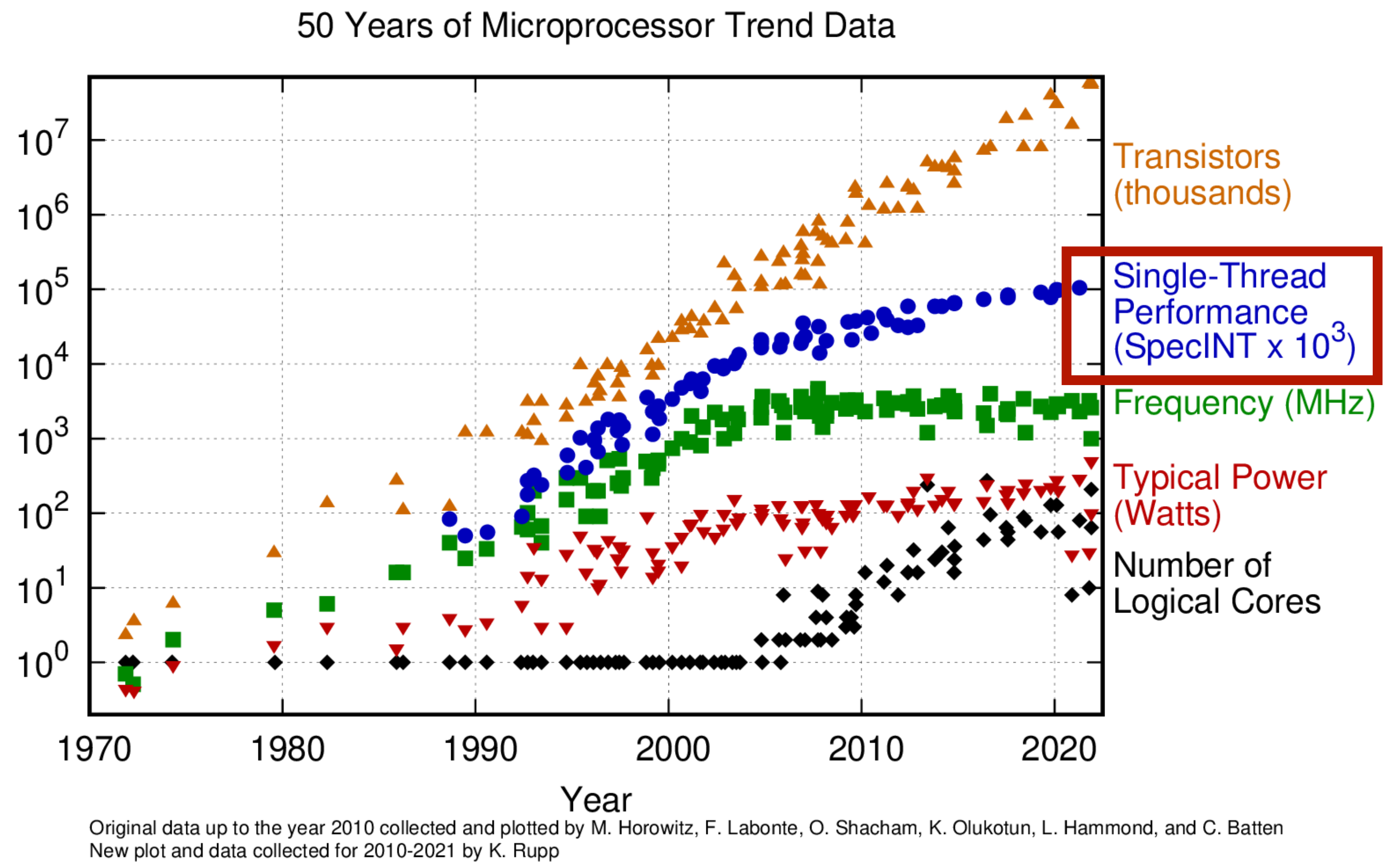
Recent Advances in Coresets for Clustering

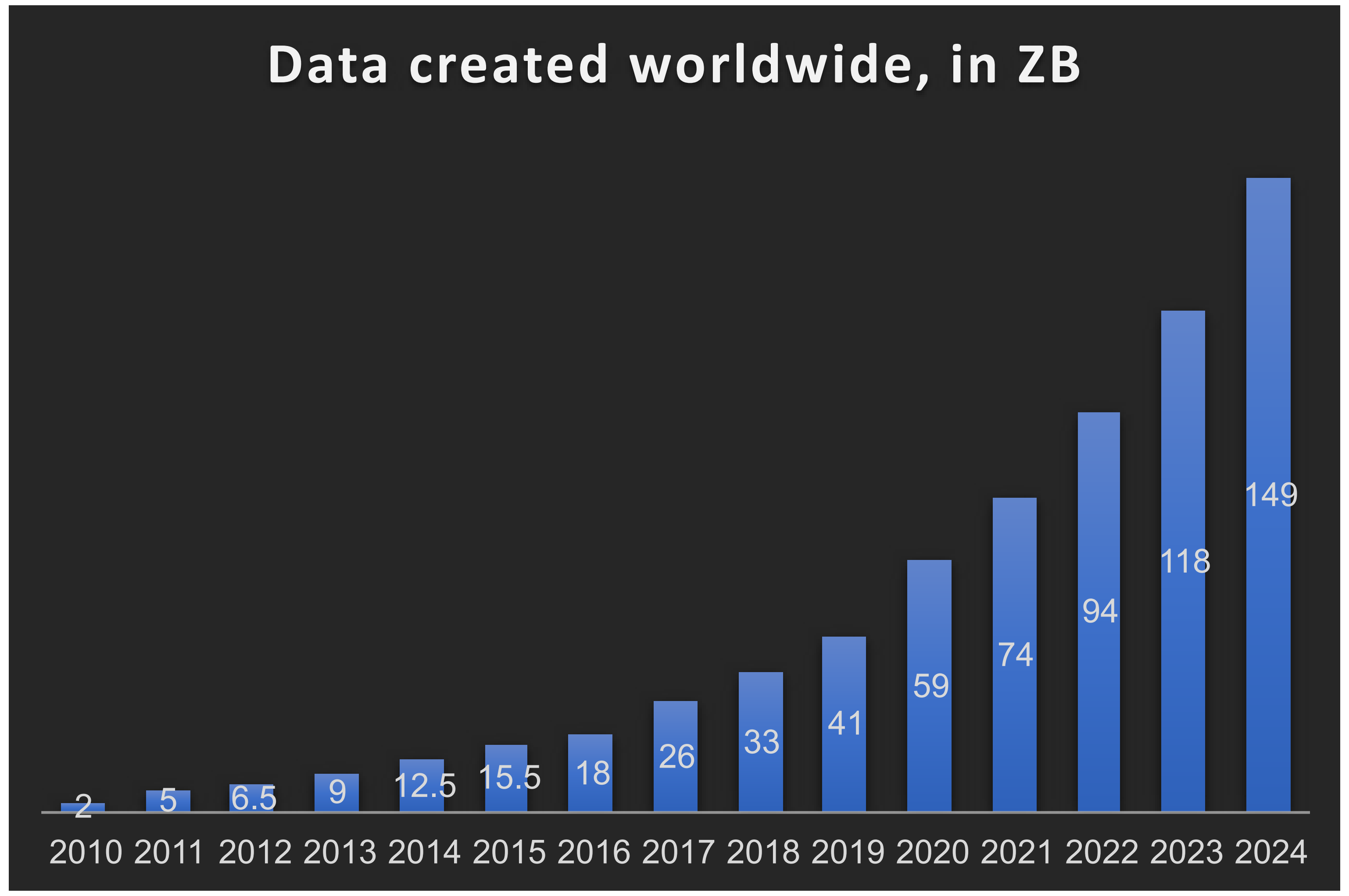
Shaofeng Jiang

北京大学前沿计算研究中心
Center on Frontiers of Computing Studies, Peking University

Sublinear Algorithms

Computational challenge of big data: even linear time/space doesn't work!





Typical sublinear models: streaming, distributed computing, sublinear time

o(n) space

o(n) communication

o(n) query

Coreset: A Data Reduction Method

For sublinear algorithm design

A problem $\mathscr{P}$ defined on big data

Sublinear/efficient algorithm

Coreset: Tiny proxy of dataset

Solve only on the tiny proxy

Classic Alg. $\mathscr{A}$

	Model 1	Model 2	...
Algorithm 1	✓	✗	
Algorithm 2	✗	✓	
⋮			

Algorithm driven design of sublinear algorithms

Features:

- Data/problem driven design of sublinear algorithms
- Existing (non-big-data) algorithms can be readily applied

Clustering

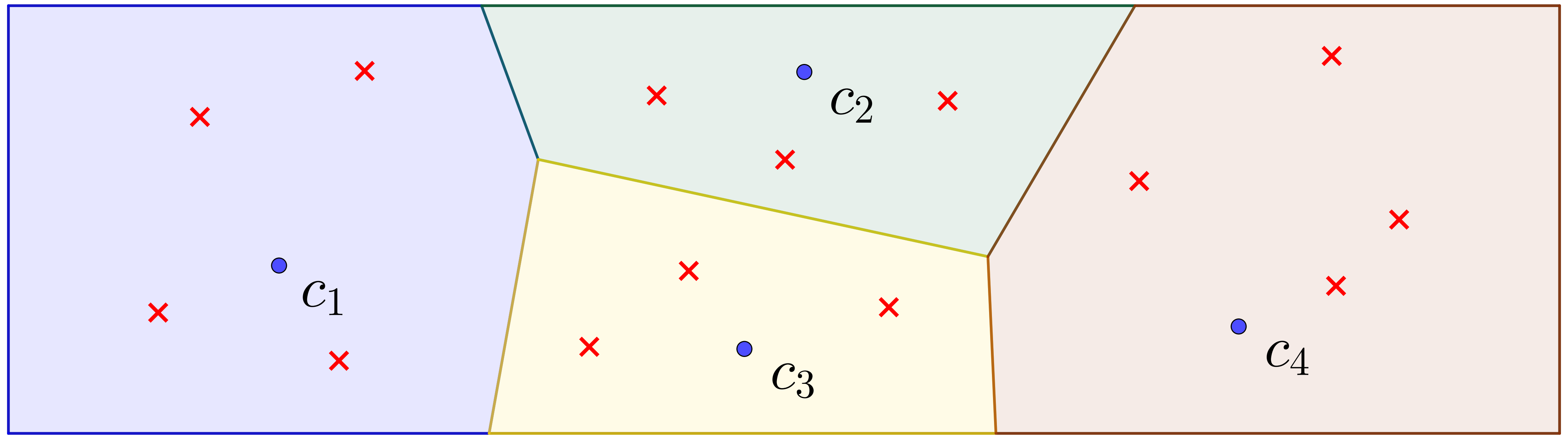
k-median: dataset $D \subset \mathbb{R}^d$, find center set $C \subset \mathbb{R}^d$ s.t. $|C| \leq k$ to minimize

$$\mathrm{cost}(D, C) := \sum_{x \in D} \mathrm{dist}(x, C)$$

$\mathrm{dist}(x, C) := \min_{c \in C} \mathrm{dist}(x, c)$, $\mathrm{dist} = \ell_2$

Related problem: k-means, $\mathrm{cost}(D, C) := \sum_{x \in D} \mathrm{dist}^2(x, C)$

Notice the square



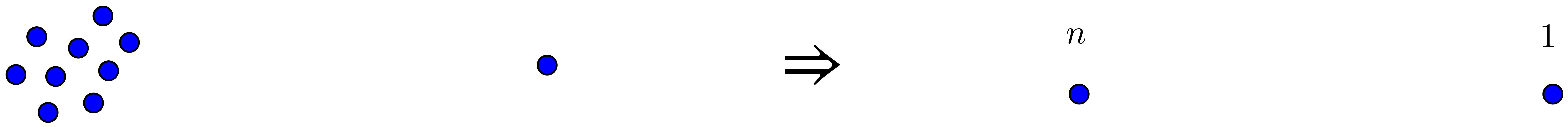
Coreset for Clustering

ϵ-Coreset is a weighted subset $S \subseteq D$ s.t. [Har-Peled-Mazumdar, STOC 04]

$$\forall C \subset \mathbb{R}^d, |C| \leq k \qquad \text{cost}(S, C) \in (1 \pm \epsilon) \cdot \text{cost}(D, C)$$

There can be infinitely many such C's!

Why weighted?



Performance measure: # of distinct elements $|S|$

Coreset -> Sublinear Algorithms

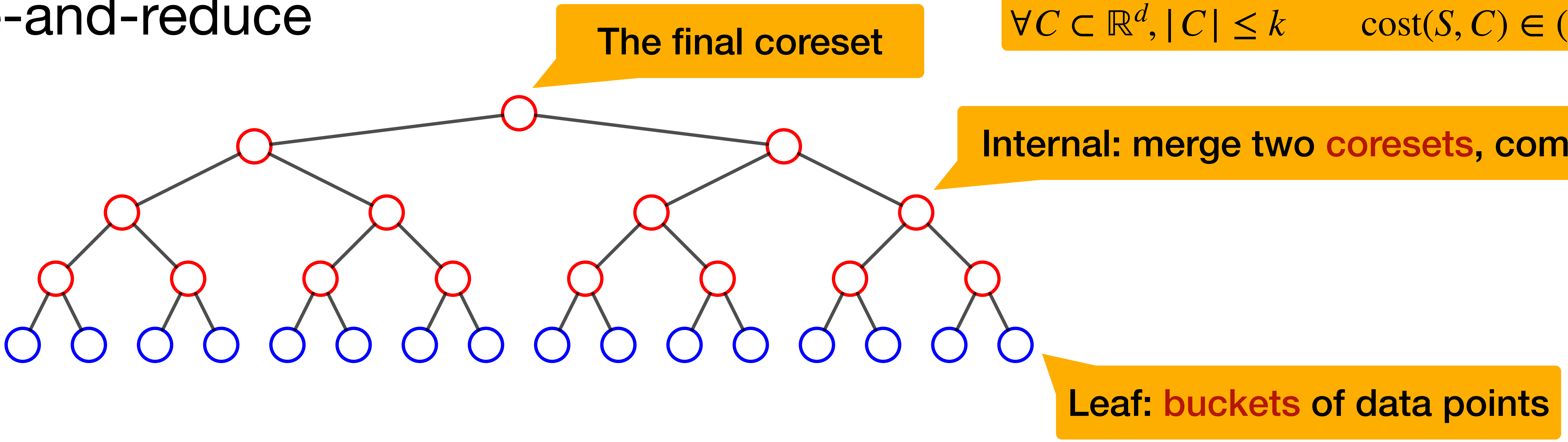
Merge-and-reduce method

Given ϵ-coreset Alg. $\mathscr{A}$, one can turn $\mathscr{A}$ into sublinear algorithms, e.g., streaming/distributed/dynamic algorithms, in a black-box way!

- Key property — composable: coreset(X) ∪ coreset(Y) is a coreset$(X \cup Y)$
- Merge-and-reduce

Follows from definition:

$\forall C \subset \mathbb{R}^d, |C| \leq k \quad \mathrm{cost}(S, C) \in (1 \pm \epsilon) \cdot \mathrm{cost}(D, C)$



Results

Size **independent of** n

Most studied: vanilla k-clustering in $\mathbb{R}^d$

Obtaining tight degree of poly is still open

- Upper bound (for k-median): $O(\min\{k^{4/3}\epsilon^{-2}, k\epsilon^{-3}, k\epsilon^{-2}d\})$
- Lower bound: $\Omega(k\epsilon^{-2})$

There's an even larger gap in the degree of poly

Extensions: size $\mathrm{poly}(k\epsilon^{-1})$

- Other metric space: doubling metrics, planar graphs etc.
- Variants: fair clustering, capacitated clustering, clustering w/ outliers etc.

Natural Idea: Sampling

Uniform sampling? Doesn’t work:

Needs to do non-uniform sampling

- Generic framework: sensitivity sampling
- More specific to clustering: hierarchical uniform sampling

Sensitivity Sampling Method

Warmup: Importance Sampling

Suppose $a_1, \ldots, a_n > 0$

Want to estimate $\sum_i a_i$, but can access a_i only through random samples

Question: How well does uniform sampling work?

- Bad example: $a_1 = 1$, but for $i > 1$, $a_i = 0$

requires $\Omega(n)$ samples to see a_1 even once

Importance Sampling

Algorithm

For some $0 < \lambda \leq 1$, suppose we have a distribution on ID $j \in [n]$ s.t.

$$p_j \geq \lambda \cdot a_j / \sum_i a_i$$

$\sigma_j := a_j / \sum_i a_i$ is called "importance score"

Low variance!

Claim: Let $\hat{Z} := a_j/p_j$. Then $\mathbb{E}[\hat{Z}] = \sum_i a_i$, $\mathrm{Var}[\hat{Z}] \leq O(\lambda^{-1}) \cdot \mathbb{E}^2[\hat{Z}]$

Unbiased

Hence, aggregate $O(1/\epsilon^2)$ i.i.d. samples yields $(1+\epsilon)$-approximation

Proof

Let $W := \sum_i a_i$. Recall $p_j \geq \lambda \cdot a_j / W$, $\widehat{Z} := a_j / p_j$

$$\mathbb{E}[\widehat{Z}] = \sum_i p_i \cdot a_i / p_i = \sum_i a_i = W$$

$$\mathbb{E}(\widehat{Z}^2) = \sum_i p_i \cdot (a_i / p_i)^2 = \sum_i a_i^2 / p_i \leq \lambda^{-1} W \sum_i a_i = \lambda^{-1} W^2$$

$$\mathrm{Var}(\widehat{Z}) = \mathbb{E}[\widehat{Z}^2] - \mathbb{E}^2[\widehat{Z}] \leq O(\lambda^{-1}) \cdot \mathbb{E}^2[\widehat{Z}]$$

Generalization: Sensitivity Sampling

Our case: for $x \in D$, let $f_x(C) := \text{dist}(x, C)$, then $\text{cost}(D, C) = \sum_{x \in D} f_x(C)$

Interpretation: sum of functions $\{f_x\}_{x \in D}$ on the same variable C

Exactly a coreset!

Goal: draw a sample of D that approximates this sum for all C simutaneously

Compare to importance samp.: sum of numbers vs sum of functions

Sensitivity Sampling

Sensitivity σ_x: analogue to importance score

$$\text{For } x \in D, \sigma_x := \sup_{C \subset \mathbb{R}^d, |C| \le k} \frac{f_x(C)}{\mathrm{cost}(D, C)}$$

The contribution of x over any possible center set
(i.e., parameter of f_x)

Claim:

Given $p_x \ge \lambda \cdot \sigma_x$, sample $x \in D$ w.p. p_x, set its weight $w(x) := 1/p_x$

Then $\forall C, \mathbb{E}[f_x(C)] = \mathrm{cost}(D, C)$ and $\mathrm{Var}[f_x(C)] \le O(\lambda^{-1} \sum_x \sigma_x) \cdot \mathbb{E}^2[f_x(C)]$

$\sum_x \sigma_x$ is called "total sensitivity"

Sensitivity Sampling

Hence: If λ and $\sum_x \sigma_x$ bounded, let S be $O(\epsilon^{-2} \log 1/\delta)$ i.i.d. samples, then

$$\forall C, \qquad \Pr[\mathrm{cost}(S, C) \in (1 \pm \epsilon) \cdot \mathrm{cost}(D, C)] \geq 1 - \delta$$

Notice: only for one C

To make it a coreset, one still needs a union bound on all C

- But C is infinitely many, even in 1D and $k = 1$ (i.e., 1-median on real line)!
- We need “clever” discretization: Sauer-Shelah-like, via VC-dimension

VC/Shattering Dimension

For $\mathbb{R}^d$, one can show that sdim is $O(d)$

Consider metric space $\mathcal{M}(V, \mathrm{dist})$

For $x \in V$, define a metric ball $B(x, r) := \{y \in V : \mathrm{dist}(x, y) \leq r\}$

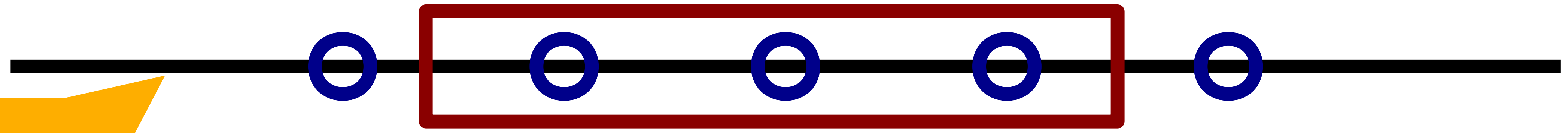
Shattering dimension, denoted as $\mathrm{sdim}(\mathcal{M})$:

Measure the complexity of $\mathcal{M}$'s metric balls

- Smallest integer t, s.t. for every $H \subseteq V$ with $|H| \geq 2$

Up to log factor to VC-dim of metric ball system

$$|\{B(x, r) \cap H : x \in V, r \geq 0\}| \leq |H|^t$$



In 1D, a ball is an interval;
m points can form $O(m^2)$ intervals, so $t = 2$

Conclusion: Coresets via Sensitivity Samp.

There's an efficient way to compute such p_x's with $\lambda = \Omega(1)$

Sensitivity sampling: Given $p_x \geq \lambda \cdot \sigma_x$

Sample $x \in D$ w.p. p_x, set its weight by $w(x) := 1/p_x$

Theorem: $\mathrm{poly}(\epsilon^{-1} \cdot \sum \sigma_x \cdot \mathrm{sdim})$ i.i.d. sensitivity samples is ϵ-coreset w.h.p.

[Feldman-Langberg, STOC 11]

For k-clustering, total sensitivity is $O(k)$
[Varadarajan-Xiao, FSTTCS 12]

Corollary: $O(kd\epsilon^{-2})$ i.i.d. sensitivity samples is ϵ-coreset for k-median in $\mathbb{R}^d$

[Feldman-Langberg, STOC 11]

Other Metrics

For clustering: given metric $\mathscr{M}(V, \mathrm{dist})$, we allow dataset $D \subseteq V$, center set $C \subseteq V$

For metrics other than $\mathbb{R}^d$, $\mathrm{poly}(k\epsilon^{-1})$ size coreset exists if sdim is bounded

- Doubling metrics [Huang-J-Li-Wu, FOCS 18]
- The shortest-path metric of graphs
  - planar/excluded-minor [Bousquet-Thomassé, Discret. Math. 15] [Braverman-J-Krauthgamer-Wu, SODA 21]
  - bounded treewidth [Baker-Braverman-Huang-J-Krauthgamer-Wu, ICML 20]
- Polygonal curves under Fréchet distance

[Braverman-Cohen-Addad-J-Krauthgamer-Schwiegelshohn-Toftrup-Wu, FOCS 22]

How to Remove Dependence on d for $\mathbb{R}^d$?

Simple approach: iterative size reduction

Informal argument:

Need a terminal embedding version of JL [Narayanan-Nelson, STOC 19]

- First do JL: reduce to $d = \log n$, leading to a coreset of size $O(\log n)$
- Iteratively running this, we have $n \to \log n \to \log \log n \ldots$
- See [Braverman-J-Krauthgamer-Wu, SODA 21]

Run for $\log^* n$ times, error can accumulate

To avoid $\log^* n$ in error bound, one needs to set ϵ carefully in each iteration

* Note: first dimension-independent results were obtained in **[Sohler-Woodruff, FOCS 18; Feldman-Schmidt-Sohler, SICOMP 20]**

Good and Bad of Sensitivity Sampling

Suitable for various problems (non-exhaustive examples):

- Projective clustering/missing value [Feldman-Schmidt-Sohler, SICOMP 20; Braverman-J-Krauthgamer-Wu, NeurIPS 21]
- Gaussian mixture model [Lucic-Faulkner-Krause-Feldman, JMLR 17]
- Logistic regression [Munteanu-Schwiegelshohn-Sohler-Woodruff, NeurIPS 18]
- Decision tree [Jubran-Shayda-Newman-Feldman, NeurIPS 21]

What's not so good:

For example capacity constraints

- Not effective to deal with constraints; sub-optimal size

More structured sampling can do better

Hierarchical Uniform Sampling Method

Hierarchical Uniform Sampling

[Chen, SICOMP 09]

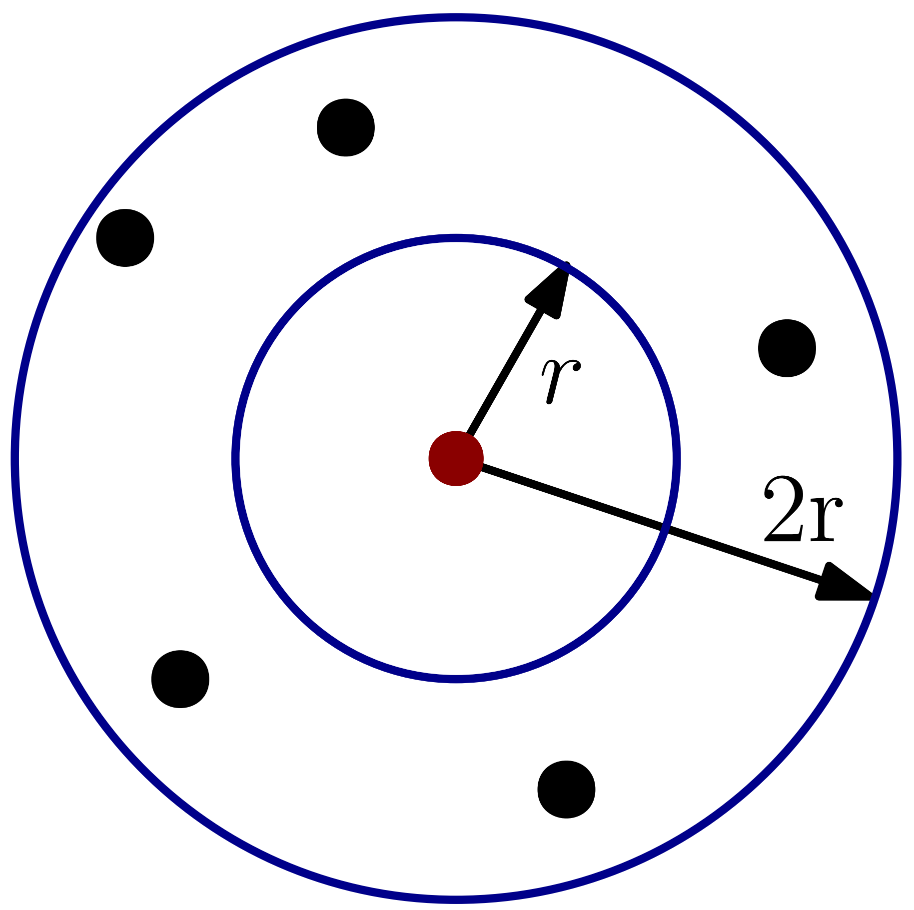
A more geometric way to construct coreset

$\mathrm{ring}(c, r, 2r) := B(c, 2r) \backslash B(c, r)$

First, consider ring dataset $R \subseteq \mathrm{ring}(c, r, 2r)$

Intuition: points in the ring have similar "importance scores"

- So uniform sampling should work



Uniform Sampling on Ring Dataset

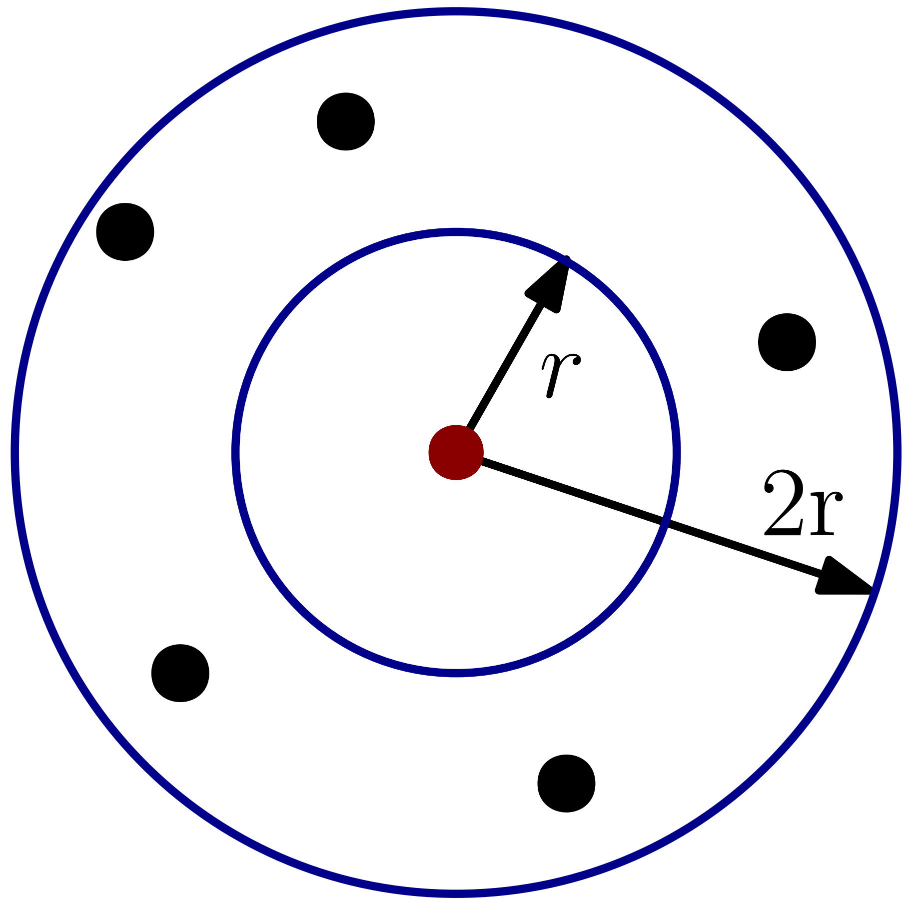
n_R is the number of points in R

Draw m uniform samples $S \subseteq R$, set $w(x) := n_R/m$ for $x \in S$

Unbiased: $\mathbb{E}[\mathrm{cost}(S, C)] = \mathrm{cost}(R, C)$

Hoeffding inequality implies w.h.p., $|\mathrm{cost}(S, C) - \mathrm{cost}(D, C)| \leq \epsilon n_R \cdot r$

Bounded terms: $\forall x, y \in D$,
$\mathrm{dist}(x, C) - \mathrm{dist}(y, C) \leq \mathrm{dist}(x, y) \leq O(r)$



Is the Addive Error $\epsilon n_R r$ Good?

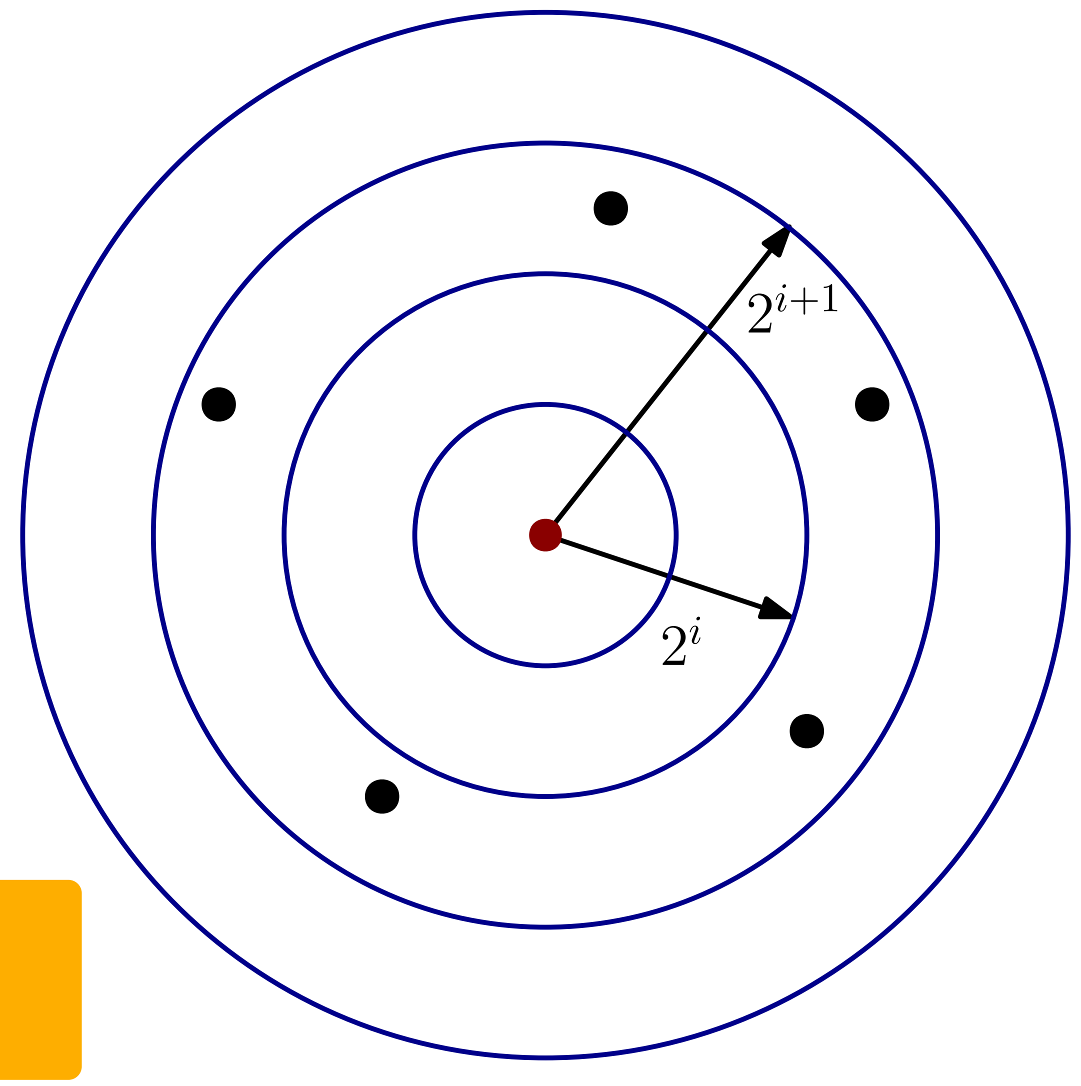
Charging $\epsilon n_R r$ to OPT, via ring decomposition

$(O(1), O(1))$-bicriteria solution also works!

Find optimal center set $C^* = \{c_1^*, \ldots, c_k^*\}$

Partition/clustering the dataset D w.r.t. C^*

For each cluster C_i^*, partition into rings of radius $r = 2^i$



For each ring R of radius r:

- Each $x \in R$ contributes $O(r)$ to OPT
- In total contribute $O(n_R r)$ since the ring has n_R points

So $\epsilon n_R r$ is ϵ to OPT!

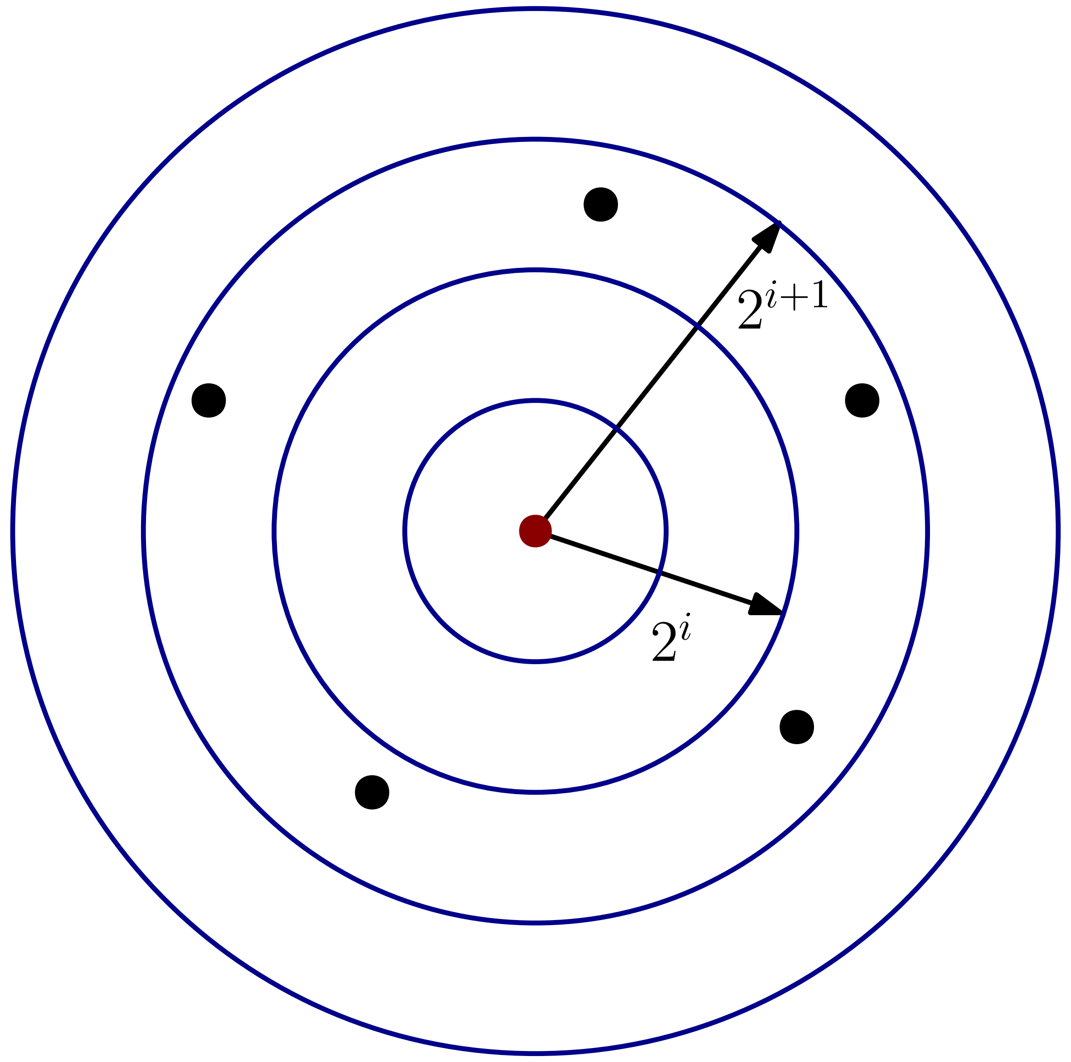
Further development

which translates to $O(\log n)$-size coreset

Naive decomposition may introduce $O(\log n)$ rings

Improved way: group several rings together, and create only $\log 1/\epsilon$ rings

- Lead to state-of-the-art coreset size [Cohen-Addad-Saulpic-Schwiegelshohn, STOC 21; Cohen-Addad-Larsen-Saulpic-Schwiegelshohn, STOC 22; Cohen-Addad-Larsen-Saulpic-Schwiegelshohn-Sheikh-Omar, NeurIPS 22]
- Also extends to constrained clustering
  - Fair clustering, capacitated clustering etc. [Braverman-Cohen-Addad-J-Krauthgamer-Schwiegelshohn-Toftrup-Wu, FOCS 22]
  - Clustering with outliers [Huang-J-Lou-Wu, ICLR 23]

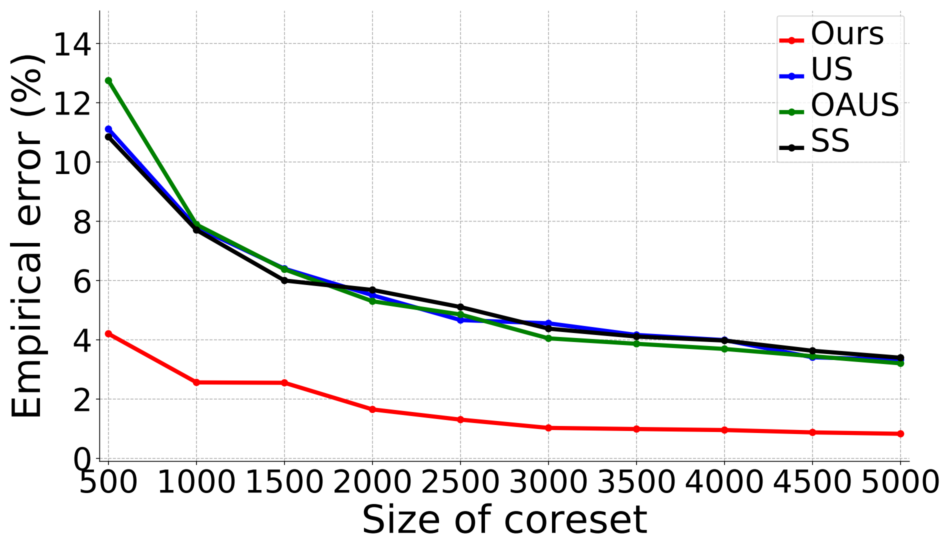


Some Experiment Results

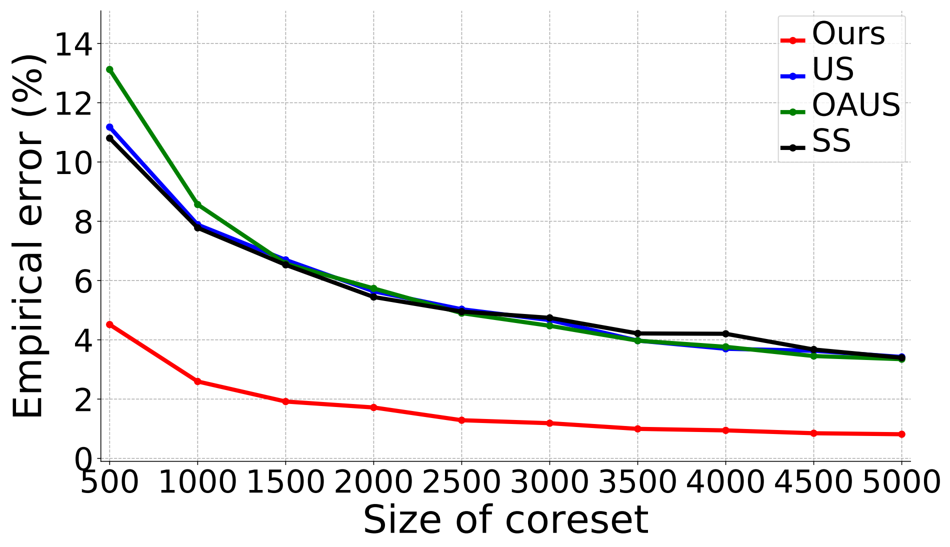
Coresets for clustering with outliers [Huang-J-Lou-Wu, ICLR 23]

SS = sensitivity sampling

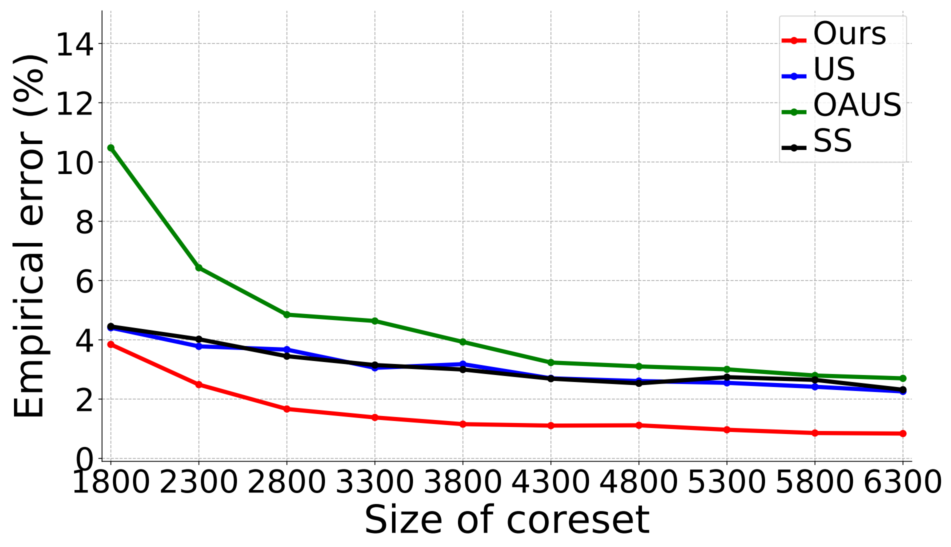
- Based on hierarchical uniform sampling; works better than SS in practice



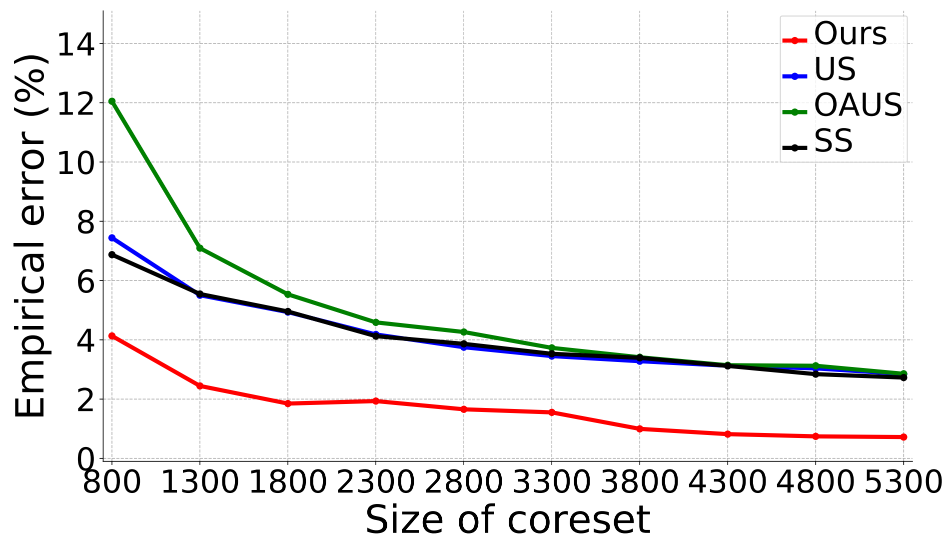
n = 50000, d = 6



n = 40000, d = 10



n = 20000000, d = 2



n = 2000000, d = 68

We also observe similar results in many other coreset papers

Speed up Approximation Algorithms

Table 2: Running time and costs for LL and LS with/without coresets. T_X and T_S are the running time without/with the coreset, respectively. Similarly, cost and cost' are the clustering costs without/with the coreset. T_C is coreset construction time. This entire experiment is repeated 10 times and the average is reported.

dataset	algorithm	cost	cost′	T_C (s)	T_S (s)	T_X (s)
Adult	LL	3.790×10^{13}	3.922×10^{13}	0.4657	0.06385	16.51
	LS	1.100×10^{9}	1.107×10^{9}	0.5300	1.147	204.8
Bank	LL	4.444×10^{8}	4.652×10^{8}	0.4399	0.05900	11.40
	LS	4.717×10^{6}	4.721×10^{6}	0.4953	1.220	186.6
Twitter	LL	3.218×10^{7}	3.236×10^{7}	0.9493	0.08289	11.27
	LS	1.476×10^{6}	1.451×10^{6}	1.064	2.135	460.2
Census1990	LL	1.189×10^{7}	1.208×10^{7}	3.673	0.4809	40.54
	LS	1.165×10^{6}	1.163×10^{6}	4.079	24.83	2405

We also observe similar results in many other coreset papers

Conclusion

Importance sampling

- Wider applicability, but may not be the end-game solution for clustering

Hierarchical uniform sampling

- Simpler, better suited (but very specific) to clustering
- Can handle constrained clustering

Future Directions

Coresets for clustering: tight bounds, i.e., tight degree of poly of ϵ, k

Beyond coreset/what's coreset cannot do for clustering:

- Size lower bound of $\Omega(k)$ for coreset — Severe limitation when k is large!
- Streaming and MPC algorithms that have $o(k)$ space usage?

A popular distributed computing model motivated by MapReduce

Beyond clustering:

- Coreset/sampling x other tasks in ML?

Thanks!