Clustering

Part 1

The Clustering Problem

Given a set of points belonging to some space, with a notion of distance between points, group the points into a number of *clusters* such that

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

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

Example

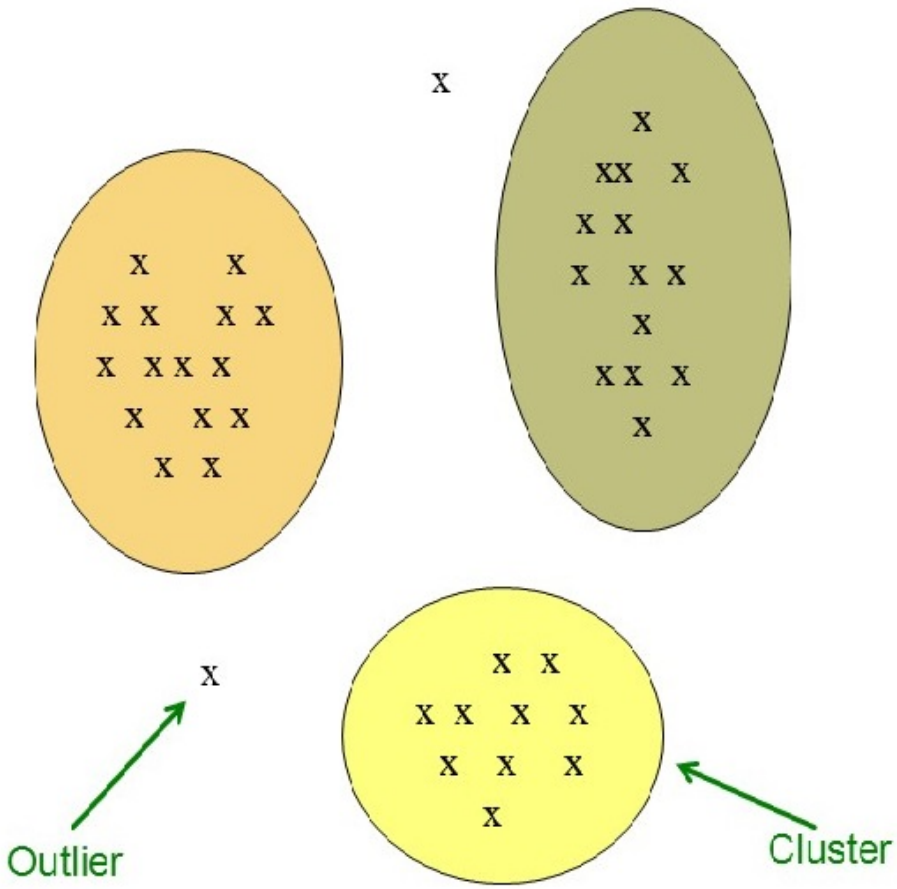


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

The clustering problem is a hard one!

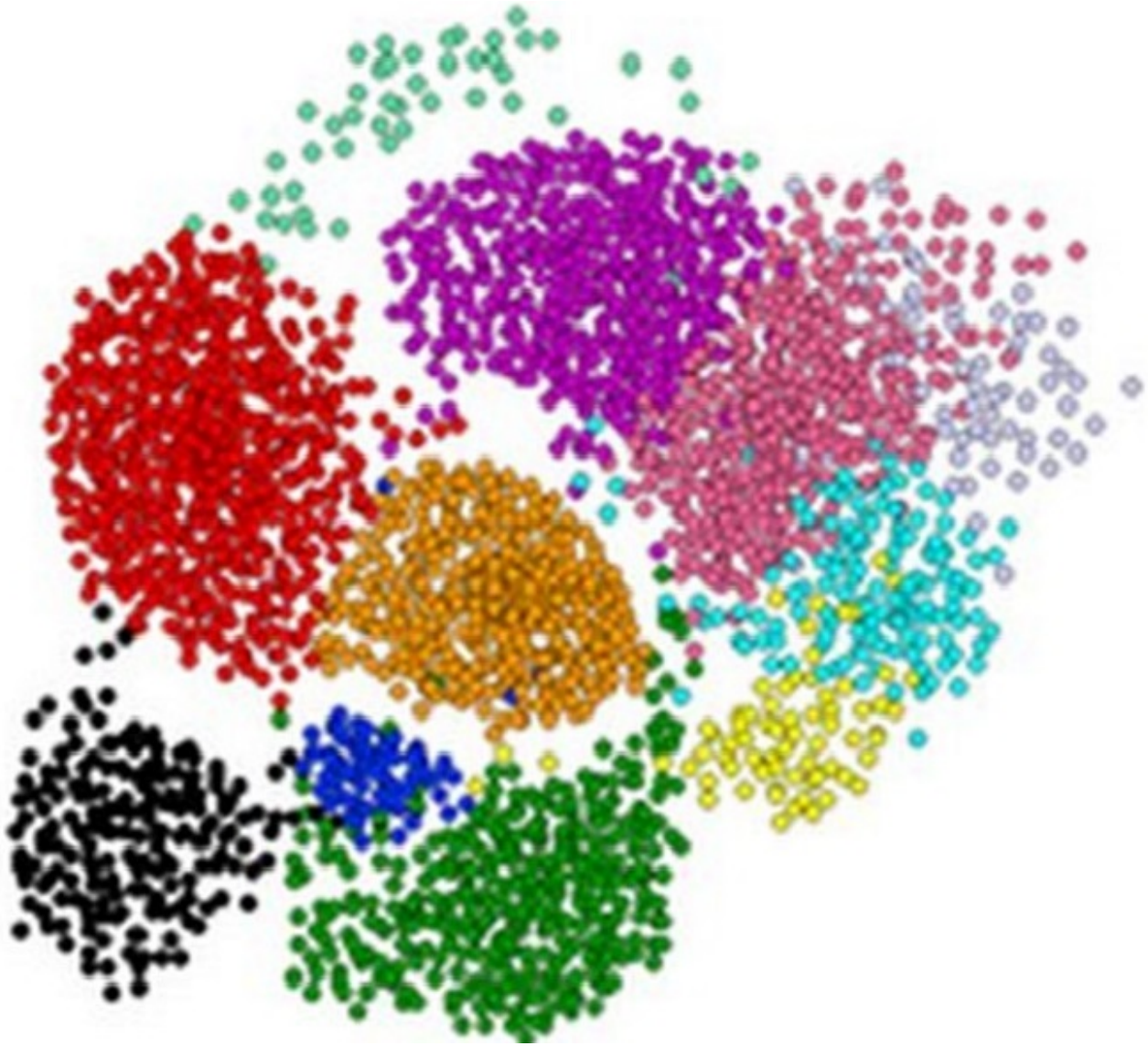


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

Applications

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

1. Preprocessing step for other mining/learning tasks. E.g., 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 pruposes: 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

Usually, the points in input to a clustering problem come 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)

Distance functions

KEY STEP: choice of a proper distance functions.

- Euclidean distances. Input = subset of $\Re^n$. Given $X, Y \in \Re^n$, with $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$, the $L_r$norm is defined as

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

  - L_2 : standard distance in $\Re^n$ (also denoted by $|| \cdot ||$)
  - L_1 : referred to as *Manhattan distance*, it is the sum of the absolute differences of coordinates in each dimension. Used in grid-like environments
  - L_∞ : (limit of L_r when r tends to ∞) it is the maximum absolute differences of coordinates, over all dimensions.

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

$$1 - \frac{|S \cap T|}{|S \cup T|}.$$

  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.

Distance functions (cont'd)

- Cosine distance. Used when points are *vectors*, for example with integral or binary coordinates. It is the angle between the two vectors, which is measured as follows. Let $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ be two vectors. Their cosine distance is

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

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

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

Distance functions (cont'd)

Observations on the cosine distance

- It takes values in $[0, \pi]$, or in $[0, \pi/2]$ in case of non-negative coordinates. Dividing by π (or $\pi/2$) the range becomes $[0, 1]$.
- In order to statisfy the second property of distance functions for metric spaces, scalar multiple of a vector must be regarded as the same vector
- The cosine distance can be used for 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 occurences in X of the i-th word of the alphabet.

Distance functions (cont'd)

- Edit distance. Used for strings. Given two strings X and Y, their edit distance is the minimum number of *deletions* of *insertions* that must be applied to transform X into Y. For example: if $X = \mathrm{ABCDE}$ and $Y = \mathrm{ACFDEG}$ we can transform X into Y as follows

  - Delete B;
  - Insert F after C;
  - Insert G after E;

  It is easy to see that in this case the edit distance is 3.

  The edit distance can be obtained as $|X| + |Y| - 2|\mathrm{LCS}(X, Y)|$, where $\mathrm{LCS}(X, Y)$ is the length of the Longest Common Subsequence of X and Y, which can be computed in $O(|X| \cdot |Y|)$ time thorugh 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. For example: for $X = (0, 1, 1, 0, 1)$ and $Y = (1, 1, 1, 0, 0)$ (i.e., $n = 5$), the Hamming distance is 2, since they differ in the first and last coordinates.

Exercise

Show that all distance functions introduced before satisfy the four requirements for a metric space (listed in the definition of metric space).

Proving the triangle inequality for the Jaccard distance is a bit tricky. See proof in Section 3.5.3 of textbook [LRU14]. There is also a direct proof in [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 an example, consider N random points in $[-1,1]^d$, where for each point the i-th coordinate is a random number in $[-1,1]$ drawn with uniform probability independently of the other coordinates and of the other points.

Curse of dimensionality (cont'd)

Let $X = (x_1, x_2, \ldots, x_d)$ and $Y = (y_1, y_2, \ldots, y_d)$ any two such points. Their L_2 distance is

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

Using Chebyshev's inequality and other standard probability facts, one can prove the following two properties hold with probability increasing with d

- $d(X, Y) \in \Theta\left(\sqrt{d}\right)$, i.e., within constant from maximum
- The cosine of the angle formed by X, Y (seen as vectors) w.r.t. to origin, i.e.,

$$\left(\frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2} \sqrt{\sum_{i=1}^{n} (y_i)^2}} \right)$$

  is $O\left(d^{-c}\right)$ for some constant $c > 0$, therefore the angle tends to $\pi/2$ (i.e., X, Y orthogonal). as d grows large.

Types of clusterings

Given a set of points in a metric space, hence a distance between points, one has to define an objective function to optimize. The objective function also allows to compare different clusterings.

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 and, possibly, the objective function depends on the chosen centers. Centers are usually input points, but, if the metric space is $\Re^d$, centers not belonging to the input points can be allowed (e.g., the average of the points)
- Clusters are required to be disjoint

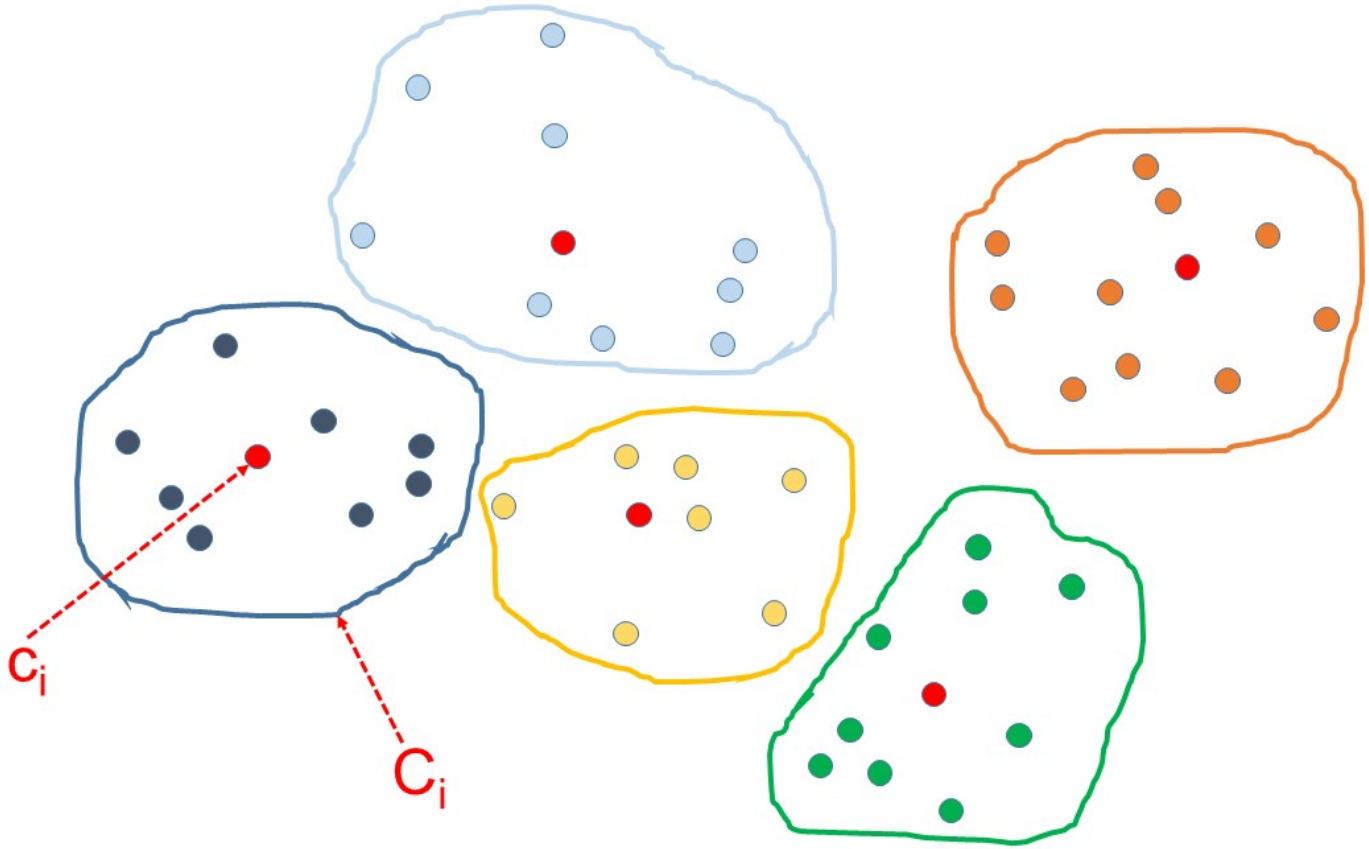
Center-based k-clusterings

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 k-clusterings (cont'd)

Center-based clustering problems aim at computing a k-clustering $\mathcal{C}$ which minimizes a suitable objective function $\Phi(\mathcal{C})$. The following are three popular objective functions.

- (k-center clustering)

$$\Phi_{\text{kcenter}}(\mathcal{C}) = \max_{i=1}^{k} \max_{a \in C_i} d(a, c_i).$$

- (k-means clustering)

$$\Phi_{\text{kmeans}}(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in C_i} (d(a, c_i))^2.$$

- (k-median clustering)

$$\Phi_{\text{kmedian}}(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in C_i} d(a, c_i).$$

Observations

- All aforementioned problems (k-center, k-means, k-median) are NP-hard. Hence, in general it is difficult to compute 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 *facilities* and a set C of *clients* are given and the objective is to find a subset of at most k facilities to open and an assignment of clients to them, so to minimize some the maximum distance of the sum of the distances between clients and their 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 clustering around these centers assign to each c_i the set of points that are closer to c_i than to any other center (ties broken arbitrarily). In the following, 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** $C_i \leftarrow \emptyset$
for each $p \in P$ **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 $\mathcal{C}$

k-center clustering

Farthest-First Traversal: algorithm

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

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

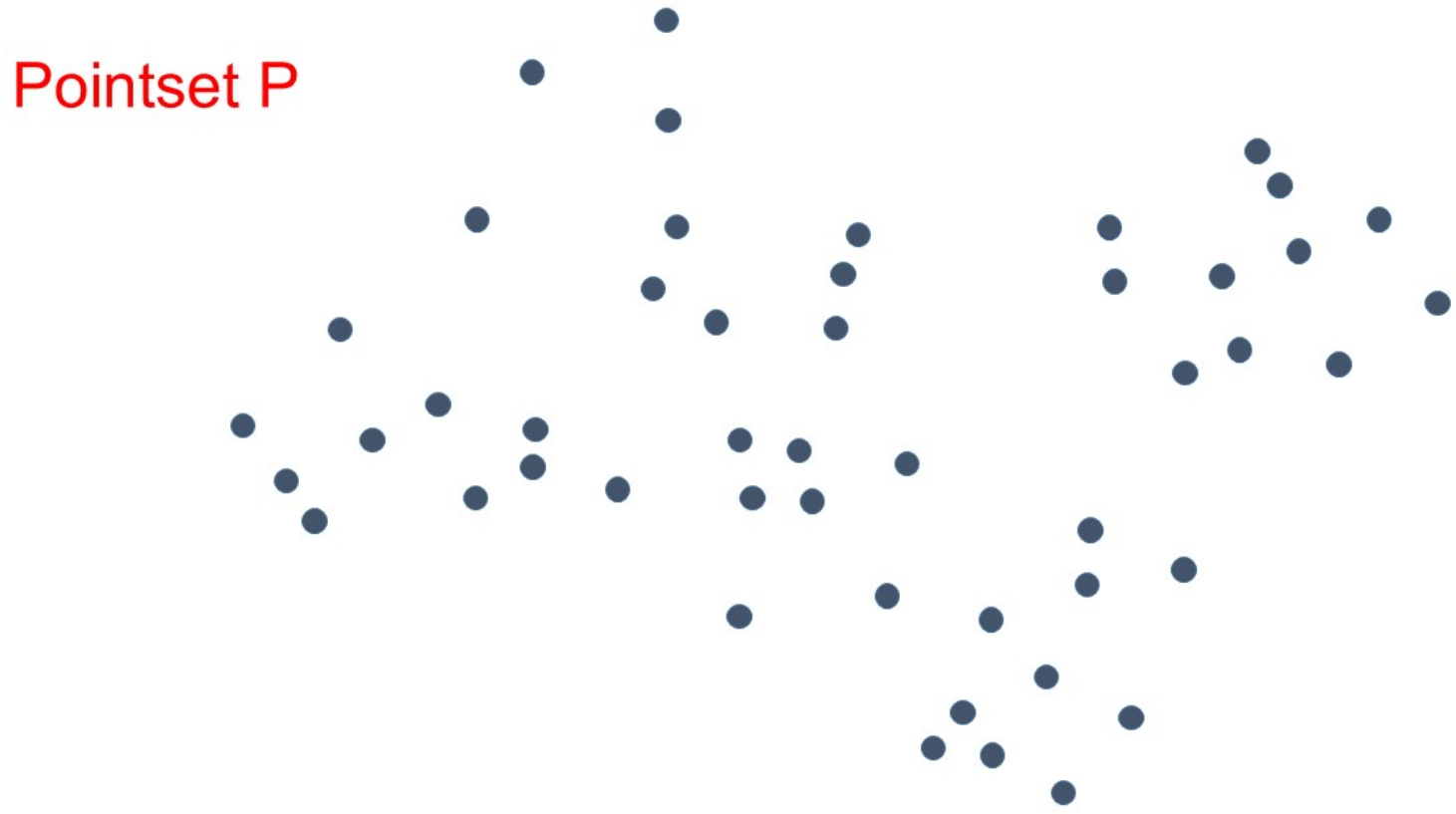
Output k-clustering $\mathcal{C} = (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$ of P.

Farthest-First Traversal: algorithm (cont'd)

$S \leftarrow \emptyset$
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)

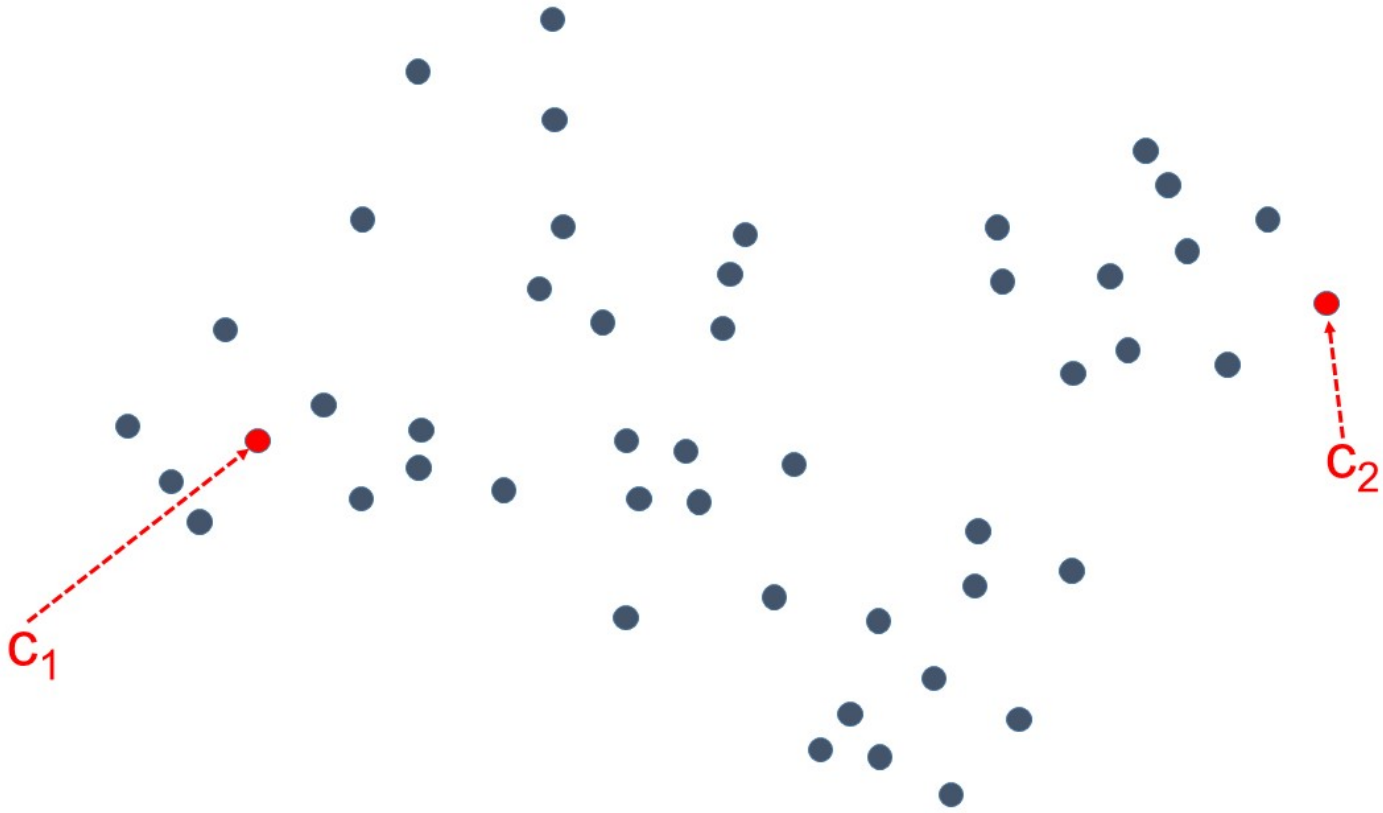
- $d(c_i, S)$ denotes the minimum distance of c_i from a point of S. That is, $d(c_i, S) = \min\{c \in S \ : \ d(c_i, c)\}$
- The assignment of points to clusters can be accomplished while determining the centers in the first for-loop.

Farthest-First Traversal: example

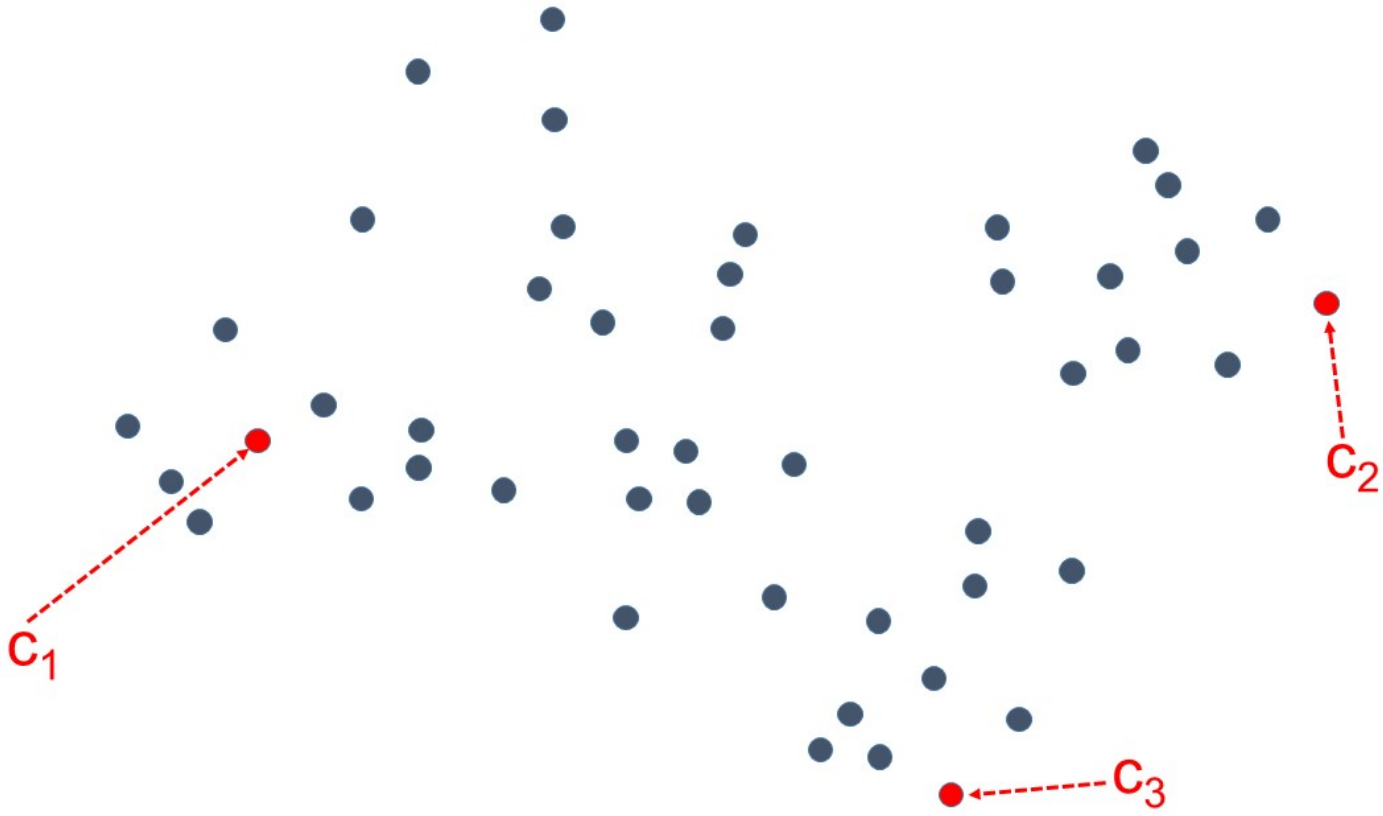


Farthest-First Traversal: example

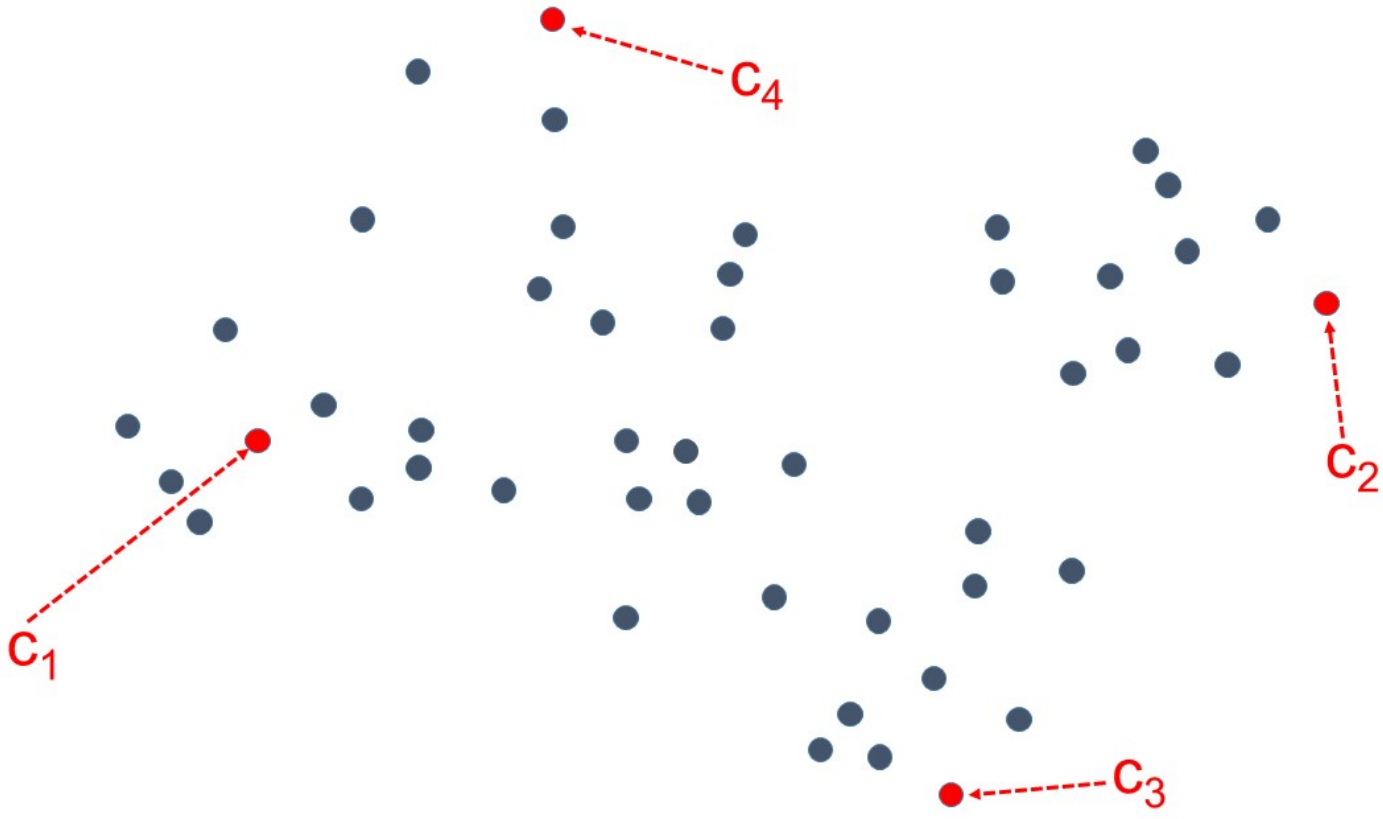
Farthest-First Traversal: example



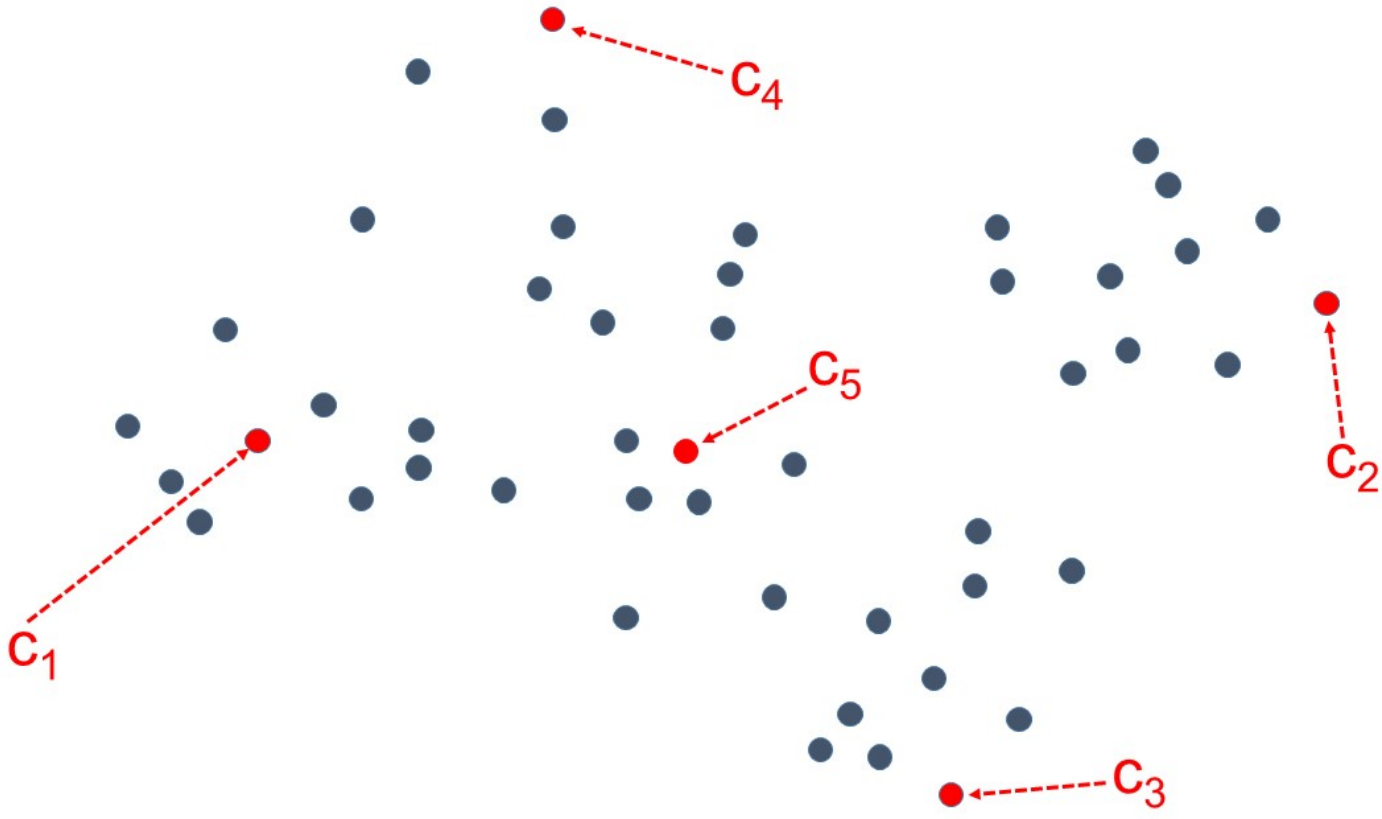
Farthest-First Traversal: example



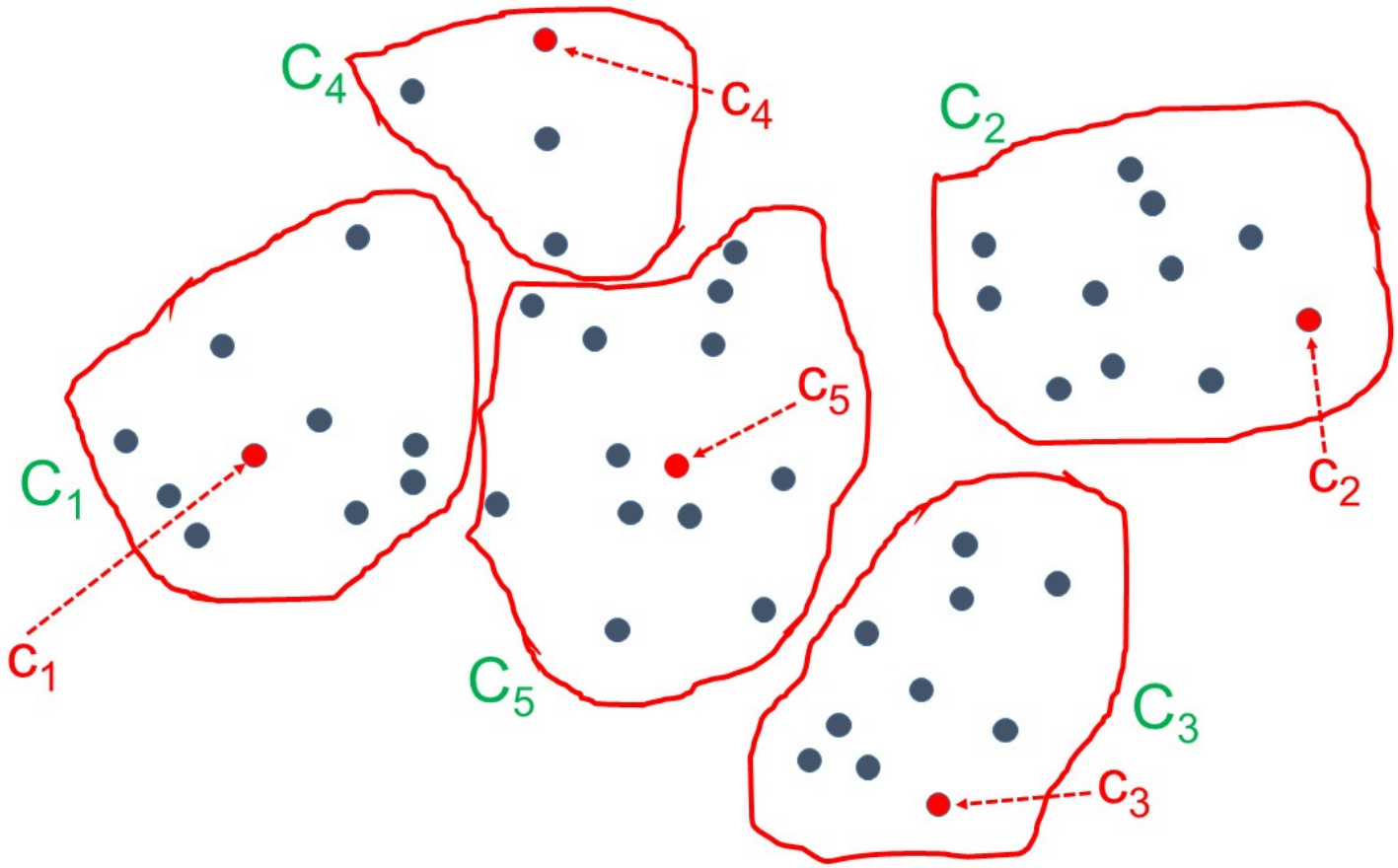
Farthest-First Traversal: example



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 $p \in P - S$ knowns 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:

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

Farthest-First Traversal: analysis (cont'd)

Proof of Theorem

Let

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

There must exists a cluster C_i and a point $q \in C_i$ such that $d(q, c_i) = \Phi_{\text{kcenter}}(\mathcal{C}_{\text{alg}})$ (i.e., q is the point which maximizes the distance from its cluster's center). Observe that:

- $d(q, c_j) \geq d(q, c_i)$, for every $j \neq i$ since q is assigned to C_i
- $d(c_{j1}, c_{j2}) \geq d(q, c_i)$, for every $1 \leq j1 \neq j2 \leq k$, since otherwise q would have been a better choice as center at some point during the algorithm (exercise)

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. □

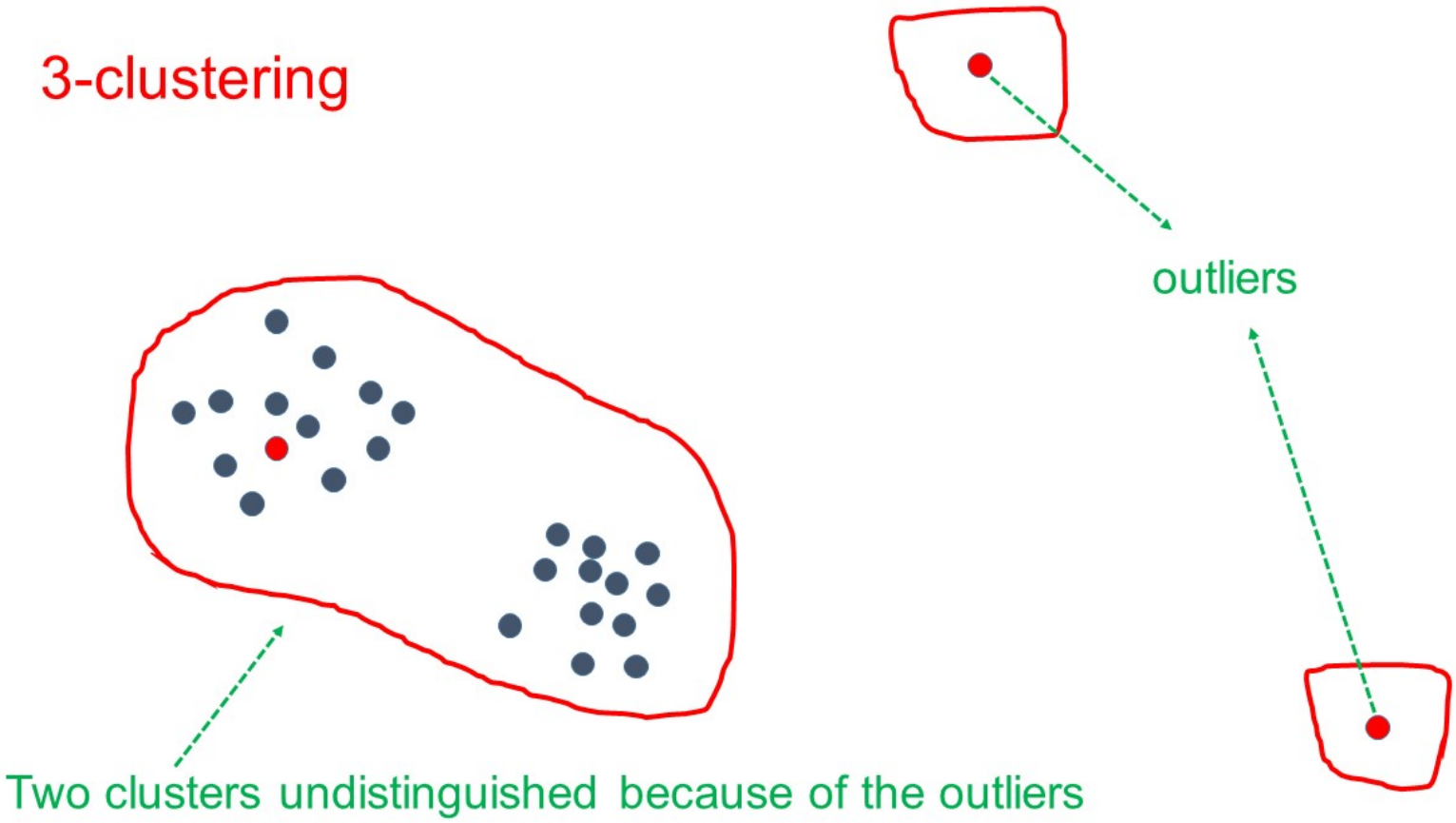
Observations on k-center clustering

- 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

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

MapReduce algorithm for k-center clustering (MR-Farthest-First Traversal)

- 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.
- Round 2: Let T_i be the set of centers of the k-clustering computed from P_i, for $1 \leq i \leq \ell$. Gather together $T = \cup_{i=1}^{k} T_i$ and run (using a single reducer) the Farthest-First Traversal algorithm on T.
- Round 3: Let $S = \{c_1, c_2, \ldots, c_k\}$ be the set of centers of the k-clustering computed from T. Compute the final clustering by running Partition(P_j, S) for each $1 \leq j \leq \ell$. Let $C_i^{(j)}$ be the cluster centered at c_i computed by Partition(P_j, S). The final cluster centered at c_i is $C_i = \cup_{j=1}^{\ell} C_i^{(j)}$.

Exercise: Specify in detail the map and reduce phases of each round.

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 $M_A = O(N)$

Note that each reducer in this case performs $O\left(k \cdot \sqrt{N \cdot k}\right)$ local computation, which is a substantial improvement w.r.t. the $O(k \cdot N)$ complexity of the sequential Farthest-First algorithm.

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:

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

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

Proof of Theorem

Consider an arbitrary index i, with $1 \leq i \leq \ell$. Let $\mathcal{C}_{\text{alg}}(P_i)$ be the k-clustering returned by the Farthest-First Traversal algorithm applied to P_i is Round 1, and let d_i be the maximum distance of a point of P_i from its cluster center (which belongs to T_i).

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,

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

Let $\mathcal{C}_{\text{alg}}(T)$ be the k-clustering returned by the Farthest-First Traversal algorithm applied to T in Round 2, and let d_T be the maximum distance of a point of T from its cluster center (which belongs to S). 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).

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

k-means clustering

Properties of Euclidean spaces

Let $X = (X_1, X_2, \ldots, X_D)$ and $Y = (Y_1, Y_2, \ldots, Y_D)$ be two points in $\Re^D$. Recall that their Euclidean distance is

$$d(X, Y) = \sqrt{\sum_{i=1}(X_i - Y_i)^2} \stackrel{\triangle}{=} ||X - Y||.$$

Definition

The centroid of a set P of N points in $\Re^D$ is

$$c(P) = \frac{1}{N} \sum_{X \in P} X,$$

where the sum is component-wise.

Observe that $c(P)$ does not necessarily belong to P

Properties of Euclidean spaces (cont'd)

Lemma

The centroid $c(P)$ of a set $P \subset \Re^D$ is the point of $\Re^D$ which minimizes the sum of the square distances to all points of P.

Proof

Consider an arbitrary point $Y \in \Re^D$. We have that

$$\begin{aligned}
\sum_{X \in P} (d(X,Y))^2 &= \sum_{X \in P} ||X - Y||^2 \\
&= \sum_{X \in P} ||X - c_P + c_P - Y||^2 \\
&= (\sum_{X \in P} ||X - c_P||^2) + N||c_P - Y||^2 + \\
&\quad +2(c_P - Y) \cdot \left(\sum_{X \in P} (X - c_P) \right),
\end{aligned}$$

where "$\cdot$" denotes the inner product.

Properties of Euclidean spaces (cont'd)

Proof. (cont'd).

By definition of c_P we have that $\sum_{X \in P}(X - c_P) = (\sum_{X \in P} X) - Nc_P$ is the all-0's vector, hence

$$\begin{aligned}
\sum_{X \in P}(d(X,Y))^2 &= \sum_{X \in P} ||X - c_P||^2 + n||c_P - Y||^2 \\
&\geq \sum_{X \in P} ||X - c_P||^2 \\
&= \sum_{X \in P}(d(X, c_P))^2
\end{aligned}$$

□

Observation: The lemma implies that when seeking a k-clustering for points in $\Re^D$ which minimizes the kmeans objective, the best center to select for each cluster is its centroid (assuming that centers can be selected outside the input points).

k-means algorithm

- Also known as Lloyd's algorithm (developed by Stuart Lloyd in 1957). In 2006 it was listed as one of the top-10 most influential data mining algorithms (A-Priori is also one of these). It is considered among the most popular clustering algorithm used in both scientific and industrial applications.
- It focuses on Euclidean spaces
- It relates to a generalization of the Expectation-Maximization algorithm

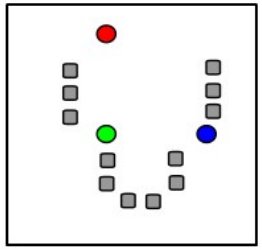
k-means algorithm (cont'd)

Input Set P of N points from $\Re^D$, integer $k > 1$

Output k-clustering $\mathcal{C} = (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$ of P, where centers need not belong to P

$S \leftarrow$ `arbitrary set of` k `points in` $\Re^D$
$\Phi \leftarrow \infty$; `stopping-condition` $\leftarrow$ `false`
while `(!stopping-condition)` **do**
  $(C_1, C_2, \ldots, C_k; S) \leftarrow$ `Partition`(P, S)
  for $i \leftarrow 1$ **to** k **do** $c_i' \leftarrow$ `centroid of` C_i
  $\mathcal{C} \leftarrow (C_1, C_2, \ldots, C_k; c_1', c_2', \ldots, c_k')$
  if $\Phi_{\text{kmeans}}(\mathcal{C}) < \Phi$ **then**
    $\Phi \leftarrow \Phi_{\text{kmeans}}(\mathcal{C})$
    $S \leftarrow \{c_1', c_2', \ldots, c_k'\}$
  else `stopping-condition` $\leftarrow$ `true`
return $\mathcal{C}$

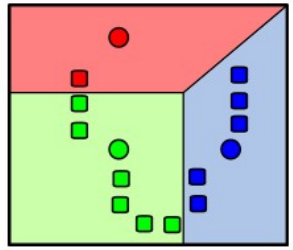
Example

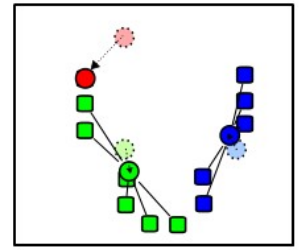


Initial centers

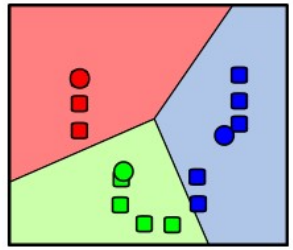
Partition *Calculation of centroids*

Iteration 1:





Iteration 2:



. . .

k-means algorithm: analysis

Theorem

The k-means algorithm always terminates, that is, it converges to a (local) optimum.

Proof.

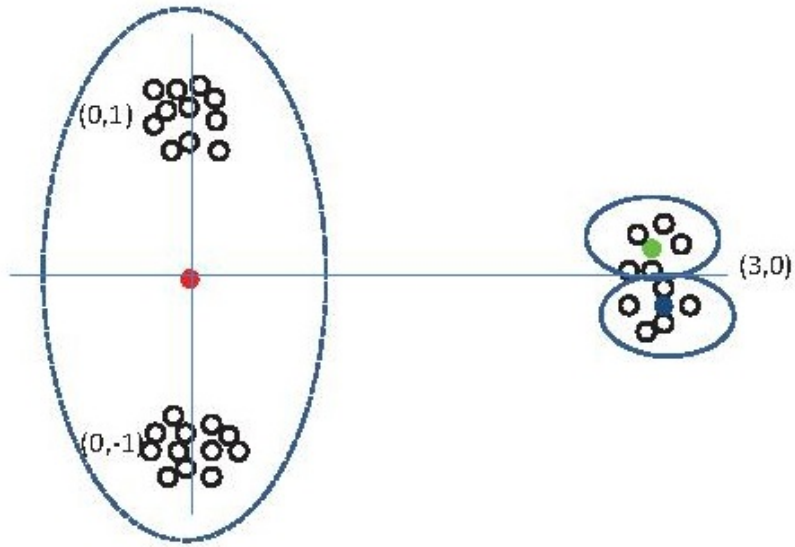
Observe that values of the objective function for the k-clusterings $\mathcal{C}$ computed in the iterations of the while-loop (except for the last iteration) form a strictly decreasing sequence. Therefore, the clusterings $\mathcal{C}$ generated at all iterations (except for the last one) must be distinct.

The clustering $\mathcal{C}$ generated at each iteration is uniquely determined by the set of clusters $C_1, C_2, \ldots, C_k$ returned by Partition.

Thus, all iterations (except for the last one) generate different sets of clusters. The theorem follows since there are k^N of possible partitions of P into k clusters (counting also permutations of the same partition). □

Observations on k-means algorithm

- The k-means algorithm may be trapped into a local optimum whose value of the objective function can be much larger than the optimal value. Consider the following example and suppose that $k = 3$.



If initially one center falls among the points on the left side, and two centers fall among the points on the right side, it is impossible that one center moves from right to left, hence the two obvious clusters on the left will be considered as just one cluster.

Observations on k-means algorithm (cont'd)

- While the k-means algorithm surely terminates, the number of iterations can be exponential in the input size.
- Besides the trivial k^N upper bound on the number of iterations, more sophisticated studies proved an $O\left(N^{kD}\right)$ upper bound which is improves upon the trivial one, in scenarios where k and D are small
- Some recent studies proved also a $2^{\Omega\left(\sqrt{N}\right)}$ lower bound on the number of iterations in the worst case.
- Despite the not so promising theoretical results, emprical studies show that, in practice, k-means requires much less than N iterations.
- In order to improve performance, without sacrificing the quality of the solution too much, one could stop the algorithm earlier, e.g., when the value of the objective function decreases by a small additive factor.

Effective initialization

- The quality of the solution and the speed of convergence of k-means depend considerably from the choice of the initial set of centers (e.g., consider the previous example)
- Typically, initial centers are chosen at random, but there are more effective ways of selecting them
- In a paper presented at ACM-SODA'07, D. Arthur and S. Vassilvitskii developed k-means++, a careful center initialization strategy that yields clusterings not too far from the optimal ones.
- A parallel variant of k-means++ (k-means|| [B+12]) is implemented in the Spark MLlib.

k-means++

Algorithm k-means++ uses the following procedure to select the set S of k centers at the start of k-means.

$c_1 \leftarrow$ `random point chosen from` P `with uniform probability`
$S \leftarrow \{c_1\}$
for $i \leftarrow 2$ **to** k **do**
  for each $p \in P - S$ **do** $d_p \leftarrow \min_{c \in S} d(c, p)$
  $c_i \leftarrow$ `random point chosen from` $P - S$`, where a point` P
    `is chosen with probability` $(d_p)^2 / (\sum_{q \in P-S} (d_q)^2)$
  $S \leftarrow S \cup \{c_i\}$

Observe that the probability values $(d_p)^2 / (\sum_{q \in P-S} (d_q)^2)$*, used to select the next center in each iteration of the for-loop, define a probability distribution.*

After the initialization, k-means++ proceeds exactly like k-means

k-means++

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

Theorem (Arthur-Vassilvitskii'07)

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

$$E[\Phi_{\text{kmeans}}(\mathcal{C}_{\text{alg}})] \leq 8(\ln k + 2) \cdot \Phi^{\text{opt}}_{\text{kmeans}}(k),$$

where the expectation is over all possible initial sets of centers produced by the novel randomized initialization strategy.

Observation: The k centers computed in the initialization phase already provide the above guarantee on the clustering quality. The successive iterations of the k-means algorithm can only improve this quality.

Observations on k-means clustering

- In essence, k-means clustering aims at minimizing cluster variance. It is typically used in Euclidean spaces and works well for ball-shaped clusters. In fact, the centroids used in the k-means algorithm make no sense in non-Euclidean spaces
- Because of the quadratic dependence on distances, k-means clustering is rather sensitive to outliers. However, the fact that the objective function sums all distances, makes it more robust than the k-center objective to noise and outliers.
- In practice, if the inital centers are suitably selected and a careful implementation is used, the k-means algorithm is rather fast and accurate. This is one of the main reasons of the popularity of the algorithm

k-median clustering

Partitioning Around Medoids (PAM) algorithm

- Devised by L. Kaufman and P.J. Rousseeuw in 1987
- Based on the local search optimization strategy
- Unlike the k-means algorithm, cluster centers belong to the input pointset and, traditionally, are referred to as medoids

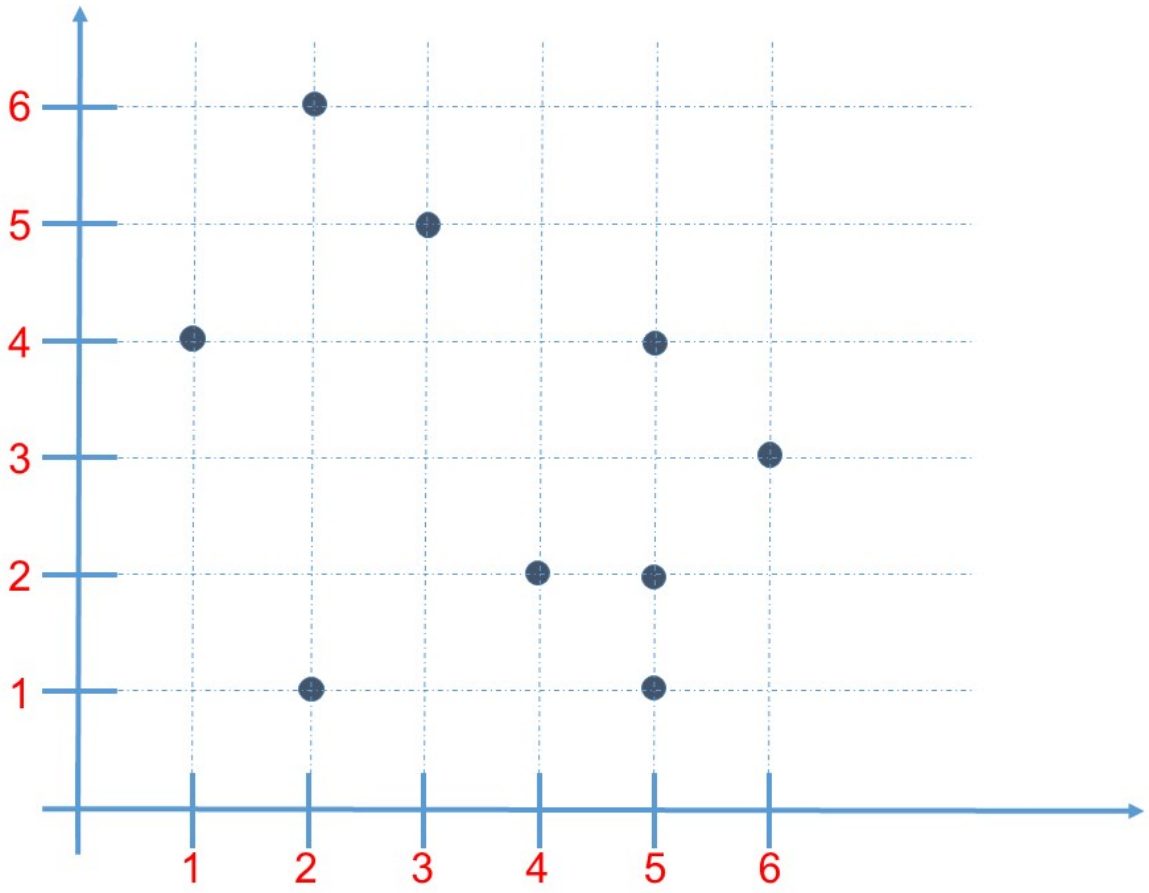
Input Set P of N points from a metric space (M, d), integer $k > 1$

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

PAM algorithm (cont'd)

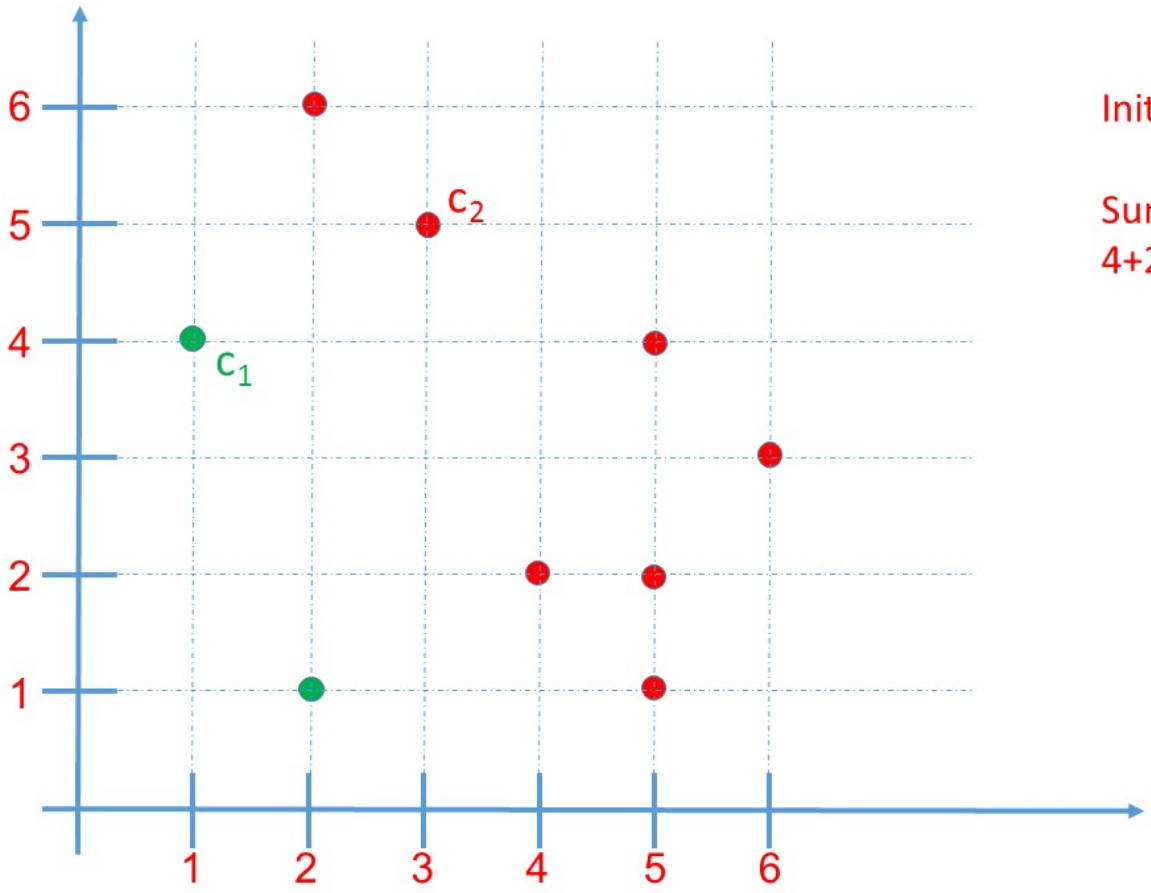
$S \leftarrow \{c_1, c_2, \ldots, c_k\}$ (arbitrary set of k points of P)
$\mathcal{C} \leftarrow$ Partition(P, S)
stopping-condition $\leftarrow$ false
while (!stopping-condition) **do**
  stopping-condition $\leftarrow$ true
  for each $p \in P - S$ **do**
    for each $c \in S$ **do** $S' \leftarrow (S - \{c\}) \cup \{p\}$
      $\mathcal{C}' \leftarrow$ Partition(P, S')
      if $\Phi_{\text{kmedian}}(\mathcal{C}') < \Phi_{\text{kmedian}}(\mathcal{C})$ **then**
        stopping-condition $\leftarrow$ false
        $\mathcal{C} \leftarrow \mathcal{C}'$
        exit both for-each loops
return $\mathcal{C}$

PAM algorithm: example



Manhattan distance

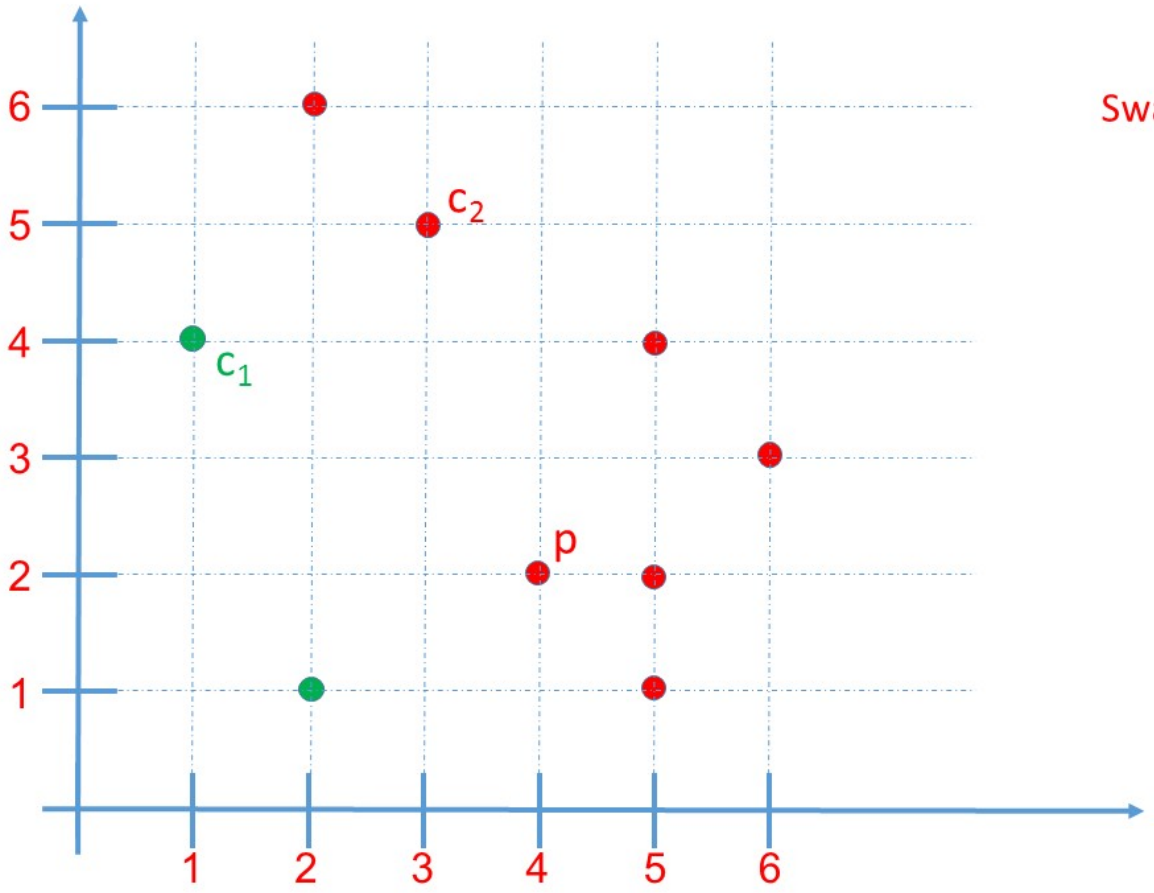
PAM algorithm: example



Initial Clusters

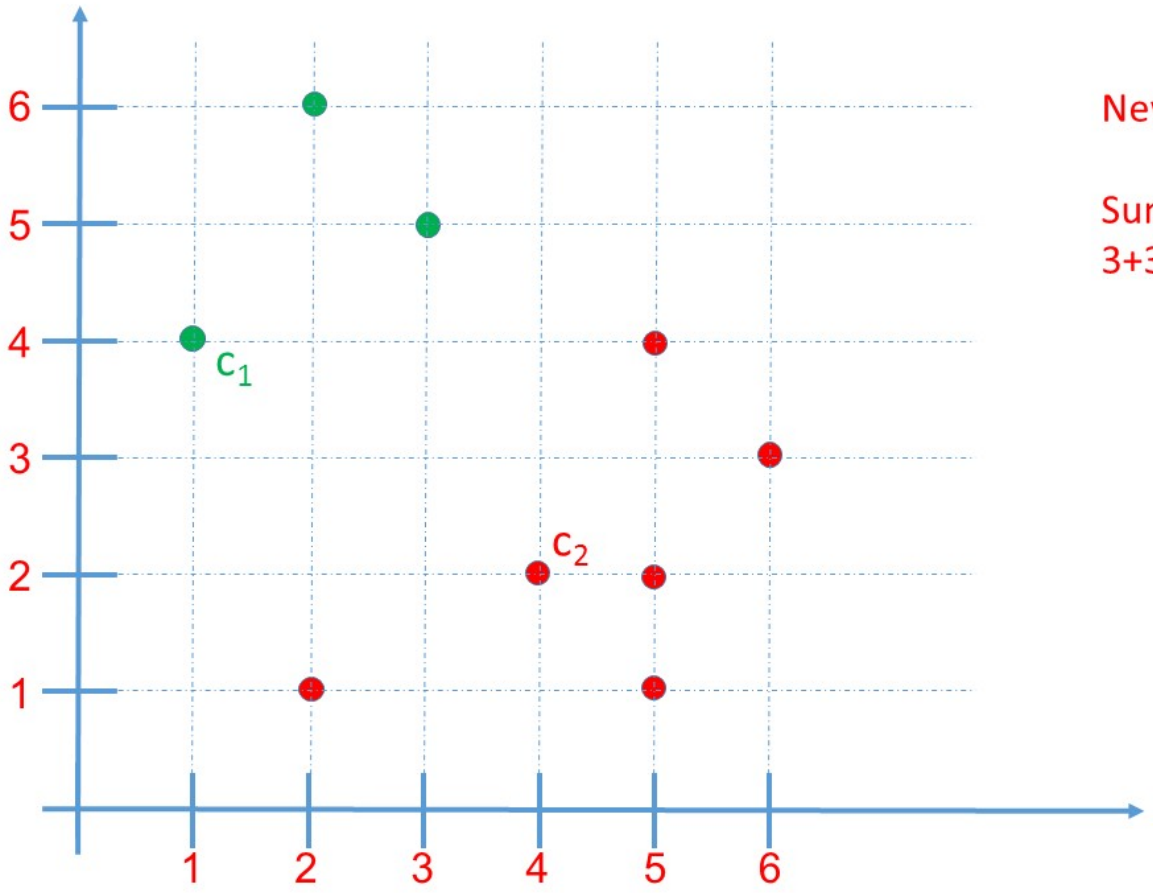
Sum of distances =
4+2+3+5+4+5+6 = 29

PAM algorithm: example



Swap: c_2 , p

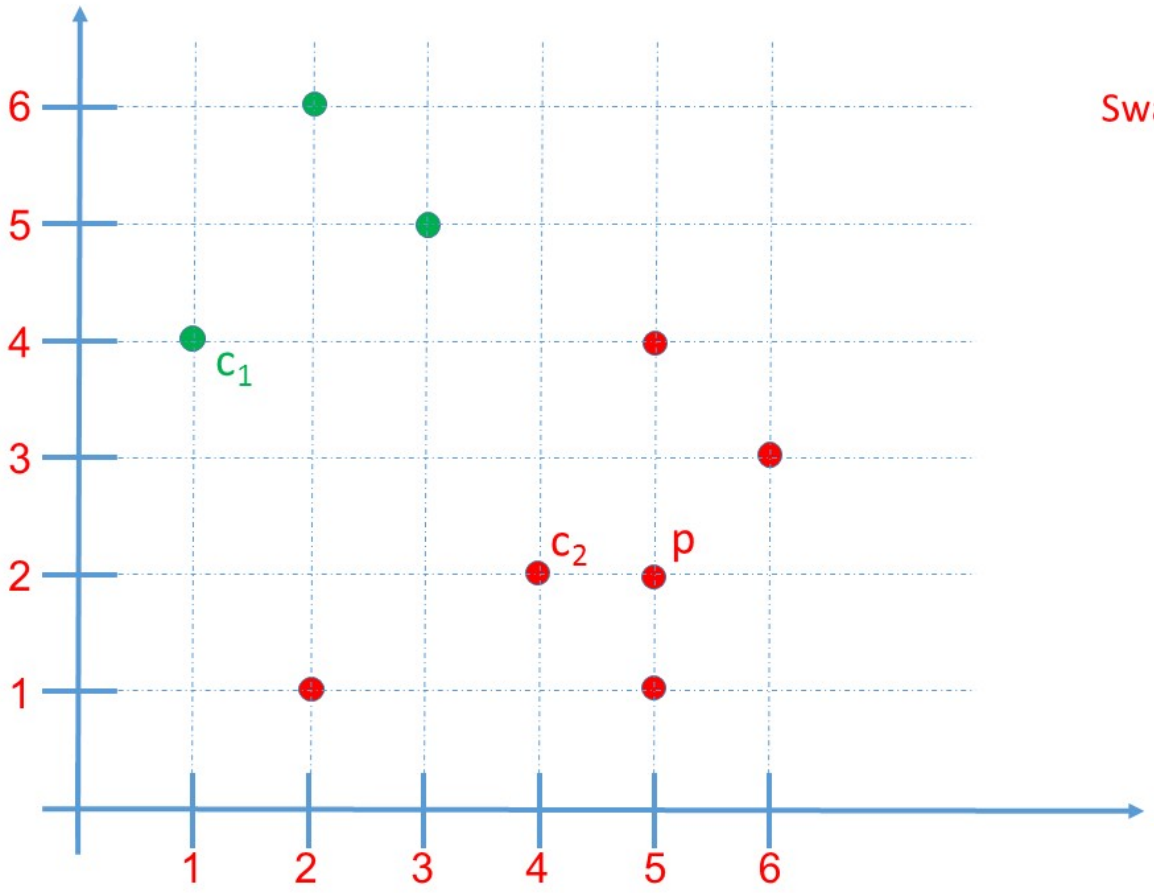
PAM algorithm: example



New Clusters

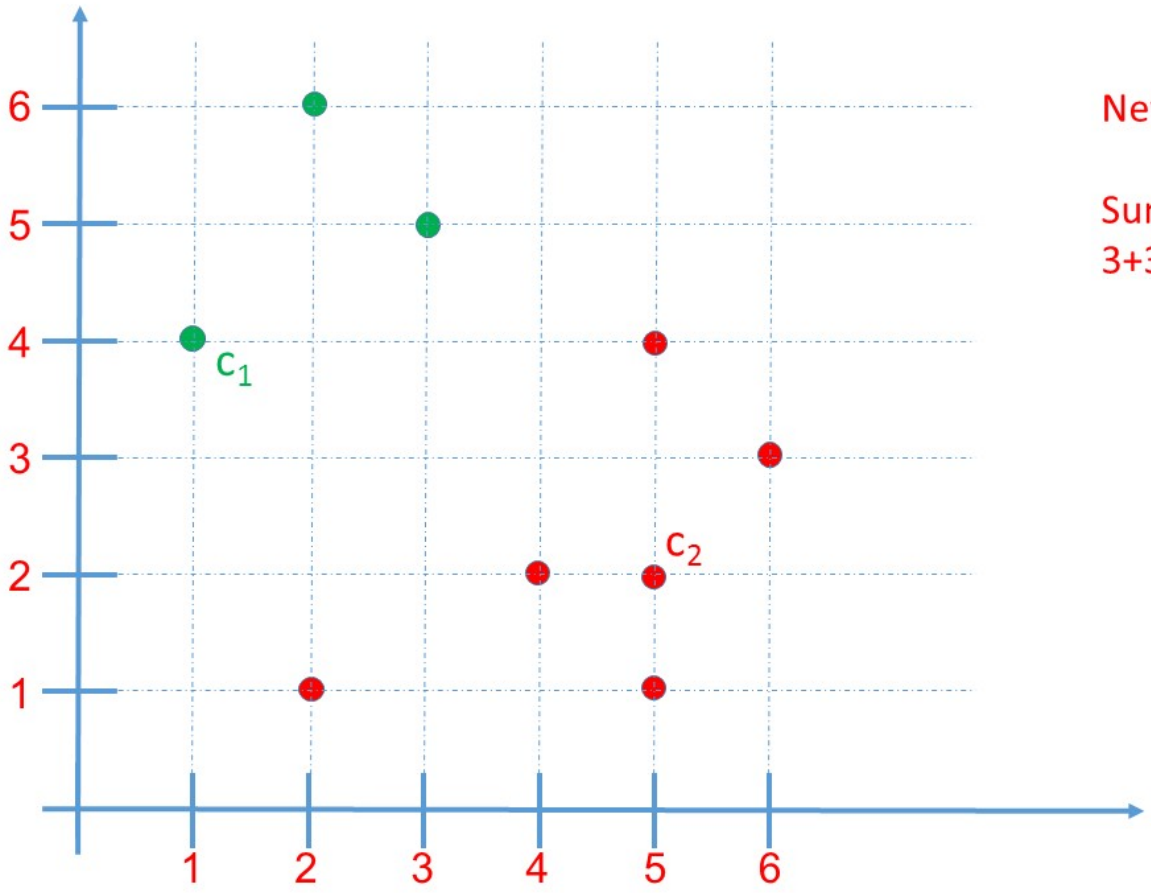
Sum of distances =
3+3+3+1+2+3+3 = 18

PAM algorithm: example



Swap: c_2 , p

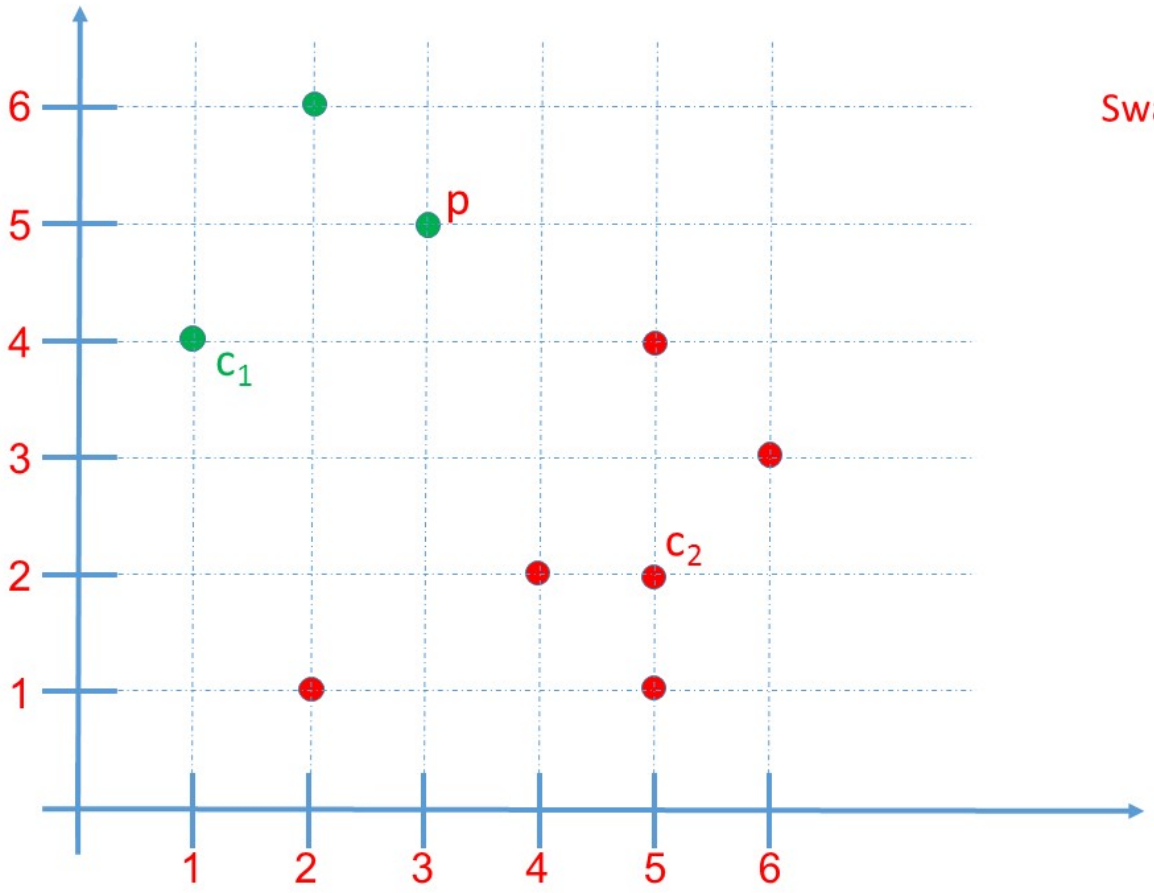
PAM algorithm: example



New Clusters

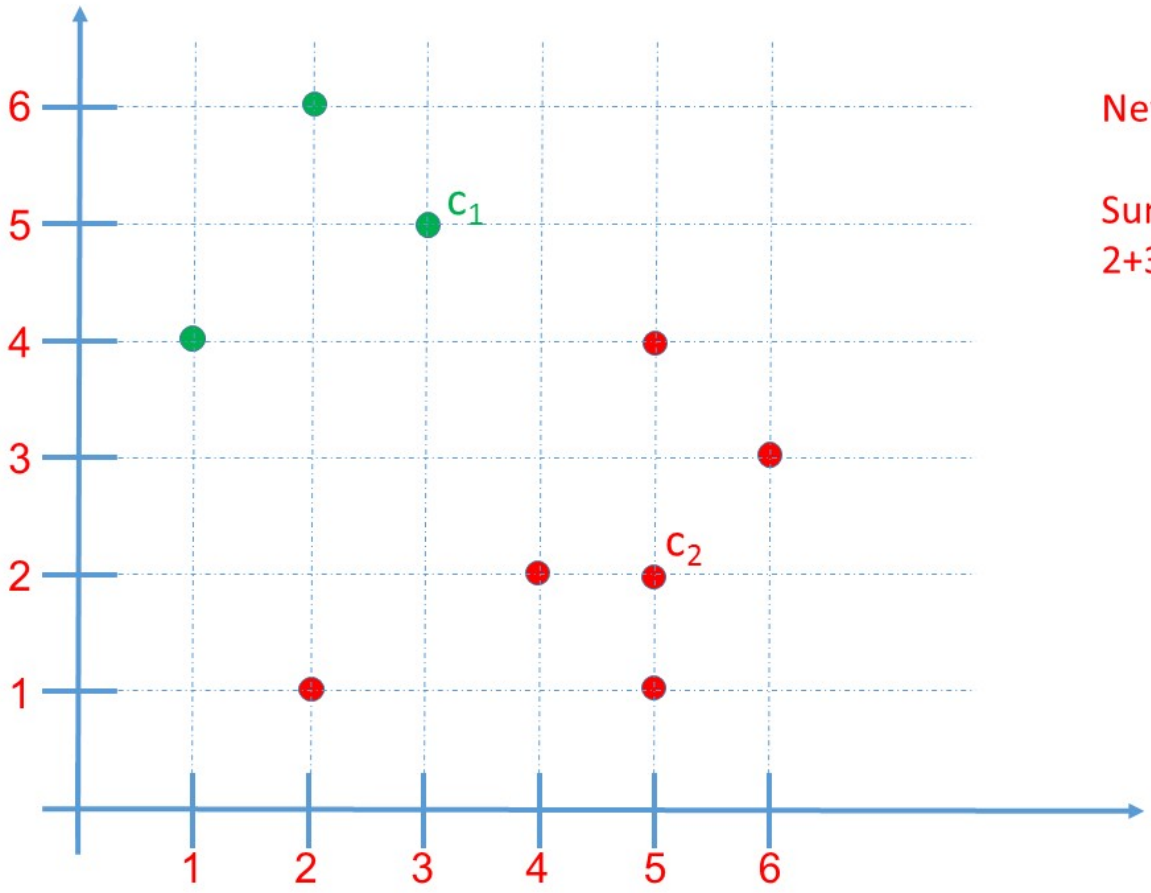
Sum of distances =
3+3+4+1+1+2+2 = 16

PAM algorithm: example



Swap: c_1 , p

PAM algorithm: example



New Clusters

Sum of distances =
2+3+4+1+1+2+2 = 15

PAM algorithm: analysis

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

Theorem (Arya et al.'04)

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

$$\Phi_{\text{kmedian}}(\mathcal{C}_{\text{alg}}) \leq 5 \cdot \Phi^{\text{opt}}_{\text{kmedian}}(k).$$

Remark: by imposing that in each iteration the new clustering $\mathcal{C}'$ decreases the objective function by at least $1 - \epsilon/(Nk)$, for some constant $\epsilon \in (0, 1)$, stopping the algorithm if this is not possibile, it can be proved that the algorithm executes a number of iterations polynomial in N and $\Phi_{\text{kmedian}}(\mathcal{C}_{\text{alg}})$ is a factor $f(\epsilon) \in O(1)$ away from the optimal value.

Observations on k-median clustering

- k-median is less sensitive to outliers than k-center and k-means since all distances (not squared) are taken into account
- The PAM algorithm works for any metric space and fatures provable performance and approximation guarantees. However, it can be quite slow in practice since in each iteration up to $(N - k) \cdot k$ swaps may need to be checked, and for each swap a new clustering must be computed
- A faster alternative is an adaptation of the k-means algorithm: in each iteration of the main while-loop, and for each current cluster C_i the new center (medoid) will be the point of C_i which mimimizes the sum of the distances to all other points of the cluster, instead of the centroid usednby k-means. This algorithm appears faster and still very accurate in practice (see [PJ09]).

How to pick the "right" value for k?

- Sometimes, the application provides a target value for k (e.g., the number of facilities we want to open)
- If such a target value of k is not known:
  - Find k-clusterings with geometrically larger values of k (i.e., $k = 2, 4, 8, \ldots$) and stop at $k = x$ if the value of the objective function (or some other metric) for $k = x$ does not improve "too much" with respect to $k = x/2$.
  - (Optional) Refine search of k in $[x/2, x]$.

Exercises

Exercise

For a pointset P from a metric space (M, d) and an integer $k > 1$, 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.

1. Show that $\Phi^{\text{opt}}_{\text{kcenter}}(k+1) \leq \Phi^{\text{opt}}_{\text{kcenter}}(k)$
2. Suppose that $\Phi^{\text{opt}}_{\text{kcenter}}(k)$ is attained by a clustering $(C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$. Show that for any point $p \in P$, with $p \in C_i$ for some i, we have that

$$C_i \subseteq \left\{ q \ : \ d(p,q) \leq 2\Phi^{\text{opt}}_{\text{kcenter}}(k) \right\}$$

Exercise

Specify in detail the map and reduce phases of each round of the MR-Farthest-First Traversal algorithm.

Exercises

Exercise

Argue that when $k = o(N)$ the MR-Farthest-First Traversal algorithm requires local space $M_L = o(N)$ and aggregate space $M_A = O(N)$.

Exercise

Let P be a set of N points from $\Re^n$ and let $k > 1$ be an interger. Using Markov's inequality and Chernoff bound show that by repeating k-means++ $c \log N$ times, for a suitable constant $c > 1$, and by taking the best clustering $\mathcal{C}_{\text{alg}}$ found among all repetitions, we have that

$$\Phi_{\text{kmeans}}(\mathcal{C}_{\text{alg}}) = O\left((\ln k)\Phi^{\text{opt}}_{\text{kmeans}}(k)\right),$$

with probability $\geq 1 - 1/N$.

References

LRU14 J. Leskovec, A. Rajaraman and J. Ullman. Mining Massive Datasets. Cambridge University Press, 2014. Sections 3.1.1 and 3.5, and Chapter 7

BHK16 A. Blum, J. Hopcroft, and R. Kannan. Foundations of Data Science. Manuscript, June 2016. Chapter 8

AV07 D. Arthur, S. Vassilvitskii. k-means++: the advantages of careful seeding. Proc. of ACM-SIAM SODA 2007: 1027-1035.

B+12 B. Bahmani et al. Scalable K-Means++. PVLDB 5(7):622-633, 2012

Arya+04 V. Arya et al. Local Search Heuristics for k-Median and Facility Location Problems. SIAM J. Comput. 33(3):544-562, 2004.

PJ09 H.S. Park, C.H. Jun. A simple and fast algorithm for K-medoids clustering. Expert Syst. Appl. 36(2):3336-3341, 2009