

Updating Persistence Dynamically

Swansea TDA Group – 04/05/21

- persistence is stable wrt changes in a filtration



i.e. Lipschitz Continuous

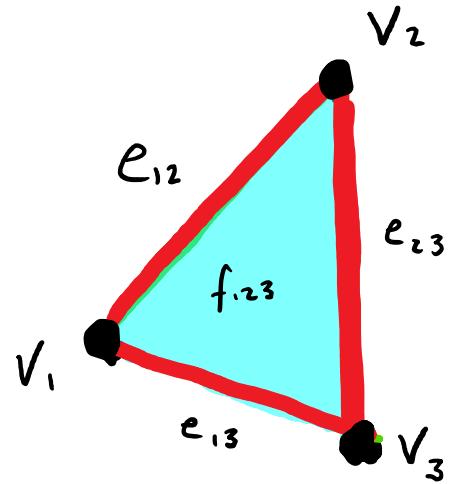
Continuous modification
of filtration F



"continuous" change in
persistence diagram

- Can we track the PH as we change F ?

The Persistence Algorithm ($\mathbb{Z}/2\mathbb{Z}$)



ordered by value

$D =$ filtration

	v_1	v_2	v_3	e_{12}	e_{23}	e_{13}	f_{123}
v_1	-	-	-	-	-	-	-
v_2	-	-	-	-	-	-	-
v_3	-	-	-	-	-	-	-
e_{12}	-	-	-	-	-	-	-
e_{23}	-	-	-	-	-	-	-
e_{13}	-	-	-	-	-	-	-
f_{123}	-	-	-	-	-	-	-

Find matrices $RU = D$ s.t.

- R is reduced: $i \neq j \Rightarrow \text{low}_R i \neq \text{low}_R j$
- U is upper Δ

Then $\left\{ [F(j), F(i)] \mid j = \text{low}_R i \right\}$ is our barcode

A basic algorithm:

```
1   |   R := D
2   |   for i := 1 to n
3   |       |   while ∃ j < i st. lowRj = lowRi
4   |       |   |   R[:, i] += R[:, j]
```

has worst case $O(n^3)$ running time and despite the many improvements made to this method, computing the persistence of large complexes is still expensive.

- o Lemma

(Cohen-Steiner,
Edelsbrunner,
Morozov. 2006)

Suppose $RU = D = R'U'$ are

two decompositions of D . Then $\forall i$

$$\text{low}_{R,i} = \text{low}_{R',i}.$$

- o i.e. any decomposition will do.

- o idea : given a change to the filtration,
try to fix an existing decomposition

Vineyards

- Suppose we change $F(i)$ from a to b .
 - Imagine this happens continuously, eg via a straight line homotopy $F(i) = tb + (1-t)a$.
 - Then as we track this path, the finest changes to D is a transposition of adjacent columns/rows.

i.e. $D' = PDP$ with $P = \begin{bmatrix} 1 & \dots & 0 & 1 \\ \vdots & \ddots & \vdots & 0 \\ 0 & \dots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ i & j & \vdots & \vdots \\ j & i & \vdots & \vdots \end{bmatrix}$

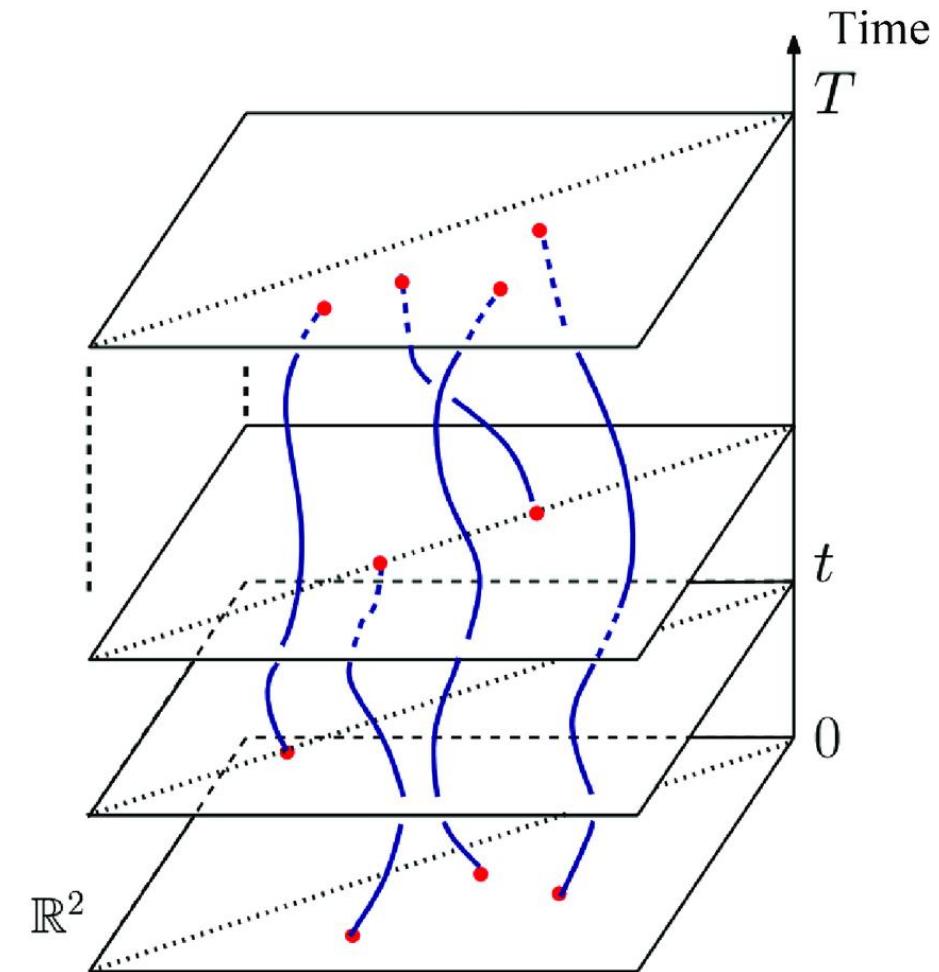
A diagram showing a 4x4 grid of dots representing a matrix. A red arrow points from the top-left dot to the bottom-right dot, indicating a transposition. The top row has dots at positions (1,1), (1,2), (1,3), and (1,4). The bottom row has dots at positions (4,1), (4,2), (4,3), and (4,4). The left column has dots at positions (1,1), (2,1), (3,1), and (4,1). The right column has dots at positions (1,4), (2,4), (3,4), and (4,4). The diagonal from (1,1) to (4,4) is solid black, while the other diagonals are dashed black.

- But $PDP = PRUP = (PRP)(PUP)$ may not be a valid decomposition:
 - PRP not reduced ($\exists i \neq j$ s.t. $\text{low}_{PRP}^i = \text{low}_{PRP}^j$) and/or ◦ PUP not upper Δ
 - Cohen - Steiner, Edelsbrunner and Morozov demonstrate how to make minor modifications to PRP and PUP to repair the decomposition with at most 4 column / row additions (i.e. $O(n)$ time).

- Now given a continuous 1-parameter family of filtrations
(or discrete sequence sampled finely enough or assuming straight line homotopy)

We can track the changes
in the persistence diagram as

a Vineyard



Moves

- What if the change to the filtration is big?
Then we might have to perform many transpositions
(worst case $O(n^2)$).
- Suppose our change moves simplex i to position j with $|i-j| \neq 1$. Then if we don't care about the intermediate steps, the Vineyard transposition approach is wasteful.

- Instead Busaryev, Dey and Wang introduced moves which track the transpositions of moving i to position j but does not guarantee that the intermediate steps are valid decompositions.
- This removes a lot of the overhead, at the cost of producing fewer persistence diagrams (lower vineyard resolution).

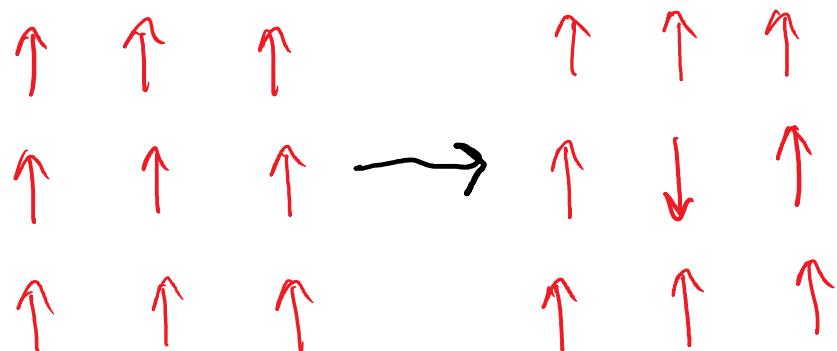
- If the change between your filtrations is fairly substantial, with many simplices changing position in the order, then some recent work by Piepenbrock and Perea examines how to perform the permutation via an (approx.) minimal number of move operations.

- Instead of $O(n^2)$ transpositions, the worst case here is $O(n - \sqrt{n})$ moves (with high probability).

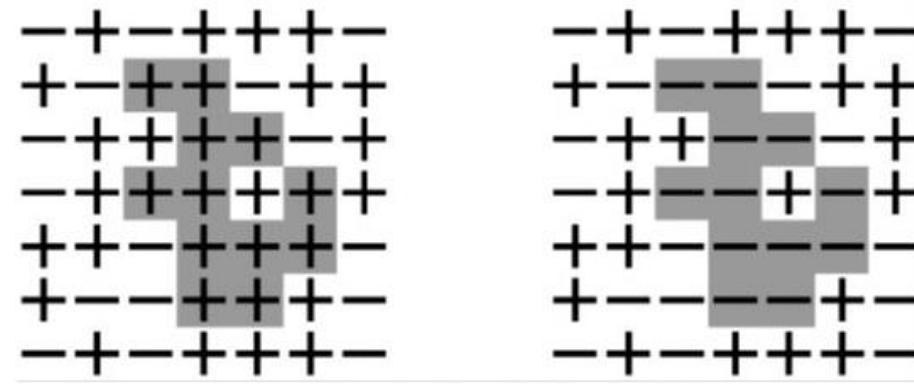
Applications to Statistical Physics?

- We study lattice models using Monte Carlo, often by proposing changes which lead to local changes in filtrations.

e.g. Metropolis



Wolff



1) Performance? \sim Samples, L^z MC steps / sample

Computing PH each time

$$\sim O(nV^3)$$

dynamic PH

$$\sim O(V^3 + nV^2L^z)$$

(z typically ~ 2)

2) Tracking the physics?

If the points in our PD correspond to physically relevant features (eg. Vortices, domain walls, etc)

then can we track when they appear / leave?

3) Autocorrelation of PH-based Statistics?

How many MC steps do we need between samples? Perhaps when we've seen that the PH has changed sufficiently (via its vineyard).