Assignment: paper summary / review

Motivation: PH on large complexes is expensive
How to compute fast.

Today: Optimizations to PH

$\begin{cases} \text{Clearing} \\ \text{Cohomology algs} \\ \text{Compression} \end{cases}$

Recall Reduction Alg:

$$0 \xleftarrow{\partial_0} C_0 \xleftarrow{\partial_1} C_1 \xleftarrow{\partial_2}$$

$$R_k = \partial_k U_k$$

Alg:

```
input: $\partial \in \mathbb{F}^{m \times n}$
return: $R \in \mathbb{F}^{m \times n}$, $U \in \mathbb{F}^{n \times n}$
initialize $R = \partial$, $U = I$
for $j = 1 .. n$:          look left
    while $\exists\, j_2 < j$ with $\text{pivot}(j_2) = \text{pivot}(j)$        ← last nonzero index
        $p = \text{pivot}(j_2)$
        $c = R[p, j] / R[p, j_2]$
        $R[:, j] -= c R[:, j_2]$
        $U[:, j] -= c U[:, j_2]$   ← // optional
return R, U
```

Way to read off PH:

- birth of homology class in dim k when we add column (simplex) to ∂_k which is zeroed out in R_k. call this simplex index i.
- death of this homology class when we add a column (simplex) to ∂_{k+1} which has i as a pivot in R_{k+1}.

Associated lemma: The addition of a simplex in a filtration either creates or destroys homology.

"Proof": the associated column in R_k is either 0 or has a pivot.

Optimization 1 (trivial): if we don't care about generators, i.e. only barcode, don't need to form matrix U

Optimization 2: clearing/killing Refs:
Chen & Kerber "Twist" 2011 ⇐
de Silva, Morozov, Vejdemo-Johansson Cohomology alg 2011

Observation: if column j of R_{k+1} has pivot i, then column i of R_k will be zero

$\Rightarrow$ if we don't care about U, just set $R_k[:, i] = 0$ without doing any work.

In order to apply this optimization, should process boundary matrices in reverse order:

1) Reduce R_{k+1} from ∂_{k+1}
2) identify pivots in R_{k+1}
3) "clear" or "kill" columns of ∂_k
4) reduce R_k from ∂_k $\Leftarrow$

Cohomology Alg:

First, what is cohomology? A big subject in mathematics. We're going to do the minimum.

Essentially, take duals of vector spaces in chain complexes to get cochain complexes.

$C^k = \underline{Hom}(C_k ; \mathbb{F})$ $x \in C_k$, $x^* \in C^k$: $x^* : x \to 1$

$\underset{\to}{\ }$

$Hom(\cdot ; \mathbb{F})$ is what is called a <u>contravariant functor</u>, meaning it reverses arrows.

objects $\to$ objects
maps $\to$ maps

$$\underline{V_0} \xrightarrow{A} V_1 \qquad \underset{\underset{V_1}{\in}}{y} = A \underset{\underset{V_0}{\in}}{x}$$

$\Downarrow$ Hom

$$\underline{V_0^*} \xleftarrow{A^*} V_1^* \qquad A^* : y^* \mapsto A^T y^*$$

$$(y \mapsto 1) \mapsto A \circ (y \mapsto 1)$$

Hom$(\cdot\,; \mathbb{F})$ turns a chain complex

$$0 \xleftarrow{\partial_0} C_0 \xleftarrow{\partial_1} C_1 \xleftarrow{\partial_2} \cdots$$

into a cochain complex

$$0 \xrightarrow{\delta^0} C^0 \xrightarrow{\delta^1} C^1 \xrightarrow{\delta^2} \cdots \qquad \delta^k = \partial_k^T$$

$$\delta^{k+1} \circ \delta^k = 0$$

define $H^*(C^*) = \ker \delta^{k+1} / \operatorname{img} \delta^k$ $\Big\}$ $(\partial_k \circ \partial_{k+1})^T = 0$

Over fields, $H^k(\underset{t}{X}) \cong H_k(X)$ (not generally true if not a field)

Lemma: Let $V_0 \xleftrightarrow{A_0} V_1 \xleftrightarrow{A_1} V_2 \leftrightarrow$

be a type-A quiver rep. Then the dual quiver rep w/ reversed arrows

$$V_0^* \xleftrightarrow{A_0^T} V_0^* \xleftrightarrow{A^T} V_1^*$$

has the same barcode.

Proof: we'll use a barcode factorization, and look at V_k.

$$\begin{tikzcd} \cdots & V_k \arrow[l, "A_{k-1}"'] & \cdots \arrow[l, "A_k"'] \end{tikzcd} = \begin{tikzcd} \cdots & V_k \arrow[l, "B_{k-1}E_{k-1}B_k^{-1}"'] & \cdots \arrow[l, "B_kE_kB_{k-1}^{-1}"'] \end{tikzcd}$$

$$\begin{tikzcd} \cdots \arrow[r, "A_{k-1}^T"] & V_k^* \arrow[r, "A_k^T"] & \cdots \end{tikzcd} = \begin{tikzcd} \cdots \arrow[r, "B_k^{-T}E_{k-1}^TB_{k-1}^T"] & V_k \arrow[r, "B_{k-1}^{-T}E_k^TB_k^T"] & \cdots \end{tikzcd}$$

change of basis via B_k on V_k gives

$$\begin{tikzcd} \cdots & V_k \arrow[l, "E_{k-1}"'] & \cdots \arrow[l, "E_k"'] \end{tikzcd}$$ easy to read off barcode.

Change of basis via B_k^{-T} on V_k^* gives

$$\begin{tikzcd} \cdots \arrow[r, "E_{k-1}^T"] & V_k^* \arrow[r, "E_k^T"] & \cdots \end{tikzcd}$$

Recall that E_k matrices have at most one non-zero in each row & column.

$\Rightarrow E_k^T$ has same property.

E_k^T identifies basis (co) vectors V_k^* w/ basis (co) vectors in V_{k+1}^*. We can check the barcode doesn't change b/c identification of indices doesn't change.

$\rightarrow$ type A quiver rep & dual have same barcode $\square$

Prop: $H^k(X) \cong H_k(X)$

Note: cochain cpx is dual type-A quiver rep of chain cpx. $\Rightarrow$ has same barcode.

From HW 2: indecomposables of chain cpx are $I[k,k+1]$ or $I[k,k]$

$\dim H_k = \#\{I[k,k]\}$

<u>basis</u> vector for $I[k,k]$ is a rep for Hom class.

similarly, $\dim H^k = \#\{I[k,k]\}$

Since barcodes are identical, dimensions same $\square$.

$\Rightarrow$ Homology & cohomology are same for a fixed space. $\Rightarrow$ can either compute $R_k = \partial_k U_k$ or

$\tilde{R}_k = \delta_k \tilde{U}_k$

and extract information.

What abt persistent versions of homology & cohomology.

Interpretation w.r.t. a filtration a bit different.

b/c cohomology is contravariant

$$\begin{tikzcd} X_0 \arrow[r] & X_1 \arrow[r] & X_2 \arrow[r] & \cdots \end{tikzcd}$$

$$\begin{tikzcd} H_k(X_0) \arrow[r] & H_k(X_1) \arrow[r] & H_k(X_2) \arrow[r] & \cdots \end{tikzcd}$$

$$\begin{tikzcd} H^k(X_0) & H^k(X_1) \arrow[l] & H^k(X_2) \arrow[l] & \cdots \arrow[l] \end{tikzcd}$$

$\uparrow$ death $\qquad\qquad\qquad\qquad$ $\uparrow$ birth

Cohomology alg: use matrices δ where basis elts are in reverse filtration order.

Can run reduction alg on this matrix.

interpretation: cohomology classes born at larger filtration values and die at smaller filtration values. $[b,d]$

Persistent homology barcode + persistent cohomology barcode are identical.

Additional details. Two $\overset{\text{more}}{\vee}$ ways to implement reduction alg. Correspond to forward/backward looking factorization algs. Standard reduction "pH col" is backward looking.

"pH row" is forward looking.

```
input: $\partial_k$   output $R_k, U_k$
initialize $R_k = \partial_k$, $U_k = I$
for $i = n \dots 1$
    inds = $\{ j \mid \text{pivot}(j) = i \}$
    $p$ = inds[0]  / first j where i appears as pivot
    for $j \in$ inds[1...]
        $c = R[i,j] / R[i,p]$
        $R[:,j] -= c R[:,p]$
        $U[:,j] -= c U[:,p]$
```

output of this alg is identical to standard alg. All we have done is reorder operations

---

called "row" alg because we eliminate all pivots in a row at once.

Original version of cohomology alg. used "row" alg w/ a clearing optimization observed large speedups.

Note that clearing in cohomology is a bit different. we process δ_k before δ_{k+1} to identify columns of δ_{k+1} which will be cleared. (dims reversed wrt. homology clearing)

Compression optimization:

Let's go back to homology:

$$0 \xleftarrow{\partial_0} C_0 \xleftarrow{\partial_1} C_1 \xleftarrow{\partial_2} \cdots$$

Clearing: identify columns of R_k which will be zero by analyzing pivots of R_{k+1}

Compression: remove rows of ∂_{k+1} by analyzing cols in R_k.

Observation: if column i is <u>not</u> zero in R_k, row i will <u>not</u> contain a pivot in R_{k+1}.

(b/c pivot in R_{k+1} = zero col in R_k)

$\rightarrow$ if we process $R_k = \partial_k U_k$ <u>before</u> looking at ∂_{k+1}, we can safely remove rows from ∂_{k+1} because they will never be used in reduction. (and are not needed for barcode)

Note: can still form U_k using this observation

Note: can apply a version of this to cohomology alg as well.

<u>Compression</u> opts Bauer, Kerber, Reininghaus 2014

Two opts:

Clearing (process dimensions in reverse order)

Compression (process dimension in order)

Is there a way to combine them?

Bauer et al 2014: yes. Chunk alg.

idea is to break up computation into chunks use clearing while working locally, and can use clearing/compression when combining results from different chunks.

Notes: pivots found locally are valid.

However, zeroing out a column in submatrix doesn't guarantee it will be zero when communicating w/ other blocks.

→ clear & compress local pairs, then finish reduction on "global columns"

Note 2: can parallelize local reductions originally implemented in PHAT

(wrapped by scikit TDA)

Empirically, working w/ cohomology calcs & opts. is faster than homology on Rips complexes. For other complexes, this may not be the case.

Ripser (wrapped by scikit TDA)
Bauer 2017. implements cohomology calcs
w/ clear/compress

Ripser ++ 2020 uses GPU for more parallelism
Even faster.
