Contents

Coreset

Shuangcheng Liu

January 19, 2024

Abstract

The concept of coreset is closely related to the notion of ϵ-net. An ϵ-net is a subset of points from a given dataset that captures the essential structure of the dataset within a specified distance ϵ. Coreset can be seen as a type of ϵ-net, where the goal is to construct a subset that approximates the original dataset within a certain tolerance.

In this paper, we introduce some concepts and properties related to Metric Space, ϵ-net, (k, z)-clustering and coreset. And then we provide a brief overview of the development process, the mathematical techniques and great ideas involved in the construction of coreset. We forcus on several specific constructions and mathematical proofs for the k-center and k-median problems in machine learning.

Keywords: coreset, ϵ-net, machine learning, clustering

1 Motivation

The motivation behind coresets arises from the need to address the challenges posed by large datasets in various aspects of data analysis and machine learning. Here are some key motivations for using coresets:

- Computational Efficiency: By constructing a smaller coreset that captures essential properties, computations can be performed more efficiently, saving time and resources.
- Memory Efficiency: Coresets provide a compact representation, reducing memory requirements, which is advantageous in memory-limited or distributed computing environments.
- Scalability: Coresets reduce data dimensionality and size, enabling computations with limited resources, ensuring scalability as datasets grow.
- Streaming Data Analysis: Coresets can be constructed incrementally, adapting to evolving data distributions in real-time, facilitating efficient analysis of streaming data.
- Privacy and Security: Coresets protect sensitive information by working with a reduced dataset, minimizing the risk of exposure.

- Communication Efficiency: Coresets minimize data transmission between computing nodes in distributed environments, improving efficiency and reducing bandwidth usage.
- Mergeability: Coresets can be merged to create a representative coreset of the entire dataset, preserving clustering structures for effective clustering algorithms. This allows for scalable and distributed clustering tasks.

Overall, the motivation for using coresets lies in their ability to provide compact representations of large datasets, enabling efficient and scalable data analysis, reducing computational and memory requirements, and addressing challenges related to privacy and communication efficiency.

2 Background

2.1 Metric Space (X, d)

A metric space consists of a set of elements X and a distance function d that assigns a non-negative real number to pairs of elements in X, satisfying the following properties:

- Non-negativity: $d(x, y) \geq 0, \ \forall x, y \in X$
- Identity of Indiscernibles: $d(x, y) = 0 \iff x = y$
- Symmetry: $d(x, y) = d(y, x), \ \forall x, y \in X$
- Triangle Inequality: $d(x, z) \leq d(x, y) + d(y, z), \ \forall x, y, z \in X$

2.2 ϵ-net

ϵ-net provides a way to approximate a metric space with a smaller set of representative points. It can help reduce the data size and improve the efficiency of various computations, such as range searching, clustering, and nearest neighbor search. Now we break down the definition.

2.2.1 Definition

Given a metric space X with distance function d, a subset S of X is an ϵ-net[12] if

- $\forall x \in X_i, \exists y \in S_i, \ s.t. \ d(x, y) \leq O(\epsilon)$ (satisfies covering)
- $\forall x, \ y \in S_i, \ d(x, y) \geq O(\epsilon)$ (satisfies packing)

2.2.2 Properties

- Cardinality: The cardinality of an ϵ-net, denoted by $N(\epsilon)$, represents the minimum number of points required to form an ϵ-net. Determining the exact cardinality depends on the specific metric space and the desired approximation tolerance.

- Size: The size of an ϵ-net, denoted by $M(\epsilon)$, refers to the diameter of the largest ball that can be covered by the ϵ-net. It measures the maximum distance between any two points and provides an upper bound on the size of the smallest covering for a set of points within the distance ϵ.

2.3 (k,z)-clustering

Clustering is a fundamental task in machine learning that involves grouping similar data points together based on their inherent characteristics. The goal is to identify meaningful patterns, structures, or subgroups within a given dataset without any prior knowledge of the class labels or ground truth. Now we break down the definition.

2.3.1 Definition

The (k,z)-clustering[12] of $X \subset \mathbf{R}^d$ is to compute a k-subset $C \subset \mathbf{R}^d$ such that

$$cost_z(X, C) = \sum_{x \in X} d^z(x, C) = \sum_{x \in X} \min_{c \in C} d^z(x, c)$$

is minimized, where $cost_z$ denotes the clustering objective.

2.3.2 Special cases

- When $z = 1$, (k,1)-clustering is k-median
- When $z = 2$, (k,2)-clustering is k-means
- When $z = \infty$, (k,∞)-clustering is k-center

2.4 Coreset

2.4.1 definition

A weighted subset[12] $S \subset X$ with weight function $w : S \to \mathbf{R}_{\geq 0}$ is an $\epsilon - coreset$ for $(k, z) - clustering$ of X, if for any k-subset $C \subset \mathbf{R}^d$, we have:

$$cost_z(S, C) := \sum_{x \in S} w(x) \cdot d^z(x, C) \in [(1 - \epsilon) \cdot cost_z(X, C), (1 + \epsilon) \cdot cost_z(X, C)]$$

The weight function is crucial here, as without weights, only uniform sampling can be performed. In that case, the size of the constructed coreset would depend on n, at least logarithmically, and potentially even as high as Ωn, where n is the size of the dataset X.

2.4.2 Relationship with (k,z)-clustering

Coresets are smaller subsets of the original dataset that preserve its clustering structure or distribution. They minimize information loss while significantly reducing data size. Coresets benefit clustering in two main ways:

- Scalability and Efficiency: Coresets reduce computational and memory constraints in clustering large-scale datasets. They provide a smaller representation that retains clustering structure, enabling more scalable and efficient clustering algorithms.
- Preserving Clustering Structure: Coresets include representative points from each cluster, allowing accurate identification of the main clusters. By applying clustering algorithms directly to the coreset, instead of the entire dataset, clustering becomes faster and more efficient.

3 Brief overview of techniques

3.1 Sampling-based Method

Coresets are constructed by selecting a subset of points from the dataset based on sampling strategies.

- Random Sampling: Randomly select a fixed number of points from the dataset as the coreset.
- K-Means++: Select centroids with probabilities proportional to their distances from previously chosen centroids, ensuring representative and well-spread points.

3.2 Optimization-Based Method

Coresets are constructed by solving optimization problems that minimize an objective function.

- Facility Location:Select representative points that minimize the total cost or distance from the dataset, often solved using greedy algorithms or linear programming.

3.3 Probabilistic Method

Coresets are constructed by assigning probabilities to data points and sampling accordingly.

- Importance Sampling: Assign weights to points based on their importance or relevance to the clustering problem, influencing their selection for the coreset.

3.4 Geometric decomposition-based Method

Coresets are constructed by partitioning the dataset into subsets based on geometric properties.

- Hierarchical Decomposition: Divide the dataset into smaller regions using hierarchical techniques like quadtree or kd-tree, selecting representative points from each level.
- Density-based Decomposition: Identify dense regions in the dataset. Points within each dense region form subsets, and representative points are selected to construct the coreset.

It's worth noting that these techniques can be combined or modified based on specific requirements or the nature of the dataset.Once the coreset is constructed, it can be used as a representative subset for clustering. The clustering algorithm is applied to the coreset, and the resulting clusters can be mapped back to the original dataset to obtain the final clustering results.

4 Concrete construction and Proof

[12]

4.1 k-center coreset

The following gives an algorithm to construct coreset for k-center problem and proves its size and correctness. The intuition of the algorithm is to use representational point to summarize nearby points.

4.1.1 Algorithm

- Construct a 2-approximate solution $C^* = \{c_1^*, \cdots, c_k^*\}$ by Gonzalez algorithm[10]:
  - Choose the first center arbitrarily.
  - Choose remaining $k-1$ centers using the following criteria: Let $c_1^*, c_2^*, \cdot, c_i^*$ be the already chosen centers. Choose c_{i+1}^* by picking the point which is farthest from already selected centers.
- Partition the Set X into k clusters $X_1, \cdots, X_k$ according to their distance to c_i^*
- Let r_i be the radius of each cluster X_i, and r denotes $\max_{i\in[k]} r_i$. Construct $O(\epsilon r)$-nets S_i of X_i
- Construct $S := \cup_{i\in[k]} S_i$

4.1.2 Analysis

Size: Since the radius of each partition X_i is no larger than r, so that every $|S_i| = O(\epsilon^{-d})$, then we have $|S| = k|S_i| = O(k\epsilon^{-d})$

Correctness: $\forall C \subset \mathbf{R}^d$, we have $\max_{x \in X} d(x, C) \geq \Omega(r)$, and we can conclude the followings according to the properties of covering and packing of $O(\epsilon r)$-net:

- $\max_{x \in S} d(x, C) \leq \max_{x \in X} d(x, C)$
-

$$\exists s \in S, d(s, C) \geq \max_{x \in X} d(x, C) - O(\epsilon r) \geq (1 - \epsilon) \max_{x \in X} d(x, C)$$
$$\implies \max_{x \in S} d(x, C) \geq (1 - \epsilon) \max_{x \in X} d(x, C)$$

So that we have $|cost_z(S, C) - cost_z(X, C)| \leq \epsilon$

4.2 1-Dimension k-median

Based on the intuition of using representational point to summarize nearby points, the following introduces the algorithm [8] to construct coreset for 1-Dimension k-median coreset .

4.2.1 Algorithm(Bucket decomposition)

As for 1-D k-median problem, the distance between two points is simply the absolute difference between their coordinates. If we denote the two points as x and y, then the distance d between them is calculated as:

$$d(x, y) = |x - y|$$

- Compute an $O(1)$-approximate value OPT^*[11]:
  - Randomly sample points from the dataset according to a certain probability distribution. The probability of a point being sampled is proportional to its contribution to the cost of the current solution.
  - For each sampled point, calculate its potential contribution to the cost of the solution if it were to be added as a center. Select the point that reduces the cost the most and add it to the set of centers.
  - Repeat the above steps until the desired number of centers (k) have been selected or the cost cannot be reduced further, output OPT^*.
  - This is a $(1 + \epsilon) - approximation$ algorithm, the actual implementation involves more complex steps such as adjusting the sampling probability and handling edge cases, but it can be implemented in linear time.

- Let bucket $B \subseteq X$, where a "bucket" refers to a partition or grouping of points from the input dataset, be a contiguous interval:
  - Define $\overline{B} = \frac{\sum_{x \in B} x}{|B|}$ as the average point of B
  - Define $\triangle(B) = \sum_{x \in B} |x - \overline{B}|$ as the cumulative error of B
- Initialize an empty set of buckets and iterate over each point in the input dataset.
- For each point, assign it to the closest bucket based on the distance to the centroid of the bucket.
- We set the threshold that $\triangle(B_i) \leq \xi$, where $\xi = \Theta(k^{-1}\epsilon \cdot OPT^*)$, and if the assigned bucket exceeds threshold, create a new empty bucket and reassign the point to the new bucket.
- Finally we can partition X into buckets $B_1, B_2, \cdots, B_m$
- Construct the coreset S as the collection of all $\overline{B}_i$ with weight $w(\overline{B}_i) = |B_i|$

The intuition of the algorithm is to represent all the points in a bucket B_i using the average point $\overline{B}_i$ and apply the weight of the bucket size to the average point.

4.2.2 Proof of coreset size

We claim that the size of the coreset $|S| = m = O(k\epsilon^{-1})$, and the proof is as follows.

Notice that at most $O(k)$ buckets contain a median point of two nearby centers or contain centers in C^*, where $C^* \subset \mathbf{R}$ is an OPT for k-median of X. (Only in the case of 1-Dimension)

- When fix a bucket B(not in those $O(k)$ buckets) and a center c outside B, then we have $2cost_1(B, c) \geq \triangle(B)$, because

$$2cost_1(B, c) = 2\sum_{x \in B}(x - c) \geq 2 \sum_{x \in B,\ x \geq \overline{B}} (x - \overline{B}) = \triangle(B)$$

- Let M denotes the collection of those $O(k)$ buckets, then we have $|\overline{M}| \geq m - O(k)$
- Also we have

$$\begin{aligned} OPT^* &\geq cost_1(X, C^*) \geq \sum_{i \in \overline{M}} cost_1(B_i, C^*) \\ &\geq \sum_{i \in \overline{M}} \triangle(B_i) \\ &\approx |\overline{M}| \cdot \xi \end{aligned}$$

- Since $\xi = \Theta(k^{-1}\epsilon \cdot OPT^*)$, so that $m = O(k\epsilon^{-1})$

The intuition is that most of the buckets $(m - O(k))$ induce expensive cost, so that by restricting the total cost to less than OPT^*, we can limit the number of the buckets to $O(k\epsilon^{-1})$.

4.2.3 Proof of correctness

We prove that S is an ϵ-coreset for k-median of X as follows:

- When fix a bucket B and a center c outside B, then we have $cost_1(B, c) = |B| \cdot d(\overline{B}, c)$, because
$$cost_1(B, c) = \sum_{b \in B}(b - c) = |B| \cdot \overline{B} - |B| \cdot c = |B| \cdot d(\overline{B}, c)$$
Notice that estimating the cost from $\overline{B}$ is equal to that from the original bucket, so that the weighted point $\overline{B}_i$ is a 0-coreset of $B_i (B_i \in \overline{M})$, which is zero-error.

- Each $B_i \in M$ induces $error \leq \triangle(B_i)$, because
$$\begin{aligned} |cost_1(B_i, C) - |\overline{B_i}| d(\overline{B_i}, c)| &\leq \sum_{x \in B_i} |d(x, C) - d(\overline{B_i}, C)| \\ &\leq \sum_{x \in B_i} d(x, \overline{B}_i) \\ &= \triangle(B_i) \end{aligned}$$

- combining the results above, for every $C \subset \mathbf{R}$, the total error of the coreset S is at most
$$\begin{aligned} |cost_1(X, C) - cost_1(S, C)| &\leq \sum_i |cost_1(B_i, C) - w(\overline{B_i}) \cdot d(\overline{B}_i, C)| \\ &\leq \sum_{i \in M} \triangle(B_i) \\ &\leq |M| \cdot \xi \\ &\leq (2k - 1) O(k^{-1} \cdot OPT^*) \\ &\leq \epsilon \cdot cost_1(X, C) \end{aligned}$$

4.3 Extension to higher dimension

Now we want to extend coreset to higher dimension, so we introduce Ring Decomposition.[2] The intuition is to partition the data points into concentric rings or annuli around a central point. Each ring represents a different level of density or distance from the center. This reduces the dimensionality of the problem and allows us to approximate the clustering structure using a smaller set of representative points.

4.3.1 Algorithm(Ring Decomposition)

- Compute an $O(1)$-approximate solution $C^* = \{c_1^*, \cdots, c_k^*\} \subset \mathbf{R}^d$ to the problem k-median of X. We partition X into $X_1, \cdots, X_k$ where X_i contains all the points $x \in X$ with closest center c_i^*. (We have already introduced the algorithm above)

- Compute $r = \epsilon \cdot \frac{cost_z(X_i, c_i^*)}{|X_i|}$ for every X_i

- Decompose X_i into rings $R_0, \cdots, R_m$, satisfying:
  - $\forall x \in R_0, d(x, c_i^*) \leq r$
  - $\forall x \in R_j (j \in [m]), 2^{j-1}r < d(x, c_i^*) \leq 2^j r$
  - Since $\forall x \in X_i, d(x, c_i^*) \leq cost_z(X_i, c_i^*)$, we have $m = O(\log \frac{n}{\epsilon})$
- Construct $S_i := \cup_{0 \leq j \leq m} S_{i,j}$, where $S_{i,j}$ satisfies
  - $S_{i,0} \leftarrow c_i^*$ where $w(c_i^*) = |R_0|$
  - $\forall j \in [m], S_{i,j} \leftarrow$ a uniform sample of $\Gamma = \tilde{O}(kd\epsilon^{-2})$ points with weight $w = \frac{|R_j|}{\Gamma}$
- Construct the coreset $S \longleftarrow \cup_{i \in [k]} S_i$, and $|S| = \tilde{O}(kd\epsilon^{-2} \log \frac{n}{\epsilon})$

4.3.2 Proof of correctness

Firstly we analyze the error of $S_{i,0}$:

$$\begin{aligned}
\forall C \subset \mathbf{R}^d,\ |cost_z(R_0, C) - cost_z(S_{i,0}, C)| &\leq \sum_{x \in R_0} |d(x, C) - d(c_i^*, C)| \leq \sum_{x \in R_0} d(x, c_i^*) \\
&\leq |R_0| \cdot r \\
&\leq |X_i| \cdot \epsilon \cdot \frac{cost_z(X_i, c_i^*)}{|X_i|} \\
&= \epsilon \cdot cost_z(X_i, c_i^*)
\end{aligned}$$

Then we analyze the error of $S_{i,j} (j \in [m])$, Consider ball $B(c_i^*, \frac{2^j r}{\epsilon})$:

- Centers outside the ball B are too far, so they induce negligible error:

$$\forall c \notin B, \forall x, y \in R_j,\ d(x, c) \in [(1 - \epsilon) \cdot d(y, c), (1 + \epsilon) \cdot d(y, c)]$$

- Consider an $\epsilon \cdot 2^{j-1}r$-net discretizing centers inside the ball $B(c_i^*, \frac{2^j r}{\epsilon})$:

$$\forall c \in B,\ \exists c' \in N,\ d(c, c') \leq \epsilon \cdot 2^{j-1} r$$

so that:

  - $|cost_z(x, c) - cost_z(x, c')| \leq \epsilon \cdot 2^{j-1}r \Rightarrow$ The error induced by $C \subset N^k$ is at most $\epsilon \cdot 2^{j-1}r \cdot |R_j| \leq \epsilon \cdot cost_z(R_j, c_i^*)$
  - Fix $C \subset N^k$, We can view $cost_z(R_j, C)$ as the expected distance from a point in R_j to its nearest center in C, multiplied by the size of R_j.
  - Similarly, $cost_z(S_{i,j}, C)$ can be viewed as the expected distance from a point in $S_{i,j}$ to its nearest center in C, multiplied by the size of $S_{i,j}$.

– Since $S_{i,j}$ is a uniform random sample of R_j, these two expected values should be close. Hoeffding's inequality gives us a bound on how much they can deviate. Specifically, if we have $\Gamma = \epsilon^{-2} \log \delta^{-1}$ samples, then with probability $1 - \delta$, the difference between these two expected values will not exceed $\epsilon \cdot cost_z(R_j, c_i^*)$, i.e.

$$cost_z(S_{i,j}, C) \in [cost_z(R_j, C) - \epsilon \cdot cost_z(R_j, c_i^*),\ cost_z(R_j, C) + \epsilon \cdot cost_z(R_j, c_i^*)]$$

with the probability of $1 - \delta$

Combining the error of $S_{i,0}$ and $S_{i,j} (j \in [m])$ together:

- $\forall X_i$ and $\forall C \subset \mathbf{R}^d$:

$$\begin{aligned} |cost_z(X_i, C) - cost_z(S_i, C)| &\leq \sum_{0 \leq j \leq m} |cost_z(R_j, C) - cost_z(S_{i,j}, C)| \\ &\leq \epsilon \cdot cost_z(X_i, c_i^*) + \sum_{0 \leq j \leq m} \epsilon \cdot (cost_z(R_j, C) + cost_z(R_j, c_i^*)) \\ &\leq \epsilon \cdot cost_z(X_i, C) + 2\epsilon \cdot cost_z(X_i, c_i^*) \end{aligned}$$

- For the whole X and the coreset S, we have:

$$\begin{aligned} |cost_z(X, C) - cost_z(S, C)| &\leq \sum_{i \in [k]} |cost_z(X_i, C) - cost_z(S_i, C)| \\ &\leq \sum_{i \in [k]} \epsilon \cdot cost_z(X_i, C) + 2\epsilon \cdot cost_z(X_i, c_i^*) \\ &\leq \epsilon \cdot cost_z(X, C) + 2\epsilon \cdot cost_z(X, C^*) \\ &\leq O(\epsilon) \cdot cost_z(X, C) \end{aligned}$$

5 Evolution of coreset and Related work

[13]

- In 2004, Sariel Har-Peled and Soham Mazumdar [1] showed the existence of small coresets for the problems of computing k-median and k-means clustering for points in low dimension. Given $X \subset \mathbf{R}^d$, they can compute coreset $C \subset X$ of size $O(k\epsilon^{-d} \log n)$, such that one can compute the k-median/means clustering on C instead of on X, and get an $(1 + \epsilon)$-approximation. The main drawback is the exponential dependency on the dimension.

- In 2009, Ke Chen [2] introduced another construction method: First, find an approximate solution , then divide the entire data set into several rings with exponentially increasing radii, known as ring decomposition, and finally sample uniformly from each ring. The size of the coreset is $poly(k, d, \epsilon^{-1}, \log n)$. Its main drawback is the inclusion of terms $\log n$, causing the core set's size to grow with n.

- In 2011, Dan Feldman and Michael Langberg [3] proposed the importance sampling framework, improving the size of the coreset to $poly(k, d, \epsilon^{-1})$. This framework observes the total sensitivity and the VC/shattering dimension of the optimization problem itself. It has been widely applied to core set constructions in various optimization problems.

- In 2018, Christian Sohler and David P. Woodruff [4] successfully eliminated the parameter associated with the coreset, improving the size of coreset to $poly(k/\epsilon)$.

- In 2021, by combining the methods above, Cohen-Addad [5] improved the size of the coreset to $\tilde{O}(k\epsilon^{-4})$. Their main idea is to apply importance sampling to different ring structures, gaining a deeper understanding of which data points play similar roles in clustering.

- Cohen-Addad [6] [7] further improved the size of the coreset to $\min\{\tilde{O}(k\epsilon^{-4}), \tilde{O}(k^{1.5}\epsilon^{-2})\}$. The key is to reduce the problem to the uniform convergence of the clustering function and better understand the interdependence of convergence properties between different core sets using the chaining method. They also provided a lower bound[6] $\Omega(k\epsilon^2)$ for the coreset size.

6 Other types of coresets

6.1 Weak coreset

A weak coreset [9][12], denoted as a weighted set of points D in $\mathbf{R}^d$, is characterized by the property that a $(1+\epsilon)$-approximation for the optimal solution of D also yields a $(1+\epsilon)$-approximation for the optimal solution of the full dataset X. In other words, any black box algorithm that computes a $(1+\epsilon)$-approximation for the coreset D also yields a $(1+\epsilon)$-approximation for the original set X. Therefore, a weak coreset can be seen as a reduction from the clustering problem with input X to the same problem with input D.

Such definition is sufficient to ensure that an approximate optimal solution of the original dataset can be reconstructed through a weak coreset. However, compared to coresets, weak coresets lack mergeability and are not easily applicable to streaming or distributed models. On the other hand, there is currently no evidence to suggest that weak coresets can be smaller than coresets.

6.2 Streaming coreset

A weak coreset D that is updated online during one pass over the n points of X, while using only $O(d \cdot |D|)$-space in memory. Streaming coresets [9] can thus be used online to compute a $(1+\epsilon)$-approximation for the optimal solution of the points in X viewed so far.

Due to the requirement of only one pass over the data and linear memory usage proportional to the size of the coreset, Streaming coresets can effectively work even in cases where the dataset is extremely large.

7 Conclusion

We have given a brief overview of some concepts and properties related to coreset, and we have shown how coresets are constructed for clustering problems in machine learning as well as how to analyze the size of a coreset and prove its correctness.

We have described algorithms to find $O(k\epsilon^{-d})$-size coreset for k-center as well as $O(k\epsilon^{-1})$-size coreset for 1-Dimensional k-median, and extend 1-D k-median to higher dimension with coreset of size $\overline{O}(kd\epsilon^{-2}\log\frac{n}{\epsilon})$. The intuition of each construction is essentially the same, which is to use representational point to summarize nearby points while preserving the geometric and statistical properties of the original dataset, and the implementation is often based on an approximate solution center set.

References

[1] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In 36th Annual ACM Symposium on Theory of Computing, pages 291–300, 2004.

[2] Ke Chen. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. SIAM Journal on Computing, 39(3):923–947, 2009.

[3] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In STOC, pages 569–578. ACM, 2011.

[4] Christian Sohler, David P. Woodruff. Strong Coresets for k-Median and Subspace Approximation: Goodbye Dimension

[5] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. June 21-25, 2021, pages 169–182. ACM, 2021.

[6] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. In Proceedings of the firty-fourth annual ACM symposium on Theory of computing, 2022.

[7] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, Chris Schwiegelshohn, and Omar Ali Sheikh-Omar. Improved coresets for Euclidean k-means. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022.

[8] Sariel Har-Peled, Akash Kushal. Smaller Coresets for k-Median and k-Means Clustering. March 28, 2005

[9] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In STOC, pages 569–578. ACM, 2011.

[10] Lecture 2: A Greedy 2-approximation for k-center. https://ugtcs.berkeley.edu/src/approx-sp19/scribe-notes-2.pdf

[11] Amit Kumar, Yogish Sabharwal, Sandeep Sen. A Simple Linear Time (1 +)-Approximation Algorithm for k-Means Clustering in Any Dimensions

[12] Lingxiao Huang. (2023, October). Development of Core Set Construction Algorithms. Retrieved from https://www.bilibili.com/video/BV1dc411f7ac/

[13] Lingxiao Huang. (2023, April). Research Notes on Coreset [from zhihu zhuan lan]. Retrieved from https://zhuanlan.zhihu.com/p/619661335/