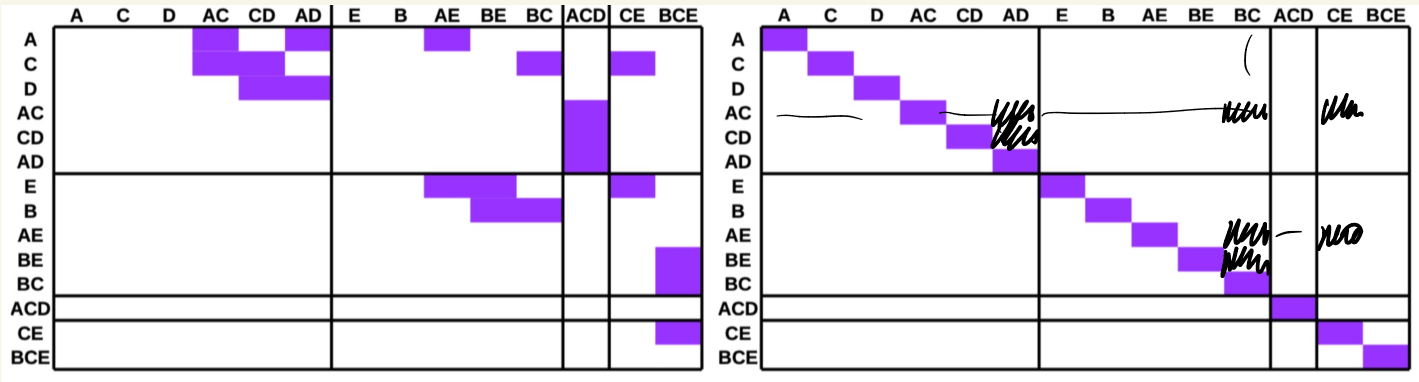
TDA - fall 2025

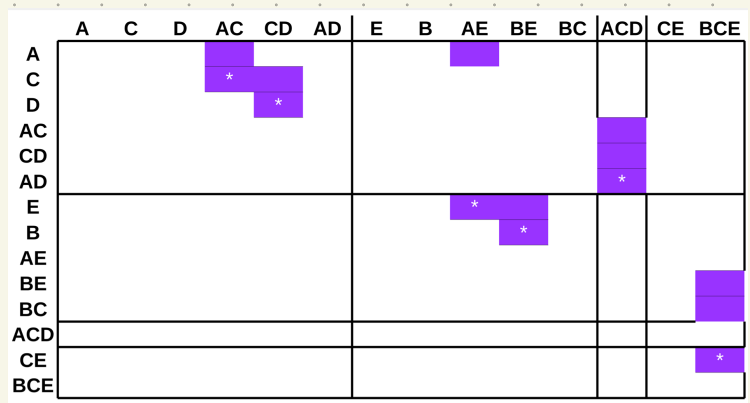
Persistence implementations

Last time: algorithm

$R = B$
for $j = 1 \cdots m$ **do**
  while $\exists\, j' < j$ with $low(j') = low(j)$ **do**
    add column j' to column j
  end while
end for



=



Complexity

With m simplices, matrix has size $m \times m$

```
R = B
for j = 1 ··· m do
    while ∃ j' < j with low(j') = low(j) do
        add column j' to column j
    end while
end for
```

repeats m times

while loop:
each such operation must move low(j) up, but other colums can become 1 during operation

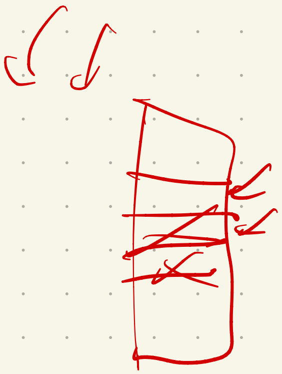
Worst case: $O(m)$ additions to cancel a row

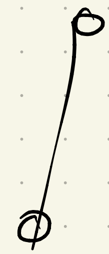
each "add" takes $O(m)$

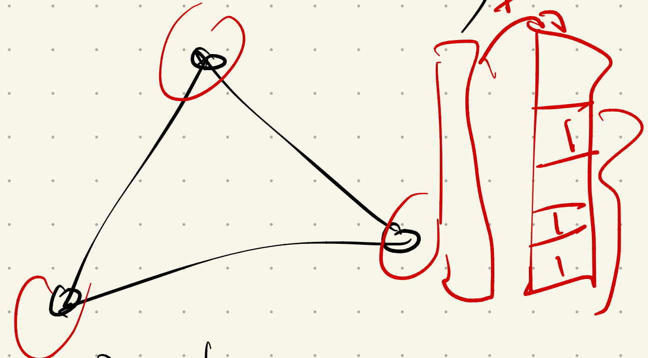
$\Rightarrow O(m^3)$

Improving this runtime

① Note that the matrix is sparse: a simplex of dimension d only has $d+1$ co-faces







So compressed representation helps in practice

② The algorithm can also be reduced to Guassian elimination

$\hookrightarrow O(n^{\omega})$ time, $\omega = [2, 2.373)$

More speedups

Matrix algorithm last time:

- Sweep columns left to right
- every addition to col j moves low[j] higher
- If additive, will 0 it out
  - ↳ might take many column ops
- But, once we zero it out, it's never used again!

Practical improvement

$\boxed{\text{Bauer et al 2014}}$

Process filtration in backwards order (highest dim first)

Then for pair $(\sigma^{p-1}, \sigma^{p})$

$\underbrace{\text{In } \partial_{p-1}}_{\text{birth}} \qquad \underbrace{\text{In } \partial_{p}}_{\text{death}}$

Result: can skip earlier column (since we know it will be 0's)

Another method: Collapses

Boissonnat et al 2018

A simplicial cone for a complex L and a vertex a not in L is

$$aL = \{ \tau \mid \tau \in L \text{ or } \tau = \sigma \cup a, \sigma \in L \}$$

A vertex is dominated if $\mathrm{lnk}(v)$ is a simplicial cone:

$\exists v' \neq v$ and $L \subseteq K$ s.t. $\mathrm{lnk}_K(v) = v'L$

collapse:

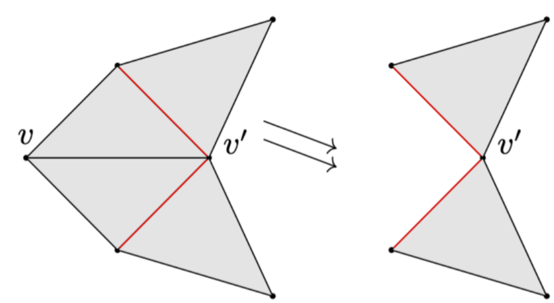


Figure 1 Illustration of an *elementary strong collapse*. In the complex on the left, v is dominated by v'. The link of v is highlighted in red. Removing v leads to the complex on the right.

If we collapse all possible vertices, get core K^c, which is unique up to isomorphism & has same homotopy type.

↳ via a retract

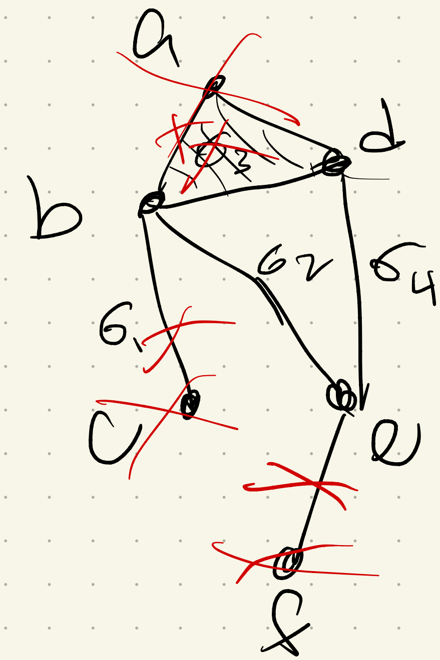
Can compute via matrix:

maximal simplicies

	σ_1	σ_2	σ_3	σ_4	σ_5
a	0	0	1	0	0
b	1	1	1	0	0
c	1	0	0	0	0
d	0	0	1	1	0
e	0	1	0	1	1
f	0	0	0	0	1

	b	d	e
σ_1	1	0	0
σ_2	1	0	1
σ_3	1	1	0
σ_4	0	1	1
σ_5	0	0	1

	σ_2	σ_3	σ_4
b	1	1	0
d	0	1	1
e	1	0	1



Aside:

Lots of work on speeding up persistence

↳ and on finding lower bounds: cases where you really need $O(n^\omega)$ time.

Not mentioned: paralleization

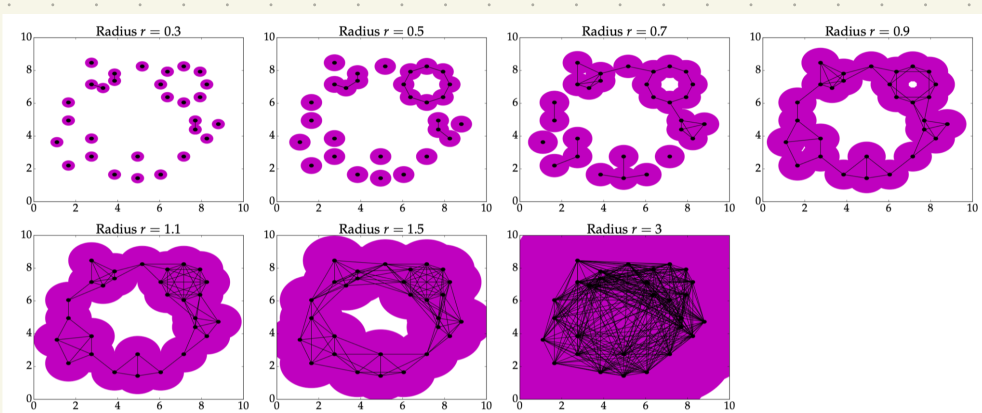
The workflow so far

Build a filtration F from a simplicial complex

↳ usually parameterized by a function f, via sublevel sets

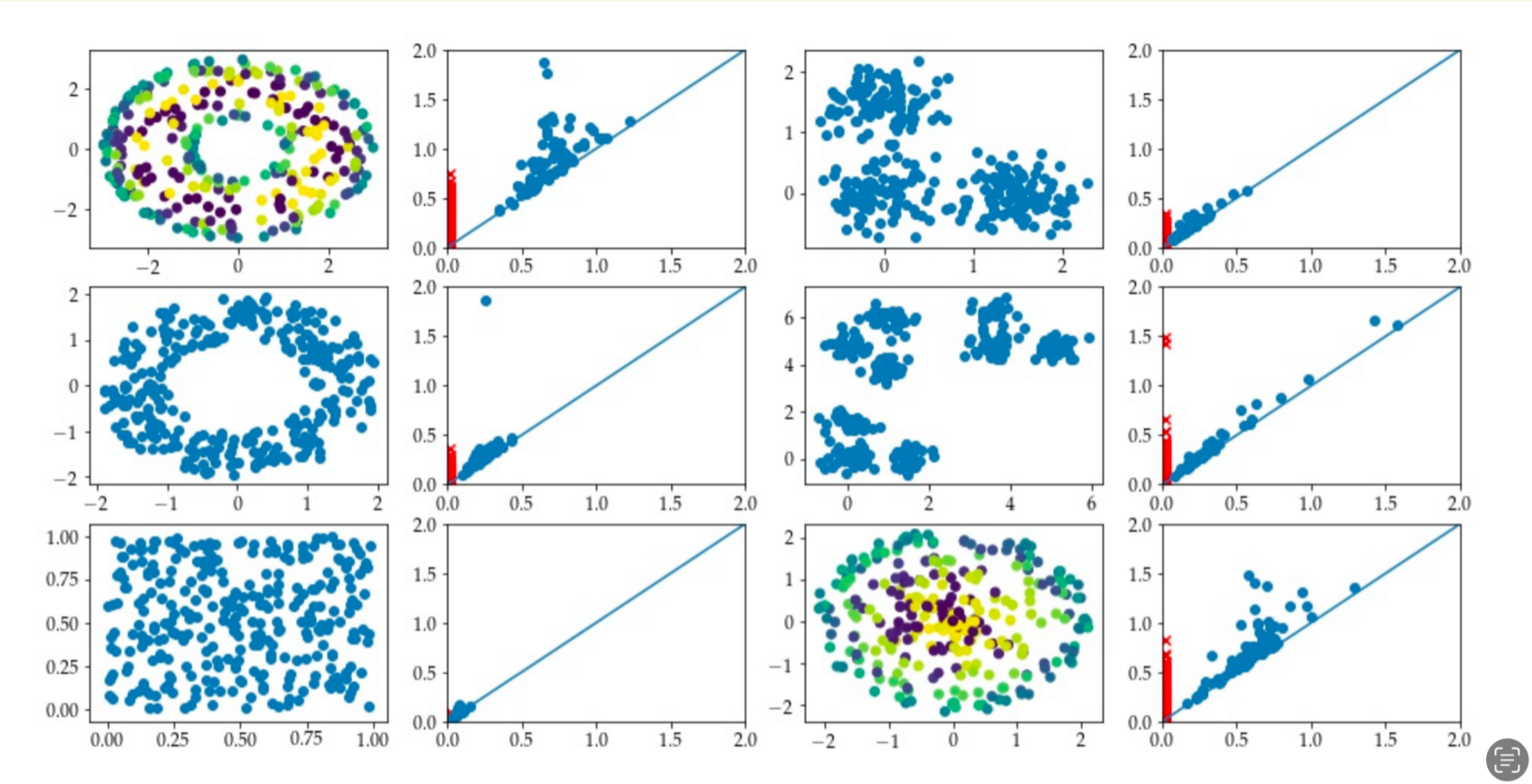
Example: $P \subseteq \mathbb{R}^n$

Build Rips filtration: for $0 \le r_1 \le \cdots \le r_k$, $K_i = VR(R, r)$



$\Rightarrow$ Persistence diagram

Result: H_0 and H_1



... now what?

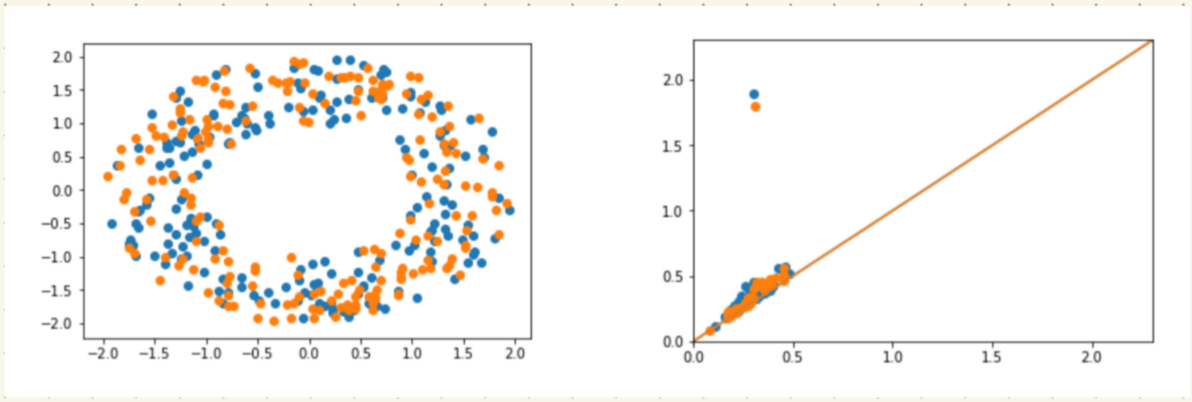
Distance measures

A distance on a set X is a function
$d: X \times X \to \mathbb{R}_{\geq 0}$ s.t. $\forall x, y, z \in X$

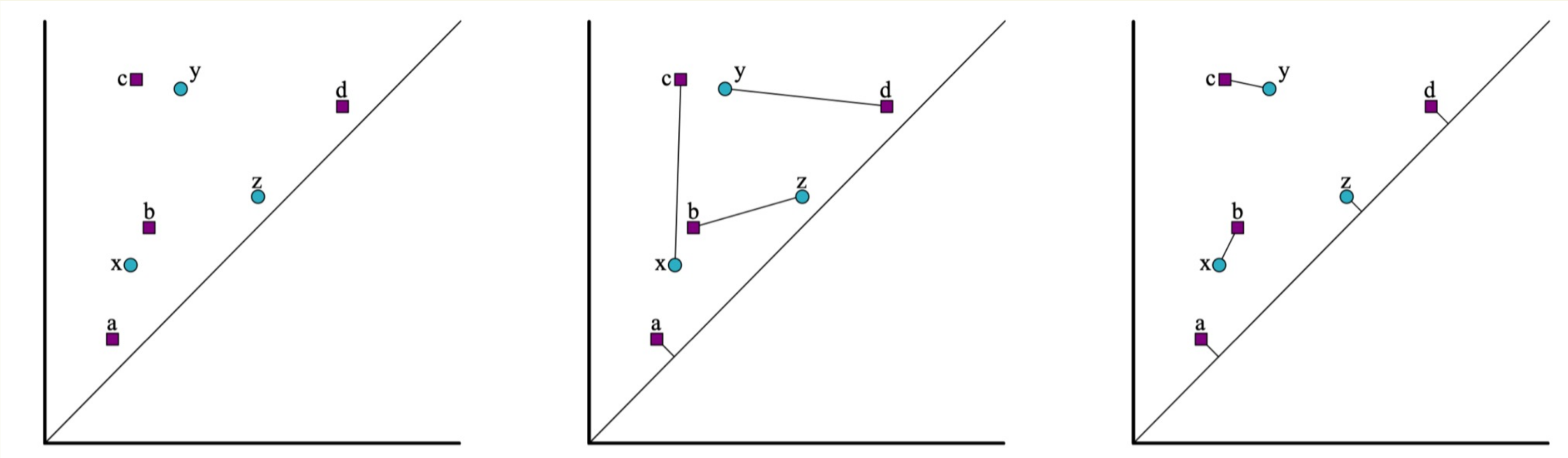
- $d(x,y) \geq 0$ + $d(x,y) = 0 \iff x = y$
- $d(x,y) = d(y,x)$
- $d(x,z) \leq d(x,y) + d(y,z)$

Our goal: distances for PDs

- $X = \{\mathrm{Dgm}\, F(K)\}_{F,K}$



How to quantify "nearby" here?

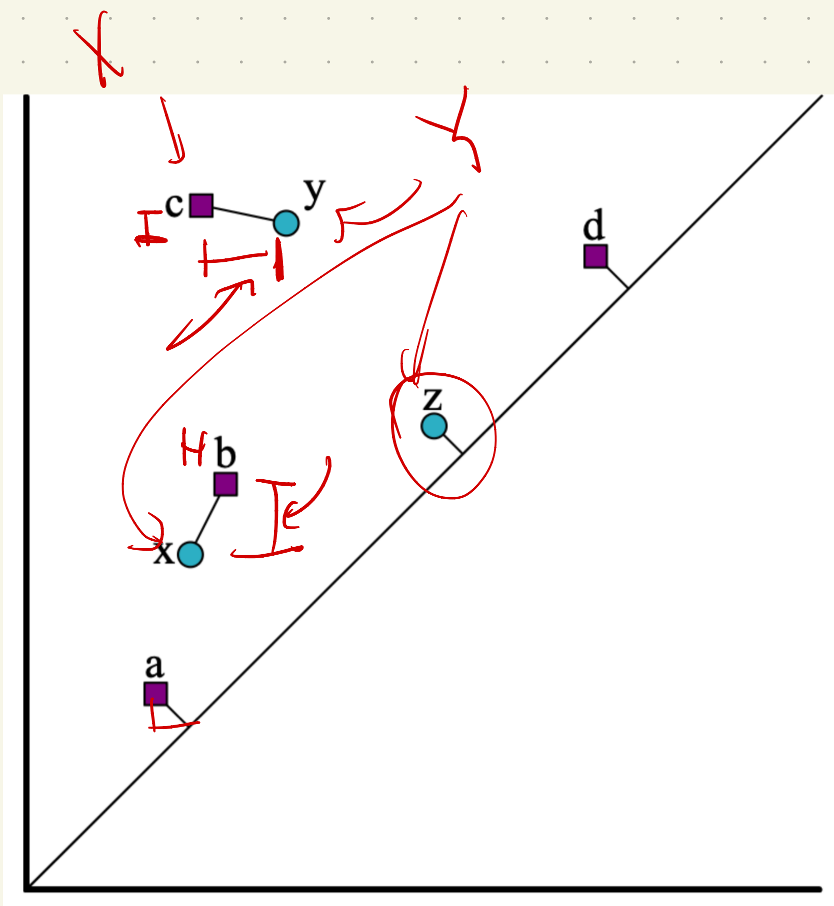


Major goal: stability

Bottleneck distance (take 1)

Given 2 diagrams $X, Y \subset \mathbb{R}^2$,

$$d_B(X,Y) = \inf_{\varphi : X \to Y} \sup_{x \in X} \| x - \varphi(x) \|_\infty$$



Here:

$\varphi(a) = \text{diagonal}$

$\varphi(b) = x$

$\varphi(c) = x$

$\varphi(d) = \text{diagonal}$

Alternate definition

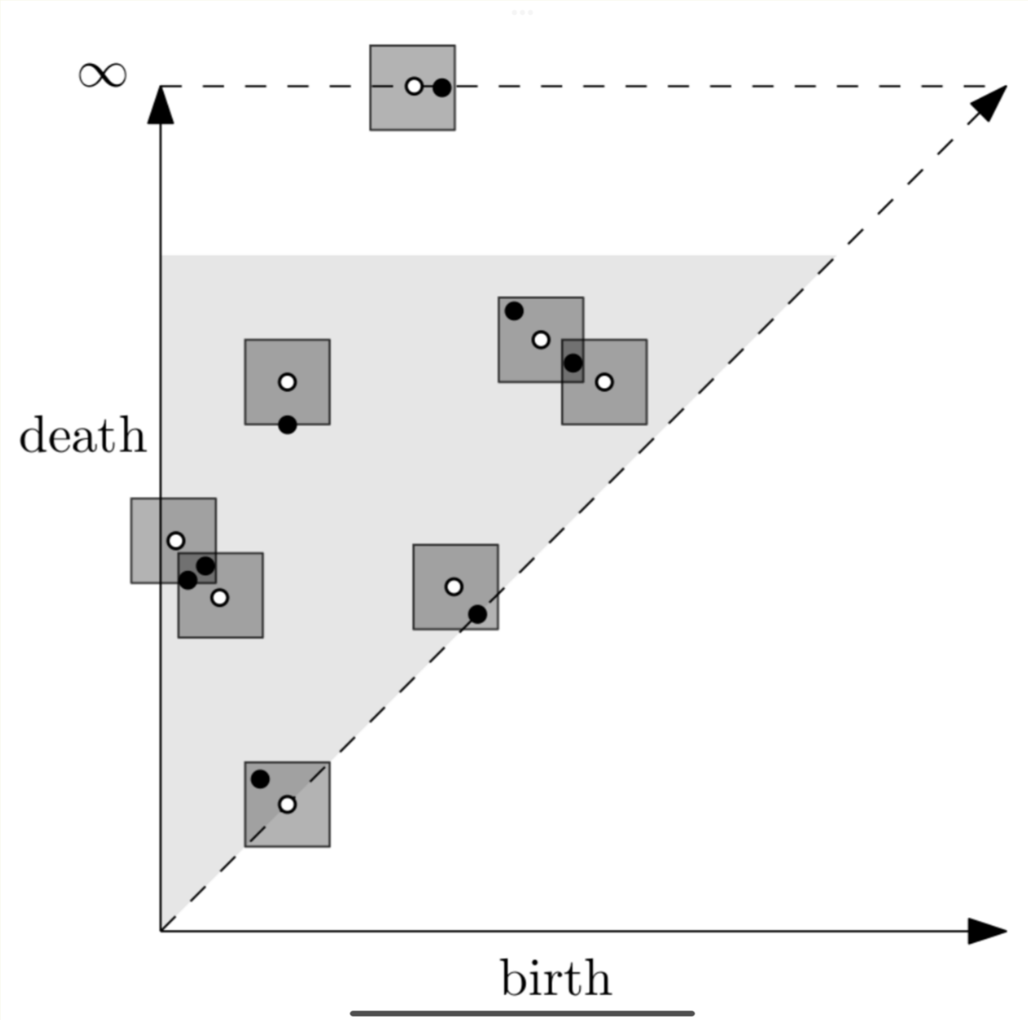
A **matching** between X + Y is a bijection on a subset of the off-diagonal points of the 2 diagrams, $X' \subseteq X$ + $Y' \subseteq Y$.

Cost of matching:

$$c(M) = \sup\{\|x - M(x)\|_\infty \mid x \in X'\} \cup \{\tfrac{1}{2}|x_1 - x_2| \text{ s.t. } (x_1, x_2) \in X \setminus X' \cup Y \setminus Y'\}$$

Bottleneck $d_B(X,Y) = \inf_M c(M)$.

Another view: L_∞ balls in $\mathbb{R}^2$

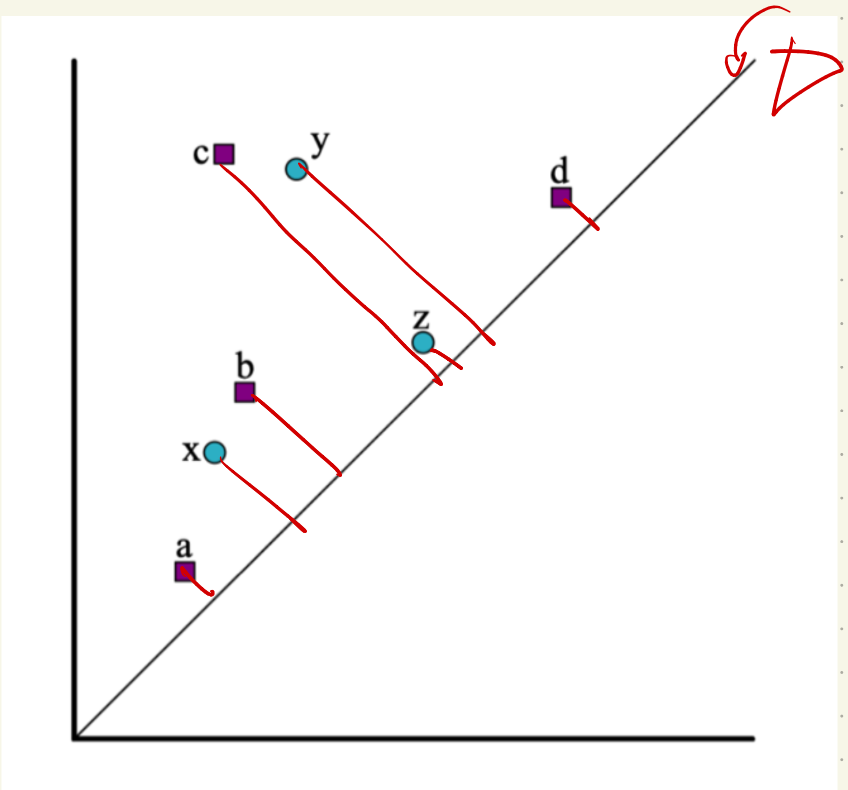


Min ε s.t.
ε balls around
every point
either:

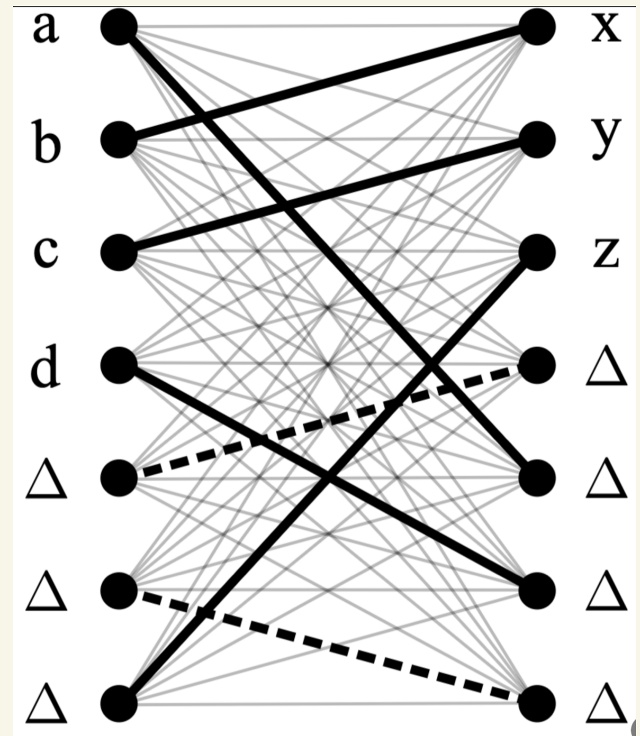
- includes at least 1 point from other set
- touches the diagonal

How to compute?

Reduce to a graph problem



$\Rightarrow$



Min cost matchings: use network flow on graph

$\rightsquigarrow O((\#\text{per points})^2)$