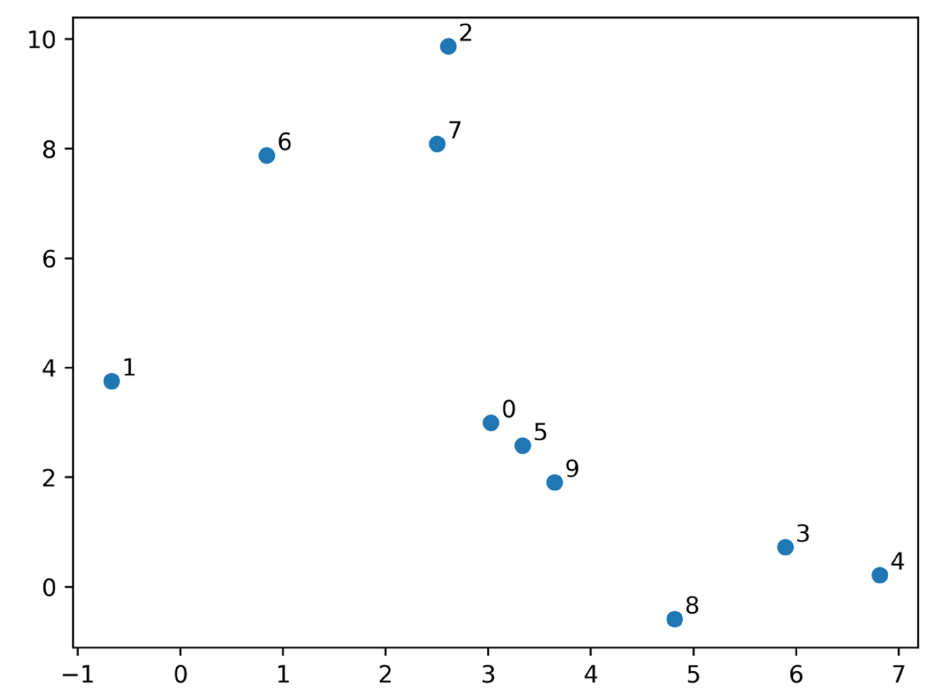
Improving Ultrametrics Embeddings Through Coresets

Vincent Cohen-Addad*, Rémi de Joannis de Verclos†, Guillaume Lagarde‡

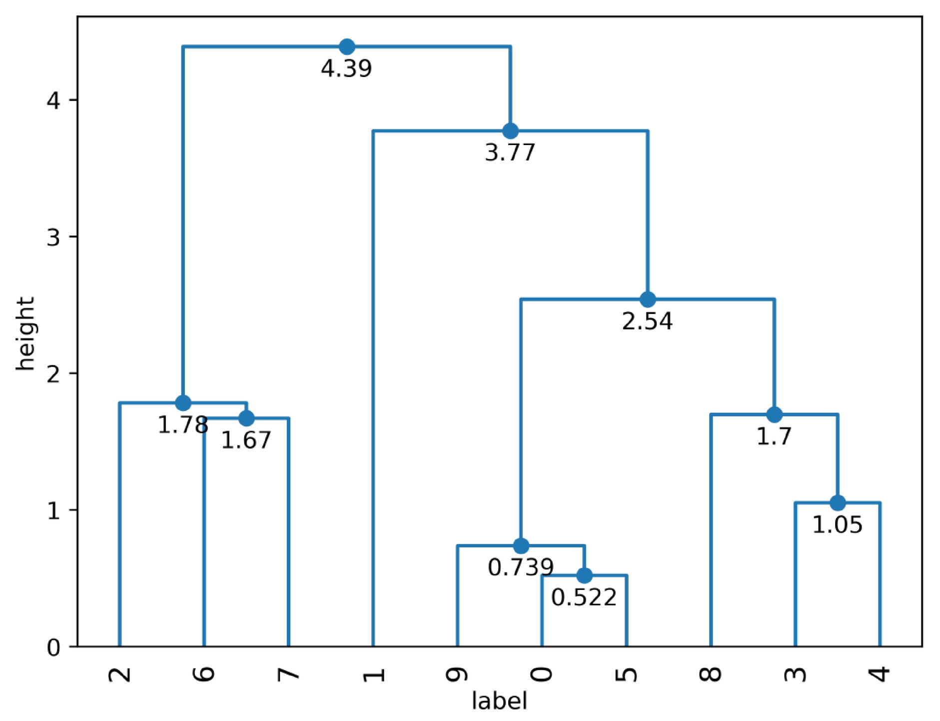
*Google, Switzerland †Independant researcher, Netherlands ‡LaBRI & CNRS, France

Problem Definition

Input: A set of points $S \subset \mathbb{R}^d$



Output: An ultrametric (hierchical clustering)



Ultrametrics: A way to formalize the notion of "hierchical clustering". It is a distance function satisfying the **ultrametric inequality**:

$$\Delta(x, y) \leq \max(\Delta(x, z), \Delta(z, y))$$

The tree and the heights induce such a distance function:

$$\Delta(x, y) = \text{height}(\text{LCA}(x, y))$$

Goal: Try to preserve as much as possible the distances by **minimizing** the **distortion**:

$$\max_{x,y} \frac{\Delta(x, y)}{\|x - y\|_2}$$

History & related algorithms

Linkage algorithms(single, complete, Ward, ...):

- $\Omega(n^2)$ running time in the best case
- $\Omega(n^2)$ memory in most cases
- unclear what loss functions they are optimizing

→ **Not well designed to handle large datasets** ($\geq 10^5$ points)

Minimizing the max distortion:

- Optimal solution in $\Theta(n^2)$ together with an $\Omega(n^2)$ lower bound [Farach, Kannan, Warnow'95]
- $5 \cdot \gamma$-approximation in time $\approx n^{1+1/\gamma^2+o(1)}$ and approximation lower bounds [Cohen-Addad, Karthik, Lagarde'20]

Our Result

For any $\gamma \geq 1, \epsilon > 0$, there is an algorithm that produces a $(\sqrt{2} + \epsilon) \cdot \gamma$-approximation for BUF_∞ in time

$$n \cdot \left(n^{1/\gamma^2 + o(1)} + d \cdot \left(\frac{\log 1/\varepsilon}{\varepsilon^2} + \frac{\log n}{\varepsilon}\right) + \frac{\log 1/\varepsilon}{\varepsilon^{4.5}} \right)$$
$$\approx n^{1+\frac{1}{\gamma^2}+o(1)}$$

Main task

Implement efficiently a data structure similar to union-find with three primitives:

1. Union(S_1, S_2), merges S_1 and S_2 into a single set
2. Find(S), gives a representative member of S
3. Cut-weight$(S_1, S_2) = \max_{x \in S_1, y \in S_2} \|x - y\|_2$

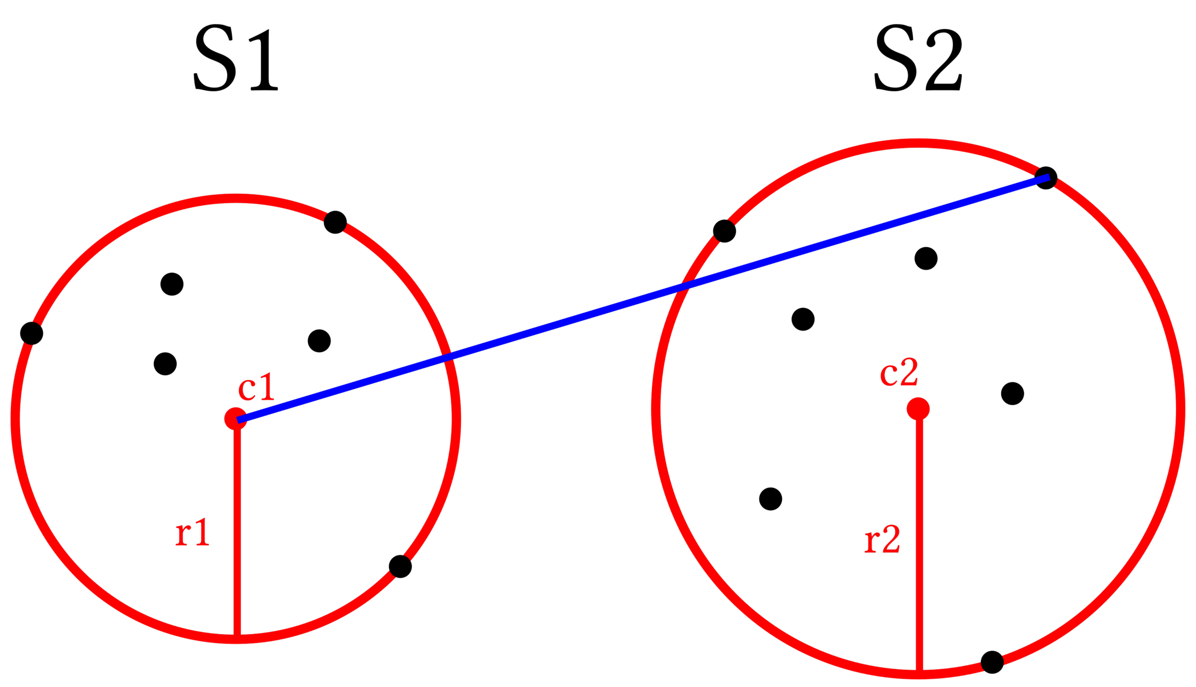
How: Coresets!

Coreset: small set of points of a larger set for which the minimum enclosing balls are nearly preserved

- There exists coresets of size $O(1/\varepsilon)$, and this quantity depends neither on the number of points nor on the space dimension! [Badoiu, Clarkson'03]
- Easy and fast to compute

Approximating the cut weights

Main idea: use coresets to keep track of approximate minimum enclosing balls [Kumar, Mitchell, Yildirim'03] for every sets and use this information to approximate the cut weights



Observation

If the minimum enclosing balls are exact, then $r_1 + \max_{z \in S_2} \|z - c_1\|_2$ is a $\sqrt{2}$-approximation of the cut weight between S_1 and S_2

Experimental results

The algorithm was implemented in `Cython`

	$N = 10^5, d = 100$	$N = 10^6, d = 50$
CKL	4.9s	42s
CoreSet	3.8s	58s
Ward (fastcluster)	2141s	$\geq 10h$
Single (fastcluster)	842s	$\geq 10h$
Centroid (fastcluster)	1364s	$\geq 10h$

	MICE		PENDIGITS		SHUTTLE	
	dist	$T(s)$	*dist*	$T(s)$	*dist*	$T(s)$
CoreSet	15.13	0.056	27.26	0.41	106.8	5.44
CKL	38.20	0.022	82.58	0.25	379.9	4.11
Ward (fastcluster)	433.78	0.074	7311	1.83	7311	25.06
Single (fastcluster)	4.92	0.045	13.86	0.94	29.9	26.4
Centroid (fastcluster)	8.98	0.083	33.09	1.82	183	30.2
Average (scikit-learn)	9.7	0.037	27.52	4.70	-	