Clustering

(Part 1)

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

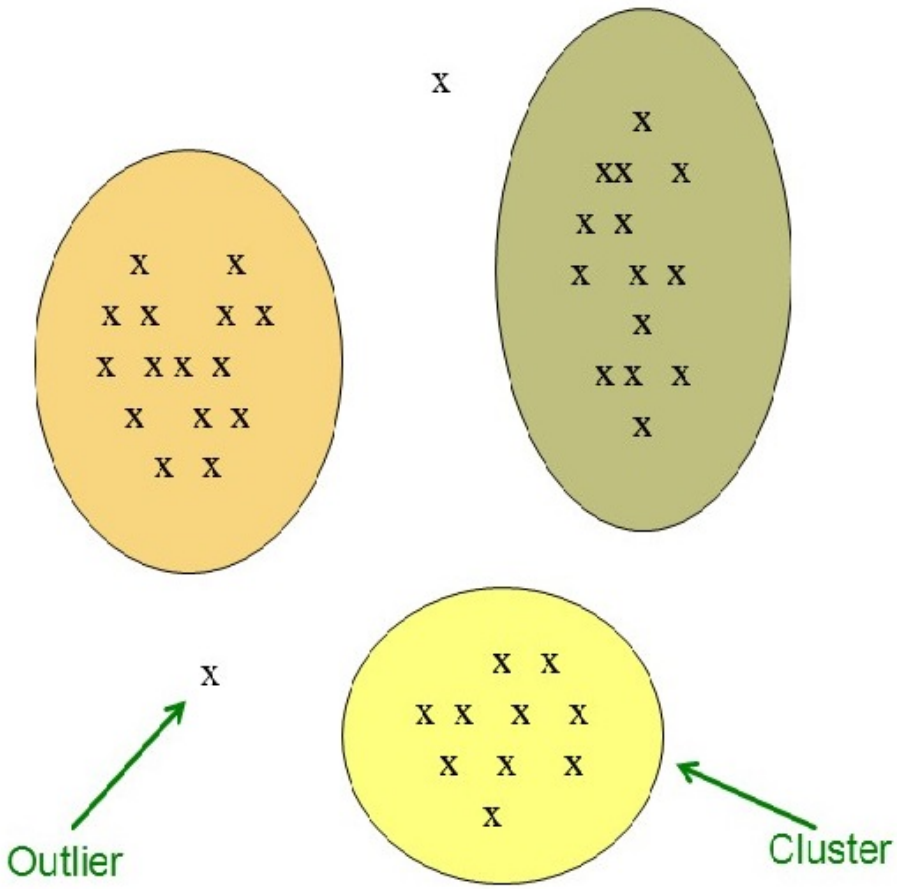


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

The clustering problem is a hard one!

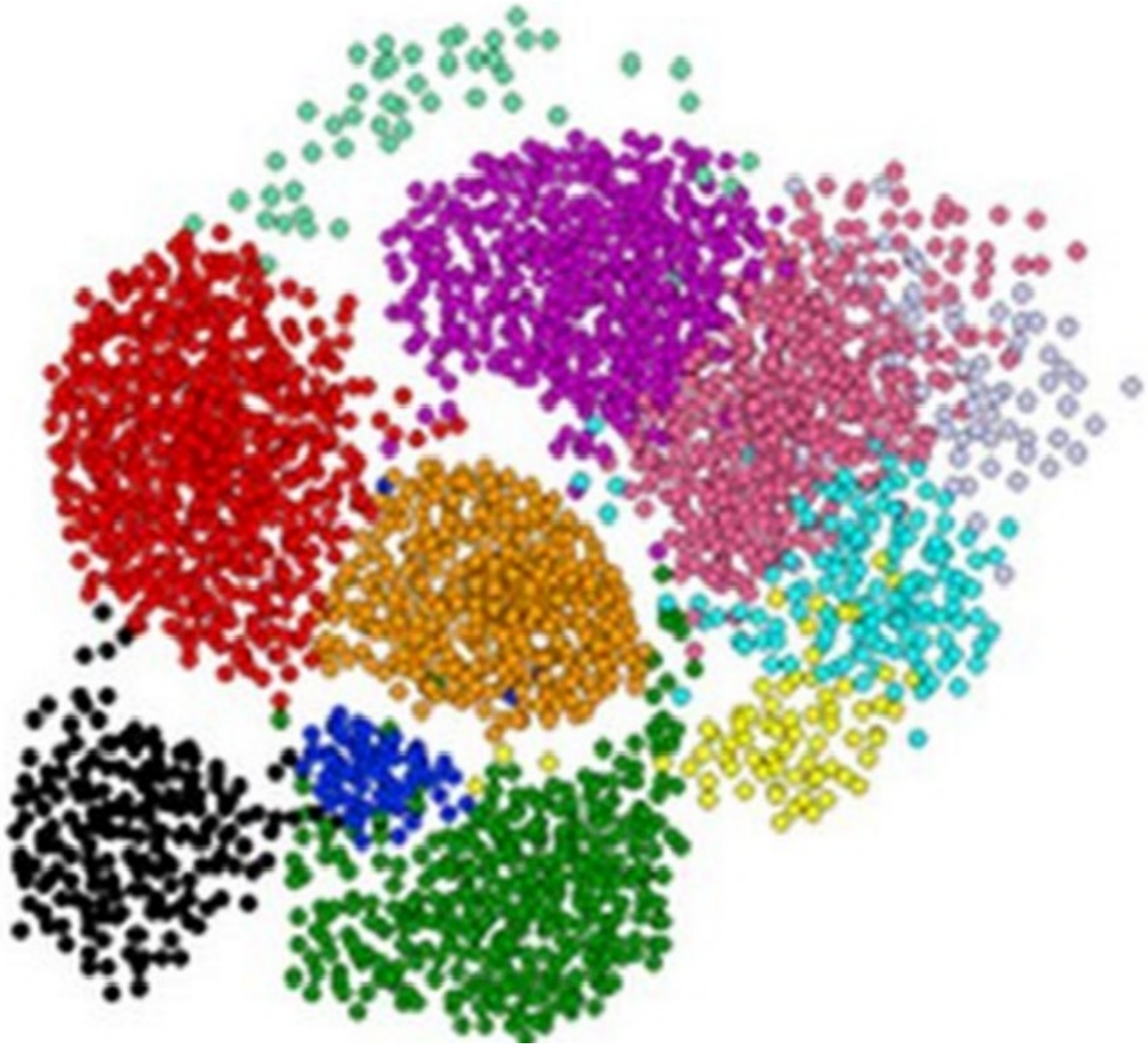


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

Applications

Clustering or cluster analysis is a fundamental task in data analysis, which finds applications in many different fields. Some examples:

1. Clustering is often a useful preprocessing step in other analysis tasks. E.g., it can be used for summarization, compression, outlier detection, class identification
2. Marketing. Clustering is used to partition consumers (potential customers) into segments based on shared characteristics. Then, high-yield segments can become targets of marketing campaigns or new products development.
3. Biology. Clustering of protein primary structures (amino acid sequences) is used to identify protein families and has important applications (e.g., in phylogenetic analysis)

Applications (cont'd)

1. Image processing. Clustering is used to analyze digital images for several purposes: e.g., object recognition; identification of different types of tissues in PET scans; identification of areas of similar land use in satellite pictures;
2. Social network analysis. Clustering is used to detect communities
3. Information retrieval. Clustering is used to categorize documents or web pages based on their topics which are not explicitly given but are inferred from their contents
4. Wireless sensor networks. Clustering is used to identify suitable cluster leaders which can play the role of communication hubs. In general, this is an instance of the well know facility location problem.

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

$$d : M \times M \to \Re$$

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)

Remark: Several options are often available for the distance function. The choice of a suitable distance is crucial for the effectiveness of the clustering.

Distance functions

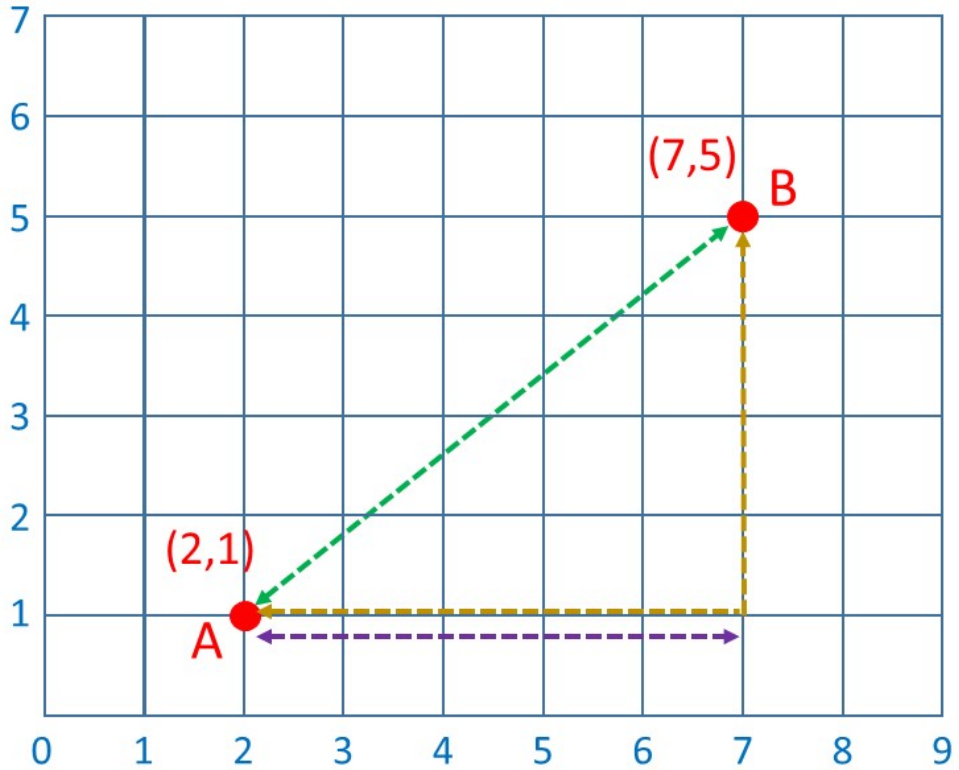
- Euclidean distances. Let $X, Y \in \Re^n$, with $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$. For $r \geq 1$ the $L_r - \text{distance}$ between X and Y (a.k.a. $L_r - \text{norm}$)

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

  - $r = 2$: standard distance in $\Re^n$ (also denoted by $|| \cdot ||$)
  - $r = 1$: referred to as *Manhattan distance*, it is the sum of the absolute differences of coordinates in each dimension. Used in grid-like environments
  - $r = \infty$: (limit for r that tends to ∞) it is the maximum absolute differences of coordinates, over all dimensions.

Distance functions (cont'd)

Examples of Euclidean distances:



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

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

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

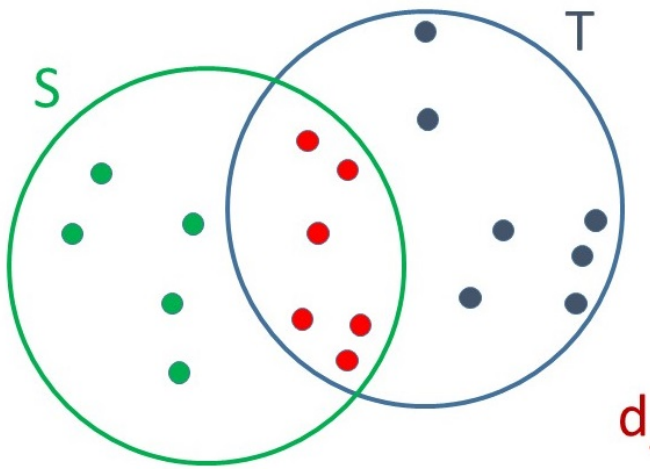
Distance functions (cont'd)

- 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 defined as

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

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:



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

Distance functions (cont'd)

- Cosine (or angular) distance. It is used when points are *vectors* in $\Re^n$ (or, in general in an inner-product space). Given two vectors $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ in $\Re^n$, their cosine distance is the angle between them, that is

$$d_{\text{cosine}} = \arccos\left(\frac{X \cdot Y}{||X|| \cdot ||Y||}\right) = \arccos\left(\frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n}(x_i)^2}\sqrt{\sum_{i=1}^{n}(y_i)^2}}\right)$$

Example: For $X = (1, 2, -1)$ and $Y = (2, 1, 1)$ the cosine distance is

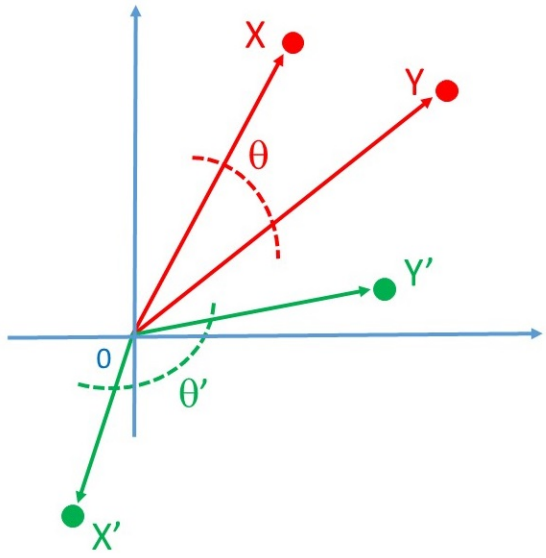
$$\arccos\left(\frac{3}{\sqrt{6}\sqrt{6}}\right) = \arccos(1/2) = \pi/3.$$

Remark. The cosine distance is often used in *information retrieval* to assess similarity between documents. Consider an alphabet of n words. A document X over this alphabet can be represented as an n-vector, where x_i is the number of occurrences in X of the i-th word of the alphabet.

Distance functions (cont'd)

Observations on the cosine distance

- For non-negative coordinates it takes values in $[0, \pi/2]$, while for arbitrary coordinates the range is $[0, \pi]$. Dividing by $\pi/2$ (or π) the range becomes $[0, 1]$.



$\theta \in [0,\pi/2]$ (coordinates ≥ 0)

$\theta' \in [0,\pi]$ (arbitrary coordinates)

- In order to satisfy the second property of distance functions for metric spaces, scalar multiples of a vector must be regarded as the same vector

Distance functions (cont'd)

- Edit distance. Used for strings. Given two strings X and Y, their edit distance is the minimum number of deletions and insertions that must be applied to transform X into Y.

  Example: for $X = \mathsf{ABCDE}$ and $Y = \mathsf{ACFDEG}$, we have $d_{\mathsf{edit}}(X, Y) = 3$, since we can transform X into Y as follows:

  $$ABCDE \xrightarrow{\text{delete B}} ACDE \xrightarrow{\text{insert F}} ACFDE \xrightarrow{\text{insert G}} ACFDEG,$$

  and less than 3 deletions/insertions cannot be used.

  Remark: It is immediate to show that in all cases

  $$d_{\mathsf{edit}}(X, Y) = |X| + |Y| - 2|\mathsf{LCS}(X, Y)|,$$

  where $\mathsf{LCS}(X, Y)$ is the length of the Longest Common Subsequence of X and Y, which can be computed in $O(|X| \cdot |Y|)$ time through dynamic-programming.

Distance functions (cont'd)

- Hamming distance. Used when points are *vectors* (as for cosine distance) over some n-dimensional space. Most commonly, it is used for binary vectors. The Hamming distance between two vectors is the number of coordinates in which they differ.

  Example: for $X = (0, 1, 1, 0, 1)$ and $Y = (1, 1, 1, 0, 0)$ (i.e., $n = 5$)

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

  since they differ in the first and last coordinates.

Distance functions (cont'd)

Exercise

Show that the L_1 and Edit distances satisfy the four requirements for a metric space.

Observation: Proving the triangle inequality for the Jaccard distance requires tedious yet simple calculations. If interested, look at the paper: *M. Levandowsky, D. Winter. Distance between sets. Nature 234:34-35, 1971.*

Curse of dimensionality

Random points in a high-dimensional metric space tend to be

- Sparse
- Almost equally distant from one another
- Almost orthogonal (as vectors) to one another .

As a consequence, in high dimensions distance functions may loose effectiveness in assessing similarity/dissimilarity. However, this holds in particular when points are random, and it might be less of a problem in some real-world datasets.

Curse of dimensionality (cont'd)

As an example, consider a set S of N random points in $[0,1]^t$, where for each point the i-th coordinate is a random number in $[0,1]$ drawn with uniform probability independently of the other coordinates and of the other points.
Let $X = (x_1, x_2, \ldots, x_t)$ and $Y = (y_1, y_2, \ldots, y_t)$ two points of S.
Their L_2 distance is

$$d_{L_2}(X,Y) = \sqrt{\sum_{i=1}^{t}(x_i - y_i)^2} \leq \sqrt{t}.$$

Theorem

If $t \geq 4 \log_2 N$, then with probability $\geq 1 - 1/N$ every two points of S are at distance $\geq \sqrt{t}/(\sqrt{2} \cdot 24) = \Omega\left(\sqrt{t}\right)$ from one another.

Curse of dimensionality (cont'd)

Proof of Theorem

Fix an arbitrary pair of distinct points $X, Y \in S$. We now show that

$$\Pr\left(d_{L_2}(X,Y) \leq \sqrt{t}/(\sqrt{2}\cdot 24)\right) \leq 1/N^2.$$

- Let p be the probability that $|x_i - y_i| \leq 1/24$, for any $1 \leq i \leq t$. It is easy to see that $p \leq 1/12$.
- The number of coordinates i such that $|x_i - y_i| \leq 1/24$ is a Binomial(t,p) random variable, which we call V_{XV}.
- Now observe that if $d_{L_2}(X,Y) \leq \sqrt{t}/(\sqrt{2}\cdot 24)$ the number of coordinates i such that $|x_i - y_i| > 1/24$ must be less than $t/2$ and, therefore, those such that $|x_i - y_i| \leq 1/24$ (i.e., V_{XY}) must be at least $t/2$. This implies that

$$\Pr\left(d_{L_2}(X,Y) \leq \sqrt{t}/(\sqrt{2}\cdot 24)\right) \leq \Pr\left(V_{XV} \geq t/2\right).$$

(Recall that if an event A implies an event B, then $\Pr(A) \leq \Pr(B)$.)

Curse of dimensionality (cont'd)

Proof of Theorem (cont'd).

- Since $t/2 \geq 6tp = E[V_{XY}]$, by Chernoff bound we obtain that $\Pr(V_{XV} \geq t/2) \leq 2^{-t/2} \leq 1/N^2$. Therefore

$$\Pr\left(d_{L_2}(X, Y) \leq \sqrt{t}/(\sqrt{2} \cdot 24)\right) \leq 1/N^2.$$

Since there are less than N^2 pairs of points in S, the union bound and the above inequality bound imply that the probability that there exist two points of S at distance at most $\sqrt{t}/(\sqrt{2} \cdot 24)$ is less than $1/N$. □

Types of clusterings

Given a set of points in a metric space, hence a distance between points, a clustering problem often specifies an objective function to optimize. The objective function also allows to compare different solutions.

Objective functions 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.

  When centers are required, the value of the objective function depends on the selected centers. Cluster centers must belong to the underlying metric space but, sometimes, they are constrained to belong to the input set.
- Disjoint clusters are sought.

Background

Combinatorial optimization problem

A (combinatorial) optimization problem is a computational problem $\Pi(I, S, \Phi, m)$ defined by a set of *instances* I, a set of *solutions* S, an *objective function* Φ that assigns a real value to each solution $s \in S$, and $m \in \{\min, \max\}$. For each instance $i \in I$ there is a subset $S_i \subseteq S$ of *feasible solutions*. Given $i \in I$ the problem requires to find an *optimal solution*, namely a feasible solution $s \in S_i$ which minimizes $\Phi(s)$, if $m = \min$ (minimization problem), or maximizes $\Phi(s)$, if $m = \max$ (maximization problem)[1]

Observation: For several important optimization problems no algorithms are known which find an optimal solution in time polynomial in the input size, and it is unlikely that such algorithms exist (*NP-hard problems*). Therefore, especially when thinking of big data, one has to aim at approximations of optimal solutions.

[1] Technically, if S_i is not finite min and max should become inf and sup.

Background (cont'd)

Approximation algorithm

For $c \geq 1$, a c-approximation algorithm A for a combinatorial optimization problem $\Pi(I, S, \Phi, m)$ is an algorithm that for each instance $i \in I$ returns a feasible solution $A(i) \in S_i$ such that $\Phi(A(i)) \leq c \min_{s \in S_i} \Phi(s)$, if $m = \min$, or $\Phi(A(i)) \geq (1/c) \max_{s \in S_i} \Phi(s)$, if $m = \max$. The value c is called approximation ratio, and the solution $A(i)$ is called c-approximate solution or c-approximation

Observation: the approximation introduces a flexibility in selecting the solution to a given instance, which algorithms can exploit to run more efficiently.

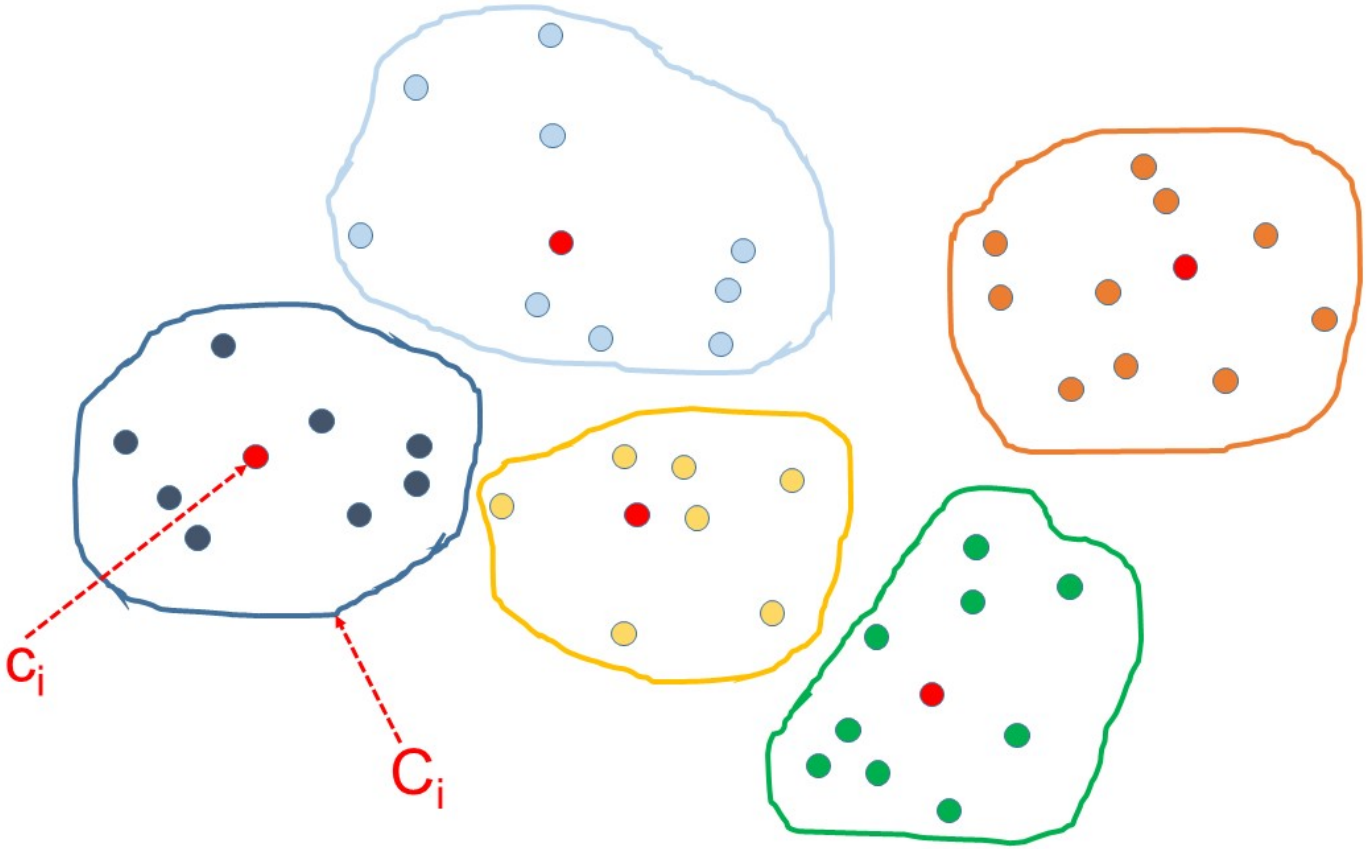
Center-based clustering

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 $1 \leq i \leq k$.

Observe that the above definition requires the centers to belong to the clusters, hence to the pointset. Whenever appropriate we will discuss the case when the centers can be chosen more freely from the metric space.

Example of 5-clustering



Center-based clustering (cont'd)

Definition

Consider e metric space (M, d) and an integer $k > 0$. A k-clustering problem for (M, d) is an optimization problem whose instances are the finite pointsets $P \subseteq M$. For each instance P, the problem requires to compute a k-clustering $\mathcal{C}$ of P (feasible solution) which minimizes a suitable objective function $\Phi(\mathcal{C})$.

The following are three popular objective functions.

- $\Phi_{\text{kcenter}}(\mathcal{C}) = \max_{i=1}^{k} \max_{a \in C_i} d(a, c_i)$ (k-center clustering)
- $\Phi_{\text{kmeans}}(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in C_i} (d(a, c_i))^2$ (k-means clustering).
- $\Phi_{\text{kmedian}}(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in C_i} d(a, c_i)$ (k-median clustering).

In other words, under the above functions, the problem requires to find the k-clustering that minimizes, respectively, the maximum distance (k-center), or the average square distance (k-means), or the average distance (k-median) of a point to its cluster center.

Observations

- All aforementioned problems (k-center, k-means, k-median) are NP-hard. Hence, in general it is impractical to search for optimal solutions.
- There are several efficient approximation algorithms that in practice return good-quality solutions. However, dealing efficiently with large inputs is still a challenge!
- k-center and k-median belong to the family of facility-location problems. In these problems, a set F of candidate *facilities* and a set C of *clients* are given and the objective is to find a subset of at most k candidate facilities to open,and an assignment of clients to them, so to minimize some the maximum or average distance between a client and its assigned facilities. In our formulation, each input point represents both a facility and a client. Numerous variants of these problems have been studied in the literature.
- k-means objective is also referred to as Sum of Squared Errors (SSE)

Partitioning primitive

Let P be a pointset and $S \subseteq P$ a set of k selected centers. For *all* previously defined clustering problems, the best k-clustering around these centers is the one where each c_i belongs to a distinct cluster and each other point is assigned to the cluster of the closest c_i (ties broken arbitrarily).

We will use primitive Partition(P, S) to denote this task:

Partition(P, S)

Let $S = \{c_1, c_2, \ldots, c_k\} \subseteq P$
for $i \leftarrow 1$ **to** k **do** $C_i \leftarrow \{c_i\}$
for each $p \in P - S$ **do**
  $\ell \leftarrow \texttt{argmin}_{i=1,k}\{d(p, c_i)\}$ // *ties broken arbitrarily*
  $C_\ell \leftarrow C_\ell \cup \{p\}$
$\mathcal{C} \leftarrow (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$
return $\mathcal{C}$

Exercise

Let $|P| = N$ and suppose that each point $x \in P$ is represented as a key-value pair $(\mathrm{ID}_x, (x, f))$, where ID_x is a distinct integer in $[0, N)$, and f is a binary flag which is 1 if $x \in S$ and 0 otherwise. Design a 1-round MapReduce algorithm that implements Partition(P, S) with $M_L = O(k)$ and $M_A = O(N)$. (Assume that k and N are known.)

k-center clustering

Farthest-First Traversal: algorithm

- 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 $k > 1$

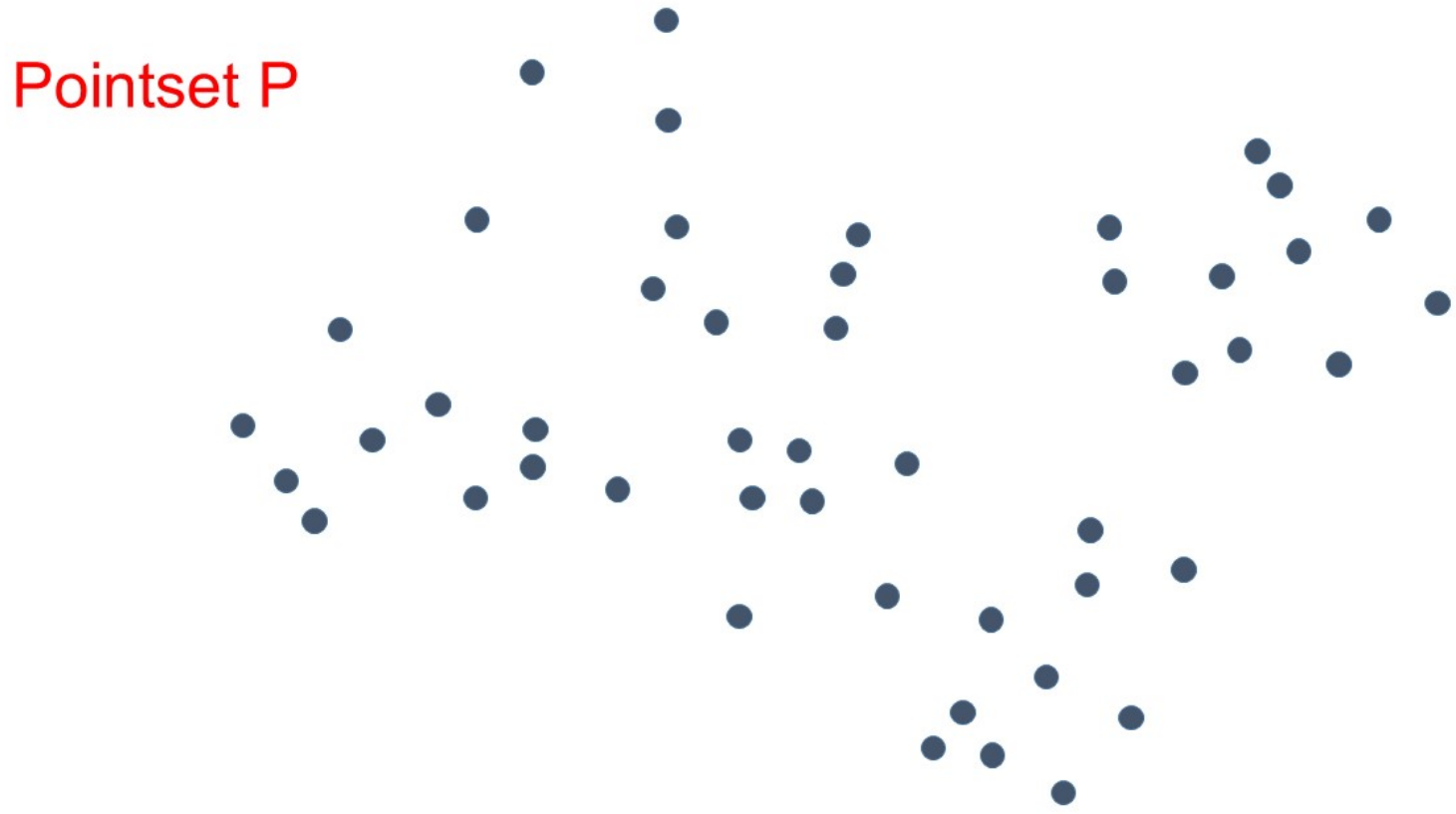
Output k-clustering $\mathcal{C} = (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$ of P.

Farthest-First Traversal: algorithm (cont'd)

$S \leftarrow \emptyset$ {c_1} //c_1\in P arbitrary point
for $i \leftarrow 1$ **to** k **do**
  Find the point $c_i \in P - S$ that maximizes $d(c_i, S)$
  $S \leftarrow S \cup \{c_i\}$
return Partition(P, S)

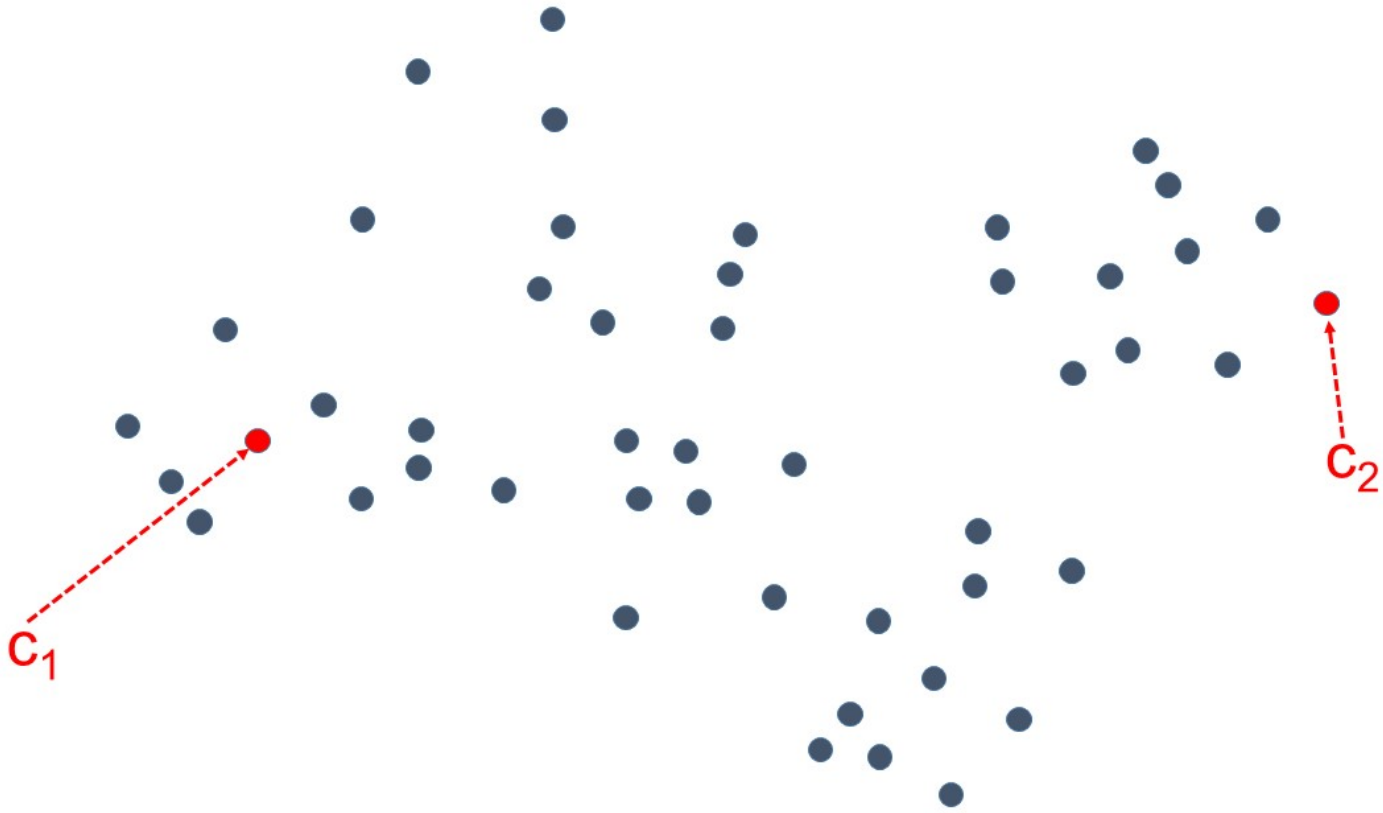
- $d(c_i, S)$ denotes the minimum distance of c_i from a point of S. That is, $d(c_i, S) = \min\{c \in S \ : \ d(c_i, c)\}$
- The assignment of points to clusters can be accomplished while determining the centers in the first for-loop.

Farthest-First Traversal: example

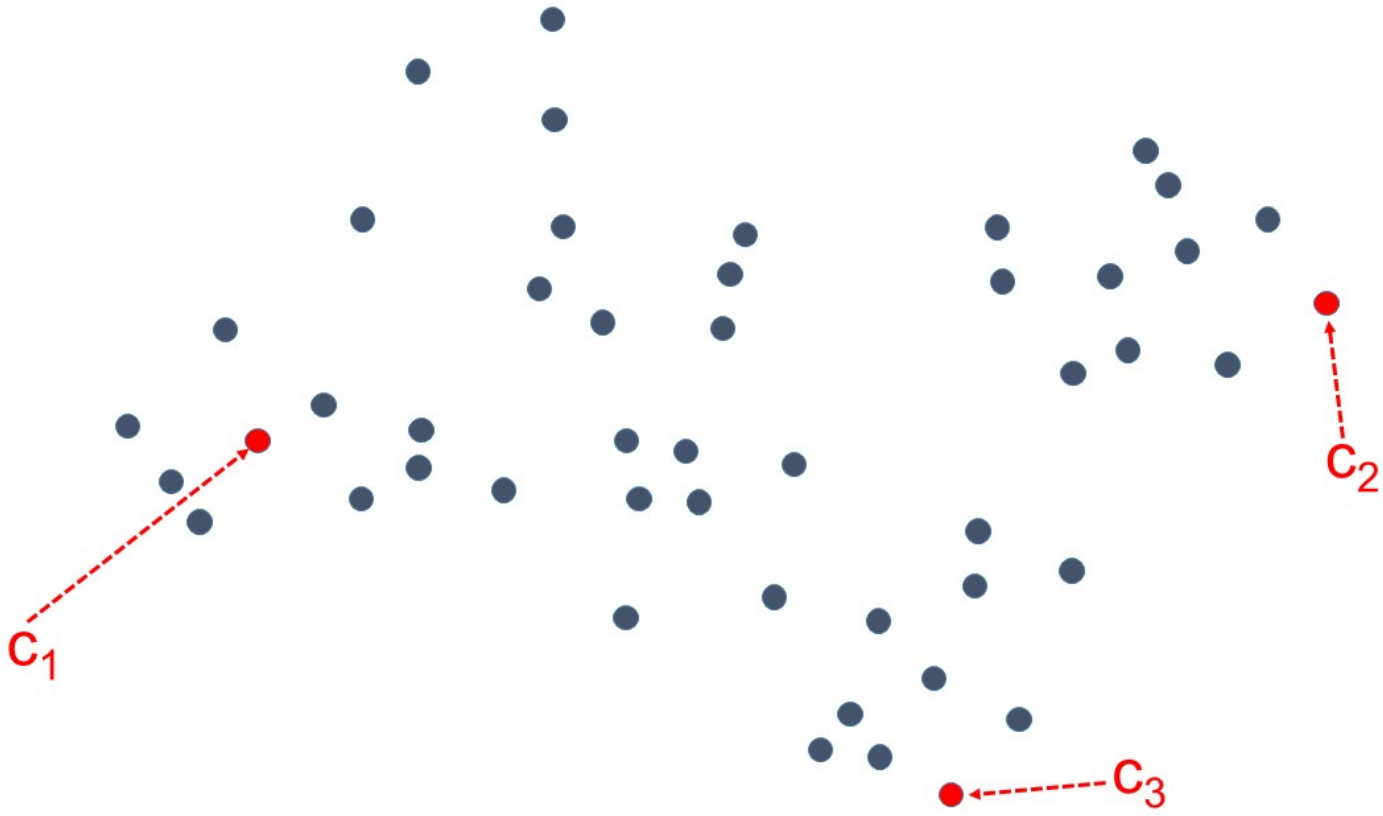


Farthest-First Traversal: example

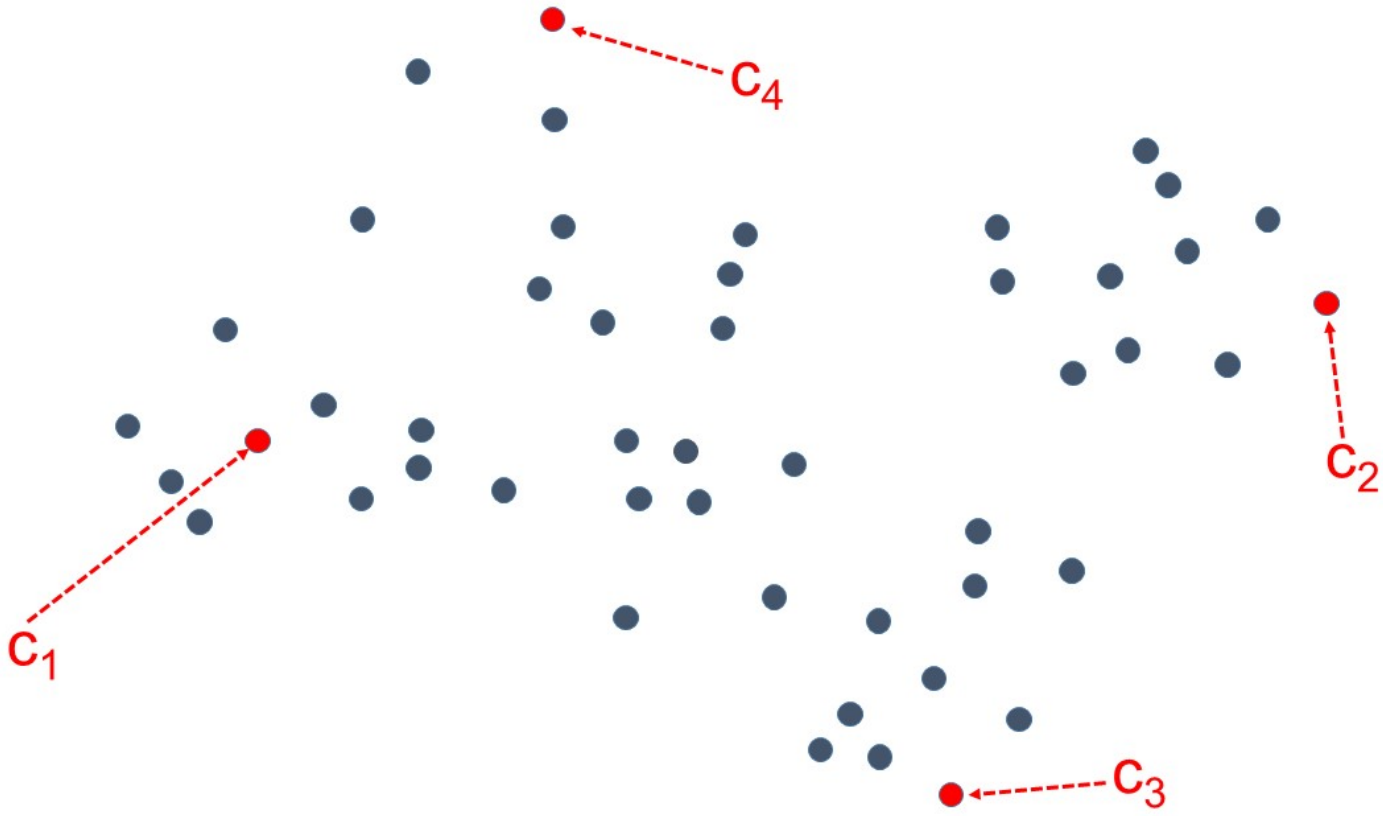
Farthest-First Traversal: example



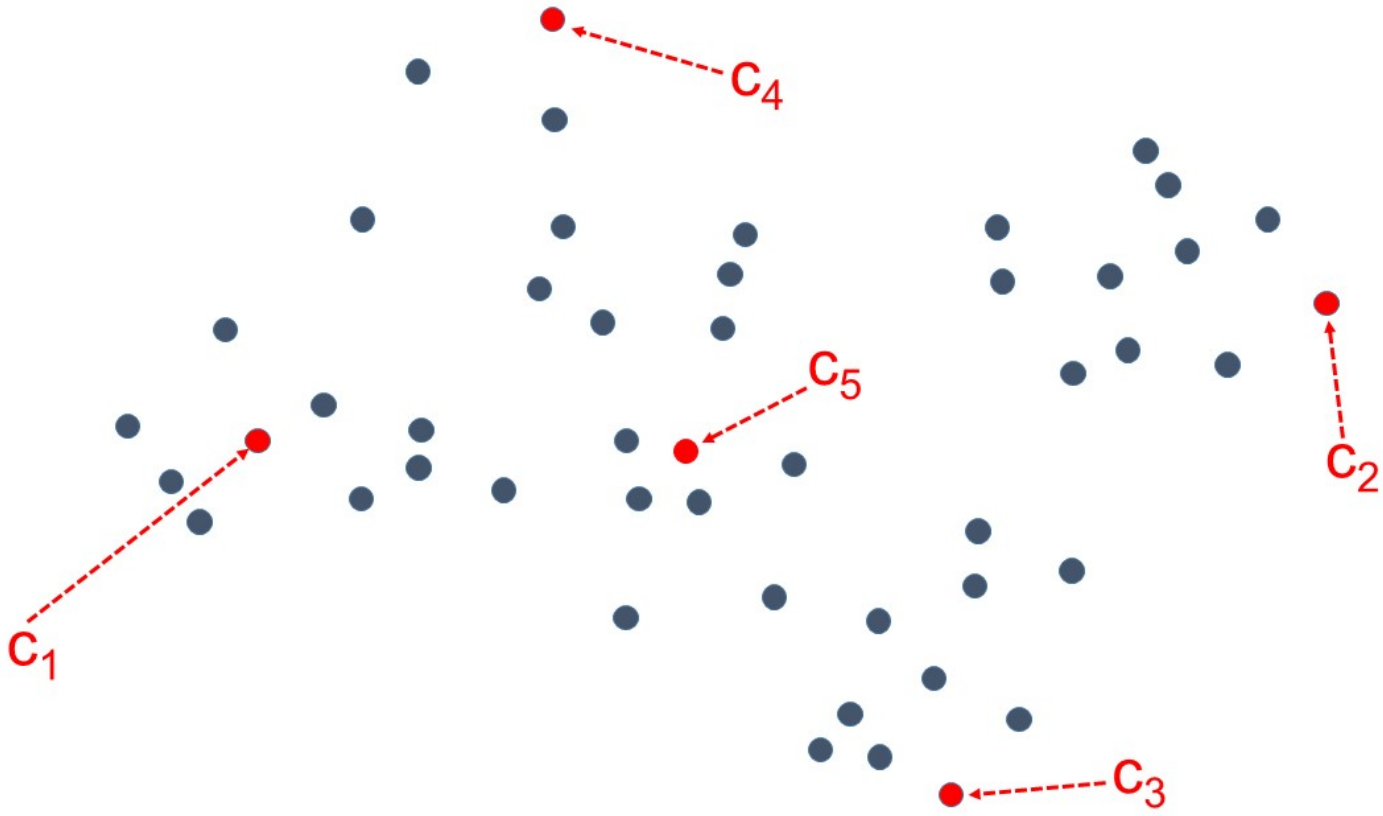
Farthest-First Traversal: example



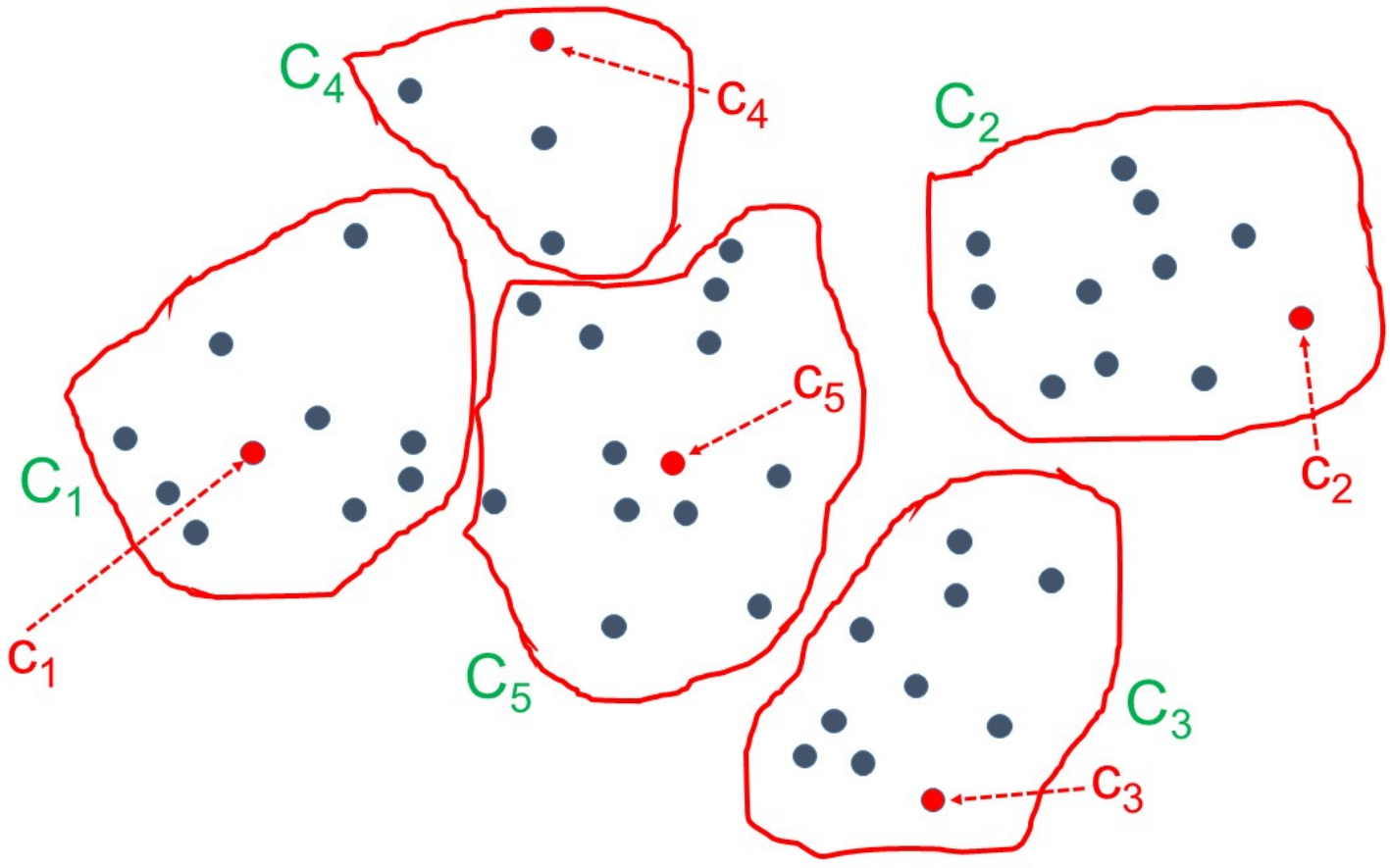
Farthest-First Traversal: example



Farthest-First Traversal: example



Farthest-First Traversal: example



Farthest-First Traversal: analysis

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 first 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.

Theorem

Let $\Phi^{\text{opt}}_{\text{kcenter}}(k)$ be the minimum value of $\Phi_{\text{kcenter}}(\mathcal{C})$ over all possible k-clusterings $\mathcal{C}$ of P, and let $\mathcal{C}_{\text{alg}}$ be the k-clustering of P returned by the Farthest-First Traversal algorithm. Then:

$$\Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}}) \leq 2 \cdot \Phi^{\text{opt}}_{\text{kcenter}}(k).$$

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

Farthest-First Traversal: analysis (cont'd)

Proof of Theorem

Let

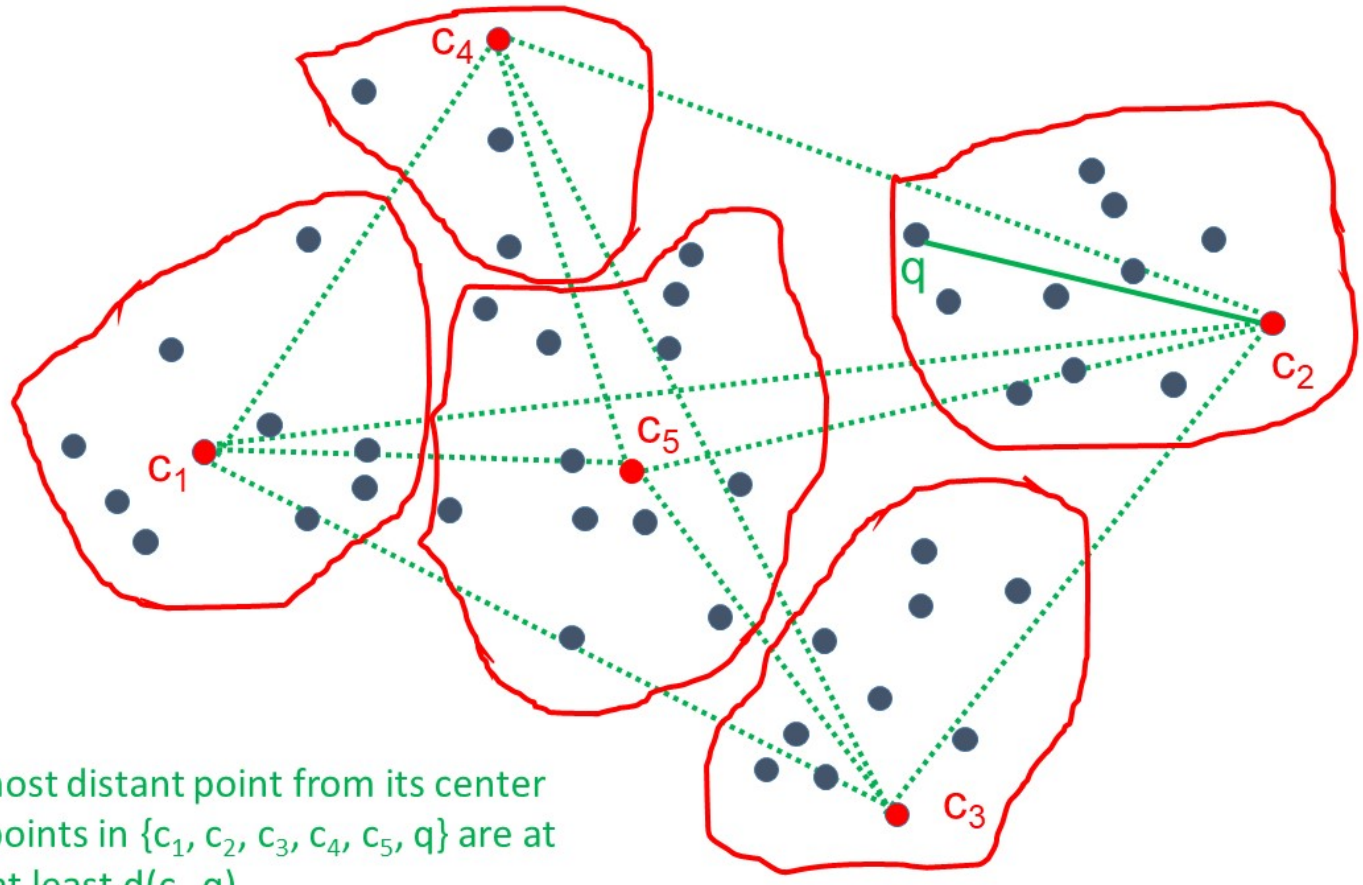
$$\mathcal{C}_{\text{alg}} = (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k),$$

and let q be the point with maximum distance from its assigned cluster's center, say c_i. Therefore, $d(q, c_i) = \Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}})$.

Observe that:

- For every $j \neq i$ we have $d(q, c_j) \geq d(q, c_i)$ (since c_i is q's assigned center)
- For every $1 \leq j1 < j2 \leq k$ we have $d(c_{j1}, c_{j2}) \geq d(q, c_i)$ (otherwise the algorithm would have chosen q as center in iteration $j2$).

Farthest-First Traversal: analysis (cont'd)



- 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 at least $d(c_2,q)$

Farthest-First Traversal: analysis (cont'd)

Proof of Theorem (cont'd).

Therefore there exists $k+1$ points, namely $c_1, c_2, \ldots, c_k, q$ whose pairwise distances are all $\geq d(q, c_i) = \Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}})$. Two of them must fall in the same cluster of the optimal clustering.

Suppose that x, y are two points which belong to the same cluster of the optimal clustering with center c, and such that $d(x, y) \geq \Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}})$. By the triangle inequality, we have that

$$\Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}}) \leq d(x, y) \leq d(x, c) + d(c, y) \leq 2\Phi^{\text{opt}}_{\text{kcenter}}(k),$$

and the theorem follows. □

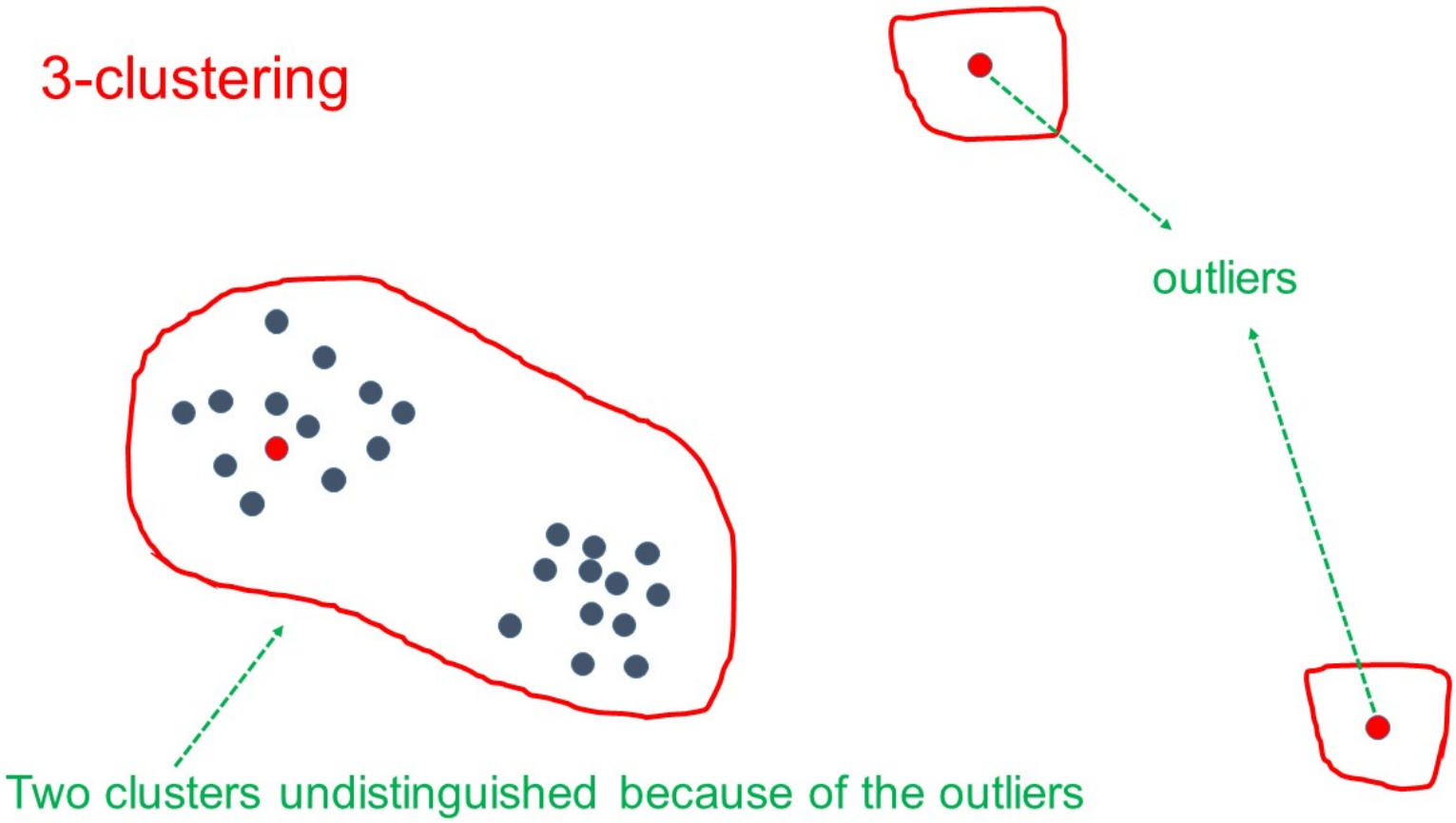
Observations on k-center clustering

- k-center clustering provides a strong guarantee on how close each point is to the center of its cluster.
- However, for noisy pointsets (e.g., pointsets 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).
- For any fixed $\epsilon > 0$ it is NP-hard to compute a k-clustering $\mathcal{C}_{\text{alg}}$ with

$$\Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}}) \leq (2 - \epsilon)\Phi^{\text{opt}}_{\text{kcenter}},$$

  hence the Farthest-First Traversal is likely to provide almost the best approximation guarantee obtainable in polynomial time.

Example: noisy pointset



k-center clustering for big data

How can we compute a "good" k-center clustering of a pointset P that is too large for a single machine?

We employ a coreset-based approach:

1. Extract from P a small T (coreset) of representatives.
2. Compute within T the set S of k cluster centers.
3. Compute the final clustering around the centers of S.

Observations:

- The coreset-based approach is effective when Steps 1 and 3 can be performed efficiently (e.g., in parallel), and the coreset T contains a good set of centers
- Coresets are used to confine computations which are too expensive to run on the whole input on small instances.
- A coreset needs not be a simple subset of the input.

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. The following MapReduce algorithm (MR-Farthest-First Traversal) computes a good k-center clustering. (The details of the map and reduce phases of each round are left as an exercise.)

- Round 1: Partition P arbitrarily in ℓ subsets of equal size $P_1, P_2, \ldots, P_\ell$ and execute the Farthest-First Traversal algorithm on each P_i separately to identify a set T_i of k centers, for $1 \leq i \leq \ell$.
- Round 2: Gather the coreset $T = \cup_{i=1}^{k} T_i$ (of size $\ell \cdot k$) and run, using a single reducer, the Farthest-First Traversal algorithm on T to identify a set $S = \{c_1, c_2, \ldots, c_k\}$ of k centers.
- Round 3: Execute Partition(P, S) (see previous exercise on how to implement Partition(P, S) in 1 round).

Observation: note that in Rounds 1 and 2 the Farthest-First traversal algorithm is used to determine only centers and not complete clusterings.

Analysis of MR-Farthest-First Traversal

Assume $k = o(N)$. By setting $\ell = \sqrt{N/k}$, it is easy to see that the 3-round MR-Farthest-First traversal algorithm uses

- Local space $M_L = O\left(\sqrt{N \cdot k}\right) = o(N)$
- Aggregate space $M_A = O(N)$

Theorem

Let $\Phi^{\text{opt}}_{\text{kcenter}}(k)$ be the minimum value of $\Phi_{\text{kcenter}}(\mathcal{C})$ over all possible k-clusterings $\mathcal{C}$ of P, and let $\mathcal{C}_{\text{alg}}$ be the k-clustering of P returned by the MR-Farthest-First Traversal algorithm. Then:

$$\Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}}) \leq 4 \cdot \Phi^{\text{opt}}_{\text{kcenter}}(k).$$

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

Analysis of MR-Farthest-First Traversal (cont'd)

Proof of Theorem

For $1 \leq i \leq \ell$, consider the clustering of P_i induced by the set of centers T_i identified in Round 1, and let d_i be the maximum distance of a point of P_i from its cluster center. By reasoning as in the proof of the previous theorem, we can show that there exist $k+1$ points in P_i whose pairwise distances are all $\geq d_i$. At least two such points, say x, y must belong to the same cluster of the optimal clustering for P (not P_i!), with center c. Therefore,

$$d_i \leq d(x,y) \leq d(x,c) + d(c,y) \leq 2\Phi^{\text{opt}}_{\text{kcenter}}(k).$$

Consider now the clustering of T induced by the set of centers S identified in Round 2, and let d_T be the maximum distance of a point of T from its cluster center. The same argument as above, shows that $d_T \leq 2\Phi^{\text{opt}}_{\text{kcenter}}(k)$.

Analysis of MR-Farthest-First Traversal (cont'd)

Proof of Theorem (cont'd).

Finally, consider an arbitrary point $p \in P$ and suppose, w.l.o.g., that $p \in P_i$ for some $1 \leq i \leq \ell$. By combining the two observations made before, we conclude that there must exist a point $t \in T_i$ (hence $t \in T$) and a point $c \in S$, such that

$$\begin{aligned} d(p,t) &\leq 2\Phi^{\text{opt}}_{\text{kcenter}}(k) \\ d(t,c) &\leq 2\Phi^{\text{opt}}_{\text{kcenter}}(k). \end{aligned}$$

Therefore, by triangle inequality, we have that

$$d(p,c) \leq d(p,t) + d(t,c) \leq 4\Phi^{\text{opt}}_{\text{kcenter}}(k),$$

and this immediately implies that $\Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}}) \leq 4 \cdot \Phi^{\text{opt}}_{\text{kcenter}}(k)$. □

Observations on MR-Farthest-First Traversal

- The sequential Farthest-First Traversal algorithm is used both to extract the coreset and to compute the final set of centers. It provides a good coreset since it ensures that any point not in the coreset be well represented by some coreset point.
- The main feature of MR-Farthest-First Traversal is that while only small subsets of the input are processed at once, and many of them in parallel, the final approximation is not too far from the best achievable one.
- By selecting $k' > k$ centers from each subset P_i in Round 1, the quality of the final clustering improves. In fact, it can be shown that when P satisfy certain properties and k' is sufficiently large, MR-Farthest-First Traversal returns a $(2+\epsilon)$-approximation for any constant $\epsilon > 0$, still using sublinear local space and linear aggregate space.