Clustering (Part 1)

Motivations for studying clustering:

* Foundamental primitive for data analysis
* Challenging when dealing with massive data
* Clustering can be used as preprocessing for extracting representative samples

OUTLINE

1. Definition of clustering
2. Basic notions:
   - Metric space.
   - Distance function.
   - Optimization problem and approximation algorithm.
3. Center-based clustering

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

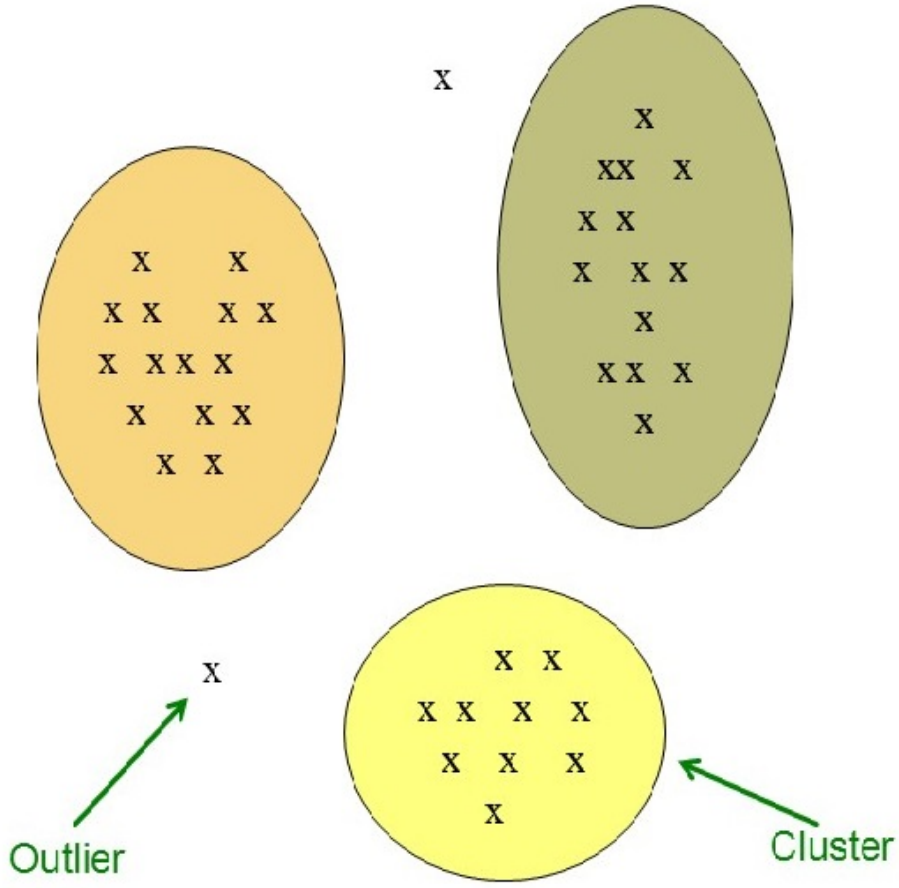


Figure from [Leskovec et al.: Mining Massive Datasets, 2014].

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

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

$t \gg r$

$t = d(x,z) \leq d(x,y) + d(y,z) = d(x,y) + r$

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

$\Rightarrow \quad d(x,y) - r \leq d(x,z) \leq d(x,y) + r$

$d(x,y) - r \leq t \leq d(x,y) + r$

$\Rightarrow$ $d(x,y)$ approximates t well

Distance functions

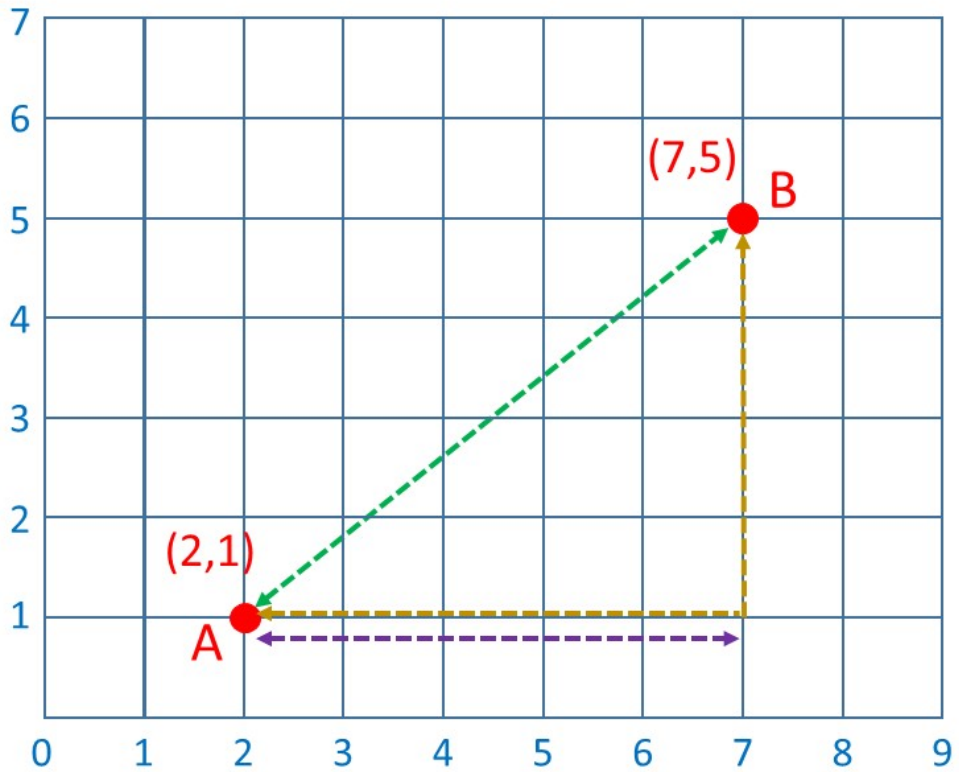
- Minkowski distances. Let $X, Y \in \Re^n$, with $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$. For $r > 0$ the $L_r - \text{distance}$ between X and Y (a.k.a. $L_r - \text{norm}$)

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

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

Distance functions

Examples of Minkowski distances:



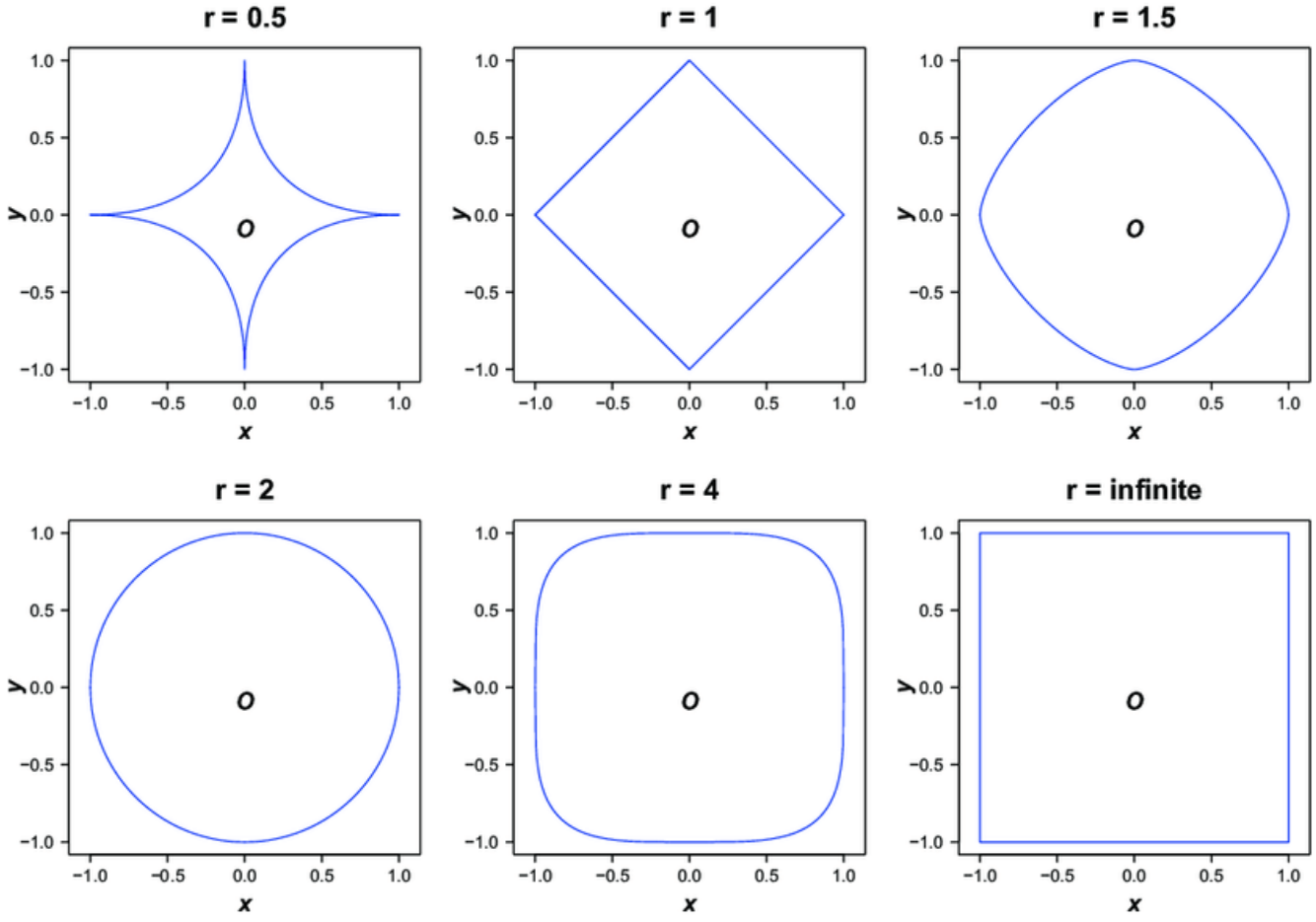
$d_{L1}(A,B) = (7-2)+(5-1) = 9$

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

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

Distance functions

Comparison of Minkowski distances:



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

- Cosine (or 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 cosine distance is the angle between them, that is,

$$d_{\text{cosine}}(X, Y) = \arccos\left(\frac{X \cdot Y}{||X|| \cdot ||Y||}\right) \quad \leftarrow \text{inner product}$$

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

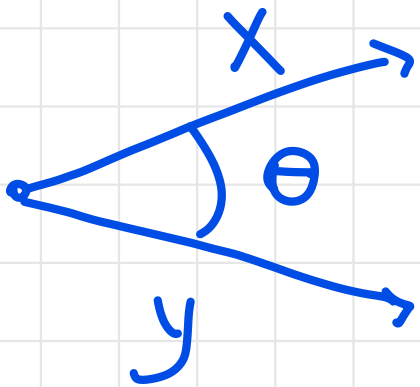
Example: For $X = (1, 2, -1)$ and $Y = (2, 1, 1)$

$$d_{\text{cosine}}(X, Y) = \arccos\left(\frac{3}{6}\right) = \arccos\left(\frac{1}{2}\right) = \frac{\pi}{3}$$

Remark. *Often used in information retrieval to assess similarity between 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

Geometric interpretation of cosine distance



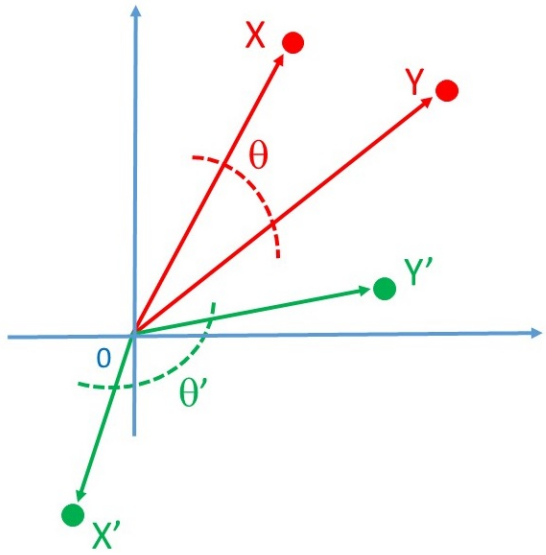
$$\cos\theta = \frac{x \cdot y}{\|x\| \cdot \|y\|}$$

$$\Rightarrow \underset{\text{cosine}}{d(x, y)} = \theta \quad \in [0, \pi]$$

Distance functions

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

- 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 $Y = (1,1,1,0,0)$

$d_{\text{Hamming}}(X, Y) = 2$ $\quad x_0 \neq y_0 \quad x_4 \neq y_4$

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

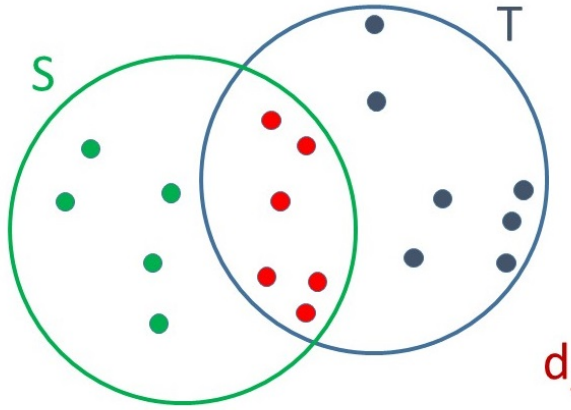
Distance functions

- 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 defined as

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

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:



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

Distance functions

Which distance should we use?

- Minkowski and Cosine 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; measurements of physical/chemical processes.
  - Cosine distance: it measures the ratios among features' values, rather than their absolute values. Example: relevance of a document since documents can have different lengths (objects (100,100,100) and (800,800,800) are equal, but they differ from (100,100, 50)).

Distance functions

- 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.

Distance functions

Observation: All of the above distances satisfy the four requirements for a metric space. (Proving the triangle inequality for the Jaccard distance requires tedious yet simple calculations. If interested, see: *M. Levandowsky, D. Winter. Distance between sets. Nature 234, 1971.*)

Exercise

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

Curse of dimensionality

Consider a metric space with points in $\Re^n$. The dimension n can be small (e.g., $n = 3$ for GPS data) but also very large (e.g., $n >> 100$ for text documents).

The curse of dimensionality refers to issues that arise when processing data in high-dimensional spaces.

- Quality: Distance functions may loose effectiveness in assessing similarity;
- Performance: The running times may have a linear or even exponential dependency on n.

Curse of dimensionality

Quality

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

Performance.

- *Clustering*: In many cases, the running time of a clustering algorithm can be expressed as $O(C_{dist} \cdot N_{dist})$, where N_{dist} is the number of distance computations and C_{dist} is the cost of a distance computation ($C_{dist} = \Theta(n)$ for an n-dimensional Minkowski/Cosine/Hamming/Jaccard distances).

- *Nearest Neighbor Search problem*: given a set S of m points in $\Re^n$, construct a data structure that, for a given query point $q \in \Re^n$, efficiently returns the closest point $x \in S$ to q. The problem requires time:

$$\Theta(\min\{n \cdot m, 2^n\}).$$

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

Optimization problem Π is defined by

* Set of instances I ($\equiv$ inputs)
* $\forall i \in I$ set S_i of FEASIBLE SOLUTIONS
* Objective function ϕ

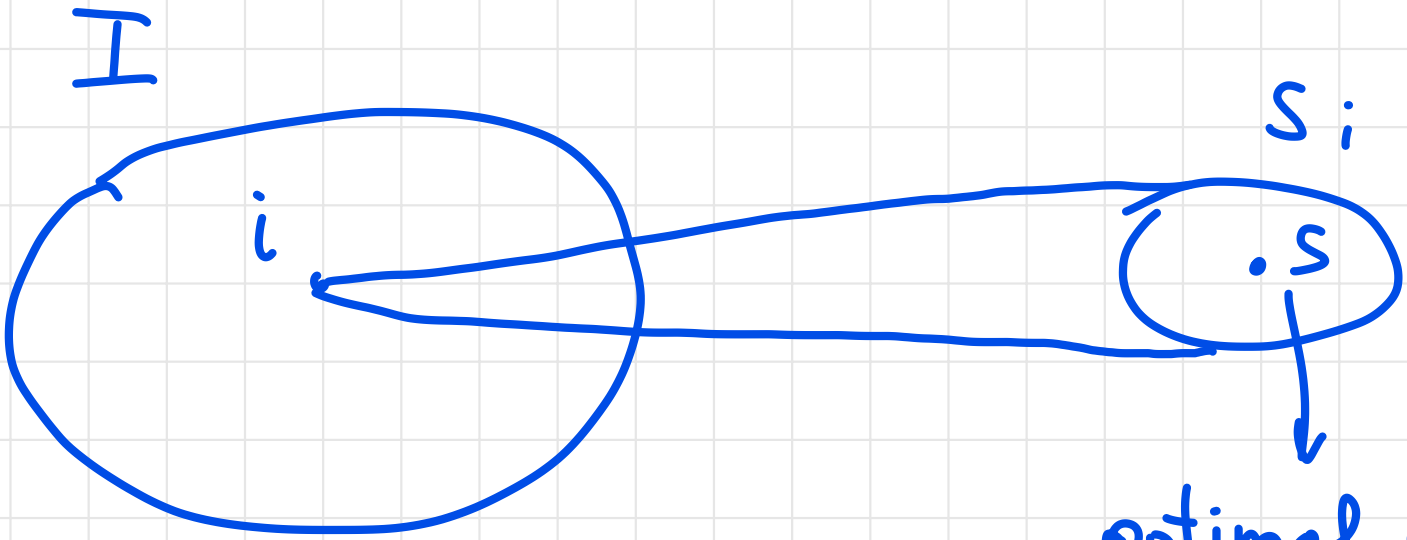
PROBLEM: Given $i \in I$ find $s \in S_i$ (OPTIMAL SOLUTION) such that

$$\phi(s) = \min \{\phi(s') : s' \in S_i\}$$ MINIMIZATION PROBLEM

or

$$\phi(s) = \max \{\phi(s') : s' \in S_i\}$$ MAXIMIZATION PROBLEM

Background: optimization problem



Obs. There might be more than one optimal solution

Background: optimization problem

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.

Background: approximation algorithm

Let Π be an optimization problem

a **c-approximation algorithm** A for Π ($c \geq 1$)

is an algorithm such that for each $i \in I$

returns a solution $A(i) \in S_i$ such that

$$\phi(A(i)) \leq c \cdot \min_{s' \in S_i} \phi(s')$$ MINIMIZATION PROBLEM

or

$$\phi(A(i)) \geq \frac{1}{c} \max_{s' \in S_i} \phi(s')$$ MAXIMIZATION PROBLEM

Background: approximation algorithm

Terminology

c = approximation factor or approximation ratio

$A(i)$ = c-approximate solution for i or a c-approximation of the optimal solution for i

Observation

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

* widens the spectrum of possible solutions that can be returned
* In the big data computing realm polynomial-time (even quadratic) algorithms may be impractical

⇒ resorting to approximations becomes even more crucial

Center-based clustering

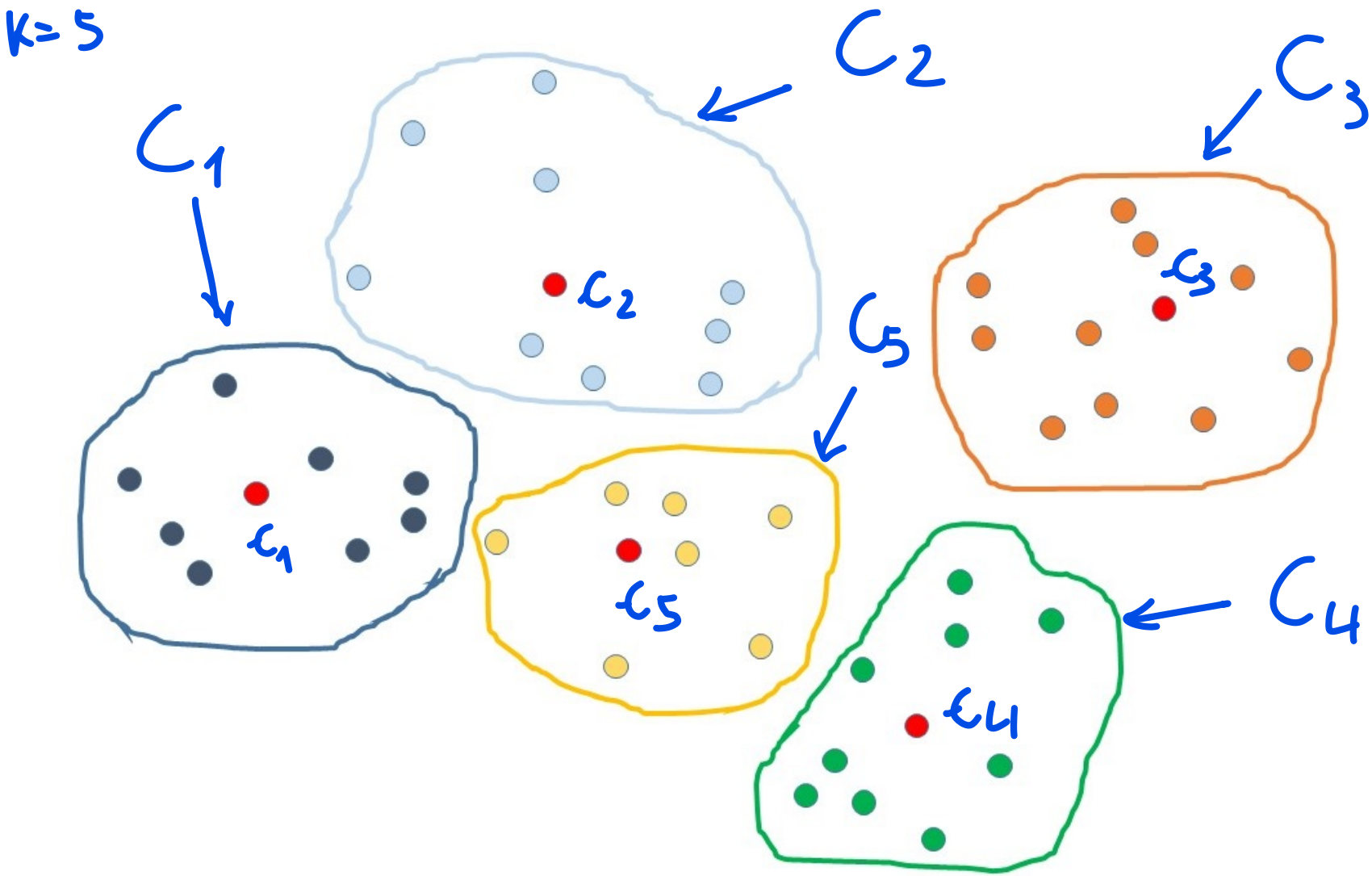
Center-based clustering

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., $P = C_1 \cup C_2 \cup \cdots \cup C_k$ → disjoint clusters
- $c_1, c_2, \ldots, c_k$ are suitably selected centers for the clusters, where $c_i \in C_i$ for every $1 \leq i \leq k$.

Remark: 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

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.

Center-based clustering: formal definitions

It is easy to see that for the k-center/means/median clustering problems, given a set S of k centers the best partitioning of the points around these centers, w.r.t. the objective function, is the one that assigns each point to the closest center (ties broken arbitrarily).
This allows us to focus on the center selection, as expressed by the formal definition given in the next slide.

Given P and a set $S \subseteq P$ of k centers, the k-clustering of P INDUCED by S is the one that assigns each $x \in P$ to the cluster of the closest center (break ties arbitrarily) and this k-clustering is the one the minimizes the objective function among all clusterings with centers $= S$

Center-based clustering: formal definitions

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

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

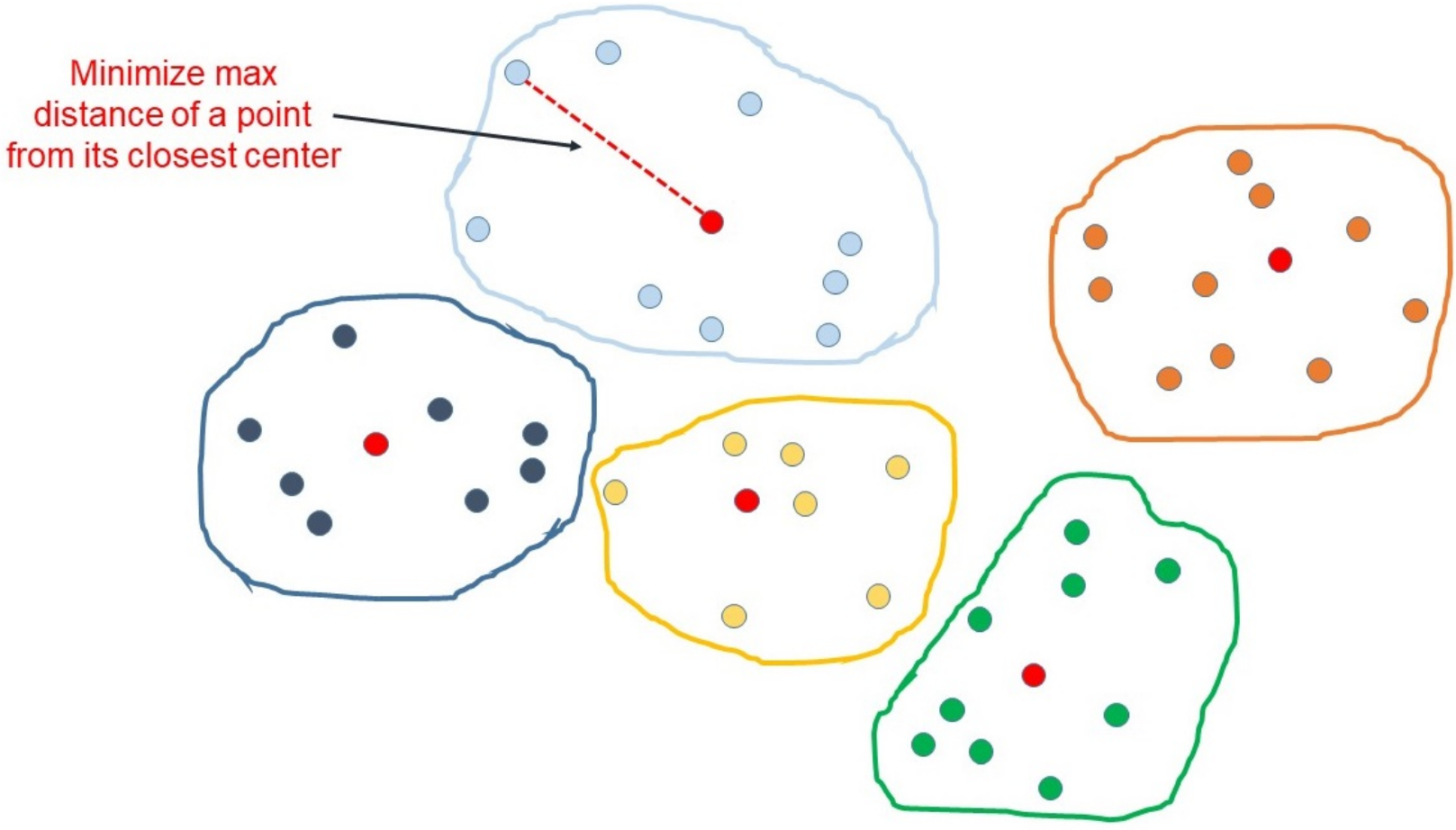
Definition

Let $k > 0$ and let (M, d) be a metric space. The k-center/k-means/k-median clustering problems are optimization problems that, given a finite pointset $P \subseteq M$, require to determine a subset $S \subseteq P$ of k *centers* which minimizes the following respective objective functions:

output $\equiv S$

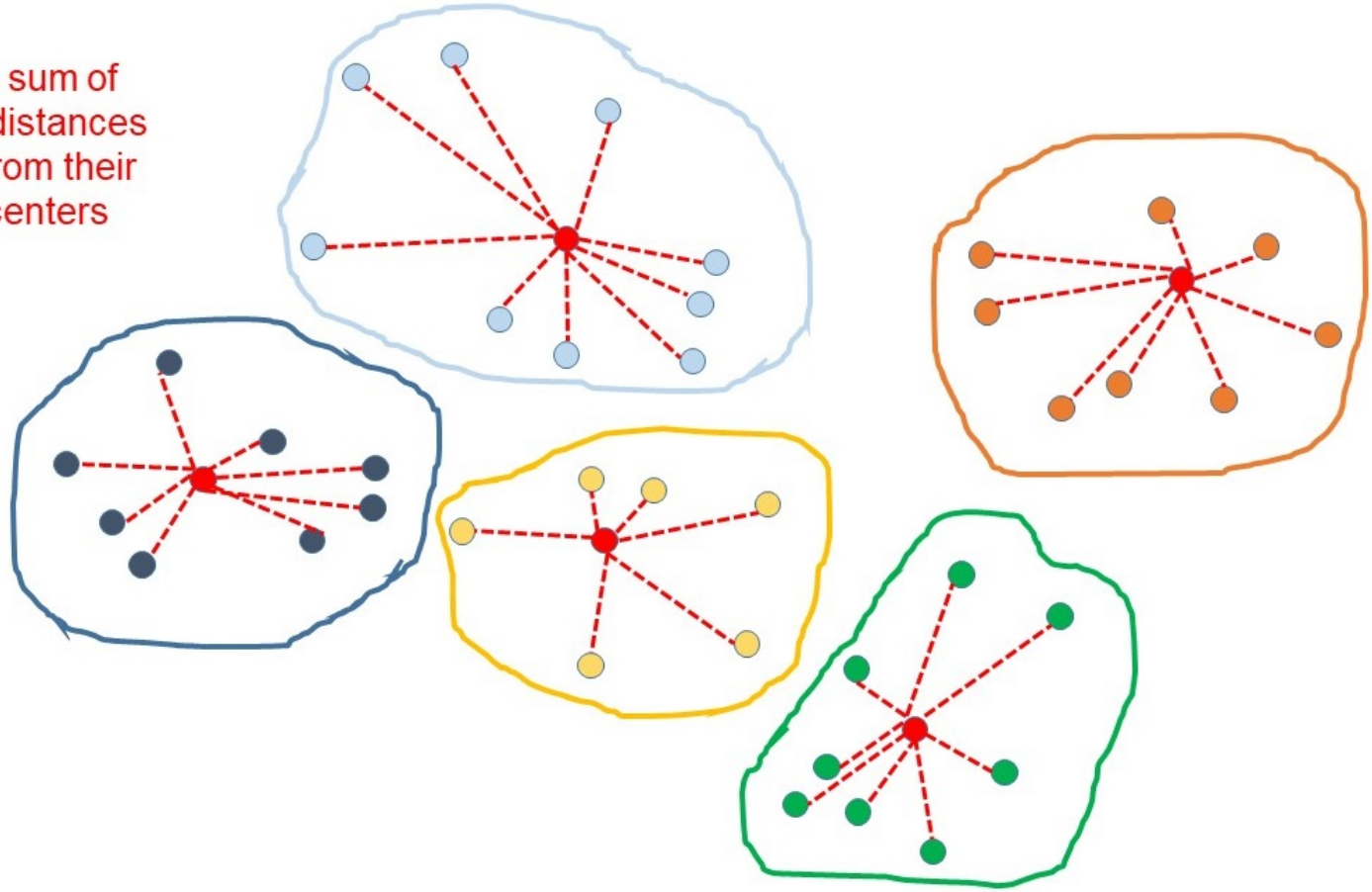
- $\Phi_{\text{kcenter}}(P, S) = \max_{x \in P} d(x, S)$ (k-center clustering)
- $\Phi_{\text{kmeans}}(P, S) = \sum_{x \in P} (d(x, S))^2$ (k-means clustering).
- $\Phi_{\text{kmedian}}(P, S) = \sum_{x \in P} (d(x, S))$ (k-median clustering).

Center-based clustering: k-center objective



Center-based clustering: k-means/median objectives

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



Center-based clustering: formal definitions

Input instance: pointset P and $k \in [0, |P|]$

Feasible solutions: $\{S \subseteq P : |S| = k\}$

Optimal solution: $S^* \subseteq P$ $|S^*| = k$

which minimizes:

$$\phi_{\text{kcenter/kmeans/kmedian}}(P, S^*)$$

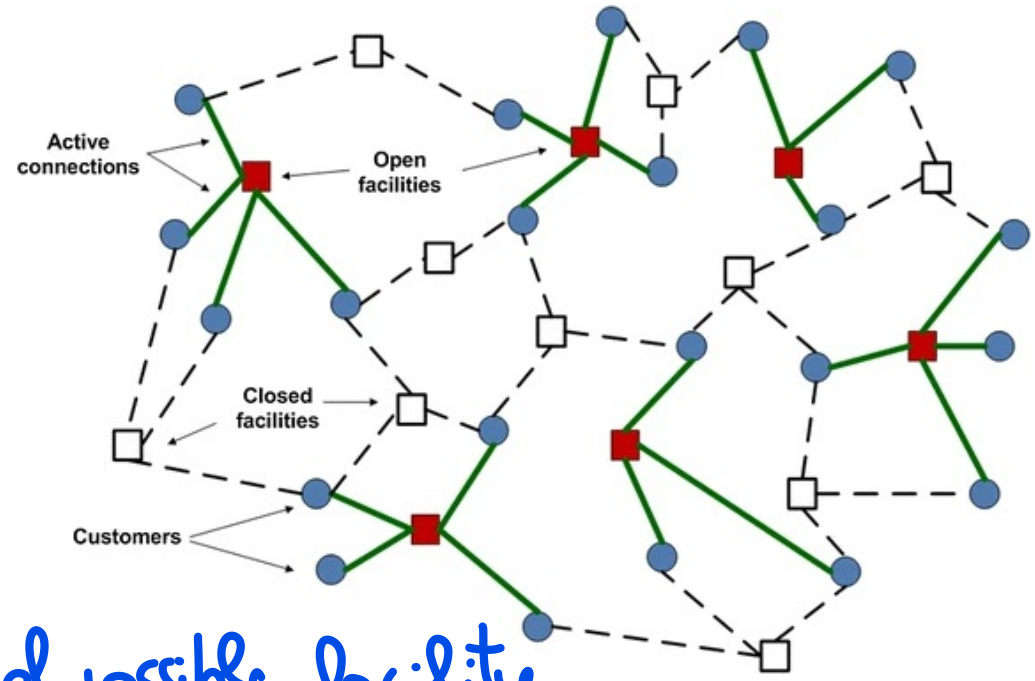
Recall: S^* may not be unique

Observations

- All aforementioned problems (k-center, k-means, k-median) are NP-hard $\Rightarrow$ 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!
- 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



□ ≡ location of possible facility

○ ≡ location of potential customer

k ≡ max # facilities to open minimizing max/avg distance between customers and facilities

Notations

For a pointset P from a metric space (M, d), define:

- $\Phi^{\text{opt}}_{\text{kcenter}}(P, k) = \min\{\Phi_{\text{kcenter}}(P, S) : \ S \subseteq P \text{ of size } k\}$.
- $\Phi^{\text{opt}}_{\text{kmeans}}(P, k) = \min\{\Phi_{\text{kmeans}}(P, S) : \ S \subseteq P \text{ of size } k\}$.
- $\Phi^{\text{opt}}_{\text{kmedian}}(P, k) = \min\{\Phi_{\text{kmedian}}(P, S) : \ S \subseteq P \text{ of size } k\}$.

Assignment primitive: Assign(P,S)

Assign $(P,\ S = \{c_1, c_2, \ldots, c_k\})$

for $i \leftarrow 1$ to k do $c_i.\text{cluster} \leftarrow i$

foreach $x \in P - S$ do {

$l \leftarrow \underset{c_i \in S}{\text{argmin}}\ d(x, c_i)$ // ties broken // arbitrarily

$x.\text{cluster} \leftarrow l$

}

Assignment primitive: Assign(P,S)

Exercise

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)