informatics

Article

Tree-Based Algorithm for Stable and Efficient Data Clustering

Hasan Aljabbouli [1], Abdullah Albizri [2,*] and Antoine Harfouche [3]

1 Department of Information Systems, Colorado State University Global, Salida Way, Aurora, CO 80526, USA; hasan.aljabbouli@csuglobal.edu
2 Department of Information Management & Business Analytics, Montclair State University, Montclair, NJ 07043, USA
3 Department of SEGMI–CEROS, University Paris Nanterre, 92000 Nanterre, France; antoine.h@parisnanterre.fr
* Correspondence: albizria@montclair.edu

Received: 31 August 2020; Accepted: 22 September 2020; Published: 27 September 2020

Abstract: The K-means algorithm is a well-known and widely used clustering algorithm due to its simplicity and convergence properties. However, one of the drawbacks of the algorithm is its instability. This paper presents improvements to the K-means algorithm using a K-dimensional tree (Kd-tree) data structure. The proposed Kd-tree is utilized as a data structure to enhance the choice of initial centers of the clusters and to reduce the number of the nearest neighbor searches required by the algorithm. The developed framework also includes an efficient center insertion technique leading to an incremental operation that overcomes the instability problem of the K-means algorithm. The results of the proposed algorithm were compared with those obtained from the K-means algorithm, K-medoids, and K-means++ in an experiment using six different datasets. The results demonstrated that the proposed algorithm provides superior and more stable clustering solutions.

Keywords: K-means algorithm; data clustering; Kd-tree structure

1. Introduction

Machine learning techniques have been gaining significant interest among researchers and practitioners. Data clustering, which is the division of data into groups of similar objects [1], is a popular data mining and machine learning task. Recent studies emphasize the rising potential of data clustering in useful applications of machine learning in different areas [2]. In addition, data clustering is often used as a pre-processing technique in data mining. However, the data clustering performance is a challenging problem that has been addressed in many contexts by researchers in many disciplines [3].

In general, data clustering is divided into crisp and fuzzy clustering. Crisp clustering is used to indicate a clustering process in which each data point belongs to one cluster, while fuzzy clustering indicates a clustering process in which each data point can belong to more than one cluster at the same time. The widely known K-means algorithm [4–6] is the most utilized crisp clustering algorithm [7]. This method suffers from a number of weaknesses, such as instability and the large numbers of neighborhood searches required. The stability of the K-means algorithm in reducing distortion, which is the sum of squared errors "E", was examined in prior research [8–10] as a mechanism for selecting the number of clusters.

Instability means that the K-means algorithm produces different results each time because of the randomness of the initialization stage. This randomness affects both the number of iterations needed to find a solution and the final solution itself. A reliable initialization process of the centers usually leads to a high-quality final solution. Another drawback of the K-means algorithm is the large number of neighbor searches required to find the data points within one cluster (to find data points which are close to each center).

Therefore, this paper addresses two critical problems: (1) the problem of instability of the K-means algorithm (2) the large number of nearest neighbor searches required by the K-mean algorithm. This is achieved by replacing the randomness choice of initial centers of the clusters in K-means using an efficient variant of Kd-tree structure with a new splitting mechanism of nodes. The remainder of this paper is organized as follows. First, in Section 2, we introduce the Kd-tree and summarize its usage in data clustering. Section 3 describes the proposed algorithm in detail. The results and the discussion of the findings with the summary are presented in Sections 4 and 5 respectively.

2. The Kd-Tree and Data Clustering

Developed around 1962, K-means is the most popular data clustering algorithm which is designed to create clusters that minimize the squared error between the cluster's points and the centroid (mean) of the cluster [11,12]. Improving the K-means algorithm has been the focus of many studies up until this point, producing multiple different variations and methods focusing on improving speed [13], selection of initial cluster centers [14,15], or reduction in the number of iterations [16].

To introduce our novel framework, we need to introduce the Kd-tree [17,18] which is a multi-dimensional binary tree used for organizing data points in K-dimensional space in order to decrease the time of the nearest neighbor search. The tree consists of a number of nodes and leaves where the root node contains all data points. Each node in the tree has two child nodes, while leaves are without child nodes. Data points, or references to them, are stored in leaves only.

The Kd-tree uses splitting planes that are perpendicular to one of the coordinate system axes, chosen so that it goes through one of the points in the tree. There are several variants of the Kd-tree according to the stopping criterion or the applied splitting plan.

The idea of using the Kd-tree for clustering was firstly introduced by Moore [19]. It was used for estimating the parameters of a mixture of Gaussian clusters to reduce the cost of the EM-based clustering.

To speed up the K-means algorithm and make it tractable for large datasets, the blacklisting algorithm [20,21] was proposed. This algorithm updates data points in each cluster in bulk instead of updating each data point in each cluster separately. It is based on a variant of the Kd-tree called MRKd-tree (Multiple Resolution K-dimensional tree) [22].

In 2002, the filtering algorithm [23] was introduced, the algorithm is an efficient variant of the K-means using the Kd-tree. It uses the Kd-tree as a data structure for storing multidimensional data points. Each node of this tree is represented by a box containing a set of data points. The root node is the box that contains all of the points set, while a leaf is a node that contains one point. The splitting criterion used in this algorithm depends on hyper-planes splitting orthogonally to the longest side of the node through the median coordinate of the associated points.

The filtering algorithm starts by creating the Kd-tree for the given data points. For each node in the tree, the algorithm maintains a set of candidate centers. The candidate centers for the root consist of all "k" centers. The algorithm propagates for each node candidate which is the closest to the midpoint of this node. A filtering process then starts to select the nearest center from these candidates by filtering (pruning) any center which is not closer to any part of the node than any others. If there is no nearest center for all points in the node, then this process will recur on its children.

To improve the filtering algorithm, the fast-greedy algorithm as well as the restricted filtering algorithm were proposed [24]. The former modified the boundaries of the node, and the latter added a new threshold for the direct computation of distance and for Kd-tree splitting.

Aiming to increase the number of seeds until "k" seeds are found, the global K-means algorithm [25] was proposed. This algorithm employs the Kd-tree to use the centers of "k" created buckets as seeds for the K-means algorithm. It uses a variant of the Kd-tree of a splitting criterion that splits each bucket along the direction of the maximum variance.

In 2007, Redmond and Heneghan described their method for initializing the K-means algorithm [26]. The Kd-tree is used to perform a density estimation of the data. The algorithm then chooses "k" seeds for the K-means algorithm in the highest density nodes of the Kd-tree. This method uses a variation of the Kd-tree which splits data along the median. Additionally, existing studies used the Kd-tree to improve the speed of K-means utilizing cluster center displacement [27].

In this paper, we develop an algorithm to enhance the K-means algorithm performance using an improved Kd-tree structure. The improved Kd-tree is used as a data structure to improve the choice of initial centers and to reduce the number of the nearest neighbor searches. It is also combined with an efficient center insertion technique to propose an incremental operation that overcomes the instability problem of the K-means algorithm.

3. The Proposed Algorithm

The proposed algorithm depends on the operation of an inline construction—during the execution (not pre-prepared)—for an improved variant of the Kd-tree, which is an important feature when dealing with different datasets as the algorithm can adjust the number of levels in the Kd-tree according to the required number of clusters. In this variant, the region of each node is represented by a hyper-rectangle identified by two vectors "Hmax" and "Hmin".

When creating a new node, the algorithm calculates a number of attributes for the node and uses them in the next steps to decrease the mathematical calculations required in each step. The new suggested structure includes a number of additional attributes compared to those used in the filtering and fast greedy algorithms to make the computing process faster. Each node in the proposed structure has a number of attributes. These attributes of each node need to be computed just once when the node is created.

This tree is further improved by implementing a new splitting method that is suitable for clustering (Figure 1). The splitting point is selected to be the average of the farthest two neighboring data points in that dimension, and happens to the widest dimension every time the algorithm wants to split a leaf.

The mechanism of assigning a leaf "L" to a cluster is simple (Figure 2). This mechanism is illustrated in detail in Kanungo et al. 2002. It depends on the extreme point "p" in the direction of the leaf "L" and the two clusters' centers "c1" and "c2". By calculating the distance from "p" to "c1" and "c2", it will be known which of "c1" and "c2" is closer to "L" and whether all the data points in "L" belong to the cluster of center "c1" or "c2".

The extreme point "p" can be determined as the corner of the leaf, as follows:

$$p(d) = \begin{cases} Hmax(d), & c_2(d) > c_1(d) \\ Hmin(d), & Otherwise \end{cases} \tag{1}$$

where:

$p(d)$: is the coordinate of p in dimension (d).
$c(d)$: is the coordinate of c in dimension (d).

The proposed algorithm is an incremental method and it uses an efficient initialization method. It was necessary to propose a new tree-building method of a dynamic construction in runtime.

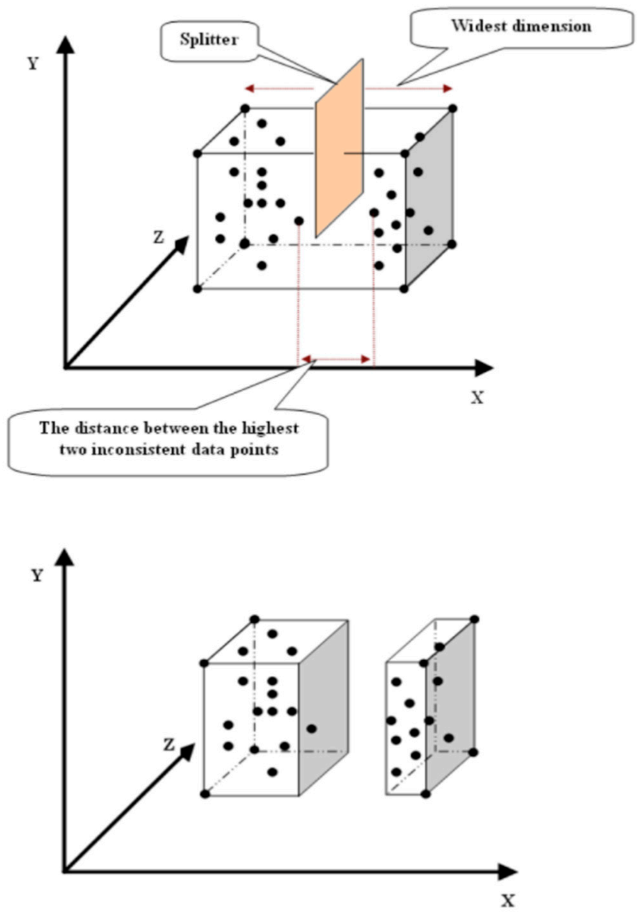

Splitter
Widest dimension
Y
Z
X
The distance between the highest two inconsistent data points
Y
Z
X

Figure 1. The proposed new splitting method for a node.

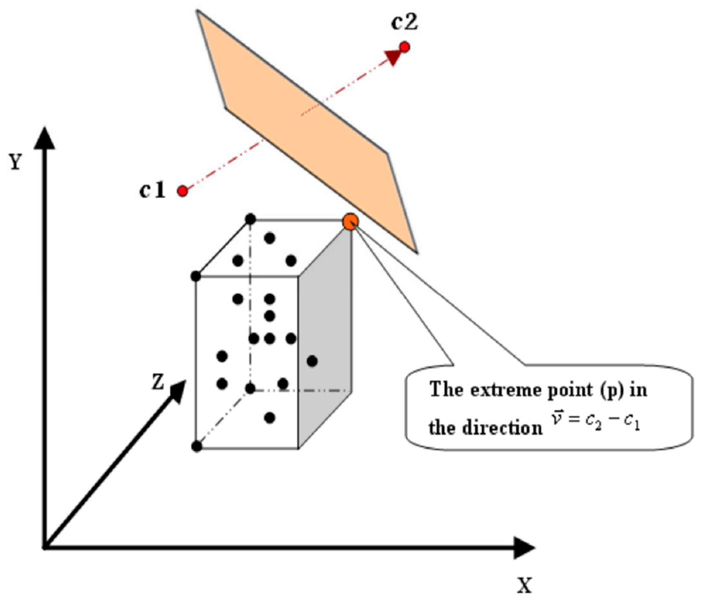
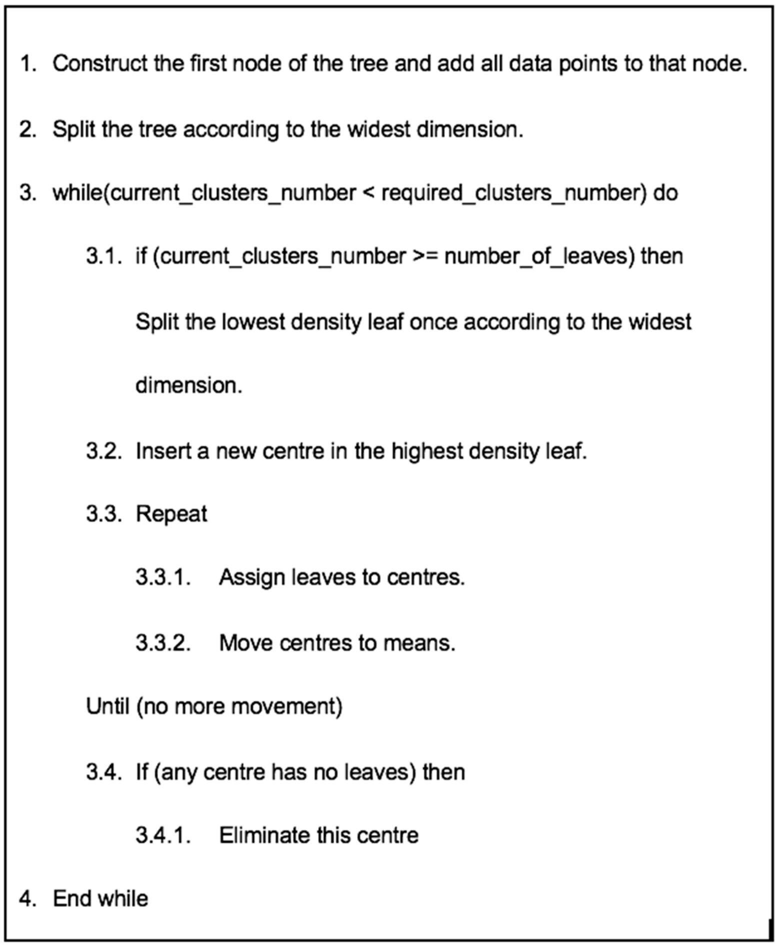

c2
c1
Y
Z
X
The extreme point (p) in the direction $\vec{v} = c_2 - c_1$

Figure 2. The mechanism of assigning a leaf to a cluster.

Unlike the filtering, restricted, and fast greedy algorithms, the proposed algorithm does not create the whole tree starting from all data points in the root node and continuing until each node has one data point (e.g., the filtering algorithm), nor creating the whole tree with nodes until a predefined threshold (a specific number of items in each node) is met (e.g., the fast-greedy algorithm). The proposed algorithm starts with one node that contains all data points. According to the required number of clusters, the algorithm decides whether to continue the splitting process for each node separately or to stop. Figure 3 illustrates the pseudo-code of the proposed algorithm.



```
1. Construct the first node of the tree and add all data points to that node.
2. Split the tree according to the widest dimension.
3. while(current_clusters_number < required_clusters_number) do
    3.1. if (current_clusters_number >= number_of_leaves) then
         Split the lowest density leaf once according to the widest
         dimension.
    3.2. Insert a new centre in the highest density leaf.
    3.3. Repeat
         3.3.1.  Assign leaves to centres.
         3.3.2.  Move centres to means.
    Until (no more movement)
    3.4. If (any centre has no leaves) then
         3.4.1.  Eliminate this centre
4. End while
```

Figure 3. The pseudo-code of the proposed algorithm.

The algorithm starts by adding references to all data points in one node which is the root of the tree–Step 1. When creating any node in the tree, the algorithm calculates the density of the node "density", the number of items "number_of_items", the total sum of all items in each dimension "total_sum", and the upper and lower bounds of the node dimensions "Hmax" and "Hmin", respectively.

Step 2: the algorithm finds the widest dimension of the whole node and splits the node into two leaves. If the widest dimension is "d", the algorithm then finds the two data points furthest apart in this dimension and calculates the mean of these two data points to be the splitting point of the node.

Step 3: while the required number of clusters is not yet met, and the current number of clusters is equal to, or more than, the current number of leaves, then the lowest density leaf is split in Step 3.1.

Step 3.1: the algorithm splits the lowest density leaf to produce more leaves to insert the new center in one of them.

Step 3.2: the algorithm inserts a new center to be the mean of the highest density leaf of the tree.

Step 3.3: the main steps of the K-means algorithm are applied using the created tree.

The algorithm assigns all leaves to an appropriate center and then each center is moved to be the mean of all points of the leaves which belong to it.

Step 3.4: If any center is inserted with no leaves belonging to it, the algorithm eliminates that center.

4. Experiments

The algorithm was applied to three artificial datasets: data1, data2 and data3. It was also applied to three real datasets: Iris [28], Crude Oil [29], and Vowel [30] The number of features, instances and clusters of each dataset is provided in Table 1.

Table 1. Dataset Description.

Dataset	Data1	Data2	Data3	Iris	Crude Oil	Vowel
#Features	2	2	10	4	5	3
#Instances	76	900	1000	150	56	871
#Classes	3	9	2	3	3	6

The results of the proposed algorithm were compared to those of the K-means algorithm, K-medoid, and K-means++ (other variants of the K-means algorithm). The clustering criterion "*E*" (Equation (2)) was used to evaluate the performance of the algorithms. The smaller the value of this metric, the better the clustering results. All algorithms were executed many times (100 times). The average, minimum and maximum values of "*E*" were noted.

$$E = \sum_{j=1}^{k} \sum_{i=1}^{n_j} ||x_i^{(j)} - c_j||^2 \tag{2}$$

where:

$$||x_i^{(j)} - c_j||^2$$

is a chosen distance measure between a data point $x_i{}^{(j)}$ in the cluster$^{(j)}$ and the cluster center c_j. n_j is the number of data points in the cluster c_j. and k is the number of clusters.

The value of the distortion of the proposed algorithm was compared to that of the K-means, K-medoids, and K-means++ algorithms in Table 2. As the proposed algorithm is also intended to reduce the number of nearest neighbor searches, Table 3 lists the number of distance calculations between data points and cluster centers in each dataset for both the K-means and the proposed algorithms:

Table 2. Clustering results on the used datasets.

Data/Algorithm	K-Means	K-Medoids	K-Means++	Proposed Algorithm
Data 1	55.1581 ± 6.9611	54.3962 ± 6.4543	49.1530 ± 4.3717	**47.6100 ± 0.0000**
Data 2	615.0904 ± 26.8660	640.7589 ± 32.8690	**597.9887 ± 17.2748**	971.2000 ± 0.0000
Data 3	**863.2801 ± 1.5688**	945.5977 ± 11.2057	863.7684 ± 1.4762	864.6000 ± 0.0000
Iris	104.3849 ± 11.7980	103.8354 ± 10.4880	**98.0069 ± 4.4933**	98.6399 ± 0.0000
Crude oil	279.6143 ± 0.1867	305.8225 ± 22.1917	279.5339 ± 0.1989	**278.9000 ± 0.0000**
Vowel	153,468.1732 ± 4163.8986	158,295.8453 ± 8676.5586	151,960.0763 ± 3323.1807	**151,900.0000 ± 0.0000**

The best performing algorithm is bolded for each dataset.

Table 3. Number of nearest neighbor (ngh) queries in the algorithms for the used datasets.

Data/Algorithm	K-Means	Proposed Algorithm
Data 1	1368	**228**
Data 2	24,300	**9000**
Data 3	6000	**2000**
Iris	1800	**450**
Crude oil	1176	**168**
Vowel	20,904	**5226**

The least number of ngh queries is bolded for each dataset.

As evident in the results of the experiments, the proposed clustering algorithm gives stable results in all cases and outperforms the K-means, K-medoid, and K-means++ in most cases. It yields a lower value of the distortion "E", which means better clustering solutions in most cases. The stability of the proposed algorithm can be seen in Figures 4–9. The algorithm gives the same value of the distortion in 100 different runs, which means overcoming the problem of instability.

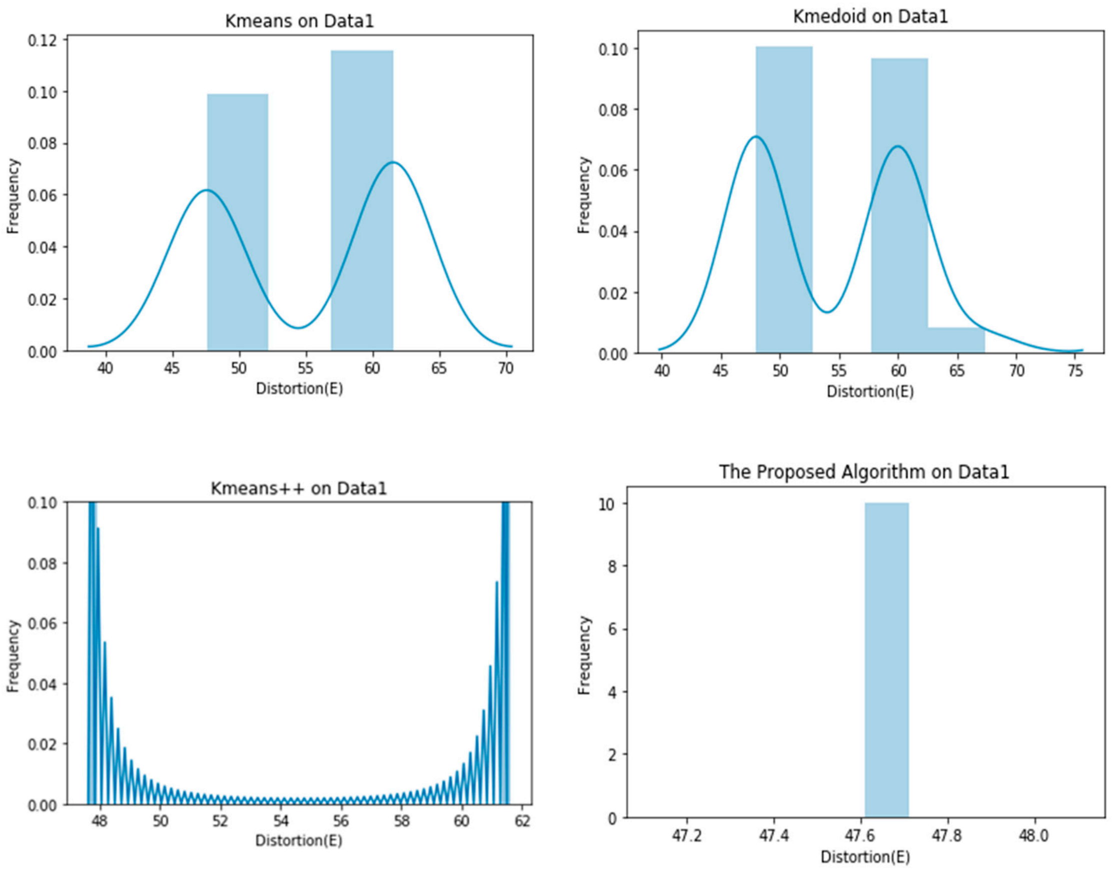


Figure 4. Graphical representation of E obtained by 100 runs of different algorithms on Data1.

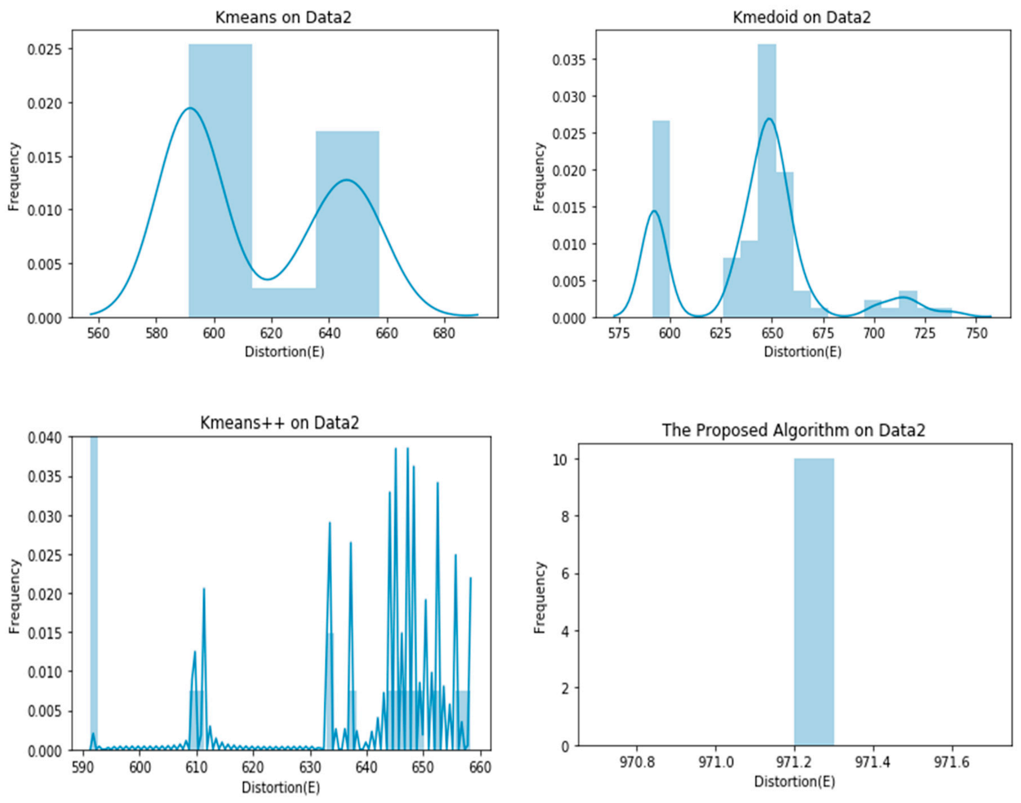


Figure 5. Graphical representation of E obtained by 100 runs of different algorithms on Data2.

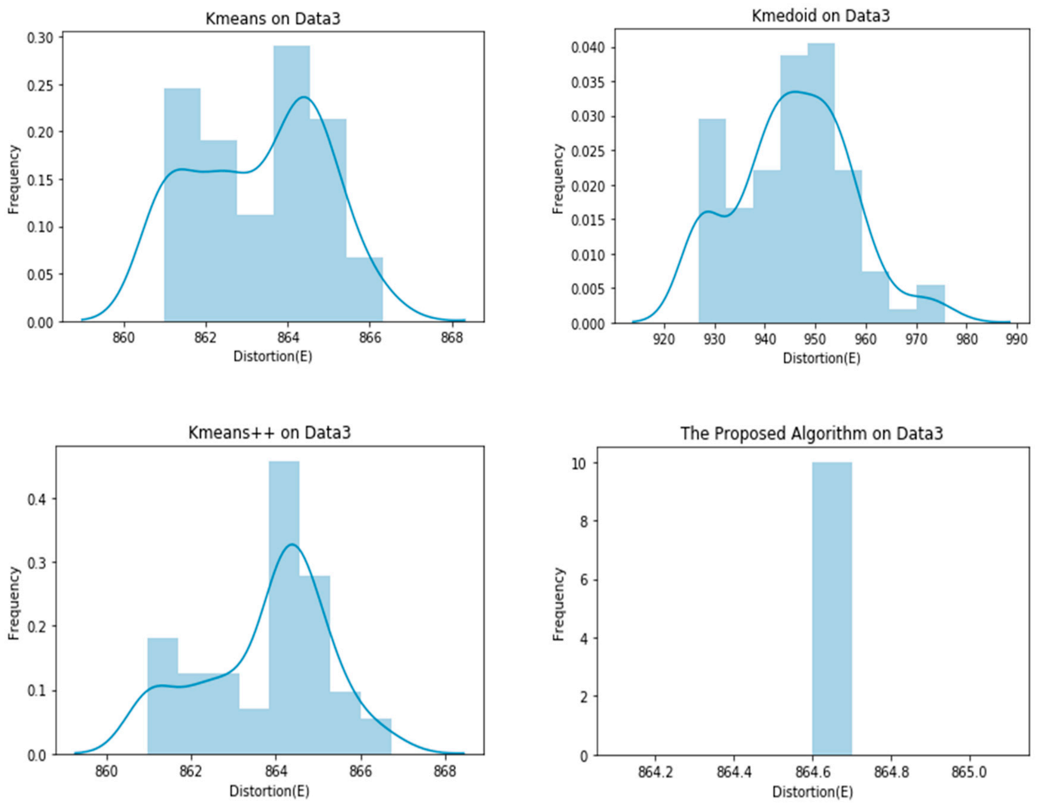


Figure 6. Graphical representation of E obtained by 100 runs of different algorithms on Data3.

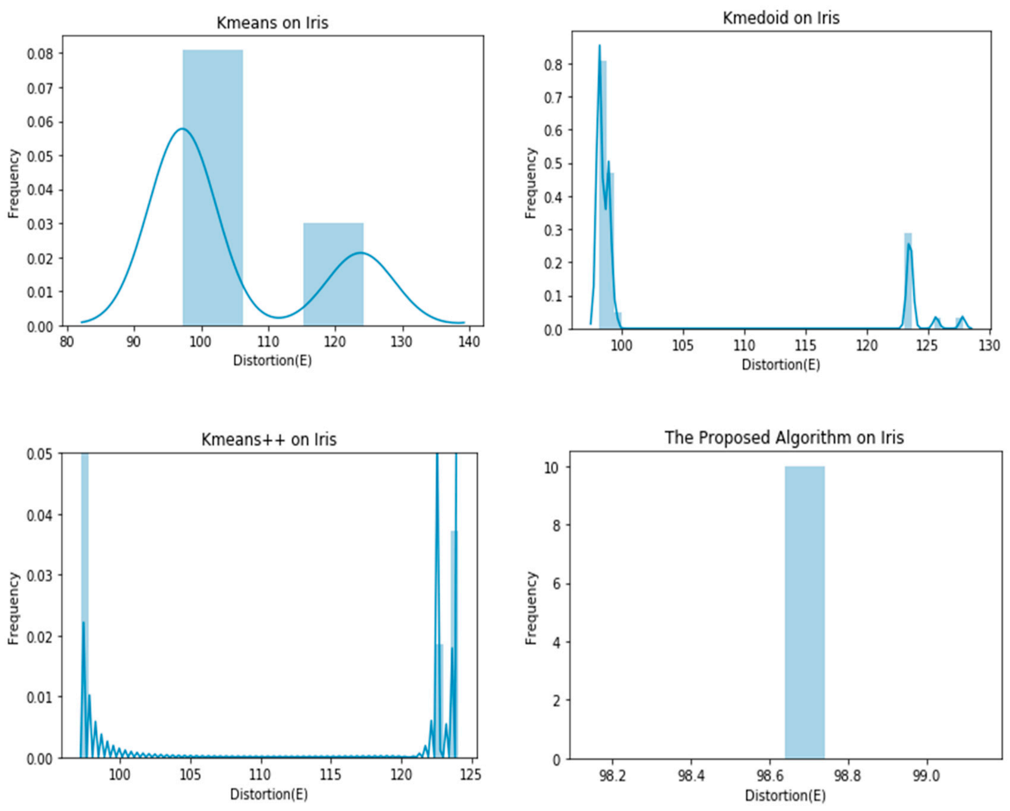


Figure 7. Graphical representation of E obtained by 100 runs of different algorithms on Iris dataset.

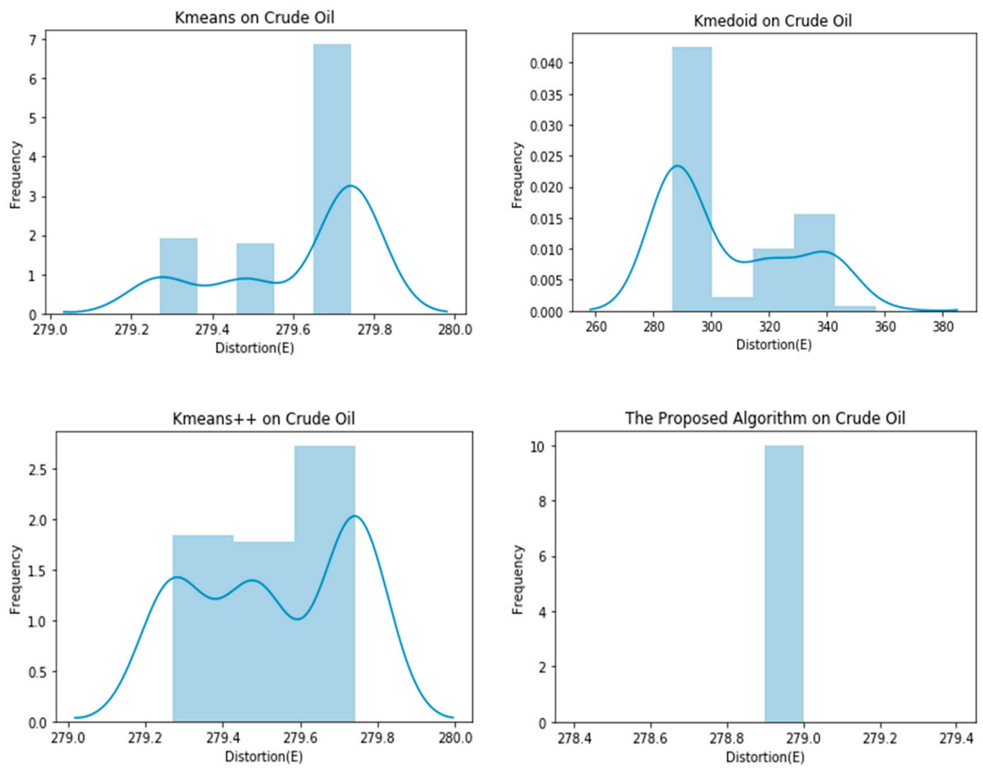


Figure 8. Graphical representation of E obtained by 100 runs of different algorithms on the Crude Oil dataset.

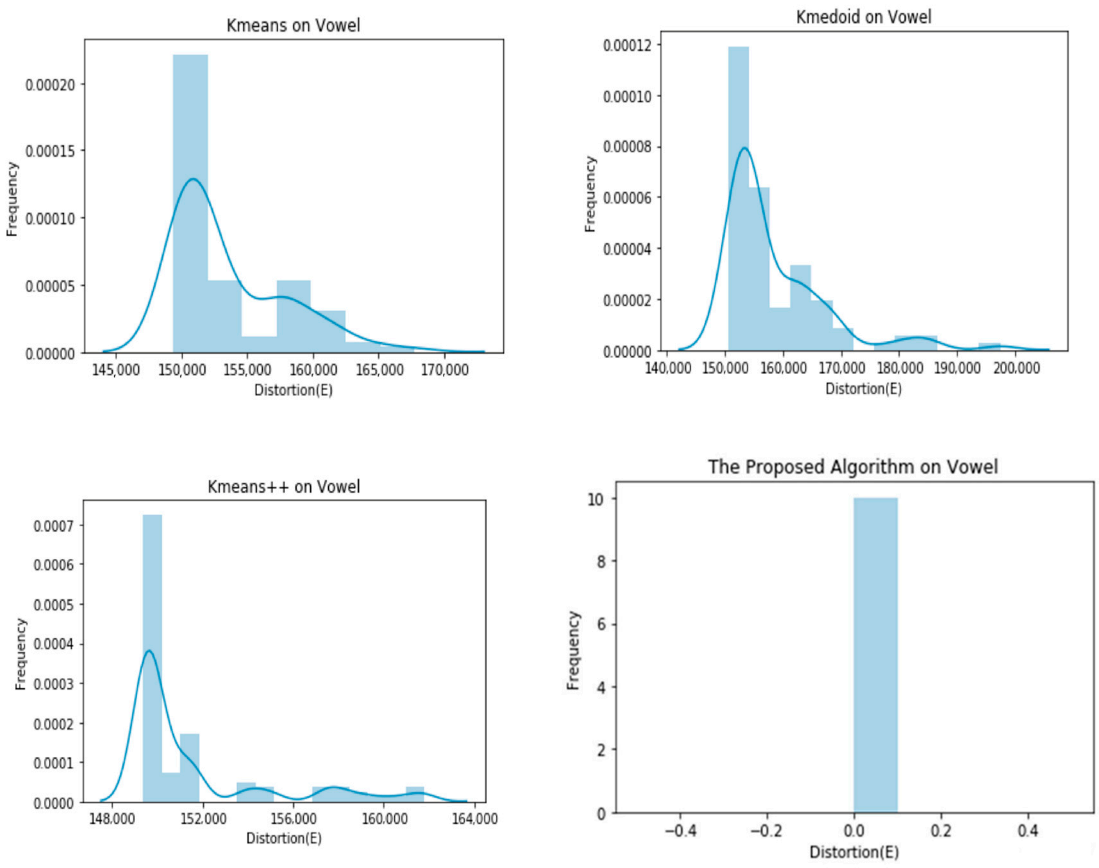


Figure 9. Graphical representation of E obtained by 100 runs of different algorithms on the Vowel dataset.

5. Discussion

The results of the experiments presented several major findings. First, the proposed algorithm suggests a significant improvement vis-à-vis the original K-means clustering in terms of the dynamic nature of the number of leaves related to the clustering process. The proposed algorithm mechanism of tree construction reduces space complexity drastically. For example, if the average height of the Kd-tree reduces by just one unit, which happens due to the incremental mechanism of the algorithm, the number of nodes to be expanded is reduced to almost half. Additionally, the “number of leaves” is invariably less than the “number of points”, and only in the worst case are they equal to each other. As a result of this, it is clear that the usage of the proposed Kd-tree structure reduces the time needed to assign data points to clusters. The complexity of the proposed algorithm is affected by the number of clusters. The higher the number of clusters, the greater is the time the algorithm takes. In addition, the quality of the solution is generally superior in most cases.

Second, another major advantage of the proposed algorithm is that it can deal with large datasets (large numbers of data points) due to its ability to deal with data points in bulks.

Third, compared to other Kd-tree-based clustering algorithms, the proposed Kd-tree structure has an inline building method. This reduces the complexity of the algorithm considerably by not running the splitting process and the K-means steps on all leaves of the pre-built tree (as in the blacklisting, filtering, and fast greedy algorithms); the proposed algorithm runs the steps on a limited number of leaves of the dynamic tree only. In other words, in the blacklisting and filtering algorithms, a whole Kd-tree is built before running the main algorithm steps. This means a longer running time and higher complexity for the algorithm.

Fourth, in the fast-greedy algorithm, despite the improvement of the tree-building process by using a predefined threshold to stop the Kd-tree splitting, it is still not a fully inline building method.

This is due to that the tree must be built before running the main algorithm and it is difficult to specify the stopping threshold if there is no previous knowledge about the dataset.

Fifth, another major finding is that the proposed algorithm reduces the complexity considerably. This is due to the fact that the proposed algorithm does not need to find a set of points that could act as an appropriate set for the insertion of a new center and then assigning leaves to centers, which is in contrast to the fast-greedy algorithm. The proposed algorithm leads to inserting the new center directly to be the mean of the highest density leaf, where a better new center can be added, as has already been proved in Kanungo et al. [22].

6. Conclusions

In conclusion, this paper presented an algorithm to improve the performance of the K-means clustering algorithm by discussing a new Kd-tree-based clustering method. The proposed algorithm is used to overcome the instability and the large number of nearest neighbor searches of the K-means algorithm. The proposed algorithm was compared with the K-means algorithm, K-medoids, and K-means++ in an experiment using six different datasets. The results demonstrated that the proposed algorithm gives superior and more stable clustering solutions.

Another major contribution presented is that the proposed Kd-tree structure improved the process needed to assign data points to clusters which causes the number of neighborhood searches to decrease. Future research could investigate the frameworks to reduce the number of parameters needed to be configured for the proposed algorithm considering automatic detection of the number of clusters.

Author Contributions: Conceptualization, H.A., A.A. and A.H.; Data curation, H.A.; Formal analysis, H.A. and A.A.; V.A.A.; V.H.A.; Writing—original draft, H.A.; Writing—review & editing, A.A. and A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Berkhin, P. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*; Springer: Berlin, Germany, 2006; pp. 25–71.
2. Abdullah, S.S.; Rostamzadeh, N.; Sedig, K.; Garg, A.X.; McArthur, E. Visual Analytics for Dimension Reduction and Cluster Analysis of High Dimensional Electronic Health Records. *Informatics* **2020**, *7*, 17. [CrossRef]
3. Jones, P.J.; James, M.K.; Davies, M.J.; Khunti, K.; Catt, M.; Yates, T.; Rowlands, A.V.; Mirkes, E.M. FilterK: A new outlier detection method for k-means clustering of physical activity. *J. Biomed. Inform.* **2020**, *104*, 103397. [CrossRef] [PubMed]
4. Jain, A.K.; Dubes, R.C. *Algorithms for Clustering Data*; Prentice Hall: Englewood Cliffs, NJ, USA, 1988; p. 304.
5. Jain, A.K.; Murty, M.N.; Flynn, P.J. Data Clustering: A Review. *ACM Comput. Surv.* **1999**, *31*, 264–323. [CrossRef]
6. MacQueen, J.B. Some Methods for Classification and Analysis of Multivariate Observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Berkeley, CA, USA, 1967; pp. 281–297.
7. Dobbins, C.; Rawassizadeh, R. Towards Clustering of Mobile and Smartwatch Accelerometer Data for Physical Activity Recognition. *Informatics* **2018**, *5*, 29. [CrossRef]
8. Kuncheva, L.I.; Vetrov, D.P. Evaluation of stability of k-means cluster ensembles with respect to random initialization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1798–1808. [CrossRef]
9. Rakhlin, A.; Caponnetto, A. Stability of K-means clustering. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 4–7 December 2016; City University of Hong Kong: Hong Kong, China, 2007; pp. 1121–1128.
10. Steinley, D. Stability analysis in K-means clustering. *Br. J. Math. Stat. Psychol.* **2008**, *61*, 255–273. [CrossRef]

11. Steinley, D. K-means Clustering: A Half-Century Synthesis. *Br. J. Math. Stat. Psychol.* **2007**, *59*, 1–34. [CrossRef] [PubMed]
12. Zhao, H.; Ram, S. Clustering Schema Elements for Semantic Integration of Heterogeneous Data Sources. *J. Database Manag.* **2004**, *15*, 88–106. [CrossRef]
13. Zhu, E.; Zhang, Y.; Wen, P.; Liu, F. Fast and stable clustering analysis based on Grid-mapping K-means algorithm and new clustering validity index. *Neurocomputing* **2019**, *363*, 149–170. [CrossRef]
14. Khan, S.S.; Ahmad, A. Cluster Center Initialization Algorithm for K-means Clustering. *Pattern Recognit. Lett.* **2004**, *25*, 1293–1302. [CrossRef]
15. Xu, J.; Xu, B.; Zhang, W.; Zhang, W.; Hou, J. Stable initialization scheme for k-means clustering. *Wuhan Univ. J. Nat. Sci.* **2009**, *14*, 24–28. [CrossRef]
16. Arora, P.; Virmani, D.; Jindal, H.; Sharma, M. Sorted K-means towards the enhancement of K-means to form stable clusters. In Proceedings of the International Conference on Communication and Networks, Ahmedabad, India, 19–20 February 2016; Spring: Singapore, 2017; pp. 479–486.
17. Bentley, J.L. Multidimensional Divide and Conquer. *Commun. ACM* **1980**, *23*, 214–229. [CrossRef]
18. Friedman, J.H.; Bentley, J.L.; Finkel, R.A. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.* **1977**, *2*, 209–226. [CrossRef]
19. Moore, A. Very Fast EM-Based Mixture Model Clustering Using Multiresolution Kd-trees. In *Advances in Neural Information Processing Systems II (NIPS)*; MIT Press: Breckenridge, CO, USA, 1999; pp. 543–549.
20. Pelleg, D.; Moore, A. Accelerating Exact K-means Algorithms with Geometric Reasoning. In Proceedings of the 5th ACM International Conference of the Special Interest Group on Knowledge Discovery and Data Mining (ACM-SIGKDD-99), San Diego, CA, USA, 15–18 August 1999; pp. 277–281.
21. Pelleg, D.; Moore, A. *Accelerating Exact K-Means Algorithms with Geometric Reasoning-Technical Report*; School of Computer Science, Carnegie Mellon University: Pittsburgh, PA, USA, 2000.
22. Moore, A.; Lee, M.S. Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets. *J. Artif. Intell. Res.* **1998**, *8*, 67–91. [CrossRef]
23. Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y. An Efficient K-means Clustering Algorithm: Analysis and Implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 881–892. [CrossRef]
24. Hussein, N. A Fast Greedy K-Means Algorithm. Master's Thesis, University of Amsterdam, Amsterdam, The Netherlands, 2002.
25. Likas, A.; Vlassis, N.; Verbeek, J.J. The Global K-means Clustering Algorithm. *Pattern Recognit.* **2003**, *36*, 451–461. [CrossRef]
26. Redmond, S.J.; Heneghan, C. A Method for Initialising the K-means Clustering Algorithm Using Kd-Trees. *Pattern Recognit. Lett.* **2007**, *28*, 965–973. [CrossRef]
27. Lai, J.Z.; Huang, T.J.; Liaw, Y.C. A fast k-means clustering algorithm using cluster center displacement. *Pattern Recognit.* **2009**, *42*, 2551–2556. [CrossRef]
28. Asuncion, A.; Newman, D.J. UCI Machine Learning Repository. University of California, School of Information and Computer Science. 2007. Available online: http://www.ics.uci.edu/~{}mlearn/MLRepository.html (accessed on 15 January 2020).
29. Johnson, R.A.; Wichern, D.W. *Applied Multivariate Statistical Analysis*, 5th ed.; Prentice Hall: Englewood Clifs, NJ, USA, 2001; p. 767.
30. Grabmeier, J.; Rudolph, A. Techniques of Cluster Algorithms in Data Mining. *Data Min. Knowl. Discov.* **2002**, *6*, 303–360. [CrossRef]