Hierarchical Agglomerative Clustering

February 5, 2018

When data is easily "clusterable", most clustering algorithms work quickly and well.

When data is *not* easily "clusterable", then *no algorithm* will find good clusters

What is clustering?

Input Dataset $X = \{x_1, x_2, \dots x_n\}$

distance $d : X \times X \to \mathbb{R}$

input

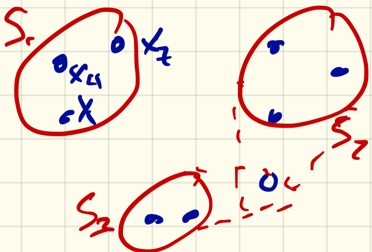
$X \subset M = \mathbb{R}^d \qquad d : M \times M \to \mathbb{R}$

Output $C(X) = \{S_1, S_2 \dots S_k\}$

(1) $S_i \subset X$ ← ith "cluster"

(2) $S_i \cap S_j = \phi \quad i \neq j$ "hard clustering" vs. 'soft'

(3) $\bigcup_i S_i = S_1 \cup S_2 \dots \cup S_k = X$



Usually some objective function

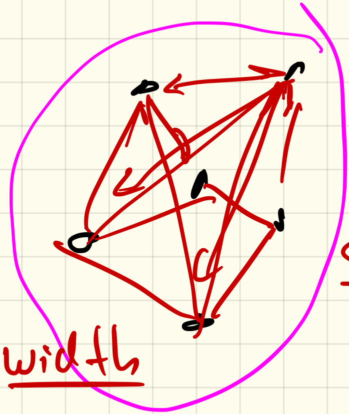
Clustering Objective

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

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

Euclidean

large : $\frac{\text{split}}{\text{width}}$ or split - width



S_ℓ

width

$d(x_i, x_j)$ for $x_i, x_j \in S_\ell$

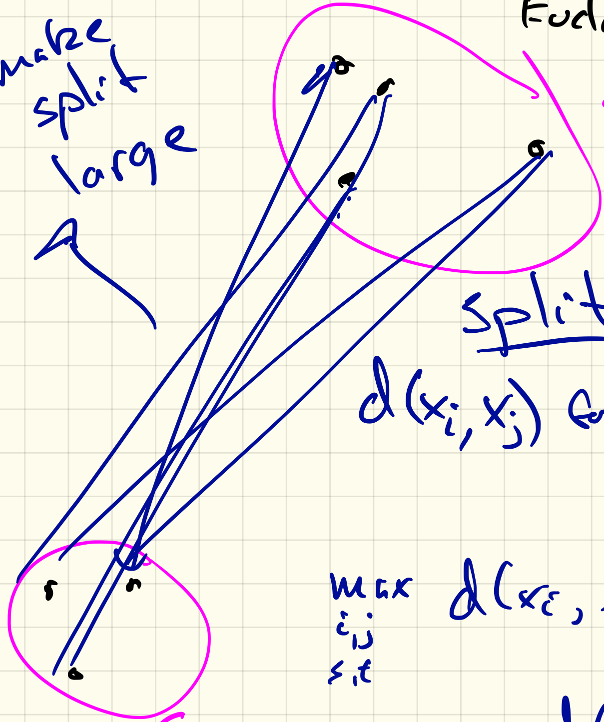
$\sum_{i,j} d(x_i, x_j)$

$\max d(x_i, x_j)$

$\sum_i d(x_i, x_j)^2$

$\Rightarrow$ make width small

make split large



S_s

S_t

split

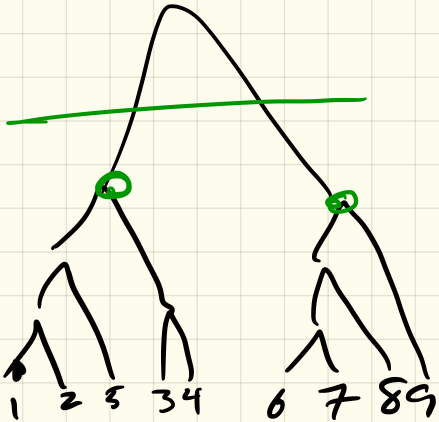
$d(x_i, x_j)$ for $x_i \in S_s$, $x_j \in S_t$

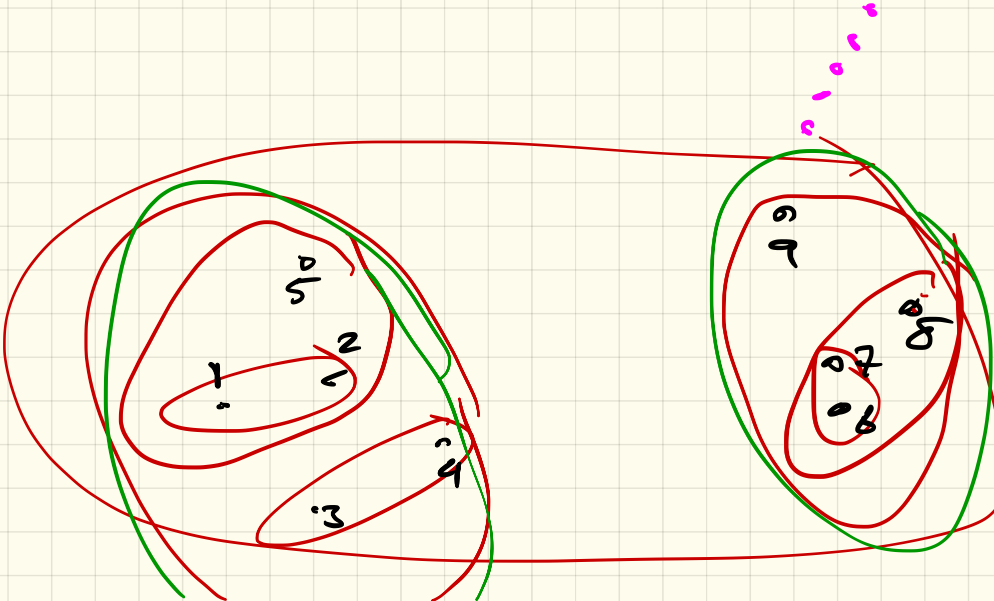
$\max_{i,j,s,t} d(x_i, x_j)$

$\text{sum}_{i,j} \max_{s,t} d(x_i, x_j)^2$

Hierarchical Agglomerative Clustering

Algo: If 2 points (clusters) are close enough, put them in same cluster.



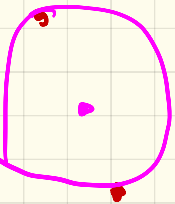


0. Each $x_i \in X$ a separate cluster S_i $\leftarrow$ n clusters

1. while (2 clusters are close enough)
   - 1a. Find closest two clusters S_i, S_j
   - 1b. Merge $S_i, S_j \Rightarrow$ new cluster S_k

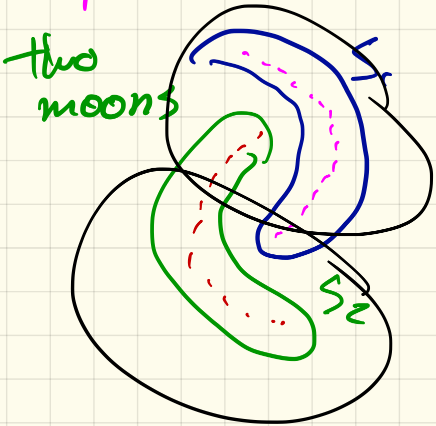
How to define $d(S_i, S_j)$

- Define center $c_i \leftarrow S_i$, $c_j \leftarrow S_j$ $\quad d(c_i, c_j)$
  - + geometric median: $c_i = \underset{c \in M}{\arg\min} \sum_{x \in S_i} d(x, c)$
  - + mean $c_i = \underset{c \in \mathbb{R}^d}{\arg\min} \sum_{x \in S_i} \|x - c\|^2 = \frac{1}{|S_i|} \sum_{x \in S_i} x$
  - + restrict some $c \in S_i$
  - + center of minimum enclosing ball



- **Single Link**: $d(S_i, S_j) = \min_{x \in S_i,\ x' \in S_j} d(x, x')$

  closest dist between pts in clusters



- **Average Link**

$$d(S_i, S_j) = \frac{1}{Z} \sum_{x \in S_i}' \sum_{x' \in S_j}' d(x, x')$$

- **Furthest Link**

$$d(S_i, S_j) = \max_{x \in S_i\ x' \in S_j} d(x, x')$$

- Define goodness of fit of model M

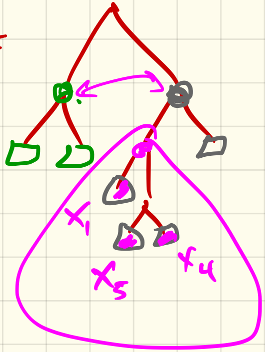
$$g(M, S) > \text{threshold} = 10 \qquad \text{cluster's}$$

$$d(S_i, S_j) = -g(M, S_i \cup S_j) \qquad M = f(S)$$

Dasgupta 2016

similarity $s_{ij} = S(x_i, x_j)$

$H =$



$H(1 \wedge 4)$

$$\mathrm{cost}(H) = \sum_{x_i, x_j \in X} s_{ij} \left[\#_{\text{leaves}} H(i \wedge j) \right]$$

$H(i \wedge j)$ = smallest subtree containing x_i, x_j

$$\# H(1 \wedge 4) = 3 = |\{x_1, x_4, x_5\}|$$

goal: Find H minimize $\mathrm{cost}(H)$

Efficiency

$O(n^3)$ time

loop $O(n)$

1. Find closest pair $\overbrace{\boxed{(s_i, s_j)}}^{O(n^2)}$

2. Merge $(s_i, s_j) \to s_i$

check $O(n^2)$ pairs i, j

$d(s_i, s_j)$ often update $O(1)$ time, is $O(n)$ time

$d(s_i, s_j), d(s_i, s_e), d(s_j, s_e)$

Merge $s_i, s_j \to s_m$

$d(s_m, s_e) = \min(d(s_e, s_i), d(s_e, s_j))$

still SLOW

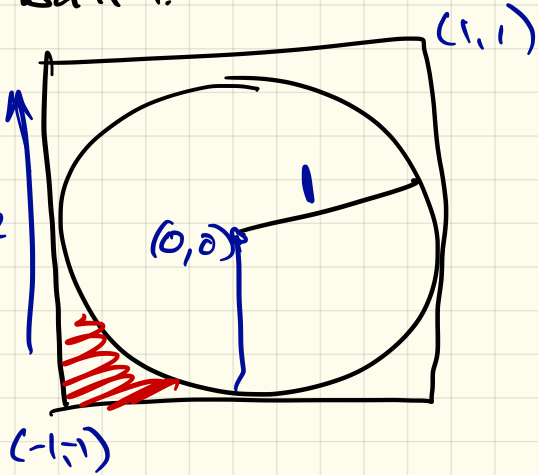
often w/ priority queue $\to O(n^2 \log n)$

Why k-d trees not worth in high dim?

battle: boxes vs. balls.

$$\mathrm{Vol}(\mathrm{Ball}(d)) = \frac{\pi^{d/2}}{\Gamma(d/2+1)} \approx \frac{\pi^{d/2}}{(d/2)!}$$

$d \to \infty \to 0$ (less than 1)^2
$d > 5$



$$\mathrm{Vol}(\mathrm{Box}(d)) = 2^d$$

$d \to \infty \to$ vol exponentially to ∞