Implementation, Evaluation, and Comparison of K-Means and Other Clustering Algorithms

Carlos Jiménez, Sheena Lang, Zachary Parent, and Kacper Poniatowski

November 30, 2024

1 Introduction

This report focuses on evaluating various clustering techniques across three distinct datasets: Mushroom, Vowel, and Hepatitis, as detailed in Section 2. The clustering algorithms examined include K-Means, Fuzzy K-Means, OPTICS, Spectral Clustering, Global K-Means, and GMeans, as described in Sections 4.1 to 4.5 . We perform multiple experimental runs with different hyperparameters and assess the performance of each method using four evaluation metrics: two internal metrics (Davies-Bouldin Index and Calinski-Harabasz Index) and two external metrics (Adjusted Rand Index and F-Measure) as seen in 4.6. The results and analysis of these experiments are presented in Section 5.

2 Data

This section describes the datasets used in our study and details the preprocessing steps applied to prepare them for analysis.

3 Dataset Selection and Characteristics

This section outlines our dataset selection criteria and analyzes the characteristics of the chosen datasets for evaluating clustering algorithms.

3.1 Dataset Selection

In this project, we aimed to select three datasets that offer substantial variability across different aspects to evaluate the performance of various clustering algorithms, such as K-Means, G-Means, OPTICS, gs-FCM, etc.

The criteria for dataset selection were centered around diversity in dataset size, attribute types, class distribution, and missing data patterns. These factors allowed us to thoroughly analyze the efficiency and effectiveness of each clustering algorithm under different scenarios.

Dataset size variation To examine the scalability of the clustering algorithms, we selected one small dataset (Hepatitis), one medium-sized dataset (Vowel), and one large dataset (Mushroom). This setup enables us to assess the computational requirements and clustering performance of the algorithms across varying dataset sizes.

Attribute type diversity We chose datasets with diverse attribute types, including nominal, numerical, and mixed attributes. This ensures that the clustering algorithms are evaluated for their ability to handle varying data representations and preprocessing requirements.

Class distribution Although clustering is an unsupervised learning task, the class distributions of the datasets can serve as a baseline for validating the quality of clusters. We included one dataset with balanced classes (Vowel), one with slightly imbalanced classes (Mushroom), and one with a heavily imbalanced distribution (Hepatitis).

Missing data patterns We considered datasets with varying levels of missing data. The Hepatitis dataset contains missing values, challenging the algorithms to handle incomplete information, while the Vowel and Mushroom datasets are complete, ensuring a comparison of clustering performance in the absence of missing data.

3.2 Dataset Characteristics

- **Mushroom Dataset**
  - 8,124 instances
  - 22 nominal attributes
  - Binary classification (edible vs. poisonous, used for validation purposes)
  - Nearly balanced classes
  - No missing values
- **Hepatitis Dataset**
  - 155 instances
  - Mixed attribute types (13 nominal and 6 numerical)
  - Binary classification (survive vs. die, used for validation purposes)
  - Imbalanced classes (79.35% majority class)
  - 6.01% missing values
- **Vowel Dataset**
  - 990 instances
  - Mixed attribute types (3 nominal and 10 numerical)
  - Multi-Class classification (11 classes, used for validation purposes)
  - Balanced classes
  - No missing values

3.3 Selection Criteria

We selected the Mushroom, Hepatitis, and Vowel datasets to provide complementary characteristics for evaluating clustering algorithms. Our selection criteria focused on:

- **Dataset size variation** (small, medium, large)
- **Attribute type diversity** (nominal, numerical, mixed)

3.4 Data Preprocessing

This section details the preprocessing steps applied to prepare both datasets for analysis.

3.4.1 Handling Different Data Types and Ranges

To manage the varying types and ranges of attributes in our datasets, we implemented specific preprocessing techniques. For the nominal attributes in both the Mushroom and Hepatitis datasets, we used label encoding. This technique converts categorical values into numerical labels, enabling the algorithms to interpret the data correctly. While we considered one-hot encoding to avoid implying any ordinal relationship among categories, we opted for label encoding due to its simplicity and reduced dimensionality, as one-hot encoding would significantly increase dimensions and lead to a sparse space, making accurate predictions more challenging, particularly for KNN with the Mushroom dataset's numerous nominal features [1].

For the numerical attributes in the Hepatitis dataset, we applied min-max scaling to rescale the data to a fixed range of [0, 1]. This normalization is crucial for distance-based algorithms like KNN and SVM, ensuring all features contribute equally to model performance. We evaluated other scaling methods, such as standardization, but chose min-max scaling for its effectiveness in maintaining the original data distribution [4].

We implemented specific preprocessing techniques based on the characteristics of each dataset:

- **Nominal Attributes**
  - Applied label encoding to both datasets
  - Chose label encoding over one-hot encoding to prevent dimensionality explosion
  - Particularly important for the Mushroom dataset's numerous categorical features
- **Numerical Attributes**
  - Applied min-max scaling to the Hepatitis dataset's numerical features
  - Normalized all values to [0, 1] range
  - Essential for distance-based algorithms (KNN and SVM)

3.4.2 Missing Value Treatment

Addressing missing values is a critical step in preparing our datasets for analysis, as they can significantly impact model performance. In the case of the nominal attributes in both the Mushroom and Hepatitis datasets, we opted to impute missing values with the majority class. This method is straightforward and effective for maintaining dataset integrity. However, it can also introduce bias, particularly in the Hepatitis dataset, where the majority class represents 79.35% of instances. Relying on this method may lead to a situation where the imputed values disproportionately favor the majority class, thereby affecting the overall distribution and potentially skewing the results [4].

For the numerical attributes in the Hepatitis dataset, we used the mean of the available data to fill in missing values. This approach preserves the overall data distribution and is easy to implement, but it is not without its drawbacks. The mean can be heavily influenced by outliers, which might distort the data and lead to less accurate predictions. This is especially important in medical datasets, where extreme values may carry significant meaning.

We also considered employing K-Nearest Neighbors (KNN) for imputing missing values, as it could provide a more nuanced approach by considering the nearest data points for each instance. However, we ultimately decided against this option to avoid introducing bias into our evaluation. Since KNN is one of the algorithms we are testing, using it for imputation could influence its performance and lead to skewed results. Therefore, we chose the more straightforward methods of majority class imputation for nominal values and mean imputation for numerical values, allowing for a clearer assessment of the models' effectiveness without confounding factors.

We employed different strategies for handling missing values based on attribute type:

- **Nominal Attributes**
  - Imputed with mode (majority class)
  - Applied to both datasets
  - Potential limitation: May reinforce majority class bias
- **Numerical Attributes**
  - Imputed with mean values
  - Applied only to Hepatitis dataset
  - Preserves overall distribution while handling missing data

Alternative approaches such as KNN-based imputation were considered but rejected to avoid introducing bias into our evaluation of KNN as a classifier. Our chosen methods provide a balance between simplicity and effectiveness while maintaining data integrity.

4 Methods

This section provides an overview of the clustering algorithms used in our analysis, detailing their mechanisms, key parameters, and parameter variations. The methods include K-Means (Section 4.1), Improved K-Means (Section 4.2), Fuzzy C-Means (Section 4.3), OPTICS (Section 4.4), and Spectral Clustering (Section 4.5).

4.1 K-Means

K-Means is one of the most widely used clustering algorithms due to its simplicity and efficiency. It partitions a dataset into a predefined number of clusters by iteratively refining cluster centroids based on the distance between data points and the centroids.

4.1.1 Mechanism

The K-Means algorithm operates by minimizing the within-cluster variance, which is defined as the sum of squared distances between each point and the centroid of its assigned cluster. The process begins with the initialization of k centroids, where k represents the number of clusters. These centroids can either be selected randomly or provided as input.

Following initialization, each data point x_i is assigned to the nearest centroid c_j based on a distance metric, typically the Euclidean distance. This assignment is computed using the formula:

$$\text{Cluster}(x_i) = \arg\min_j ||x_i - c_j||_2^2.$$

After assigning clusters, the centroids are updated by recalculating their positions as the mean of all points assigned to each cluster. The new centroid c_j is determined using the equation:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i,$$

where C_j is the set of points in cluster j, and $|C_j|$ is the number of points in that cluster.

Finally, the algorithm checks for convergence by comparing the updated centroids to the previous ones. If the difference between the new and old centroids is less than a predefined tolerance (ϵ), or if the maximum number of iterations is reached, the algorithm terminates. The difference is computed as:

$$\Delta = \sum_{j=1}^{k} ||c_j^{(t)} - c_j^{(t-1)}||_2^2 < \epsilon.$$

K-Means is computationally efficient but sensitive to the initial positions of centroids, which can cause it to converge to a local minimum.

4.1.2 Parameter Grid

K-Means involves several important parameters, which include:

- **Number of Clusters ($n_{\textbf{clusters}}$)**: Determines the number of clusters (k) to form.
- **Maximum Iterations ($max_{\textbf{iterations}}$)**: The maximum number of iterations allowed for convergence.
- **Tolerance (ϵ)**: The convergence threshold based on the change in centroids.
- **Random State**: Controls the random seed for reproducibility of results.

The parameter variations used in our experiments are summarized in Table 1.

Table 1: K-Means Parameter Grid

Parameter	**Values**
Number of Clusters (n_{clusters})	2, 3, 4, 5, 6, 8, 10
Maximum Iterations ($max_{\text{iterations}}$)	100, 300, 500
Tolerance (ϵ)	$1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}$
Random State	1, 2, 3, 4, 5

4.2 Improved K-Means

This section provides an overview of the improved K-Means algorithms implemented, Global K-Means and G-Means.

4.2.1 Global K-Means

The global K-Means algorithm is an advanced clustering approach that improves upon the traditional K-Means by introducing an intelligent centroid initialization and incremental clustering strategy. Our implementation focuses on addressing key limitations of standard clustering techniques through a novel candidate selection and optimization mechanism.

Mechanism

The Global K-Means algorithm operates through a progressive clustering process with several key improvements:

The clustering process starts with a single cluster by using standard K-Means with $k = 1$, and incrementally adds clusters. For each iteration, the optimal location for the new centroid is determined by minimising the Within-Cluster Sum of Squares (WCSS). Candidate points for the new centroid are selected based on their minimum distance from existing centroids, with the number of candidates dynamically adjusted based on the current cluster count. An efficient selection method is used via `np.argpartition`, ensures scability.

The algorithm refines centroid placement by using vecotorised WCSS computation, reducing the computational complexity of the algorithm by simultaneously calculating the WCSS for all candidates. The configuration with the lowest WCSS is selected as the optimal solution for the current cluster count.

A unique aspect of this implementation is the caching mechanism. Intermediate results such as cluster labels, centroids, and distance matrices are stored persistently. This allows the algorithm to resume from cached states for higher values of k, thus reducing computational overhead dramatically by preventing recomputations of lower values of k. The caching mechanism is hash-based, ensuring compatibility with different datasets and configurations.

The implementation also includes an adaptive candidate reduction strategy. As the number of clusters increases, the number of candidate points are reduced to prevent the algorithm from becoming computationally infeasible. This adaptive strategy ensures that the algorithm remains efficient and scalable for large datasets, while maintaining high-quality clustering results.

Parameter Grid

The Global K-Means algorithm is highly configurable, with several key parameters that can be tuned to optimize performance:

- **Number of Clusters ($n_{\textbf{clusters}}$)**: Determines the number of clusters (k) to form.
- **Maximum Iterations ($max_{\textbf{iterations}}$)**: The maximum number of iterations allowed for convergence.
- **Tolerance (ϵ)**: The convergence threshold based on the change in centroids.
- **Random State**: Controls the random seed for reproducibility of results.

The parameter grid for Global K-Means is shown in Table 2.

Table 2: Global K-Means Parameter Configuration

Parameter	Values
Number of Clusters	[2, 3, 5, 10, 11, 12]
Maximum Iterations	[100, 300, 500]
Convergence Tolerance	$\{1\times10^{-5}, 1\times10^{-4}, 1\times 10^{-3}\}$
Random State	[1, 2, 3, 4, 5]

4.2.2 G-Means

The G-Means algorithm is an advanced clustering techniques that improves upon the standard K-Means algorithm by addressing a fundamental limitation: the need to specify a predetermined number of clusters. G-Means dynamically determines the optimal number of clusters by recursively splitting clusters based on statistically validating their Gaussian distribution.

Mechanism

The algorithm begins the clustering process by initialising a single cluster, obtained using standard K-Means with $k = 1$. For each cluster, the data points are split into two subclusters by applying K-Means with $k = 2$. The resulting clusters are then evaluated using Anderson-Darling Gaussianity test. This test evaluates whether the data points in the cluster are Gaussian distributed.

If the test indicates both subclusters are Gaussian distributed, the split is rejected and the original cluster remains unchanged. If at least one of the subclusters are not Gaussian distributed, the split is accepted and the process is repeated recursively for each subcluster until all clusters are Gaussian distributed, or the user-defined maximum depth is reached.

Before applying the Gaussianity test, the algorithm projects the data onto the principal components using Principal Component Analysis (PCA) to ensure the Anderson-Darling test is applied to the most significant directions in the data.

To prevent over-segmentation, a minimum number of observations (min_{obs}) is enforced for each cluster. If a cluster has fewer observations than the minimum threshold, it is not split further.

Parameter Grid

The G-Means algorithm offers several configurable parameters to fine-tune its clustering behavior:

- **Strictness** (s): Controls the sensitivity of the Gaussianity test.
- **Minimum Observations** ($\text{min}_{\textbf{obs}}$): Prevents splitting clusters with too few data points.
- **Maximum Depth** ($\text{max}_{\textbf{depth}}$): Limits the recursive splitting process.
- **Random State**: Ensures reproducibility of results.

The parameter grid for Global K-Means is shown in Table 3.

Table 3: G-Means Parameter Configuration

Parameter	Values
Strictness	[0, 1, 2, 3, 4]
Minimum Observations	[1, 5, 10]
Maximum Depth	[5, 10, 15]
Random State	[1, 2, 3, 4, 5]

4.3 Fuzzy C-Means

To be more precise, we implemented the ***Generalized Suppressed Fuzzy C-Means*** (gs-FCM) algorithm, which extends the Suppressed Fuzzy C-Means (s-FCM) approach by introducing a time-invariant, context-sensitive suppression rule [7, 6].

The s-FCM algorithm itself incorporates a constant suppression factor to enhance clustering performance. Intuitively, the gs-FCM algorithm retains the same primary objective as the original Fuzzy C-Means (FCM) algorithm: to partition a dataset into a predefined number of clusters, allowing data points to belong to multiple clusters with varying degrees of membership [3]. The addition of the suppression mechanism improves convergence efficiency and robustness, particularly in scenarios with imbalanced or noisy data.

4.3.1 Mechanism

Initially, the user specifies the number of clusters c and sets the fuzzy exponent $m > 1$, which controls the degree of fuzziness. Cluster prototypes are then initialized, either by applying intelligent initialization principles or by randomly selecting input vectors. A suppression rule and its corresponding parameter are chosen from predefined options, typically referenced in a lookup table. For this implementation, the suppression rule chosen is defined as follows:

$$\alpha_k = \frac{1}{1 - u_w + u_w \cdot (1 - \text{param})^{\frac{2}{1-m}}},$$

where u_w is the fuzzy membership of the winning cluster, param is the suppression parameter, and m is the fuzzy exponent.

At each iteration, fuzzy memberships are calculated using the standard FCM formula. For each data point $\mathbf{x}_k$, the algorithm determines the winning cluster (i.e., the cluster prototype closest to $\mathbf{x}_k$) and calculates the suppression rate α_k using the chosen rule. Suppressed fuzzy memberships are then computed using a modified formula that incorporates α_k, effectively reducing the influence of over-represented data points.

Cluster prototypes are updated using the suppressed memberships, ensuring that the influence of individual data points on the cluster centers aligns with the suppression mechanism. These steps are repeated iteratively until convergence, typically measured by the norm of the variation in cluster prototypes between successive iterations.

This suppression mechanism enhances the robustness of the clustering process, addressing imbalances in data distribution and improving the interpretability and quality of the clustering results.

4.3.2 Parameter Grid

gs-FCM involves the following parameters:

- **Number of Clusters**: Determines the number of clusters (k) to form.
- **Fuzzyness**: Controls the degree of fuzziness in the membership function.

* It's important to add that there's a suppression factor α that varies for each data point (i.e.m it is context/data sensitive) following a pre-defined suppresion rule.

The parameter variations used in our experiments are summarized in Table 4.

Table 4: gs-FCM Parameter Grid

Parameter	**Values**
Number of Clusters	2, 3, 5, 10, 11, 12
Fuzzyness	1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0
Suppression Factor	Varies for each data point (context/data sensitive)

By tuning these parameters, gs-FCM can adapt to various data distributions and noise levels, making it a powerful tool for clustering complex datasets.

4.4 OPTICS

OPTICS (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm that creates an augmented ordering of data points to identify cluster structures [2].

4.4.1 Mechanism

OPTICS works by computing a special ordering of points in the dataset based on their density-reachability relationships. The algorithm introduces two key concepts: core-distance and reachability-distance.

The core-distance represents how dense the neighborhood around a point is, measured as the minimum radius needed to classify a point as a core point (containing a minimum number of neighbors). If a point doesn't have enough neighbors, its core-distance is undefined.

The reachability-distance between two points measures how close they are from a density-connectivity perspective. It considers both the direct distance between points and the density of the neighborhood they're in, ensuring that points in dense regions are considered more closely connected than those in sparse regions.

The algorithm processes points in a specific order, maintaining a priority queue ordered by reachability-distance. For each point:

1. The core-distance is computed
2. For each unprocessed neighbor, the reachability-distance is calculated
3. Points are added to the priority queue based on their reachability-distance
4. The process continues with the point having the smallest reachability-distance

This ordering produces a reachability plot, where valleys in the plot represent clusters. The ξ parameter is used to identify significant drops in reachability that indicate cluster boundaries. The minimum cluster size parameter ensures that identified clusters have a meaningful number of points.

Unlike traditional clustering algorithms, OPTICS does not explicitly produce clusters but rather provides a density-based ordering that can be used to extract clusters of varying density. This makes it particularly effective at finding clusters of arbitrary shape and identifying noise points in the dataset.

4.4.2 Parameter Grid

OPTICS involves several important parameters, which include:

- **Metric**: The distance metric used for calculating point distances.
- **Algorithm**: The algorithm used to compute the nearest neighbors.
- **Minimum Samples** ($min_{\textbf{samples}}$): The number of samples in a neighborhood for a point to be considered a core point.
- **Xi** (ξ): The minimum steepness on the reachability plot that constitutes a cluster boundary.
- **Minimum Cluster Size** ($min_{\textbf{cluster}}$): The minimum number of samples in a cluster.

The parameter variations used in our experiments are summarized in Table 5.

Table 5: OPTICS Parameter Grid

Parameter	**Values**
Metric	euclidean, manhattan
Algorithm	auto, ball_tree
Minimum Samples	5, 10, 20
Xi	0.01, 0.05, 0.1
Minimum Cluster Size	5, 10, 20

4.5 Spectral Clustering

Spectral clustering is a technique that performs dimensionality reduction before clustering by using the eigenvectors of a similarity matrix [5].

4.5.1 Mechanism

The algorithm proceeds in three main steps:

1. Constructs a similarity matrix using either RBF kernel or k-nearest neighbors to capture relationships between data points
2. Computes the normalized Laplacian matrix and extracts its eigenvectors, creating a lower-dimensional representation that emphasizes cluster structure
3. Applies either k-means or cluster_qr to this transformed space to obtain the final clustering

This approach is particularly effective at identifying clusters of arbitrary shape, as it does not make assumptions about the geometric structure of the clusters.

4.5.2 Parameter Grid

Spectral Clustering involves several important parameters:

- **Number of Neighbors** ($n_{\textbf{neighbors}}$): The number of neighbors for affinity matrix construction.
- **Affinity**: The type of affinity matrix to construct.
- **Eigen Solver**: The eigenvalue decomposition strategy.
- **Assign Labels**: The strategy for assigning labels in the embedding space.
- **Random State**: Controls reproducibility of results.

The parameter variations used in our experiments are summarized in Table 6.

Table 6: Spectral Clustering Parameter Grid

Parameter	**Values**
Number of Neighbors	5, 10, 20
Affinity	nearest_neighbors, rbf
Eigen Solver	arpack, lobpcg
Assign Labels	kmeans, cluster_qr
Random State	1, 2, 3, 4, 5

4.6 Evaluation Metrics

We evaluated our clustering approaches using two internal and two external metrics to ensure a comprehensive assessment by capturing different aspects of clustering performance. The **Davies-Bouldin Index (DBI)** measures the compactness and separation of clusters, where lower values indicate better clustering. The **Calinski-Harabasz Index (CHI)** evaluates the ratio of between-cluster variance to within-cluster variance, favoring solutions with well-separated, compact clusters; higher scores are better. These two internal metrics together provide insights into the geometric quality of clusters.

On the other hand, external metrics assess alignment with ground truth labels. The **Adjusted Rand Index (ARI)** quantifies the agreement between true and predicted labels while correcting for chance, making it robust for comparing different clustering solutions. The **F-measure**, derived from precision and recall, evaluates how well the predicted clusters capture the true ones, focusing on class overlap. Together, these external metrics assess clustering accuracy and consistency, while complementing the internal metrics by incorporating the dataset's known structure.

5 Results and Analysis

5.1 K-Means

5.2 Improved K-Means

This section provides an overview of the results obtained from the improved K-Means algorithms implemented, Global K-Means and G-Means.

5.2.1 Global K-Means

5.2.2 G-Means

5.3 Fuzzy C-Means

5.4 OPTICS

5.5 Spectral Clustering

5.6 Summary

6 Conclusion

6.1 Key Findings

-

6.2 Practical Implications

-

6.3 Limitations and Future Work

-

6.4 Final Remarks

References

[1] Suad A Alasadi and Wesam S Bhaya. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12(16):4102–4107, 2017.

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60, June 1999.

[3] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984.

[4] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 international conference on management of data*, pages 2201–2206, 2016.

[5] Anil Damle, Victor Minden, and Lexing Ying. Simple, direct and efficient multi-way spectral clustering. *Information and Inference: A Journal of the IMA*, 8(1):181–203, 06 2018.

[6] Jiu-Lun Fan, Wen-Zhi Zhen, and Wei-Xin Xie. Suppressed fuzzy c-means clustering algorithm. *Pattern Recognition Letters*, 24(9-10):1607–1612, 2003.

[7] László Szilágyi and Sándor M Szilágyi. Generalization rules for the suppressed fuzzy c-means clustering algorithm. *Neurocomputing*, 139:298–309, 2014.

7 Appendix