Recursive Density-Based Refinement of K-Means for Automatic Cluster Splitting

Md. Mahir Uddin
ID: 0122430022
Computer Science and Engineering
United International University, Dhaka, Bangladesh
Email: muddin2430022@mscse.uiu.ac.bd

Abstract- **K-means clustering is one of the most widely employed algorithms for unsupervised learning. However, its performance heavily depends on the user-defined number of clusters (k), which can lead to suboptimal clustering results when misestimated. This paper proposes a novel modification to k-means by introducing a **density-driven recursive splitting mechanism** that automatically detects and corrects clusters that have been erroneously merged. Starting with an arbitrary value of k, our method iteratively refines the clusters by detecting low-density regions, identifying potential cluster splits, and recursively applying k-means with k=2 to further partition these clusters. The method is robust to irregularly shaped and density-varying clusters, removing the need for prior knowledge of the ideal number of clusters. Experimental results on real-world datasets demonstrate that the approach provides more accurate and adaptive cluster assignments compared to traditional k-means, even when the initial k is set to 1. This method is particularly useful for applications where true cluster structure is unknown and dynamic adjustments are critical.**

Keywords: K-means clustering, density-based clustering, recursive clustering, unsupervised learning.

I. INTRODUCTION

Clustering is a fundamental technique in unsupervised learning, widely used in various domains such as image processing, bioinformatics, anomaly detection, and market segmentation. Among clustering methods, k-means is one of the most widely adopted due to its simplicity, computational efficiency, and scalability [1]. The algorithm partitions a dataset into a predefined number of clusters, where each point is assigned to the nearest cluster centroid. Through an iterative refinement process, k-means minimizes intra-cluster variance, producing compact and well-separated clusters in many scenarios.

Despite its widespread use, k-means clustering has several inherent limitations that can impact its effectiveness, particularly in complex real-world datasets. A major drawback of k-means is its dependence on a predefined number of clusters (k), which must be specified before execution [2]. Selecting an inappropriate k value can lead to suboptimal clustering, where distinct clusters are merged or a single cluster is unnecessarily split [3]. Heuristic methods such as the Elbow Method [4] and Silhouette Score [5] attempt to estimate the optimal k, but these approaches do not guarantee the best possible clustering for all datasets. Furthermore, k-means assumes that clusters are convex and roughly spherical [6], making it ineffective for datasets containing irregularly shaped or density-varying clusters.

Another major challenge of k-means is its sensitivity to centroid initialization. Poorly initialized centroids can result in convergence to local optima, leading to unstable clustering results [7]. While initialization techniques such as k-means++ [8] mitigate this issue to some extent, they do not fully resolve the problem. Additionally, k-means is susceptible to outliers, which can disproportionately influence centroid positions and distort cluster assignments [9]. More critically, k-means lacks an inherent mechanism to detect when two distinct clusters have been mistakenly merged into one due to centroid placement or density distribution [10].

A critical unresolved challenge in k-means clustering is its inability to detect and correct clusters that are erroneously merged due to overlapping densities or initialization artifacts. To address this, we propose a density-aware splitting mechanism that enhances k-means by automatically identifying merged clusters and recursively partitioning them into sub-clusters without requiring additional user-defined parameters.

Our method operates as follows: after initial k-means clustering, each cluster is analyzed by dividing its bounding box into a grid of squares, where the grid resolution and density thresholds are derived dynamically from the cluster's spatial distribution. Low-density regions are mapped as "chains" spanning the cluster, indicating potential splits. If such chains are detected, the cluster is subdivided using k-means with k=2, and the process repeats recursively until no further splits are warranted.

The key advantages of our approach are threefold. First, it eliminates manual parameter tuning by inferring thresholds directly from data. Second, it retains the computational simplicity of k-means while enabling detection of non-spherical or density-varying clusters. Third, it operates as a post hoc refinement layer, making it compatible with existing k-means variants (e.g., k-means++). By bridging the gap between parametric efficiency and density-based robustness, our method extends k-means' applicability to complex real-world datasets.

II. LITERATURE REVIEW

Researchers have long sought to address the limitations of k-means, particularly its dependence on predefined cluster counts and its inability to handle complex cluster geometries. Early approaches focused on improving initialization robustness. For instance, Arthur and Vassilvitskii introduced k-means++ (2007), which probabilistically seeds initial centroids to reduce sensitivity to poor initialization, though it still requires an explicit k value [11]. To automate cluster count selection, heuristic methods like the Elbow Method [12] and Silhouette Analysis [13] were proposed. These techniques iteratively evaluate clustering quality across candidate k values but struggle with datasets where clusters vary significantly in density or shape.

For non-convex or density-varying clusters, density-based methods like DBSCAN (Ester et al., 1996) and OPTICS (Ankerst et al., 1999) emerged as alternatives. These algorithms identify clusters based on local point density, enabling detection of arbitrarily shaped structures. However, they lack the computational efficiency of k-means and often require tuning density thresholds, limiting their scalability [14],[15].

Hierarchical extensions to k-means, such as bisecting k-means (Steinbach et al., 2000), recursively partition clusters to refine results. While this approach mitigates initialization sensitivity, it inherits k-means' bias toward spherical clusters and struggles with overlapping distributions [16]. More recently, Gaussian Mixture Models (GMMs) (Reynolds, 2009) and spectral clustering (Ng et al., 2002) have been used to model complex cluster geometries. However, GMMs assume parametric distributions, and spectral clustering's reliance on graph Laplacians introduces computational overhead [17],[18].

Efforts to dynamically determine k include x-means (Pelleg and Moore, 2000), which uses Bayesian Information Criterion (BIC) to split clusters during runtime, and G-means (Hamerly and Elkan, 2003), which employs statistical tests to validate cluster normality. While these methods reduce dependency on predefined k, they often fail to detect merged clusters caused by subtle density variations or irregular boundaries [19],[20]. Similarly, dip-means (Kalogeratos and Likas, 2012) uses multimodality tests to split clusters but requires user-defined significance thresholds [21].

Recent advances in deep learning, such as deep embedded clustering (DEC) (Xie et al., 2016), combine autoencoders with k-means to learn latent representations for improved clustering. While DEC achieves state-of-the-art results on high-dimensional data, it introduces complexity and training instability, limiting its applicability to resource-constrained environments [22].

Despite these advancements, no method fully resolves the challenge of post hoc detection and correction of merged clusters in k-means. Most approaches either require prior assumptions about cluster geometry, sacrifice computational efficiency, or lack interpretability. This gap motivates our work.

III. Methodology

The proposed method refines traditional k-means clustering by iteratively detecting and splitting erroneously merged clusters using density-based analysis. Unlike conventional k-means, which relies on a predefined number of clusters, this approach dynamically adjusts k by analyzing the density distribution within clusters. Through a recursive process, the method systematically identifies and separates incorrectly merged clusters without requiring user-defined parameters. The complete workflow consists of five key stages, detailed below.

A. Dataset Preparation

For the experiments, we generated a synthetic two dimensional dataset using the make_blobs function from Scikit-learn. The dataset consists of 4,500 data points distributed across 8 clusters, with each cluster following a Gaussian distribution. The standard deviation of clusters was set to 0.60, ensuring moderate overlap between clusters. A fixed random seed **(**random_state=46**)** was used to maintain reproducibility.

The dataset was designed to challenge traditional k-means clustering by including closely positioned clusters that may be erroneously merged when an incorrect number of clusters (**k**) is chosen. This allows a thorough evaluation of the proposed density-aware splitting mechanism in resolving such clustering errors.

B. Initial Clustering

The process begins by applying k-means clustering with an initial k, preferably chosen lower than the actual number of clusters. The dataset is partitioned into k clusters, each represented by a centroid and a set of assigned data points. For this experiment, value for k is taken 1,3 and 5.

C. Outlier Removal and Bounding Box Construction

To focus on the core structure of each cluster, outliers and edge points are removed before further analysis. Outliers and edge points are defined as points that deviate significantly from the cluster center. Specifically, any point located beyond 1.5σ from the cluster centroid—where σ is the standard deviation of distances from the centroid—is considered an outlier or edge points and discarded.

Once outliers are removed, a bounding box is constructed around the remaining cluster points. This bounding box is defined by the minimum and maximum coordinates in both dimensions:

(x_min, y_min), (x_min, y_max), (x_max, y_min), (x_max ,y_max)

D. Density Matrix Generation

Once the bounding box is established, it is divided into a grid of small squares for density analysis. The side length of each square is set to 5% of the smallest bounding box dimension:

$$\text{Grid Size}=0.05\times\min(x\_max-x\_min, y\_max-y\_min) \quad (1)$$

For each square in the grid, the density is calculated based on the number of points it contains. A density threshold is defined as 50% of the cluster's average density, given by:

$$\text{Threshold}= 0.75 \text{ x } \frac{\text{Bounding Box Area}}{\text{Total Points}} \text{ x Square Area} \quad (2)$$

Using this threshold, each square is labeled as high-density (1) if its point count meets or exceeds the threshold, and low-density (0) otherwise. The resulting binary density matrix provides a structured representation of the cluster's spatial distribution, allowing for efficient identification of low-density regions.

E. Low-Density Chain Detection

Since edge artifacts can influence density calculations, the outermost 15% of rows and columns are trimmed from the density matrix. This ensures that the analysis focuses on the core region of the cluster, reducing boundary effects.

The remaining density matrix is then analyzed for low-density chains—continuous sequences of connected low-density squares (0s) that indicate potential cluster boundaries. These chains are detected using 8-directional breadth-first search (BFS) to find connected low-density paths. A cluster is flagged for splitting if a low-density chain satisfies either of the following conditions:

1. Top-to-bottom chain: A continuous low-density path spans from the top edge to the bottom edge of the trimmed density matrix.
2. Left-to-right chain: A continuous low-density path spans from the left edge to the right edge of the trimmed density matrix.

The presence of such chains suggests that the cluster actually contains multiple sub-clusters, requiring further subdivision.

F. Recursive Cluster Splitting

If a low-density chain is detected, the cluster undergoes further subdivision using k-means with k=2. The original cluster is split into two sub-clusters, each with its own centroid and assigned data points. The two resulting sub-clusters replace the original cluster, and both are independently analyzed from Step B onward.

This process—outlier removal, density matrix generation, and low-density chain detection—recursively repeats for each newly formed sub-cluster. If a sub-cluster is found to contain additional low-density chains, it is further subdivided. This iterative refinement continues until no further splits are necessary, meaning all clusters are free of incorrectly merged sub-clusters.

G. Silhouette Score Calculation

The silhouette score was used to assess the quality of the clustering results. It measures how similar each point is to its own cluster compared to other clusters, with a higher score indicating better-defined clusters. The silhouette score was calculated for the clustering results obtained from both k-means and the subsequent split-cluster method, applied to the results after clustering with various values of k.

The formula for silhouette score s(i) for each point is given by:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i),\ b(i))} \tag{3}$$

Where a(i) is the average distance from point i to all other points in the same cluster and b(i) is the minimum average distance from point i to all points in any other cluster.

H. Computation Time

The computational time required for both k-means clustering and the split-cluster method was measured to understand the scalability of the algorithm, especially when applied to larger datasets. The time taken for k-means clustering was recorded for different values of k, including k=1, 3, 5, and 8, providing insight into how the choice of k impacts the computational burden. After performing k-means clustering, the time required to execute the split-cluster method was also recorded for each value of k, illustrating how the efficiency of the algorithm changes with the initial clustering. The total computational time for both k-means and the subsequent split-cluster method was evaluated, highlighting how time complexity evolves as k increases. The recorded times were then visualized in a plot to compare the execution time of the two methods, enabling an analysis of the trade-off between clustering quality and computational cost. This evaluation emphasizes the performance of the proposed method in practical applications.

The proposed method provides an adaptive clustering solution that eliminates the need for a predefined k. Even if k is initially set as low as 1, meaning that all points are treated as a single cluster, the algorithm automatically determines the correct number of clusters through recursive splitting. This approach is particularly effective in cases where clusters have irregular shapes or varying densities, as it dynamically adjusts to the underlying data distribution.

IV. Results and Discussion

To evaluate the effectiveness of our density-aware splitting mechanism, we conducted experiments on synthetic datasets. The primary objective was to validate the algorithm's ability to detect and resolve erroneously merged clusters, even when initialized with an incorrect number of clusters (k). We compared our method against traditional k-means clustering to demonstrate its superiority in recovering true cluster structures.

To better understand the structure of the dataset used in our experiments, we visualize the generated data points in Fig. 1. The dataset consists of eight well-separated clusters, providing a suitable test case for evaluating the effectiveness of our proposed clustering refinement method. The visualization highlights the initial cluster distribution before applying any clustering algorithms.

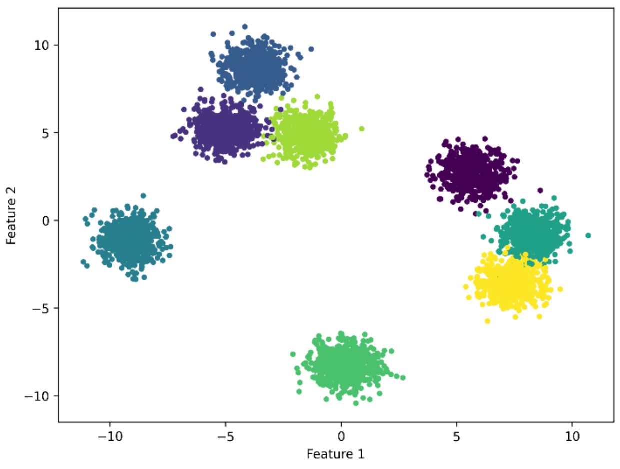

Fig. 1. Synthetic dataset based on actual cluster

To illustrate the limitations of traditional k-means clustering, we applied the algorithm to our synthetic dataset with k=1,3, and 5. These values were chosen to demonstrate how the choice of k impacts clustering results, particularly in cases where the true number of clusters is unknown. Fig. 2., Fig. 3., and Fig. 4. show the clustering outcomes for k=1,3, and 5, respectively. As expected, k-means struggles to recover the true cluster structures when k is mis-specified, by merging distinct clusters. These results highlight the need for a mechanism to automatically detect and resolve merged clusters, as proposed in our method. If k is set higher than the actual number of clusters, it may lead to excessive fragmentation. Since our method focuses on splitting rather than merging, we consider only k values less than or equal to the actual number of clusters in this experiment.

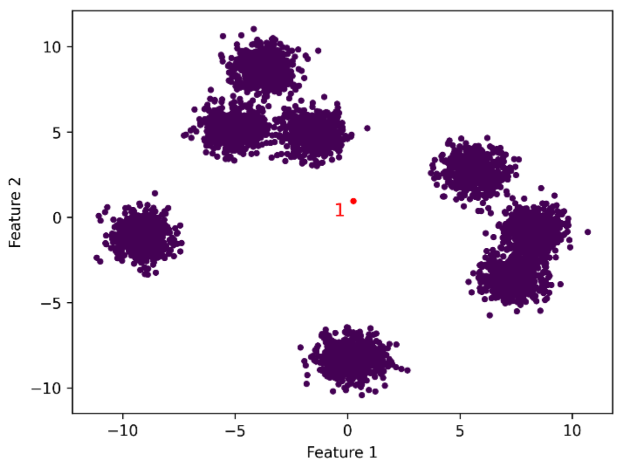

Fig. 2. K-means clustering with k=1

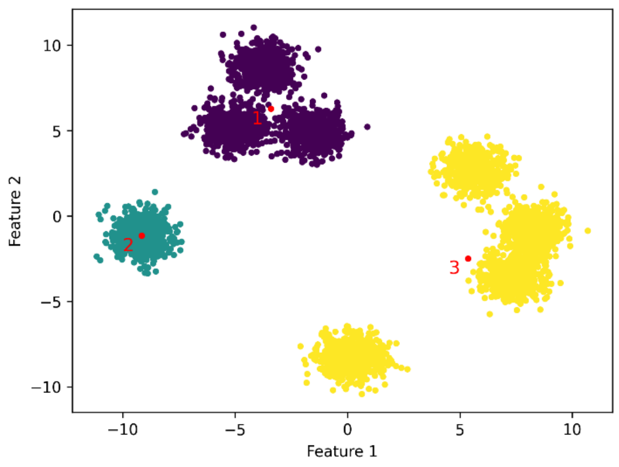

Fig. 3. K-means clustering with k=3

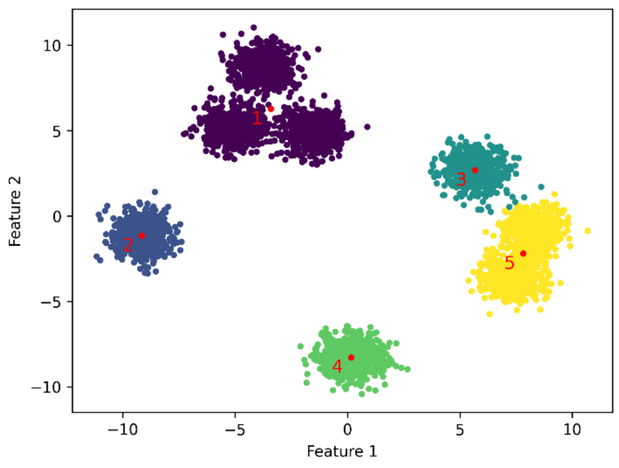

Fig. 4. K-means clustering with k=5

To demonstrate the robustness of our density-aware splitting mechanism, we applied the algorithm to the output generated by k-means with different values of k (1, 3, and 5). In all cases, the final clustering results were identical, so only a single representative figure is shown in Fig. 5.

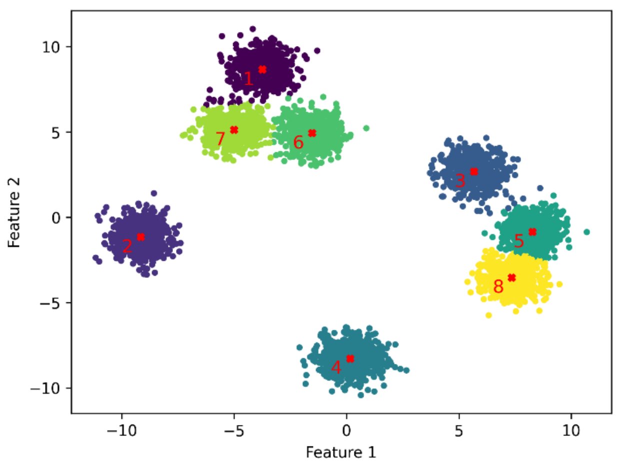

Fig. 5. Clustering result after applying split cluster method

To assess the quality of the clustering results, we calculated the silhouette scores for various values of k (excluding k=1) for both k-means clustering and after applying the split cluster method. The silhouette score measures how well-separated the clusters are by comparing the similarity of points within the same cluster to those in different clusters. Note that the silhouette score is not calculated for k=1, as at least two clusters are required to compute the score. After applying the density-aware split clustering algorithm, we evaluated the scores for all resulting clusters to gauge its effectiveness in improving clustering accuracy. The following bar chart in Fig. 6. illustrates the silhouette scores for different values of k in k-means clustering, alongside the scores achieved after the split cluster method, highlighting the comparison in clustering quality between the two approaches.

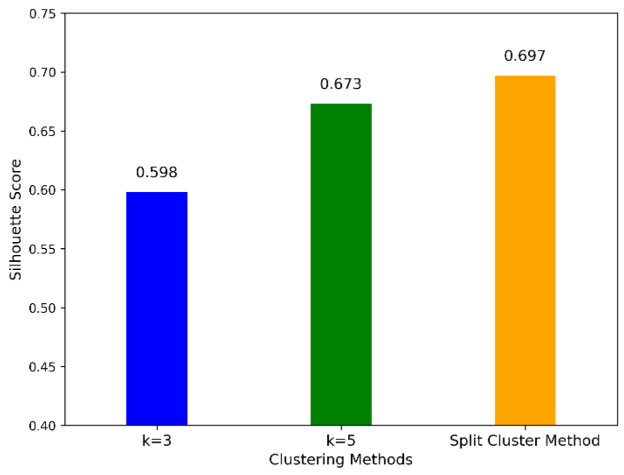

Fig. 6. Silhouette score at different stages

To assess the computational efficiency of both the k-means clustering and the split-cluster method, we measured the execution time for various values of k. The following figure shows the computational time for each clustering process (k-means with different k values) and the corresponding time for applying the split-cluster method after each k-means clustering. The comparison highlights the additional time required for the density-aware splitting step after the initial clustering phase.

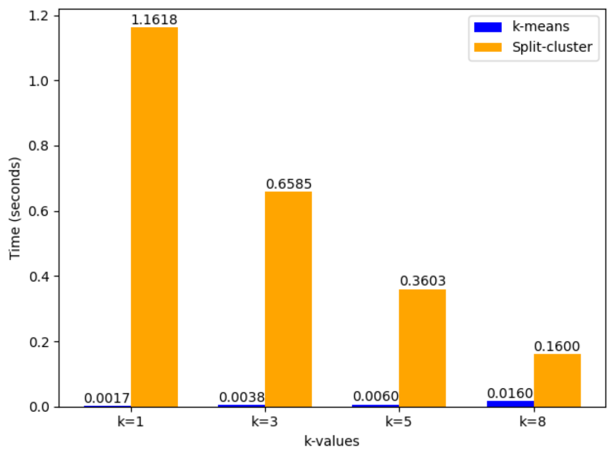

Fig. 7. Time required to converge for K-means and split cluster method for various values of k

The results in Fig. 5. clearly demonstrate the robustness of our density-aware split-cluster mechanism. When k-means clustering was initiated with k = 1, 3, and 5, the subsequent application of the split-cluster algorithm produced identical final clustering outcomes. This consistency confirms that the proposed method can recover the true cluster structure regardless of the initial value of k. However, it is important to note that if k is set higher than the actual number of clusters—or if k-means erroneously splits a single cluster into two—the algorithm does not merge these over-segmented clusters. In such cases, the method fails to correct the misclassification, which is a critical limitation.

The quality of the clustering was further evaluated using silhouette scores, as shown in Fig. 6. The silhouette score increased after applying the split-cluster method, with scores for k = 3 and k = 5 improving notably. This improvement indicates that the split-cluster mechanism effectively resolves merged clusters, leading to a more accurate delineation of the data structure. In other words, the enhanced silhouette scores reflect a better balance between intra-cluster cohesion and inter-cluster separation, confirming the efficacy of our approach in refining clustering outcomes.

In terms of computational expense, Fig. 7. provides a detailed comparison between the k-means clustering and the split-cluster method across different k values. The split-cluster algorithm is considerably more time-consuming than the k-means clustering step. For example, when k = 1, the split-cluster method takes over one second—approximately 72 times longer than k-means clustering with an appropriately chosen k. Although the execution time for the split-cluster method decreases significantly as k increases (with k = 3 and k = 5 requiring much less time), it remains substantially higher compared to the k-means phase. This indicates that while our method enhances clustering quality, its computational overhead is a major concern, particularly for large datasets. It requires a trade-off between clustering accuracy and efficiency. Therefore, further optimization of the algorithm is necessary to reduce the runtime and make it more practical for real-world applications.

V. Limitations and Future Work

Despite the promising performance of our density-aware split-cluster method, several limitations warrant consideration. First, the computational expense remains a significant concern. As illustrated in Fig. 7., the split-cluster algorithm, particularly when initiated with a low k value (e.g., k=1), incurs a substantially higher processing time compared to standard k-means clustering—up to 72 times longer. Although the execution time decreases with increasing k, it still remains considerably high, highlighting the need for further optimization, especially for large-scale datasets.

Second, our current approach does not include a merging mechanism. In scenarios where k-means over segments the data—splitting a single cohesive cluster into multiple clusters—the split-cluster method is solely designed to subdivide clusters based on low-density boundaries and does not merge erroneously separated clusters. Consequently, when over-segmentation occurs, the algorithm fails to rectify these errors, leading to suboptimal clustering outcomes.

Third, while the underlying concept of our method is applicable to higher-dimensional datasets, the experiments conducted in this study were limited to a two-dimensional dataset. This restricts the generalizability of our findings, and further investigation is needed to assess the performance and scalability of the method in higher dimensions.

Finally, an additional limitation arises from the initial k-means clustering stage. When k-means mis-assigns edge data points to the wrong clusters, the subsequent split-cluster process, which operates on a per-cluster basis, does not correct these erroneous assignments. As a result, misclassified points persist even after the splitting procedure, impacting the overall clustering accuracy.

Future work will address these challenges by optimizing the algorithm to reduce computational overhead, incorporating a merging mechanism to resolve over-segmentation, and extending the approach to higher-dimensional data. Moreover, we plan to develop strategies to rectify the mis-assignment of edge data points during the initial clustering phase, ultimately aiming for a more accurate and robust clustering framework.

VI. Conclusion

In this paper, we proposed a density-aware splitting mechanism for k-means clustering aimed at overcoming the limitations of traditional k-means, particularly in handling merged clusters. By detecting low-density boundaries within clusters, our approach effectively refines the initial clustering results obtained from k-means, as demonstrated on a synthetic dataset. The experimental results confirm that the proposed method consistently recovers the true cluster structure, regardless of the initial value of k, while yielding improved silhouette scores. Although the additional computational overhead, especially for low k values, remains a concern, the promising performance of the split-cluster algorithm highlights its potential for more robust clustering applications. Future work will focus on incorporating a merging mechanism to address over-segmentation, refining the algorithm to handle mis-assigned edge data points, and extending the approach to high-dimensional datasets.

Overall, our findings contribute to the development of more accurate and reliable clustering methods, offering valuable insights for further research in this area.

References

[1] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proc. 5th Berkeley Symp. Math. Statist. Probab., 1967, pp. 281–297.

[2] A. K. Jain, "Data clustering: 50 years beyond k-means," Pattern Recognit. Lett., vol. 31, no. 8, pp. 651–666, 2010.

[3] C. C. Aggarwal and C. K. Reddy, Data Clustering: Algorithms and Applications. CRC Press, 2014.

[4] R. L. Thorndike, "Who belongs in the family?," Psychometrika, vol. 18, no. 4, pp. 267–276, 1953.

[5] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," J. Comput. Appl. Math., vol. 20, pp. 53–65, 1987.

[6] M. E. Celebi et al., "A comparative study of efficient initialization methods for the k-means clustering algorithm," Expert Syst. Appl., vol. 40, no. 1, pp. 200–210, 2013.

[7] P. Fränti and S. Sieranoja, "k-means properties on six clustering benchmark datasets," Appl. Intell., vol. 48, no. 12, pp. 4743–4759, 2018.

[8] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in Proc. Annu. ACM-SIAM Symp. Discrete Algorithms, 2007, pp. 1027–1035.

[9] H.-P. Kriegel et al., "Outlier detection in arbitrarily oriented subspaces," in Proc. IEEE 12th Int. Conf. Data Min., 2012, pp. 379–388.

[10] G. Hamerly and C. Elkan, "Learning the k in k-means," in Adv. Neural Inf. Process. Syst., vol. 16, 2003, pp. 281–288.

[11] [1] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," in SODA, 2007.

[12] [2] R. L. Thorndike, "Who Belongs in the Family?," Psychometrika, 1953.

[13] [3] P. J. Rousseeuw, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," J. Comput. Appl. Math., 1987.

[14] [4] M. Ester et al., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases," in KDD, 1996.

[15] [5] M. Ankerst et al., "OPTICS: Ordering Points to Identify the Clustering Structure," in SIGMOD, 1999.

[16] [6] M. Steinbach et al., "A Comparison of Document Clustering Techniques," in KDD Workshop on Text Mining, 2000.

[17] [7] D. Reynolds, "Gaussian Mixture Models," in Encyclopedia of Biometrics, 2009.

[18] [8] A. Ng et al., "On Spectral Clustering: Analysis and an Algorithm," in NIPS, 2002.

[19] [9] D. Pelleg and A. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in ICML, 2000.

[20] [10] G. Hamerly and C. Elkan, "Learning the k in k-means," in NIPS, 2003.

[21] [11] A. Kalogeratos and A. Likas, "Dip-means: An Incremental Clustering Method for Estimating the Number of Clusters," in NIPS, 2012.

[22] [12] J. Xie et al., "Unsupervised Deep Embedding for Clustering Analysis," in ICML, 2016.