Clustering

(Part 2)

k-means clustering

General 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 discovering ball-shaped clusters.
- 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.
- Some established algorithms for k-means clustering perform well in practice although only recently a rigorous assessment of their performance-accuracy tradeoff has been carried out.

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

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

Definition

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

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

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

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

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 need not necesseraly belong to the input pointset).

Lloyd's algorithm

- Also known as k-means algorithm, it was developed by Stuart Lloyd in 1957. In 2006 it was listed as one of the top-10 most influential data mining algorithms. It is considered among the most popular clustering algorithm used in both scientific and industrial applications.
- It focuses on Euclidean spaces and does not require cluster centers to belong to the input pointset. (However, it can be restricted to satisfy this latter requirement.)
- It relates to a generalization of the Expectation-Maximization algorithm

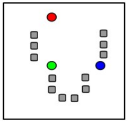
Lloyd's algorithm (cont'd)

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 which is a good approximation to the k-means problem for P. The 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 $C_i$
  $S' \leftarrow \{c'_1, c'_2, \ldots, c'_k\}$
  Let $\mathcal{C} = (C_1, C_2, \ldots, C_k; S')$
  **if** $\Phi_{\mathrm{kmeans}}(\mathcal{C}) < \Phi$ **then**
    $\Phi \leftarrow \Phi_{\mathrm{kmeans}}(\mathcal{C})$
    $S \leftarrow S'$
  **else** stopping-condition $\leftarrow$ true
**return** $\mathcal{C}$
```

Example

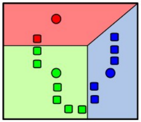


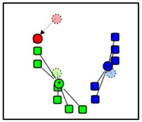
Initial centers

Partition

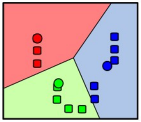
Calculation of centroids

Iteration 1:





Iteration 2:



. . .

Lloyd's algorithm: analysis

Theorem

The Lloyd's algorithm always terminates.

Proof.

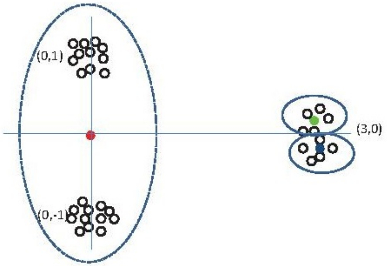
Let $\mathcal{C}_t$ be the k-clusterings computed by the algorithm in Iteration $t \geq 1$ of the while loop (line 3 of the loop). We have:

- By construction, the objective function values $\Phi_{\text{kmeans}}(\mathcal{C}_t)$'s, for every t except for the last one, form a strictly decreasing sequence. Also, in the last iteration (say Iteration t^*) $\Phi_{\text{kmeans}}(\mathcal{C}_{t^*}) = \Phi_{\text{kmeans}}(\mathcal{C}_{t^*-1})$, since the invocation of Partition and the subsequent recomputation of the centroids in this iteration cannot increase the value of the objective function.
- Each $\mathcal{C}_t$ is uniquely determined by the partition of P into k clusters computed at the beginning of Iteration t.
- By the above two points we conclude that all $\mathcal{C}_t$'s, except possibly the last one, are determined by distinct partitions of P.

The theorem follows immediately since there are k^N possible partitions of P into k clusters (counting also permutations of the same partition). $\square$

Observations on Lloyd's algorithm

- Lloyd's algorithm may be trapped into a local optimum whose value of the objective function can be much larger than the optimal value. Hence, no guarantee can be proved on its accuracy. 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 Lloyd's algorithm (cont'd)

- While the 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, the algorithm requires much less than N iterations, and, if properly seeded (see next slides) it features guaranteed accuracy.
- 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 Lloyd's algorithm 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 [AV07], D. Arthur and S. Vassilvitskii developed k-means++, a careful center initialization strategy that yields clusterings not too far from the optimal ones.

k-means++

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** $i \leftarrow 2$ **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 $(d_p)^2/(\sum_{q \in P-S}(d_q)^2)$
   $S \leftarrow S \cup \{c_i\}$
**return** $S$
```

Observation: Altough k-means++ already provides a good set of centers for the k-means problem (see next theorem) it can be used to provide the initialization for Lloyd's algorithm, whose iterations can only improve the initial solution.

k-means++ (cont'd)

Details on the selection of c_i in k-means++

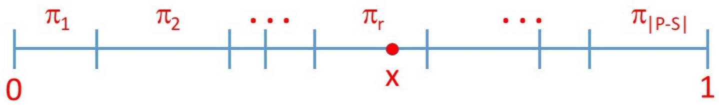
Consider Iteration i of the for loop of k-means++ ($i \geq 2$)

Let S be the current set of $i-1$ already discovered centers.

- Let $P - S = \{p_j \,:\, 1 \leq j \leq |P - S|\}$ (arbitrary order), and define $\pi_j = (d_{p_j})^2/(\sum_{q \in P-S}(d_q)^2)$. Note that the π_j's define a probability distribution, that is, $\sum_{j \geq 1} \pi_j = 1$.
- Draw a random number $x \in [0,1]$ and set $c_i = p_r$, where r is the unique index between 1 and $|P - S|$ such that

$$\sum_{j=1}^{r-1} \pi_j \leq x \leq \sum_{j=1}^{r} \pi_j.$$

k-means++ (cont'd)



k-means++ (cont'd)

Theorem (Arthur-Vassilvitskii'07)

Let $\Phi^{\rm opt}_{\rm kmeans}(k)$ be the minimum value of $\Phi_{\rm kmeans}(\mathcal{C})$ over all possible k-clusterings $\mathcal{C}$ of P, and let $\mathcal{C}_{\rm alg} = Partition(S, P)$ be the k-clustering of P induced by the set S of centers returned by the k-means++. Note that $\mathcal{C}_{\rm alg}$ is a random variable which depends on the random choices made by the algorithm. Then:

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

where the expectation is over all possible sets S returned by k-means++, which depend on the random choices made by the algorithm.

N.B. We skip the proof which can be found in the original paper.

Remark: There exists a sequential constant-approximation algorithm that runs in cubic time for pointsets in $\Re^D$ with constant D, however it is impractical for large datasets. (See references in [B+12].)

k-means clustering in MapReduce

The following two exercises study the efficient implementation of k-means++ and of Lloyd's algorithm in MapReduce.

Consider a set P be a set of N points from $\Re^D$, with D constant, and an integer $k \in (1, \sqrt{N}]$.

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 $M_A = O(N)$.

Exercise

Show how to implement one iteration of Lloyd's algorithm in MapReduce using a constant number of rounds, local space $M_L = O\left(\sqrt{N}\right)$ and aggregate space $M_A = O(N)$. Assume that at the beginning of the iteration a set S of k centers and a current estimate Φ of the objective function are available. The iteration must compute the partition of P around S, the new set S' of k centers, and it must update Φ, unless it is the last iteration.

k-means clustering in MapReduce (cont'd)

- In practice, one could initialize the centers using k-means++ and then run a few rounds of Lloyd's algorithm, until the quality of the clustering shows little improvement.
- In order to reduce the number of iterations of k-means++, hence the number of MapReduce rounds, a parallel variant of k-means++ (k-means$||$) has been proposed in [B+12]: in this variant, a coreset of $> k$ centers is selected in a few iterations and then k centers are extracted from the coreset using a weighted version of k-means++ or of any other algorithm for k-means clustering.
- k-means$||$ is implemented in the Spark MLlib.

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

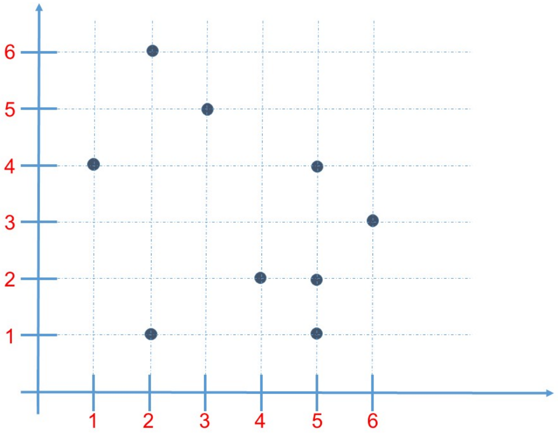
Input Set P of N points from a metric space (M, d), integer $k > 1$

Output k-clustering $\mathcal{C} = (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$ of P which is a good approximation to the k-median problem for P. The centers must belong to P.

PAM algorithm (cont'd)

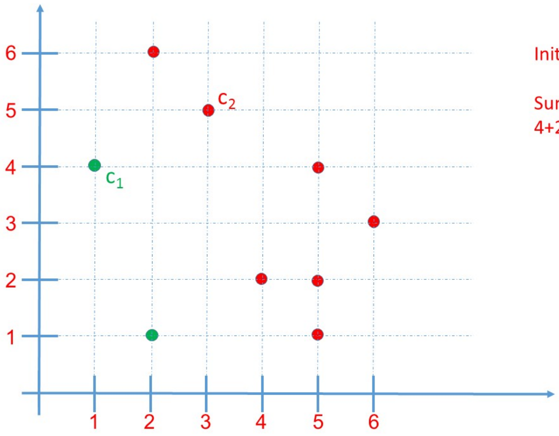
```
$S \leftarrow \{c_1, c_2, \ldots, c_k\}$ (arbitrary set of $k$ points of $P$)
$\mathcal{C} \leftarrow$ Partition$(P, S)$
stopping-condition $\leftarrow$ false
**while** (!stopping-condition) **do**
  stopping-condition $\leftarrow$ true
  **for each** $p \in P - S$ **do**
    **for each** $c \in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

PAM algorithm: example



Manhattan distance

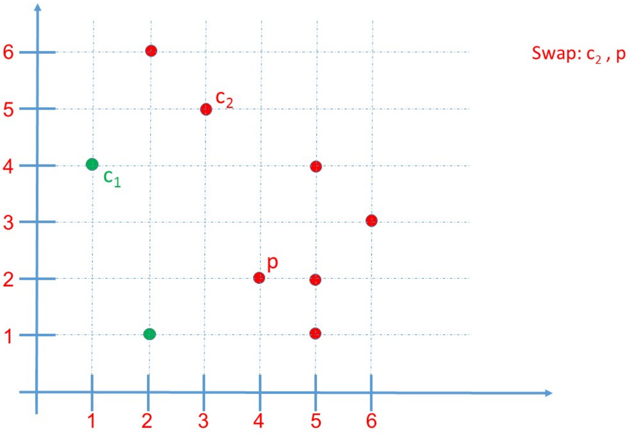
PAM algorithm: example



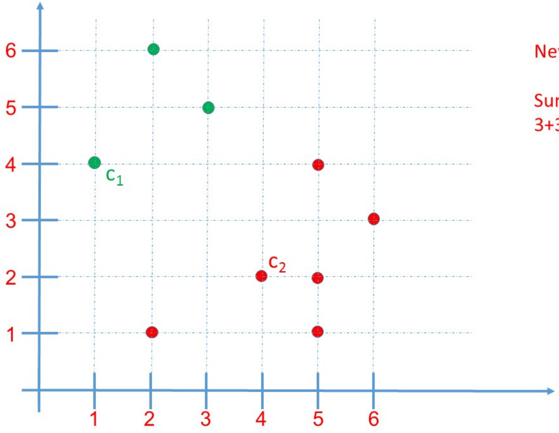
Initial Clusters

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

PAM algorithm: example



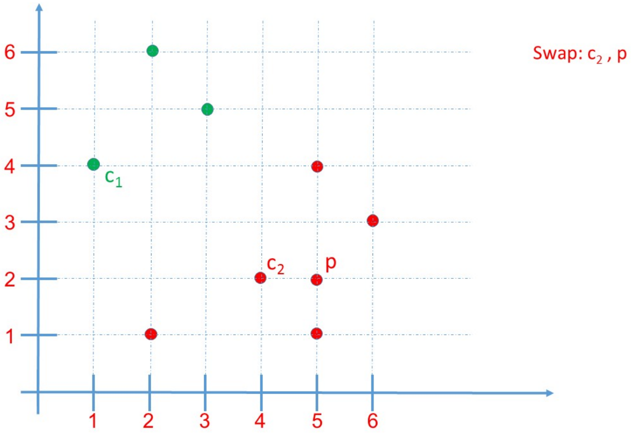
PAM algorithm: example



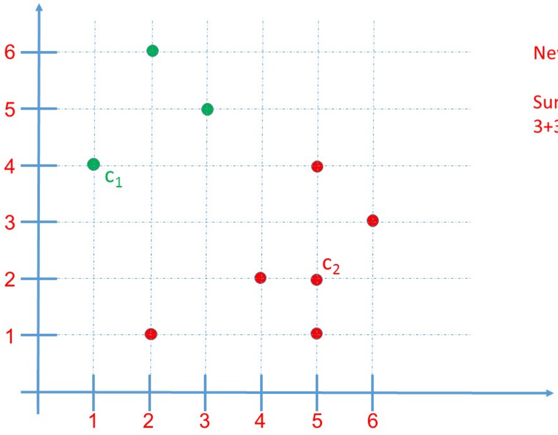
New Clusters

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

PAM algorithm: example



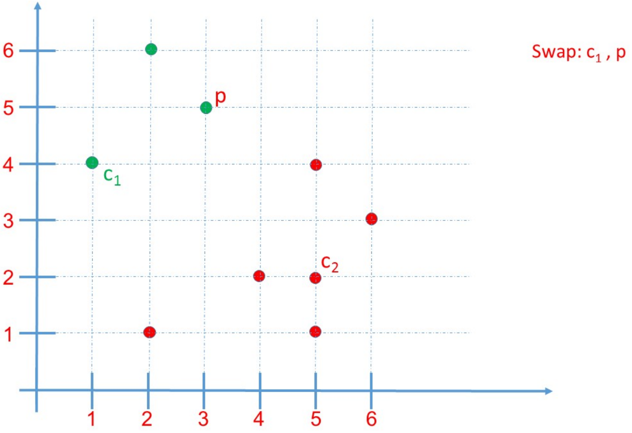
PAM algorithm: example



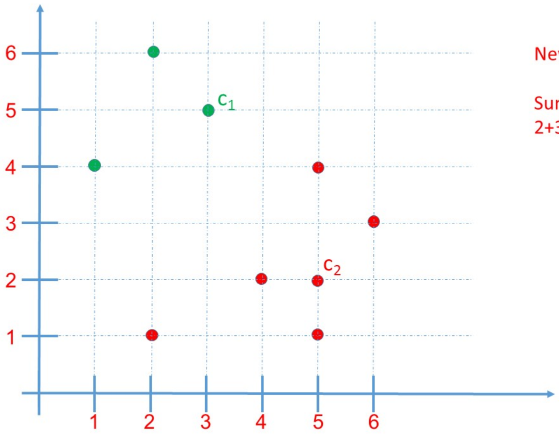
New Clusters

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

PAM algorithm: example



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

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

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 $\Phi_{\rm kmedian}(\mathcal{C}_{\rm alg})$ 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 is very 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 used by 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 $[x/2, x]$.

Weighted k-median clustering

One can define the following, more general, weighted variant of the k-median clustering problem.

Let P be a set of N points from a metric space (M, d), and let $k > 1$ be an integer. Suppose that for each point $p \in P$, an integer weight $w(p) > 0$ is also given. We want to compute the k-clustering $\mathcal{C} = (C_1, C_2, \ldots, C_k; c_1, c_2, \ldots, c_k)$ of P which minimize the following objective function

$$\Phi^{w}_{\mathrm{kmedian}}(\mathcal{C}) = \sum_{i=1}^{k} \sum_{p \in C_i} w(p) \cdot d(p, c_i),$$

that is, the average weighted distance of a point from its cluster center.

By considering each point of P as representing $w(p)$ identical copies of p, one can easily devise a weighted version of the PAM algorithm to solve Weighted k-median clustering with the same approximation guarantees.

k-median clustering in MapReduce: MR-PAM

The PAM algorithm is impractical for large datasets.

The following coreset-based MapReduce algorithm (MR-PAM) is more suitable for the big data scenario.

- Round 1: Partition P arbitrarily in ℓ subsets of equal size $P_1, P_2, \ldots, P_\ell$ and execute the PAM algorithm independently on each P_i. For $1 \leq i \leq \ell$, let T_i be the set of k centers of the clustering computed on P_i and, for each $p \in T_i$, let $w(p)$ be the number of points in the cluster centered at p.
- Round 2: Gather the coreset $T = \cup_{i=1}^{k} T_i$ of $\ell \cdot k$ points, together with their weights. Using a single reducer, run the weighted version of PAM on T to identify a set $S = \{c_1, c_2, \ldots, c_k\}$ of k centers.
- Round 3: Run Partition(P, S) to return the final clustering.

MR-PAM: analysis

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

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

In [EIM11] the authors claim that MR-PAM is a 15-approximation algorithm. In fact, by using an α-approximation algorithms instead of (weighted) PAM in Rounds 1 and 2, they claim that the final approximation factor is 3α.

In order to ensure polynomial time for the reduce functions, in Rounds 1 and 2 one should run the version of PAM that stops whenever the improvement in the objective function is below a fixed threshold.

Exercises

Exercise

Let $\Phi_{\mathrm{kmeans}}^{\mathrm{opt}}(k)$ be the minimum value of $\Phi_{\mathrm{kmeans}}(\mathcal{C})$ over all possible k-clusterings $\mathcal{C}$ of P, and let $\mathcal{C}_{\mathrm{alg}} = \mathsf{Partition}(S, P)$ be the k-clustering of P induced by the set S of centers returned by the k-means++.

1. Using Markov's inequality determine a constant $c > 0$ such that

$$\Pr\left(\Phi_{\mathrm{kmeans}}(\mathcal{C}_{\mathrm{alg}}) \leq c \cdot \Phi_{\mathrm{kmeans}}^{\mathrm{opt}}(k)\right) \geq 1/2.$$

2. Show that by running $O(\log N)$ independent instances of k-means++ and by taking the best clustering $\mathcal{C}_{\mathrm{best}}$ found among all repetitions, we have that

$$\Pr\left(\Phi_{\mathrm{kmeans}}(\mathcal{C}_{\mathrm{best}}) \leq c \cdot \Phi_{\mathrm{kmeans}}^{\mathrm{opt}}(k)\right) \geq 1 - 1/N.$$

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

EIM11 A. Ene, S. Im, B. Moseley. Fast clustering using MapReduce. Proc. of KDD 2011: 681-689.