Clustering (Part 3)

OUTLINE

1. k-means clustering
   - Review of popular algorithms: LLoyd's, k-means++.
   - Coreset-based algorithm: MR-kmeans.
2. k-median clustering (sketch)
3. Clustering evaluation

k-means clustering

Problem definition

WE FOCUS ON: k-**means clustering for Euclidean spaces**

Given a pointset P of N points in $\Re^D$, under the standard L_2-distance function d, and an integer $k > 0$, determine a set $S \subset \Re^D$ of k centers which minimizes

$$\Phi_{\text{kmeans}}(P, S) = \sum_{x \in P} (d(x, S))^2.$$

Observations:

- k-means clustering aims at minimizing cluster variance
- Unlike the formulation presented in the previous set of slides, now the centers are not required to belong to the input P.

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 L_2-distance is

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

Definition

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

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

sum component-wise

where the sum is component-wise. Note that $c(P)$ does not necessarily belong to P.

Fact

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.

Lloyd's algorithm

In 2006 it was included in the top-10 most influential data mining algorithms

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

Output: Set $S \subset \Re^D$ of k centers with small $\Phi_{\text{kmeans}}(P, S)$.

$S = \{c_1, c_2, \ldots, c_k\} \leftarrow$ arbitrary set of k points in $\Re^D$;

$\Phi \leftarrow \Phi_{\text{kmeans}}(P, S)$;

while (true) **do**

  for $1 \leq i \leq k$ **do**

    $C_i \leftarrow$ points of P closest to c_i; // break ties arbitrarily

    $c_i \leftarrow$ centroid of C_i;

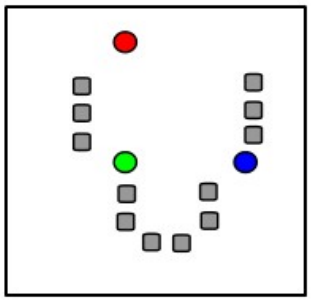
  Let $S = \{c_1, c_2, \ldots, c_k\}$;

  if $(\Phi_{\text{kmeans}}(P, S) < \Phi)$ **then** $\Phi \leftarrow \Phi_{\text{kmeans}}(P, S)$;

  else return S;

$$c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Example

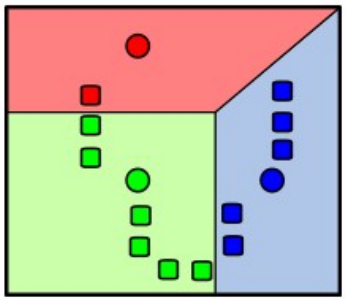


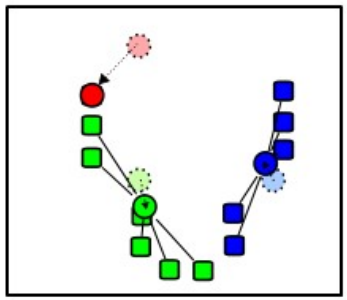
Initial centers

Partition

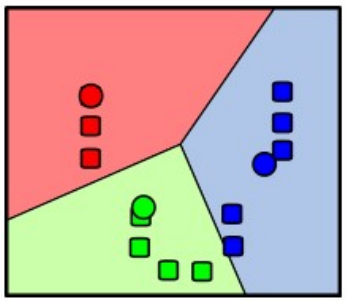
Calculation of centroids

Iteration 1:





Iteration 2:



. . .

Weaknesses of Lloyd's algorithm

1. The number of iterations can be exponential in the input size (in the worst case, $\geq 2^{\Omega(\sqrt{N})}$).

   COUNTERACTION: Fix number of iterations or stop iterations when the improvement of the objective function drops below a fixed threshold.

2. The algorithm may be trapped into a local optimum with objective function arbitrarily far from optimal value.

   COUNTERACTION: Make a careful selection of initial centers (which improves also convergence).

Rather than selecting the initial set of centers in an arbitrary or random fashion, we can use k-means++

k-means++ (2007)

Let P be a set of N points from $\Re^D$, and let $k > 1$ be an integer.

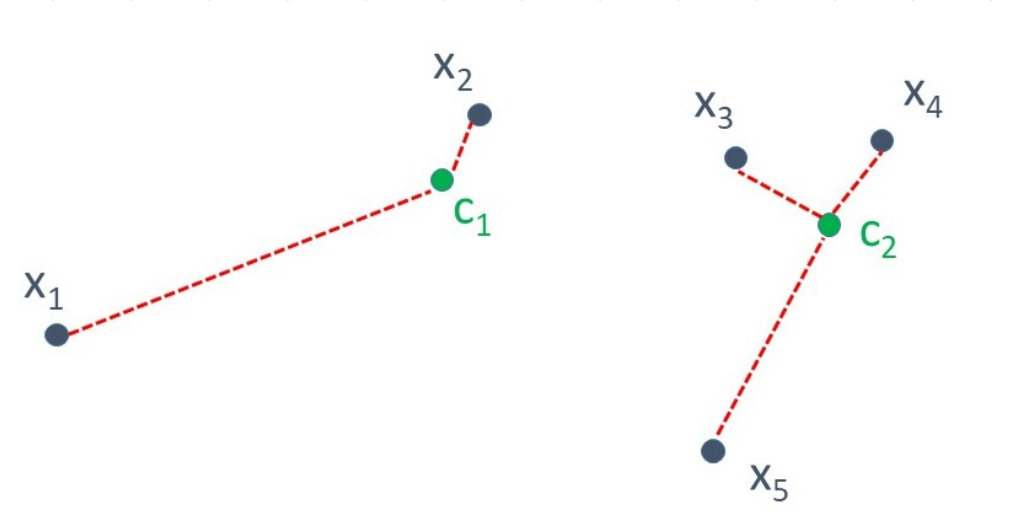
Algorithm k-means++ computes a (initial) set S of k centers for P using the following procedure (note that $S \subseteq P$):

$c_1 \leftarrow$ random point chosen from P with uniform probability;
$S \leftarrow \{c_1\}$;
for $2 \leq i \leq k$ **do**
    foreach $x \in P - S$ **do** $\pi(x) \leftarrow (d(x,S))^2 / \sum_{y \in P-S} (d(y,S))^2$;
    $c_i \leftarrow$ random point in $P - S$ according to distribution $\pi(\cdot)$;
    $S \leftarrow S \cup \{c_i\}$
return S

$\sum_{x \in P-S} \pi(x) = 1$

Obs. The set S returned by k-means++ is a subset of P

k-means++



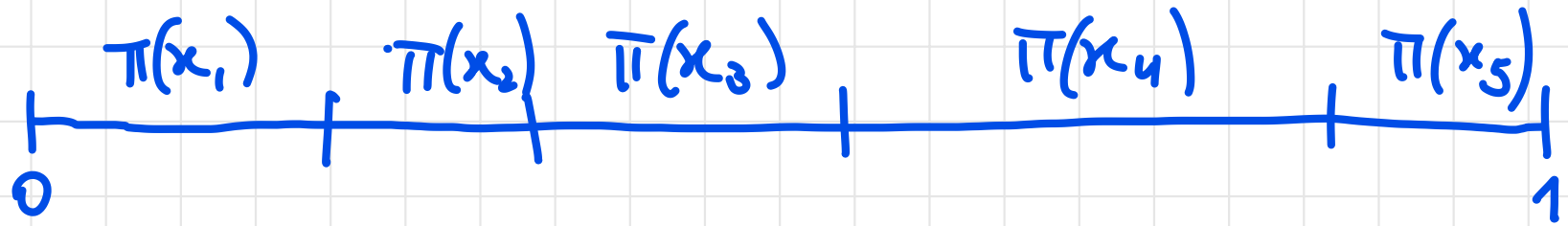
Selection of c_3

$$\pi(x_1) = \frac{(d(x_1, c_1))^2}{(d(x_1,c_1))^2 + (d(x_2,c_1))^2 + (d(x_3,c_2))^2 + (d(x_4,c_2))^2 + (d(x_5,c_2))^2}$$

k-means++

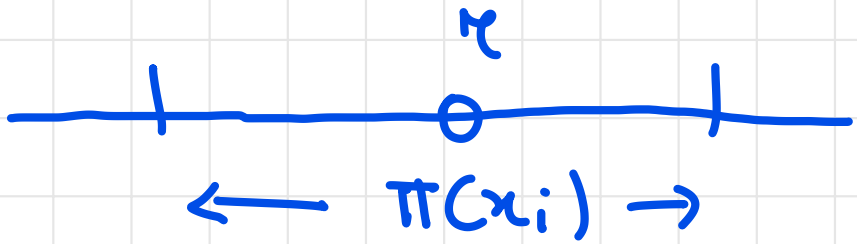
Suppose that we know $\pi(x) \quad \forall x \in P - S$

How do we determine the next center



Draw a random number $r \in [0, 1]$ with uniform probability

if



Then select x_i as next center

k-means++

Theorem (Arthur-Vassilvitskii'07)

Let S be the set of centers returned by k-means++ for a pointset P. We have

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

IMPORTANT REMARK: S and, consequently, $\Phi_{\text{kmeans}}(P,S)$, are random variables which depend on the random center selection!

Observations

What do we know so far about k-means clustering

* IN PRACTICE

  k-means++ followed by some iterations of Lloyd's algorithm yields a good-quality solution. This approach can be implemented efficiently, even in MapReduce, but for very large datasets and if many iterations of Lloyds are executed, then performance degrades

* IN THEORY: if we want constant approximations

- 50 approximation (somewhat efficiently)
- 6.36 approximation (inefficiently)

they cannot handle large datasets

* For general metric spaces, getting good solutions efficiently is even harder

k-means clustering for big data

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

Main alternatives:

- Distributed (e.g., MapReduce) implementation of Lloyd's algorithm and k-means++.

  (The next two exercises explore this alternative.)

- Coreset-based approach, similar in spirit to what was done for k-center. This approach lends itself to efficient distributed (e.g., MapReduce) as well as streaming implementations.

Lloyd's and k-means++ in MapReduce

Do the following two exercises assuming that the algorithms are applied to a set P of N points from $\Re^D$, with D constant, and that the target number of clusters is $k \in (1, \sqrt{N}]$.

Exercise

Show how to implement Lloyd's algorithm in MapReduce using $O(1)$ rounds per iteration, local space $M_L = O\left(\sqrt{N}\right)$ and aggregate space $M_A = O(N)$. (**Hint:** Make the set of centers and Φ global variables.)

Exercise

Show how to implement algorithm k-means++ in MapReduce using $O(k)$ rounds, local space $M_L = O\left(\sqrt{N}\right)$ and aggregate space $M_A = O(N)$.

Observations

* Spark (mllib library) provides implementations of Lloyd's and k-means++ algorithm for k-means++ Spark uses a parallel version called k-means||

* k-means++ yields also good BICRITERIA solutions

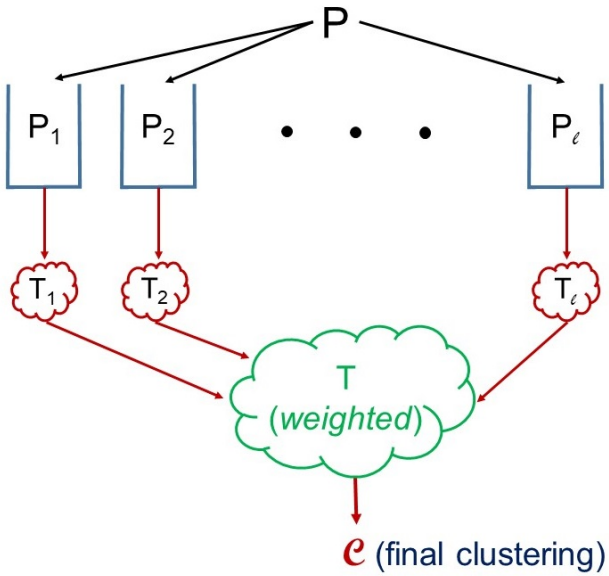
k-means++ $\Rightarrow$ S of $k' > k$ centers, say $k' = \beta k$ $\beta \geq 1$

$\beta \in O(1)$ is sufficient to ensure that

$$\phi_{kmeans}(P, S) = O(1)\, \phi^{opt}_{kmeans}(P, k)$$

Coreset-based approach for k-means

We use a (composable) coreset-based approach similar as for k-center.



- The local coresets T_i's and the final clustering $\mathcal{C}$ are computed using a sequential algorithm for k-means.
- Each point of T is given a weight equal to the number of points in P it represents, so to account for their cumulative contribution to the objective function

Need a weighted variant of k-means clustering.

Weighted k-means clustering

Weighted variant of the k-means clustering problem

Input:

- Set P of N points from $\Re^D$.
- Weight $w(x) > 0$ for every $x \in P$.
- Target number k of clusters.

Output: Set S of k centers in $\Re^D$ minimizing the following objective function.

$$\Phi^w_{\text{kmeans}}(P, S) = \sum_{x \in P} w(x) \cdot (d(x, S))^2.$$

weighted obj. function

Weighted k-means clustering: observations

- The unweighted k-means clustering problem is a special case of the weighted variant where all weights are equal to 1.
- For integer weights, most known algorithms can be adapted to solve the weighted variant. It is sufficient to regard each point x as representing $w(x)$ identical copies of itself. The same approximation guarantees are maintained.

Lloyd's algorithm with weights

Modifications to the algorithm

* Computation of a centroid c_i for C_i becomes

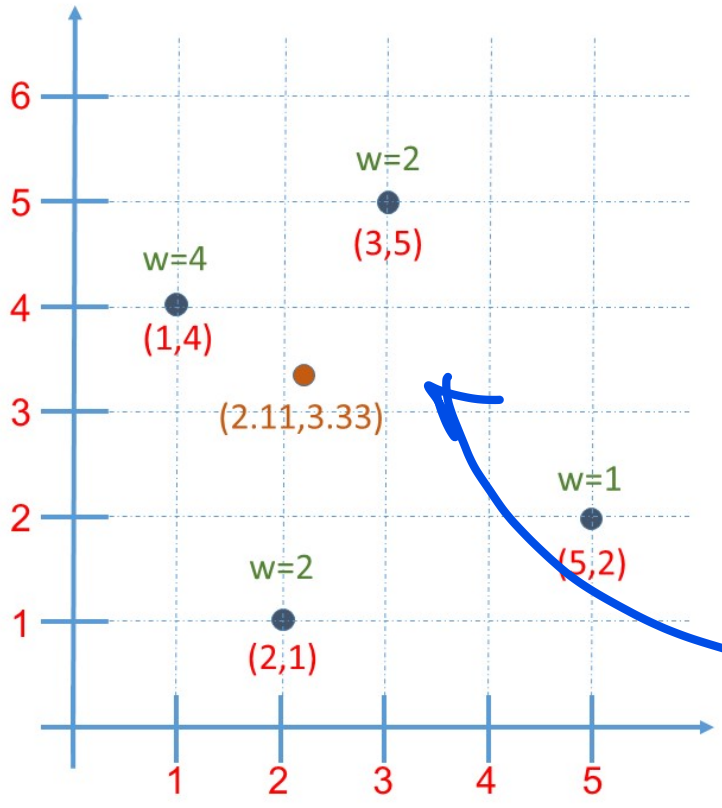
$$c_i = \frac{1}{\sum_{x \in C_i} w(x)} \sum_{x \in C_i} w(x) \cdot x \longrightarrow \text{vector}$$

$w(x)$: scalar

* Objective function

$$\phi_{kmeans}(P,S) \longrightarrow \phi^{w}_{kmeans}(P,S)$$

Example of weighted centroid



centroid (red)

$$\left(\frac{4+6+4+5}{9}, \frac{16+10+2+2}{9}\right) =$$

$$\left(\frac{19}{9}, \frac{30}{9}\right) \simeq (2.11, 3.33)$$

k-means++ algorithm with weights

Modifications to the algorithm

$S \equiv$ current set of centers

For each $x \in P - S$ the probability of being selected as next center is

$$\pi(x) = \frac{w(x)\left(d(x,S)\right)^2}{\sum_{y \in P-S} w(y)\cdot\left(d(y,S)\right)^2}$$

Coreset-based MapReduce algorithm for k-means

MR-kmeans($\mathcal{A}$): MapReduce algorithm for k-means (described in the next slide) which uses a sequential algorithm $\mathcal{A}$ for k-means, as an argument (functioal approach!).

In principle, one can instantiate $\mathcal{A}$ with any sequential algorithm, as long as the following characteristics are met:

- $\mathcal{A}$ solves the more general weighted variant.
- $\mathcal{A}$ requires space proportional to the input size (if larger space is needed, the analysis of M_L for the MapReduce algorithm must be amended).

MR-kmeans($\mathcal{A}$)

Input Set P of N points in $\Re^D$, integer $k > 1$, sequential k-means algorithm $\mathcal{A}$.

Output Set S of k centers in $\Re^D$ which is a good solution to the k-means probelm on P.

Round 1:

- Map Phase: Partition P arbitrarily in ℓ subsets of equal size $P_1, P_2, \ldots, P_\ell$.
- Reduce Phase: for every $i \in [1, \ell]$ separately, run $\mathcal{A}$ on P_i (with unit weights) to determine a set $T_i \subseteq P_i$ of k centers, and define
  - For each $x \in P_i$, the proxy $\tau(x)$ as x's closest center in T_i.
  - For each $y \in T_i$, the weight $w(y)$ as the number of points of P_i whose proxy is y.

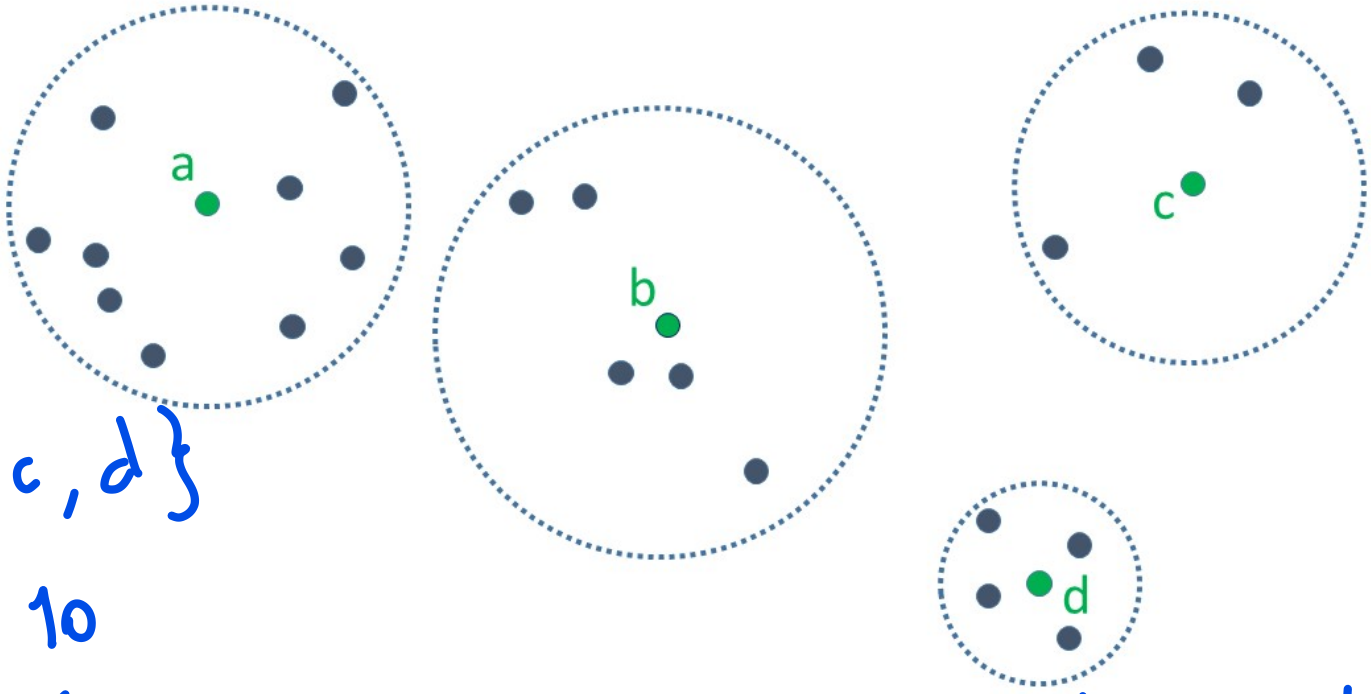
MR-kmeans($\mathcal{A}$)

Round 2:

- Map Phase: empty.
- Reduce Phase: gather the coreset $T = \cup_{i=1}^{\ell} T_i$ of $\ell \cdot k$ points, together with their weights, and run, using a single reducer, $\mathcal{A}$ on T (with the given weights) to determine a set $S = \{c_1, c_2, \ldots, c_k\}$ of k centers, which is then returned as output..

Example for Round 1

Set T_i (green points) computed in a partition P_i



$T_i = \{a, b, c, d\}$

$w(a) = 10$

$w(b) = 6$

$w(c) = 4$

$w(d) = 5$

if any of a, b, c, d is not in T_i we subtract 1 from its weight

Analysis of MR-kmeans($\mathcal{A}$)

Assume $k \leq \sqrt{N}$. By setting $\ell = \sqrt{N/k}$, it is easy to see that MR-kmeans($\mathcal{A}$) requires

Same analysis as for MR-FFT

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

The quality of the solution returned by the algorithm is stated in the following theorem, which is proved in the next few slides.

Theorem

Suppose that $\mathcal{A}$ is an α-approximation algorithm for weighted k-means clustering, for some $\alpha > 1$, and let S be the set of k centers returned by MR-kmeans($\mathcal{A}$) on input P. Then:

$$\Phi_{\text{kmeans}}(P, S) = O\left(\alpha^2\right) \Phi^{\text{opt}}_{\text{kmeans}}(P, k).$$

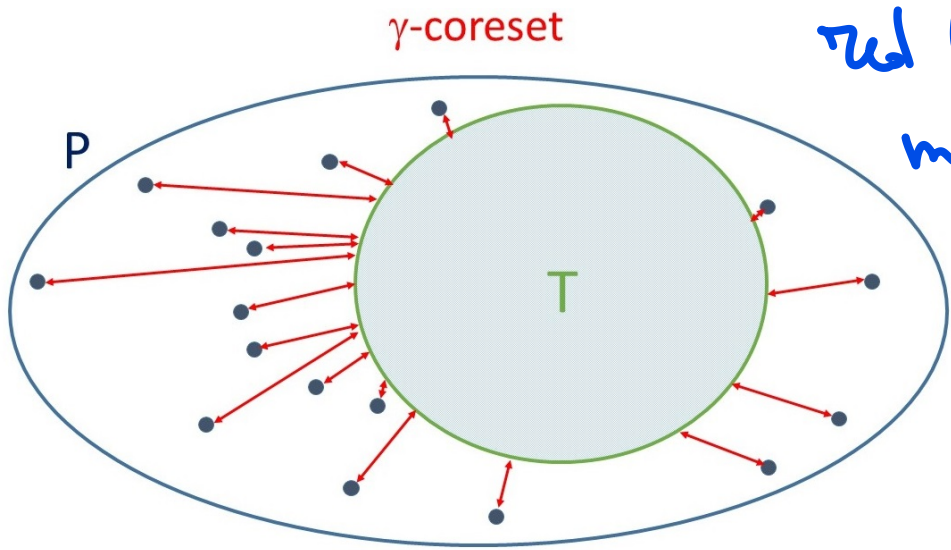
That is, MR-kmeans($\mathcal{A}$) is an $O\left(\alpha^2\right)$-approximation algorithm.

Analysis of MR-kmeans($\mathcal{A}$)

Definition

Given a pointset P, a coreset $T \subseteq P$ and a proxy function $\tau : P \to T$, we say that T is a γ-coreset for P, k and the k-means objective if

$$\sum_{p \in P} (d(p, \tau(p)))^2 \leq \gamma \cdot \Phi^{\text{opt}}_{\text{kmeans}}(P, k).$$



red lines represent map τ

The smaller γ and the better T represents P

Proof of Theorem

Proof idea

Claim 1 The coreset T computed by MR-kmeans(A) together with the map τ, is an α-coreset

Recall $T = \bigcup_{i=1}^{\ell} T_i$ T_i = α-approximation for kmeans on P_i

$\forall x \in P_i : \tau(x) \in T_i$ closest point of T_i to x

$\Rightarrow d(x, \tau(x)) = d(x, T_i)$

Proof of Theorem

* S_i^{OPT} ≡ optimal solution of k-means on P_i
* S^{OPT} ≡ optimal solution of k-means on P

$$\sum_{x \in P} (d(x, \tau(x)))^2 = \sum_{i=1}^{l} \sum_{x \in P_i} (d(x, \tau(x))^2$$

$$= \sum_{i=1}^{l} \sum_{x \in P_i} (d(x, T_i))^2$$

$$\leq \sum_{i=1}^{l} \alpha \sum_{x \in P_i} (d(x, S_i^{OPT}))^2$$

(since T_i is an α-approx. sol.)

$$= \alpha \sum_{i=1}^{l} \sum_{x \in P_i} (d(x, S_i^{OPT}))^2$$

Proof of Theorem

$$\leq \alpha \sum_{i=1}^{\ell} \sum_{x \in P_i} \left(d(x, S^{OPT})\right)^2$$

$$= \alpha \sum_{x \in P} \left(d(x, S^{OPT})\right)^2$$

$$= \alpha \, \phi_{\text{kmeans}}^{OPT}(P, k)$$

$\Rightarrow T$ with proxy function $r(\cdot)$ is an α-consistent for P, k, and k-means $\square$

Proof of Theorem

Claim 2 If T with proxy function $w(\cdot)$ is an α coreset for P, k and k-means, then the solution S computed by running A on T (weighted according to w) is such that

$$\phi_{kmeans}(P, S) = O(\alpha^2)\, \phi^{opt}_{kmeans}(P, k)$$

WE SKIP THIS PROOF

Proof of Theorem

Remark: It can be shown, in general, that if T is a γ-coreset, and we run an α-approximate algorithm A on T we get an $O((1+\gamma)\cdot\alpha)$ approximation.

The theorem follows immediately from the 2 claims

Proof of Theorem

Proof of Theorem

Observations

- Two different k-means sequential algorithms can be used in R1 and R2. For example, one could use k-means++ in R1, and k-means++ plus LLoyd's in R2.
- In practice, the algorithm is fast and accurate.
- If $k' > k$ centers are selected from each P_i in R1 (e.g., through k-means++), the quality of T, hence the quality of the final clustering, improves.

especially when P comes from a space of low dimensionality

k-median clustering (sketch)

k-median clustering

Given a pointset P from a metric space (M, d) and an integer $k > 0$, the k-median clustering problem requires to determine a set $S \subset P$ of k centers which minimizes:

$$\Phi_{\text{kmedian}}(P, S) = \sum_{x \in P} d(x, S).$$

- $O(1)$ sequential approximation algorithms for k-median are known but the most accurate ones are computationally intensive.
- The most popular sequential algorithm, PAM: Partitioning Around Medoids (a.k.a. k-medoids algorithm) is based on a local search optimization strategy.
- A coreset-based approach similar to MR-kmeans can be employed where the use of accurate sequential strategies is confined to a coreset much smaller than the input.

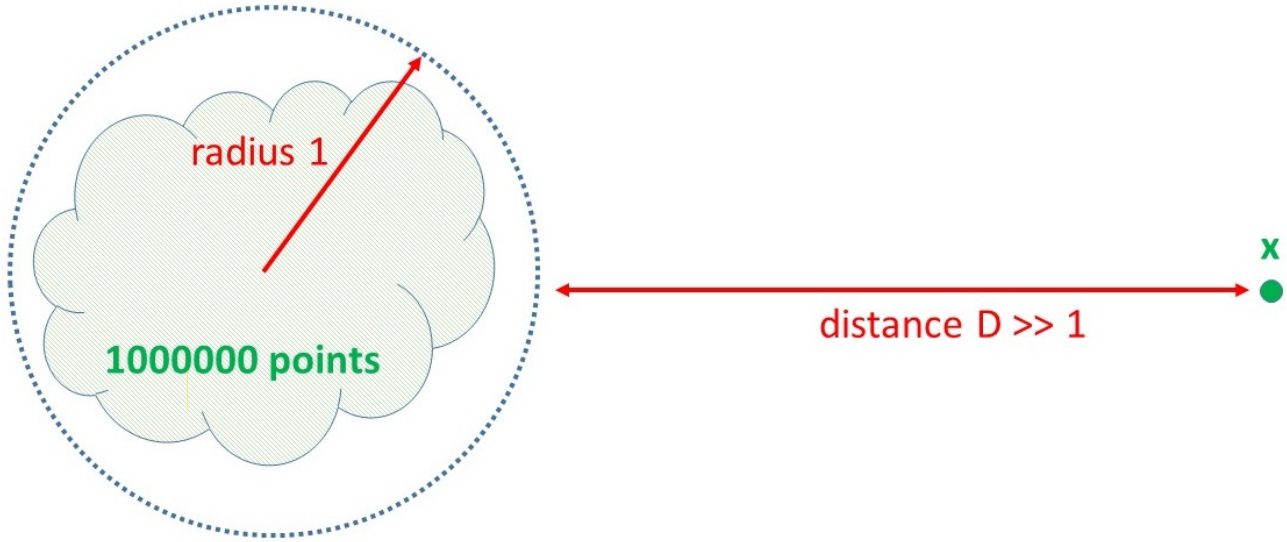
k-center vs k-means vs k-median

Some final considerations on the 3 center-based clustering problems.

- All 3 problems tend to return spherically-shaped clusters.
- They differ in the center selection strategy since, given the centers, the best clustering around them is the same for the 3 problems.
- Suitable coreset-based strategies can be employed for the 3 problems, featuring:
  - Ability to handle huge inputs.
  - Tunable performance-accuracy tradeoffs.
  - Very good behaviour in practice.
- The selection of the best k is often difficult, and this is a drawback common to the 3 problems.

k-center vs k-means vs k-median

The following example shows that different impact that outliers can have for the 3 problems.



For $k \geq 2$, the relevance of the outlier x in the 3 problems is different

- **k-center:** x must be always selected as center!
- **k-means:** x must be selected as center if $D \geq 1000$!
- **k-median:** x must be selected as center if $D \geq 1000000$!

Exercises

Exercise

Let S be the set of k centers returned by k-means++. Using Markov's inequality determine a value $\alpha > 1$ such that

$$\Pr\left(\Phi_{\text{kmeans}}(P, S) \leq \alpha \cdot \Phi^{\text{opt}}_{\text{kmeans}}(P, k)\right) \geq 1/2.$$

Recall that for a real-valued nonnegative random variable X with expectation $E[X]$ and a value $a > 0$, the Markov's inequality states that

$$\Pr(X \geq a) \leq \frac{E[X]}{a}.$$

Exercises

The next exercise shows how to make the approximation bound of algorithm k-means++ to hold with high probability rather than in expectation.

Exercise

From before, we know that if S is the set of centers computed by k-means++ for P, then there is a value $\alpha > 1$ such that

$$\Pr\left(\Phi_{\text{kmeans}}(P,S) \leq \alpha \cdot \Phi^{\text{opt}}_{\text{kmeans}}(P,k)\right) \geq 1/2.$$

Show that the above probability can be made $\geq 1 - 1/N$, if we run several *independent instances* of k-means++, and return the set of centers which yields the minimum value of the objective function, among the ones computed in the various runs.

Clustering evaluation

Unsupervised Evaluation

- Let P be a set of N points from a metric space (M, d)
- Let $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ be a partition of P into k clusters (i.e., a k-clustering ignoring cluster centers, if available).

Goal of unsupervised evaluation: assess the *quality* of $\mathcal{C}$ without reference to external information or ground truth.

Quality measures:

- Specific objective function, *if any*, which was used to select $\mathcal{C}$ among all feasible clusterings. For example k-center/k-means/k-median objective functions which, however, capture only intra-cluster similarity.
- Silhouette coefficient, which captures both intra-cluster similarity and inter-cluster dissimilarity.

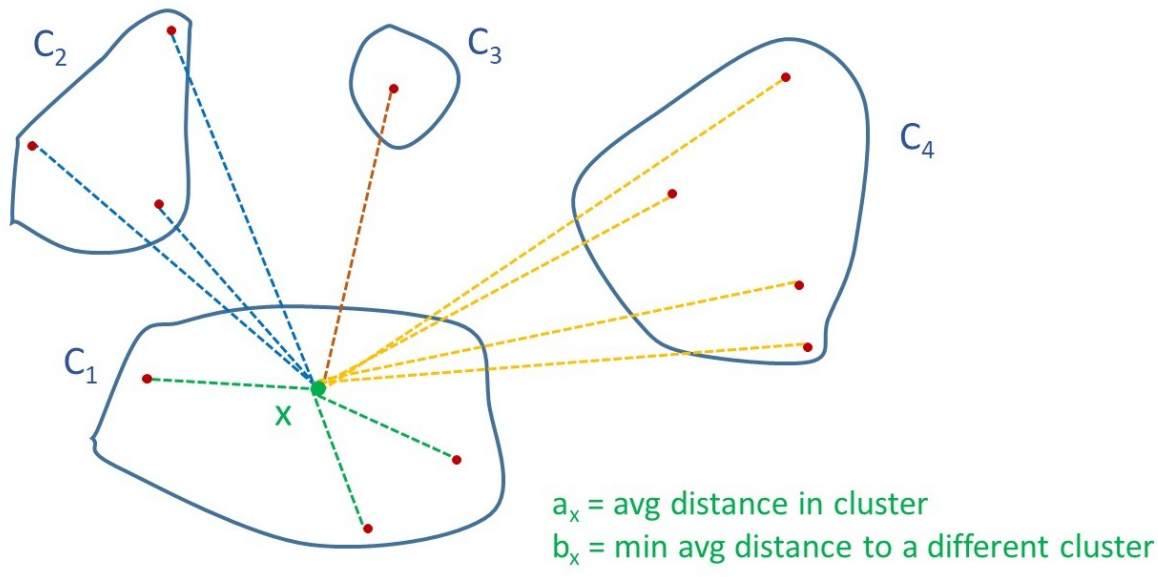
Observation: the silhouette coefficient is often used to select the most suitable number of clusters k, for a given clustering algorithm.

Unsupervised evaluation: silhouette

For a point $x \in P$ belonging to some cluster C_i let

$$\begin{aligned} a_x &= d_{\text{sum}}(x, C_i)/|C_i| \\ b_x &= \min_{j \neq i} d_{\text{sum}}(x, C_j)/|C_j|, \end{aligned}$$

where $d_{\text{sum}}(x, C)$ denotes the sum of the distances between x and the points of a cluster C (i.e., $d_{\text{sum}}(x, C) = \sum_{y \in C} d(x, y)$).



Unsupervised evaluation: silhouette

Definition: silhouette coefficient

Let $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ be a partition of P into k clusters. For any $x \in P$ the silhouette coefficient for x is

$$s_x = \frac{b_x - a_x}{\max\{a_x, b_x\}},$$

To measure the quality of $\mathcal{C}$ we define the average silhouette coefficient as

$$s_{\mathcal{C}} = \frac{1}{|P|} \sum_{x \in P} s_x.$$

Unsupervised evaluation: silhouette

Note that

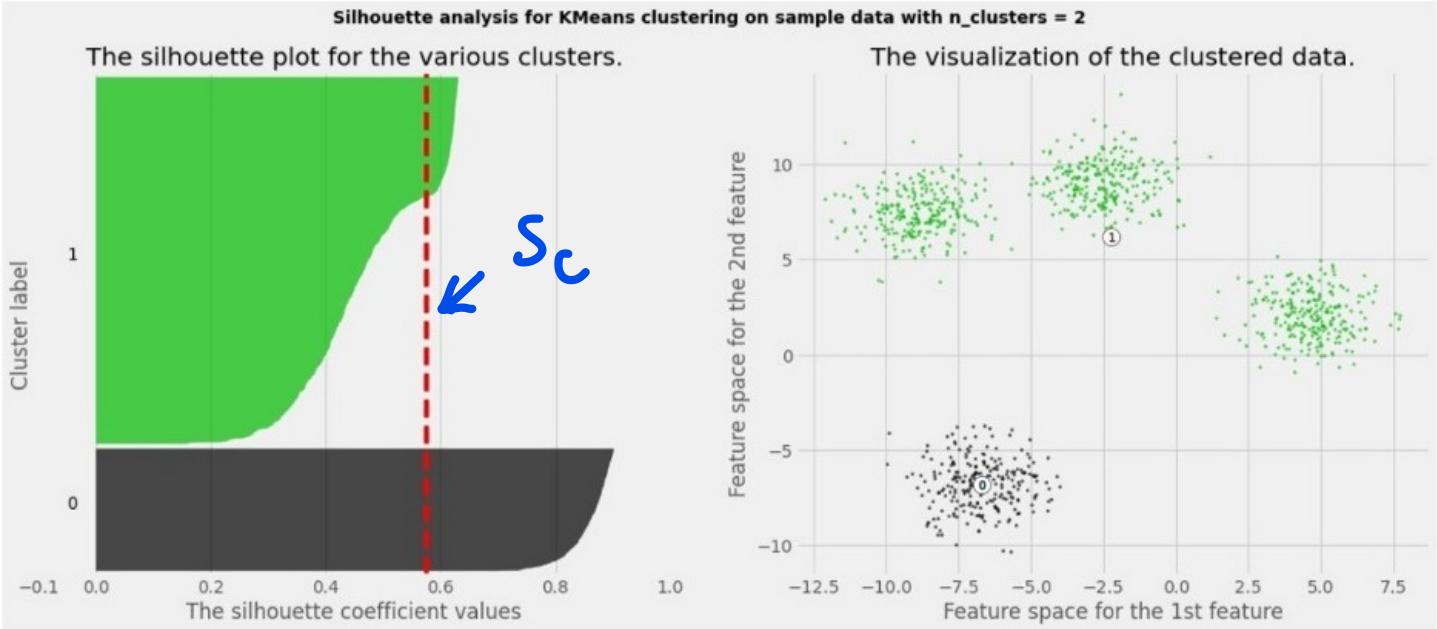
$$-\max\{a_x, b_x\} \leq b_x - a_x \leq \max\{a_x, b_x\}$$

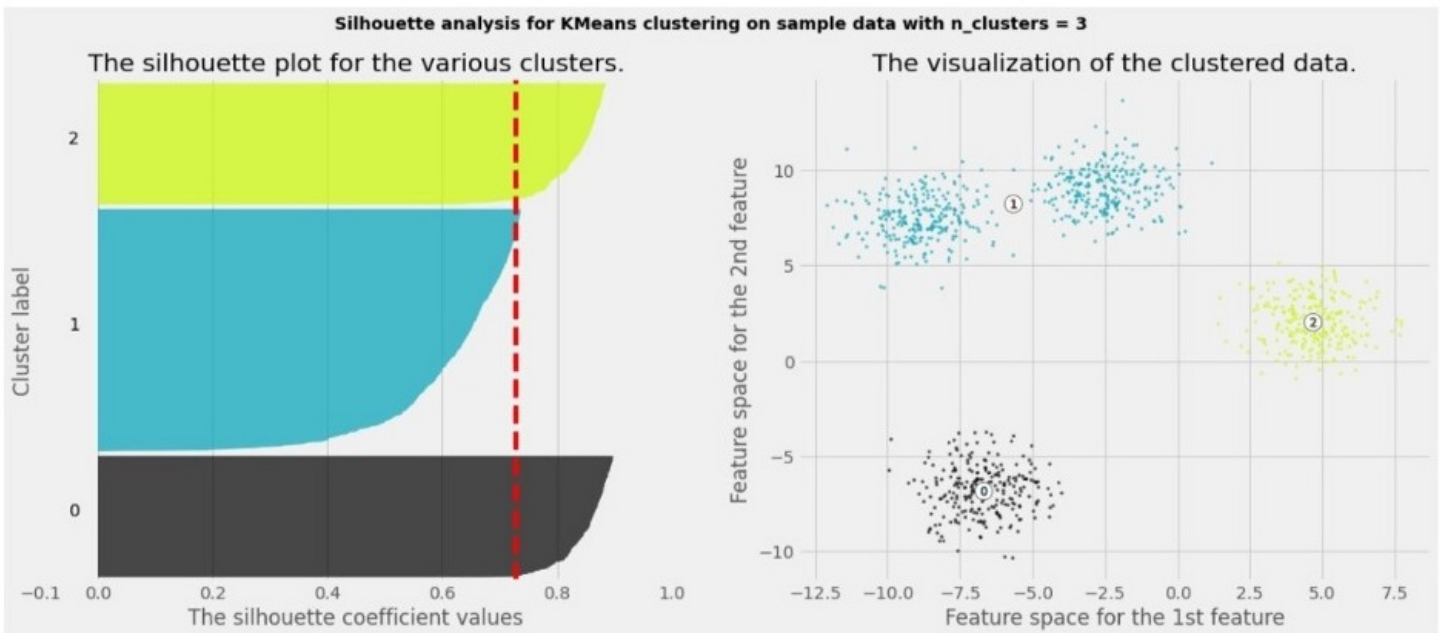
$$\Rightarrow \quad s_x \in [-1, 1] \quad \Rightarrow \quad s_C \in [-1, 1]$$

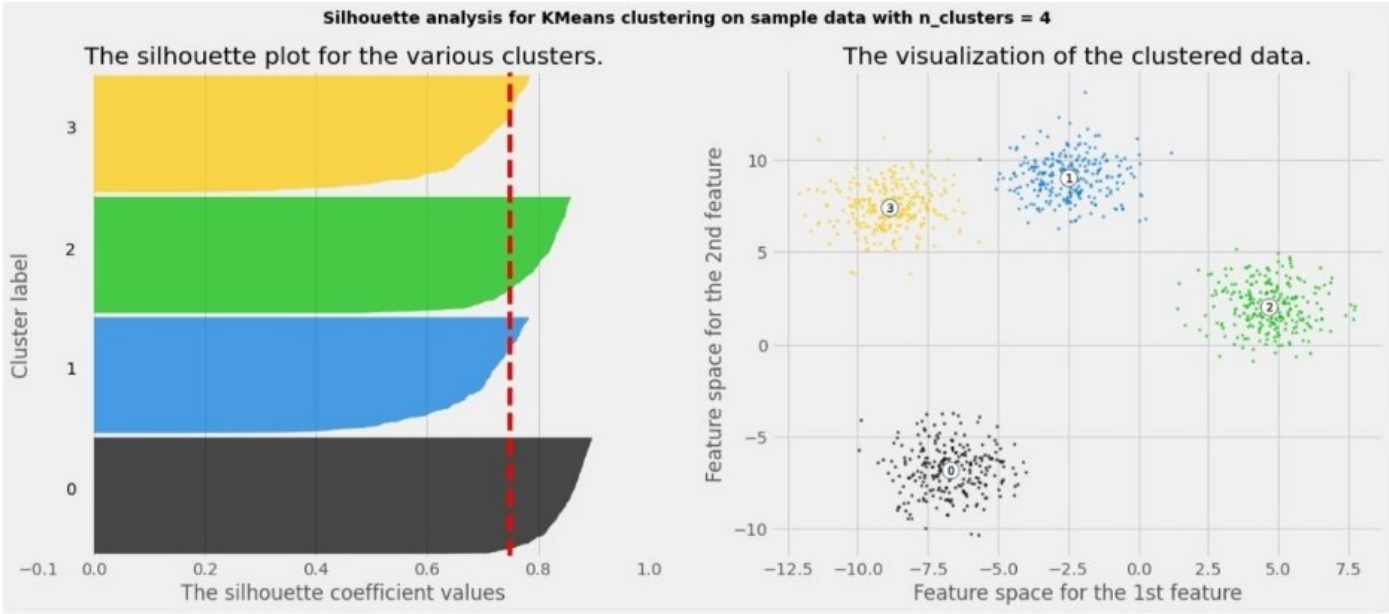
s_x close to $-1 \Rightarrow a_x \gg b_x \Rightarrow x$ is not clustered well

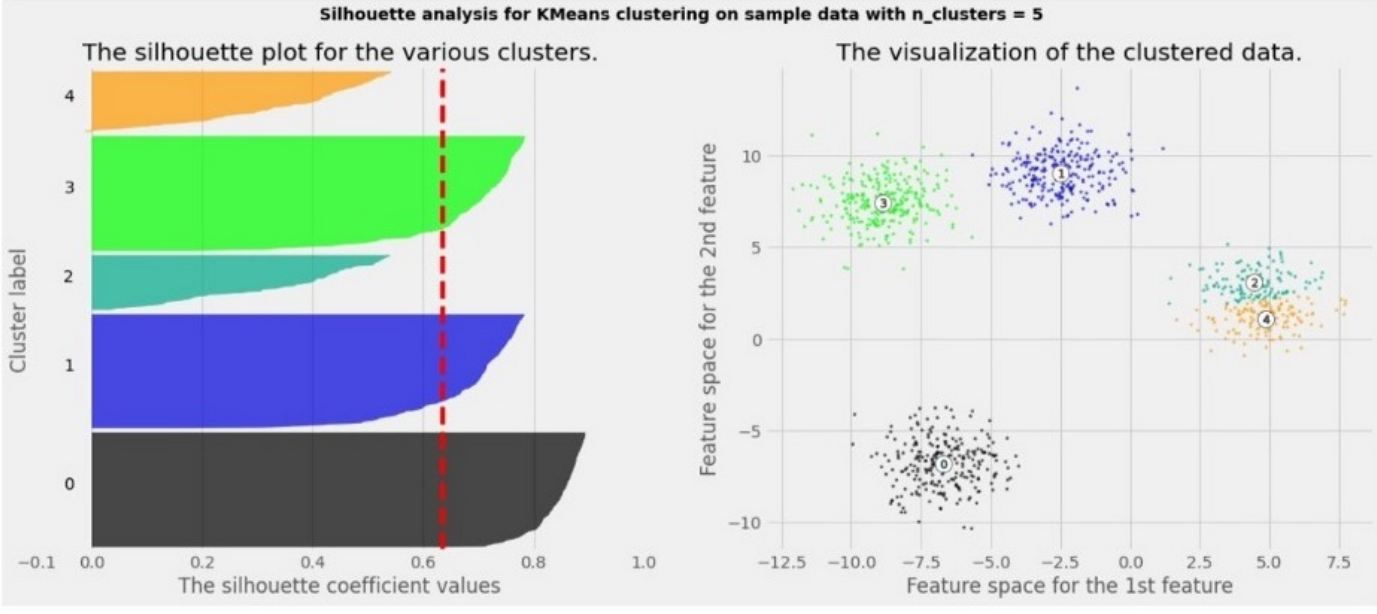
s_x close to $1 \Rightarrow a_x \ll b_x \Rightarrow x$ is clustered well

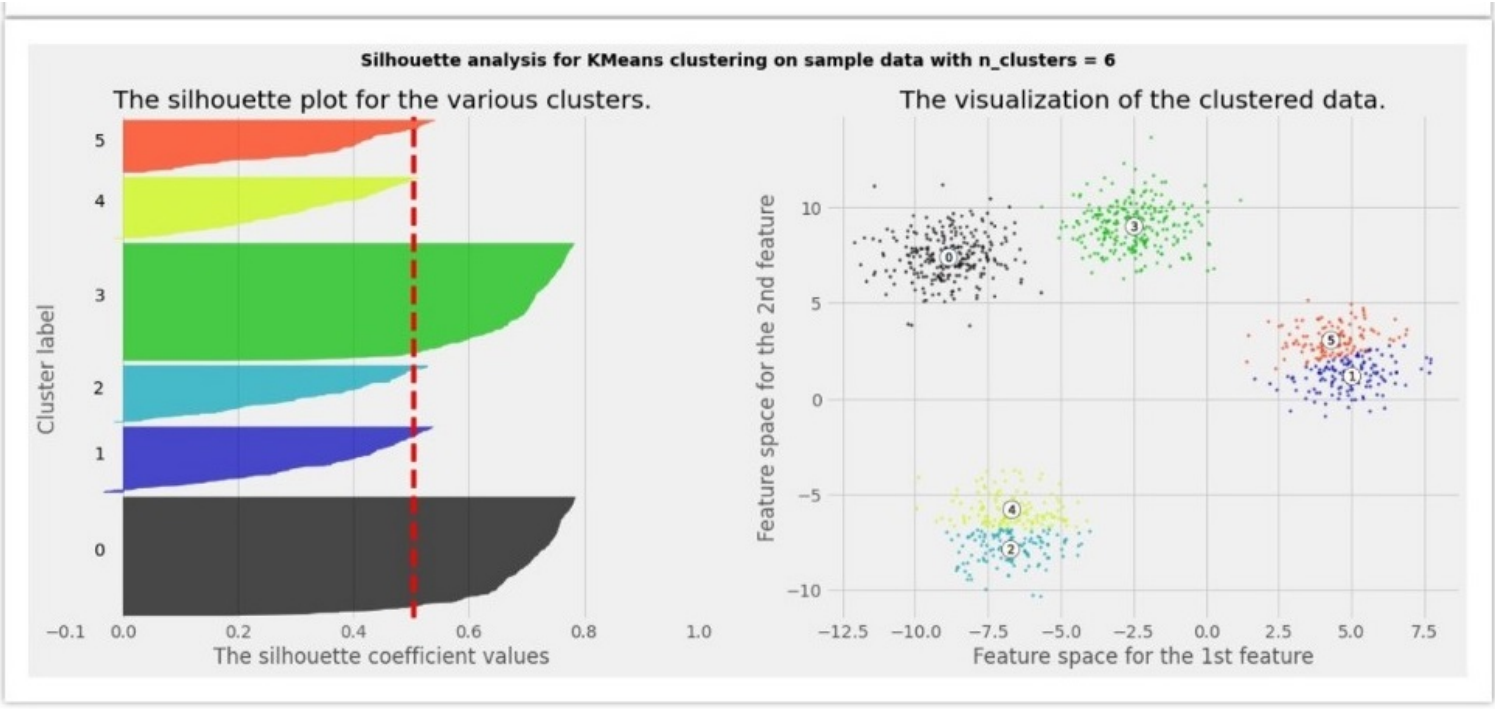
If s_C is close to 1 then $\mathcal{C}$ is a good clustering (i.e., most points are clustered well)


Silhouette analysis for KMeans clustering on sample data with n_clusters = 2
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature
Sc


Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 6
The silhouette plot for the various clusters.
Cluster label
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
10
5
0
−5
−10
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
Feature space for the 1st feature

Computational considerations

- The straightforward computation of $s_{\mathcal{C}}$ requires $\Theta\left(|P|^2\right)$ distance computations, unfeasible for very large $|P|$.
- Simplified computation for cosine and squared Euclidean distances ($\Theta\left(|P|k\right)$ distance computations). Methods are provided in the Spark library.
- There exhist heuristics to approximate $s_{\mathcal{C}}$, some with provable guarantees (from Padova!).

Approximating the sum of distances

Consider a point x and a set C. The following method can be used to approximate $d_{\text{sum}}(x, C)$. Let $n = |C|$.

1. Fix a target sample size $t \in [1, n]$.
2. Select a sample $S_t \subset C$, where each $y \in C$ is independently included in S_t with probability t/n (*Poisson sampling*)..
3. Compute the approximation

$$\tilde{d}_{\text{sum}}(x, C, t) = \frac{n}{t} \sum_{y \in S_t} d(x, y)$$

Remark. *$\tilde{d}_{\text{sum}}(x, C, t)$ is a random variable, whose value depends on the random sample S_t.*

Proposition

$\tilde{d}_{\text{sum}}(x, C, t)$ is an unbiased estimator of $d_{\text{sum}}(x, C)$, i.e.,

$$E[\tilde{d}_{\text{sum}}(x, C, t)] = d_{\text{sum}}(x, C).$$

Proof of Proposition

For each $y \in C$ define a random variable $V_x(y)$

$$V_x(y) = \begin{cases} d(x,y) & \text{if } y \in S_t \text{ with prob. } t/n \\ 0 & \text{if } y \notin S_t \text{ with prob } 1 - t/n \end{cases}$$

$$\tilde{d}_{sum}(x, C, t) = \frac{n}{t} \sum_{y \in S_t} d(x,y)$$

$$= \frac{n}{t} \sum_{y \in C} V_x(y)$$

Proof of Proposition

$$E[\tilde{d}_{sum}(s,C,t)] = E\left[\frac{n}{t}\sum_{y\in C} V_x(y)\right]$$

$$= \frac{n}{t}\sum_{y\in C} E[V_x(y)]$$

$$E[V_x(y)] = \frac{t}{n}\, d(x,y)$$

$$\Rightarrow$$

$$E[\tilde{d}_{sum}(x,C,t)] = \frac{n}{t}\cdot\sum_{y\in C}\frac{t}{n}\, d(x,y)$$

$$= \sum_{y\in C} d(x,y) = d_{sum}(x,C) \qquad \square$$

Approximating the sum of distances

Observations:

- The absolute error of the estimate $|\tilde{d}_{\text{sum}}(x, C, t) - d_{\text{sum}}(x, C)|$ can be upper bounded analytically as a function of t (see [EW04]).
- While the theoretical bound is somewhat weak, in practice the error is rather low even with small t.

Exercise

Let P be a pointset of N points from a metric space (M, d) and let $C \subseteq P$ be a subset of size n. Show how to compute $\tilde{d}_{\text{sum}}(x, C, t)$ efficiently in MapReduce, for all $x \in P$. Assume that t and n are know and

- $t \in O(\log N)$ but n can be large (up to N);
- each point $x \in P$ is represented by a pair $(ID_x, (x, f_x))$, where ID_x is a distinct integer in $[0, N-1]$ and f_x is a binary flag with value 1 if $x \in C$, and 0 otherwise.

Approximating the Silhouette

Let $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ be a partition of P into k clusters.

Fix a sample size t and define $t_i = \min\{t, |C_i|\}$, for $1 \leq i \leq k$.

For $1 \leq i \leq k$ and $x \in C_i$, define

$$\tilde{a}_x = \tilde{d}_{\text{sum}}(x, C_i, t_i)/|C_i| \quad \text{and} \quad \tilde{b}_x = \min_{j \neq i} \tilde{d}_{\text{sum}}(x, C_j, t_j)/|C_j|,$$

The approximate silhouette coefficient of x is

$$\tilde{s}_x = \frac{\tilde{b}_x - \tilde{a}_x}{\max\{\tilde{a}_x, \tilde{b}_x\}},$$

The approximate average silhouette coefficient of $\mathcal{C}$ is

$$\tilde{s}_{\mathcal{C}} = (1/|P|) \sum_{x \in P} \tilde{s}_x.$$

IMPORTANT: $\forall$ cluster C_i compute only one sample S_i using parameter t_i

Exercise

Determine the expected number of distance computations needed to obtain $\tilde{s}_{\mathcal{C}}$.

$\forall x \in P$ and $\forall i \quad 1 \leq i \leq k$

We must compute all distances between x and the sample S_i of C_i

$\Rightarrow$ The total number of distance computations is

$$N \cdot \sum_{i=1}^{k} |S_i|$$

$$E\left[N \cdot \sum_{i=1}^{k} |S_i|\right] = N \cdot \sum_{i=1}^{k} E[|S_i|]$$

$$= N \cdot \sum_{i=1}^{k} \frac{t_i}{|C_i|} \cdot |C_i|$$

$$= N \sum_{i=1}^{k} \min\{t, |C_i|\}$$

$$\leq N \cdot k \cdot t = \sigma(N^2)$$

when k, t are sufficiently small

Other evaluation metrics

- Supervised evaluation: assessment of the quality of a clustering with reference to external information. For example, Entropy can be used to measure the coherence of a clustering with respect to a ground truth (e.g., class labes).
- Clustering tendency: assessment whether the data contain meaningful clusters, namely clusters that are unlikely to occur in random data. The Hopkins statistic can be used to this purpose.

Clustering evaluation is well described in [TSK06]

Summary of Parts 1, 2 and 3

- Metric space and prominent distance functions.
- Combinatorial optimization problem and approximation algorithm.
- k-center clustering:
  - Definition of the problem.
  - Farthest-First Traversal algorithm.
  - (Composable) coreset technique.
  - MR-Farthest-First Traversal.
- k-means clustering:
  - Definition of the problem.
  - Review of Lloyd's and k-means++ algorithms.
  - Weighted variant of the problem.
  - MR-kmeans
- k-median clustering (sketch)

Summary of Parts 1, 2 and 3

- Unsupervised clustering evaluation:
  - Silhouette coefficient.
  - Approximating the sum of distances.
  - Approximating the Silhouette.
  - Other evaluation metrics (sketch).

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

BHK18 A. Blum, J. Hopcroft, and R. Kannan. Foundations of Data Science. Manuscript, June 2018. Chapter 7

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

C+19 V. Cohen-Addad et al. Tight FPT Approximations for k-Median and k-Means. Proc. of ICALP 2019: 42:1-42:14.

Wei16 D. Wei A Constant-Factor Bi-Criteria Approximation Guarantee for k-means++. Proc. of NIPS 2016: 604-612.

TSK06 P.N.Tan, M.Steinbach, V.Kumar. Introduction to Data Mining. Addison Wesley, 2006. Chapter 8.

EW04 D. Eppstein, J. Wang: Fast Approximation of Centrality. J. Graph Algorithms Appl. 8: 39-45 (2004).