Unsupervised Learning: Improving the K-Means Algorithm

Juri Kembügler
Friedrich-Alexander Universität
Erlangen Nürnberg, Germany
juri.kembuegler@fau.de

Abstract

K-Means is a widely used clustering algorithm, but its effectiveness is often limited by poor centroid initialization, leading to suboptimal and unstable results. This paper investigates an improved approach proposed by İhsanoğlu and Zaval [12], which addresses this issue by identifying and correcting conflicting clusters and mega clusters. The method iteratively relocates centroids from dense regions to underrepresented areas, refining the result without restarting the algorithm. Our experiments on synthetic datasets demonstrate that the improved method outperforms both standard K-Means and K-Means++ in terms of homogeneity and silhouette scores, while also achieving consistent and low-variance convergence across multiple runs.

1. Introduction

Digitalization has permeated various aspects of our daily lives, from sports and fitness tracking to video streaming, as well as digital newspaper consumption. A key consequence of this transformation in recent years is the vast amount of data generated [8]. In 2010, the global data volume was approximately two zettabytes, whereas by 2024, it had surged to around 150 zettabytes (1 zettabyte $\approx 10^{21}$ bytes) [7]. Notably, since the onset of the global COVID-19 pandemic in 2020, the growth rate of data generation and consumption has accelerated significantly [7]. Thus, extracting meaningful insights from this data requires efficient and robust algorithms.

In this context, Machine Learning (ML) has become essential for recognizing patterns and enabling data-driven decisions. One important branch of ML is unsupervised learning, where algorithms uncover hidden structures in data without predefined labels [3]. A fundamental technique in this domain is clustering, with K-Means being one of the most widely used methods [4, 8]. However, the performance of K-Means is highly dependent on the initial choice of centroids [6, 12].

Contributions. In this article, we make the following contributions: We address K-Means' sensitivity to centroid initialization by investigating the adjustment strategy proposed by İhsanoğlu and Zaval [12], which corrects conflicting and mega clusters through a simple statistical post-processing step. We formalize the centroid relocation method in clear pseudocode, and evaluate its effectiveness across four synthetic datasets with varying cluster separability. Compared to K-Means and K-Means++, the improved approach achieves consistently higher homogeneity and silhouette scores. It also shows stability over 1,000 runs, with low variances in clustering results—demonstrating both improved quality and robustness.

2. Background and Related Work

Clustering is a key technique in unsupervised ML, where the goal is to identify inherent groupings in data without the use of labelled examples. While supervised learning relies on input-output pairs to train predictive models, unsupervised approaches such as clustering work solely with input data to uncover structure, summarize datasets, or extract meaningful patterns [3]. This section provides a focused overview of clustering methods, categorization of clustering algorithms, and the different types of clusters discussed in prior literature.

2.1. Clustering

The objective of clustering is to group similar objects based on their characteristics. The concept of '*similarity*' implies the use of a distance metric to quantify the relationships between data points. One of the most widely used distance metrics is the L2 norm (Euclidean distance) [3]. Alternative metrics such as the L1 norm, Itakura-Saito or Bregman distance can also be employed, depending on the specific application and data properties [8].

Clustering is mainly used to get an overview of data, to summarize it and to simplify human interpretation by categorizing the data and highlighting characteristics that distinguish the data points. However, it can also be used as a

preprocessing step for other methods, *e.g.*, feature creation for supervised learning [8].

Common clustering algorithms. Overall, clustering algorithms can be classified into two categories: *hierarchical* and *partitional* algorithms. Hierarchical algorithms build clusters by recursively merging related data points, whereas partitional algorithms divide the dataset into distinct clusters simultaneously [4, 8].

Hierarchical clustering methods include *agglomerative* approaches (such as single-, complete-, and average-linkage) and *divisive*, *e.g.*, monothetic and polythetic clustering. Partitional techniques, on the other hand, encompass algorithms such as *K-Means*, *Genetic Algorithms*, *Gaussian Mixture Models*, *Fuzzy C-Means*, and *DBSCAN* [4].

2.2. Types of Clusters

Kapil Joshi *et al.* [9] categorize clusters into four distinct types. In general, the effectiveness of a clustering algorithm heavily depends on the nature of these underlying structures [4]. For instance, density-based algorithms like DBSCAN are particularly effective when dealing with non-convex cluster shapes [2], whereas k-Means tends to perform best on well-separated, spherical clusters [8].

According to Kapil Joshi *et al.* [9], the four types of clusters are defined as follows:

- **Well-separated clusters:** A point in a cluster is closer or more similar to any point in the same cluster as to a point in another cluster.

Figure 1: Well-separated Clusters [9].

- **Centre-based clusters:** A point in a cluster is closer or more similar to the 'centre' of this cluster than to the centre of another cluster.

Figure 2: Centre-based clusters [9].

- **Contiguous clusters:** A point in a cluster is closer to one or more points in the same cluster as to any other point not in the cluster.

Figure 3: Contiguous clusters [9].

- **Density-based clusters:** A cluster is a dense region of points separated from other high-density regions by low-density regions.

Figure 4: Density-based clusters [9].

3. K-Means

This section introduces the K-Means clustering algorithm, detailing its historical origins and general procedure. We then focus on Lloyd's algorithm, the most common implementation, and explain its iterative optimization steps. The limitations of K-Means are discussed, including challenges in choosing the number of clusters and initializing centroids. Finally, we present the K-Means++ algorithm, a widely used technique to improve centroid initialization.

3.1. Background and Procedure

In contrast to other algorithms, K-Means was discovered independently in various research fields, first by Steinhaus (1956) and Lloyd (proposed 1957; published 1982) and later by Ball and Hall (1965) and MacQueen (1967) [8].

Despite variations in their implementation, all K-Means algorithms follow the same fundamental procedure for partitioning data into clusters. Given an input dataset represented as a $n \times d$ matrix—where n is the number of data points and d the number of features—K-Means requires the number of desired clusters, a similarity metric and an initialization strategy. The core algorithm, illustrated in Fig. 5, proceeds as follows:

Require: X, k, similarity metric, initialization strategy
1: Initialize k centroids $\mu_{1,\dots,k}$ using the chosen strategy
2: **repeat**
3: **for** each $x_i \in X$ **do**
4: Assign x_i to the cluster with the nearest centroid based on the similarity metric
5: **for** each cluster $j = 1$ to k **do**
6: $\mu_j \leftarrow$ mean of all x assigned to cluster j
7: **until** no data point changes its assigned cluster

While the pseudocode above reflects the standard implementation, they differ subtly in their objectives and approaches. For example, unlike Lloyd's algorithm, which updates centroids only after all data points have been reassigned, MacQueen's algorithm performs incremental updates, adjusting the affected cluster's centroid immediately when a data point moves from one cluster to another [11].

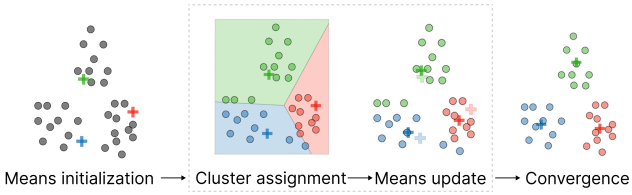

Figure 5: K-Means procedure visualized [1].

3.2. Lloyd's algorithm

One of the most widely used implementations of K-Means is Lloyd's algorithm [3], which iteratively refines cluster assignments and centroids through a simple yet effective approach. Given a dataset $X = \{ x_i \}_{i=1}^{N}$ of d-dimensional points, the goal is to iteratively refine the centroids $C = \{ \mu_i \}_{i=1}^{k}$ to partition the data into k clusters. The assignment of data points to clusters is encoded by a responsibility vector $R \in \{ 0, 1 \}^{N \times k}$, where $r_{nk} = 1$ indicates that point x_n belongs to cluster k, and 0 otherwise.

To quantify clustering quality, we define the loss function, also known as the *sum-of-squared errors (SSE)*, as the sum of squared distances between data points and their assigned centroids:

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{c=1}^{k} r_{nc} \|x_n - \mu_c\|^2. \tag{1}$$

To minimize the loss $\mathcal{L}$, Lloyd's algorithm iterates between two steps [3, 6]:

1. **Assignment Step:** Each data point is assigned to the nearest centroid by updating the responsibility vector:

$$r_{nc} = \begin{cases} 1, & \text{if } c = \arg\min_{c'} \|x_n - \mu_{c'}\| \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

2. **Update Step:** The cluster centroids are recomputed as the mean of all assigned points:

$$\mu_c' = \frac{\sum_{n=1}^{N} r_{nc} x_n}{\sum_{n=1}^{N} r_{nc}}. \tag{3}$$

These steps repeat until the assignments R no longer change, indicating convergence. Since the loss function $\mathcal{L}$ is non-increasing at each iteration, the algorithm is guaranteed to reach a local minimum, though not necessarily the global optimum.

3.3. Problems and limitations

Despite its simplicity and efficiency, K-Means has several limitations [4, 6, 8]. Since it relies on distance calculations, standardizing data is often necessary to ensure equal feature contributions [11]. The algorithm's sensitivity to the choice of k and centroid initialization can significantly impact performance. This section explores these challenges and strategies to address them.

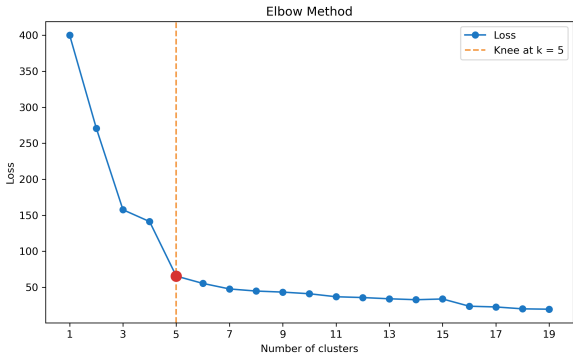

Figure 6: Elbow plot using the Mall Customer Segmentation Data [10].

Choose k. One of the few adjustable parameters in Lloyd's algorithm, apart from the initialization method, is the number of clusters k. Examining the loss function in Equation (1), we observe that a smaller number of clusters generally results in a higher loss, as data points are assigned to fewer centroids. Conversely, the loss decreases as k increases, reaching its minimum when each data point forms its own cluster. However, this solution is not meaningful, as it eliminates the concept of clustering altogether [3]. Since there is no optimal choice for k, Jain [8] suggests running the algorithm with multiple values of k and selecting the most significant one based on evaluation criteria.

A widely used heuristic for determining k is the *elbow method*. This approach involves plotting the loss function against k and identifying the point where the decrease in loss slows significantly—forming an '*elbow*' in the curve (see Fig. 6). The fundamental concept behind this method is that above a certain k, additional clusters result in diminishing returns in loss reduction [3].

Centroid initialization. The simplest method for initializing centroids is to select them randomly. However, as previously discussed, Lloyd's algorithm converges to a local minimum, making the final clustering outcome highly sensitive to the initial centroid selection. Consequently, poor initialization can lead to suboptimal solutions [6, 12].

According to Fränti and Sieranoja [6], there are two main ways to mitigate K-Means' sensitivity to centroid initialization: (1) using improved initialization techniques and (2) running the algorithm multiple times and selecting the best result based on a predefined criterion. Several heuristics have been proposed to enhance initialization:

- **Random points:** Centroids are chosen randomly from the dataset.

- **Furthest point heuristic:** The first centroid is chosen randomly, and subsequent centroids are selected to be as far as possible from those already chosen.
- **Sorting heuristic:** Data points are sorted by a criterion, *e.g.*, projection along a principal component and used for initialization.
- **Density-based:** This method selects centroids in high-density regions.
- **Projection-based:** A dimensionality reduction technique identifies representative points for initialization.
- **Splitting technique:** A small number of centroids are initialized first, then iteratively refined by splitting.

Despite these efforts, *"a clear state-of-the-art is missing"* [6], meaning no single method consistently outperforms others across all datasets. However, one widely adopted initialization method is K-Means++, which balances randomness with a distance-based heuristic to improve clustering performance.

3.4. K-Means++

As previously discussed, K-Means is highly sensitive to the initial choice of centroids. K-Means++ is a popular initialization strategy designed to mitigate this issue by selecting centroids in a more systematic manner. The algorithm, outlined in the following pseudocode, proceeds as follows:

Require: X, k

1: Randomly select the first centroid μ_1 from X
2: **for** $i = 2$ to k **do**
3: Compute the distance $D_j = d(x_j, \mu_{\text{neigh}})$ from each $x_j \in X$ to the nearest centroid μ_{neigh}
4: Select the next centroid μ_i with probability proportional to ${D_j}^2$

K-Means++ generally improves clustering performance by offering more effective initial centroid placement [6]. Nevertheless, due to its randomness, it does not entirely eliminate the risk of converging to suboptimal solutions [3, 12].

4. Statistically improving K-Means

As noted by Fränti and Sieranoja [6], K-Means presents several opportunities for optimization. However, a key limitation is that centroids struggle to relocate effectively when they are initially placed far from their optimal positions. To address this issue, İhsanoğlu and Zaval [12] propose an initial solution based on simple statistical methods, facilitating more efficient centroid movement.

4.1. Identifying issues

As previously mentioned, one of the main challenges in K-Means is that poorly initialized centroids may struggle to reach their optimal positions, if at all. This often leads to an uneven distribution of centroids—some regions may contain too many, while others are underrepresented [6]. In order to characterize such imbalances, İhsanoğlu and Zaval [12] propose separate approaches for *conflicting clusters* and *mega clusters*.

Conflicting Clusters. This type of cluster arises when multiple centroids are located too close to one another, typically indicating redundant clustering in a dense region. To detect them, we first compute the average distance between each centroid and its nearest neighbour:

$$\text{AvgDist} = \frac{1}{k} \sum_{i=1}^{k} d(\mu_i, \mu_{\text{neigh}}). \tag{4}$$

Where k is the number of clusters, $d(x, y)$ is the distance function, *e.g.*, Euclidean, and μ_{neigh} denotes the closest centroid to μ_i. Using this average, we define a threshold by introducing a tunable hyperparameter t. Any pair of centroids closer than $\text{AvgDist}/t$ is considered a conflicting cluster:

$$C_i \text{ is conflicting} \iff d(\mu_i, \mu_{\text{neigh}}) < \frac{\text{AvgDist}}{t}. \tag{5}$$

Mega Clusters. When too few centroids are assigned to a region, this can lead to the formation of large, coarse-grained clusters. These *mega clusters* often encompass multiple natural groupings. To detect them, we first calculate the intra-cluster variance:

$$\text{Var}(C_i) = \frac{1}{\sum_{n=1}^{N}(r_{ni}) - 1} \sum_{n=1}^{N} r_{ni} \|x_n - \mu_i\|^2. \tag{6}$$

Here, r_{ni} indicates cluster membership, and $\|x_n - \mu_i\|^2$ measures the squared distance from the data point to the cluster centroid. The mega cluster is then defined as the cluster with the highest variance:

$$C_{\max} = \arg\max_{C_i} \text{Var}(C_i). \tag{7}$$

4.2. The improved algorithm

Based on the previously defined concepts of conflicting and mega clusters, İhsanoğlu and Zaval [12] propose an extension to K-Means that iteratively relocates redundant centroids. The procedure is as follows: run K-Means (or K-Means++) as usual, identify conflicting and mega clusters, then iteratively move a centroid from a conflicting cluster to a randomly chosen point in the mega cluster. This process is repeated until no conflicting clusters remain or a maximum number of iterations is reached.

This post-processing strategy refines the clustering outcome without re-running the full algorithm, making it computationally efficient, compared to current solutions. By relocating centroids from overcrowded to underrepresented regions, the method directly addresses initialization-induced imbalance. Although conceptually simple, it was

shown by İhsanoğlu and Zaval to enhance cluster separation and compactness across a range of datasets. The pseudocode below summarizes the method:

Require: X, k, init_func, max_iter $\in \mathbb{N}$, $t \in [1, \infty)$

$C \leftarrow \texttt{init\_func}(X, k)$

for $i = 1$ To max_iter **do**

  $C, R \leftarrow \texttt{converge\_to\_clusters}(C, X, k)$

  $conflicts \leftarrow \texttt{Get\_conflicts}(C, t)$

  if $\texttt{len}(conflicts) = 0$ **then**

    break

  $mega\_cl \leftarrow \texttt{Get\_mega\_cluster}(X, k, R)$

  $conflict \leftarrow \texttt{rand.choice}(conflicts)$

  $C[conflict] \leftarrow \texttt{rand.choice}(mega\_cl)$

return $C,\ R$

4.3. Challenges

While the improved method introduces additional statistical computations and iterations, it differs from traditional approaches in that it does not restart the entire clustering process. Instead, it incrementally refines an initial K-Means result [12]. This targeted adjustment increases the likelihood of escaping poor local minima and reaching a more optimal clustering configuration—something that is often challenging to achieve through multiple K-Means runs alone [6].

A key limitation of the approach is its dependence on the choice of the threshold parameter t. To ensure meaningful identification of conflicting clusters, t must satisfy $t > 1$; otherwise, nearly all clusters would be classified as conflicting. Conversely, if t is set too high, the algorithm may terminate prematurely without any adjustments.

One potential strategy is to begin with a small value of t and incrementally increase it until convergence is achieved before reaching the maximum number of iterations. However, this introduces ambiguity: Early convergence could either indicate a well-chosen threshold or that K-Means had already produced a near-optimal solution, making it difficult to assess the effectiveness of t in isolation.

5. Results and Performance Evaluation

We evaluate the improved clustering algorithm against K-Means and K-Means++ using synthetic datasets, focusing on clustering quality and stability. Performance is measured with homogeneity and silhouette scores over several independent runs to assess both accuracy and consistency.

5.1. Methodology

To ensure reliable results, we conduct $1,000$ independent runs of each algorithm across multiple datasets. The performance of the algorithms is measured using two common clustering evaluation metrics: homogeneity and silhouette score. For each dataset, we compute the mean and variance of these scores to assess both the quality and stability of the clustering results.

Datasets. The following synthetic datasets, as introduced in [5], are used to evaluate clustering performance:

- **A-Sets:** Artificial datasets with a large number of well-separated clusters. They test the algorithm's ability to accurately partition clearly defined groups.
- **S-Sets:** Synthetic datasets characterized by overlapping or closely spaced clusters. These challenge the algorithm's precision in distinguishing densely packed data.

Together, these datasets provide a set of diverse and complementary perspectives on clustering quality under varying structural conditions.

Metrics. To assess the quality of the clustering results, we employ the following used evaluation metrics [12]:

- **Homogeneity Score:** This metric evaluates whether each cluster contains only data points from a single ground truth class. A score of 1.0 indicates perfectly homogeneous clusters.
- **Silhouette Score:** This metric measures how well each data point matches to its own cluster compared to other clusters. It evaluates the extent to which a data point is closer to points within its own cluster than to points in other clusters. The score ranges from -1 to 1, with higher values indicating that data points are well-clustered, meaning they are both tightly grouped within their cluster and distinct from points in other clusters.

5.2. Measurement results

Table 1 presents the clustering performance of the three algorithms—K-Means, K-Means++, and the improved method—on two synthetic datasets: *S1* and *A1*. Each algorithm was evaluated using the Homogeneity Score (H-Score) and Silhouette Score (S-Score), along with their respective variances, based on 1,000 independent runs.

Across both datasets and metrics, the improved method consistently outperforms K-Means and K-Means++ in terms of both mean score and stability. Notably, the variance for the improved method is below 10^{-4} in all cases, indicating consistency in its clustering results. This reflects the algorithm's robustness in overcoming initialization sensitivity, a common issue in K-Means-based approaches.

For example, on the S1 dataset, the improved algorithm achieves an H-Score of 0.9863 and an S-Score of 0.7113, surpassing K-Means++ (0.9504 and 0.6540, respectively) and K-Means (0.9209 and 0.6121). A similar pattern is observed in the A3 dataset, where the improved approach

		K-Means		K-Means++		Improved	
		Score	**Variance**	**Score**	**Variance**	**Score**	**Variance**
S1-Dataset	**H-Score**	0.9209	0.0008	0.9504	0.0006	**0.9863**	$\approx \mathbf{0}$
	S-Score	0.6121	0.0017	0.6540	0.0015	**0.7113**	$\approx \mathbf{0}$
S3-Dataset	**H-Score**	0.7656	0.0003	0.7701	0.0003	**0.7941**	$\approx \mathbf{0}$
	S-Score	0.4609	0.0003	0.4642	0.0003	**0.4915**	$\approx \mathbf{0}$
A1-Dataset	**H-Score**	0.9181	0.0010	0.9492	0.0006	**0.9978**	$\approx \mathbf{0}$
	S-Score	0.5162	0.0009	0.5454	0.0006	**0.5951**	$\approx \mathbf{0}$
A3-Dataset	**H-Score**	0.9360	0.0002	0.9587	0.0001	**0.9982**	$\approx \mathbf{0}$
	S-Score	0.5160	0.0004	0.5439	0.0003	**0.5999**	$\approx \mathbf{0}$

Table 1: Clustering performance on four synthetic datasets [5], evaluated using Homogeneity (H-Score) and Silhouette (S-Score). Each method was run 1,000 times; mean and variance are reported. The improved method shows consistently higher scores and near-zero (below 10^{-4}) variances, indicating stable convergence. The data is publicly available here: https://cs.uef.fi/sipu/datasets/.

again reaches near-perfect homogeneity and the highest silhouette value.

These results demonstrate the effectiveness of the conflict resolution and redistribution strategy in enhancing both clustering accuracy and reliability.

6. Conclusion

In this paper, we revisited the well-known K-Means algorithm and highlighted a central issue: its tendency to become stuck in suboptimal configurations due to poorly initialized centroids. Building on İhsanoğlu and Zaval's method [12], we implemented the enhancement that iteratively relocates centroids from densely populated regions (conflicting clusters) to underrepresented ones (mega clusters), leading to improvements in clustering performance and stability.

Our evaluation across multiple synthetic datasets shows that the approach consistently outperforms both K-Means and K-Means++ in terms of homogeneity and silhouette scores. Moreover, the method achieves low variance in repeated runs, suggesting robust and reliable behaviour.

While the method introduces a new hyperparameter t, we discussed its influence and outlined basic strategies for its selection. Although further evaluation on real-world datasets is required, the approach provides a lightweight refinement step that can improve K-Means results without fully reinitializing the algorithm.

Future work could explore adaptive threshold tuning and application to high-dimensional or real-world datasets.

References

[1] A. Amidi and S. Amidi. Stanford CS 229 Machine Learning Cheatsheets, 2018. Licensed under MIT License.

[2] S. Bhargav and M. Pawar. A Review of Clustering Methods forming Non-Convex clusters with, Missing and Noisy Data. *International Journal of Computer Sciences and Engineering*, 3:39–44, 2016.

[3] W. J. Deuschle. *Undergraduate Fundamentals of Machine Learning*. Bachelor's thesis, Harvard College, 2019.

[4] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[5] P. Fränti and S. Sieranoja. K-means properties on six clustering benchmark datasets. *Applied Intelligence*, 48(12):4743–4759, 2018. [Online; last accessed on April 6, 2025].

[6] P. Fränti and S. Sieranoja. How much can k-means be improved by using better initialization and repeats? *Pattern Recognition*, 93:95–112, 2019.

[7] IDC and Statista. Volumen der weltweit erstellten, erfassten, kopierten und konsumierten Daten von 2010 bis 2028, 2024. [Online; last accessed on March 15, 2025].

[8] A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[9] K. Joshi, H. Gupta, P. Chaudhary, and P. Sharma. Survey on Different Enhanced K-Means Clustering Algorithm. *International Journal of Engineering Trends and Technology (IJETT)*, 27(4):178–182, 2015.

[10] Kaggle. Mall Customer Segmentation Data, 2018. [Online; last accessed on March 19, 2025].

[11] L. Morissette and S. Chartier. The k-means clustering technique: General considerations and implementation in Mathematica. *Tutorials in Quantitative Methods for Psychology*, 9(1):15–24, 2013.

[12] A. İhsanoğlu and M. Zaval. Statistically Improving K-means Clustering Performance. In *2024 32nd Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, Turkey, 2024.