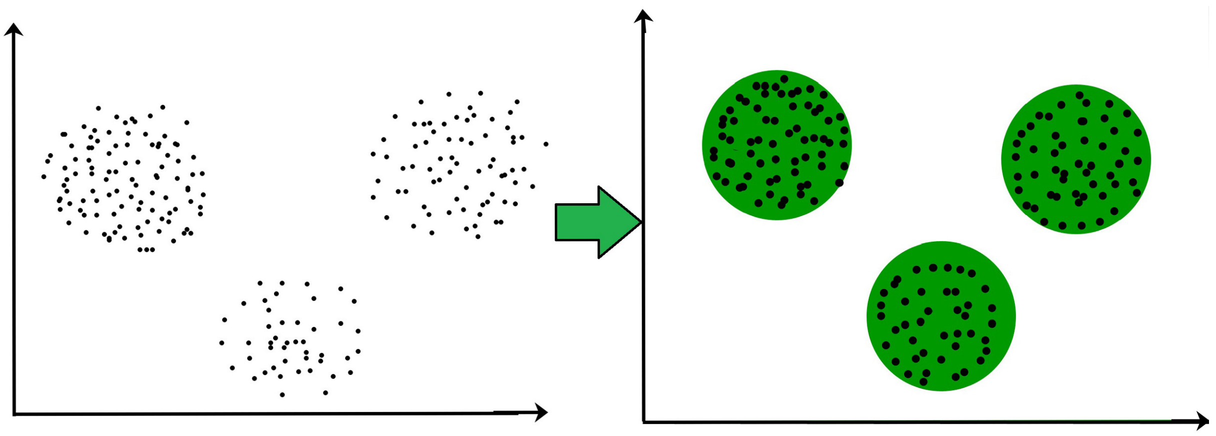
Advanced Algorithms Project

Clustering Problems

Phung Dang Bao Ngoc i6290653
Bregje Derks i6293858

Group 6
EBC2121



School of Business and Economics
Maastricht University
April 2024

Contents

List of Algorithms

1 Introduction: Problem Description

In this project, we design and investigate algorithms that attempt to solve k-median problem and k-center with outliers problem.

For the k-median problem, given a set of points $P \in \mathbb{R}^d$ and a number $k \in \mathbb{N}$, we find $Q \subseteq P$ with $|Q| = k$ such that we minimize $\sum_{p \in P} \min_{q \in Q} d(p, q)$ where $d(p, q)$ is the standard Eclidean distance between p and q.

For the k-center with outliers, given a set of points $P \subseteq \mathbb{R}^d$ and numbers $k \in \mathbb{N}$ and $m \in \mathbb{N}$, we try to find a subset $Q \subseteq P$ with $|Q| = k$ such that most of the points in P (at least m many) are concentrated around the points in Q. Formally, we want to select Q and determine a radius $r \geq 0$ so that for at least m many of the points $p \in P$ we have $d(p, q) \leq r$ for some $q \in Q$.

K-median problem and k-center with outliers problem are an interesting topic of research since it has high applicability to real life problems, such as facility location decision, healthcare planning, telecommunication network design, etc. However, these problems are not simple to solve. They are NP-hard; therefore, we cannot solve them to optimality in polynomial time. Hence, we devise and propose two algorithms for each problem with inspiration from past research and our own ideas, as well as providing a detailed analysis of the problem. We implement our algorithms in Java (See attached Java files) to experiment with the algorithms and report the results.

2 Literature Review

In the past, extensive research has been conducted into solving these problems. Lloyd (1982) proposes Lloyd's algorithm, also known as k-means clustering algorithm, which does not have a fixed approximation rate and has a runtime of $\mathcal{O}(nkt)$, where n is the number of data points and t is the number of iterations until convergence. Kaufman and Rousseeuw (1990) published their research on Partitioning around Medoids (PAM) algorithm, which although is more robust to outliers, can be more computationally expensive for large datasets. This algorithm also does not have a constant approximation rate and has a run time complexity of $\mathcal{O}(k(n-k)^2)$. Gonzalez's algorithm (Gonzalez (1985)) has a 2-approximation rate and a runtime complexity of $\mathcal{O}(nk)$.

There are many other algorithms that attempts to solve the k-median problems and extend the research on this matter. There are Charikar et al. (2002)'s algorithm is based on primal-dual schema concept, combined with a greedy approach; or Bâdoiu and Clarkson (2003)'s algorithm that is based on core-set and has an $(1 + \epsilon)$ approximation rate.

Although the algorithm research on these problems is extensive, researchers are still challenged with the trade-off between solution quality and runtime complexity of the algorithm, since the problems at hand are NP-hard and cannot be solved in polynomial time.

In the following sections of the project, we build our algorithms with inspiration from Lloyd's and Gonzalez's Algorithm, along with Local Outlier Factor (Breunig et al. (2000)), and our own ideas to create the followingly proposed algorithms, while analyzing the solution quality and runtime of them.

3 K-Median Problem

3.1 Proposed Algorithms for K-Median

Firstly, to solve the aforementioned k-median problem, we propose two algorithms: one inspired by the work of Lloyd's and Gonzalez's algorithm, and one inspired by the idea of range-based initialization.

3.1.1 Algorithm 1: A combination of Lloyd's and Gonzalez's Algorithm

The first algorithm (Algorithm 1: See Appendix for the pseudocode) has the initialization step similar to Gonzalez's algorithm, where we choose an arbitrary point from the dataset as the first center. Then, for subsequent centers, we select points that are the furthest away in Euclidean distance from the nearest chosen centers. Afterwards, we assign all the remaining data points to the nearest center, forming a cluster. This initialization is 2-approximation and run in $\mathcal{O}$(nk) (Gonzalez (1985)).

We then attempt to improve this solution further by implementing an adapted version of Lloyd (1982)'s algorithm that recalculates the centers of cluster after assignment, and repeat the assignment process and update centers until the centers do not change. In other words, we reassign centers and clusters until the algorithm converges to a solution.

3.1.2 Algorithm 2: Range-based Initialization

In the second algorithm (Algorithm 2: See Appendix), we initialize choosing k centers by randomizing one point in each set to be the center. The sets are constructed by dividing the range of either x-coordinates or y-coordinates (whichever is larger) into k equal-sized sets. Subsequently, we implement the recalculating assignment method similar to Lloyd (1982)'s algorithm until convergence.

3.2 Instance Analysis

3.2.1 Algorithm 1: Solution Quality and Time Complexity Analysis

This algorithm leverages Gonzalez's Algorithm to select initial centers, which can be beneficial when dealing with datasets that are more complex, such as datasets with outliers. By starting with centers that are well-distributed across the dataset, Algorithm 1 may achieve more stability, minimizing the impact of outliers and ensuring more consistent clustering results.

While Algorithm 1 may perform better in specific dataset scenarios, like those with uneven distributions or different cluster densities, it may not consistently outperform Algorithm 2 in runtime or accuracy, particularly with well-behaved datasets.

To illustrate this, we investigate an example of instance 001. For this instance, the runtime of this algorithm is fast. For most runs, the algorithms converges within 5-6 iterations. However, the total distance of points to their centers vary largely: from approximately 168 million to 225 million (Euclidean distance unit), further demonstrates that the solution quality depends on the result of the randomized initialization process.

3.2.2 Algorithm 2: Solution Quality and Time Complexity Analysis

Regarding run time complexity, this algorithm tends to have a straightforward implementation and efficient convergence, particularly when the dataset is well-distributed and does not contain significant outliers. The initialization based on the range of the data can help in quickly identifying suitable starting centers, leading to faster convergence in some cases.

This algorithm does not have a constant approximation rate since the solution quality depends on the randomization in initialization process. However, when the dataset follows a relatively uniform distribution without significant outliers, Algorithm 2 can produce accurate clustering results due to its effective initialization and convergence mechanism.

For instance 001, this algorithm tends to take more iterations to converge, ranging from 5 to 15 iterations. The total distance of points to their centers are around 214 - 250 million (Euclidean distance unit). That means that sometimes Algorithm 2 outperforms Algorithm 1, but this event is not consistent.

4 K-Centers with Outliers Problem

For k-centers with outliers problem, along with minimizing the Euclidean distance, we also need to minimize the radius so that at least m many points $p \in P$ are covered within the radius. This requires alterations in our approach compared to k-median. We propose two algorithms for this problem: one using Local Outlier Factor (LOF) to remove outliers from data inputs, then treat the data as k-median problem; and one using a heuristic that combines binary search and greedy algorithm to find such radius.

4.1 Proposed Algorithms for K-Median with Outliers

4.1.1 Algorithm 3: Local Outlier Factor and a heuristic

As mentioned above, Algorithm 3 (See Appendix) uses LOF concept to eliminate outliers from the data set. To ensure that at least m points are covered by the radius r, we only eliminate$(n - m)$ "outliers" from the dataset, where n is the number of total data points. Since we later treat this "cleaned" data as k-median problem, we are minimizing the Euclidean distance to centers for these m data points.

LOF works by comparing the local reachability density around a specific point with the density around its nearby points. If a point has a significantly lower density than its neighbors, it's considered an outlier. We then sort the data points by this LOF score and eliminate the outliers. Then we treated this "cleaned" dataset, where $|P| = m$ instead of $|P| = n$, as a k-median problem by implementing either Algorithm 1 or Algorithm 2 as mentioned in Section 3.

4.1.2 Algorithm 4: A Combination of Binary Search and Greedy Algorithm

In Algorithm 4 (See Appendix) , we take advantage of binary search to find the optimal radius. We notice that the radius is bounded by the lower bound of 0 and an upper bound of the maximum distance between any two points in P $(max_{i,j \in 1,\ldots,n, i \neq j} d(p_i, p_j))$. We take the radius as the midpoint of these two bounds, then check whether there exists a center selection and point assignment that can cover at least m points.

If there exists such assignment, we try to reduce the radius further by setting the new upper bound by the current midpoint r, to see if there is a smaller radius r that can cover m points. Otherwise, we increase r by setting lower bound to current midpoint r. This procedure stops when r converges to a number, by checking if the difference between the lower and upper bound has fallen below the precision limit (set to $1e - 5$).

To check whether there exists an assignment that can cover at least m points for a given r, we implemented a greedy heuristic that select centers by greedily adding the points that can cover most points within the radius r. In this step, we also save the best "greedy" center selection and point assignment to report as the result.

4.2 Instance Analysis

4.2.1 Algorithm 3: Solution Quality and Time Complexity Analysis

LOF (Local Outlier Factor) identifies outliers by examining how densely points are clustered together in their local neighborhoods. This method improves the clustering by efficiently removing data points that significantly deviate from the overall pattern. While LOF excels in datasets with clear density variations and local structures, its performance may be less reliable in datasets with uniform distributions or complex noise patterns.

The solution quality of this algorithm also depends on the heuristic chosen to solve the k-median problem of newly transformed data. Fore more detail on these heuristics, see Section 3. The general idea is that when removed outliers, Algorithm 2 may have better performance than Algorithm 1 since the data has overall has a more even distribution pattern compared to the original inputs.

In terms of minimized distance, the output total distance varies greatly. In some cases, LOF combined with Algorithm 1 or 2 outperforms Algorithm 4, while in some cases the opposite happens. For instance 001, the radius of this algorithm, which is around 164,000, seems to be significantly larger than Algorithm 4, which is around the 60-70,000 range.

4.2.2 Algorithm 4: Solution Quality and Time Complexity Analysis

By using binary search to search for optimal radius r, we are efficiently searching through the range of r to find the optimal r. However, generally, the number of iterations for binary search is $\mathcal{O}(log(n)$, while finding the upper bound (largest distance between any two points) of radius and the greedy algorithm to check if the radius can cover m points both have runtime of $\mathcal{O}(n^2)$. This algorithm overall has runtime complexity of $\mathcal{O}(n^2 log(n))$.

This runtime complexity might not pose a significant issue to relatively small datasets, but since it is pseudo-polynomial, as the input size increases, the algorithm's runtime performance might deteriorate.

Similar to Algorithm 3, this algorithm's minimized distance seems to vary widely each run; however, we observe that this algorithm seems to have better performance in minimizing the radius r compared to Algorithm 3.

5 Conclusion

Although intriguing and relevant to solve, k-median and k-center with outliers problems are challenging since it is NP-hard. We attempt to implement our algorithms in Java to tackle these problems and provide a detailed analysis on their solution quality and time complexity. Through our analysis, all our algorithms have its advantages and disadvantages regarding solution quality and runtime; therefore, in order to choose which one to implement, we need to have an understanding of the input sizes, distribution patterns to choose the appropriate algorithms. For further improvements of these algorithms, it is interesting to dive deeper to improve the setbacks of these algorithms.

6 Appendix

```
Algorithm 1: K-Median Problem - A Combination of Gonzalez's and Lloyd's Algorithm
Data: Set of points $P \subseteq \mathbb{R}^d$, number $k \in \mathbb{N}$
Result: Subset $Q \subseteq P$ such that $|Q| = k$
Randomize $q_1 \in P$, add $q_1$ to Q;
while $|Q| < k$ do
    Find remaining centers by iteratively choosing the point that has the furthest minimum
      distance to any center in $Q$ by ;
    Adding new center $p$ to Q: $\max_{p \in P, p \notin Q}(\min_{q \in Q}(d(p, q)))$ ;
    Assign $p \notin Q$ to clusters by: ;
    $p \in C_q$ (cluster of q being center) if $\min_{q \in Q} d(p, q)$ ;
end
Update centers by recalculating within clusters by: ;
for all clusters $C_q$ do
    Find median point $q^* = (\bar{x}, \bar{y})$ ;
    $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \forall i \in C_q$ ;
    $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad \forall i \in C_q$ ;
    New center: point $p \in P$ such that $\min_{p \in C} d(p, q^*)$
end
Repeat assigning points to clusters until convergence (centers do not change: $q = q^*$ )
```

```
Algorithm 2: K-Median Problem - Range-Based Initialization
Data: Set of points $P \subseteq \mathbb{R}^d$, number $k \in \mathbb{N}$
Result: Subset $Q \subseteq P$ such that $|Q| = k$
Choose k centers by selecting one random center in each set;
Set $S_i$ is constructed as: ;
$range = \max{(x_{max} - x_{min}), (y_{max} - y_{min})}$ ;
Choose x or y-range based on which is larger ;
$intervalSize = range \div k$   AND   $start = (useX? \quad x_{min} : y_{min})$ ;
$S_i = [start + (i-1) * intervalSize \quad ; \quad start + i * intervalSize]$ ;
Update centers by recalculating within clusters by: ;
for all clusters $C_q$ do
    Find median point $q^* = (\bar{x}, \bar{y})$ ;
    $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \forall i \in C_q$ ;
    $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad \forall i \in C_q$ ;
    New center: point $p \in P$ such that $\min_{p \in C} d(p, q^*)$
end
Repeat assigning points to clusters until convergence (centers do not change: $q = q^*$ )
```

Algorithm 3: K-Median with Outliers: Local Outlier Factor combined with Algorithm 1 or Algorithm 2

```
for all points $p \in P$ do
    Find k nearest neighbors of p;
    Compute reachability distance for each neighbor ;
    Calculate local reachability density of p by: ;
    $density\_p = 1/$ (average(reachability_distance_of_neighbors) + $\epsilon$)
end
for all points $p \in P$ do
    Find k-nearest neighbors of p;
    Calculate LOF of p by: ;
    $lof\_p = 0$;
    for each neighbor $n$ of $p$ do
        $lof\_p+ = density\_n \div density\_p$;
        $lof\_p/ = k$ ;
    end
end
Sort LOF values;
Remove $(number\_of\_points - m)$ outliers ;
Implement clustering algorithm 1 or algorithm 2 to find centers and radius
```

Algorithm 4: K-Median with Outliers: Binary Search and Greedy Algorithm

```
Binary search for optimal radius by: ;
$low = 0$ & $high = \max_{i,j \in 1,\ldots,n, i \neq j} d(p_i, p_j)$ ;
Compute midpoint radius $r = \frac{low+high}{2}$ ;
Test if m points can be covered with k centers each having radius r by canCoverMPoints;
Update radius bounds: if m points can be covered, $high = r$, if not $low = r$ ;
canCoverMPoints: ;
for each centers up to $k$ do
    Identify the best center c that minimizes coverage: $c = \text{argmax}_{p \in P} \mid p_i : d(p, p_i) \leq r$ and not covered [i] ;
    Mark points as covered: for selected $c$ do
        Mark all points within radius of c as covered: $covered[i] =$ true if $d(c, p_i) \leq r$
    end
end
```

References

Bâdoiu, M., & Clarkson, K. L. (2003). Smaller core-sets for balls. *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 801–802.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). Lof: Identifying density-based local outliers, 93–104. https://doi.org/10.1145/342009.335388

Charikar, M., Guha, S., Tardos, É., & Shmoys, D. B. (2002). A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, *65*(1), 129–149. https://doi.org/https://doi.org/10.1006/jcss.2002.1882

Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, *38*, 293–306. https://doi.org/https://doi.org/10.1016/0304-3975(85)90224-5

Kaufman, L., & Rousseeuw, P. (1990). Partitioning around medoids (program pam). In *Finding groups in data* (pp. 68–125). John Wiley Sons, Ltd. https://doi.org/https://doi.org/10.1002/9780470316801.ch2

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489