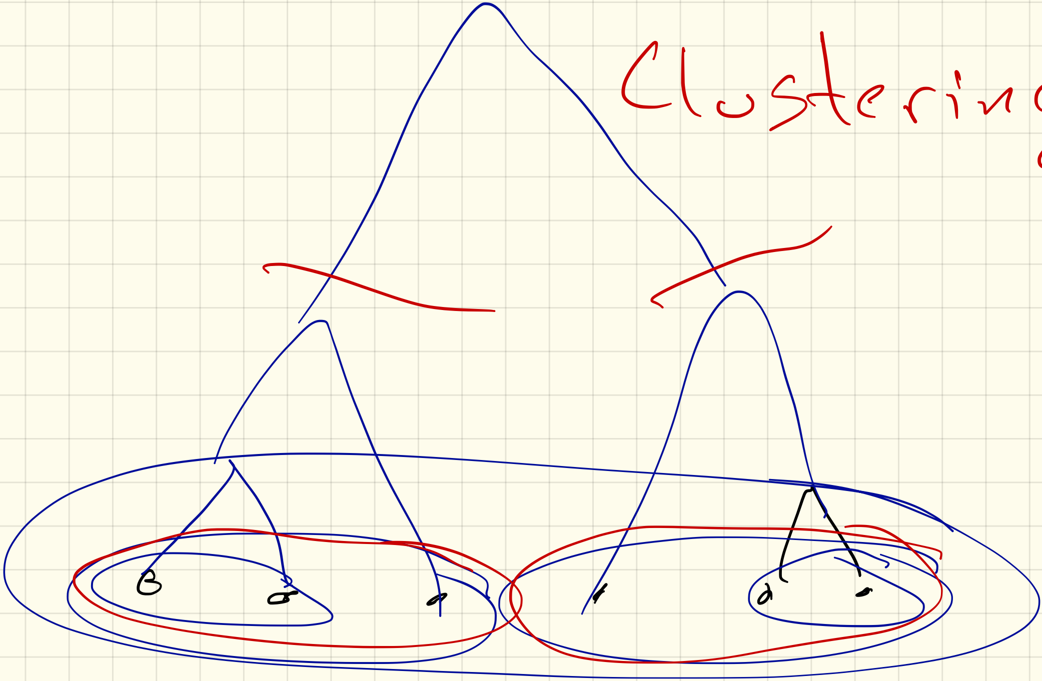
Data Mining

L8 – Hierarchical Clustering



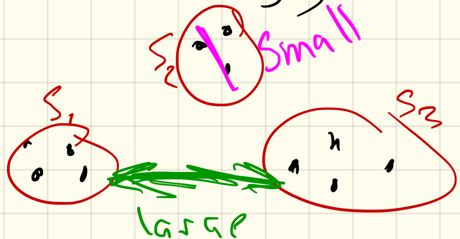
What is Clustering?

<u>Input</u> $\underset{\text{data}}{X} \subset \mathcal{M}$ (e.g. $\mathcal{M} = \mathbb{R}^d$)

distance $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$

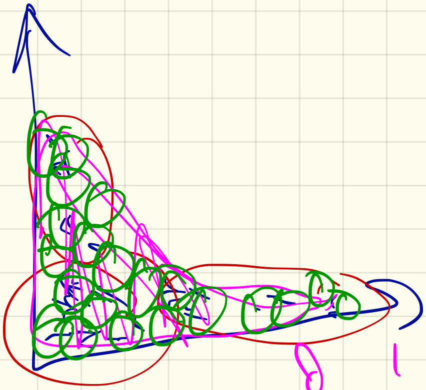
<u>Goal</u>: $\vec{S} = \{S_1, S_2, \ldots S_k\}$

- $S_i \subset X$
- $S_i \cap S_j = \emptyset$ (hard clustering)
- $\bigcup_{i=1}^{k} S_i = X$

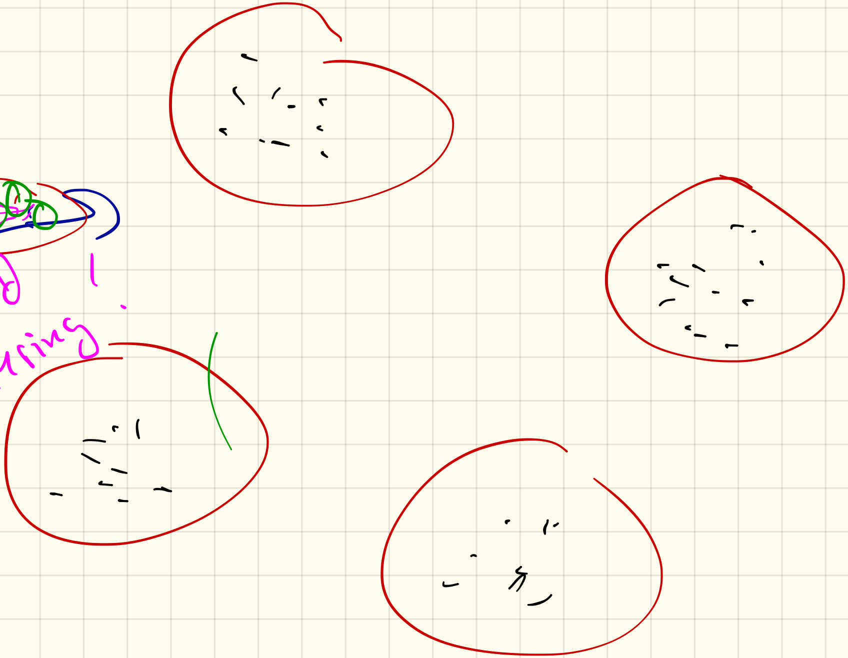


Most Examples $X \subset \mathbb{R}^2$

$d = \| \cdot - \cdot \|$



no good clustering!



1. plot
2. draw circles

Hierarchical Agglomerative Clustering

- If two points (or clusters) are close enough

→ put together in same cluster.

0. Each $x_i \in X$ → separate cluster S_i
   ($|X| = n$ → n clusters)

1. while (2 clusters S_i, S_j are close enough) threshold
   - Find closest pair S_i, S_j
   - Merge S_i, S_j → single cluster $S = S_i \cup S_j$

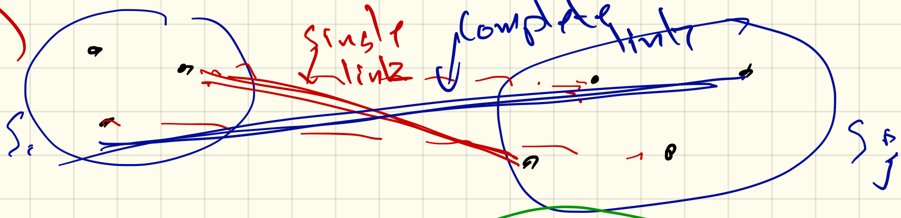
Distance between clusters S_i, S_j

$$\boxed{d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}} \qquad x \in \mathcal{M}$$

At step 1, $S_i = \{x_i\}$, $S_j = \{x_j\}$, then $d(S_i, S_j) = d(x_i, x_j)$

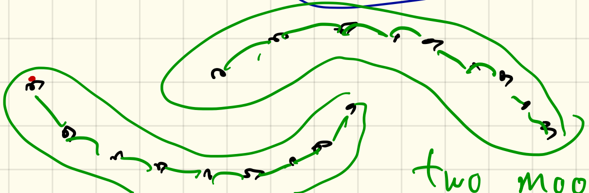
Single-Link $S_i = \{x_1, x_3, x_7\}$ $\quad S_j = \{x_2, x_8, x_9, x_{10}\}$

$$d(S_i, S_j) = \min_{\substack{x_i \in S_i \\ x_j \in S_j}} d(x_i, x_j)$$



Complete-Link

$$d(S_i, S_j) = \max_{\substack{x_i \in S_i \\ x_j \in S_j}} d(x_i, x_j)$$



two moons

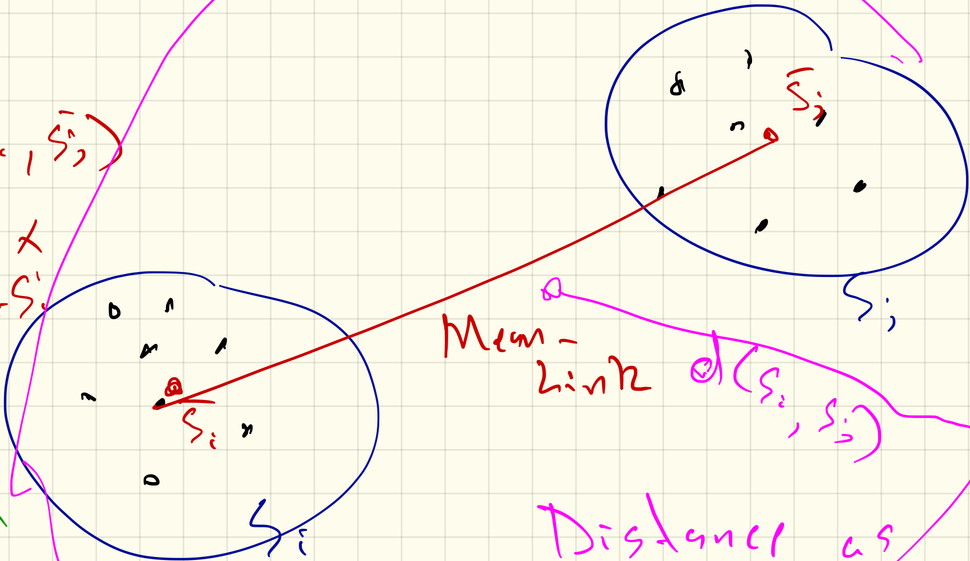
Mean-Link

$$d(S_i, S_j) = d(\bar{S}_i, \bar{S}_j)$$

$$\bar{S}_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

replace "centroid"

- must be data point
- random point $\in S_i$

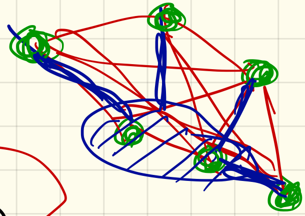


Mean-Link $d(S_i, S_j)$

Distance as size

$d(S_i, S_j) =$ radius minimum ball contains $S_i \cup S_j$

Efficiency HAC



- Data is large: $n = |X|$ big
- Distance is not Euclidean in $\mathbb{R}^2$
- $(n-1) = O(n)$ Rounds (merge S_i, S_j)
  - Find closest pair

first half of merges $(n/2)$
at least $n/2$ pairs
↳ check $\binom{n/2}{2} = O(n^2)$
↳ linear $\boxed{O(n)}$

only update distance
last merged cluster

$O(n \cdot n \log n)$
$\boxed{= O(n^2 \log n)}$ slow

$\times\ O(n^2) \leftarrow \max / \min \rightarrow O(n)$

other algo

↳ $O(nk)$
↑ # clusters
much faster

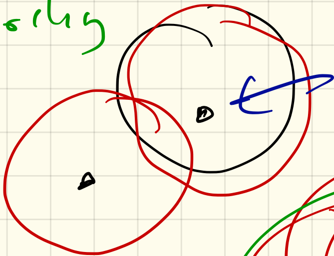
DB-Scan

density-based

$X \subset \mathbb{R}^d$

$d = \| \cdot - \cdot \|$ Euclidean

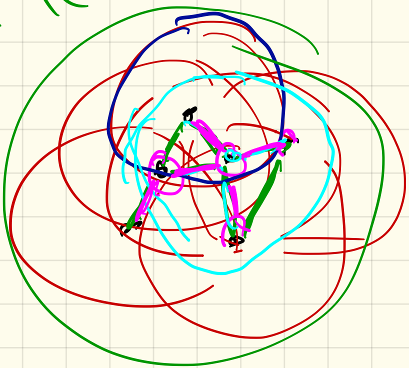
- like single-link
- no hierarchy
- faster



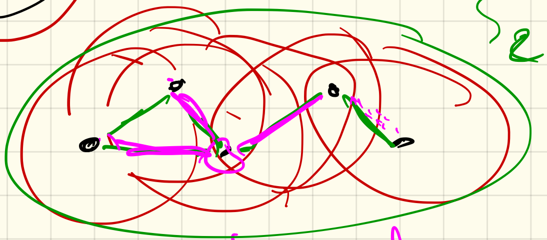
not in clusters

LSH

$\hookrightarrow O(n \cdot T(LSH(n)))$



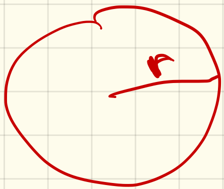
S_1



S_2

core pts = degree ≥ 3

"minPts"



$(r = \varepsilon)$