Ripser

or: the unexpected efficiency of persistent cohomology

Ulrich Bauer

TUM

July 25, 2016

ATMCS7, Torino

Design goals

Goals for previous projects:

- PHAT: fast persistence computation
  (boundary matrix reduction only)
- DIPHA: distributed persistence computation

Design goals

Goals for previous projects:

- PHAT: fast persistence computation (boundary matrix reduction only)
- DIPHA: distributed persistence computation

Goals for Ripser:

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

Design goals

Goals for previous projects:

- PHAT: fast persistence computation (boundary matrix reduction only)
- DIPHA: distributed persistence computation

Goals for Ripser:

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

Features:

- time- and memory-efficient
- less than 1000 lines of code in a single C++ file
- support for coefficients in prime finite fields
- no external dependencies

The past

Matrix reduction

Setting:

- finite metric space, n points
- persistent homology for k-skeleta of *Vietoris–Rips filtration*
- homology H_d in dimensions $0 \leq d < k$

Notation:

- D: boundary matrix of filtration
- R_i: ith column of R

Matrix reduction

Setting:

- finite metric space, n points
- persistent homology for k-skeleta of *Vietoris–Rips filtration*
- homology H_d in dimensions $0 \leq d < k$

Notation:

- D: boundary matrix of filtration
- R_i: ith column of R

Algorithm:

- $R = D$, $V = I$
- while $\exists i < j$ with $\operatorname{pivot} R_i = \operatorname{pivot} R_j$
  - add R_i to R_j, add V_i to V_j

Matrix reduction

Setting:

- finite metric space, n points
- persistent homology for k-skeleta of *Vietoris–Rips filtration*
- homology H_d in dimensions $0 \leq d < k$

Notation:

- D: boundary matrix of filtration
- R_i: ith column of R

Algorithm:

- $R = D$, $V = I$
- while $\exists i < j$ with $\operatorname{pivot} R_i = \operatorname{pivot} R_j$
    - add R_i to R_j, add V_i to V_j

Result:

- $R = D \cdot V$ is reduced (unique pivots)
- V is full rank upper triangular

Lessons from PHAT

Two optimizations speed up computation considerably:

- Clearing positive columns
  [Chen, Kerber 2011]
- Persistent cohomology
  [de Silva, Morozov, Vejdemo-Johannson 2011]

But only when *both* are used in conjuction!

Compatible basis cycles

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

$$P = \{i : R_i = 0\} \qquad \textit{positive} \text{ indices,}$$

$$N = \{j : R_j \neq 0\} \qquad \textit{negative} \text{ indices,}$$

$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential} \text{ indices.}$$

Then

Compatible basis cycles

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

$$P = \{i : R_i = 0\} \qquad \textit{positive} \text{ indices,}$$
$$N = \{j : R_j \neq 0\} \qquad \textit{negative} \text{ indices,}$$
$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential} \text{ indices.}$$

Then

$$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\} \qquad \text{is a basis of } Z_*,$$

Compatible basis cycles

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

$$P = \{i : R_i = 0\} \qquad \textit{positive} \text{ indices,}$$
$$N = \{j : R_j \neq 0\} \qquad \textit{negative} \text{ indices,}$$
$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential} \text{ indices.}$$

Then

$$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\} \qquad \text{is a basis of } Z_*,$$
$$\Sigma_B = \{R_j \mid j \in N\} \qquad \text{is a basis of } B_*,$$

Compatible basis cycles

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

$$P = \{i : R_i = 0\} \qquad \textit{positive} \text{ indices,}$$
$$N = \{j : R_j \neq 0\} \qquad \textit{negative} \text{ indices,}$$
$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential} \text{ indices.}$$

Then

$$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\} \qquad \text{is a basis of } Z_*,$$
$$\Sigma_B = \{R_j \mid j \in N\} \qquad \text{is a basis of } B_*,$$
$$\Sigma_Z = \Sigma_B \cup \{V_i \mid i \in E\} \qquad \text{is } \textit{another} \text{ basis of } Z_*.$$

Compatible basis cycles

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

$$P = \{i : R_i = 0\} \qquad \textit{positive} \text{ indices,}$$
$$N = \{j : R_j \neq 0\} \qquad \textit{negative} \text{ indices,}$$
$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential} \text{ indices.}$$

Then

$$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\} \qquad \text{is a basis of } Z_*,$$
$$\Sigma_B = \{R_j \mid j \in N\} \qquad \text{is a basis of } B_*,$$
$$\Sigma_Z = \Sigma_B \cup \{V_i \mid i \in E\} \qquad \text{is } \textit{another} \text{ basis of } Z_*.$$

Persistent homology is generated by the basis cycles Σ_Z.

Compatible basis cycles

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

$$P = \{i : R_i = 0\} \qquad \textit{positive} \text{ indices,}$$
$$N = \{j : R_j \neq 0\} \qquad \textit{negative} \text{ indices,}$$
$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential} \text{ indices.}$$

Then

$$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\} \qquad \text{is a basis of } Z_*,$$
$$\Sigma_B = \{R_j \mid j \in N\} \qquad \text{is a basis of } B_*,$$
$$\Sigma_Z = \Sigma_B \cup \{V_i \mid i \in E\} \qquad \text{is } \textit{another} \text{ basis of } Z_*.$$

Persistent homology is generated by the basis cycles Σ_Z.

- Columns with non-essential positive indices never used!

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential positive indices
- Reduce boundary matrices of $\partial_d : C_d \to C_{d-1}$ in decreasing dimension $d = k \ldots 1$
- Whenever $i = \operatorname{pivot} R_j$
  - Set R_i to 0

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential positive indices
- Reduce boundary matrices of $\partial_d : C_d \to C_{d-1}$ in decreasing dimension $d = k \ldots 1$
- Whenever $i = \text{pivot}\, R_j$
  - Set R_i to 0
  - Set V_i to R_j

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential positive indices
- Reduce boundary matrices of $\partial_d : C_d \to C_{d-1}$ in decreasing dimension $d = k \ldots 1$
- Whenever $i = \text{pivot}\, R_j$
  - Set R_i to 0
  - Set V_i to R_j

Note:

- reducing *positive* columns typically harder than negative
- with clearing: need only reduce *essential* positive columns

Counting column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k} \binom{n}{d+1} = \sum_{d=1}^{k} \underbrace{\binom{n-1}{d}}_{\text{negative}} + \sum_{d=1}^{k} \underbrace{\binom{n-1}{d+1}}_{\text{positive}}$$

Counting column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k} \binom{n}{d+1} = \sum_{d=1}^{k} \underbrace{\binom{n-1}{d}}_{\text{negative}} + \sum_{d=1}^{k} \underbrace{\binom{n-1}{d+1}}_{\text{positive}}$$

$k = 3, n = 192$: $\quad 56\,050\,096 = 1\,161\,471 + 54\,888\,625$

Counting column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k} \binom{n}{d+1} = \sum_{d=1}^{k} \underbrace{\binom{n-1}{d}}_{\text{negative}} + \sum_{d=1}^{k} \underbrace{\binom{n-1}{d+1}}_{\text{positive}}$$

$k = 3, n = 192$: $\quad 56\,050\,096 = 1\,161\,471 + 54\,888\,625$

- using clearing:

$$\sum_{d=1}^{k} \underbrace{\binom{n-1}{d}}_{\text{negative}} + \underbrace{\binom{n-1}{k+1}}_{\text{essential}} = \sum_{d=1}^{k} \binom{n-1}{d+1}$$

Counting column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k} \binom{n}{d+1} = \sum_{d=1}^{k} \underbrace{\binom{n-1}{d}}_{\text{negative}} + \sum_{d=1}^{k} \underbrace{\binom{n-1}{d+1}}_{\text{positive}}$$

$k = 3, n = 192$: $\quad 56\,050\,096 = 1\,161\,471 + 54\,888\,625$

- using clearing:

$$\sum_{d=1}^{k} \underbrace{\binom{n-1}{d}}_{\text{negative}} + \underbrace{\binom{n-1}{k+1}}_{\text{essential}} = \sum_{d=1}^{k} \binom{n-1}{d+1}$$

$k = 3, n = 192$: $\quad 54\,888\,816 = 1\,161\,471 + 53\,727\,345$

Persistent cohomology

Idea [de Silva, Morozov, Vejdemo-Johannson 2011]:

- same barcodes
- reduce coboundary matrix in reversed filtration order
- observation: computation often much faster (why?)

Persistent cohomology

Idea [de Silva, Morozov, Vejdemo-Johannson 2011]:

- same barcodes
- reduce coboundary matrix in reversed filtration order
- observation: computation often much faster (why?)

Clearing for persistent cohomology:

- reduce in increasing dimension $d = 0, \ldots, k-1$
- negative becomes (dual) positive
- positive non-essential becomes (dual) negative
- essential stays (dual) essential

Counting cohomology column reductions

- standard matrix reduction:

$$\sum_{d=0}^{k-1} \binom{n}{d+1} = \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d}}_{\text{(dual) positive}} + \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d+1}}_{\text{(dual) negative}}$$

Counting cohomology column reductions

- standard matrix reduction:

$$\sum_{d=0}^{k-1} \binom{n}{d+1} = \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d}}_{\text{(dual) positive}} + \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d+1}}_{\text{(dual) negative}}$$

$k = 3, n = 192$: $\quad 1\,179\,808 = 18\,337 + 1\,161\,471$

Counting cohomology column reductions

- standard matrix reduction:

$$\sum_{d=0}^{k-1} \binom{n}{d+1} = \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d}}_{\text{(dual) positive}} + \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d+1}}_{\text{(dual) negative}}$$

$k = 3, n = 192$: $\quad 1\,179\,808 = 18\,337 + 1\,161\,471$

- using clearing:

$$\underbrace{\binom{n-1}{0}}_{\text{essential}} + \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d+1}}_{\text{(dual) negative}} = \sum_{d=0}^{k-1} \binom{n-1}{d}$$

Counting cohomology column reductions

- standard matrix reduction:

$$\sum_{d=0}^{k-1} \binom{n}{d+1} = \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d}}_{\text{(dual) positive}} + \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d+1}}_{\text{(dual) negative}}$$

$k = 3, n = 192$: $\quad 1\,179\,808 = 18\,337 + 1\,161\,471$

- using clearing:

$$\underbrace{\binom{n-1}{0}}_{\text{essential}} + \sum_{d=0}^{k-1} \underbrace{\binom{n-1}{d+1}}_{\text{(dual) negative}} = \sum_{d=0}^{k-1} \binom{n-1}{d}$$

$k = 3, n = 192$: $\quad 1\,161\,472 = 1 + 1\,161\,471$

The present

Ripser design principles

Don’t store what you can compute:

- filtration (from distance matrix)
- boundary matrix D (from n, d)
- reduced matrix R (from matrices D, V)
- reduction matrix V (from persistence pairs)

Ripser design principles

Don't store what you can compute:

- filtration (from distance matrix)
- boundary matrix D (from n, d)
- reduced matrix R (from matrices D, V)
- reduction matrix V (from persistence pairs)

Store only:

- persistence pairs
- negative column indices (sorted by filtration order)
- current column of R (in heap, comparison based)

Observations

For a typical input:

- V has very few off-diagonal entries
- most negative columns of D are already reduced from the beginning *(apparent persistence pairs)* and correspond to pairs of persistence 0

Observations

For a typical input:

- V has very few off-diagonal entries
- most negative columns of D are already reduced from the beginning *(apparent persistence pairs)* and correspond to pairs of persistence 0

Example: $k = 3, n = 192$:

- Only $191 + 53 + 601 = 845$ out of $1\,161\,471$ pairs are not apparent 0-persistence pairs

Conclusion

Can compute much larger instances than previous software

- H^2 persistence for data with 1681 points,
  in about 30 minutes using 20GB RAM
- Available at `http://git.io/ripser`
- Bring your own data set to the hands-on demonstration!