Coresets for k-median/means Clustering

TOP: Data Clustering 076/091

Instructor: Sayan Bandyapadhyay

Portland State University

Outline

1. Coresets for k-median/means
2. Coreset Construction for k-median

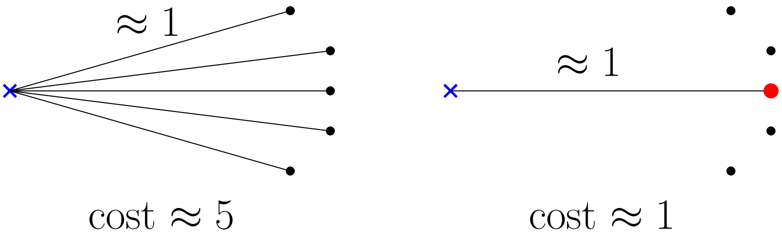
Euclidean k-median

Given a set X of n points in $\mathbb{R}^d$

- Find a set C of k points (cluster centers) in $\mathbb{R}^d$ that minimizes,

$$\text{cost}(X, C) = \sum_{p \in X} ||p - NearestCenter(p)||$$

Coresets



Cost comparison of original dataset and coreset

Coresets

Given a set S of points with weight w_p for each $p \in S$, the k-median cost of S w.r.t a set of centers C

$$\mathrm{wcost}(S, C) = \sum_{p \in S} w_p \cdot ||p - NearestCenter(p)||$$

Coresets

Given a set S of points with weight w_p for each $p \in S$, the k-median cost of S w.r.t a set of centers C

$$\mathrm{wcost}(S, C) = \sum_{p \in S} w_p \cdot ||p - NearestCenter(p)||$$

For the original set X, $w_p = 1$ for all $p \in X$, so

$\mathrm{wcost}(X, C) = \mathrm{cost}(X, C)$

Coresets

Given a set S of points with weight w_p for each $p \in S$, the k-median cost of S w.r.t a set of centers C

$$\text{wcost}(S, C) = \sum_{p \in S} w_p \cdot ||p - NearestCenter(p)||$$

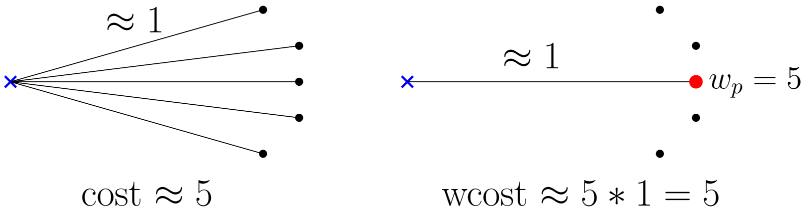
For the original set X, $w_p = 1$ for all $p \in X$, so

$\text{wcost}(X, C) = \text{cost}(X, C)$

(ϵ-coreset.) A weighted subset $S \subseteq X$ is an ϵ-coreset if for any set of k centers C,

$$(1 - \epsilon) \cdot \text{cost}(X, C) \leq \text{wcost}(S, C) \leq (1 + \epsilon) \cdot \text{cost}(X, C)$$

Back to Our Example



Outline

The General Idea

- Similar to k-center
- Compute an approximate clustering

The General Idea

- Similar to k-center
- Compute an approximate clustering
- From each cluster, carefully pick a weighted subset of points

The General Idea

- Similar to k-center
- Compute an approximate clustering
- From each cluster, carefully pick a weighted subset of points
- The main obstacle is that clusters are not balls

The General Idea

- Similar to k-center
- Compute an approximate clustering
- From each cluster, carefully pick a weighted subset of points
- The main obstacle is that clusters are not balls
- We will take multiple balls per cluster

Approximate Euclidean k-median Clustering

- How to compute a good clustering?

Approximate Euclidean k-median Clustering

- How to compute a good clustering?
- How about applying the local search algorithm?

Approximate Euclidean k-median Clustering

- How to compute a good clustering?
- How about applying the local search algorithm?
- Local search works with finite set of centers

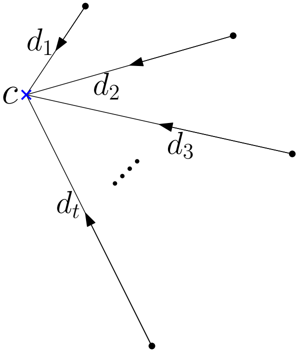
Approximate Euclidean k-median Clustering

- How to compute a good clustering?
- How about applying the local search algorithm?
- Local search works with finite set of centers
- Set X as the set of centers

Approximate Euclidean k-median Clustering

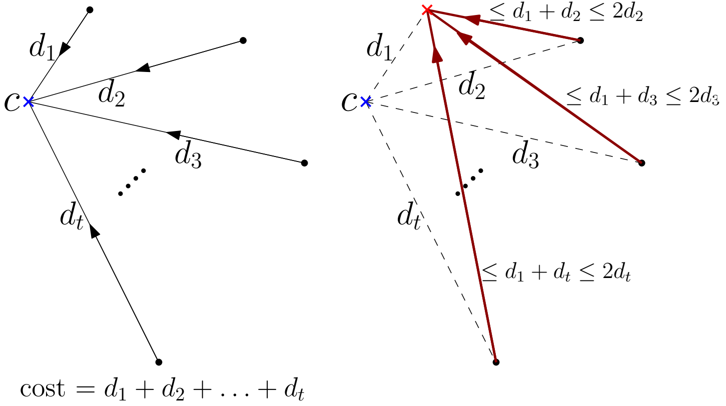
- How to compute a good clustering?
- How about applying the local search algorithm?
- Local search works with finite set of centers
- Set X as the set of centers
- How much is the cost increase if we use X instead of $\mathbb{R}^d$?

A Cluster with a Center not from X

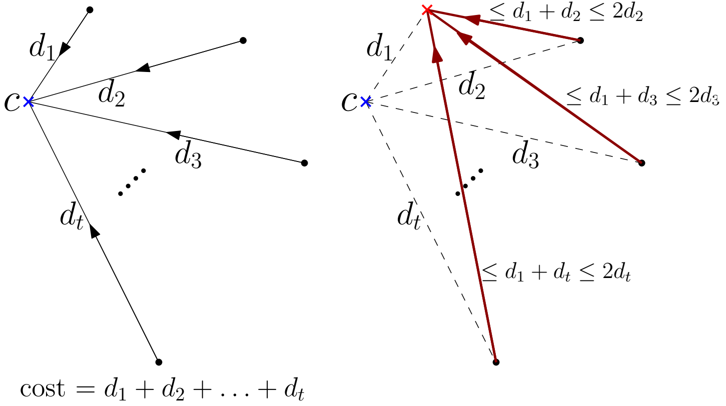


$\text{cost} = d_1 + d_2 + \ldots + d_t$

Reassign Points to the One Closest to c



Reassign Points to the One Closest to c



There is a subset of k centers of X that has cost $\leq 2 \cdot$ OPT

Approximate Euclidean k-median Clustering

- How to compute a good clustering?
- How about applying the local search algorithm?
- Local search works with finite set of centers
- Set X as the set of centers
- How much is the cost increase if we use X instead of $\mathbb{R}^d$? 2-factor!!

Approximate Euclidean k-median Clustering

- How to compute a good clustering?
- How about applying the local search algorithm?
- Local search works with finite set of centers
- Set X as the set of centers
- How much is the cost increase if we use X instead of $\mathbb{R}^d$? 2-factor!!

Local search yields a 3-factor approximation to the best solution

Approximate Euclidean k-median Clustering

- How to compute a good clustering?
- How about applying the local search algorithm?
- Local search works with finite set of centers
- Set X as the set of centers
- How much is the cost increase if we use X instead of $\mathbb{R}^d$? 2-factor!!

Local search yields a 3-factor approximation to the best solution $\Longrightarrow$ we obtain a clustering of cost $\leq 6\cdot$ OPT

Back to Coreset Construction (Chen '09)

- Compute a 6-approximate solution A of the input X

Back to Coreset Construction (Chen '09)

- Compute a 6-approximate solution A of the input X
- Let $\mu = \mathbf{cost}(X, A)/n$ be the average cost

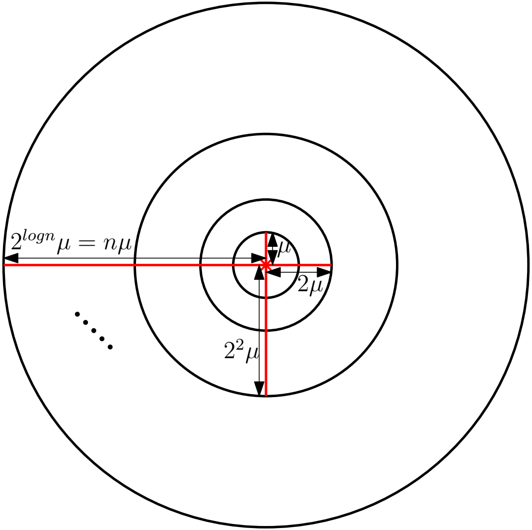
Back to Coreset Construction (Chen '09)

- Compute a 6-approximate solution A of the input X
- Let $\mu = \mathbf{cost}(X, A)/n$ be the average cost
- Maximum $||p - NearestCenter(p)|| \leq \mathbf{cost}(X, A) = n \cdot \mu$

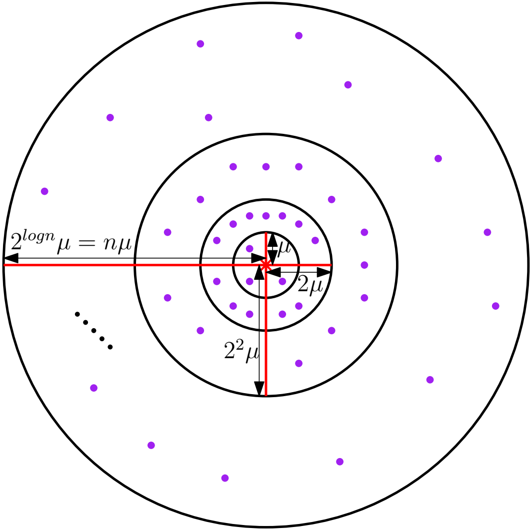
Back to Coreset Construction (Chen '09)

- Compute a 6-approximate solution A of the input X
- Let $\mu = \mathbf{cost}(X, A)/n$ be the average cost
- Maximum $||p - NearestCenter(p)|| \leq \mathbf{cost}(X, A) = n \cdot \mu$
- For each cluster, do the following

Division of a Cluster into Rings



All Points in the Cluster are in the Rings



Coreset Construction

- Points in each ring j are similar: $||p - c|| \in [2^{j-1}\mu, 2^j\mu]$

Coreset Construction

- Points in each ring j are similar: $||p - c|| \in [2^{j-1}\mu, 2^j\mu]$
- Set s to be a sample size

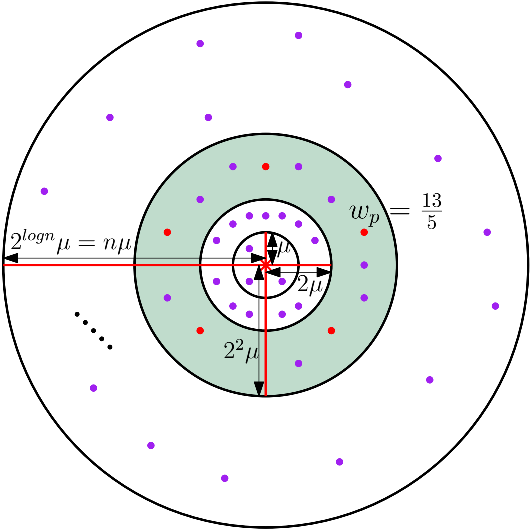
Coreset Construction

- Points in each ring j are similar: $||p - c|| \in [2^{j-1}\mu, 2^j\mu]$
- Set s to be a sample size
- Pick a uniformly random sample of size s from each ring R and add them to coreset

Coreset Construction

- Points in each ring j are similar: $||p - c|| \in [2^{j-1}\mu, 2^j\mu]$
- Set s to be a sample size
- Pick a uniformly random sample of size s from each ring R and add them to coreset
- Distribute the total weight $|R|$ equally to the chosen coreset points from R, i.e., $w_p = |R|/s$

A Random Sample is Picked from each Ring



Coreset Bounds

- Total $k(\log n + 1)$ rings: coreset size = $s \times k(\log n + 1)$

Coreset Bounds

- Total $k(\log n + 1)$ rings: coreset size = $s \times k(\log n + 1)$
- Pick a sample size $s = O(kd \log n / \epsilon^3)$
  - Cost in coreset within $(1 \pm \epsilon)$ of the original cost

Coreset Bounds

- Total $k(\log n + 1)$ rings: coreset size = $s \times k(\log n + 1)$
- Pick a sample size $s = O(kd \log n/\epsilon^3)$
  - Cost in coreset within $(1 \pm \epsilon)$ of the original cost
- Total coreset size: $O(k^2 d \log^2 n/\epsilon^3)$