Coreset Technique

(Part 2)

OUTLINE

1. Coresets and Composable Coresets (Part 1)
2. Case study: k-center-clustering (Part 1)
3. Coresets of other clustering problems
   - k-means/median
   - Clustering evaluation

k-means/median

Definition of the problems

Let us review the two problems

Given a pointset P of N points from a metric space (M, d), determine a set $S \subset P$ of k centers which minimizes

$$\Phi_{\text{kmeans}}(P, S) = \sum_{x \in P} (d(x, S))^2 \qquad \text{(k-means)}$$

$$\Phi_{\text{kmedian}}(P, S) = \sum_{x \in P} (d(x, S)) \qquad \text{(k-median)}$$

Observation: depending on the application and/or the algorithm used, the requirement that S be a subset of P may be lifted, and the centers can be allowed to be aribtrary points of M.

Current popular algorithms for k-means/median

Lloyd's algorithm: (a.k.a. "k-means algorithm")

- APPLICABILITY. Used only for k-means when $M = \mathbb{R}^D$, $d(\cdot, \cdot)$ is the standard Euclidean distance (L_2-distance), and centers can be selected outside P.
- ACCURACY. If initial centers are selected well (e.g., through k-means++) it usually provides good solutions, but, if not, it may be trapped into local optima.
- EFFICIENCY. Efficient implementations are provided by most common software packages, but a limit on the number of iterations is needed in case of very slow convergence.
- SUITABILITY FOR MASSIVE INPUTS. Yes if data can be processed by a distributed platform and only few iterations are executed. However, it is not suitable to process data streams,

Current popular algorithms for k-means/median

k-means++ algorithm: (by Arthur & S. Vassilvitskii, 2007).

$c_1 \leftarrow$ 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

Observation: k-means++ is a randomized algorithm and it is known that in expectation, the returned solution is an α-approximation, with $\alpha = \Theta(\ln k)$.

Current popular algorithms for k-means/median

k-means++ algorithm:

- APPLICABILITY. It can be used for both k-means and k-median, and enforces $S \subset P$.
- ACCURACY. It provides decent solutions, but it is often useful to refine them to get better accuracy.
- EFFICIENCY. Very easy and efficient implementation.
- SUITABILITY FOR MASSIVE INPUTS. Yes if data can be processed by a distributed platform and k is small. However, it is not suitable to process data streams.

Current popular algorithms for k-means/median

Partitioning Around Medoids (PAM) algorithm (a.k.a. k-medoids).

Devised by Kaufman and Rousseeuw in 1987, it is based on a local search strategy which starts from an arbitrary solution S and progressively improves it by performing the best swap between a point in S and a point in $P - S$, until no improving swap exists.

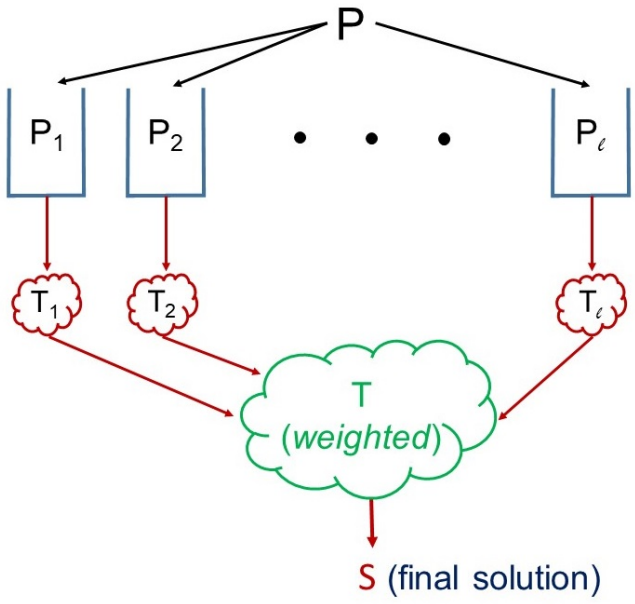
- APPLICABILITY. Mainly used for k-median, and enforces $S \subset P$.
- ACCURACY. It provides good solutions.
- EFFICIENCY. Very slow since each iterations requires checking about $N \cdot k$ possible swaps and the convergence can be very slow.
- SUITABILITY FOR MASSIVE INPUTS. Not at all!

Consequences of the above scenario

Clustering of massive data remains very challenging when

Coreset-based approach for k-means/median

The composable coreset technique can be employed to devise k-means/median algorithms which are able: (i) to handle massive data (in a distributed setting); and (ii) to provide good accuracy.

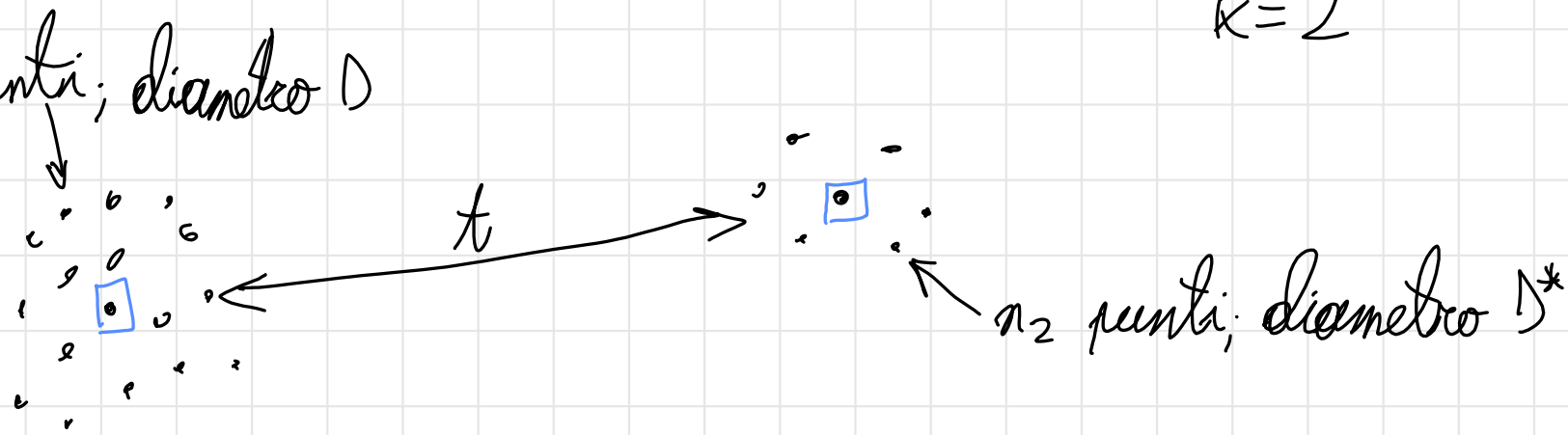


- Each point $x \in T$ is given a weight $w(x)$ = the number of points in P for which x is the closest representative in T.
- The local coresets T_i's are computed using a sequential algorithm for k-means/median.
- The final solution S is also computed using a sequential algorithm for k-means/median, adapted to handle weights.

Why are weights needed?

palle hanno diametro $D \ll t$

$K = 2$



- caso 1: centri in palle diverse $\Rightarrow$
  $\Rightarrow$ costo k-means $\sim n_1 D^2 + n_2 D^{*2} = c_1$
- caso 2: centri in palle con + punti ($n_1 >> n_2$)
  $\Rightarrow$ costo k-means $\sim n_1 \left(\frac{D}{2}\right)^2 + n_2 t^2 = c_2$

$\Downarrow$

2 cluster: divido distanza media

$$C_1 - C_2 = n_1 D^2 + n_2 D^{*2} - \left(n_1 (D/2)^2 + n_2 t^2\right) =$$
$$= n_1 \frac{3}{4}\left(\frac{D}{2}\right)^2 + n_2 (D^{*2} - t^2) \sim n_1 \frac{3}{4}\left(\frac{D}{2}\right)^2 - n_2 t^2 > 0 \Rightarrow n_2 > n_1 \frac{3}{16}\left(\frac{D}{t}\right)^2$$

in questo caso, clustering caso 1 migliore

Se $n_2 < n_1 \frac{3D}{4t^2}$, caso 2 migliore

Per k-median, rimuovo tutti quadrati $\left(n_2 < n_1 \frac{D}{2t}\right)$

⇓

diversa sensibilità a outlier:
in k-means servono meno punti per avere soluzione diversa $\left(\frac{D}{2t} < 1\right)$

From now on we focus on k-means

What we will say applies as well to k-median

(with very minor adaptations)

Weighted k-means clustering

Weighted variant of the k-means clustering problem

Input:

- Set P of N points from $\mathbb{R}^D$.
- Integer weight $w(x) > 0$ for every $x \in P$.
- Target number k of clusters.

Output: Set S of k centers in $\mathbb{R}^D$ minimizing

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

⇒ molto simile a og

Observations:

- This formulation allows centers outside P.
- If $w(x) = 1$ for every $x \in P$ we have the standard k-means clustering problem.

How to adapt known algorithms to handle weights

Lloyd: cambiare formula centroide e usare costi pesati in condizione loop

$$c_w = \frac{\sum_{x \in S} w(x) \cdot x}{\sum_{x \in S} w(x)}$$

K-means++: cambiare distribuzione in ciclo e selezione c_1

$$\pi(x) = \frac{w(x)\,(d(x,S))^2}{\sum_{y \in P \setminus S} w(y)\,(d(y,S))^2} \qquad \text{prob.} \ \frac{w(x)}{\sum_{y \in P} w(y)}$$

In pratica, usiamo k-means++ e poi alcune iterazioni di Lloyd

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 (functional approach!), such that:

- $\mathcal{A}$ solves the more general weighted variant.
- $\mathcal{A}$ requires space proportional to the input size.

lineare

data in input

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

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

senza pesi

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

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.

vero per k-means++, non per Lloyd

emit $(\cdot, (T_i, w(T_i)))$

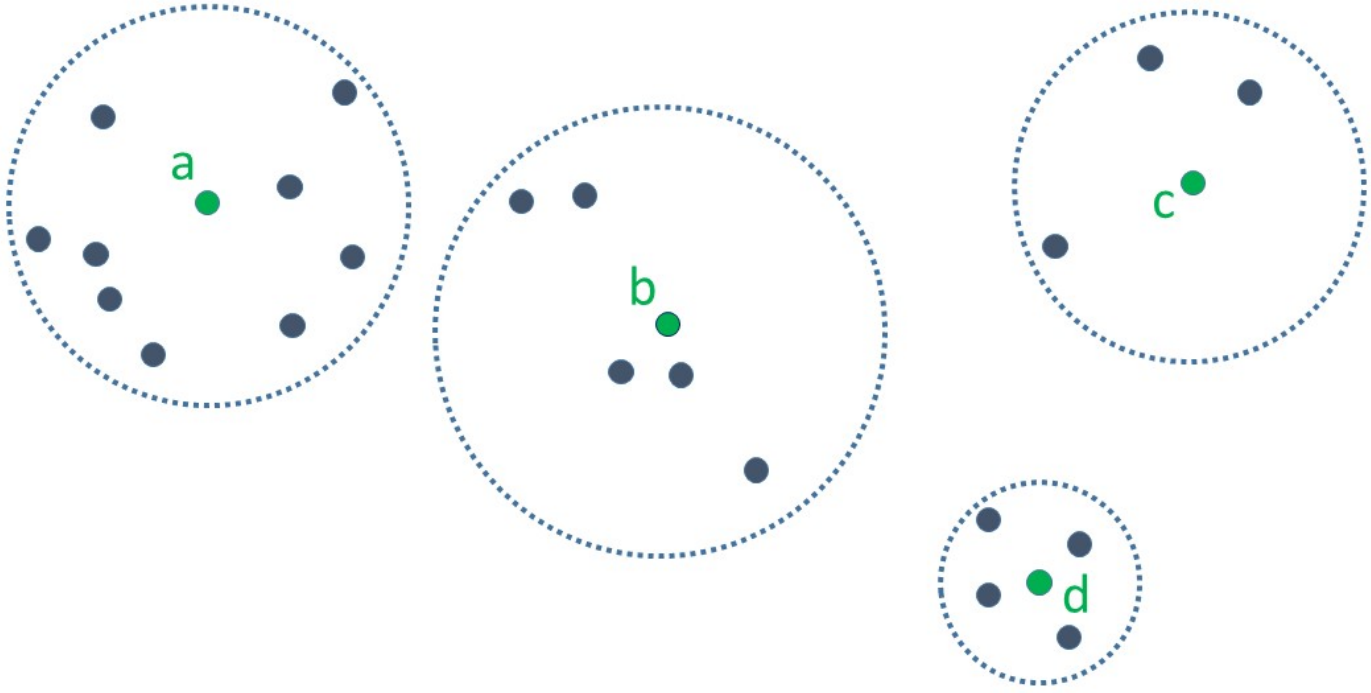
si può rimuovere, ma serve per garanzie teoriche

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



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 round 1 round 2

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

Se sappiamo solo $S_A(N)$ (non che $= O(N)$)

$\frac{N}{\ell} \to \frac{N}{\ell} + S_A\left(\frac{N}{\ell}\right)$; $\ell k \to \ell k + S_A(\ell k)$

$\ell = \sqrt{N/k}$ può non essere ottimo

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

Does MR-kmeans($\mathcal{A}$) provide accurate solutions?

Assume that $\mathcal{A}$ is an α-approximation algorithm for weighted k-means, for some $\alpha > 1$.

Theorem

Let S be the set of k centers returned by MR-kmeans($\mathcal{A}$) on input P. Then:

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

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

Why is this result useful?

1 - possiamo calcolare soluzione approx per grande insieme P con $|P| >>$ memoria processori

2 - A eseguito su insieme piccolo $(\frac{N}{\ell}, k)$ $\Rightarrow$ possiamo anche calcolare sol. esatte con A con algoritmi lenti se $\frac{N}{\ell}, k$ piccoli

NO dimostrazione

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

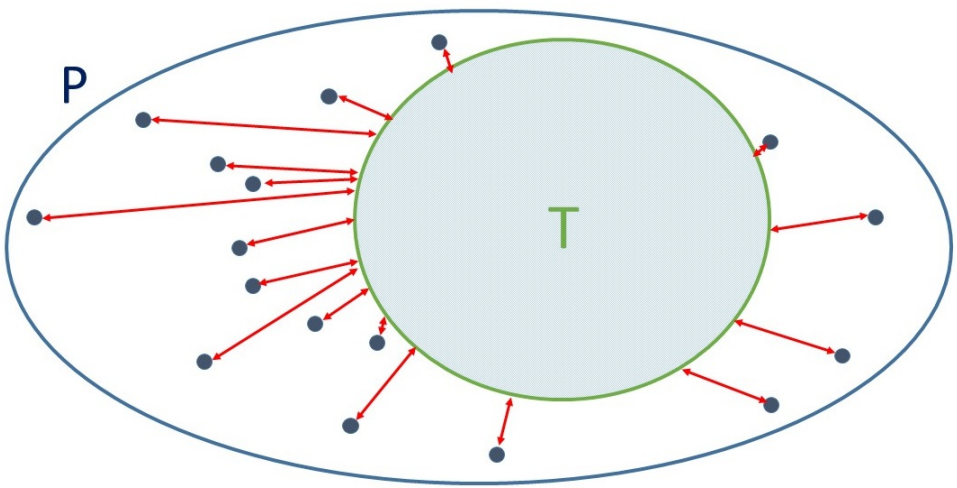
The above theorem is proved by combining the following two lemmas.

Lemma 1: coreset quality

Let $T = \cup_{i=1}^{\ell} T_i \subseteq P$ be the coreset computed by MR-kmeans($\mathcal{A}$) and let $\tau : P \to T$ be the associated proxy function.

Then, T is an α-coreset for P, k and the k-means objective, that is:

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



quanto lontano è rumore di punti fuori da T

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

Lemma 2: final solution quality

Let $T \subseteq P$, with associated proxy function $\tau : P \to T$, be an α-coreset for P, k and the k-means objective. Suppose that the points of T are weighted according to the proxy function τ.

Then, the solution S computed by $\mathcal{A}$ on the weighted coreset T, is an $O(\alpha^2)$-approximate solution to k-means for P, that is

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

teorema riguarda tutto P ⇒ usa due lemmi, "uno per round" ⇒
⇒ se ho α-coreset in input per round 2, teorema vale

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

- For k-means (as well as for k-center and k-median) good coresets are obtained by combining solutions to the same problem on smaller partitions. However, this is not always the case for other problems (e.g., diameter, diversity maximization).
- 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.

oversampling

Exercises

Exercise

Recall that k-means++ is a randomized algorithm and the quality of the solution is a random variable. Letting S be the set of centers returned by k-means++ for P it is known that there is a value $\alpha = \Theta(\ln k)$ such that

sol approx. α-approx

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

⇗ corretto con prob. alta

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.

eseguo h ind. inst. di k-means++ su P ⇒ operazioni random: selezione primo punto e selezione punto in loop ⇒ indipendenza fondamentale ⇒ ⇒ usare seed diversi per istanze

istanze

S_1 sol. 1; $c_1 = \phi_{k\text{means}}(P, S_1)$

S_2 // 2; $c_2 =$ // (P, S_2)

$\vdots$ $\vdots$; ; ; $\vdots$

S_h // h; $c_h =$ // (P, S_h)

soluzione finale: insieme che minimizza

$$S = \arg\min_{S_i,\, i \in [1,h]} c_i = \arg\min_{S_i,\, i \in [1,h]} \phi_{k\text{means}}(P, S_i)$$

si può dimostrare che, se h abbastanza grande

$$\Pr\left[\phi_{k\text{means}}(P, S) \leq \alpha\, \phi^{opt}_{k\text{means}}(P, K)\right] \geq 1 - \frac{1}{N}, \quad \alpha = \vartheta(\ln K)$$

Alg. fallisce quando $\phi_{k\text{means}}(P, S) > \phi^{opt}_{k\text{means}}(P, K)$

succede quando ogni istanza fallisce

$$\Pr[\text{alg. fallisce}] \geq \Pr[h \text{ istanze falliscono}] \leq \left(\frac{1}{2}\right)^h \leq \frac{1}{N} \Rightarrow$$

$$\Rightarrow N \leq 2^h \Rightarrow h \geq \log N$$

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.

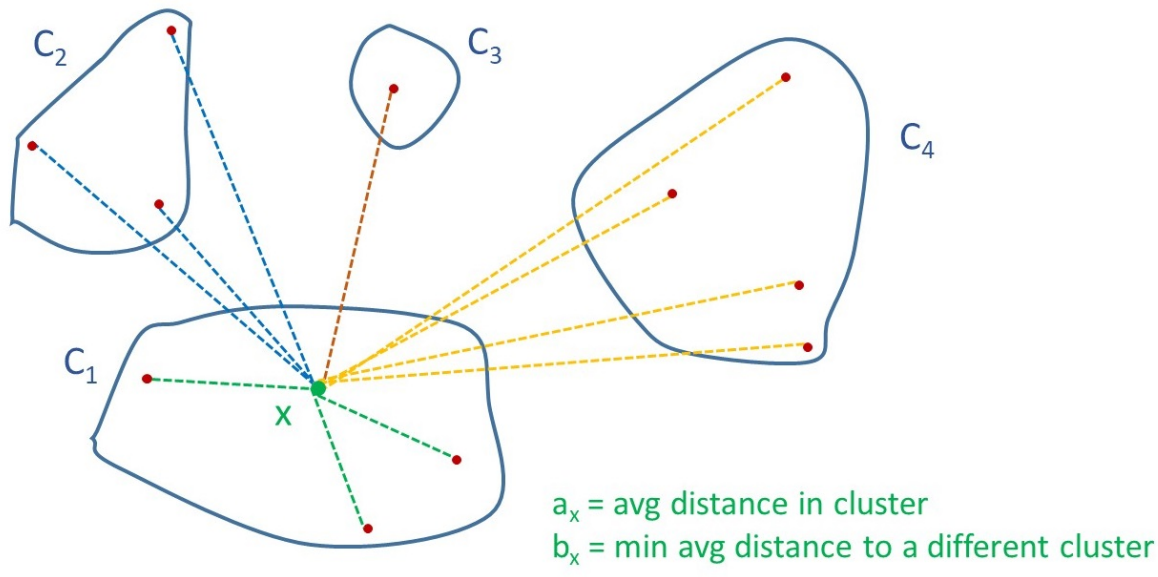
Observation: the silhouette coefficient is often used to select the most suitable number of clusters k, for a given clustering algorithm.

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

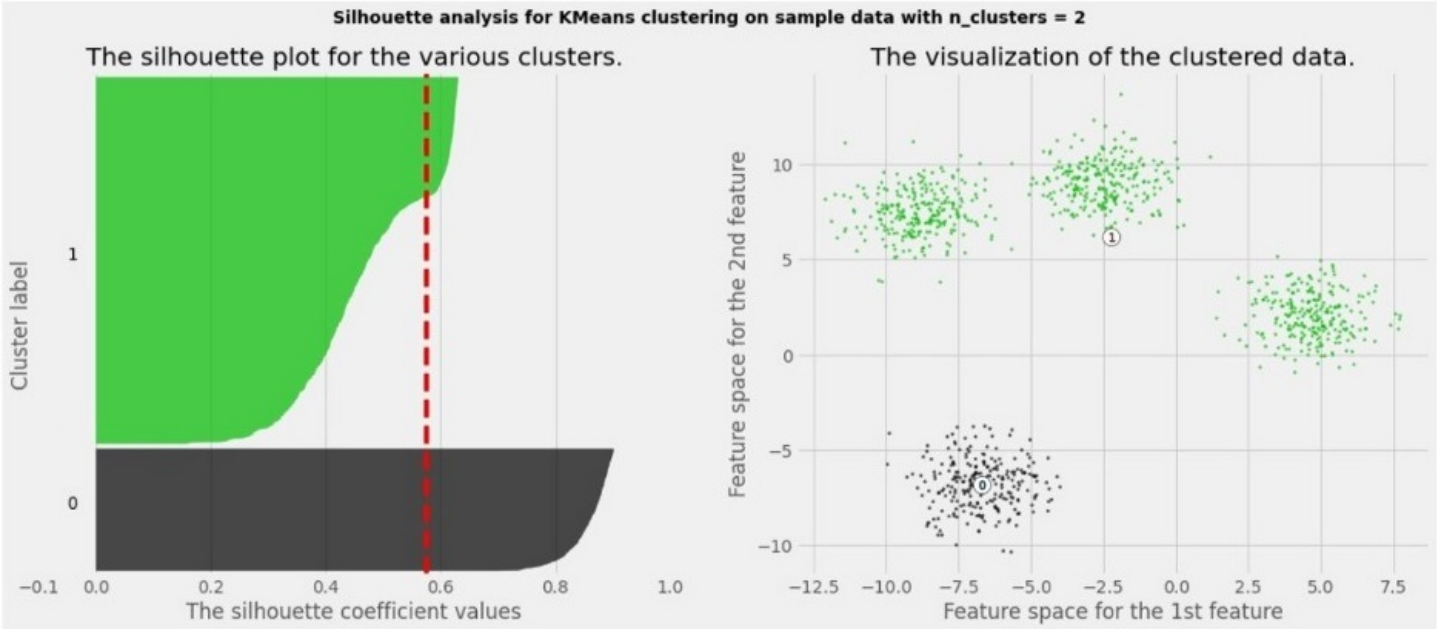
Let $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ be a partition of P into k clusters. For any $x \in P$ the silhouette coefficient for x is

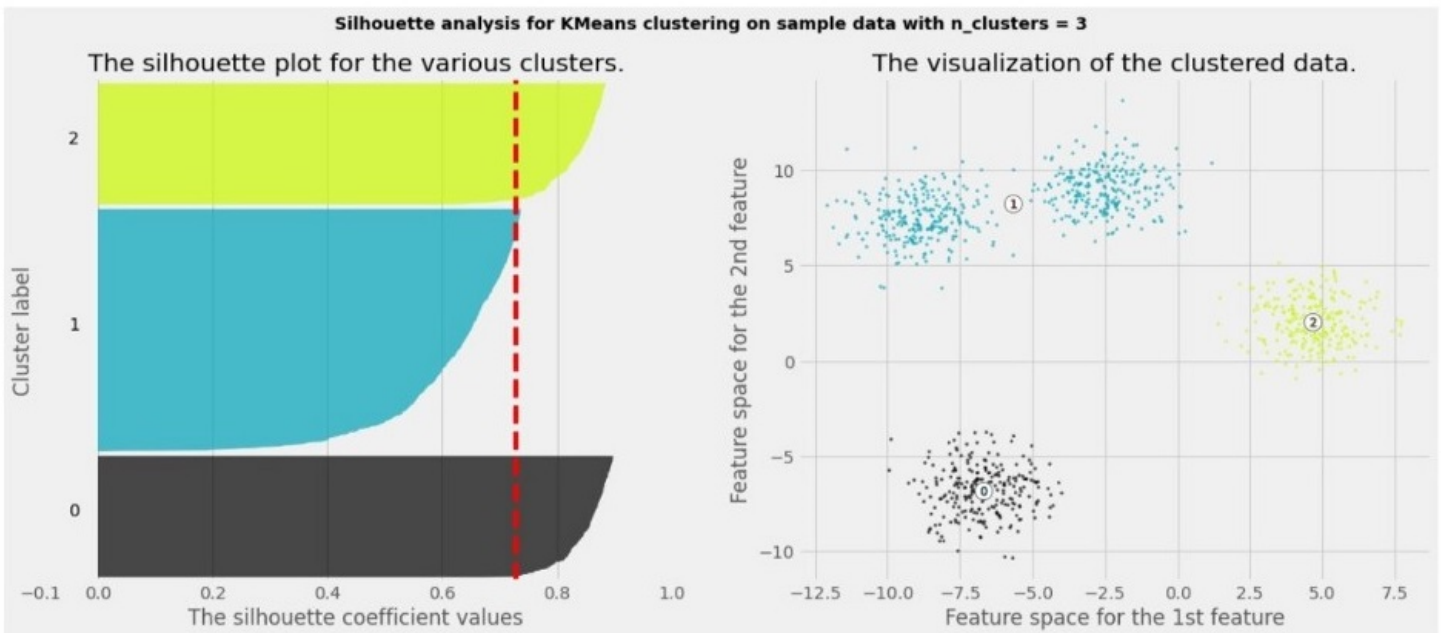
$$s_x = \frac{b_x - a_x}{\max\{a_x, b_x\}},$$

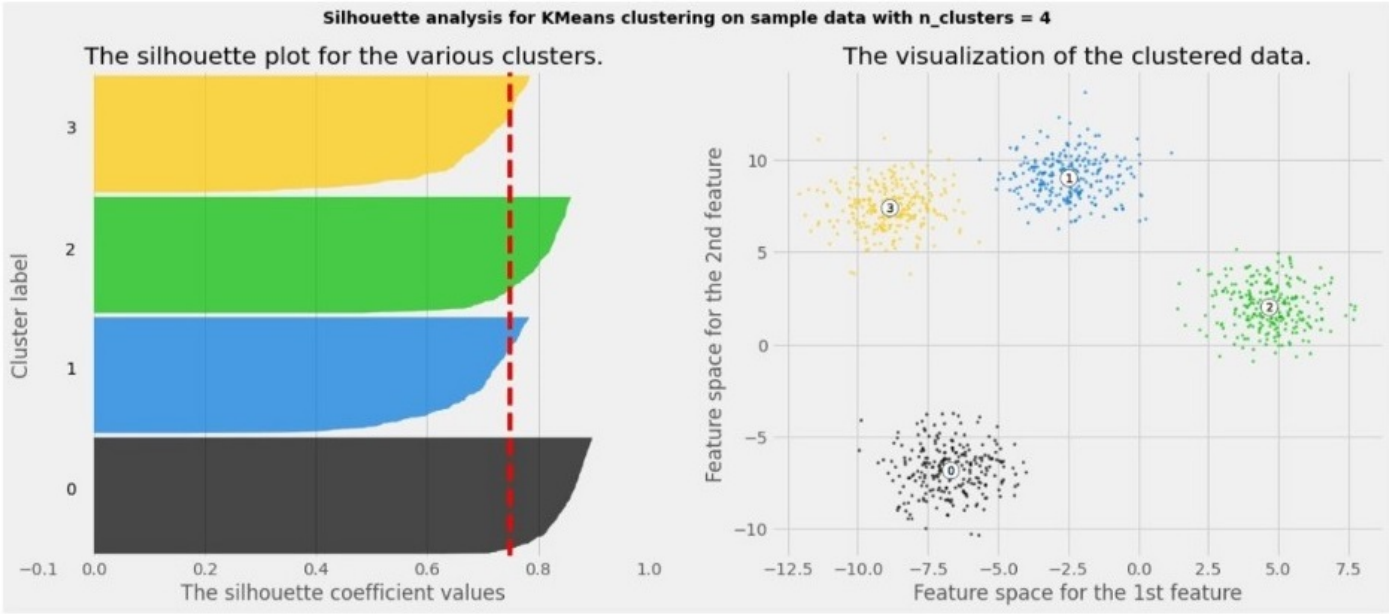
To measure the quality of $\mathcal{C}$ we define the average silhouette coefficient as

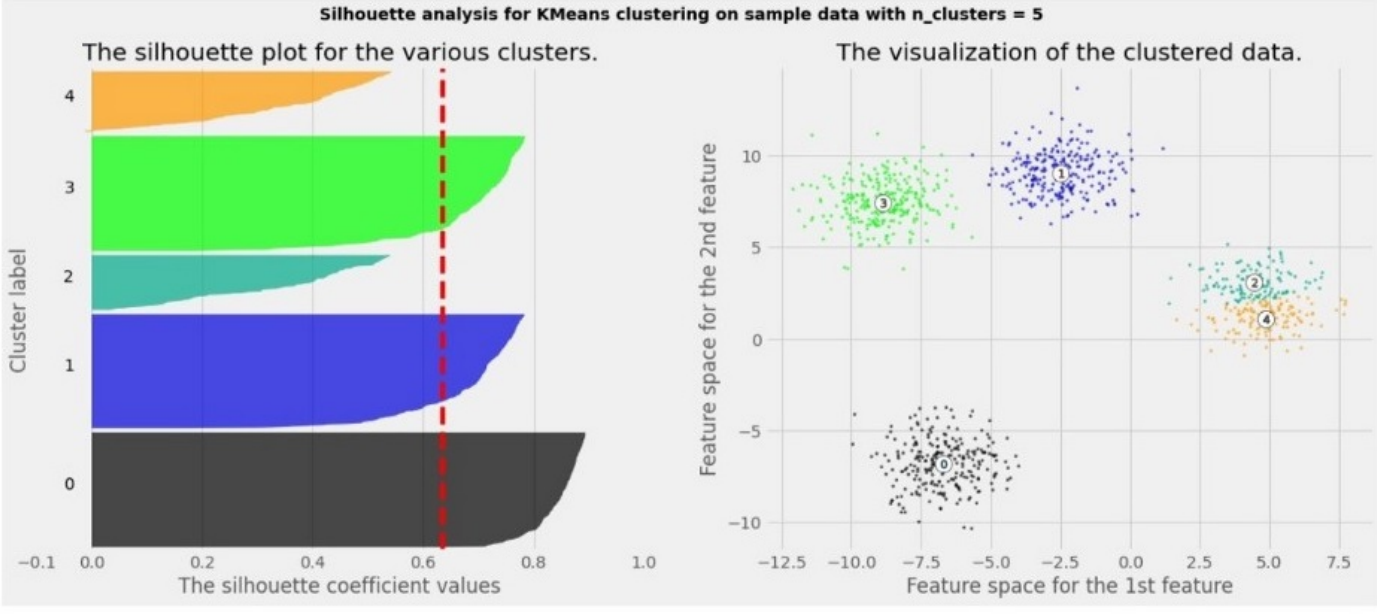
$$s_{\mathcal{C}} = \frac{1}{|P|} \sum_{x \in P} s_x.$$

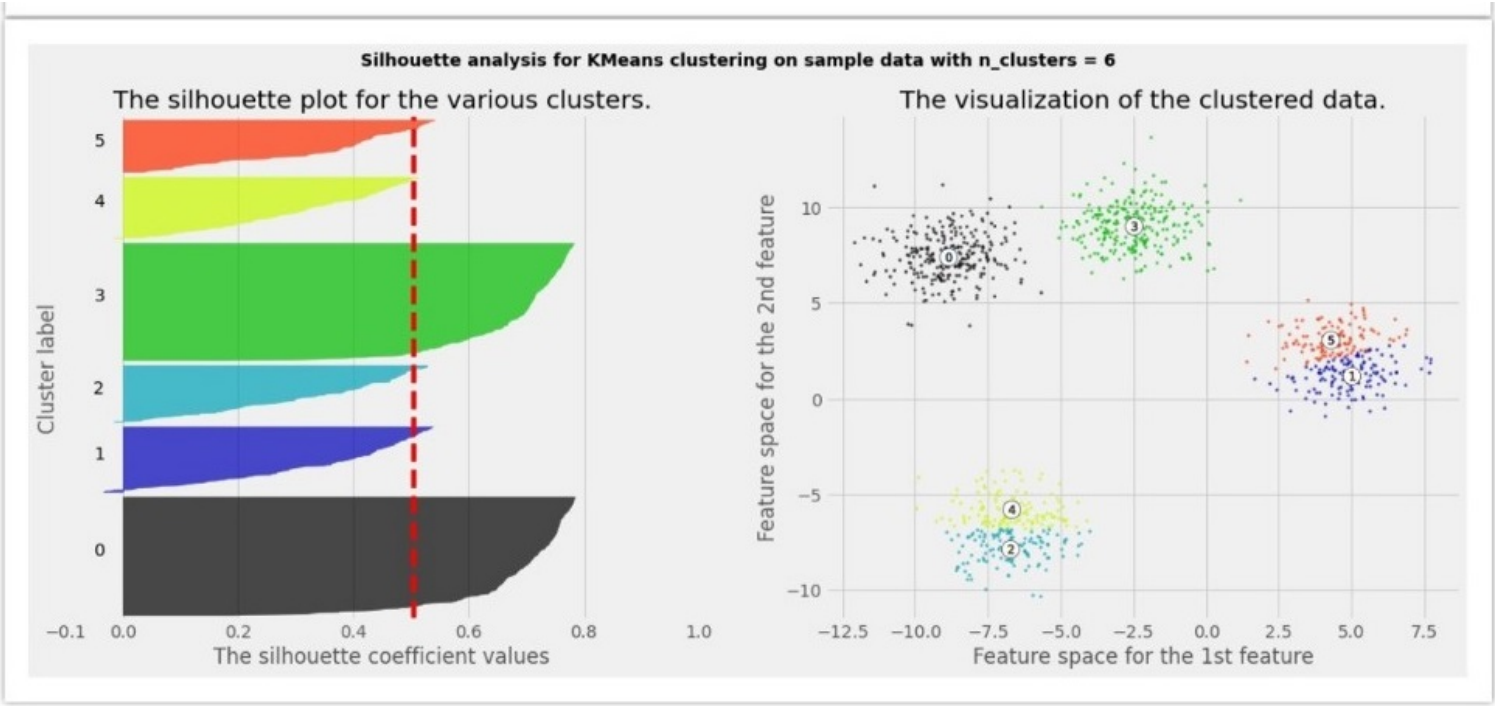
Unsupervised evaluation: silhouette


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


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

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

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

Exercise

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

Summary of Parts 1 and 2

- (Composable) coreset technique.
- Metric space and prominent distance functions.
- Combinatorial optimization problem and approximation algorithm.
- k-center clustering:
  - Definition of the problem.
  - Farthest-First Traversal algorithm.
  - MR-Farthest-First Traversal.
- k-means/median clustering:
  - Definition of the problem.
  - Critical review of known algorithms.
  - Weighted variant of the problem.
  - MR-kmeans
- Unsupervised clustering evaluation:
  - Silhouette coefficient.
  - Approximating the sum of distances.
  - Approximating the Silhouette.

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.

CPPU17 M. Ceccarello, A. Pietracaprina, G. Pucci, E. Upfal. MapReduce and Streaming Algorithms for Diversity Maximization in Metric Spaces of Bounded Doubling Dimension. Proc. VLDB Endow. 10(5): 469-480 (2017)

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).