epsilon_{i-1}, p_{t^{(i)}}}$ in $f|_{I_p}$ is replaced with a filtration containing $N^{(i,t)}$ complexes:

$$
VR_{\epsilon_{i-1}, p_{t^{(i)}}} \setminus \bigcup_{b=2}^{N^{(i,t)}} \sigma^{(b)}_{p_{t^{(i)}}} \subset VR_{\epsilon_{i-1}, p_{t^{(i)}}} \setminus \bigcup_{b=3}^{N^{(i,t)}} \sigma^{(b)}_{p_{t^{(i)}}} \subset \cdots \subset VR_{\epsilon_{i-1}, p_{t^{(i)}}} \setminus \bigcup_{b=N^{(i,t)}-1}^{N^{(i,t)}} \sigma^{(b)}_{p_{t^{(i)}}} \subset VR_{\epsilon_{i-1}, p_{t^{(i)}}} \setminus \sigma^{(N^{(i,t)})}_{p_{t^{(i)}}} \subset VR_{\epsilon_{i-1}, p_{t^{(i)}}}.
$$

This gives the induced simplexwise refinement of $f|_{I_p}$. Denote this refinement by f_σ.

This completes the simplexwise filtration of VR_{ϵ_m}. Note that this total simplicial order on VR_{ϵ_m} depended only on diameters $\{\epsilon_i\}_{i=0}^m$.

§iv.) Observation: Note that each complex in f_σ can be uniquely identified by the triplet $(i, t, b_{min}) \in \mathbb{N}^3$, where b_{min} is the minimum index of the removed union. Let $b_{min} = 1$ for each complex $VR_{\epsilon_{i\text{-}1}, p_{f(i)}} \in f|_{I_p}$. Let the map $\rho : \mathbb{N}^2 \to \mathbb{N}$ be given by $\rho(i, t) = 2^{i-1}(2t - 1)$. Note that ρ is a bijection. Then the map $\phi : \mathbb{N}^3 \to \mathbb{N}$ given by $\phi(i, t, b_{min}) = \rho(\rho(i, t), b_{min})$ is a bijection between triplets and natural numbers. That is, (i, t, b_{min}) can be uniquely identified with the natural number $\phi(i, t, b_{min})$. The collection $\{\phi(i, t, b_{min})\}$ is not ordered so it is not a meaningful index set for f_σ. However, if we perhaps wish to retain the information contained in the triplet and so encode an identification between complexes and triplets, it may be efficient to store triplets in a natural number. We could incorporate computation of triplets into the section §v function refinDeX.

§v.) Simplify notation by rewriting f_σ with an ordered $\mathbb{N}$-index:
Note: we proceed from a programmatic context, departing from the theoretical setting of sections §i-iv. Had sections §i-iii been programmatic rather than theoretic, an ordered $\mathbb{N}$-index of f_σ would have been a natural consequence in section §iii and this section §v would be unnecessary.

We do not explicitly store filtration index sets. A filtration is represented by an array of simplicial complex type data structures and the array index represents the complex index. We write a C++ pseudocode function refinDeX that returns array f_σ, given input array $f|_I$, where $f|_I[i] = VR_{\epsilon_i}$, $i \in \{0, \ldots, m\}$. Note that f_σ contains $\sum_{i=0}^{m} \sum_{t=0}^{|D_i|\cdot 1} N^{(i,t)} \leq 2^{|X|}$ complexes.

```
int* refinDeX(f|_I, |X|) {
   SC f_σ[2^|X|];  //declare array of simplicial complex (SC) objects
   int curr = 0;  //current number of SC in f_σ
   int |D_i|;  //count of dimensions added at index i in f|_I
   int N^(i,t);  //count of simplices per dimension
   for (int i = 0; i ≤ m; i++) {
     |D_i| = f|_I[i].getdimcount();  //suppose that such a function exists
     for (int t = 0; t ≤ |D_i| − 1; t++) {
       for (int y = 0; y ≤ N^(i,t) − 1; y++) {
         f_σ[curr + y] = σ^(y+1)_{p_{f(i)}};}
       curr += N^(i,t);}}
   int s = f_σ.length() - 1;
   return f_σ;}
```

We now denote f_σ by

$$K_0 \subset K_1 \subset \cdots \subset K_{s\text{-}1} \subset K_s = VR_{\epsilon_m}.$$

3. Homology

Let $\{\sigma_p\} \subset K_r \subset P(X)$ be the set of p-simplices in K_r, $r \in \{0, \ldots, s\}$ with the simplicial orientation induced by the order on X. A p-chain is a map κ on $\{\sigma_p\}$ to the formal finite sum $\Sigma_{i=1}^{|\{\sigma_p\}|}\eta_i\sigma_p^{(i)}$, with coefficients η_i in an arbitrary Abelian group G and $\sigma_p^{(i)} \in \{\sigma_p\}$, such that $\kappa(\sigma_p) = \text{-}\kappa(\sigma'_p)$ if σ_p and σ'_p are opposite orientations. The collection of p-chains is a free Abelian additive group, denoted $C_p(K_r)$, with basis $\{\sigma_p\}$.

For simplicity we let $G = \mathbb{Z}_2$, consistent with the program input argument given in chapter 2, section §i. Then simplicial orientation is irrelevant, the sum of two simplices is the symmetric difference $\sigma_p + \sigma'_p = \{\sigma_p \cup \sigma'_p\} \backslash \{\sigma_p \cap \sigma'_p\}$, and $C_p(K_r)$ is a $|\{\sigma_p\}|-$dimensional vector space.

Let simplex $\sigma_p = \{v_0, \ldots, v_p\} \subset X$ be a basis chain in $C_p(K_r)$. Equivalently, let σ_p be a p-simplex in K_r. The boundary homomorphism $\delta_p^r : C_p(K_r) \to C_{p\text{-}1}(K_r)$ given by

$$\delta_p^r(\sigma_p) = \delta_p^r(\{v_0, \ldots, v_p\}) = \sum_{i=0}^{p}(-1)^i(\sigma_p \backslash \{v_i\})$$

is a linear map on chains:

$$\delta_p^r(\sum_{i=1}^{|\{\sigma_p\}|}\eta_i\sigma_p^{(i)}) = \sum_{i=1}^{|\{\sigma_p\}|}\eta_i\delta_p^r(\sigma_p^{(i)}).$$

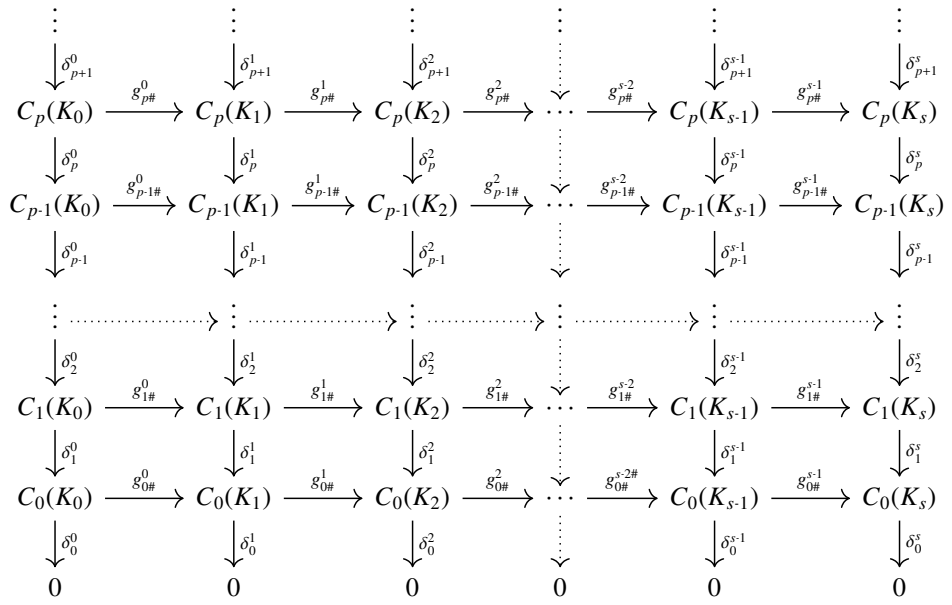
Note that $\delta_{p\text{-}1}^r \circ \delta_p^r = 0$. Equivalently, the image of δ_{p+1}^r is a subgroup of the kernel of δ_p^r:

$$\mathtt{im}\delta_{p+1}^r \subset \mathtt{ker}\delta_p^r \subset C_p(K_r).$$

Chains in $\mathtt{ker}\delta_p^r$ are called p-cycles. Chains in $\mathtt{im}\delta_{p+1}^r$ are called p-boundaries. For each $K_r \in f_\sigma$ we have the chain complex $\mathscr{C}_r = \{C_p(K_r), \delta_p^r\}$:

$$\cdots \xrightarrow{\delta_{p+1}^r} C_p(K_r) \xrightarrow{\delta_p^r} C_{p\text{-}1}(K_r) \xrightarrow{\delta_{p\text{-}1}^r} \cdots \xrightarrow{\delta_1^r} C_0(K_r) \xrightarrow{\delta_0^r} 0.$$

Let $g^r : K_r \to K_{r+1}$ be the simplicial inclusion map given by f_σ, for each $r \in \{0, \ldots, s\text{ - }1\}$. If $\sigma_p = \{v_0, \ldots, v_p\}$ is a p-simplex in K_r, then $g^r(\sigma_p) = \{g(v_0), \ldots, g(v_p)\}$ spans a q-simplex in K_{r+1}, $q \leq p$. Each g^r induces a chain map, a collection of homomorphisms $\{g_{p\#}^r : C_p(K_r) \to C_p(K_{r+1})\}$, one per dimension p, where $g_{p\#}^r(\sigma_p) = 0$ if $\mathtt{dim}g_{p\#}^r(\sigma_p) < p$. Since $\delta_p^{r+1} \circ g_{p\#}^r = g_{p\text{-}1\#}^r \circ \delta_p^r$, we have a commutative diagram:



Each column is a chain complex. Dimension is fixed in each row. A persistent homology group is computed across a row.

The quotient group $H_p(K_r) = \mathtt{ker}\delta_p^r \setminus \mathtt{im}\delta_{p+1}^r$ is called the p^{th} homology group of K_r. An element of $H_p(K_r)$ is a coset of $\mathtt{im}\delta_{p+1}^r$ in $\mathtt{ker}\delta_p^r$ called a homology class or a p-class:

$$\begin{aligned} H_p(K_r) &= \{c + B : c \in \mathtt{ker}\delta_p^r, B \in \mathtt{im}\delta_{p+1}^r\} \\ &= \{[c] : c \in \mathtt{ker}\delta_p^r\}. \end{aligned}$$

Representatives of a particular homology class are said to be homologous cycles. Although homology groups with coefficients in an arbitrary Abelian group G may contain torsion subgroups, $H_p(K_r)$ is free Abelian in coefficient field $G = \mathbb{Z}_2$. The topological invariant $\beta_p^r = \mathtt{rank}H_p(K_r)$ is called the Betti number of K_r in dimension p.

Since chain homomorphism $g_{p\#}^r$ and boundary homomorphism δ_p^r commute, $g_{p\#}^r$ preserves cycles and boundaries:

$$\begin{aligned} g_{p\#}^r(\mathtt{ker}\delta_p^r) &\subset \mathtt{ker}\delta_p^{r+1}, \\ g_{p\text{-}1\#}^r(\mathtt{im}\delta_p^r) &\subset \mathtt{im}\delta_p^{r+1}. \end{aligned}$$

Therefore, $g_{p\#}^r$ induces a homomorphism $g_{p*}^r : H_p(K_r) \to H_p(K_{r+1})$ and thus we have a persistence module for each dimension p:

$$H_p(K_0) \xrightarrow{g_{p*}^0} H_p(K_1) \xrightarrow{g_{p*}^1} H_p(K_2) \xrightarrow{g_{p*}^2} \cdots \xrightarrow{g_{p*}^{s\text{-}2}} H_p(K_{s\text{-}1}) \xrightarrow{g_{p*}^{s\text{-}1}} H_p(K_s).$$

Note that $(g_p^{r+1} \circ g_p^r)_\# = g_{p\#}^{r+1} \circ g_{p\#}^r$ and therefore $(g_p^{r+1} \circ g_p^r)_* = g_{p*}^{r+1} \circ g_{p*}^r$. Let $g_p^{i,j}$ denote the composition $g_{p*}^{j\text{-}1} \circ \cdots \circ g_{p*}^{i+1} \circ g_{p*}^i$, $0 \le i < j \le s$. The p^{th} persistent homology groups of K_s are the images $H_p^{i,j} = \mathtt{im}g_p^{i,j}$, for all pairs $i < j$ and $H_p^{i,j} = H_p(K_i)$ if $i = j$. Equivalently, the p^{th} persistent homology groups of K_s are the quotient groups:

$$H_p^{i,j} = \mathtt{ker}\delta_p^i \setminus (\mathtt{ker}\delta_p^i \cap \mathtt{im}\delta_{p+1}^j).$$

The topological invariant $\beta_p^{i,j} = \mathtt{rank}H_p^{i,j}$ is called the p^{th} persistent Betti number. We continue this algebraic discussion at the end of chapter 4. First we introduce persistence pairs.

4. Persistence Pairs

Suppose $\sigma_p^{(i)} = K_i \setminus K_{i\text{-}1}$ is the simplex added at index i in f_σ. Let $K_{-1} = \emptyset$. When $\sigma_p^{(i)}$ is added to $K_{i\text{-}1}$ then either one independent p-cycle and its associated p-class are created (so β_p^i equals $\beta_p^{i\text{-}1}+1$) or one existing independent p-1-cycle and its associated p-1-class are destroyed (so $\beta_{p\text{-}1}^i$ equals $\beta_{p\text{-}1}^{i\text{-}1}-1$). In the former situation, $\sigma_p^{(i)}$ is called a creator; in the latter, a destructor. The addition of a simplex is the only event that creates or destroys classes.

Each p-class $[c]$ is uniquely identified by a persistence pair $(\sigma_p^{(i)}, \sigma_{p+1}^{(j)})$, where $\sigma_p^{(i)}$ creates $[c]$ and $\sigma_{p+1}^{(j)}$ destroys $[c]$. A p-class represents a p-dimensional topological feature. The persistence pair gives the persistence $j-i$ of the feature in f_σ and identifies the particular complexes $K_i, \ldots, K_{j\text{-}1}$ spanned. If $[c]$ is never destroyed then $[c] \in H_p(K_s)$ and we let $\sigma_{p+1}^{(j)} = \infty$.

When destructor $\sigma_{p+1}^{(j)}$ is added at index j in f_σ, the independent destroyed cycle is determined uniquely up to homology to be the boundary $\delta_{p+1}^j(\sigma_{p+1}^{(j)})$ so the destroyed class $[\delta_{p+1}^j(\sigma_{p+1}^{(j)})]$ is uniquely determined. Thus we have a unique identification between destroyed classes and destructors. Hence the destructor in a persistence pair is uniquely determined. Determining the creator in a pair is a rather more opaque process.

When a p-class $[c]$ is created at index i, multiple cycles may be created, say c' and c''. Only one cycle is independent. The others are dependent. Dependent cycles are combinations of the independent cycle and other preexisting independent cycles, say c''' created at index $i' < i$. That is, $c' + c'' = c'''$. Unlike with destruction, the independent cycle is not uniquely determined or canonically defined and may be arbitrarily chosen from among the created cycles, c' and c''. Without loss of generality, if we choose $[c] = [c']$ then we are choosing c' as the independent cycle and we are choosing $\sigma_p^{(i)}$ as the creator of $[c']$ and $\sigma_p^{(i')}$ as the creator of $[c'']$. This is unsatisfactory since the persistence pair of a class should be uniquely determined. The creator should not be arbitrarily chosen. We introduce an algorithm that uniquely pairs creators with destructors, thus serving as a canonical determinant of the independent cycle and eliminating the independent/dependent cycle conundrum. It is called the Pairing algorithm:

i.) Let destructor $\sigma_{p+1}^{(j)}$ be given. We want to determine the paired creator.

ii.) Let cycle $c = \delta_{p+1}^j(\sigma_{p+1}^{(j)})$ and let $[c]$ denote the destroyed class.

iii.) Let $i_{max} = \mathtt{max}\{i : \sigma_p^{(i)} \in c\}$.

iv.) If $\sigma_p^{(i_{max})}$ is unpaired then $\sigma_p^{(i_{max})}$ is the creator and the algorithm terminates. Note that c is the independent cycle.

v.) Else if $\sigma_p^{(i_{max})}$ is paired then c is a dependent cycle and $\sigma_p^{(i_{max})}$ created a cycle $c' \neq c$ such that $[c']$ has persistence pair $(\sigma_p^{(i_{max})}, \sigma_{p+1}^{(i')})$, $i' \in \{i_{max}, \ldots, j\text{ - }1\}$. Set c equal to $c+c'$ and return to step iii.

The program output is the collection of persistence pairs $(\sigma_p^{(i)}, \sigma_{p+1}^{(j)})$ such that $K_i, K_j \in f|_I$. Such persistence pairs depend exclusively upon program input. Note that internal operations may give rise to other, extraneous, persistence pairs. In particular, if $K_i \in f_\sigma \setminus f|_I$ or $K_j \in f_\sigma \setminus f|_I$, then $(\sigma_p^{(i)}, \sigma_{p+1}^{(j)})$ is a byproduct of simplicial ordering.

We continue the algebraic discussion of chapter 3:

Whereas β_p^i counts all p-classes in $H_p(K_i)$, $\beta_p^{i,j}$ counts those p-classes in $H_p(K_i)$ that persist through $H_p(K_j)$. Such classes may also exist in $H_p(K_t)$, $t < i$ or $t > j$. The p-classes that are created at i and destroyed at j form a p^{th} pair group of K_s:

$$P_p^{i,j} = (\mathtt{img}_{p*}^{i,j\text{-}1} \cap \mathtt{ker} g_{p*}^{j\text{-}1,j}) \setminus (\mathtt{img}_{p*}^{i\text{-}1,j\text{-}1} \cap \mathtt{ker} g_{p*}^{j\text{-}1,j}),$$

with associated pair count:

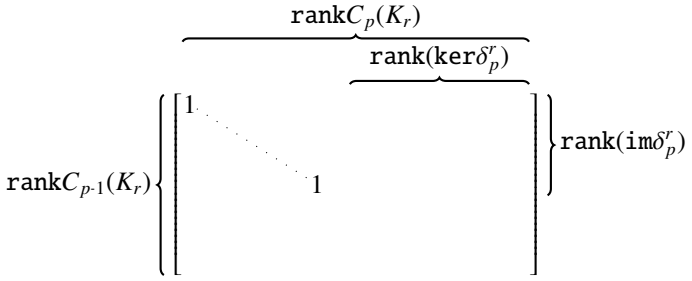
$$\mu_p^{i,j} = \beta_p^{i,j\text{-}1} - \beta_p^{i,j} - (\beta_p^{i\text{-}1,j\text{-}1} - \beta_p^{i\text{-}1,j}).$$

Since f_σ is a simplexwise filtration then either $P_p^{i,j}$ is equal to the unique class $[c]$ with persistence pair $(\sigma_p^{(i)}, \sigma_{p+1}^{(j)})$ and $\mu_p^{i,j} = 1$ or $P_p^{i,j}$ is empty and $\mu_p^{i,j} = 0$.

5. Boundary Matrices

Each boundary homomorphism $\delta_p^r : C_p(K_r) \to C_{p\text{-}1}(K_r)$ is represented by a matrix $[\delta_p^r]$ where columns correspond to $C_p(K_r)$ basis simplices, rows correspond to $C_{p\text{-}1}(K_r)$ basis simplices, and $[\delta_p^r]_{ij} \in \{0, 1, -1\}$ is equal to the coefficient of the i^{th} row basis simplex $\sigma_{i\bullet}$ in the formal sum $\delta_p^r(\sigma_{\bullet j})$. Note that $[\delta_p^r]_{ij} = 0$ if $\sigma_{i\bullet}$ is not a face of $\sigma_{\bullet j}$ and $-1 = 1$ given input parameter $G = \mathbb{Z}_2$.

The Smith normal form of $[\delta_p^r]$, denoted by $S[\delta_p^r]$ and derived by elementary row and column operations, gives $\texttt{rank}(\texttt{ker}\delta_p^r)$ and $\texttt{rank}(\texttt{im}\delta_p^r)$. Similarly, $S[\delta_{p+1}^r]$ gives $\texttt{rank}(\texttt{ker}\delta_{p+1}^r)$ and $\texttt{rank}(\texttt{im}\delta_{p+1}^r)$. Thus given the p and p+1 boundary matrices we can derive the Betti number $\beta_p^r = \texttt{rank}C_p(K_r) - \texttt{rank}S[\delta_p^r] - \texttt{rank}S[\delta_{p+1}^r]$. Moreover, we can augment $[\delta_p^r]$ with identity matrices to record the change of bases induced by elementary operations in the process of reducing $[\delta_p^r]$ to $S[\delta_p^r]$. This gives bases for $\texttt{ker}\delta_p^r$ and $\texttt{im}\delta_p^r$ so we can compute bases for homology groups.



The reduced Smith normal form of $[\delta_p^r]$, denoted by $S[\delta_p^r]$.

Rather than use individual boundary matrices, persistent homology programs represent δ_p^r, for all pairs r, p, in a single matrix called a filtration boundary matrix, denoted by $[D]$. Rows and columns correspond to simplices, with order given by f_σ. That is, row $[D]_{i\bullet}$ and column $[D]_{\bullet i}$ correspond to simplex $\sigma^{(i)} = K_i \setminus K_{i\text{-}1}$, $i \in \{0, \ldots, s\}$. Let $K_{\text{-}1} = \emptyset$. Note that we index $[D]$ by zero. Element $[D]_{ij}$ is 1 if $\sigma^{(i)}$ is in the boundary of $\sigma^{(j)}$ and zero otherwise.

Define $\texttt{low}[D]_{\bullet j}$ to be $\texttt{max}\{i : [D]_{ij} = 1\}$ if column j is nonzero and to be undefined otherwise. A matrix $[D]$ is said to be reduced if $\texttt{low}[D]_{\bullet j} \neq \texttt{low}[D]_{\bullet j'}$ for all distinct columns $j \neq j'$. Let $[R]$ denote the reduced form of $[D]$. If the row $\texttt{low}[R]_{\bullet j}$, column j element of $[R]$ is nonzero then it is said to be a pivot, denoted $\texttt{piv}_j$.

A persistence pair $(\sigma_p^{(i)}, \sigma_{p+1}^{(j)})$ is uniquely identified with $\texttt{piv}_j$, where $i = \texttt{low}[R]_{\bullet j}$ and $\sigma_{p+1}^{(j)} \neq \infty$.

A persistence pair $(\sigma_p^{(i)}, \infty)$ is uniquely identified by a zero i^{th} row $[R]_{i\bullet}$ (*i.e.* $\sigma_p^{(i)}$ creates a class that is never destroyed) and an undefined $\texttt{low}[R]_{\bullet i}$ (*i.e.* the absence of a pivot in column $[R]_{\bullet i}$ indicates that $\sigma_p^{(i)}$ is not a destructor).

Hence the strategy of persistent homology programs to identify pivots and thus persistence pairs.

A procedure to reduce $[D]$ to $[R]$ is called a reduction algorithm. Persistent homology programs vary in what reduction algorithms they implement. A reduction algorithm is generally chosen to minimize runtime or memory usage. Both factors depend on input parameters.

6. The Standard Algorithm

The Standard reduction algorithm processes columns of $[D]$ one at a time by ascending column index. The column being processed is called the active column. Processing consists of elementary column operations on the active column by lower index columns. Processing is complete when the active column is either a zero column or contains a pivot. We implement the Standard algorithm in a C++ pseudocode function:

```
int* standard([D]) {
   int s+1 = [D].length();
   int R[s+1][s+1];
   int low[R]•j;
   [R]•0 = [D]•0;  //set first column of [R] equal to first column of [D]
   for (int j = 1; j ≤ s; j++) {  //traverse columns of [D]
     low[R]•j = low[D]•j;  //initiate low[R]•j
     int left = 0;
     while (left < j) {
       if (low[R]•j == low[R]•left) {
         [R]•j = [R]•j + [R]•left;
         left = 0;}
       else if (low[R]•j != low[R]•left) {
         left += 1;}}}
   return [R];}
```

Rather than reducing $[D]$ to $[R]$, the `standard()` function views $[D]$ and constructs a new object $[R]$ so $[D]$ remains unchanged.

If the parameter is a copy of $[D]$ rather than a pointer to $[D]$ then `standard()` can be simplified to reduce the copy directly rather than to construct $[R]$. Runtime or memory usage may increase by such a parameter type modification. If we still pass a pointer to $[D]$ but reduce $[D]$ directly rather than construct $[R]$ then `standard()` may be similarly simplified and return 0 rather than $[R]$. Then runtime and memory usage may be minimized but $[D]$ is changed.

This completes our construction of a simple persistent homology program.

The remaining chapters discuss some common reduction algorithms.

7. Clearing

Suppose that we are processing active column $[D]_{\bullet j}$ and we complete processing column $[D]_{\bullet j}$ by identifying pivot $\mathtt{piv}_j$ at row $\mathtt{low}[D]_{\bullet j}$. This pivot uniquely identifies persistence pair $(\sigma_p^{(\mathtt{low}[D]_{\bullet j})}, \sigma_{p+1}^{(j)})$. Since $\sigma_p^{(\mathtt{low}[D]_{\bullet j})}$ is a creator then $\sigma_p^{(\mathtt{low}[D]_{\bullet j})}$ is not a destructor so $\mathtt{piv}_{\mathtt{low}[D]_{\bullet j}}$ does not exist. That is, column $\mathtt{low}[D]_{\bullet j}$ is pivotless. Set column $\mathtt{low}[D]_{\bullet j}$ to zero. The process of setting a column identified as pivotless to zero is called clearing.

In the Standard algorithm a column is not identified as eligible for clearing until after it has been processed since $\mathtt{low}[D]_{\bullet j} < j$ and we process by ascending column index. Therefore, we cannot leverage clearing to reduce processing. In fact, clearing would be unnecessary and increase processing time. Clearing is optimal when a nonzero column is identified as pivotless prior to its processing.

In the next chapter we introduce an algorithm amenable to optimization by clearing.

We give an informal algebraic proof for clearing:

Suppose we identify persistence pair $(\sigma_p^{(\mathtt{low}[R]_{\bullet j})}, \sigma_{p+1}^{(j)})$.

Note that $\mathtt{im}\delta_{p+1}^r \subset \mathtt{ker}\delta_p^r$ implies $\delta_p^r \circ \delta_{p+1}^r = 0$ and so $[D]^2 = 0$. Thus $[D][R] = 0$. In particular, the p-boundary of column $[R]_{\bullet j}$ is a zero column vector: $[D][R]_{\bullet j} = 0$.

By Standard algorithm processing, column $\mathtt{low}[R]_{\bullet j}$ of $[R]$ is a linear combination of column $\mathtt{low}[R]_{\bullet j}$ of $[D]$ and columns of $[R]$ of lower index than $\mathtt{low}[R]_{\bullet j}$. The columns of $[R]$ of lower index than $\mathtt{low}[R]_{\bullet j}$ are each linear combinations of columns of $[D]$ of lower index than $\mathtt{low}[R]_{\bullet j}$. Since column $[R]_{\bullet j}$ has a pivot at row $\mathtt{low}[R]_{\bullet j}$, then $[D][R]_{\bullet j} = 0$ implies that column $\mathtt{low}[R]_{\bullet j}$ of $[D]$ is a linear combination of columns of $[D]$ of lower index than $\mathtt{low}[R]_{\bullet j}$, so column $\mathtt{low}[R]_{\bullet j}$ of $[R]$ is a linear combination of columns of $[D]$ of lower index than $\mathtt{low}[R]_{\bullet j}$. This implies that column $\mathtt{low}[R]_{\bullet j}$ of $[R]$ is zero since columns in $[R]$ are linearly independent.

8. The Twist Algorithm

The Twist reduction algorithm processes columns of $[D]$ by descending dimension of corresponding simplices σ_p. For each dimension p, columns are processed by the Standard algorithm with clearing.

Suppose we are implementing Twist and identify persistence pair $(\sigma_p^{(\texttt{low}[R]_{\bullet j})}, \sigma_{p+1}^{(j)})$. Column $\texttt{low}[R]_{\bullet j}$ has not yet been processed since $p < p{+}1$. Hence, although clearing is inefficient in the Standard algorithm, it is generally optimal in Twist.

Observe that column $\texttt{low}[R]_{\bullet j}$ is zero when $p = 0$ so clearing is unnecessary and inefficient.

9. Cohomology

The filtration coboundary matrix $[D]_\perp$ is the anti-transpose of $[D]$. Rather than construct $[D]$ and reduce to $[R]$, a persistent homology program may determine persistence pairs by constructing $[D]_\perp$ and reducing to $[R]_\perp$. Persistence pair $(\sigma_p^{(i)}, \sigma_{p+1}^{(j)})$ of $[R]$ uniquely corresponds to persistence pair $(\sigma_p^{(s+1-j+1)}, \sigma_{p+1}^{(s+1-i+1)})$ of $[R]_\perp$.

For every homology reduction algorithm there exists a cohomology analogue applied to $[D]_\perp$. Moreover, a process called cohomology dualization translates cohomology analogue algorithms into homology algorithms. Similarly, homology dualization translates homology algorithms into cohomology algorithms. Therefore, rather than convert $[D]$ into $[D]_\perp$ (computationally expensive), a persistent homology program can construct $[D]$ then apply homology or dualized cohomology algorithms. Similarly, rather than convert $[D]_\perp$ into $[D]$, a persistent homology program can construct $[D]_\perp$ then apply cohomology or dualized homology algorithms.

Observe that in the homology Twist algorithm clearing is not an option for the highest dimension $p_{max} = \texttt{max}\{p : \sigma_p \in K_s\}$ since we process columns of $[D]$ by descending dimension p. Depending on the distribution of simplex dimensions $p \in \{0, \ldots, s\}$, it may be optimal to process columns by ascending dimension p. Analogous to the homology Twist algorithm is the cohomology Twist algorithm that processes $[D]_\perp$ by ascending dimension p. Rather than construct $[D]$ and dualize the cohomology Twist algorithm, construct $[D]_\perp$ and use the cohomology Twist algorithm directly. This illustrates that the optimal choice between $[D]$ and $[D]_\perp$ is determined by the program input arguments.

This paper is completed.

REFERENCES

[1] James R. Munkres, *Elements of Algebraic Topology*, Addison–Wesley, Menlo Park, 1984.
[2] James R. Munkres, *Topology*, Second Edition, Prentice Hall, 2000.
[3] Allen Hatcher, *Algebraic Topology*, Cambridge University Press, Cambridge, 2002.
[4] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian, *Topological Persistence and Simplification*, Discrete & Computational Geometry 28 (2002), 511-533.
[5] Afra Zomorodian and Gunnar Carlsson, *Computing Persistent Homology*, Proceedings of the Twentieth Annual Symposium on Computational Geometry (2004), 347-356.
[6] James McCaffrey, *Generating the mth lexicographical element of a mathematical combination*, MSDN Library 7, no. 07 (2004).
[7] Herbert Edelsbrunner and John Harer, *Persistent Homology-a Survey*, Contemporary mathematics 453, no. 26 (2008), 257-282.
[8] Robert Ghrist, *Barcodes: the Persistent Topology of Data*, Bulletin of the American Mathematical Society 45, no. 1 (2008), 61-75.
[9] Gunnar Carlsson, *Topology and Data*, Bulletin of the American Mathematical Society 46, no. 2 (2009), 255-308.
[10] Paul Bendich and John Harer, *Persistent Intersection Homology*, Foundations of Computational Mathematics 11, no. 3 (2011), 305-336.
[11] Chao Chen and Michael Kerber, *Persistent Homology Computation with a Twist*, Proceedings 27th European Workshop on Computational Geometry, vol. 11 (2011), 197-200.
[12] Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson, *Dualities in Persistent (Co) Homology*, Inverse Problems 27, no. 12, 124003 (2011).
[13] Herbert Edelsbrunner and Dmitriy Morozov, *Persistent Homology*, Handbook of Discrete and Computational Geometry, Chapman and Hall/CRC (2017), 637-661.
[14] Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner, *PHAT–Persistent Homology Algorithms Toolbox*, Journal of Symbolic Computation 78 (2017), 76-90.
[15] Nina Otter, Mason A. Porter, Ulrike Tillmann, Peter Grindrod, and Heather A. Harrington, *A Roadmap for the Computation of Persistent Homology*, EPJ Data Science 6 (2017), 1-38.
[16] Eashwar V. Somasundaram, Shael E. Brown, Adam Litzler, Jacob G. Scott, and Raoul R. Wadhwa. *Benchmarking R Packages for Calculation of Persistent Homology*, The R Journal 13, no. 1 (2021), 184.
[17] Ulrich Bauer, *Ripser: efficient computation of Vietoris–Rips persistence barcodes*, Journal of Applied and Computational Topology 5, no. 3 (2021), 391-423.
[18] Tamal Krishna Dey and Yusu Wang, *Computational Topology for Data Analysis*, Cambridge University Press, 2022.
[19] Asilata Bapat, Robyn Brooks, Celia Hacker, Claudia Landi, and Barbara I. Mahler, *Morse-based Fibering of the Persistence Rank Invariant*, Research in Computational Topology 2 (2022), Cham: Springer International Publishing, 27-62.
[20] Vidit Nanda, *Computational and Algebraic Topology* course notes, https://people.maths.ox.ac.uk/nanda/cat/TDANotes.pdf.