

Coreset Technique

(Part 1)

OUTLINE

- ① Coresets and Composable Coresets
- ② Case study: k-center clustering
 - Basic notions on clustering
 - k-center problem
 - k-center as a coresset primitive
- ③ Coresets for other clustering problems (Part 2)

Coresets and Composable Coresets

Coreset technique

Suppose that we want to solve a problem Π on instances P which are too large to be processed by known algorithms.

Coreset technique

- ① Extract a "small" subset T from P (dubbed coreset), making sure that it represents P well, w.r.t. solutions to Π .
- ② Run best known (possibly slow) sequential algorithm for Π on the small coresset T , rather than on the entire input P .

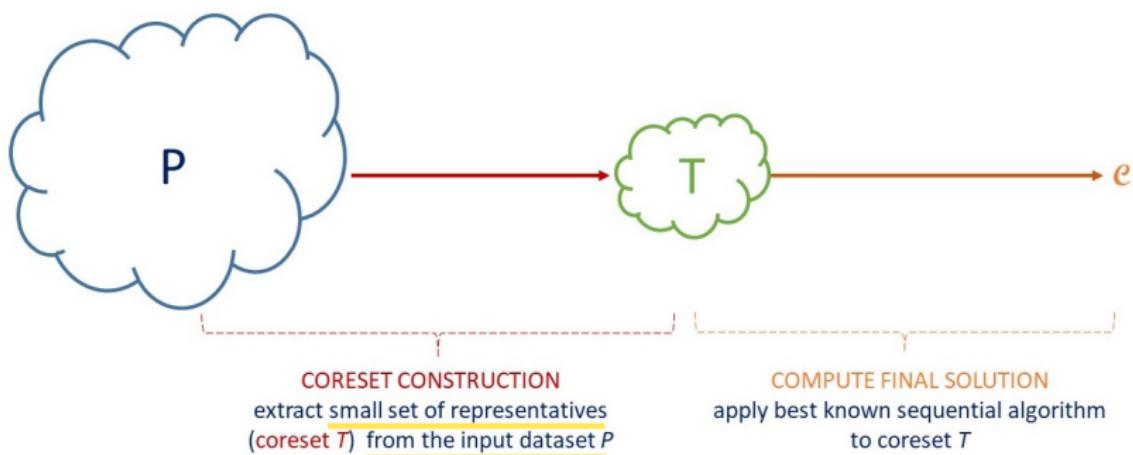
The technique is effective if

- T can be extracted efficiently by processing P , possibly in a distributed or streaming fashion.
- The solution computed on T is a good solution for Π w.r.t. the entire input P .

→ veloce

→ buona approx.

Coreset technique



Composable Coreset technique

vogliamo parallelizzare
MR

coreset specifico: poco costoso
dividendo input \Rightarrow facile implementare

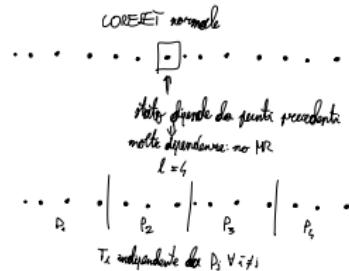
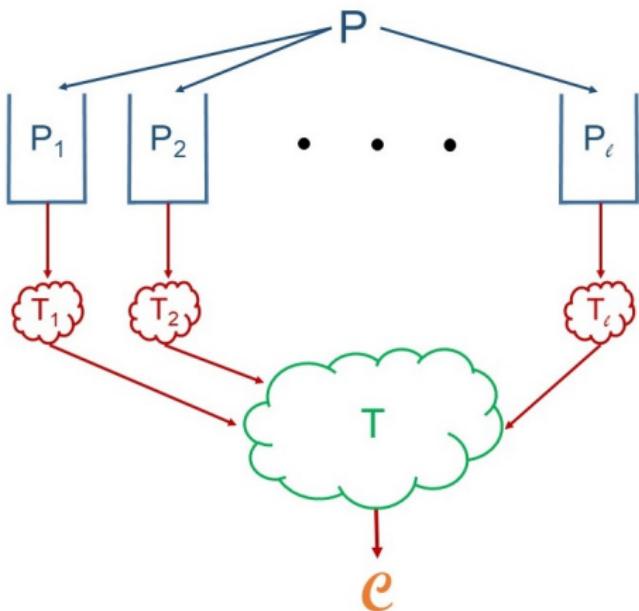
Composable Coreset technique

- ① Partition P into ℓ subsets P_1, P_2, \dots, P_ℓ , and extract a "small" coreset T_i from each P_i , making sure that it represents P_i well. } Map 1^o round
- ② Run best known (possibly slow) sequential algorithm for Π on $T = \bigcup_{i=1,\ell} T_i$, rather than on P . } Reduce 1^o
} Map 2^o

The technique is effective if

- Each T_i can be extracted efficiently from P_i in parallel for all i 's.
- The final coreset T is still small and the solution computed on T is a good solution for Π w.r.t. the entire input P .

Composable coresets technique



- Extract a *small* coreset T_i from each P_i
- Apply best known sequential algorithm to final coreset $T =$ union of the T_i 's, to compute final solution c

Case study: k-center clustering

non sceglie le formule costi \Rightarrow vogliamo alg. efficiente per
k-center

Why is clustering a relevant case study?

- 1- molto somigli
- 2- molto diverso

General definition

Given a set of points belonging to some space, with a notion of distance between points, clustering aims at grouping the points into a number of subsets (clusters) such that

- Points in the same cluster are "close" to one another
- Points in different clusters are "distant" from one another

The distance captures a notion of similarity: close points are similar, distant point are dissimilar.

A clustering problem is usually defined by requiring that clusters optimize a given objective function, and/or satisfy certain properties. Numerous clustering problems have been defined, studied and employed in applications over the years.

Example

aiuta anche in
outlier detection

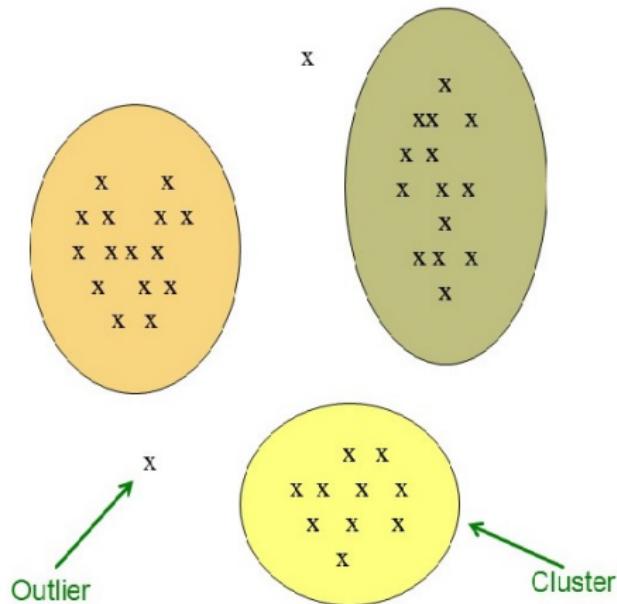


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

The clustering problem is a hard one!

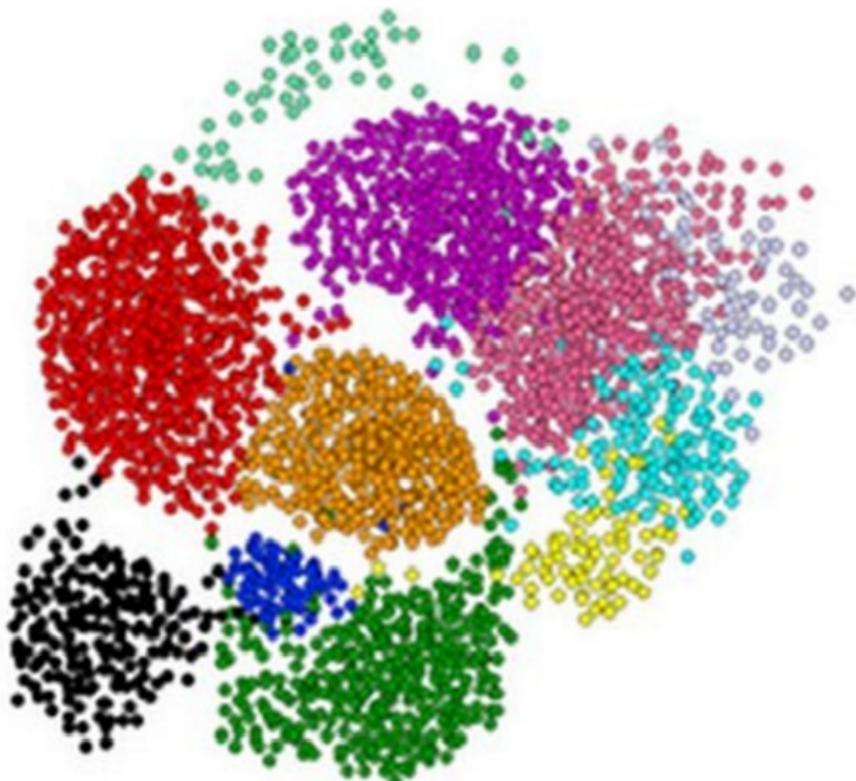


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

Importance of clustering

Main purposes of clustering:

- ① Assessing the structure of the data
- ② Grouping data for future prediction tasks
- ③ Summarizing and compressing data

Applications in many different fields. Some examples:

- **Marketing:** to segment (potential) customers.
- **Biology:** to identify protein families.
- **Image processing:** to do object recognition.
- **Information retrieval:** to categorize documents or web pages.
- **Network analysis:** to identify communities.
- **Facility location:** to decide where to open network hubs/facilities.

Metric Space

Typically, the input of a clustering problem consists of a set of points from a metric space

Definition

A metric space is an ordered pair (M, d) where M is a set and $d(\cdot)$ is a metric on M , i.e., a function

$$d : M \times M \rightarrow \mathbb{R}_0^+$$

such that for every $x, y, z \in M$ the following holds

- $d(x, y) \geq 0$; *sempre rilassata \Rightarrow traiettorie, rete temporali*
- $d(x, y) = 0$ if and only if $x = y$; *\Rightarrow PSEUDOMETRICHE*
- $d(x, y) = d(y, x)$; (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$; (triangle inequality)

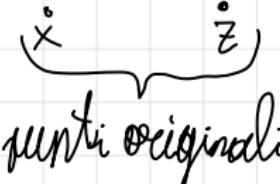
Remark: The choice of the function may have a significant impact on the effectiveness of the cluster analysis.

Why is triangle inequality useful in the big data context?

Dobbiamo rappresentare bene input

 vogliamo distanza tra x e gli altri \Rightarrow
 mi basta calcolarla da un punto

$y \Leftarrow$ osservazione su T

$d(x, z) \leq d(x, y) + d(y, z)$
 vogliamo questo
vogliamo calcolare questo

$d(x, z) \approx d(x, y) \Leftarrow$ se y vicino a z , $d(y, z) \rightarrow 0$

Distance functions

Minkowski distances Let $X, Y \in \mathbb{R}^n$, with $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$. For $r > 0$ the L_r – distance between X and Y (a.k.a. L_r – norm)

$$d_{Lr}(X, Y) = \left(\sum_{i=1}^n |x_i - y_i|^r \right)^{1/r}.$$

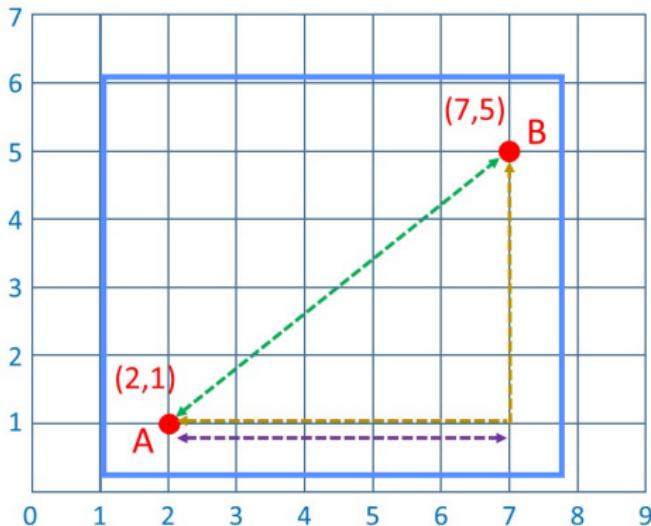
valore assoluto
non garantisce
 $d(x, y) \geq 0$

- $r = 2$ (Euclidean distance) standard distance in \mathbb{R}^n (also denoted by $\|\cdot\|$)
- $r = 1$ (Manhattan distance): used in grid-like environments L_∞
- $r = \infty$ (Chebyshev distance): maximum absolute differences of coordinates (i.e., the limit for r that tends to ∞).

Si ottiene massimo valore in somma \Rightarrow esponenti si cancellano

Distance functions

Examples of Minkowski distances:



$$d_{L1}(A,B) = (7-2)+(5-1) = 9$$

$$d_{L2}(A,B) = (5^2+4^2)^{1/2} = 6.403..$$

$$d_{L\infty}(A,B) = \max\{5,4\} = 5$$

Distance functions

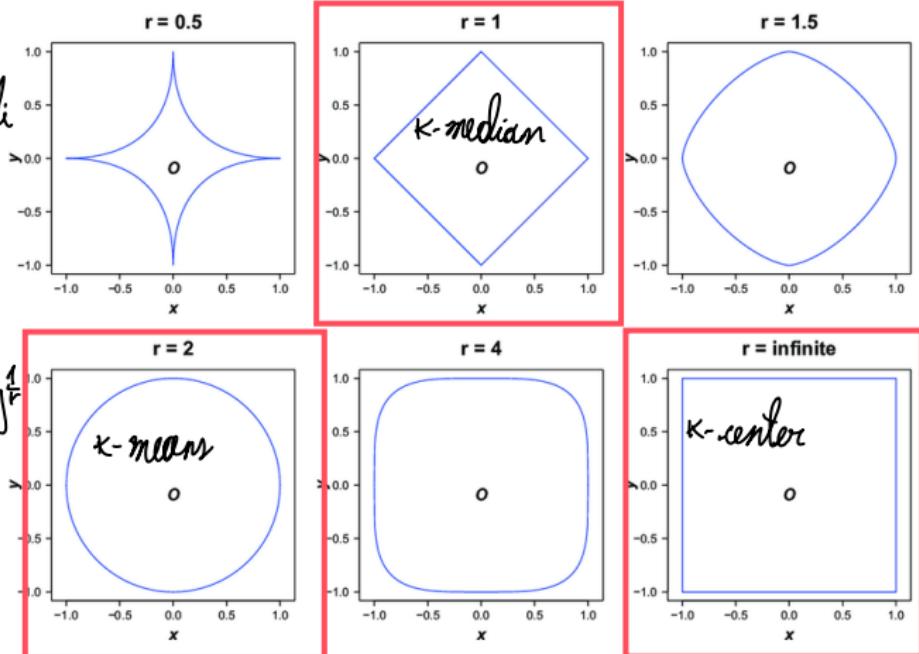
Comparison of Minkowski distances:

aumentando
 r : piú sensibili
a outlier

$$x = (1, \dots, 1)$$

$$x = (y_0, 2, \dots, 2)$$

$$\downarrow$$
$$d(x, y) = [(y_0 - 1)^r + n]^{\frac{1}{r}}$$



Points at distance ≤ 1 from center

Picture from: Urquiza-Aguiar et al., Empirical Analysis of the Minkowski Distance Order in Geographical Routing Protocols for VANETs. Proc. WWIC, 2015.

Distance functions

Angular distance. It is used when points are vectors in \mathbb{R}^n . Given $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ in \mathbb{R}^n , their angular distance is the angle between them, that is,

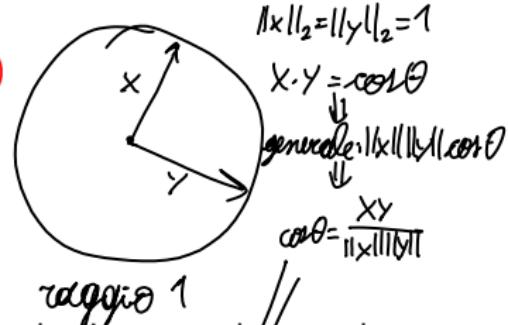
$$d_{\text{angular}}(X, Y) = \frac{1}{\pi} \arccos \left(\frac{X \cdot Y}{\|X\| \cdot \|Y\|} \right)$$

in plane definition

$$= \frac{1}{\pi} \arccos \left(\frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n (x_i)^2} \sqrt{\sum_{i=1}^n (y_i)^2}} \right) \in [0, \pi]$$

Example: For $X = (1, 2, -1)$ and $Y = (2, 1, 1)$

$$d_{\text{angular}}(X, Y) = \arccos \left[\frac{1 \cdot 2 + 2 \cdot 1 - 1 \cdot 1}{\sqrt{1+4+1} \sqrt{4+1+1}} \right] =$$
$$= \arccos \left[\frac{3}{6} \right] = 45^\circ$$

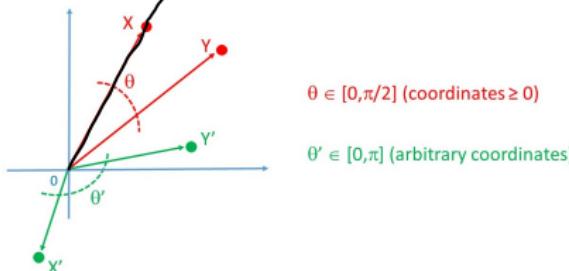


Obs: In [LRU14] it is called *cosine distance*, but in the literature the term has a different meaning.

COSINE SIMILARITY

Distance functions

For non-negative coordinates, $d_{\text{angular}}^Z(X, Y) \in [0, \pi/2]$, while for arbitrary coordinates the range is $[0, \pi]$.



To normalize values in $[0, 1]$, $d_{\text{angular}}(X, Y)$ is often divided by π or $\pi/2$ depending whether vectors have arbitrary or non-negative coordinates. Also, to satisfy the second property of metric spaces, scalar multiples of a vector are regarded as the same vector.

(L'utile per $\mathbb{R} \rightarrow \mathbb{R}$: è documenti ottenuti scrivendo testo di X e coniandolo 2 volte

Remark. This distance is used in information retrieval for documents. A document X over a vocabulary of n words can be represented as an n -vector, where x_i is the number of occurrences of the i -th word of the vocabulary in X .

Distance functions

- **Hamming distance.** Used when points are binary vectors over some n -dimensional space: $X, Y \in \{0, 1\}^n$. The Hamming distance between two vectors is the number of coordinates in which they differ.

$$d_{\text{Hamming}}(X, Y) = |\{i | x_i \neq y_i\}|$$

Example: for $X = (0, 1, 1, 0, 1)$ and $Y = (1, 1, 1, 0, 0)$

$$d_{\text{Hamming}}(X, Y) = 2$$

Observation: $d_{\text{Hamming}}(X, Y) = d_{L1}(X, Y)$

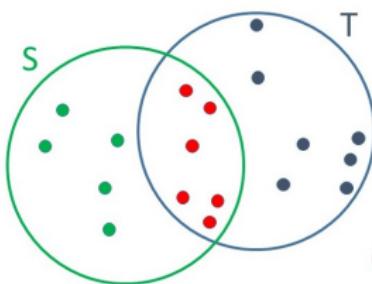
Distance functions

- **Jaccard distance.** Used when points are sets (e.g., documents seen as bags of words). Let S and T be two sets over the same ground set of elements. The Jaccard distance between S and T is

$$d_{\text{Jaccard}}(S, T) = 1 - \frac{|S \cap T|}{|S \cup T|} = \frac{|S \cup T| - |S \cap T|}{|S \cup T|}.$$

Note that the distance ranges in $[0, 1]$, and it is 0 iff $S = T$ and 1 iff S and T are disjoint. The value $|S \cap T|/|S \cup T|$ is referred to as the Jaccard similarity of the two sets.

Example:



$$d_{\text{Jaccard}}(S, T) = 1 - \frac{6}{18} = 2/3$$

Which distance should we use?

Minkowski and Angular distances are used when an object is characterized by the numerical values of its features (e.g., frequency).

- Minkowski distance: it aggregates the gap in each dimension between two objects. Examples: GPS coordinates; *valori reali* measurements of physical/chemical processes.
- Angular distance: it measures the ratios among features' values, rather than their absolute values. Example: distance between documents of different lengths, where each document is characterized by the relative frequency of words from a given vocabulary.

Which distance should we use?

Hamming and Jaccard distances are used when an object is characterized by having or not having some features (no multiplicity).

- Hamming distance: it measures the total number of differences between two objects. Examples: fixed-length binary words, genetic distance.
- Jaccard distance: it measures the ratio between the number of differences between two sets and the total number of points in the sets. Example: the topics describing a document.

Observation: All of the above distances satisfy the four requirements for a metric space.

Exercise

Show that the L_1 satisfies the four requirements for a metric space.

Types of clusterings

Given a set of points in a metric space, a clustering problem often specifies an **objective function** to optimize.

The objective function also allows to compare different solutions.

Clustering problems can be categorized based on whether or not

- A target number k of clusters is given in input.
- For each cluster a center must be identified.
- Disjoint clusters are sought.

Background: optimization problem

set di input I

$\forall i \in I \rightarrow$ set di soluzioni $x_i \Rightarrow$ può essere vuoto o non vuoto

$$x_i = \{x_{i1}, x_{i2}, \dots\}$$

ϕ = funzione obiettivo

$$\phi: S \rightarrow \mathbb{R}$$



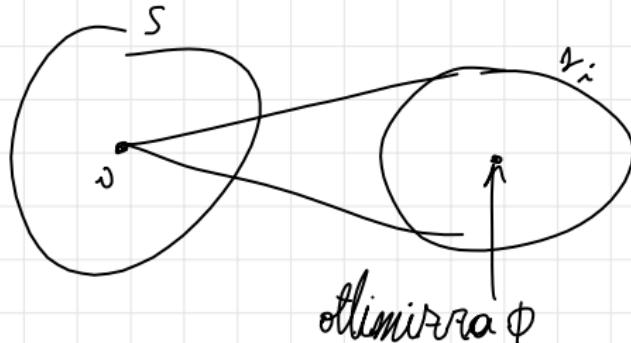
soluzioni

obiettivo: $\forall i \in I$, trovare soluzione $\hat{x} \in x_i$:

$$\hat{x} = \underset{\sigma \in x_i}{\operatorname{argmax}} \phi(\sigma) \Rightarrow \begin{array}{l} \text{problema di} \\ \text{minimizzazione} \\ (\operatorname{argmin}) \end{array}$$

Background: optimization problem

$\exists i$: ottimale è data da soluzione $\hat{o}_1, \hat{o}_2, \dots$.



Molto spesso, difficile trovare soluzioni esatte

Background: optimization problem

Background: approximation algorithm

Molto spesso, dati hanno rumore \Rightarrow sol. esatta non ha senso

Problema di max.

i input, \hat{o}_i : sol. ottima, o_i : sol. data da algoritmo A
 $\phi(o_i) \leq \phi(\hat{o}_i)$ (def. \hat{o}_i)

$\phi(o_i) \leq \frac{\phi(\hat{o}_i)}{c}$ per certo c (fattore di approssimazione)

$c = 1$: sol. esatta

vogliamo $c \geq 1$ più piccolo possibile

$c = 1 + \epsilon$ con ϵ piccolo

x fissiamo ϵ , trade-off qualità-risorse (tempo, spazio)

Background: approximation algorithm

In altri casi, c è fisso \Rightarrow e.g. $c = f(n)$ (n : dimensione input)

$$c = O(1), c = O(\log n),$$

$$c = O(n^b) \text{ con } b < 1$$

In questo caso, è dato da algoritmo

preferiamo $c = 1 + \varepsilon$

Problema di min.

input, \hat{o}_i : sol. ottima, o_i : sol. data da algor. A

$$\Phi(o_i) \geq \Phi(\hat{o}_i) \Rightarrow \Phi(o_i) \leq c \Phi(\hat{o}_i)$$

Observations

Approximation algorithms play a crucial role in big data computing for the following reasons

- For several important optimization problems finding optimal solutions for large inputs is very time consuming. E.g.,
 - NP-hard problems for which it is likely that exponential time is required. ↗ e.g. clustering
 - Problems for which polynomial-time algorithms exist but require superlinear number of operations (e.g., quadratic, cubic). ↗ e.g. similarity search, diameter computation
 $O(n) \rightarrow O(n \log n)$ $O(n^2) \rightarrow O(n)$
- Allowing for approximations widens the spectrum of possible solutions and makes it easier to find one.
- In real-world instances, often affected by noise or uncertainty, the true optimum may not be well defined.

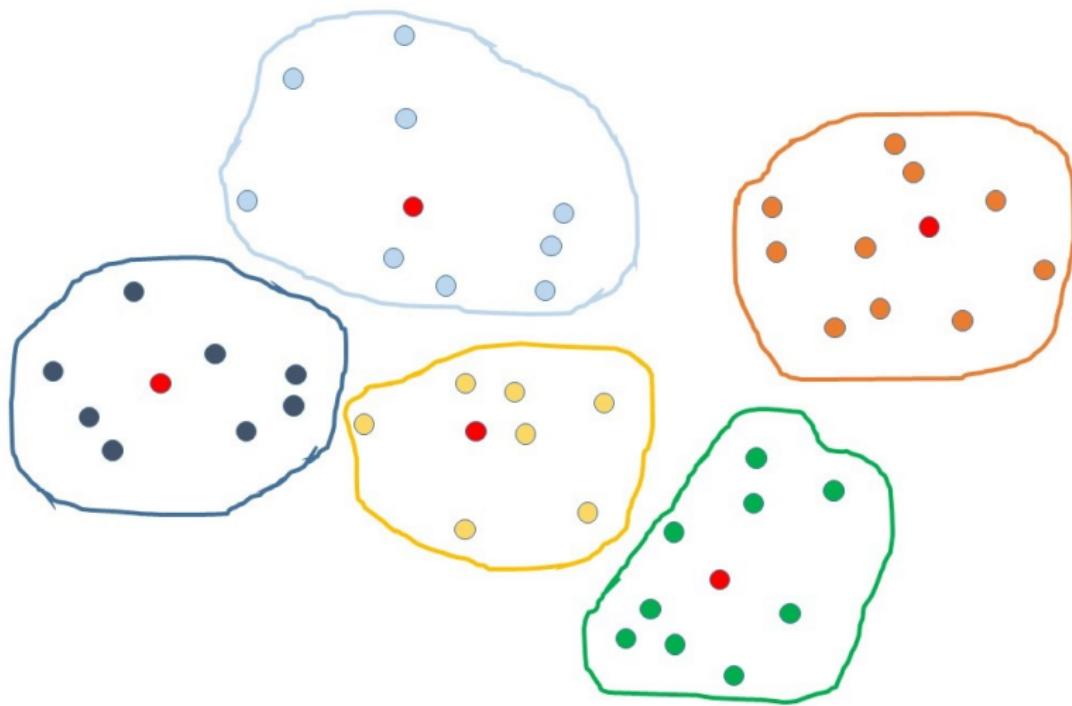
Center-based clustering

Let P be a set of N points in metric space (M, d) , and let k be the target number of clusters, $1 \leq k \leq N$. We define a k -clustering of P as a tuple $\mathcal{C} = (C_1, C_2, \dots, C_k; c_1, c_2, \dots, c_k)$ where

- (C_1, C_2, \dots, C_k) defines a partition of P , i.e.,
$$P = C_1 \cup C_2 \cup \dots \cup C_k$$
- c_1, c_2, \dots, c_k are suitably selected centers for the clusters,
where $c_i \in C_i$ for every $1 \leq i \leq k$.

Remark: in the above definition, the centers are points of P . In some cases (e.g., when points are from Euclidean spaces) centers outside P may be allowed.

Example of 5-clustering



Center-based clustering

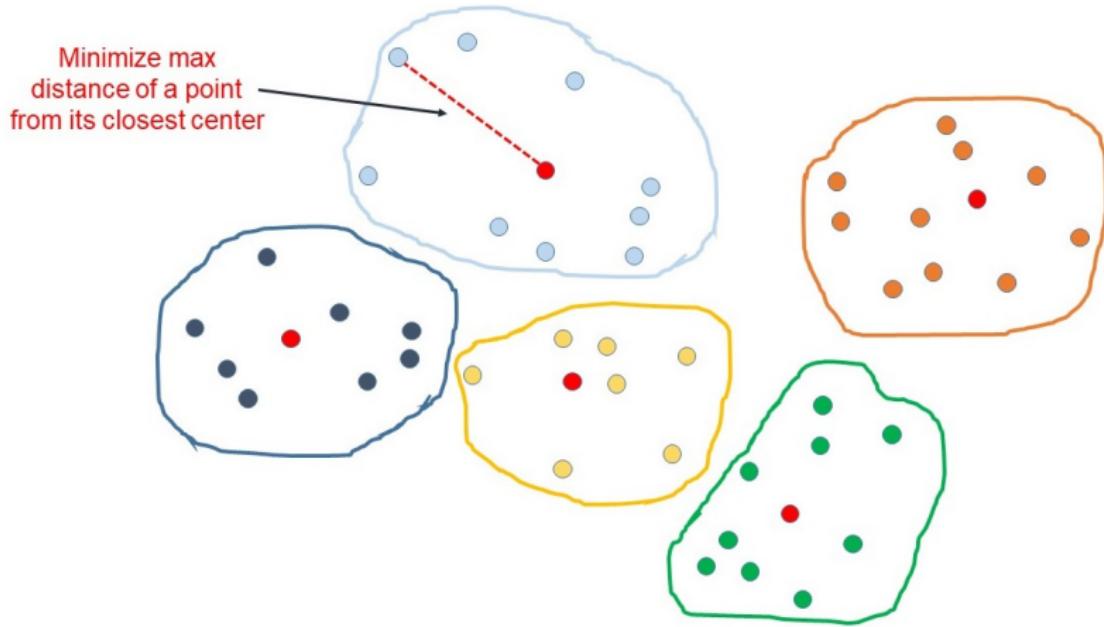
For a given input pointset P , a center-based clustering problem aims at finding a k -clustering of P which optimizes a certain objective function.

The 3 most popular objectives are:

- **k-center clustering:** aims at minimizing the maximum distance of any point from the center of its cluster
- **k-means clustering:** aims at minimizing the sum of the squared distances of the points from the centers of their respective clusters (a.k.a. Sum of Squared Errors (SSE))
- **k-median clustering:** aims at minimizing the sum of the distances of the points from the centers of their respective clusters

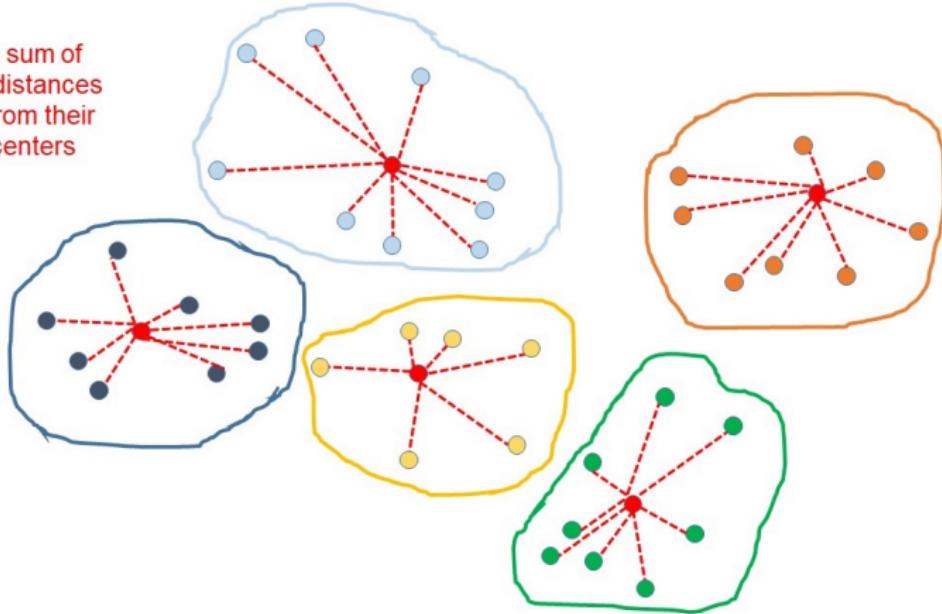
Observation: for k-means and k-median, minimizing the sum of (squared) distances is equivalent to minimizing the average (squared) distance.

k-center objective



k-means/median objectives

Minimize sum of
(squared) distances
of points from their
closest centers



Formal definitions of the problems

For any point $x \in P$ and any subset $S \subseteq P$ we define

$$d(x, S) = \min_{y \in S} d(x, y)$$

in questo caso, $S = \{\text{centri}\}$

Definition

Let (M, d) be a metric space. The **k-center/k-means/k-median** clustering problems are optimization problems that, given a finite pointset $P \subseteq M$ and an integer $k \leq |P|$, require to return the subset $S \subseteq P$ of k centers which minimizes the following respective objective functions:

- $\Phi_{\text{kcenter}}(P, S) = \max_{x \in P} d(x, S)$ (**k-center clustering**)
- $\Phi_{\text{kmeans}}(P, S) = \sum_{x \in P} (d(x, S))^2$ (**k-means clustering**).
- $\Phi_{\text{kmedian}}(P, S) = \sum_{x \in P} d(x, S)$ (**k-median clustering**).

Terminology and notations

For the k-center/means/median clustering problems we have that:

INPUT: Pointset P and integer k

FEASIBLE SOLUTIONS: All subsets $S \subseteq P$ of size k

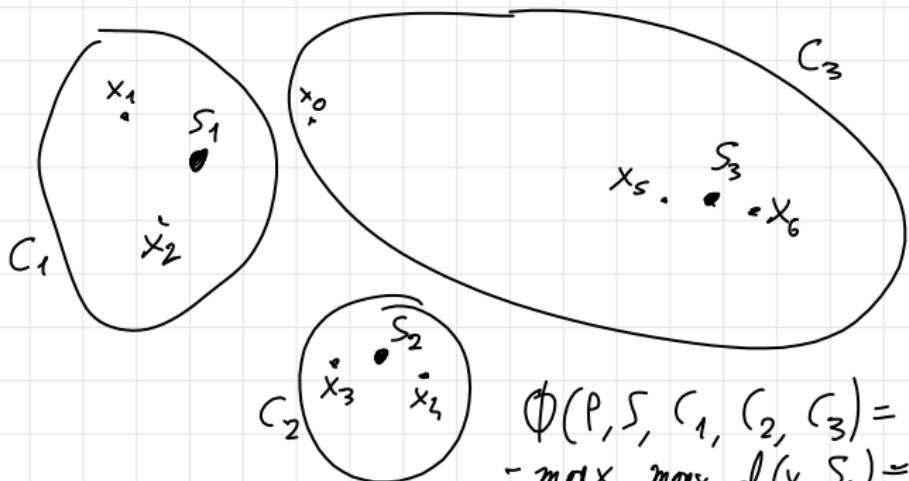
OPTIMAL SOLUTION: S^* (i.e., the feasible solution which minimizes the appropriate objective function)

We also define:

- $\Phi_{\text{kcenter}}^{\text{opt}}(P, k) = \Phi_{\text{kcenter}}(P, S^*)$.
- $\Phi_{\text{kmeans}}^{\text{opt}}(P, k) = \Phi_{\text{kmeans}}(P, S^*)$.
- $\Phi_{\text{kmedian}}^{\text{opt}}(P, k) = \Phi_{\text{kmedian}}(P, S^*)$.

Why do the problems require to return only the centers and not the whole clustering?

$$\phi_{k\text{-center}}(P, S), |S|=n$$



$$\begin{aligned}\phi(P, S, C_1, C_2, C_3) &= \\ &= \max_{C_i \in \{C_1, C_2, C_3\}} \max_{x \in C_i} d(x, S_i) = \\ &= \max \{d(x_1, S_2), d(x_0, S_3), \\ &\quad d(x_3, S_2)\} = d(x_0, S_3)\end{aligned}$$

qui posso migliorare sì.
combinando centri

Assignment primitive: Assign(P,S)

Input: P, S ($|S|=n$)

Output: $\forall x \in P$, return (x, s_x) (s_x : punto di S più vicino a x)

$$(x, \underset{y \in S}{\operatorname{arg\,min}} d(x, y))$$

Assignment primitive: Assign(P, S)

Exercise \Rightarrow ~~do you~~

Consider a pointset P of N points represented as a key-value pairs (ID_x, x) , where ID_x is a distinct integer in $[0, N)$, and x is a point. Let also S be a set of k centers. Considering S as global data, show that $\text{Assign}(P, S)$ can be implemented in MapReduce in 1 round using $M_L = O(k)$.

Assignment primitive: Assign(P,S)

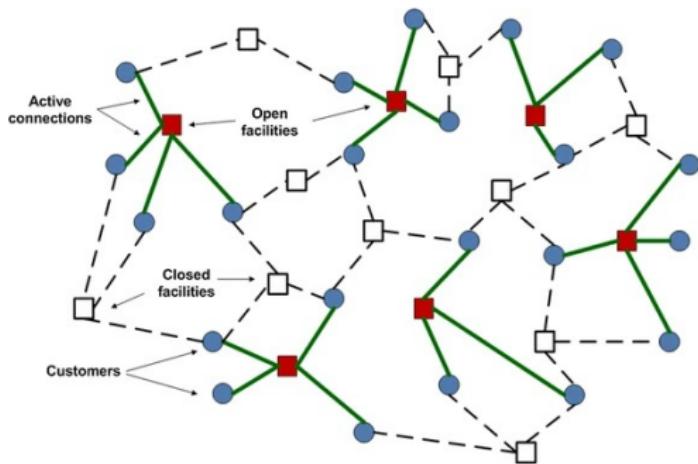
Assignment primitive: Assign(P,S)

Observations

- The k-center/means/median problems are NP-hard \Rightarrow it is impractical to search for the optimal solution S^* .
- There are several efficient approximation algorithms that in practice return good-quality solutions. However, dealing efficiently with large inputs is still a challenge!
- How do we decide which problem suites better our needs?
 - k-center is useful when we need to guarantee that every point is close to a center (e.g., if centers represent locations of crucial facilities or if they are needed as "representatives" for the data). As we will see later, it is sensitive to noise.
 - k-means/median provide guarantees on the average (square) distances. k-means is more sensitive to noise but there exist faster algorithms to solve it.

Observations

- k-center and k-median belong to the family of facility-location problems



k-center

Farthest-First Traversal: algorithm

moltissimple

- Popular 2-approximation sequential algorithm developed by T.F. Gonzalez [Theoretical Computer Science, 38:293-306, 1985]
- Simple and, somewhat fast, implementation when the input fits in main memory.
- Powerful primitive for extracting samples for further analyses

Input Set P of N points from a metric space (M, d) , integer $k > 1$

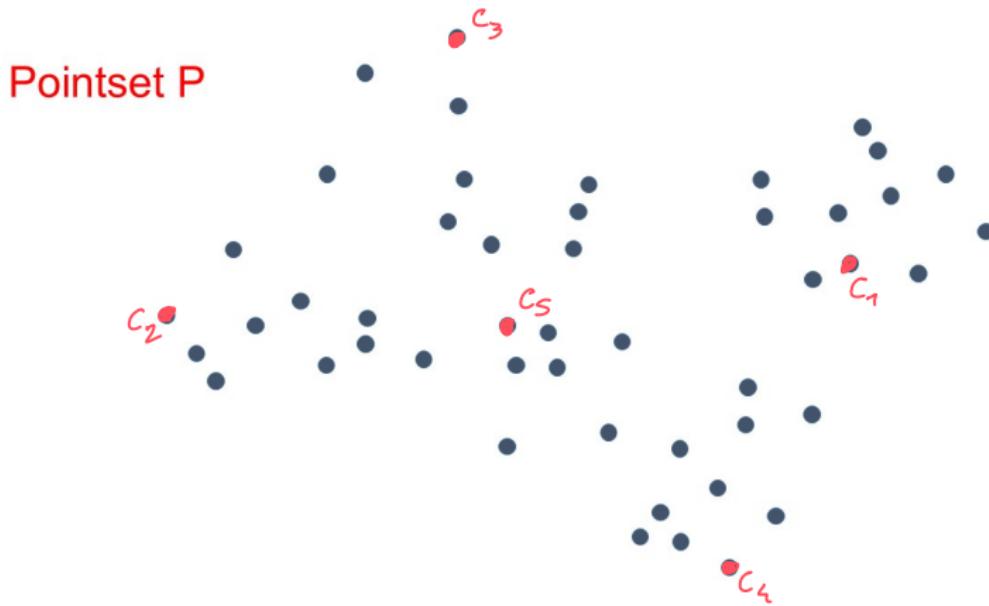
Output A set S of k centers which is a good solution to the k-center problem on P (i.e., $\Phi_{k\text{center}}(P, S)$ “close” to $\Phi_{k\text{center}}^{\text{opt}}(P, k)$)

Farthest-First Traversal: algorithm

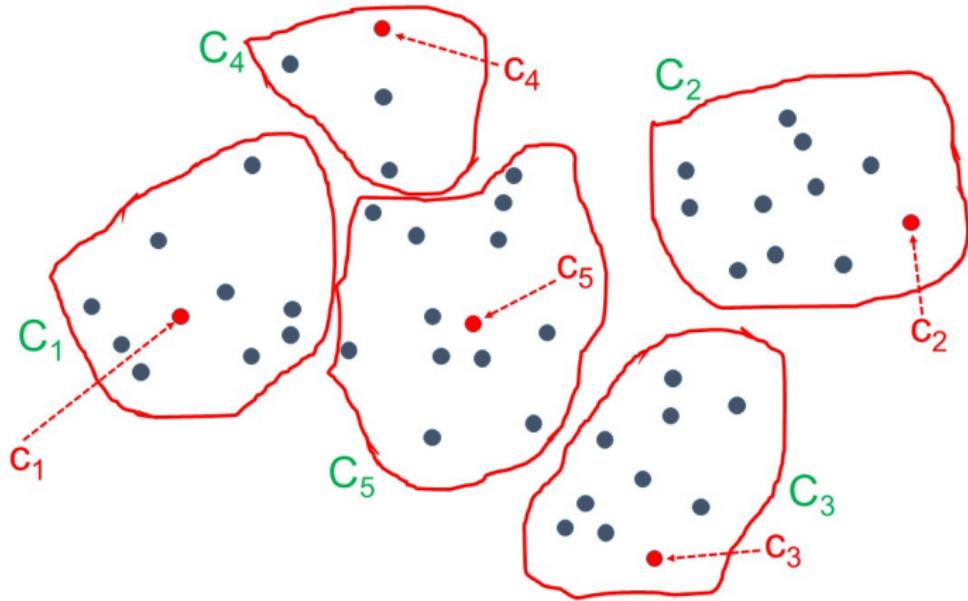
```
 $S \leftarrow \{c_1\}$  //  $c_1 \in P$  arbitrary point  
for  $i \leftarrow 2$  to  $k$  do  
    Find the point  $c_i \in P - S$  that maximizes  $d(c_i, S)$   
     $S \leftarrow S \cup \{c_i\}$   
return  $S$ 
```

Observation: The best clustering around the centers of S can be computed by invoking $\text{Assign}(P, S)$. In fact, *the assignment of each point to the closest center can be easily maintained in every iteration of the for-loop.*

Farthest-First Traversal: example ($k = 5$)



Farthest-First Traversal: example ($k = 5$)



Exercise

Show that the Farthest-First Traversal algorithm can be implemented to run in $O(N \cdot k)$ time.

Hint: make sure that in each iteration i of the for-loop each point $p \in P - S$ knows its closest center among c_1, c_2, \dots, c_{i-1} and the distance from such a center.

Farthest-First Traversal: analysis

Theorem

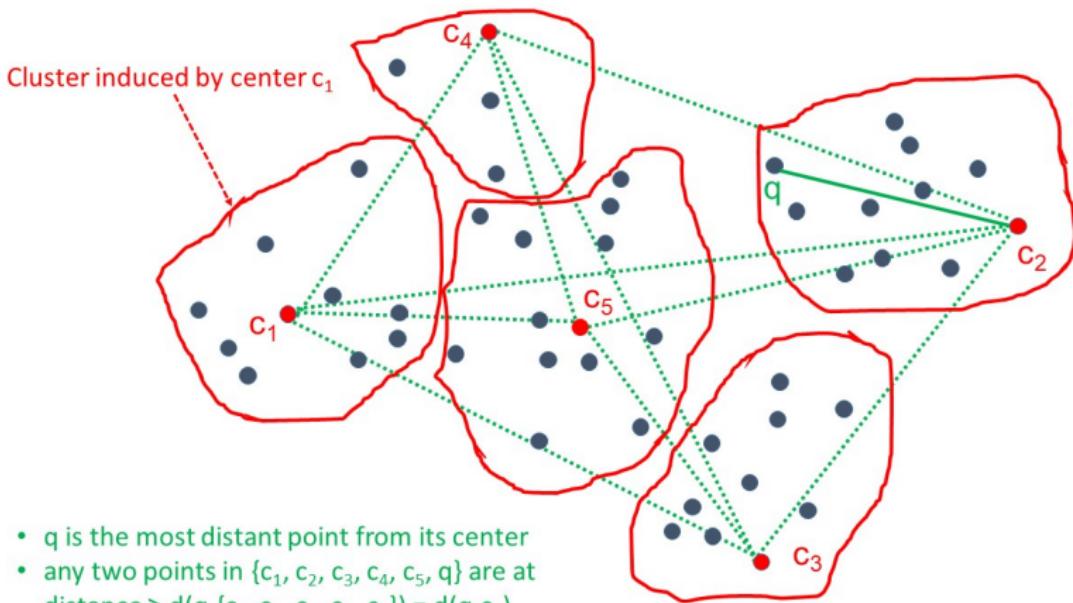
Let S be the set of centers returned by running Farthest-First Traversal on P . Then:

$$\Phi_{k\text{center}}(P, S) \leq 2 \cdot \Phi_{k\text{center}}^{\text{opt}}(P, k).$$

That is, Farthest-First Traversal is a 2-approximation algorithm.

Proof of Theorem

Proof of Theorem



- q is the most distant point from its center
- any two points in $\{c_1, c_2, c_3, c_4, c_5, q\}$ are at distance $\geq d(q, \{c_1, c_2, c_3, c_4, c_5\}) = d(q, c_2)$

Proof of Theorem

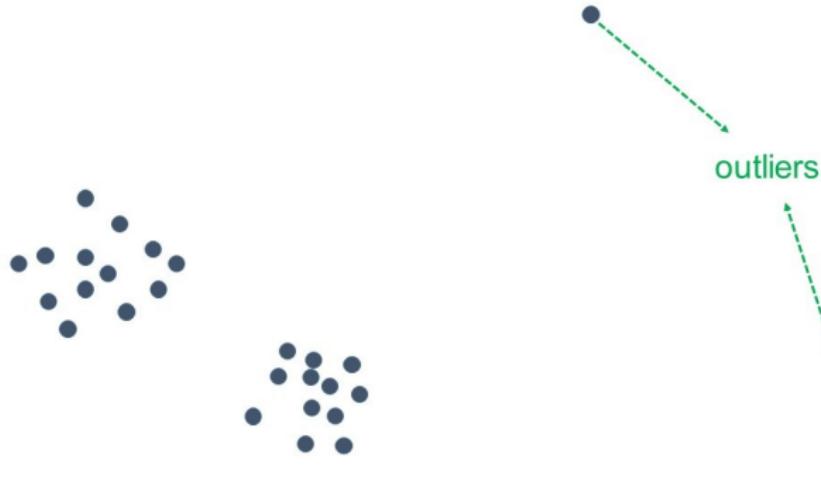
Observations on k-center clustering

- The k-center objective focuses on worst-case distance of points from their closest centers.
- Farthest-First Traversal's approximation guarantees are almost the best one can obtain in practice. It was proved that computing a c -approximate solution to k-center is NP-hard for any fixed $c < 2$.
- The k-center objective is very sensitive to noise. For noisy datasets (e.g., with outliers), the clustering which optimizes the k -center objective may obfuscate some “natural” clustering inherent in the data. See example in the next slide.

Example: noisy pointset



Example: noisy pointset



k-center clustering for big data

Observation. Farthest-First Traversal requires $k - 1$ scans of the pointset P : impractical for massive P and k not small.

How can we compute a “good” solution to k-center for a pointset P which is too large for a single machine?

We resort to the composable coresets technique.

Application of composable coresets technique to k-center clustering

MapReduce-Farthest-First Traversal

Let P be a set of N points (N large!) from a metric space (M, d) , and let $k > 1$ be an integer.

Algorithm MR-Farthest-First Traversal

- **Round 1:**
 - Map Phase: Partition P arbitrarily into ℓ subsets of equal size P_1, P_2, \dots, P_ℓ .
 - Reduce Phase: for every $i \in [1, \ell]$ separately, run Farthest-First Traversal on P_i to determine a set $T_i \subseteq P_i$ of k centers.
- **Round 2:**
 - Map Phase: empty.
 - Reduce Phase: gather the coresset $T = \bigcup_{i=1}^{\ell} T_i$ (of size $\ell \cdot k$) and run, using a single reducer, Farthest-First Traversal on T to determine a set $S = \{c_1, c_2, \dots, c_k\}$ of k centers, and return S as output.

Analysis of MR-Farthest-First Traversal

Theorem

The 2-round MR-Farthest-First Traversal algorithm can be implemented using local space $M_L = O(\sqrt{Nk})$ and aggregate space $M_L = O(N)$.

Analysis of MR-Farthest-First Traversal

Analysis of MR-Farthest-First Traversal

Analysis of MR-Farthest-First Traversal

Let T be the union of the coresets T_i computed by MR-Farthest-First Traversal on input P . The following lemma establishes the quality of T .

Lemma

For every $x \in P$ we have

$$d(x, T) \leq 2 \cdot \Phi_{\text{kcenter}}^{\text{opt}}(P, k).$$

Proof of Lemma

Analysis of MR-Farthest-First Traversal

Theorem

Let S be the set of k centers returned by running MR-Farthest-First Traversal on P . Then:

$$\Phi_{k\text{center}}(P, S) \leq 4 \cdot \Phi_{k\text{center}}^{\text{opt}}(P, k).$$

That is, MR-Farthest-First Traversal is a 4-approximation algorithm.

Proof of Theorem

Proof of Theorem

Proof of Theorem

Observations on MR-Farthest-First Traversal

- Farthest-First Traversal provides good coresets T_i 's, hence a good final coreset T since it ensures that any point not belonging to T is well represented by some coreset point.
- MR-Farthest-First Traversal is able to handle very large pointsets and the the final approximation is not too far from the best achievable one.

Low-dimensional pointsets

When P has low dimensionality the quality of the solution returned by MR-Farthest-First Traversal can be made arbitrarily close to 2 by selecting a slightly larger coresset, while still ensuring sublinear local space and linear aggregate space.

Does a random sample provide a good coresset?

Let $T \subseteq P$ be a coresset of $|T| = \sqrt{Nk}$ points, selected at random from P independently, with replacement and with uniform probability. Consider a set S of k centers computed by running Farthest-First Traversal on T .

Is S a good solution to k -center on P ?

Does a random sample provide a good coresset?

Does a random sample provide a good coresset?

k-center as a coresset primitive

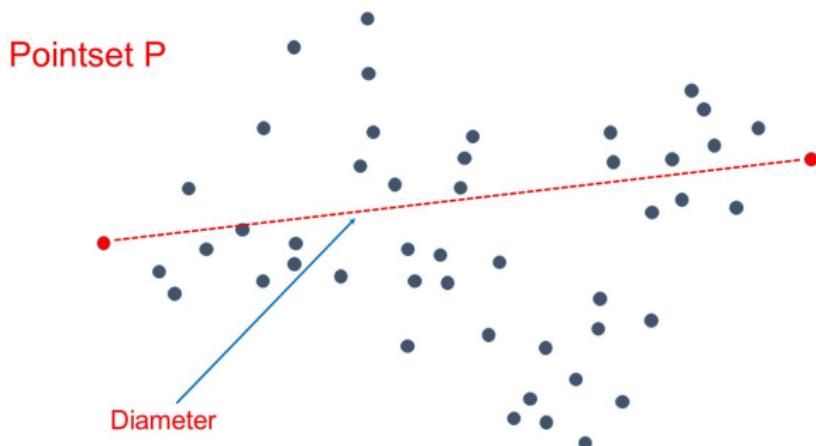
In MR-FFT, k-center is employed to extract the coresset for solving k-center itself. Now, we will see that it is also useful to extract coresets for other problems as well.

Diameter

Problem: given a set P of N points from a metric space (M, d) determine its diameter

$$d_{\max} = \max_{x,y \in P} d(x, y),$$

i.e., the maximum distance between two points.



Exact diameter computation

Sadly, the computation of the exact diameter requires almost quadratic operations (except for special cases such as the low-dimensional Euclidean spaces), hence it is impractical for very large pointsets.

Exercise

Design an MR algorithm to compute the exact diameter of a set P of N points, which requires $R = O(1)$, $M_L = O(\sqrt{N})$ and $M_A = O(N^2)$. Assume that P is initially provided as the set of pairs (i, x_i) , for $0 \leq i < N$, where the x_i 's are the points.

2-approximation to the diameter

For an arbitrary $x_i \in P$ define

$$d_{\max}(i) = \max\{d(x_i, x_j) : 0 \leq j < N\}.$$

Lemma

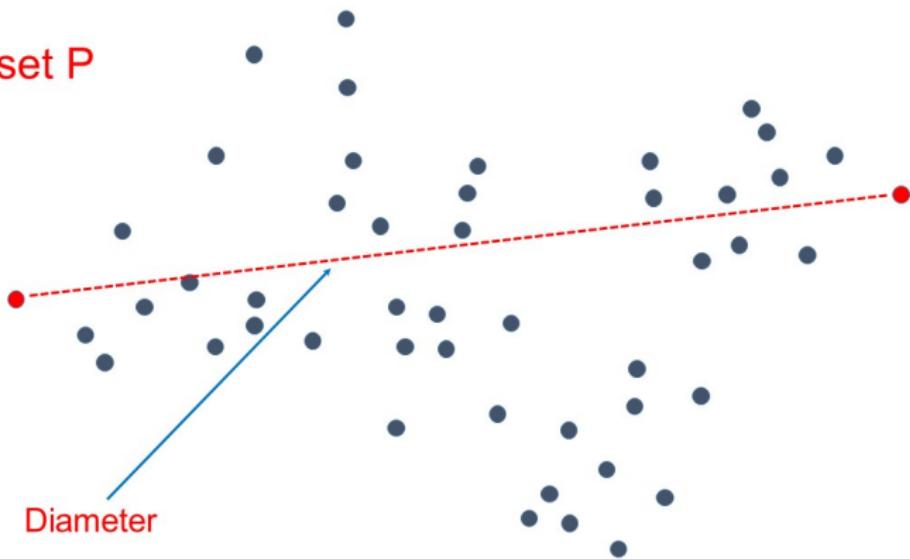
For any $0 \leq i < N$ we have $d_{\max} \in [d_{\max}(i), 2d_{\max}(i)]$.

Exercise

Show that for any arbitrary $0 \leq i < N$, the value $d_{\max}(i)$ can be computed in MapReduce using 2 rounds, $M_L = O(\sqrt{N})$ and $M_A = O(N)$.

Proof of Lemma

Pointset P



Proof of Lemma

Proof of Lemma

Better, coreset-based diameter approximation

Coreset-based approximation:

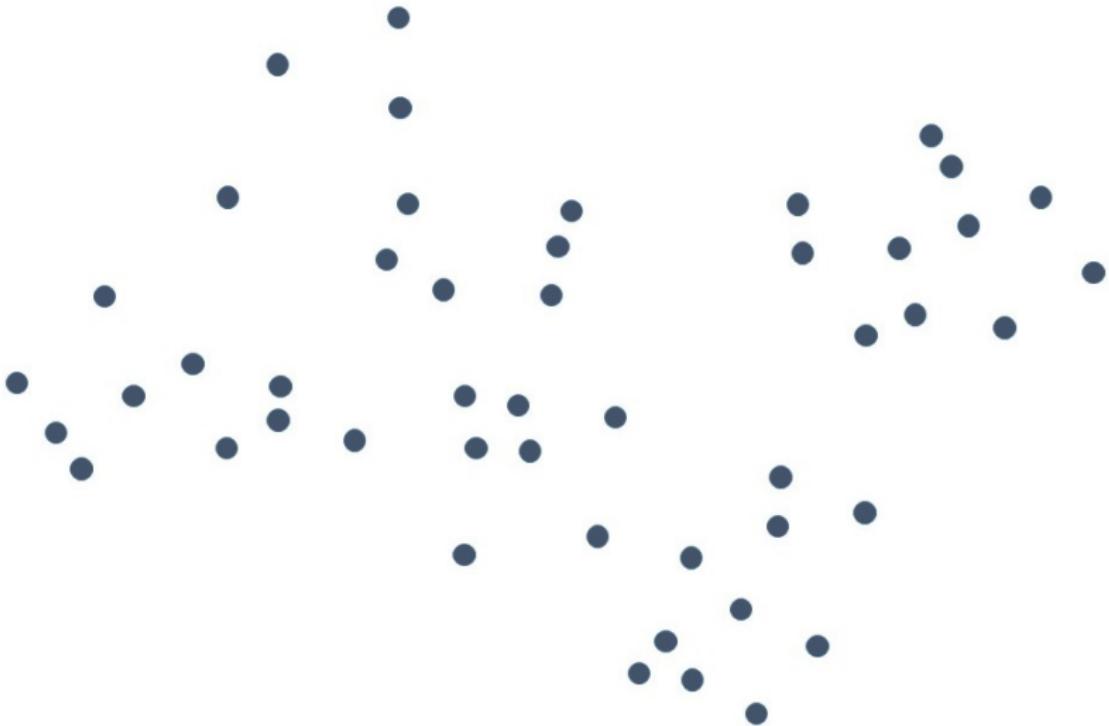
- Fix a suitable $k \geq 2$.
- Extract a coreset $T \subset P$ of size k by running a k-center clustering algorithm on P and taking the k cluster centers as set T .
- Return $d_T = \max_{x,y \in T} d(x,y)$ as an approximation of d_{\max} .

Can the approximation be computed efficiently?

If $k = O(1)$, d_T can be computed

- Sequentially: in $O(N)$ time (using Farthest-First Traversal)
- In MapReduce: in 2 rounds, with local space $M_L = O(\sqrt{N})$ and aggregate space $M_L = O(N)$ (through MR-Farthest-First Traversal)

Is d_T a good approximation of d_{\max} ?



Diversity maximization

Objective:

For a given dataset



Diversity maximization

Objective:

Determine the **most diverse** subset of size k (for a given small k)



Applications

Aggregator websites (e.g., Google News)

- Selection of documents for home page
- Diversified set of representative docs

E-commerce

- Selection of consideration set: products returned by portal upon user query
- Diversity of returned products correlates to user satisfaction

Facility location

- Selection of non-competing franchise store locations
- Selection of strategic facility locations (to avoid simultaneous attacks)

Formal definition of the problem

Diversity maximization problem: remote clique variant

Given a set P of points from a metric space (M, d) and a positive integer $k < |P|$, return a subset $S \subset P$ of k points, which maximizes the diversity function

$$\text{div}(S) = \sum_{x,y \in S} d(x, y).$$

Observations:

- NP-hard problem
- A c -approximation algorithm is known with $c = 2 - 2/k$ which, however, requires quadratic time (impractical for large inputs!)
- Other variants: different functions $\text{div}(\cdot)$ to maximize (all NP-hard)

Coreset-based approach to diversity maximization

For a given input P , define the value of the optimal solution as:

$$\text{div}^{\text{opt}}(P, k) = \max_{S \subset P, |S|=k} \text{div}(S).$$

Can we extract a good solution from a small coresset $T \subset P$? Yes, as long as T is a **good coresset**. The quality of a coresset is quantified as follows.

Definition: $(1 + \epsilon)$ -coreset

Let $\epsilon \in (0, 1)$ be an accuracy parameter. A subset $T \subset P$ is an $(1 + \epsilon)$ -coreset for the diversity maximization problem on P if

$$\text{div}^{\text{opt}}(T, k) \geq \frac{1}{1 + \epsilon} \text{div}^{\text{opt}}(P, k).$$

Obs.: since we are solving a maximization problem, the smaller ϵ the better the optimal solution in T approximates the optimal solution in P .

Coreset-based approach to diversity maximization

The following fact, establishes an interesting relation between the k-center and the diversity maximization problems.

Fact

$$\Phi_{\text{kcenter}}^{\text{opt}}(P, k) \leq \frac{\text{div}^{\text{opt}}(P, k)}{\binom{k}{2}}$$

This property, is crucially exploited by the coresnet-based approach described (at a high level) in the following slide.

Coreset-based algorithm for diversity maximization

Given P and k :

- Run Farthest-First Traversal to extract a set T' of $h > k$ centers from P , for a suitable value $h > k$.
- Consider the h -clustering induced by T' on P and select k arbitrary points from each cluster (or all points in the cluster if they are less than k).
- Gather the at most $h \cdot k$ points selected from the h clusters into a coresset T , and extract the final solution S by running the best sequential algorithm for diversity maximization on T

What is a good choice for h ?

Coreset-based algorithm for diversity maximization

Coreset-based algorithm for diversity maximization