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 coreset 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.

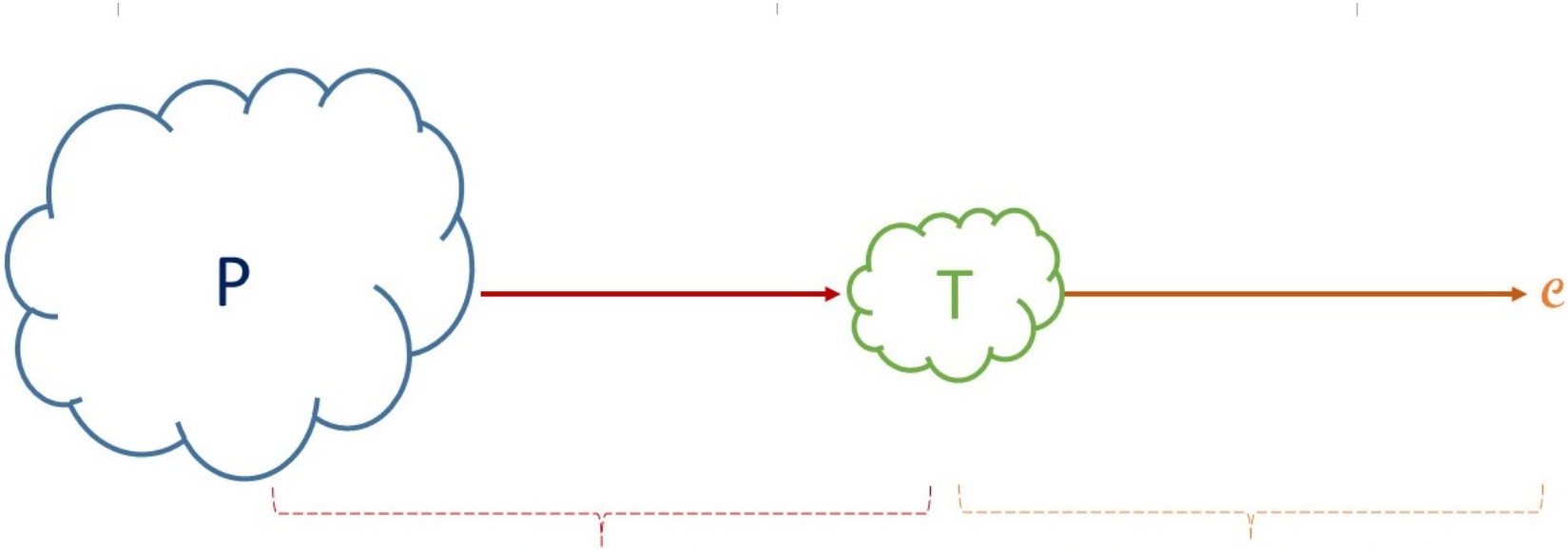
Coreset technique

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 coreset 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.

Coreset technique



Composable Coreset technique

voglio parallelizzare

caso speciale: posso costruire T dividendo input => facile implementare MR

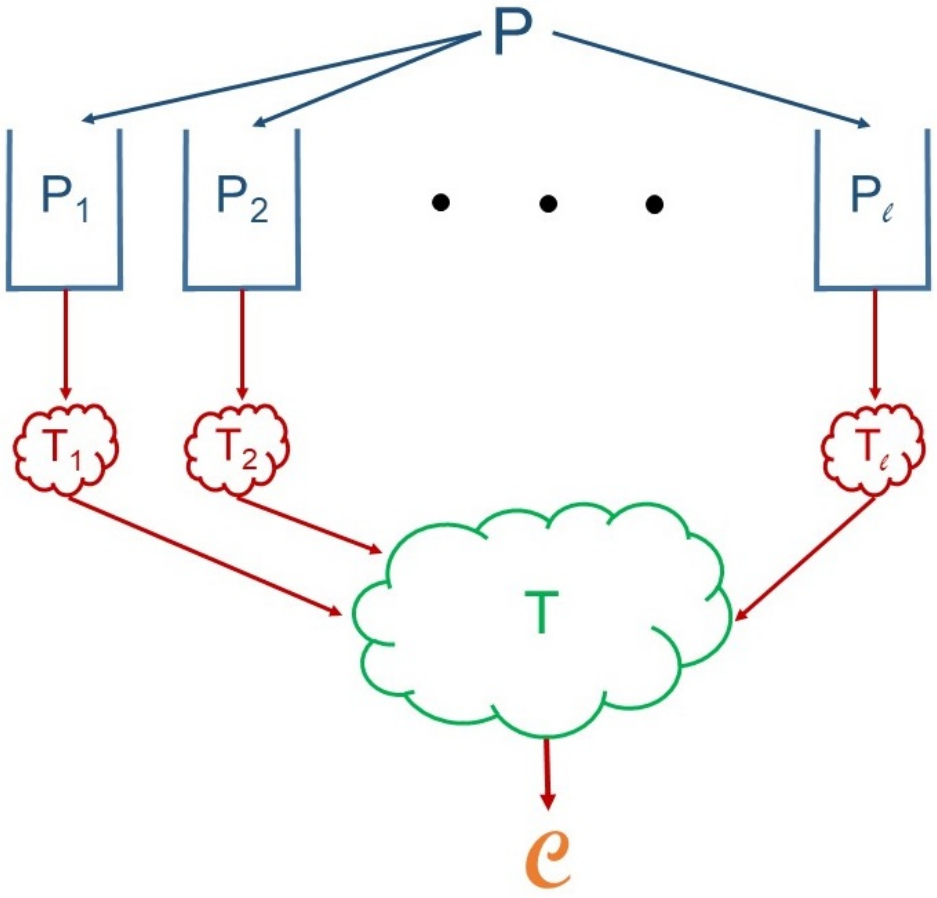
Composable Coreset technique

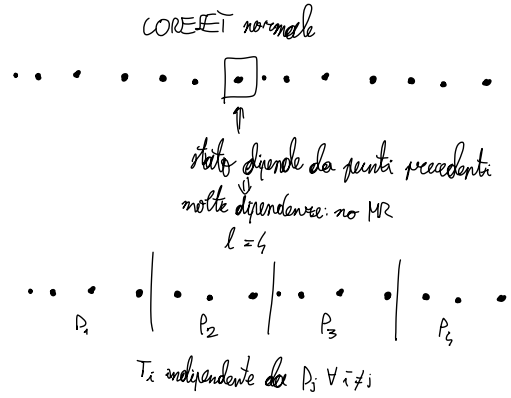
1. Partition P into ℓ subsets $P_1, P_2, \ldots, P_\ell$, and extract a "small" coreset T_i from each P_i, making sure that it represents P_i well. } Map 1° round
2. Run best known (possibly slow) sequential algorithm for Π on $T = \cup_{i=1,\ell} T_i$, rather than on P. } Reduce 1° + Map 2°

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 coreset 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 e

Case study: k-center clustering

non scegliere tra formule costi => vogliamo alg. efficiente per k-center

Why is clustering a relevant case study?

1- molto comune
2- molto costoso

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

cluster analysis in outlier detection

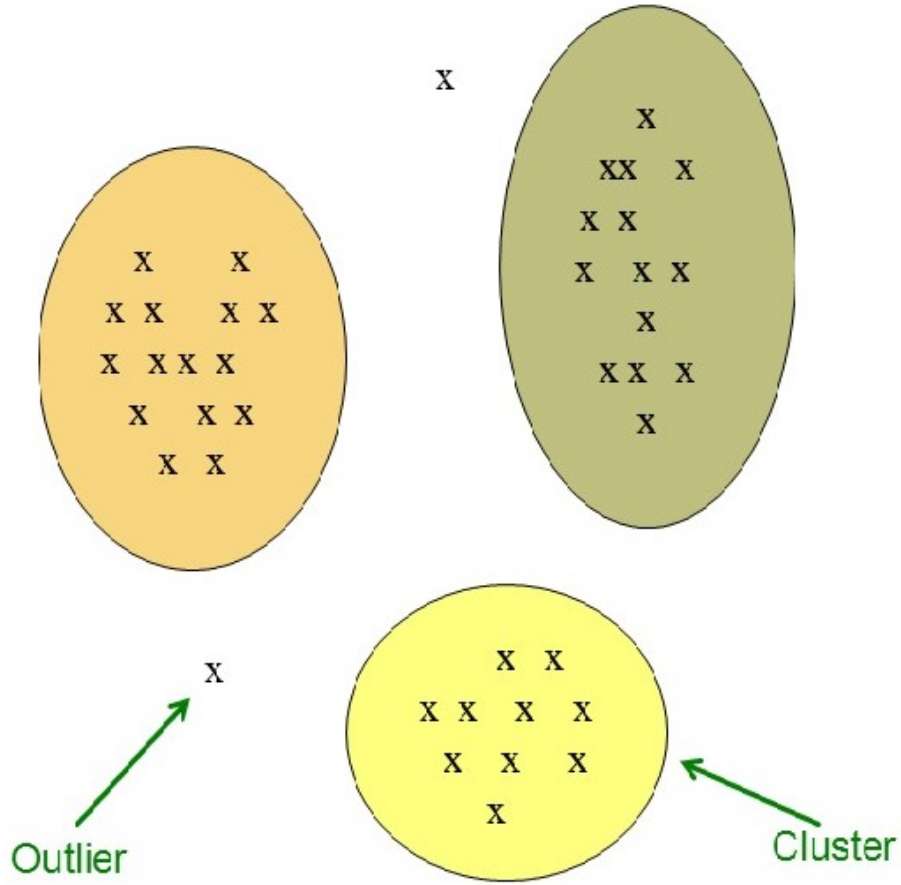


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

The clustering problem is a hard one!

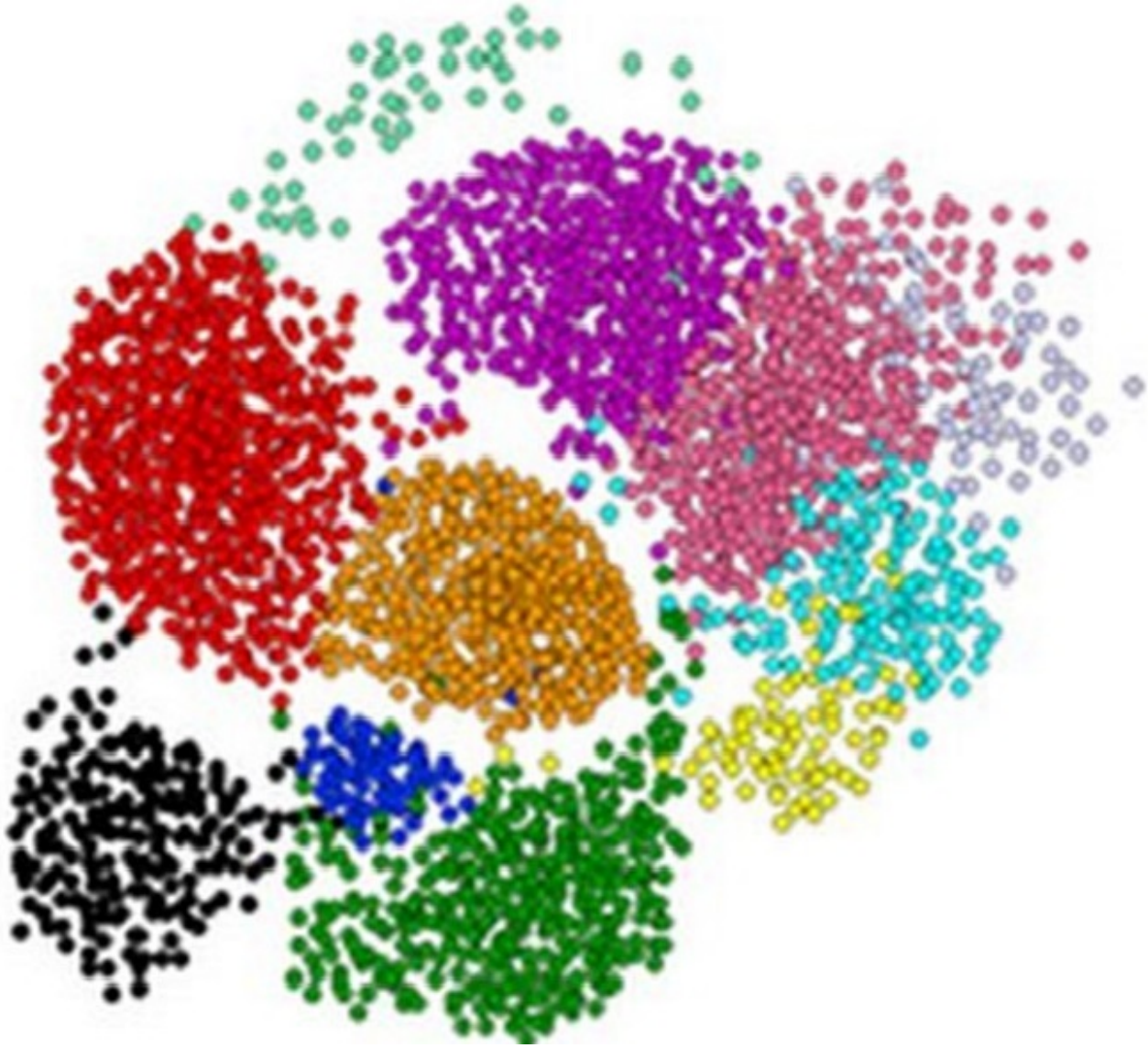


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

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

$$d : M \times M \to \Re_0^+$$

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

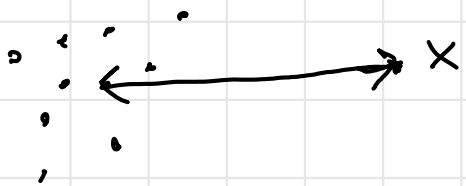
- $d(x, y) \geq 0$;
- $d(x, y) = 0$ if and only if $x = y$;
- $d(x, y) = d(y, x)$; (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$; (triangle inequality)

spesso rilassata ⇒ traiettorie, serie temporali ⇒ PSEUDOMETRICHE

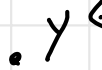
Remark: *The choice of the function may have a significant impact on the effectiveness of the cluster analysis.*

Why is triangle inequality useful in the big data context?

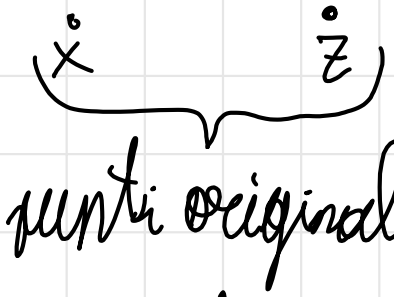
Dobbiamo rappresentare bene input



vogliamo distanza tra x e gli altri ⇒
⇒ mi basta calcolarla da 1 punto



y ⟸ osservazione su z



$d(x,z) \leq d(y,z) + d(x,y)$

$d(x,z)$ ⇓ vogliamo questo

$d(x,y)$ ⇓ possiamo calcolare questo

punti originali

$d(x,z) \sim d(x,y)$ ⇐ se y vicino a z, $d(y,z) \to 0$

Distance functions

Minkowski distances Let $X, Y \in \Re^n$ with $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$. 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 per garantire $d(X,Y) \geq 0$

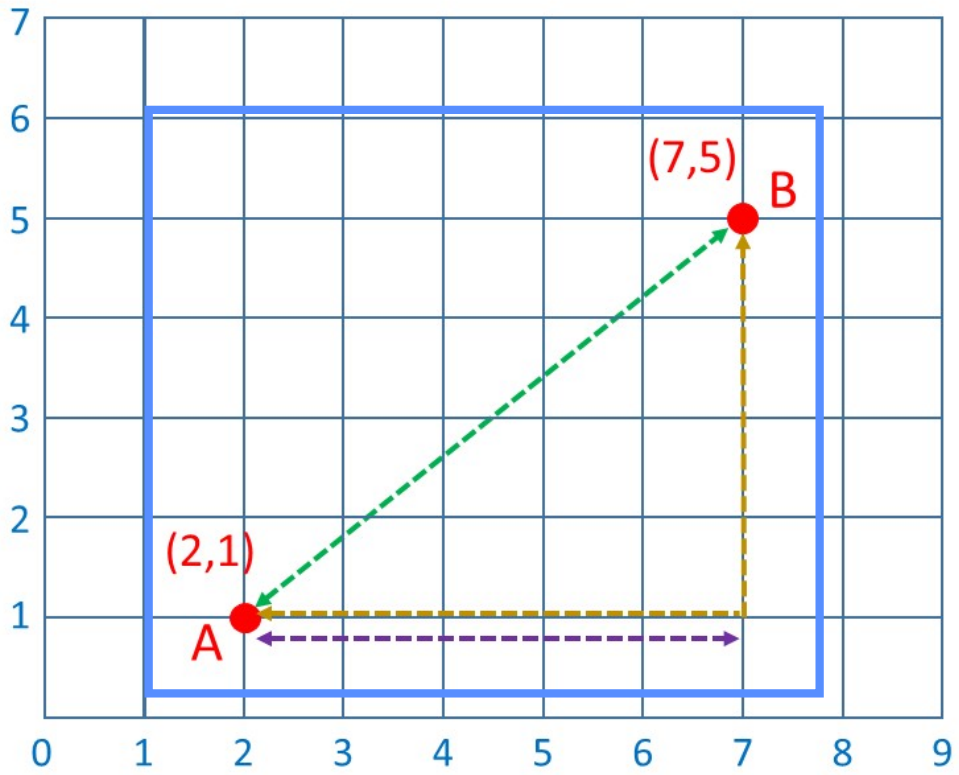
- $r = 2$ (*Euclidean distance*): standard distance in $\Re^n$ (also denoted by $||\cdot||$)
- $r = 1$ (*Manhattan distance*): used in grid-like environments
- $r = \infty$ (*Chebyshev distance*): maximum absolute differences of coordinates (i.e., the limit for r that tends to ∞).

L_∞

termine massimo domina 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:

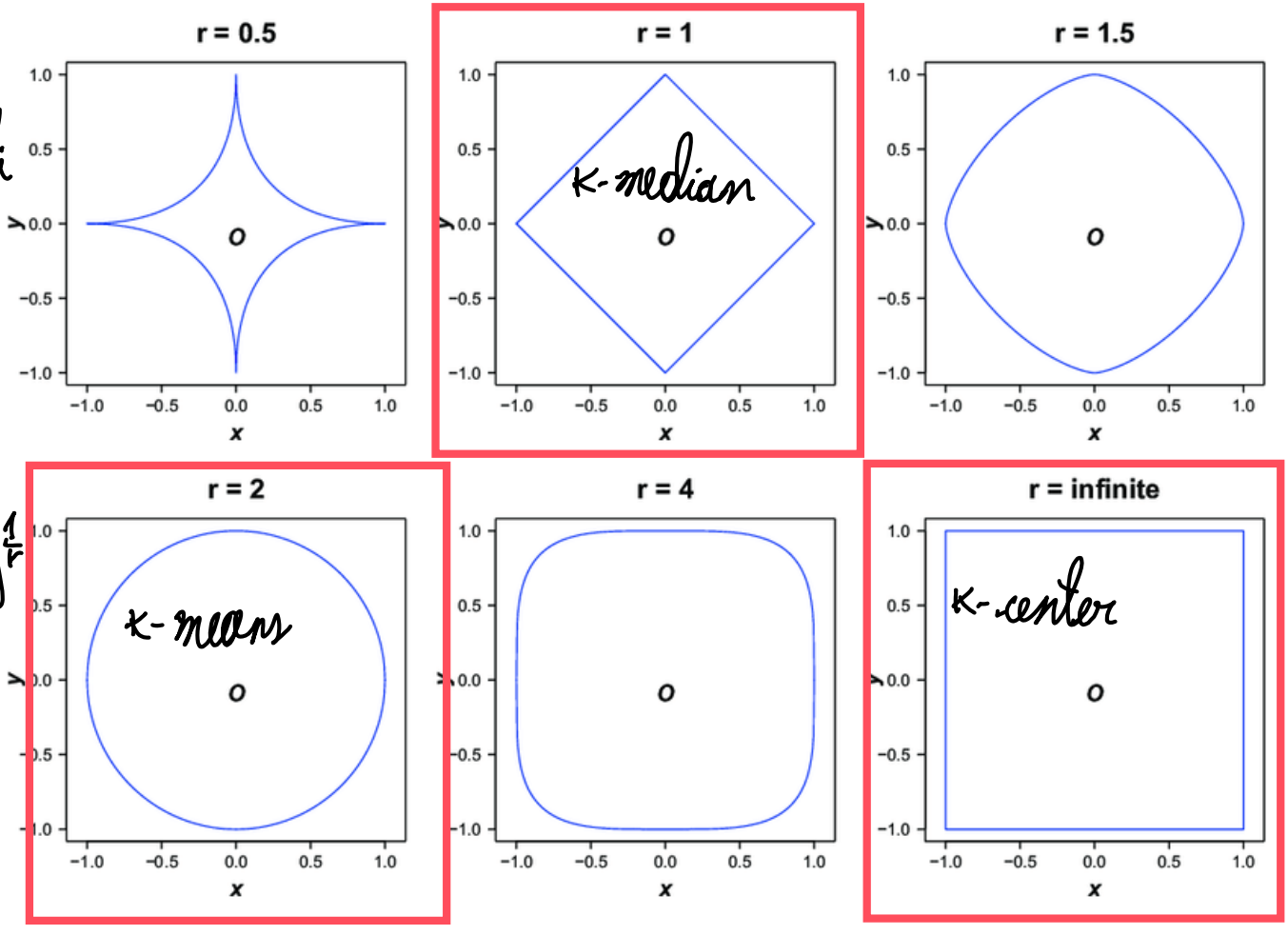
aumentiamo r: più sensibili a outlier

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

$Y = (y_0, 2, \ldots, 2)$

$\downarrow$

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



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 $\Re^n$. Given $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ in $\Re^n$, their angular distance is the angle between them, that is,

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

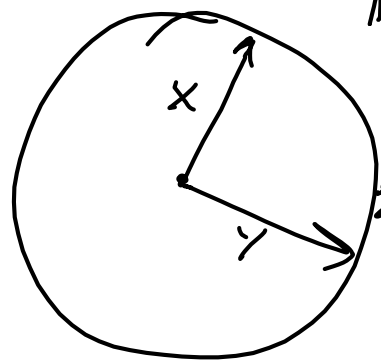
in alcune definizioni

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

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

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

$$= \arccos\left[\frac{3}{6}\right] = 45°$$



raggio 1

$\|x\|_2 = \|y\|_2 = 1$

$X \cdot Y = \cos\theta$

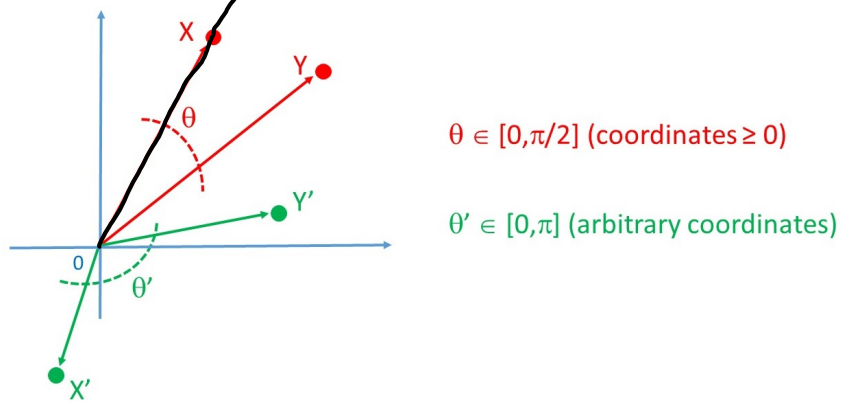
generale: $\|x\| \|y\| \cos\theta$

$\cos\theta = \frac{XY}{\|x\| \|y\|}$

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, $d_{\text{angular}}(X, Y) \in [0, \pi/2]$, 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.

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

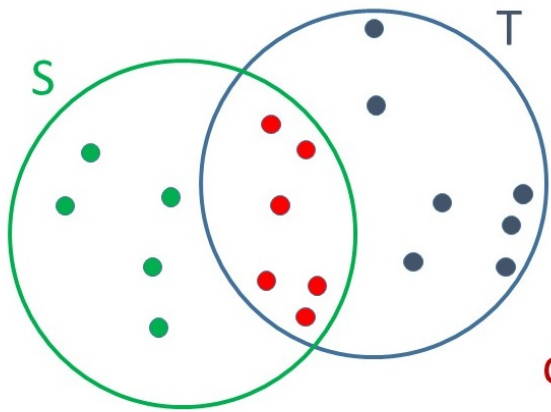
Distance functions

- **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:



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

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 $s_i \Rightarrow$ può essere vuoto o non vuoto

$\Downarrow$

$s_i = \{o_{i1}, o_{i2}, \dots\}$

ϕ = funzione obiettivo

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

$\Downarrow$

soluzioni

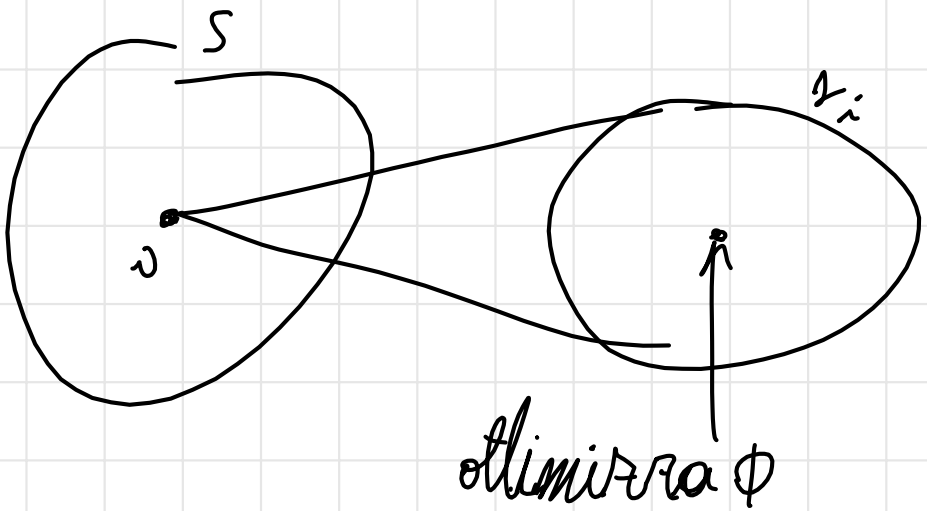
obiettivo: $\forall i \in I$, trovare soluzione $\hat{o} \in s_i$:

$\hat{o} = \underset{o \in s_i}{\text{argmax}}\ \phi(o) \Rightarrow$ problema di massimizzazione

(argmin) (minimizzazione)

Background: optimization problem

$\exists i$: ottimale è data da soluzione $\hat{\sigma}_1, \hat{\sigma}_2, \ldots$



Molto spesso, difficile trovare soluzioni esatte

Background: optimization problem

Background: approximation algorithm

Molto spesso, dato l'input → sol. esatta non ha senso

Problema di max.

i input, $\hat{\sigma}_i$: sol. ottima, σ_i sol. data da algoritmo A

$\phi(\sigma_i) \leq \phi(\hat{\sigma}_i)$ (def. $\hat{\sigma}_i$)

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

$c = 1$: sol. esatta

vogliamo $c \geq 1$ più piccolo possibile

$c = 1 + \varepsilon$ con ε piccolo

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

Background: approximation algorithm

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

$c = O(1)$, $c = O(\log n)$,
$c = O(n^b)$ con b fisso

In questo caso, è detto algoritmo

$\Downarrow$

preferiamo $c = 1 + \varepsilon$

Problema di min.
i input, $\hat{o}_i$ sol. ottima, o_i: sol. data da alg. 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 $O(n) \rightarrow O(\log n)$ $O(n^2) \rightarrow O(n)$
- 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 uncenrtainty, the true optimum may not be well defined.

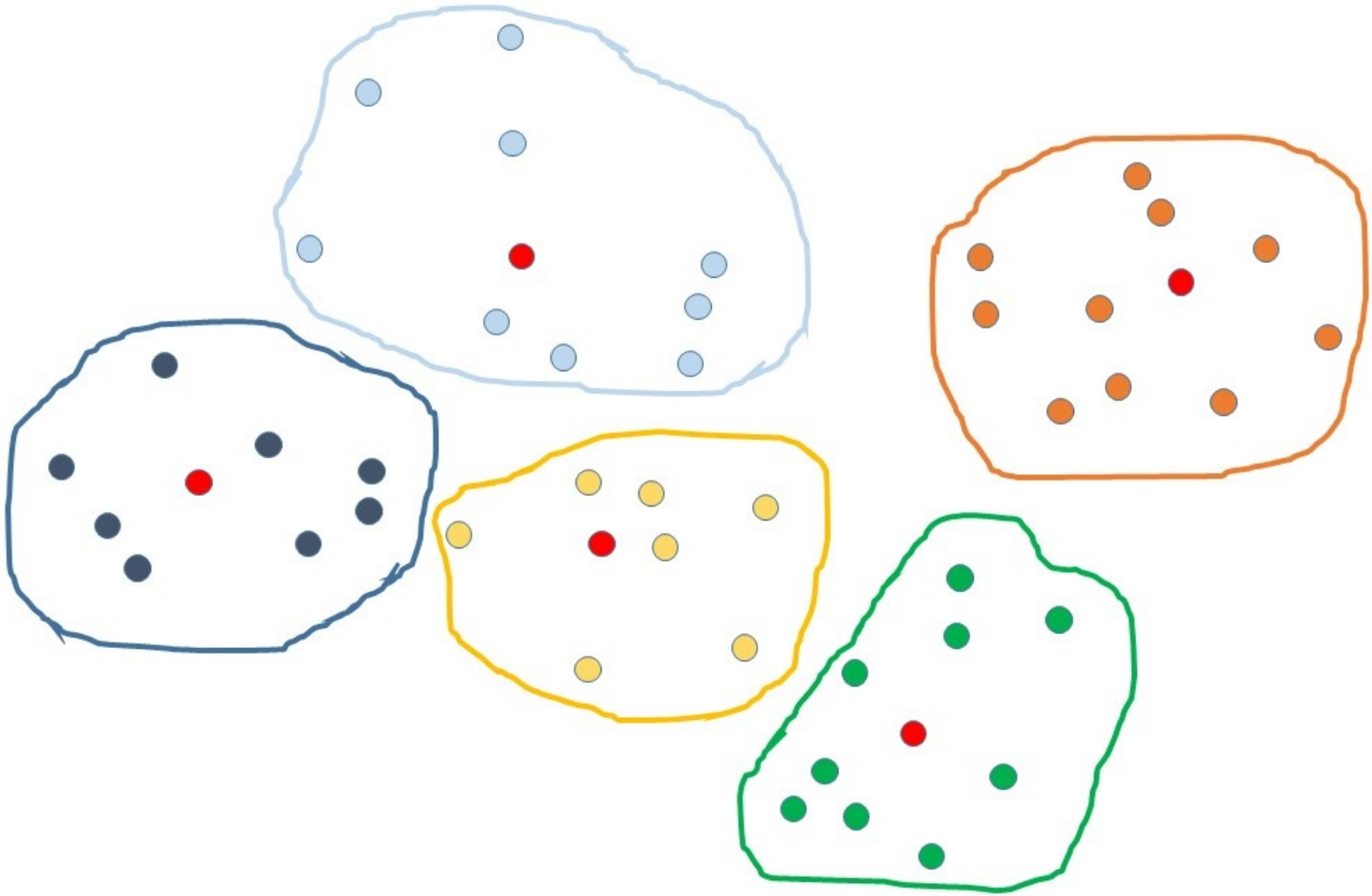
Center-based clustering

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 $\mathcal{C} = (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$ where

- $(C_1, C_2, \ldots, C_k)$ defines a partition of P, i.e., $P = C_1 \cup C_2 \cup \cdots \cup C_k$
- $c_1, c_2, \ldots, c_k$ 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



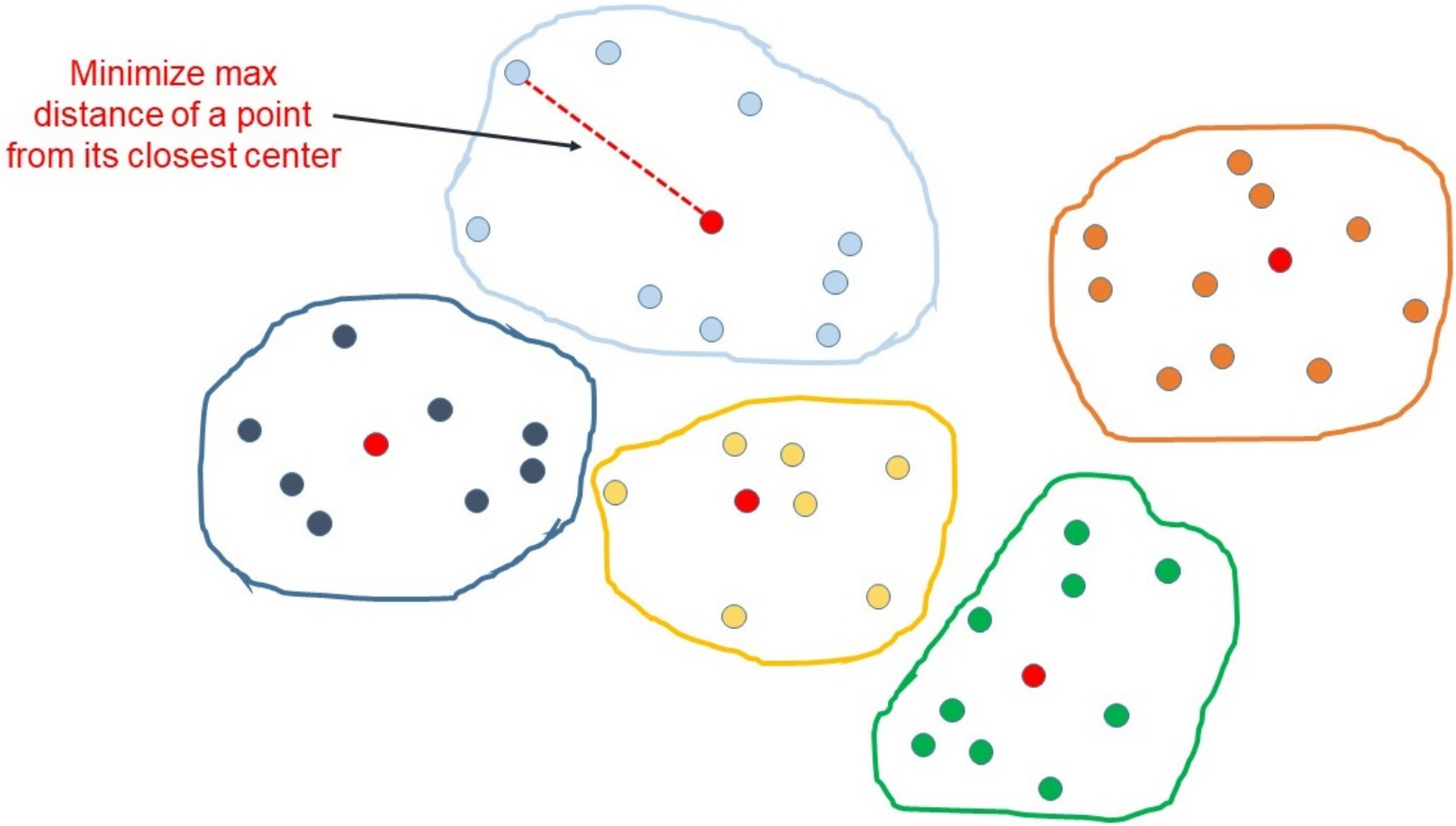
Center-based clustering

For a give 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*

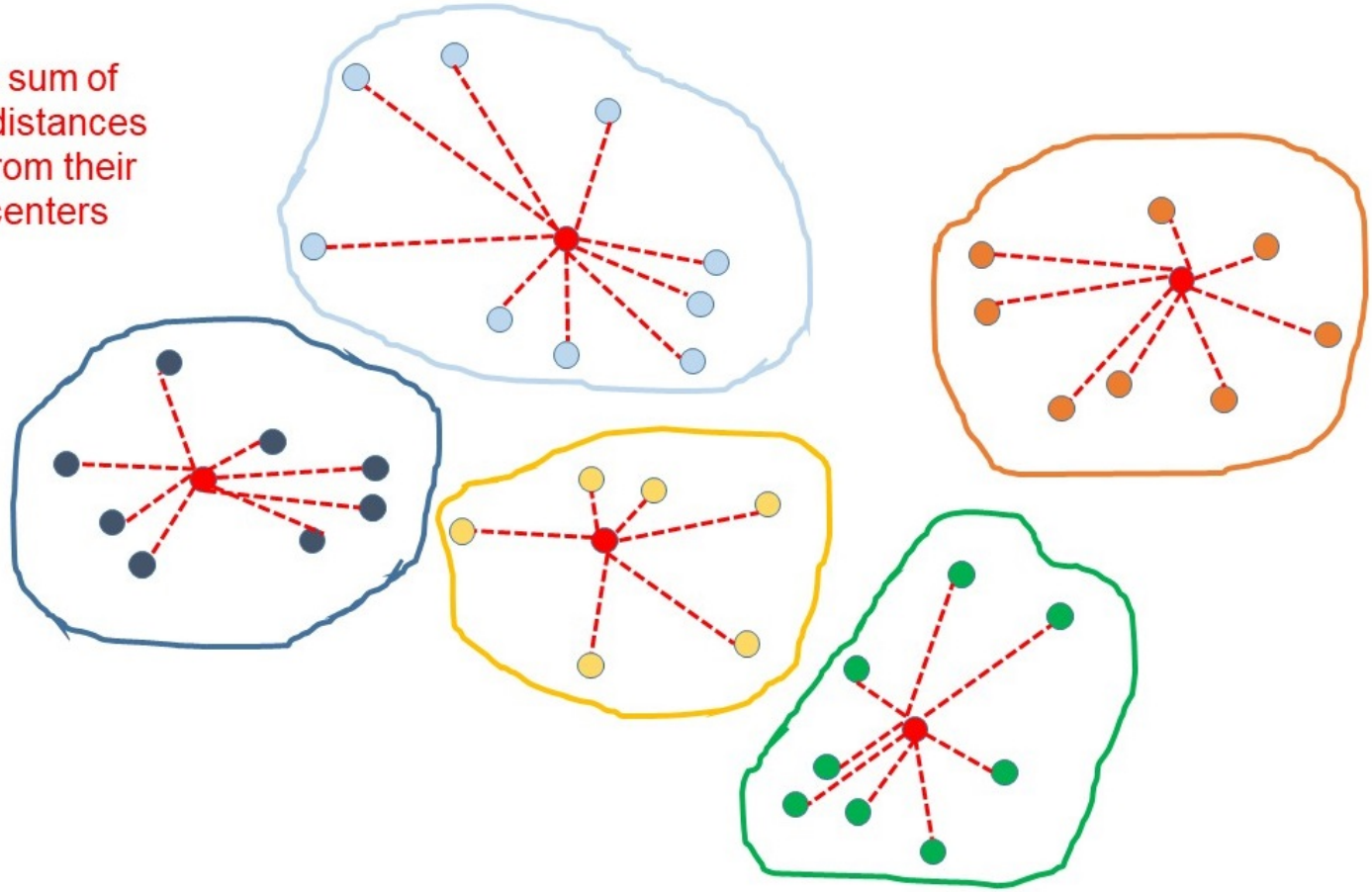
Observation: for k-means and k-median, minimizing the sum of (squared) distances is equivalent to minimizing the average (squared) distance.

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 $P \subset M$ 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

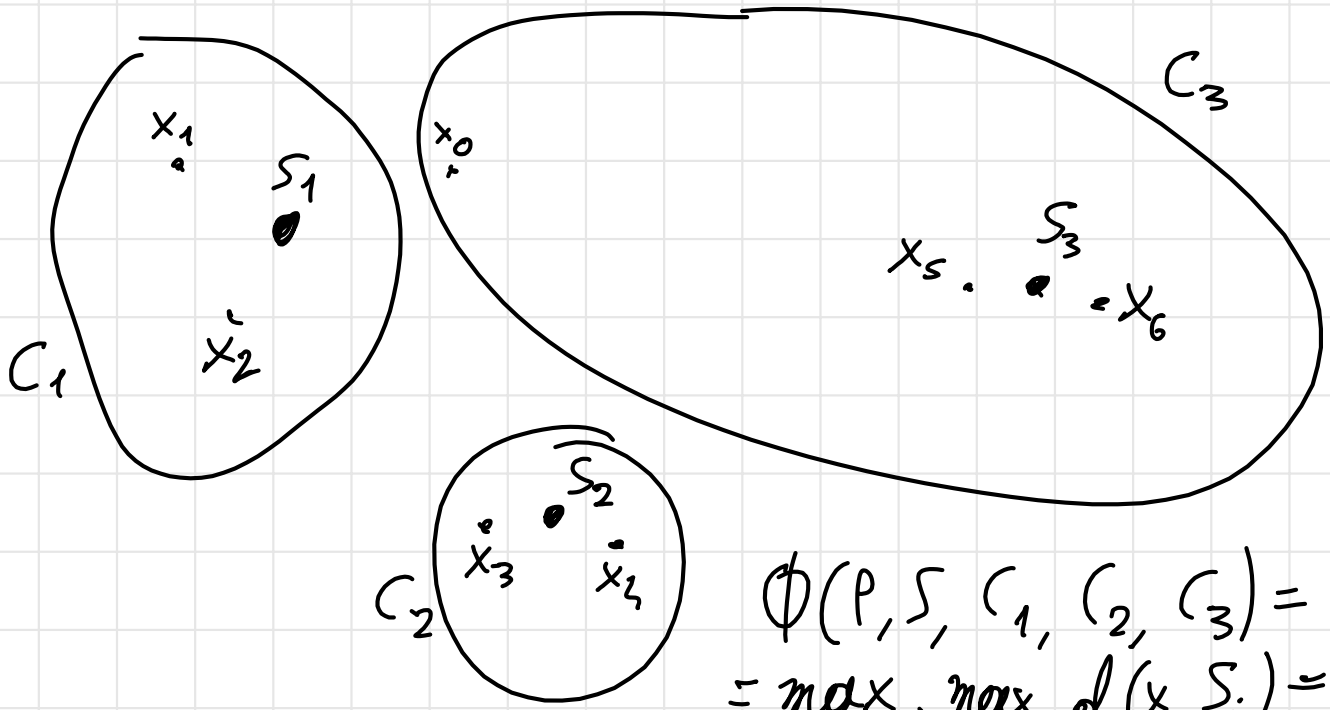
OPTIMAL SOLUTION: S^* (i.e., the feasible solution which minimizes the appropriate objective function)

We also define:

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

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

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



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

qui posso migliorare sol. cambiando 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)

$$\Downarrow$$

$$(x, \operatorname{argmin}_{y \in S}(d(x,y)))$$

Assignment primitive: Assign(P,S)

Exercise => do here

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 Assign(P, S) can be implemented in MapReduce in 1 round using $M_L = O(k)$.

Assignment primitive: Assign(P,S)

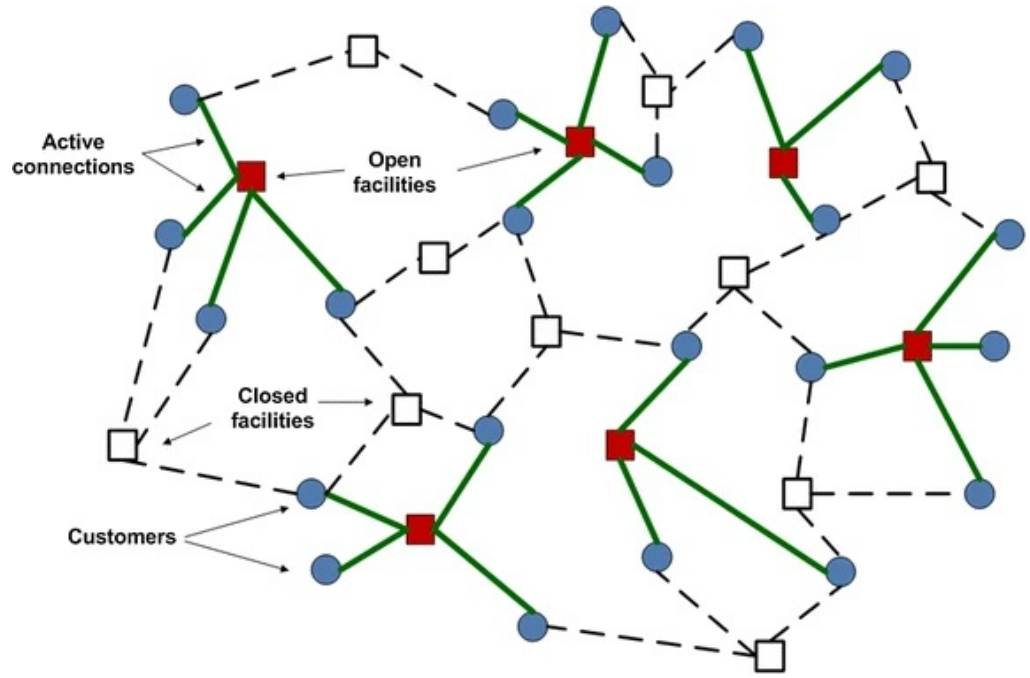
Assignment primitive: Assign(P,S)

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

molto semplice

- 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 $k > 1$

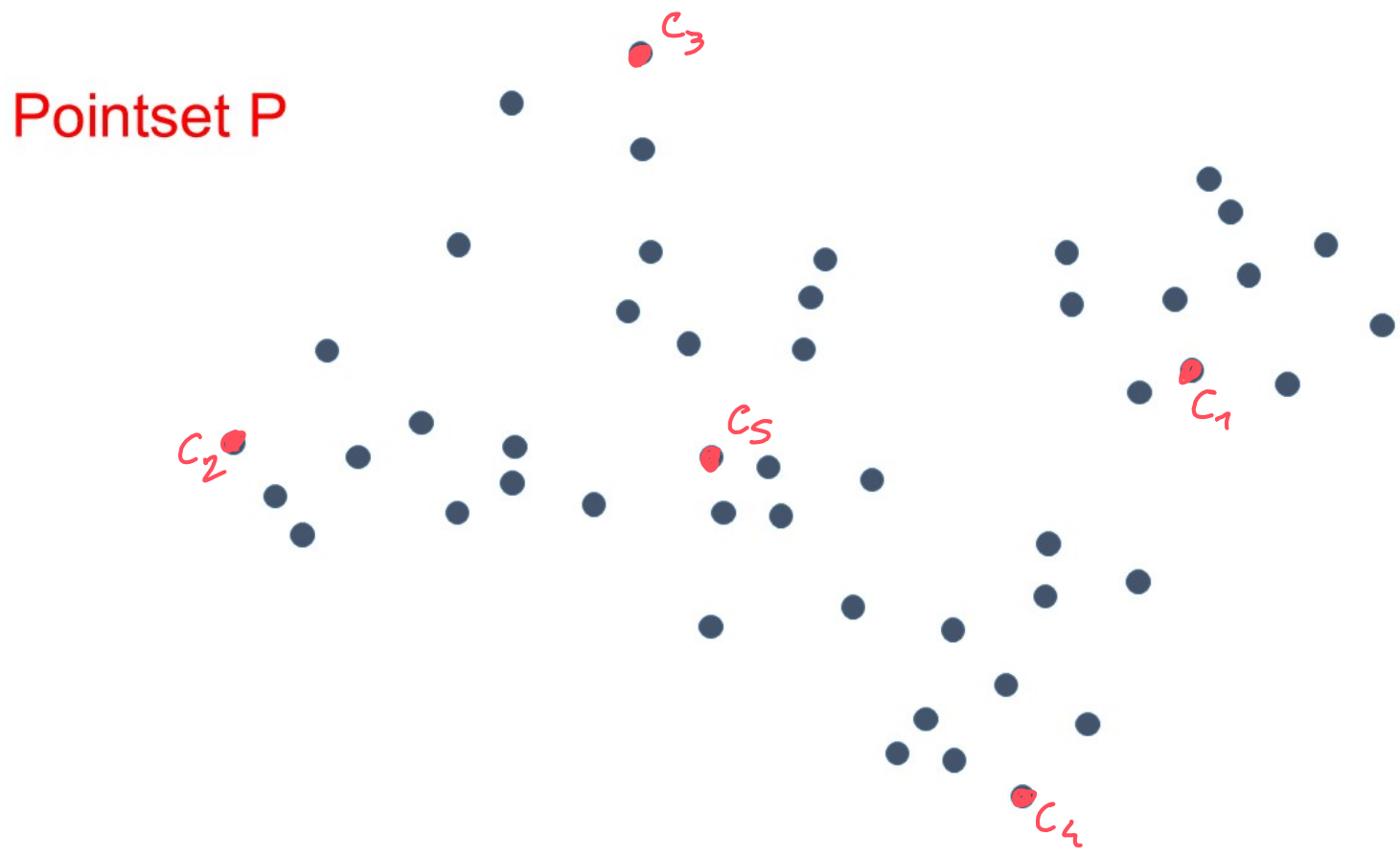
Output A set S of k centers which is a good solution to the k-center problem on P (i.e., $\Phi_{\text{kcenter}}(P, S)$ "close" to $\Phi^{\text{opt}}_{\text{kcenter}}(P, k)$)

Farthest-First Traversal: algorithm

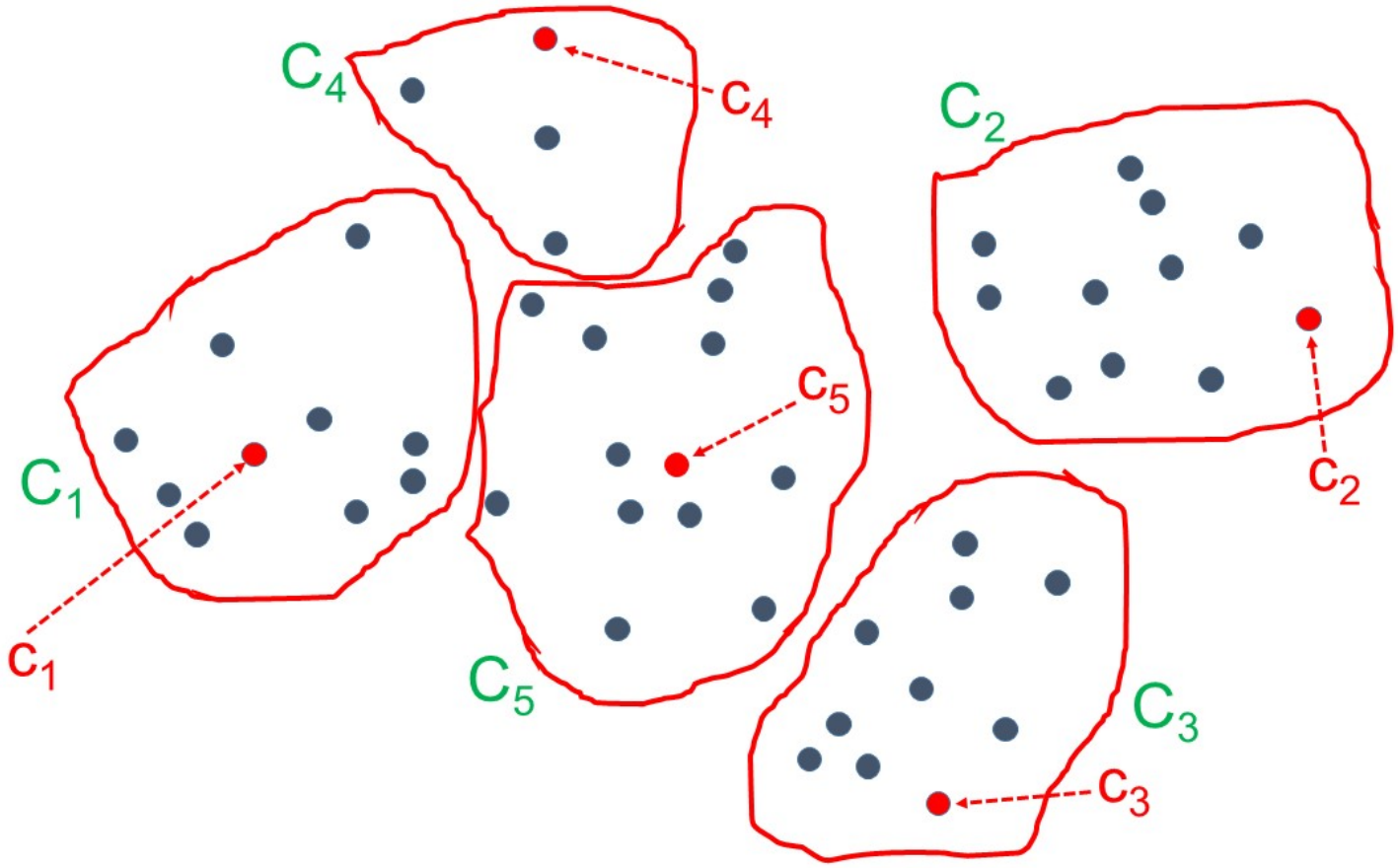
$S \leftarrow \{c_1\}$ // $c_1 \in P$ *arbitrary point*
for $i \leftarrow 2$ **to** k **do**
  `Find the point` $c_i \in P - S$ `that maximizes` $d(c_i, S)$
  $S \leftarrow S \cup \{c_i\}$
return S

Observation: The best clustering around the centers of S can be computed by invoking 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 $c_1, c_2, \ldots, c_{i-1}$ and the distance from such a center.

posso salvare distanza per rendere calcolo $d(c_i, S)$ $O(1)$

Farthest-First Traversal: analysis

Theorem

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

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

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

Proof of Theorem

$S = \{c_1, \ldots, c_k\}$ ritornato da FFT

q = punto ottenuto da iterazione extra di FFT $(q := c_{k+1})$

$\Downarrow$

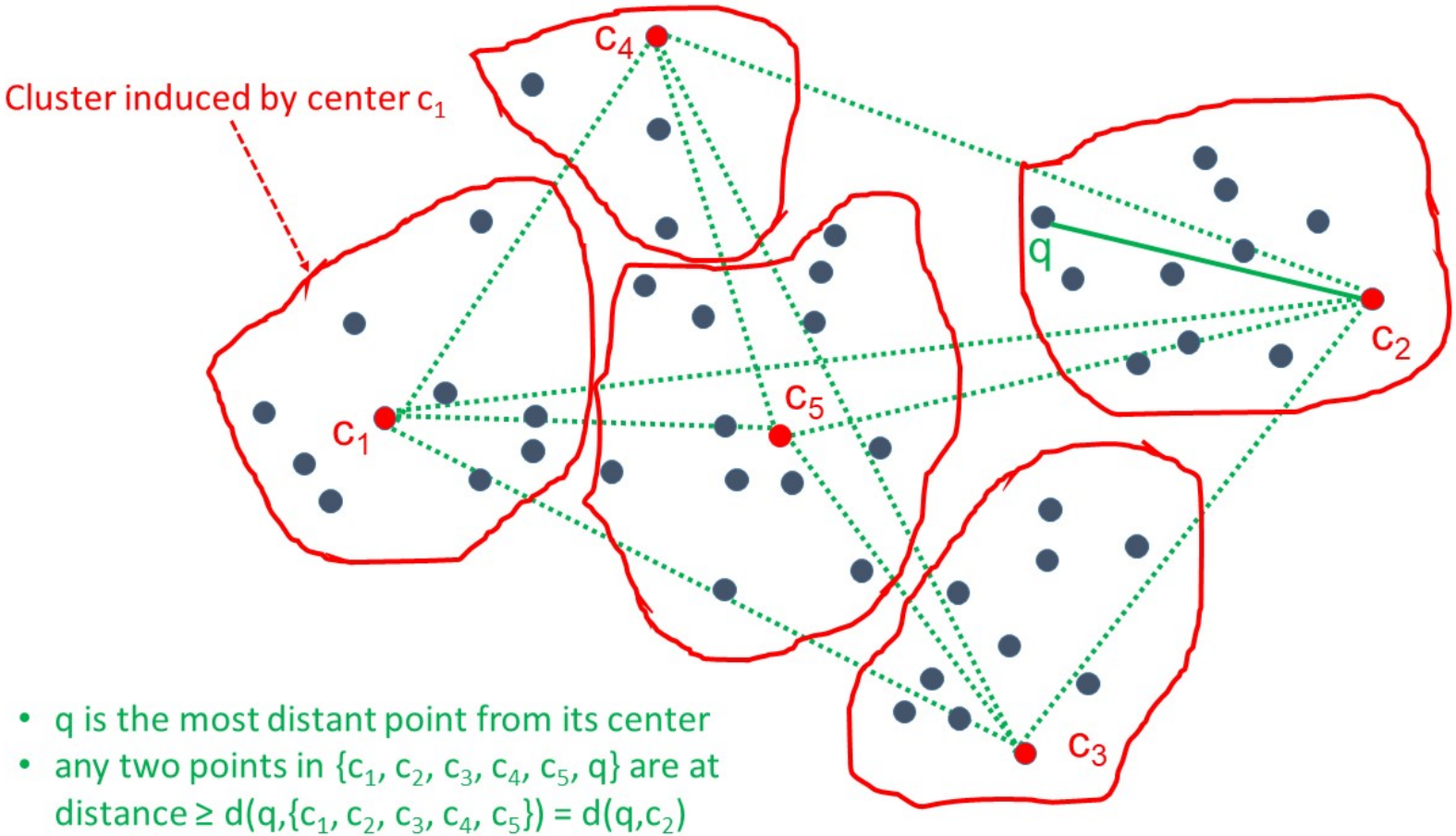
$\forall x \in P \quad d(x, S) \leq d(q, S)$

$S^* = \{c_1^*, \ldots, c_k^*\}$ sol. ottima per k-center su P

partizionano P in k insiemi $C_1^*, \ldots, C_k^*$ $\left(P = \bigcup_{i=1}^{k} C_i^*\right)$

1) $\forall x \in P \quad d(x, S^*) \leq \phi^{opt}_{k\text{-center}}(P, k)$
2) $\phi^{opt}(P, k) \leq \phi(P, S)$

Proof of Theorem

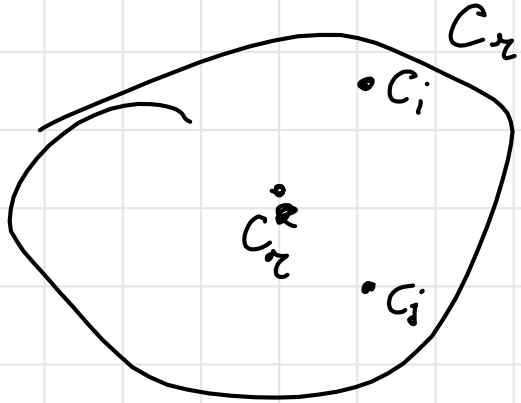


- q is the most distant point from its center
- any two points in $\{c_1, c_2, c_3, c_4, c_5, q\}$ are at distance $\geq d(q,\{c_1, c_2, c_3, c_4, c_5\}) = d(q,c_2)$

Proof of Theorem

$S' = S \cup \{q\} = \{c_1, \dots, c_{k+1}\}$

piccionaia: ci sarà insieme con 2 punti di S':
$\exists\, c_i, c_j \in S' \mid c_i, c_j \in C_r^*$ per qualche r $(i < j)$



$d(c_i, c_j) \le$
$\le d(c_i, c_r^*) + d(c_j, c_r^*) \le$ (triangle)
$\le d(c_i, S^*) + d(c_j, S^*) \le$
$\le \phi^{opt}(P,k) + \phi^{opt}(P,k) \le 2\phi^{opt}(P,k)$

mostriamo $d(x,S) \le 2\phi^{opt}(P,k)\ \forall x \in P \Rightarrow$ riduce $d(x,S)$ a $d(c_i,c_j)$
$d(x,S) = d(x, \{c_1, \dots, c_k\}) \le d(x, \{c_1, \dots, c_{j-1}\})$
| (FFT massimizza)
$\le d(c_j, \{c_1, \dots, c_{j-1}\}) \le d(c_j, \{c_i\}) = d(c_j, c_i) \le 2\phi^{opt}(P,k)$

$\phi(P,S) = \max_{x \in P} \{d(x,P)\} \le \max_{x \in P} \{2\phi^{opt}(P,k)\} \le 2\phi^{opt}(P,k)$

Proof of Theorem

Proof of Theorem

Proof of Theorem

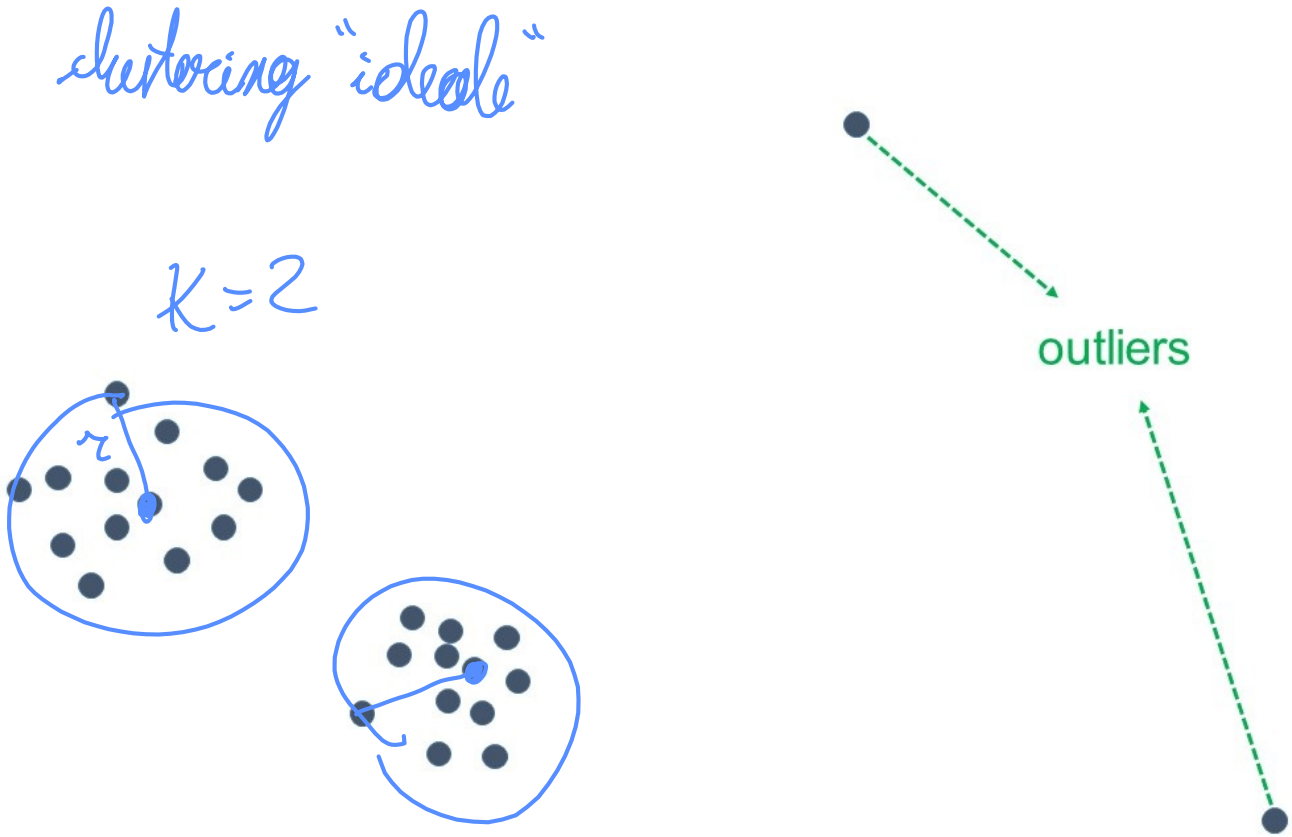
Proof of Theorem

Proof of Theorem

Observations on k-center clustering

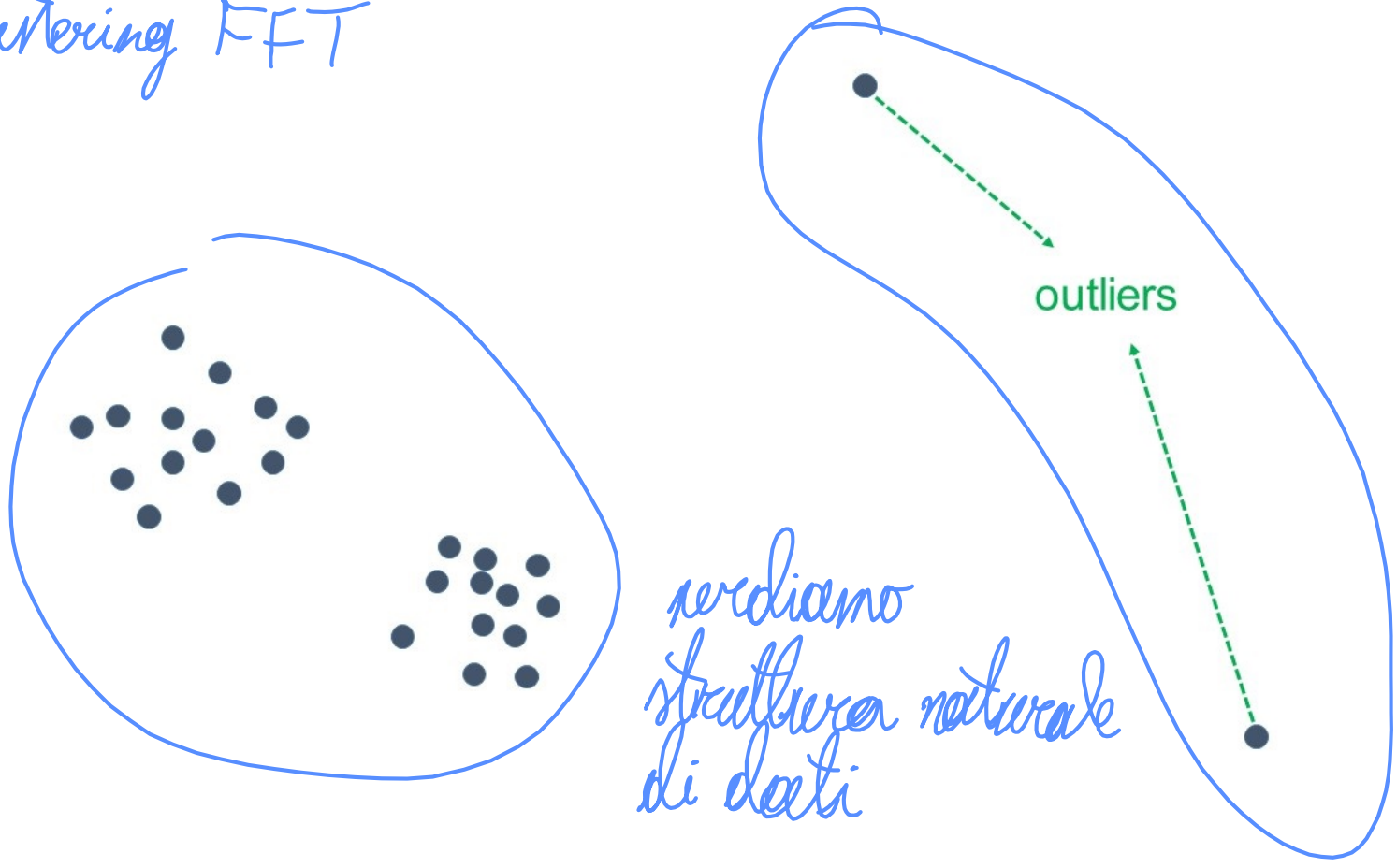
- The k-center objective focuses on worst-case distance of points from their closest centers.
- Farthest-First Traversal's approximation guarantees are almost the best one can obtain in practice. It was proved that computing a c-approximate solution to k-center is NP-hard for any fixed $c < 2$.
- The 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

clustering FFT



vediamo struttura naturale di dati

k-center clustering for big data

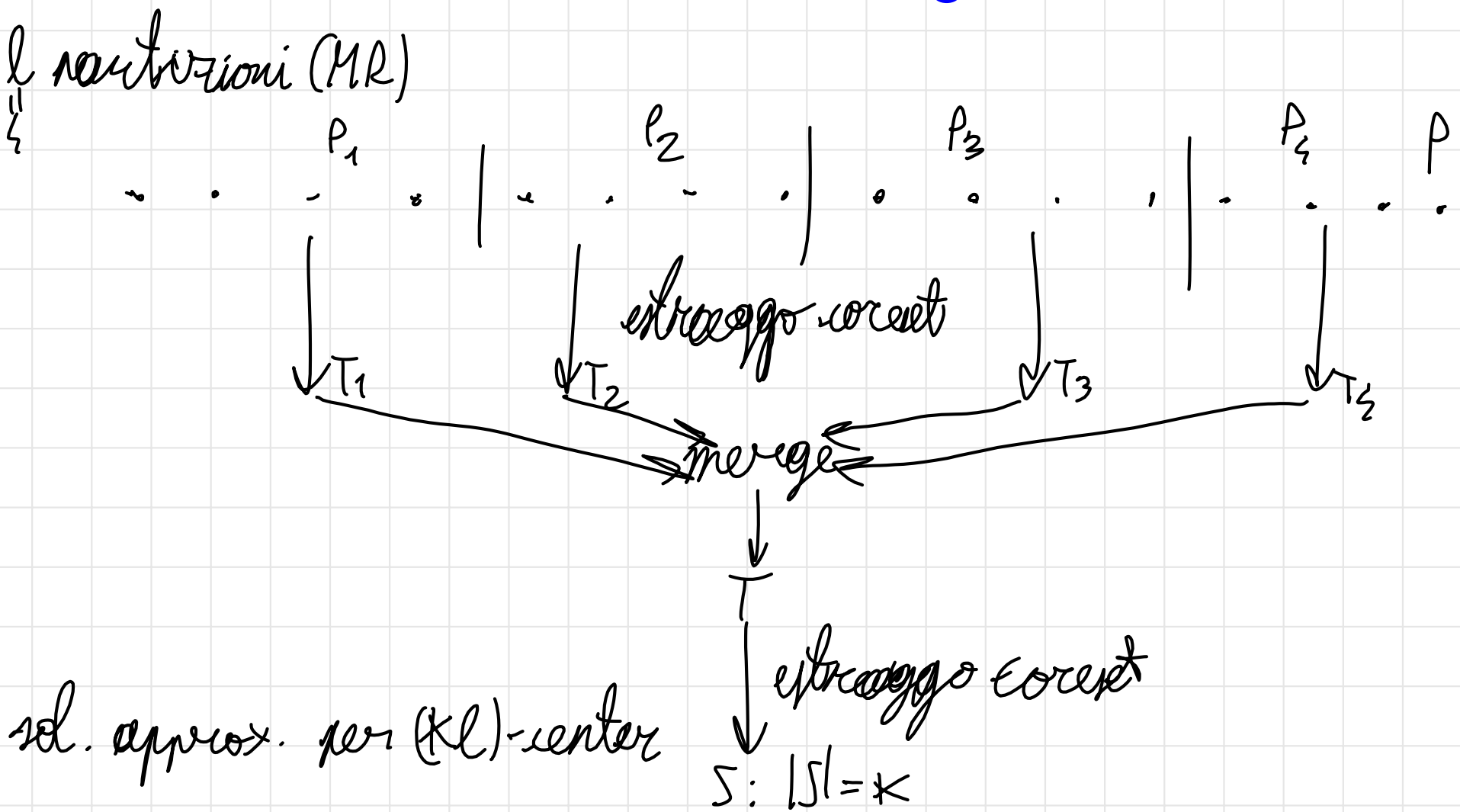
Observation. Farthest-First Traversal requires $k-1$ scans of the pointset P: impractical for massive P and k not small.

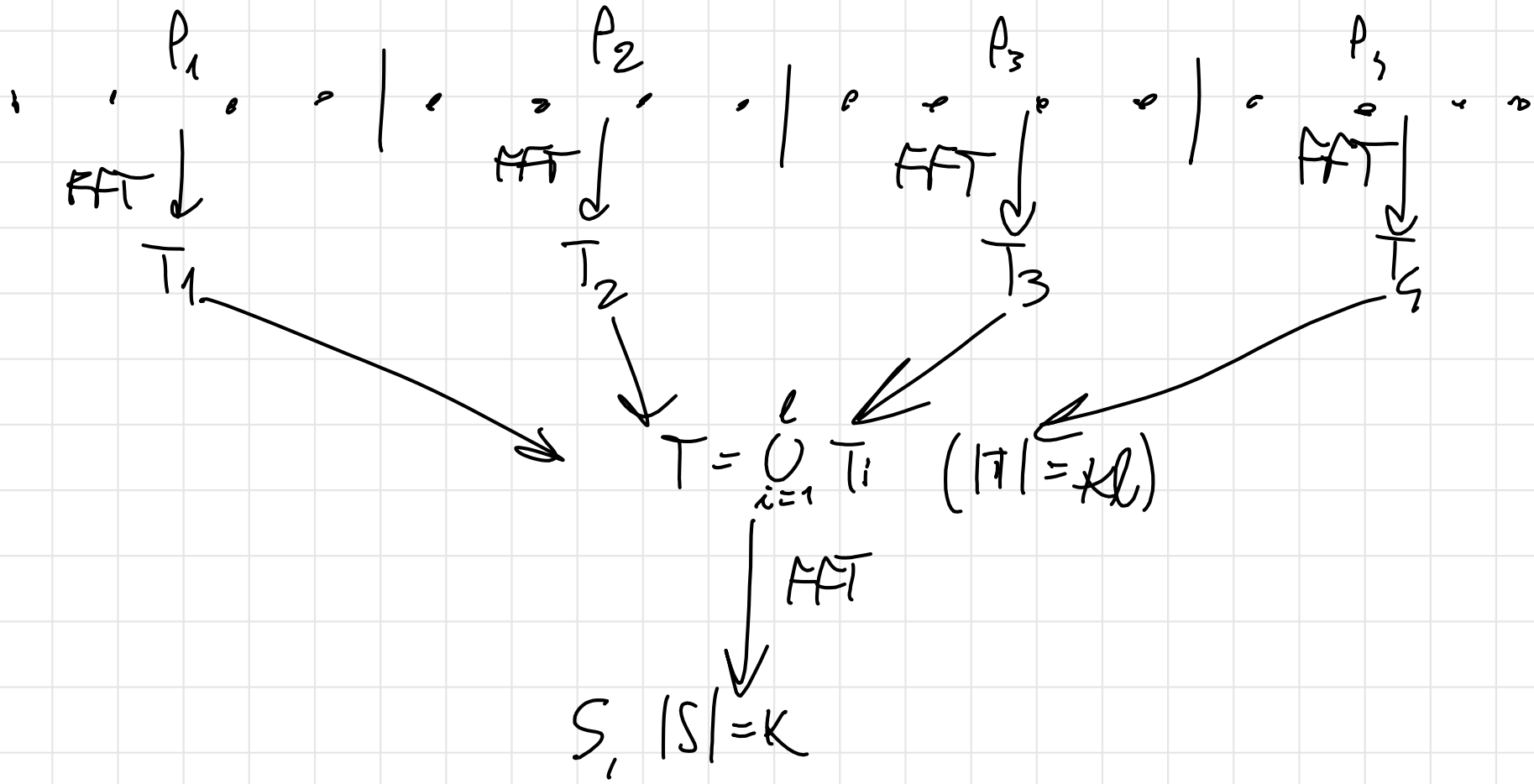
How can we compute a "good" solution to k-center for a pointset P which is too large for a single machine?

We resort to the composable coreset technique.

"parallelizzare" FFT

Application of composable coreset technique to k-center clustering





ad. approx di kcenter su P

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 $P_1, P_2, \ldots, P_\ell$.
  - 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 coreset $T = \cup_{i=1}^{\ell} T_i$ (of size $\ell \cdot k$) and run, using a single reducer, Farthest-First Traversal on T to determine a set $S = \{c_1, c_2, \ldots, c_k\}$ 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 $M_L = O\left(\sqrt{Nk}\right)$ *and aggregate space* $M_A = O(N)$.

Quale ℓ?

ROUND 1: $M_L = \max\{\overbrace{O(1)}^{\text{map}}, \overbrace{O(\frac{N}{\ell})}^{\text{reduce}}, \overbrace{O(\frac{N}{\ell})}^{\text{FFT}}\} = O(\frac{N}{\ell})$

ROUND 2: $M_L = O(k\ell)$

$M_L = \max\{O(\frac{N}{\ell}), O(k\ell)\} \leq O(\frac{N}{\ell} + k\ell)$

minimizzato da $\ell = \sqrt{\frac{N}{k}} \Rightarrow M_L = O(\sqrt{Nk})$

Analysis of MR-Farthest-First Traversal

M_A: mai + grande di input $\Rightarrow$ $M_A = O(N)$

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

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

Proof of Lemma

Proof of Lemma

Proof of Lemma

Proof of Lemma

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:

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

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 coreset, while still ensuring sublinear local space and linear aggregate space.

Does a random sample provide a good coreset?

Let $T \subseteq P$ be a coreset 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 coreset?

Does a random sample provide a good coreset?

k-center as a coreset primitive

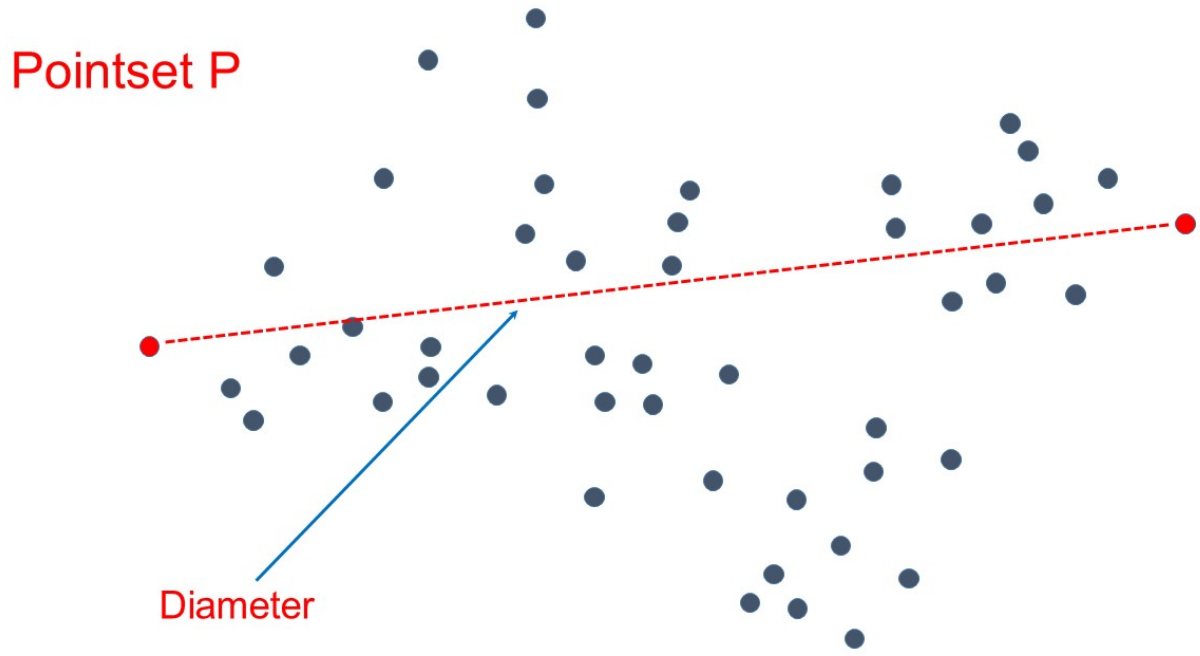
In MR-FFT, k-center is employed to extract the coreset for solving k-center itself. Now, we will see that it is also useful to extract coresets for other problems as well.

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 qudratic 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 $R = O(1)$, $M_L = O\left(\sqrt{N}\right)$ and $M_A = O\left(N^2\right)$. 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

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

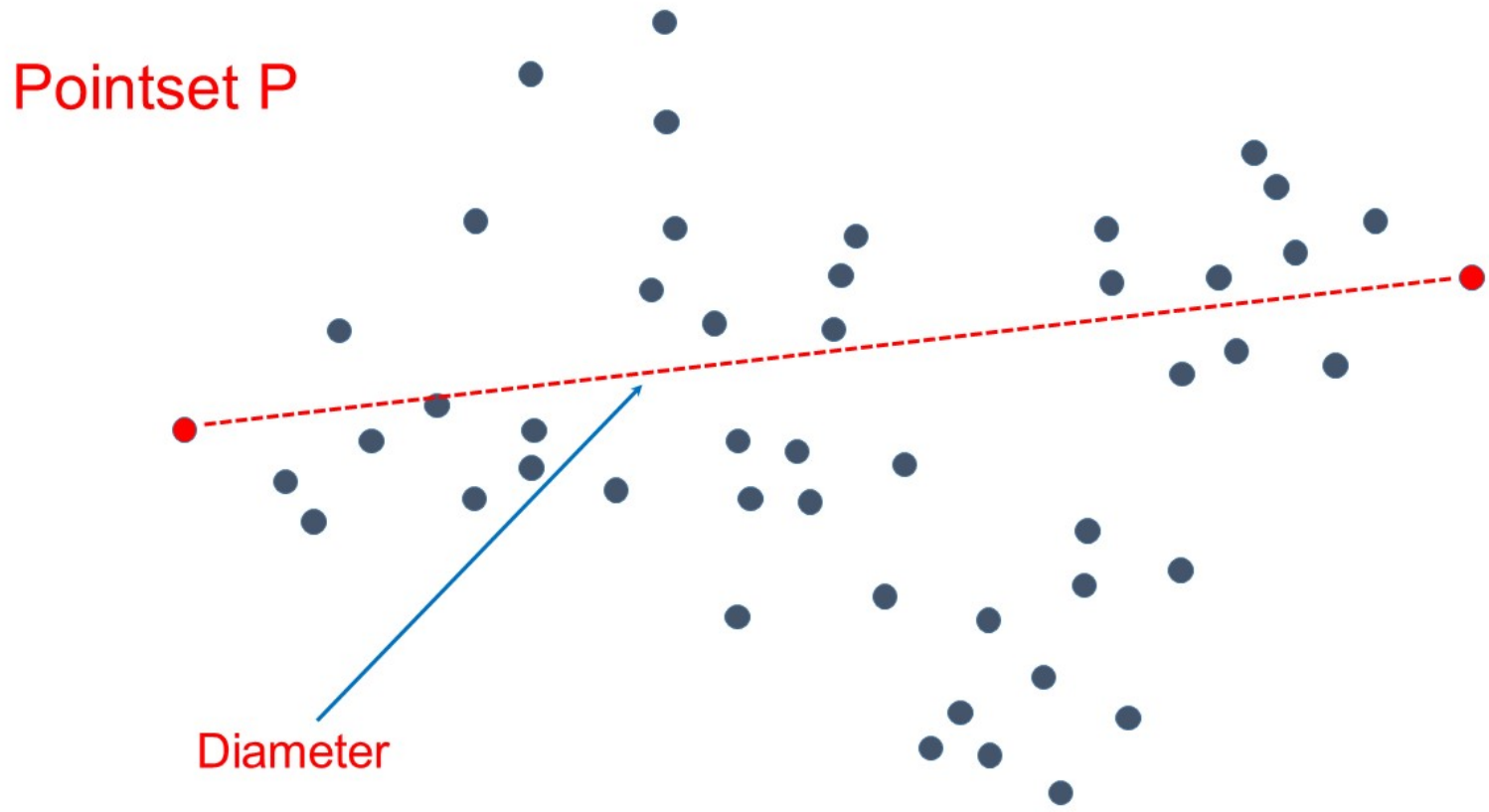
Lemma

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, $M_L = O\left(\sqrt{N}\right)$ and $M_A = O(N)$.

Proof of Lemma



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 $M_L = O\left(\sqrt{N}\right)$ 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

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

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 $\mathrm{div}(\cdot)$ to maximize (all NP-hard)

Coreset-based approach to diversity maximization

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

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

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

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

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

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

This property, is crucially exploited by the coreset-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 coreset 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

Coreset-based algorithm for diversity maximization

Coreset-based algorithm for diversity maximization

Coreset-based algorithm for diversity maximization

Coreset-based algorithm for diversity maximization