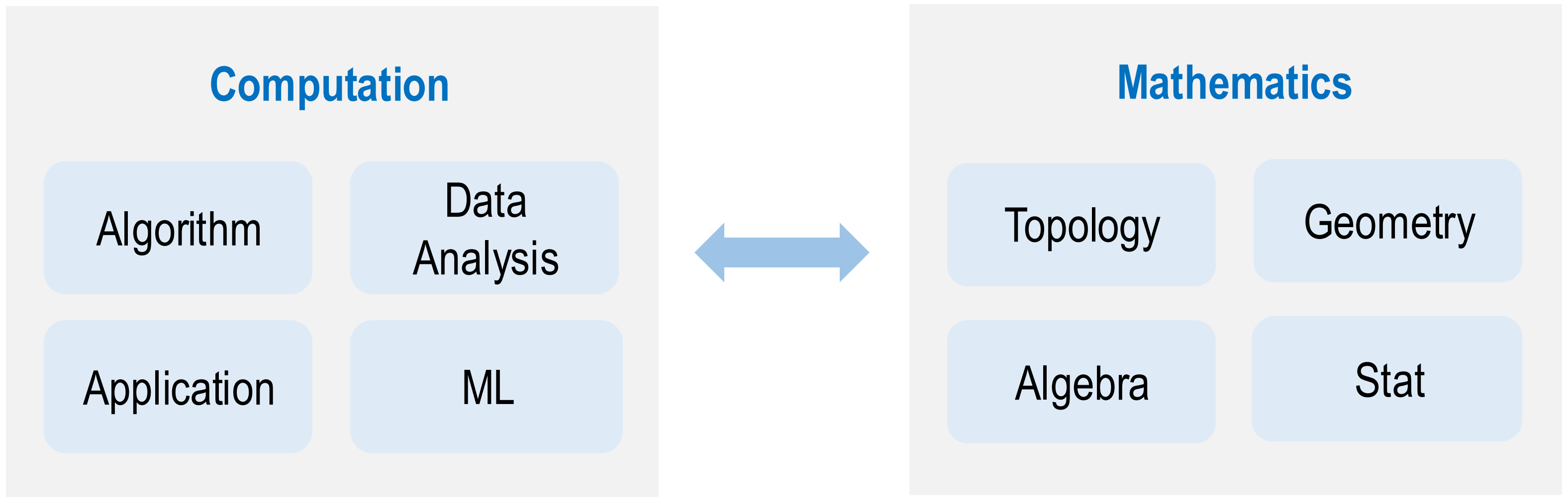
My research interests

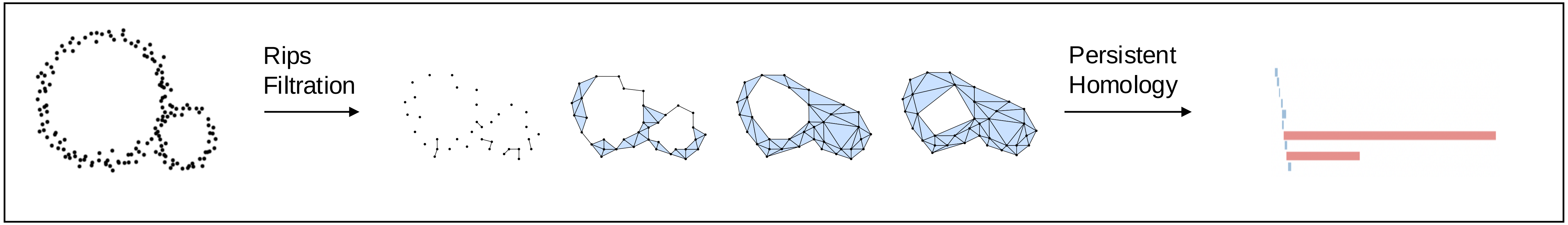
Tao Hou, CS Department

University of Oregon

Nov 2024

Research interest: Topological data analysis (TDA), topological machine learning, computational topology





Persistent homology

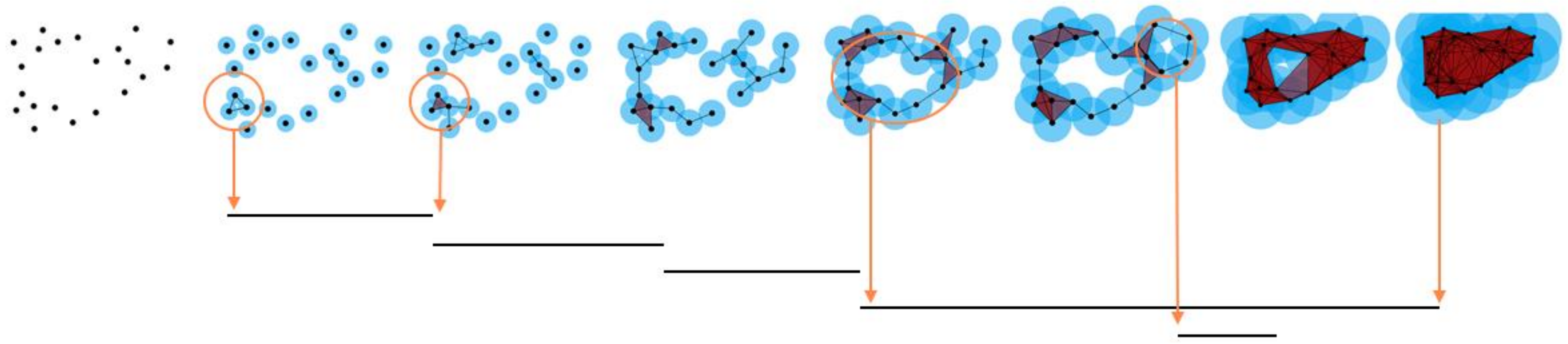
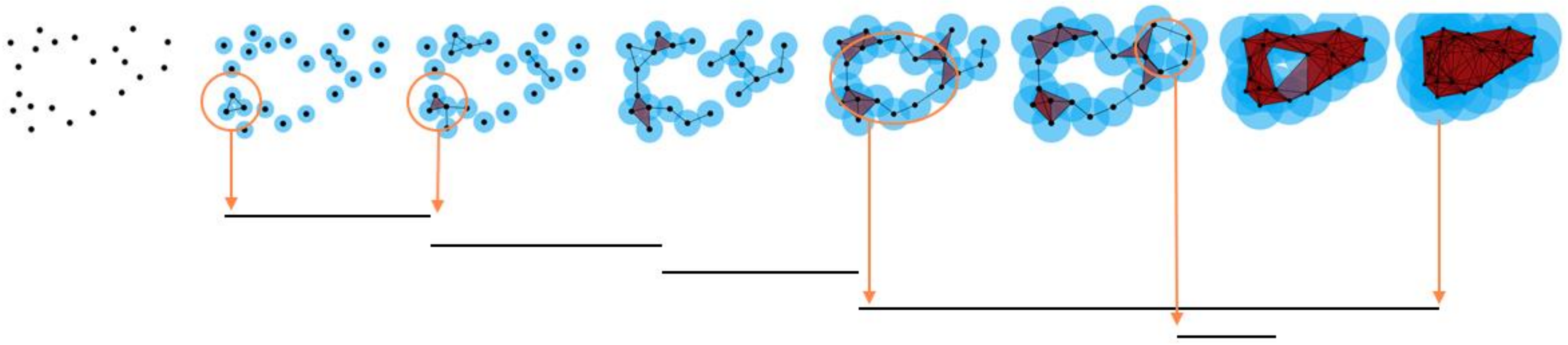


Figure source: https://graphsandnetworks.com/what-is-persistent-homology/

Persistent homology



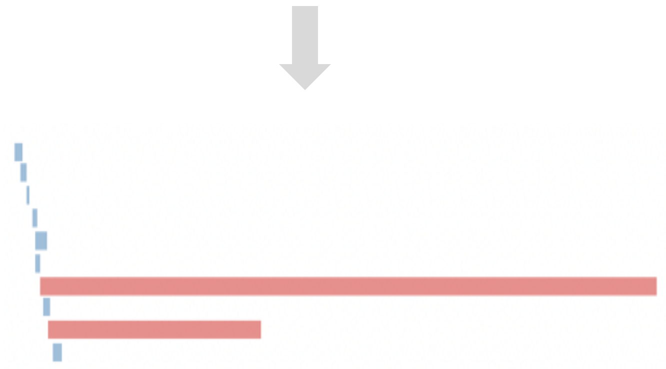
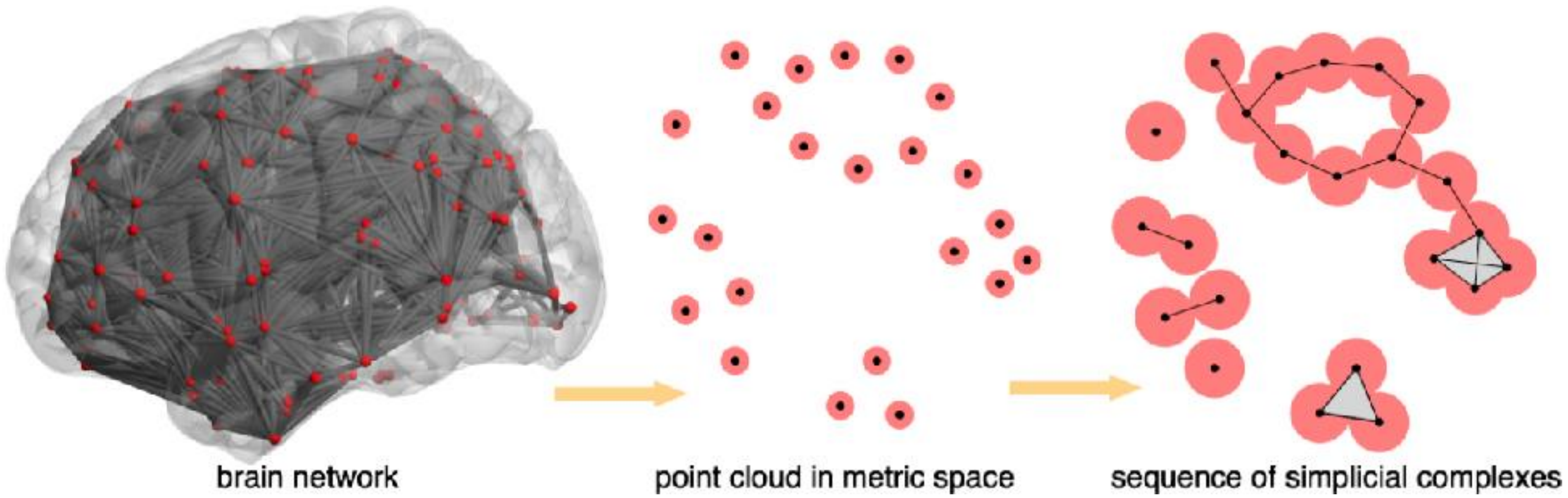


Figure source: https://graphsandnetworks.com/what-is-persistent-homology/

Application: Brain data



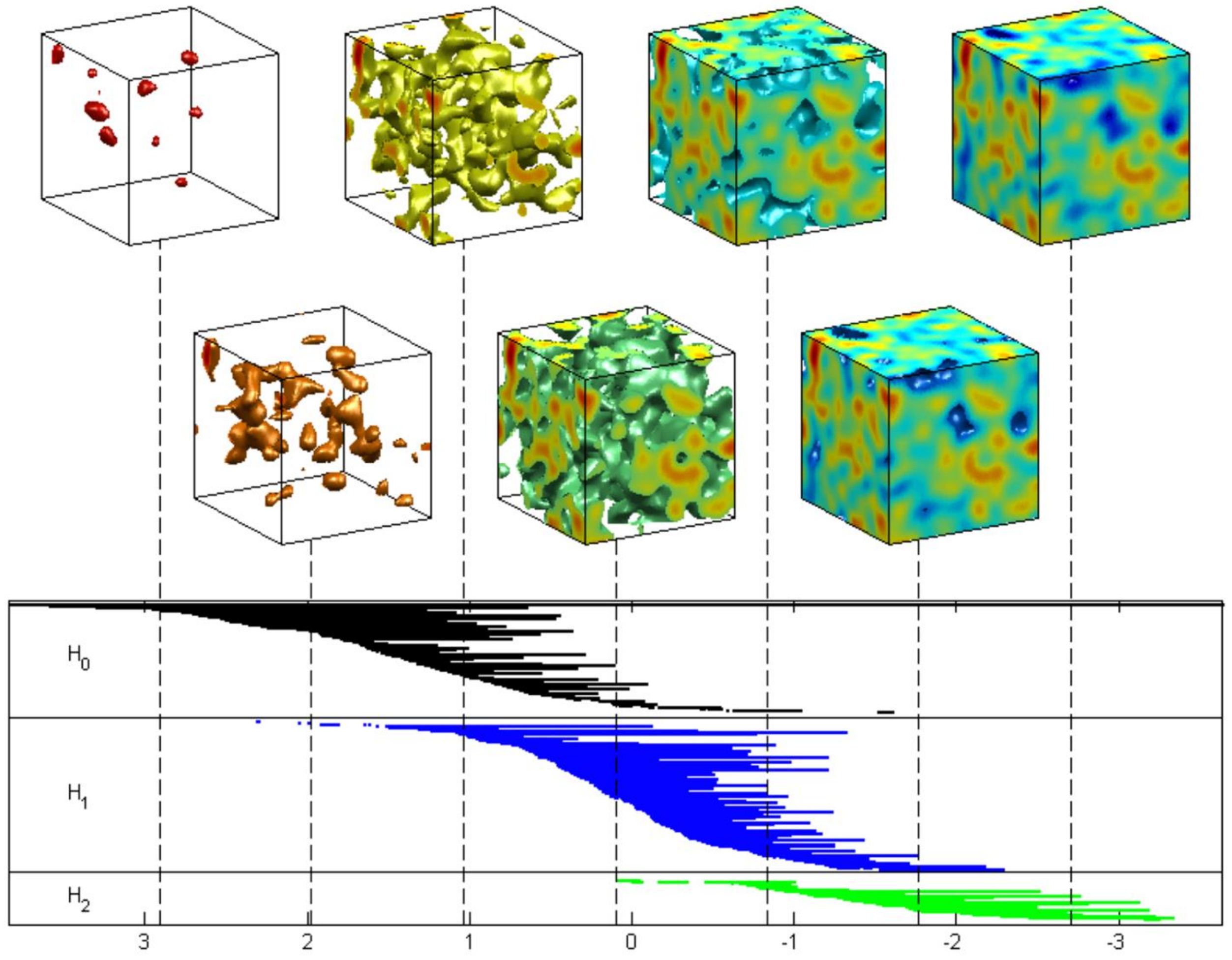
For detecting voids or tunnels (higher-dimensional holes in the brain)

Figure source: https://www.semanticscholar.org/paper/2-Application-%3A-Persistent-Homology-and-Brain-Data-Conkey/f7007652db07ebc39b57e297122e6c474e64abae (Adam Conkey)

On analyzing volume data

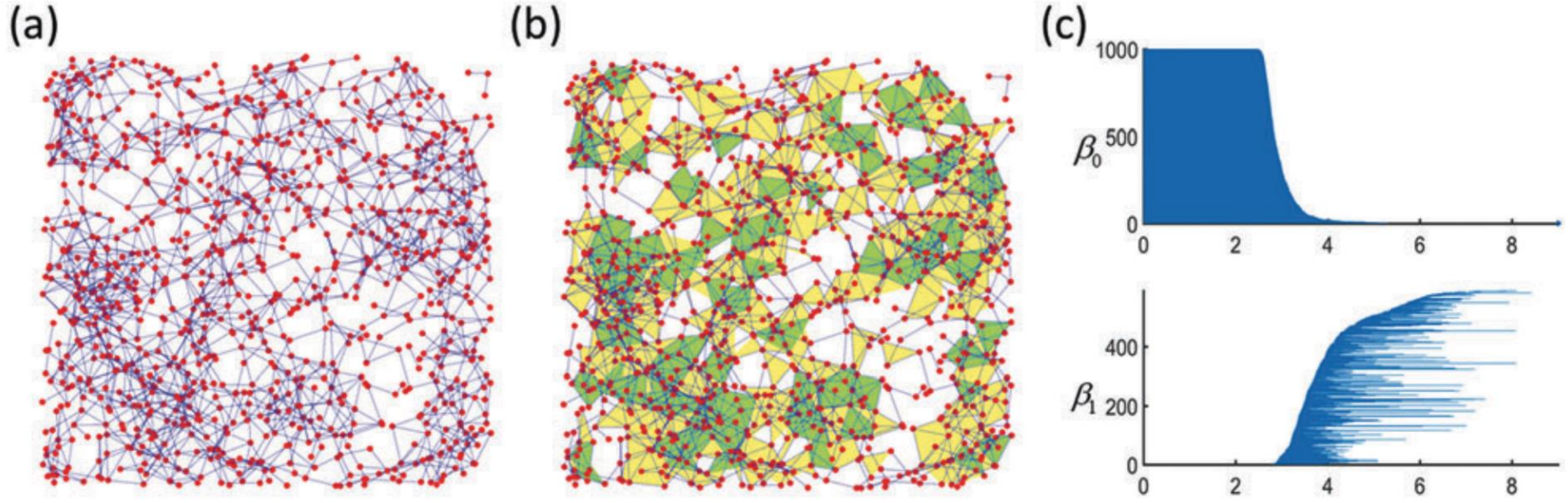
Volume data:

- Consisting of cubes, with each cube having a value
- Filter the cubes by adding them one by one based on ordering the values (e.g., adding cubes of smaller values first, then ones with bigger values)



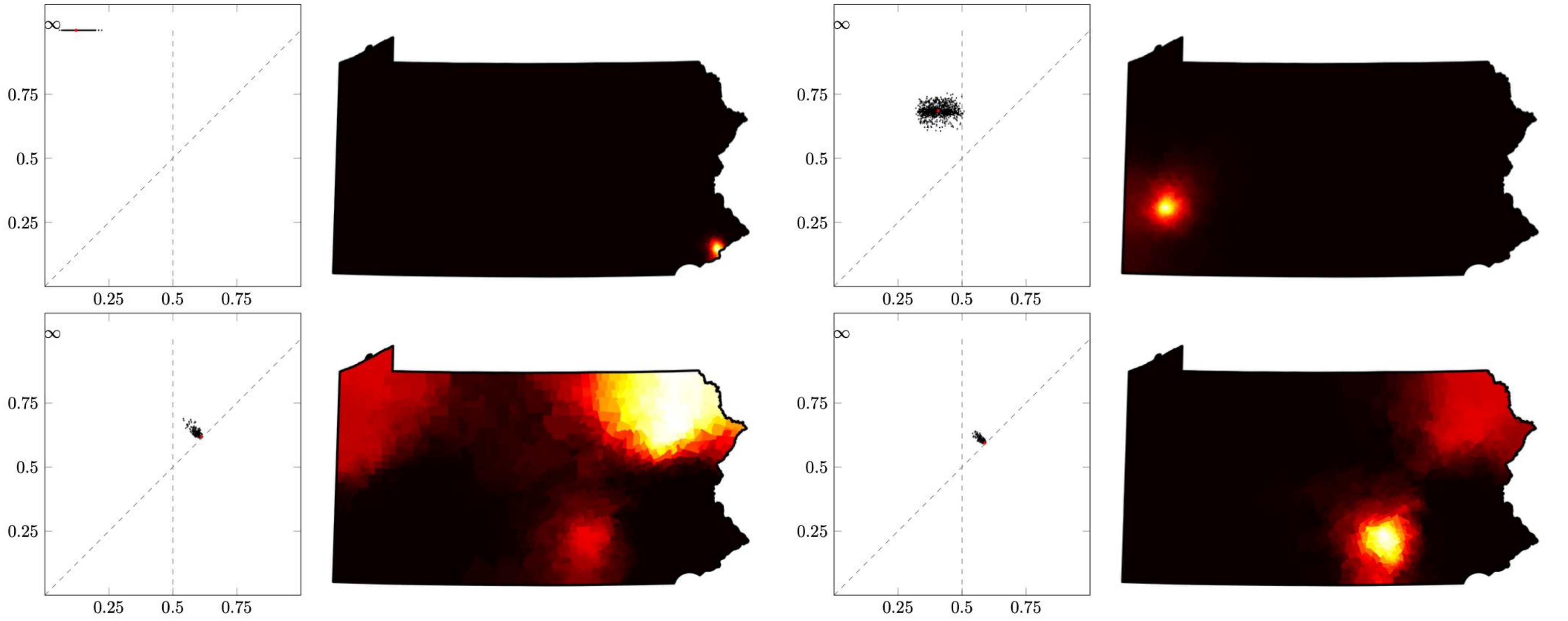
Adler, Robert J., Omer Bobrowski, Matthew S. Borman, Eliran Subag, and Shmuel Weinberger. "Persistent homology for random fields and complexes." In Borrowing strength: theory powering applications–a Festschrift for Lawrence D. Brown, vol. 6, pp. 124-144. Institute of Mathematical Statistics, 2010

Persistent homology analysis of a hydrogen-bonding network



Xia, Kelin. "Persistent homology analysis of ion aggregations and hydrogen-bonding networks." *Physical Chemistry Chemical Physics* 20.19 (2018): 13448-13460.

On gerrymandering

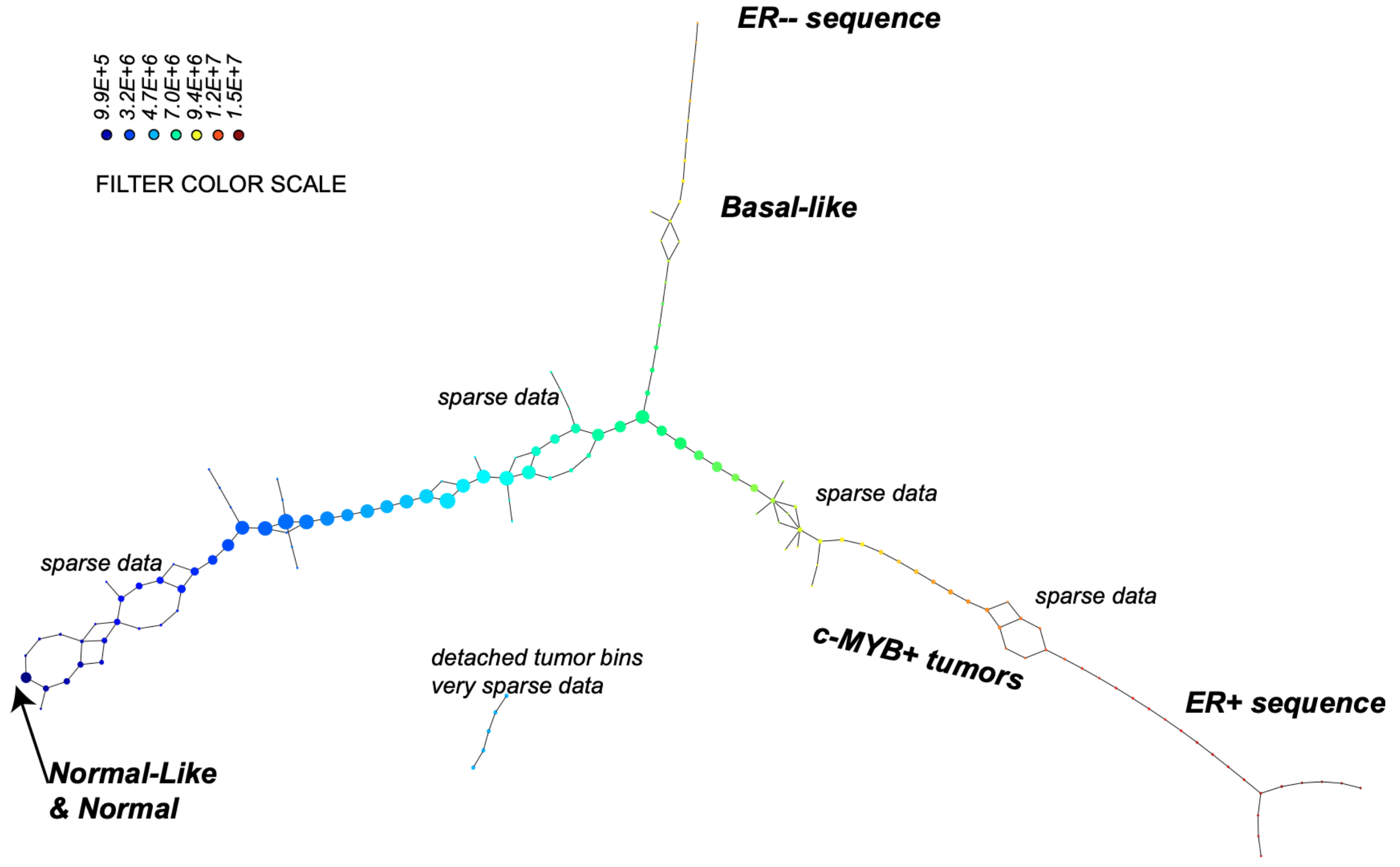


Geographical localization of Fŕechet features in Pennsylvania Congressional and Senate plans with respect to PRES16 voting

Duchin, Moon, Tom Needham, and Thomas Weighill. "The (homological) persistence of gerrymandering." *arXiv preprint arXiv:2007.02390* (2020

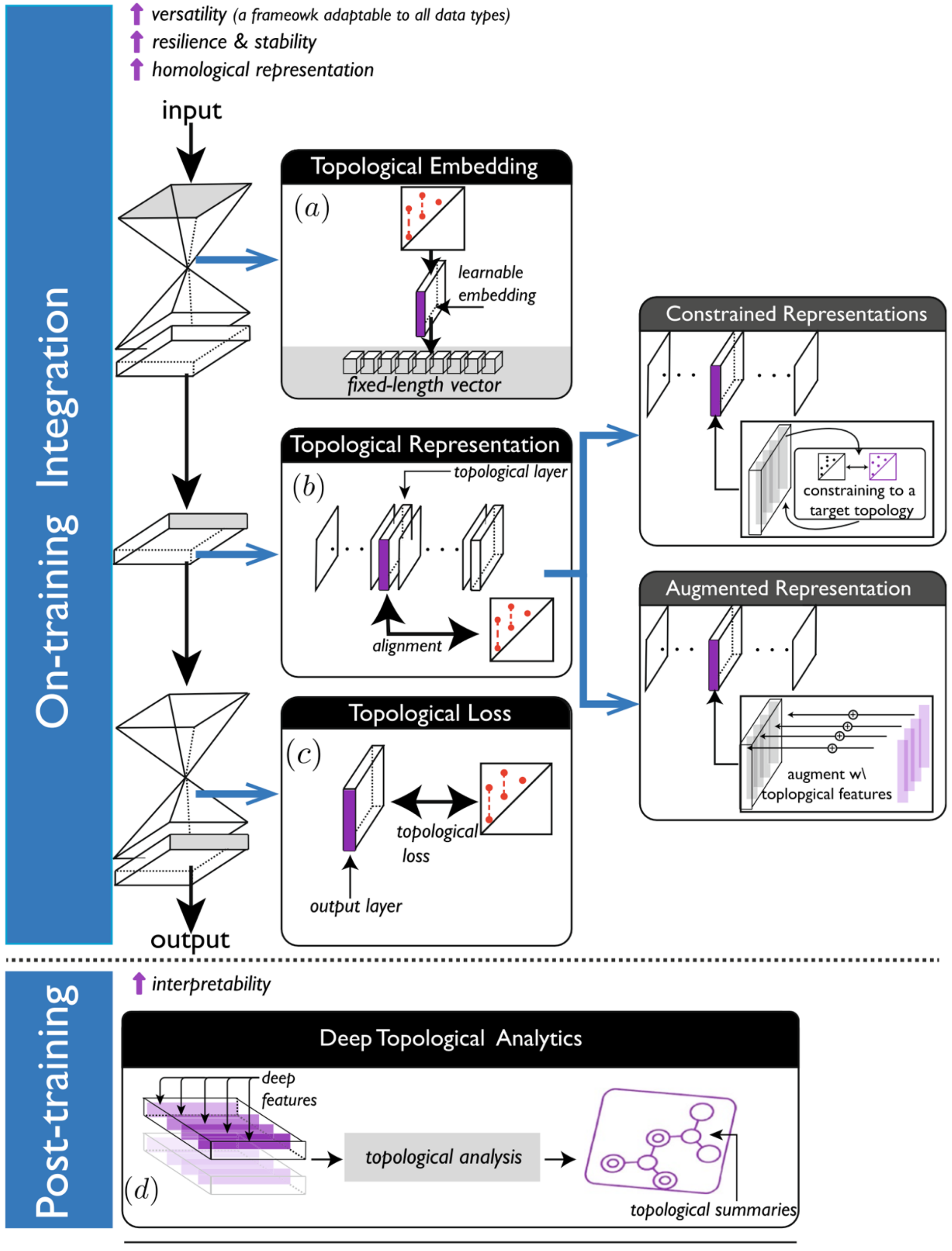
Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival

Uses mapper (another popular tool in TDA)



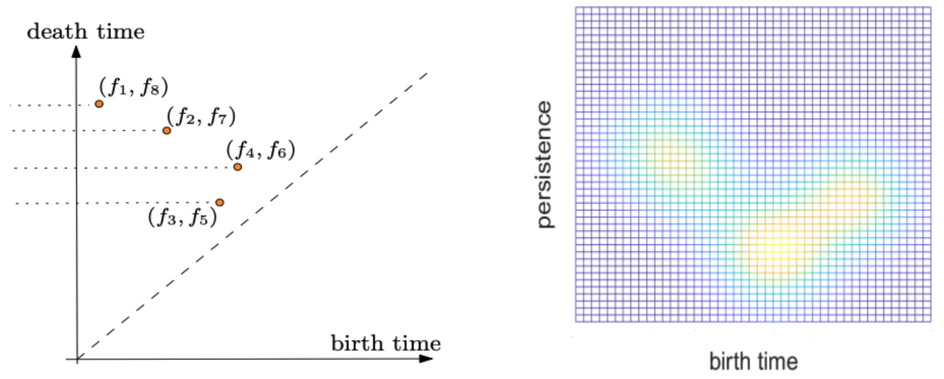
Nicolau, Monica, Arnold J. Levine, and Gunnar Carlsson. "Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival." *Proceedings of the National Academy of Sciences*

Integrating into DL at various stages

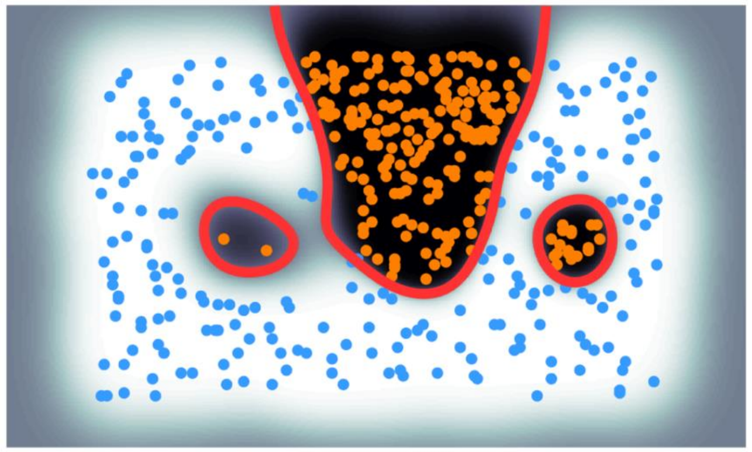


Zia, Ali, Abdelwahed Khamis, James Nichols, Usman Bashir Tayab, Zeeshan Hayder, Vivien Rolland, Eric Stone, and Lars Petersson. "Topological deep learning: a review of an emerging paradigm." *Artificial Intelligence Review* 57, no. 4 (2024): 77.

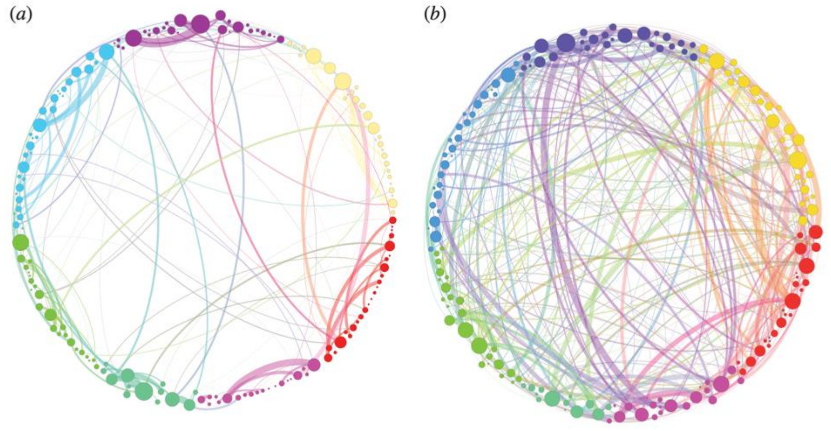
TDA: More Applications



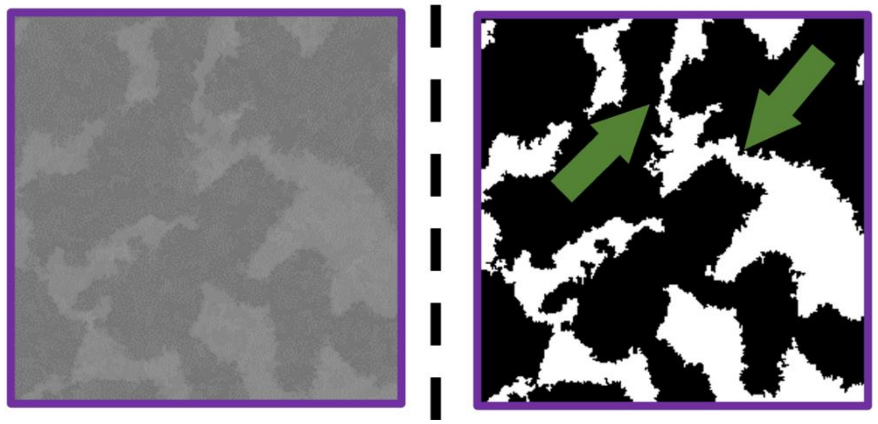
Features for ML [Zhao & Wang 19]



Topological regularizer for ML [Chen et al. 20]



Brain functional networks [Petri et al. 14]



Binarizing microstructures [Patel et al. 22]

What’s so special about persistent homology?

Theoretically justified: being stable under small perturbations (noise) of data

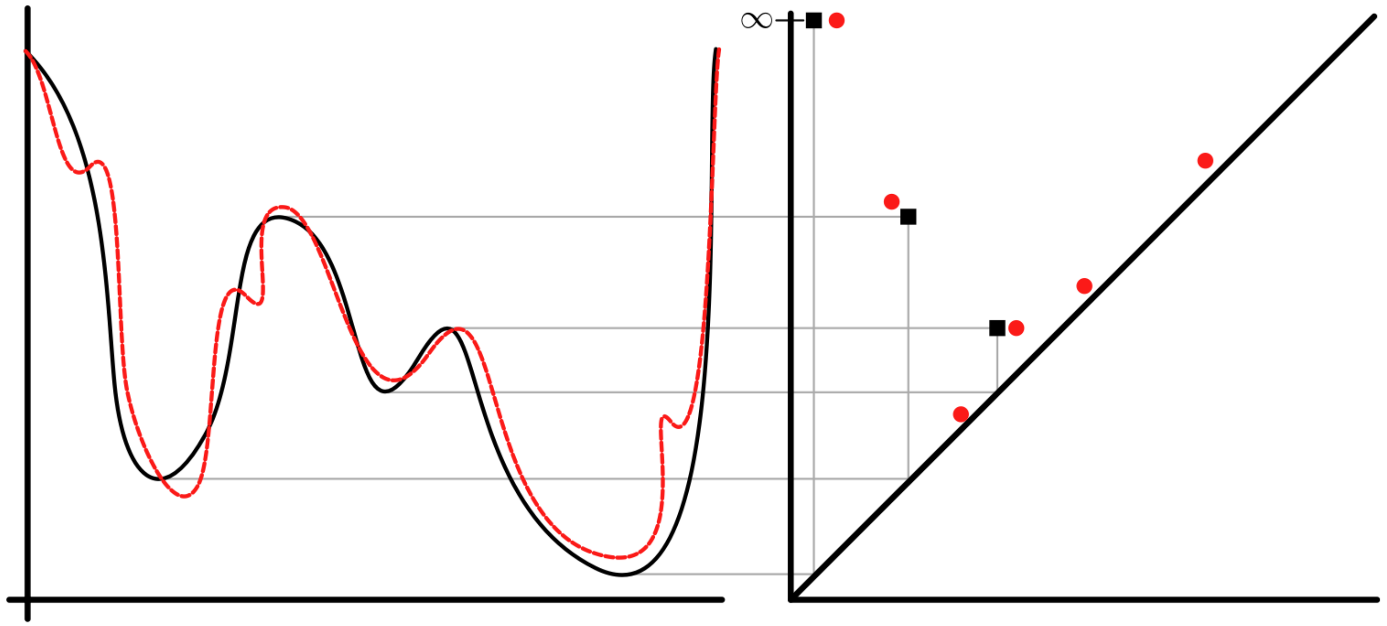


Figure from: Bendich, Paul, Sang Peter Chin, Jesse Clark, Jonathan Desena, John Harer, Elizabeth Munch, Andrew Newman et al. "Topological and statistical behavior classifiers for tracking applications." *IEEE Transactions on Aerospace and Electronic Systems* 52, no. 6 (2016): 2644-2661

What’s so special about persistent homology?

Theoretically justified: being stable under small perturbations (noise) of data

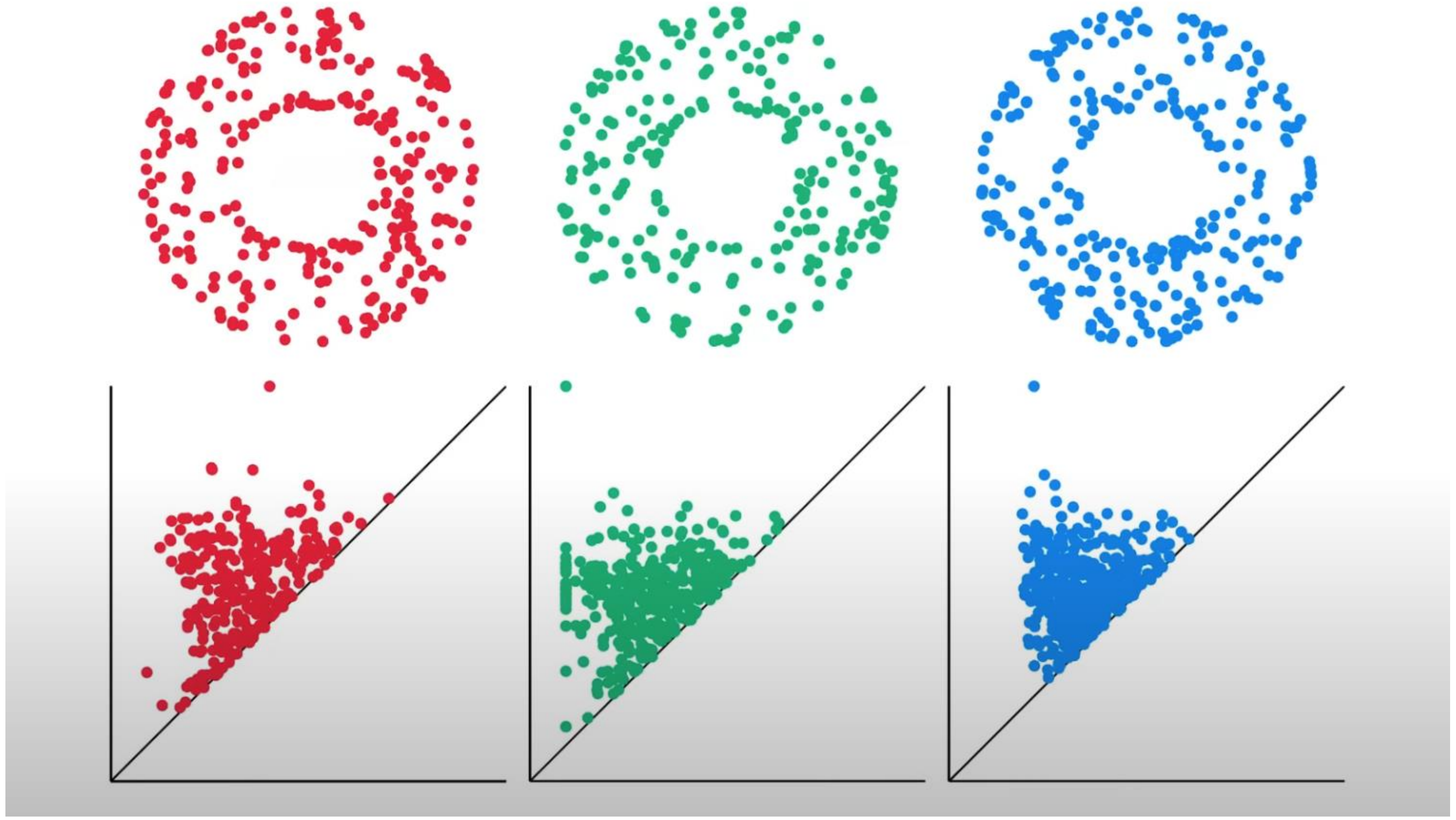
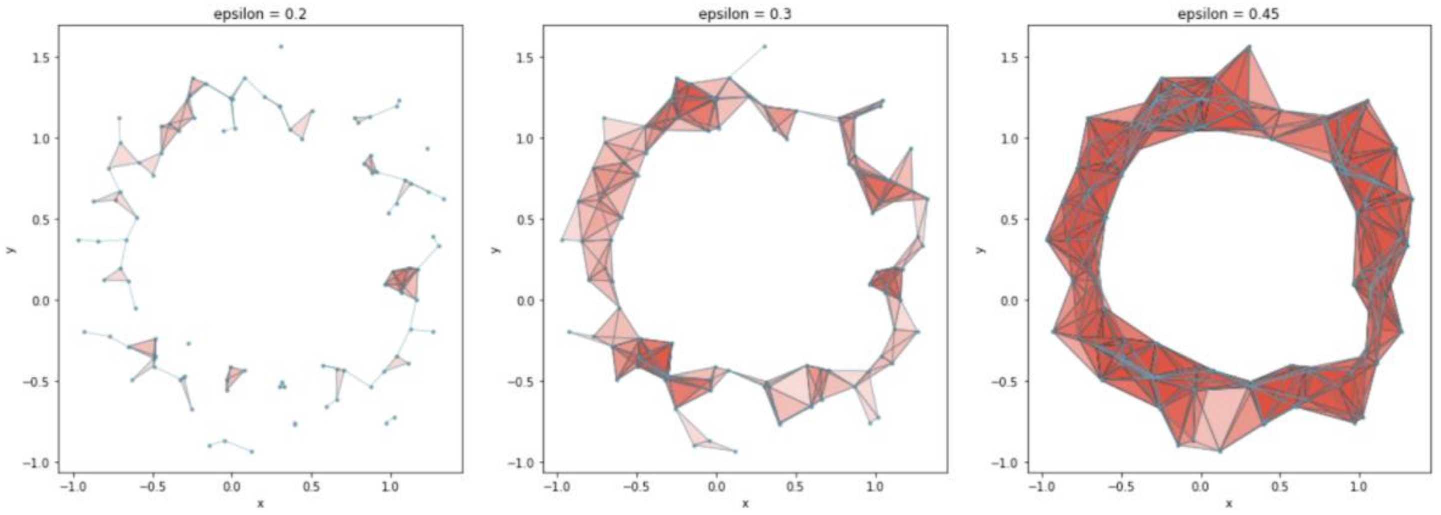


Figure from: Bastian Rieck: Topological Data Analysis for Machine Learning III: Topological Descriptors & How to Use Them (https://www.youtube.com/watch?app=desktop&v=7i1kabhl5IU)

Theoretical side of my research

- Using homology theory in algebraic topology, converting (topological) spaces into vector spaces (aka. homology groups)

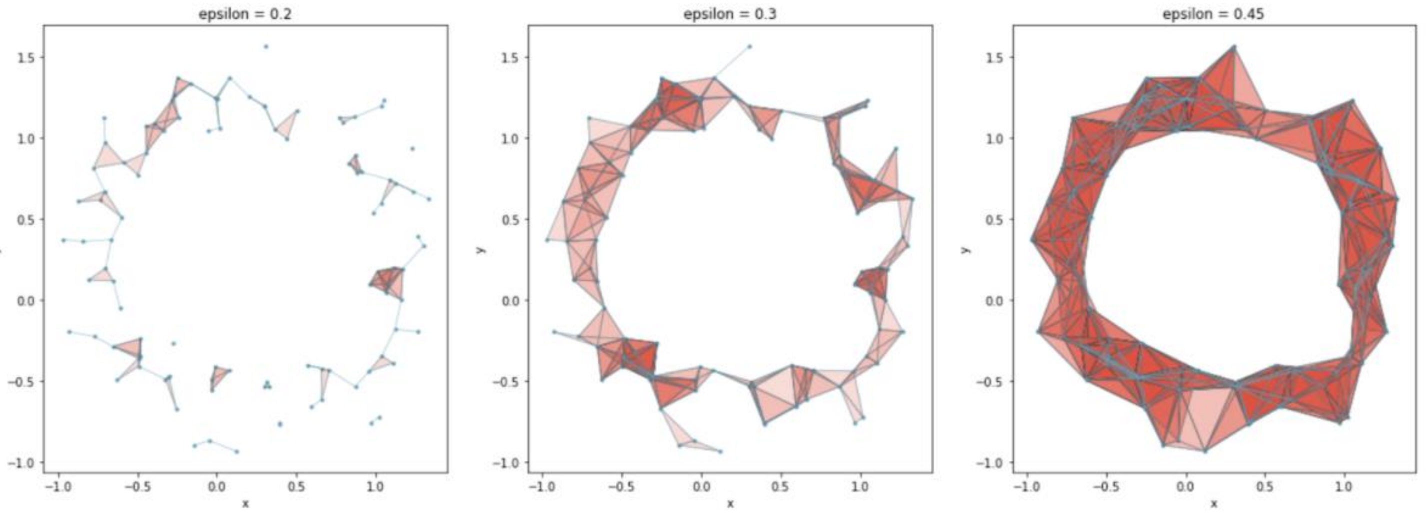


$$\begin{tikzcd}
H_0(S^{0.2}) \arrow[r] & H_0(S^{0.3}) \arrow[r] & H_0(S^{0.45}) \arrow[r, dotted] & H_0^P(S) \\
H_1(S^{0.2}) \arrow[r] & H_1(S^{0.3}) \arrow[r] & H_1(S^{0.45}) \arrow[r, dotted] & H_1^P(S) \\
H_2(S^{0.2}) \arrow[r] & H_2(S^{0.3}) \arrow[r] & H_2(S^{0.45}) \arrow[r, dotted] & H_2^P(S) \\
\vdots & \vdots & \vdots & \vdots
\end{tikzcd}$$

Figure source: Eric Bunch: https://eric-bunch.github.io/blog/topological-data-analysis-and-persistent-homology

Theoretical side of my research

- Using homology theory in algebraic topology, converting (topological) spaces into vector spaces (aka. homology groups)
- Then study the algebraic properties of the objects, e.g., dimension, homology classes, representative cycles

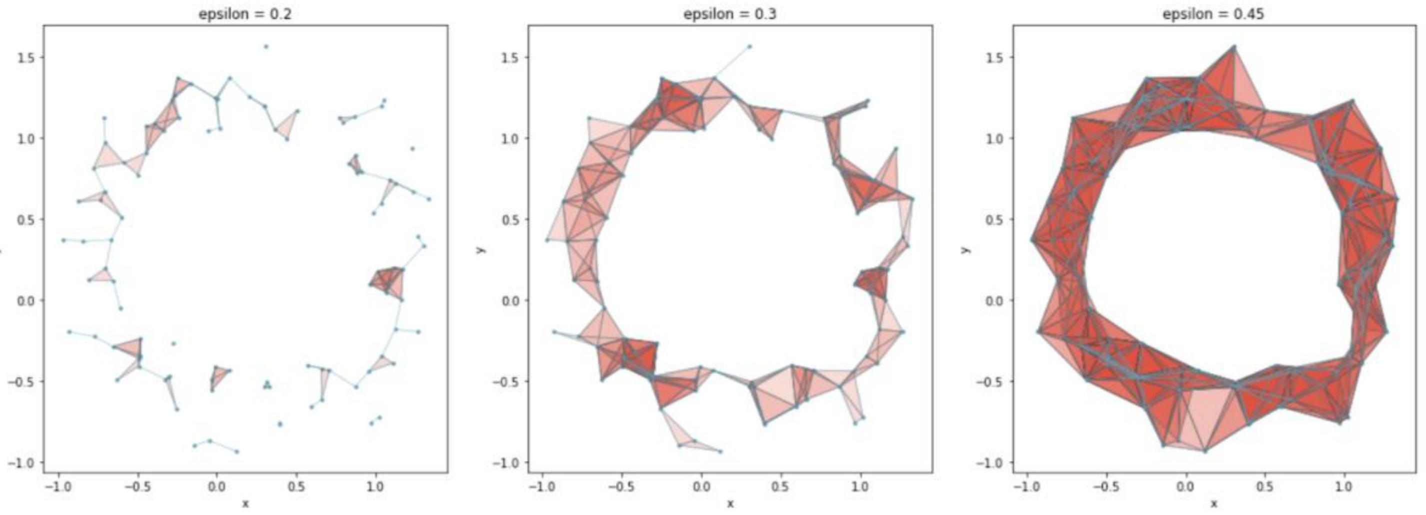


$$\begin{tikzcd}
H_0(S^{0.2}) \arrow[r] & H_0(S^{0.3}) \arrow[r] & H_0(S^{0.45}) \arrow[r, dashed] & H_0^P(S) \\
H_1(S^{0.2}) \arrow[r] & H_1(S^{0.3}) \arrow[r] & H_1(S^{0.45}) \arrow[r, dashed] & H_1^P(S) \\
H_2(S^{0.2}) \arrow[r] & H_2(S^{0.3}) \arrow[r] & H_2(S^{0.45}) \arrow[r, dashed] & H_2^P(S) \\
\vdots & \vdots & \vdots & \vdots
\end{tikzcd}$$

Figure source: Eric Bunch: https://eric-bunch.github.io/blog/topological-data-analysis-and-persistent-homology

Theoretical side of my research

- Using homology theory in algebraic topology, converting (topological) spaces into vector spaces (aka. homology groups)
- Then study the algebraic properties of the objects, e.g., dimension, homology classes, representative cycles
- My past research mainly focuses on how to compute these topological and algebraic objects (e.g., prove the NP-hardness, or come up with faster algorithms)

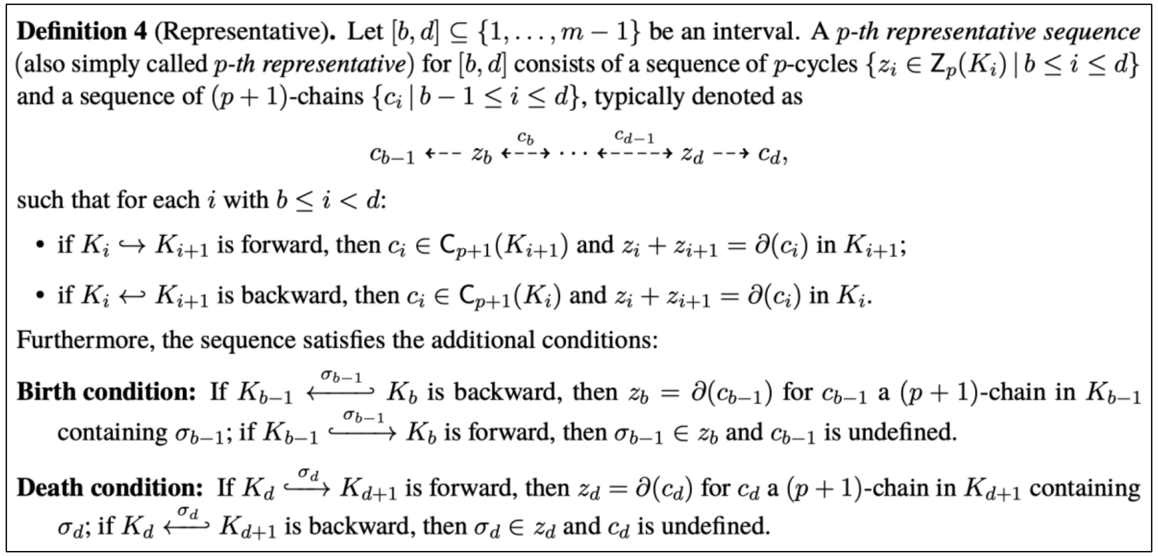


$$\begin{tikzcd}
H_0(S^{0.2}) \arrow[r] & H_0(S^{0.3}) \arrow[r] & H_0(S^{0.45}) \arrow[r, dotted] & H_0^P(S) \\
H_1(S^{0.2}) \arrow[r] & H_1(S^{0.3}) \arrow[r] & H_1(S^{0.45}) \arrow[r, dotted] & H_1^P(S) \\
H_2(S^{0.2}) \arrow[r] & H_2(S^{0.3}) \arrow[r] & H_2(S^{0.45}) \arrow[r, dotted] & H_2^P(S) \\
\vdots & \vdots & \vdots & \vdots
\end{tikzcd}$$

Figure source: Eric Bunch: https://eric-bunch.github.io/blog/topological-data-analysis-and-persistent-homology

Computing representatives for zigzag persistence

- Representative for zigzag persistence:



Definition 4 (Representative)**.** Let $[b,d] \subseteq \{1,\dots,m-1\}$ be an interval. A *p-th representative sequence* (also simply called *p-th representative*) for $[b,d]$ consists of a sequence of p-cycles $\{z_i \in \mathsf{Z}_p(K_i) \mid b \leq i \leq d\}$ and a sequence of $(p+1)$-chains $\{c_i \mid b-1 \leq i \leq d\}$, typically denoted as

$$\begin{tikzcd} c_{b-1} & z_b \arrow[l, dashed] \arrow[r, dashed, leftrightarrow, "c_b"] & \cdots \arrow[r, dashed, leftrightarrow, "c_{d-1}"] & z_d \arrow[r, dashed] & c_d, \end{tikzcd}$$

such that for each i with $b \leq i < d$:

- if $K_i \hookrightarrow K_{i+1}$ is forward, then $c_i \in \mathsf{C}_{p+1}(K_{i+1})$ and $z_i + z_{i+1} = \partial(c_i)$ in K_{i+1};
- if $K_i \hookleftarrow K_{i+1}$ is backward, then $c_i \in \mathsf{C}_{p+1}(K_i)$ and $z_i + z_{i+1} = \partial(c_i)$ in K_i.

Furthermore, the sequence satisfies the additional conditions:

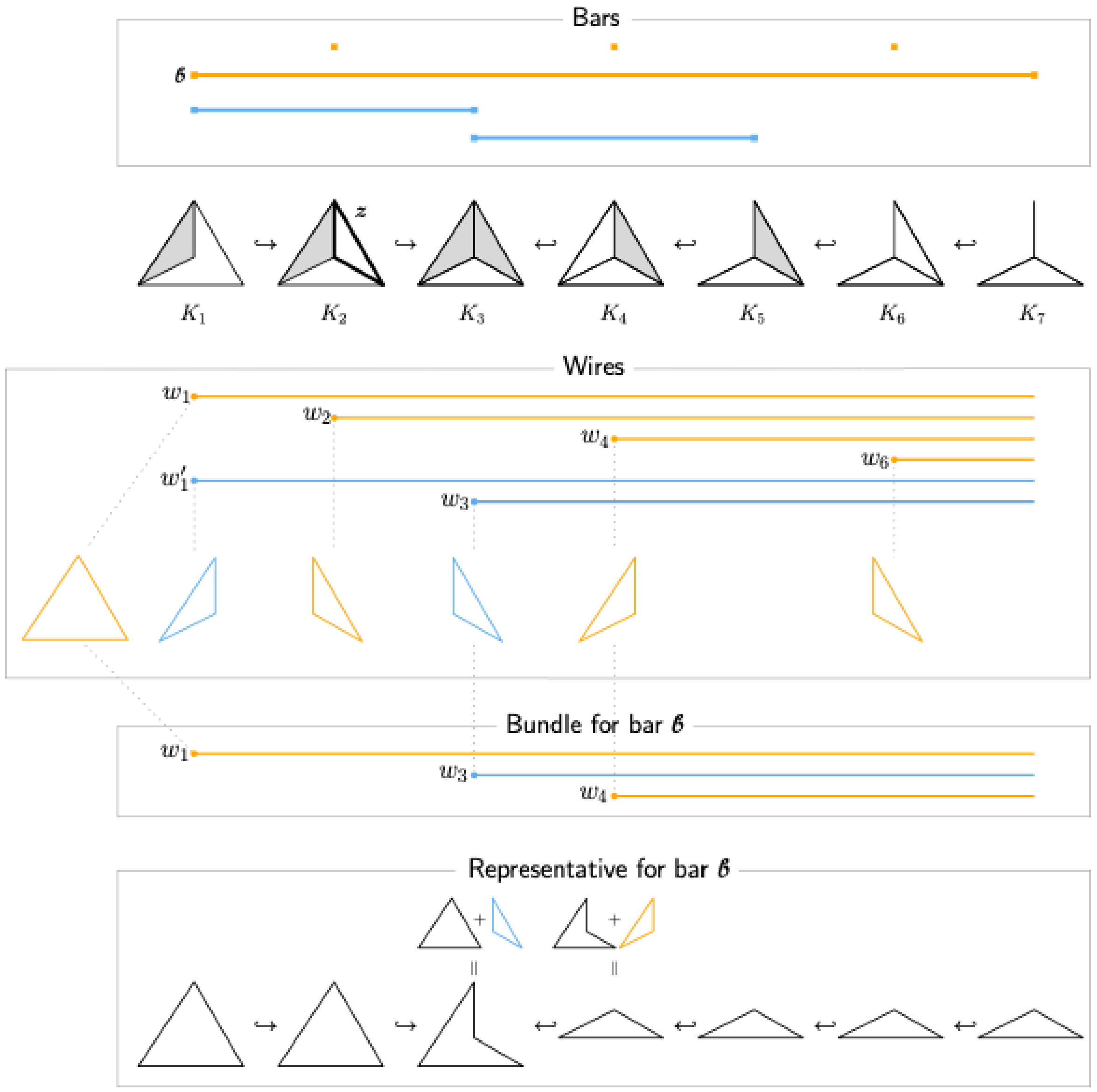
Birth condition: If $K_{b-1} \xhookleftarrow{\sigma_{b-1}} K_b$ is backward, then $z_b = \partial(c_{b-1})$ for c_{b-1} a $(p+1)$-chain in K_{b-1} containing σ_{b-1}; if $K_{b-1} \xhookrightarrow{\sigma_{b-1}} K_b$ is forward, then $\sigma_{b-1} \in z_b$ and c_{b-1} is undefined.

Death condition: If $K_d \xhookrightarrow{\sigma_d} K_{d+1}$ is forward, then $z_d = \partial(c_d)$ for c_d a $(p+1)$-chain in K_{d+1} containing σ_d; if $K_d \xhookleftarrow{\sigma_d} K_{d+1}$ is backward, then $\sigma_d \in z_d$ and c_d is undefined.

- Our improvement: $O(m^2n^2) \Rightarrow O(m^2n)$

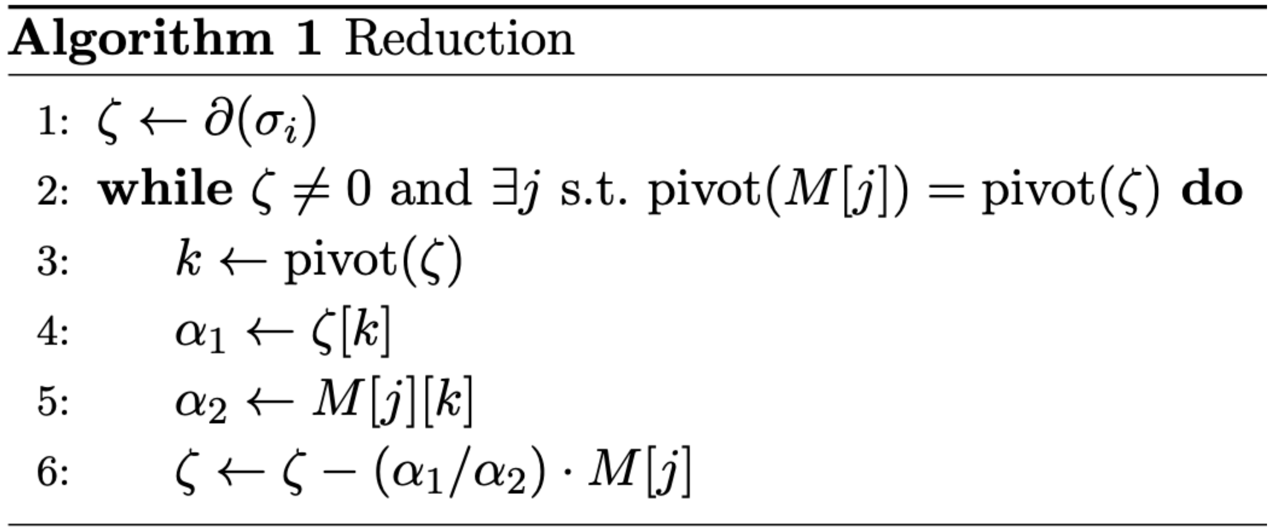
Key to bringing down the complexity

- The straightforward method takes $O(mn)$ space for storing a representative
  - Summing two representatives takes $O(mn)$ time, and hence the $O(m^2n^2)$ complexity
- We find a *compressed* way to store a representative in $O(m)$ space
  - Summing two representatives takes $O(m)$ time, and hence the $O(m^2n)$ complexity



What the algorithms that I study look like

- A lot of the algorithms I study are (more involved) versions of Gaussian elimination, which heavily relies on a process called ‘*reduction*’



Algorithm 1 Reduction

```
$\zeta \leftarrow \partial(\sigma_i)$
**while** $\zeta \neq 0$ and $\exists j$ s.t. $\mathrm{pivot}(M[j]) = \mathrm{pivot}(\zeta)$ **do**
    $k \leftarrow \mathrm{pivot}(\zeta)$
    $\alpha_1 \leftarrow \zeta[k]$
    $\alpha_2 \leftarrow M[j][k]$
    $\zeta \leftarrow \zeta - (\alpha_1/\alpha_2) \cdot M[j]$
```

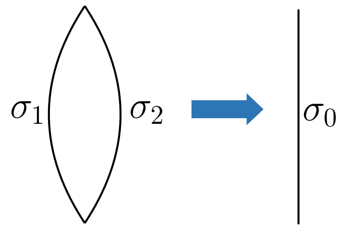
Sometimes improving the complexity relies on using some advanced data structures

- $O(m) \Rightarrow O(\sqrt{m}\log m)$: Maintain the MSF's for $\sqrt{m}$ graphs in a sequence, each MSF is a Link-Cut tree
- $O(m) \Rightarrow O(\log m)$: Use Link-Cut trees (for detecting cycles) and DFT-tree (for maintaining a merge tree)
- $O(m^{2.37}) \Rightarrow O(m \log^4 n)$: Use dynamic-connectivity and dynamic-MSF data structures

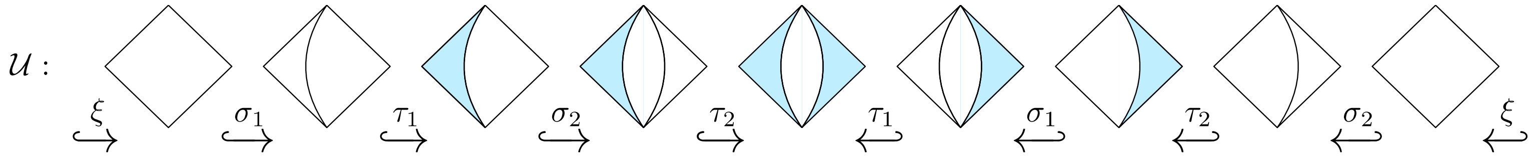
How can these improvements be done?

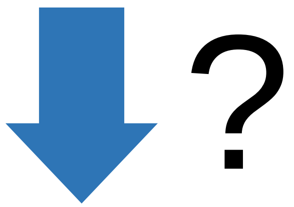
- Improving complexity for a problem typically needs a deeper understanding of the problems
- You need to make some key observations which help you make connections to existing theoretical tools or even build new tools

An example: $O(m^3) \Rightarrow O(m^2)$

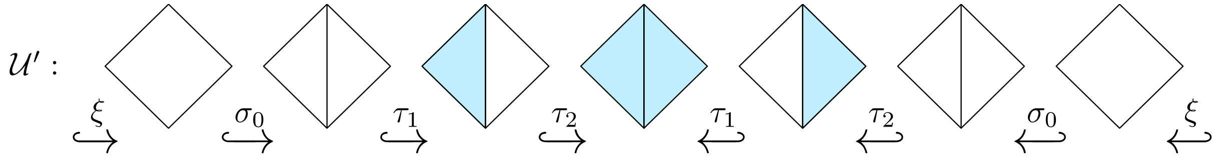


Given:

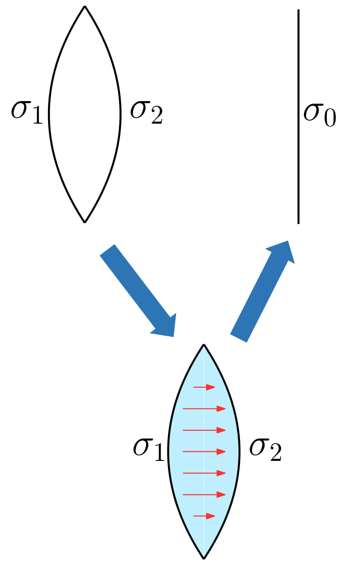




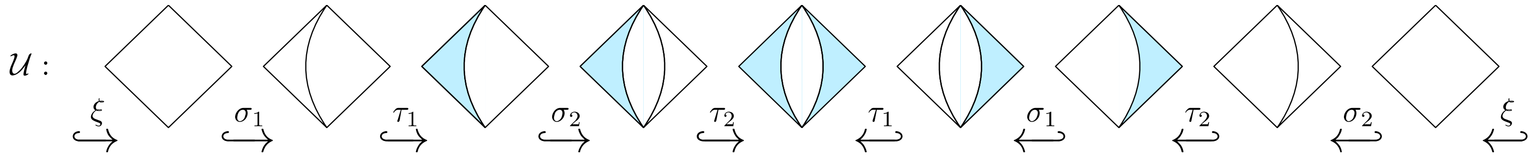
Compute:

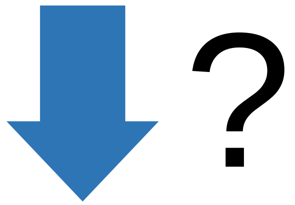


An example: $O(m^3) \Rightarrow O(m^2)$

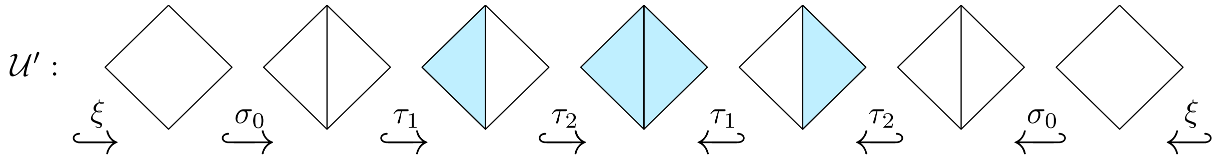


Given:

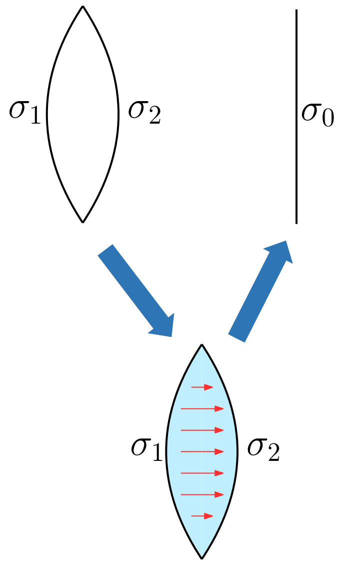




Compute:

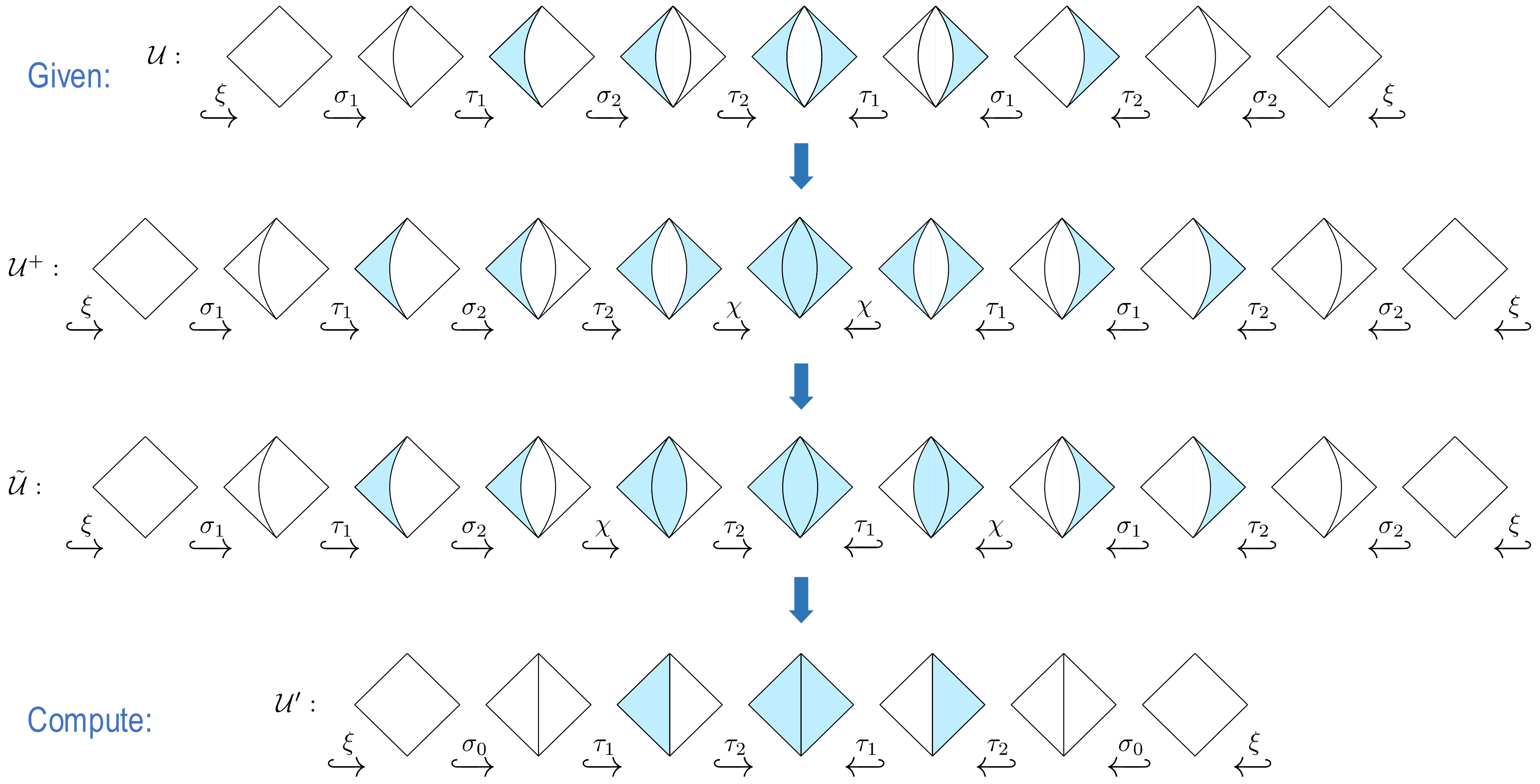


An example: $O(m^3) \Rightarrow O(m^2)$



Given:

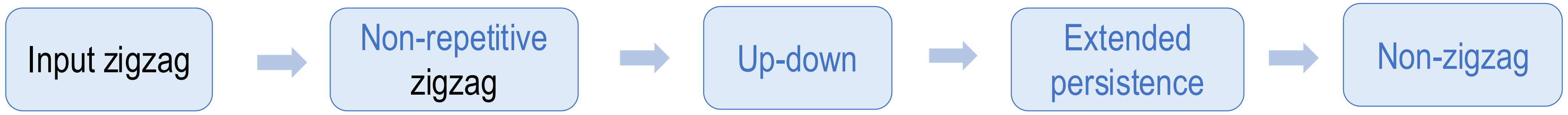
Compute:



$\mathcal{U}$:
ξ σ_1 τ_1 σ_2 τ_2 τ_1 σ_1 τ_2 σ_2 ξ
$\mathcal{U}^+$:
ξ σ_1 τ_1 σ_2 τ_2 χ χ τ_1 σ_1 τ_2 σ_2 ξ
$\tilde{\mathcal{U}}$:
ξ σ_1 τ_1 σ_2 χ τ_2 τ_1 χ σ_1 τ_2 σ_2 ξ
$\mathcal{U}'$:
ξ σ_0 τ_1 τ_2 τ_1 τ_2 σ_0 ξ

Converting the computation of zigzag persistence into computing a non-zigzag version (much faster)

- Convert input zigzag filtration to a non-zigzag filtration in linear time
- Compute barcode for non-zigzag filtration $\mathcal{F}'$
    - Fast software [Gudhi, Phat, Dionysus etc.]
- Convert barcode of $\mathcal{F}'$ to that of $\mathcal{F}$
    - $O(1)$ conversion per bar



All filtrations have the same length (the same number of addition/deletions)

Conversions 1,2,3,4:
- Done by a simple linear scan of the input filtration

NP-hardness proof for computing minimum representatives for persistent homology

Base case

Theorem. *PCYC-FIN$_1$ is NP-hard.*

- Reduction from MAX-2SAT

Theorem. *WPCYC-INF$_2^+$ is NP-hard to approximate with any fixed ratio.*

- Reduction from the *nearest codeword* problem

NP-hardness proof for computing minimum representatives for persistent homology

Generalization to higher dimension

Proposition. *PCYC-FIN*$_{p-1}$ *reduces to PCYC-FIN*$_p$ *for* $p \geq 2$.

Proposition. *WPCYC-INF*$^+_{p-1}$ *reduces to WPCYC-INF*^+_p *for* $p \geq 3$.

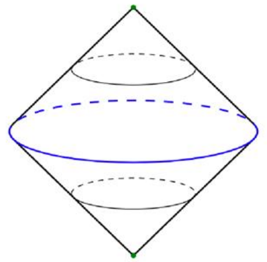
Using the *suspension* operator

(inspired by [Chen and Freedman, 2011])

Definition (Suspension)

$$\mathcal{S}K = \{\{\omega_1\}, \{\omega_2\}\} \cup K \cup \left(\bigcup_{\sigma \in K} \{\sigma \cup \{\omega_1\}, \sigma \cup \{\omega_2\}\}\right)$$

ω_1 and ω_2 are two extra vertices



Suspension of $\mathbb{S}^1 \Rightarrow \mathbb{S}^2$:

	$\widetilde{H}_0$	$\widetilde{H}_1$	$\widetilde{H}_2$	$\widetilde{H}_3$	$\cdots$
$\mathbb{S}^1$	0	$\mathbb{Z}_2$	0	0	$\cdots$
$\mathbb{S}^2$	0	0	$\mathbb{Z}_2$	0	$\cdots$

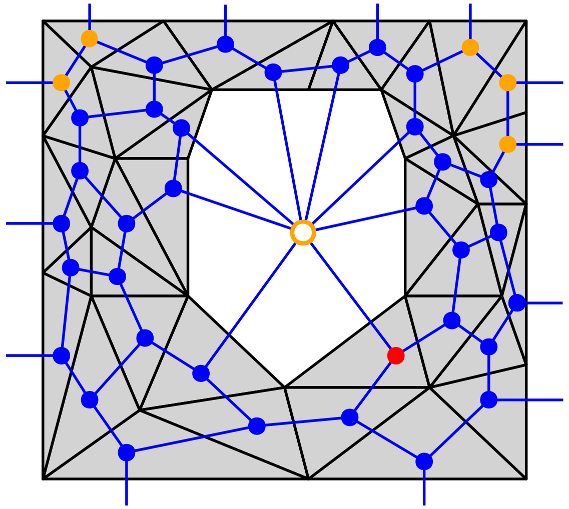
(Figure from Wikipedia)

Computing optimal representatives for *manifolds*

- Weighting an edge e in the graph:
  - Dual p-simplex in K_b: same weight
  - Otherwise: $+\infty$
- Source: Dual vertex of σ_d
- Target: Dual vertices of $(p+1)$-simplices *not* in K_d $\cup$ the *infinite vertex*

Process of algorithm:

1. Build dual graph, assign weight
2. Compute the minimum (s, t)-cut for the graph with specified source and target
3. Return the p-cycle dual to the minimum cut as a minimum persistent p-cycle for $[b, d)$



A short bio of me

- Finished bachelor’s and master’s in China
- Worked there in industry for 3 years
- A key event in my life leading me to academia: two books
  - *Linear algebra done right* by Sheldon Axler
  - *Introduction to algorithms* by CLRS
  - Honorable mention: *Introduction to Linear Algebra* by gilbert strang
- I used to read these books pages by pages, problems by problems, exercises by exercises
- Reading these books made me realize that computer theory and mathematics are something that I really want to do
- (Due to my industry working experience, I am still passionate about coding, but I don’t have too much time for coding these days…)

A short bio of me (continued)

- Started my PhD originally at Ohio State in 2016
- Transferred to Purdue w. Advisor in 2020
- Graduated in 2022
- Before coming to UO this year, I was a faculty at DePaul U. (in Chicago) for two years

Suggestions for those who want to pursue research

- Do something that truly inspires you so that you can inspire others
- Make your own observation, do original work

A course advertisement

- I will be teaching a new 410/510 on my research in Spring 2025
- Tentatively called: *Machine Learning and Data Analysis with Topological Priors*
- Applications and best practices will be stressed
- Coding libraries in TDA will be reviewed
- Mathematical background is not necessary: I will try to focus only on the intuition necessary for doing applications

Thank you!