A modified K-Means Algorithm for Partitional Clustering

BY

NAYEEM HOSSAIN

University Roll Number: 002010504043

Examination Roll Number: M6TCT23031

Registration Number: 154208 of 2020-21

Under The Guidance Of

PROF. NIRMALYA CHOWDHURY

A Thesis

Submitted in Partial Fulfilment of the Requirement for the Degree of

MASTER OF TECHNOLOGY

IN

COMPUTER TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERINGFACULTY OF ENGINEERING & TECHNOLOGY

JADAVPUR UNIVERSITY, KOLKATA

SEPTEMBER, 2023

Jadavpur University, Kolkata 700032

Department of Computer Science and Engineering

Faculty of Engineering and Technology

CERTIFICATE OF APPROVAL

This is to certify that the thesis entitled **"A Modified K-Means Algorithm for Partitional Clustering"** is a bona-fide record of work carried out by NAYEEM HOSSAIN in partial fulfilment of the requirements for the award of the degree **Master of Technology in the Department of Computer Science and Engineering, Jadavpur University** during the period of June 2022 to June 2023 (5^{th} & 6^{th} Semester). It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for the purpose for which it has been submitted.

_____

Signature of Examiner

Date:

_____

Signature of Examiner

Date:

TO WHOM IT MAY CONCERN

This is to certify that the dissertation titled **"A modified K-Means Algorithm for Partitional Clustering"** was completed by **Nayeem Hossain**, University Roll No: 002010504043, Examination Roll Number: M6TCT23031, University Registration No: 154208 of 2020-21, under the guidance and supervision **of Prof. Nirmalya Chowdhury, Department of Computer Science and Technology, Jadavpur University**. It is understood by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion there in but approve the project only for the purpose for which it has been submitted.

Prof. Nirmalya Chowdhury

(Guide)

Department of Computer Science & Engineering,

Jadavpur University

Prof. Nandini Mukherjee

Head of the Department,

Department of Computer Science And Engineering,

Jadavpur University

Prof. Arthendu Ghosal

Dean,

Faculty of engineering and Technology,

Jadavpur University

DECLARATION

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of Master of Technology of Computer Technology of the Department of Computer Science and Engineering studies.

All information in this document have been obtained and present in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: **Nayeem Hossain**

Roll Number: **002010504043**

Thesis Title: **A modified K-Means Algorithm for Partitional Clustering**

Signature With Date:

ACKNOWLEDGEMENT

First and foremost, I want to express my gratitude to God Almighty for pro voiding me with the strength, wisdom, and capability to go on this amazing adventure and to continue and successfully finish the embodied research work. I'd like to thank Professor Nirmalya Chowdhury of the Department of Computer Science and Engineering at Jadavpur University for his excellent assistance, consistent support, and inspiration during my dissertation. I owe Jadavpur University a great debt of gratitude for providing me with the chance and facilities to complete our thesis.

I am grateful to every one of the teaching and non-teaching personnel whose assistance has made our trip during my research time much easier. I would like to thank my project partners Soumya Nag, my seniors, and my friends for providing me with regular encouragement and mental support throughout our effort.

Last, but not the least, my family deserves great recognition. There are no words to express my gratitude to my mother and father for all of the sacrifices you've made on my behalf. Your prayers for me have kept me going thus far.

Nayeem Hossain

Master of Technology

Roll No: 002010504043

Exam Roll No: M6TCT23031

Registration No: 154174 of 2020-21

Department of Computer Science & Engineering

Jadavpur University, Kolkata

Contents

Chapter 1

Introduction

1.1. Introduction to Pattern Recognition

Pattern recognition is a branch of machine learning and AI that deals with the detection and interpretation of commonalities or relationships within data. These relationships can appear in a variety of forms, including images, signals, texts, and more. Ultimately, pattern recognition involves the development of algorithms and techniques that allow computers to automatically differentiate between different categories or classes of data based on those relationships.

Key Concepts:

1. **Feature Extraction:** The first step in pattern recognition is often the extraction of features. Features are characteristics or attributes of the data that can be used to differentiate different patterns. For instance, image recognition can use features such as pixel values, textures, color distribution, edges, etc.
2. **Dimensionality Reduction:** Data sets in the real world can be very complex, which can cause problems with computing and overfitting. To reduce the complexity of the data, techniques such as PCA and t-SNE can be used to reduce the number of features while still preserving the essential information.
3. **Classification Algorithms:** Once the features are defined, different classifications are applied to the data points. Some of the most common classifications are k-nearest neighbours, support vector machines, decision trees, random forests, and neural networks.
4. **Clustering:** Another key feature of pattern recognition is the ability to group similar data points together. Clustering is the process of classifying data without the need for a set of predefined classes. Clustering algorithms used in pattern recognition include K-Means, Hierarchical Cluster, and DBSCAN.
5. **Supervised vs. Unsupervised Learning:** There are two basic types of learning in pattern recognition: supervised and non-supervised learning.

   In supervised learning, a model is trained on data with known classes (labelled data). This model learns the patterns and makes predictions based on new, non-labelled data. In non-labelled learning, a model learns patterns and structures based on unlabelled data.

6. **Feature Selection:** Not all features are equal. The goal of feature selection is to find and use the most relevant features for a specific task. This helps to improve model performance, reduce complexity, and avoid overfitting.
7. **Neural Networks:** Deep learning, also known as machine learning, is a subset of deep learning. Deep learning has revolutionized the field of pattern recognition, especially in the fields of image recognition, speech recognition, and deep learning neural networks (DNNs). DNNs, particularly CNNs for images and RNNs for sequences, are highly effective at recognizing complex patterns.
8. **Applications:** Pattern recognition has a wide range of applications, including:
    - **Image Recognition:** Identifying objects, people, or scenes in images.
    - **Speech Recognition:** Converting language spoken into text.
    - **Text Classification:** Categorizing documents, sentiment analysis, and spam detection.
    - **Biometric Identification:** Fingerprint, iris recognition, and face.
    - **Medical Diagnosis:** Detecting diseases from medical images and signals.
    - **Financial Fraud Detection:** Identifying unusual patterns in transactions.
    - **Robotics:** Enabling robots to recognize and interact with their environment.
9. **Challenges:** The challenge of pattern recognition lies in the variability of the data, the problem of overfitting, the problem of noisy data and the problem of the "curse of dimensionality". The goal of constant research is to create robust algorithms that are able to adapt well to different situations.

1.1.1 What is Pattern Recognition?

Pattern recognition is the cognitive process by which people recognize and explain patterns, similarities or trends in data, information or sensory inputs. It is an essential part of human intelligence, and is used in many different areas of study, including psychology, neuroscience and computer science, as well as machine learning. Pattern recognition is the process by which people take input data and use it to identify, categorize or predict new or unknown information.

There are several steps involved in pattern recognition.

1. Data Acquisition: Collection of raw data or data input on which to perform pattern recognition. This can range from images, sounds, text, or numerical values.

2. Preprocessing: Data normalization is the process of cleaning and preparing data to remove noise, inconsistencies or irrelevant information in order to make it suitable for analysis. This process may include data standardization, filtering and transformation.

3. Feature Extraction: Ensuring that the patterns in the data are represented by specific features or characteristics. These features may be attributes, measurements or characteristics that are relevant to the analysis.

4. Feature Selection: Selecting the features that are most important while getting rid of the ones that are superfluous or not very useful. This reduces the complexity of our analysis and improves the performance of our pattern recognition algorithms.

5. Pattern Matching or Classification: This is pattern recognition at its most basic. Algorithms and techniques are used to compare the extracted features to pre-defined models or patterns. This involves finding similarities or differences in the data and classifying it into different categories or groups.

6. Training and Learning: Algorithms in machine learning and AI often need to be trained on labelled data to understand the patterns. During the training, the algorithm changes its internal parameters to better understand the patterns.

7. Testing and Validation: In this step, the algorithm's performance is evaluated based on new, non-observed data after training. The goal of this step is to determine whether the algorithm is able to generalize and correctly identify patterns.

8. Pattern Refinement: To improve the accuracy or effectiveness of the algorithm, testing results may be used to refine or improve the algorithm. This may include changing parameters or selecting different algorithms.

9. Prediction or Decision-Making: Once trained and validated, the pattern recognition system can be used to predict, make decisions, or classify incoming data.

10. Feedback Loop: In some instances, the data generated by the pattern recognition can be used to enhance the pattern recognition tasks in the future. This is often the case with adaptive or machine-learning models.

In the real world, pattern recognition is used for:

- Image and Speech Recognition: It can be used to identify things, people, or sounds in pictures or audio files.

- Medical Diagnostics: Pattern recognition for medical imaging to help diagnose diseases such as cancer or detect abnormal heartbeats.

- Financial Forecasting: Examining historical data to forecast future market movements or share prices.

- Natural Language Processing: The ability to comprehend and analyze human speech, allowing for chatbots and translation of languages.

- Biometrics: Identifying patterns in fingerprint data, iris scans or facial scans for security and identification.

- Autonomous Vehicles: How to recognize road signs, walkers, and other cars to help self-driving cars navigate.

In short, pattern recognition bridges the gap between raw data and valuable insights, allowing both humans and machines to process complex data and make informed decisions.

1.1.2 Pattern Recognitions Methods

1.1.2.1 Supervised Method

Supervised Pattern Recognition (SPR) is a form of machine learning in which an algorithm learns from labelled training data to generate predictions or decisions based on new, non-labelled data. In supervised pattern recognition, the training data is given to the algorithm along with appropriate target labels and the algorithm learns to map it to the appropriate labels based on patterns found in the trained data. Below is a description of supervised pattern recognition and an example algorithm.:

1. Data Collection and Labelling: First, a label is applied to each data point in the dataset. For example, in an email message dataset, each message is labelled as “spam” or “not spam”.

2. Feature Extraction: For example, for text data, it could be word frequency, and for images, it could be pixel values, or extracted visual features. These features are used to represent patterns that the algorithm learns from.

3. Dataset Splitting: The training set is the part of the labelled dataset that trains the algorithm, and the testing set is the part that tests the algorithm on unseen data.

4. Algorithm Selection: The supervised pattern recognition algorithm (SNCO) is selected on the basis of the type of data and the nature of the problem. One of the most widely used SNCOs is the support vector machine (SVM).

5. Training the Algorithm (Supervised Learning):

- The algorithm receives the training data and its associated labels.

- It learns to construct a model that matches the features to the appropriate labels.

- For SVM, it creates a hyperplane that separates data points from different classes while minimizing the number of misclassifications.

6. Validation and Tuning: The algorithm is tested against the testing set. The parameters of the algorithm can be changed to make it more accurate, and methods such as cross validation can be used to make sure the algorithm is reliable.

7. Prediction or Classification: Once the algorithm has been trained and fine-tuned, it can be applied to new, non-observed data points to generate labels. The algorithm then applies the predicted patterns to the input data points to generate the predicted label.

8. Evaluation: Metrics are used to evaluate the algorithm's performance. These metrics include but are not limited to:

Accuracy

Proximity

Reconciliation

F1-score

Example Algorithm: Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a supervised learning algorithm that is commonly used in classification algorithms. It works by identifying the hyperplane that is most suitable for separating data points from different classes. Here is how it works:

1. Data Representation: The input data is in the form of feature vectors and each feature vector is assigned a class label.

2. Hyperplane Construction: SVM finds the hyperplane that has the most space between them. The space is the distance from the hyperplane to the closest points of each class.

3. Soft Margin and Kernel Trick: In situations where the data is not separable, a "soft margin" allows for a little bit of misclassification. Using the kernel trick, the data is transformed into a higher dimensional space, which may make it easier to separate.

4. Classification: After the hyperplane has been defined, new points are sorted according to where they fall on the hyperplane's side. The side defines their predicted class.

5. Optimization: The goal of SVM is to reduce the classification error while maximising the margin and staying within the soft margin limits.

SVM is only one of many supervised pattern recognition algorithms. Depending on the complexity of the problem and the nature of the supervised pattern recognition algorithm, other supervised pattern recognition algorithms may be used, such as decision trees, random forest, K-nearest neighbours and neural networks.

1.1.2.2 Semi-supervised Method

The term semi-supervised means "supervised" or "unsupervised" and it refers to a combination of supervised and non-supervised learning. Semi-supervised models combine a small amount of labelled data with a large amount of unlabelled data to create models that are more reliable and accurate. This approach is especially useful when getting labelled data is costly or time consuming. Here's how it works and an example algorithm.

1. Labelled data and unlabelled data: In semi- supervised learning, we have one set of labelled data (where each data point has a label) and another set of unlabelled data (where there are no labels).

2. Feature Extraction: In a similar way to supervised learning, patterns are represented by extracting features from the data.

3. Algorithm Selection: Semi-supervised learning algorithm is selected as an appropriate algorithm. Self-Training algorithm is one such algorithm.

4. Self-Training Algorithm:

-The algorithm begins with a small labelled dataset to train the model, similar to supervised learning.

-After that, the model is trained on the unlabeled data until it has reached a certain confidence threshold. The data points that the model is confident about its predictions for are labeled as "Pseudo-Labels" and the data points that do not have a high-confidence prediction are temporarily labeled as "False-Pseudolabeled".

-After re-training, the dataset is expanded to include both the labelled and pseudo-labeled data, and the model is trained again using the expanded dataset.

5. Iteration: Pseudolabeling and re-training may be repeated for a set number of times or until convergence is achieved.

6. Validation and Tuning: An independent validation set evaluates the model's performance to measure model accuracy and generalization.

7. Final Model and Prediction: As in supervised learning, once the model is performing well, it can then be used to forecast labels for new, non-observed data points.

The self-training algorithm mentioned above is an example of a semi-supervised model. Other semi-supervised models include co-training models, multi-view models, and ladder networks, all of which have their own unique features and strengths.

Semi supervised learning is especially useful in situations where labelled data is limited or costly, but there is a large amount of unlabelled data that can still provide useful insights into the underlying patterns.

In short, semi-super supervised pattern recognition combines the benefits of both labelled and unlabelled data to improve model performance and make better use of available resources.

1.1.2.3 Unsupervised Method

Unsupervised Pattern Recognition (UPR) is a type of machine learning in which the algorithm learns unlabelled patterns from unstructured data without the help of pre-defined labels. The purpose of unstructured data is to uncover hidden patterns, relationships, and clusters within the dataset. Unstructured methods are especially useful when we need to explore and discover the underlying patterns in a dataset. Below is a description of how unstructured data can be used for unstructured data and an example algorithm:

1. Data Collection: The process of unsupervised learning begins with the collection of an unlabelled dataset. Unlabelled data can be text, images or numerical values.

2. Feature Extraction: Intrinsic elements are extracted from data to capture its intrinsic properties. These elements aid in representing the data in a manner that allows for pattern identification.

3. Algorithm Selection: Unsupervised learning algorithms are selected based on the type of data and problem. One of the most popular unsupervised learning algorithms is K-Means Clustering.

4. K-Means Clustering:

- Initialization: K initial cluster centroids are chosen randomly from the data points.

- Assignment: Each data point is assigned to the nearest centroid, forming clusters.

- Update Centroids: The centroids of the clusters are recalculated based on the mean of the data points in each cluster.

- Iteration: Steps 2 and 3 are repeated iteratively until convergence (when centroids stop moving significantly) or a predetermined number of iterations is reached.

5. Cluster Interpretation: Once the algorithm has connected the dots, the individual pieces of data that have been grouped together give us an idea of how the data is organized. All the data points in the same group have similar features.

6. Dimensionality Reduction (Optional): In addition to supervised learning, unsupervised learning techniques such as PCA or t-distributed stochastic neighbour embedding (TSNE) can be used to reduce the data dimensionality while preserving its essential properties.

7. Visualization and Analysis: We can visualize the results to see the connections between the data points and the clusters. This step helps us to make sense of the patterns found.

8. Interpretation and Insight Generation: By exploring the properties of each cluster data point, we can gain insights, form inferences, and potentially learn more about the data.

9. Application and Future Exploration: The patterns found can be used to segment customers, group similar documents, or find anomalies. Additional research may include optimizing the algorithm or performing further analysis.

Unsupervised learning can be used to explore data without any prior knowledge about the class or category. It can be used to segment customers, model topics, detect anomalies, and more. K-Means is one of the most widely used unsupervised learning algorithms for classifying data.

Chapter 2

Clustering Techniques

2.1 Introduction

Cluster algorithms are essential for pattern recognition because they group similar data points together according to their intrinsic similarity. Cluster algorithms are designed to reveal underlying structure within data without using pre-defined class labels. Cluster methods are important for understanding data distribution, finding patterns, and uncovering hidden relationships.

One of the most widely used cluster algorithms is called K-Means. K-Means is a simple algorithm that groups data points into ‘k’ clusters (‘k’ being a parameter that can be specified by the user). The algorithm sorts data points into clusters by iteratively assigning them to the nearest cluster centroid (centroid). The centroids are then recalculated based on newly assigned points until convergence, where the centroids do not significantly change.

Here's a step-by-step outline of the K-Means algorithm:

1. Initialization: Randomly select 'k' initial cluster centroids.

2. Assignment: Assign each data point to the nearest centroid, forming 'k' clusters.

3. Centroid Update: Recalculate the centroids of each cluster based on the mean of the data points in that cluster.

4. Iteration: Repeat steps 2 and 3 until the centroids no longer change significantly or a predetermined number of iterations is reached.

5. Convergence: When the algorithm converges, each data point is associated with a specific cluster, and the centroids represent the centres of the clusters.

When clusters are well separated, K-Means performs well. However, it may not be able to handle convex clusters or different cluster sizes. Other cluster algorithms such as Hierarchical, DBSCAN, and GMM offer alternative approaches to deal with different types of data distribution and cluster shapes.

Cluster algorithms are used for a variety of purposes, including customer segmentation and image compression. They can also be used for document categorization, gene expression analysis, and more. Cluster algorithms offer valuable insights into data structures and patterns, helping in decision making and knowledge discovery.

2.2 Clustering Techniques

2.2.1 Hierarchical Technique

The goal of hierarchical grouping is to classify data points according to their similarity. In contrast to K-Means, it does not have a fixed number of clusters. Instead, it groups data points into tree-like structures called dendrograms. These dendrograms capture different degrees of granularity. This allows users to study the structure of the data from large groups to individual data points.

One of the most popular algorithms for hierarchical grouping in pattern recognition & data analysis is Agglomeration Hierarchical Cluster.

Here's an overview of the Agglomerative Hierarchical Clustering algorithm:

1. Data Representation: Begin by representing each data point as an individual cluster.

2. Pairwise Distance Computation: Gain an idea of the distance between each pair of clusters by using a distance metric (for numerical data, Euclidean distance). For categorical data, use Jaccard similarity.

3. Merge Closest Clusters: Determine the nearest two clusters based on the distance calculated and combine them into one cluster. The algorithm behavior is affected by the selection of distance measure and the linkage criterion (measurement of distance between clusters).

4. Update Distance Matrix: Calculate the distance from the new cluster to the remaining clusters based on the selected linkage criterion.

5. Iteration: Repeat steps 3 and 4 iteratively until all data points are merged into a single cluster or a predetermined number of clusters is reached.

6. Dendrogram Creation: A dendrogram is the result of the merging process. A dendrogram shows how clusters form and how they connect at different levels.

The result is a hierarchical dendrogram which provides information about the relationships and the structure of the data. We can set a limit on the hierarchical dendrogram to determine the number of clusters we want to see. We can cut the hierarchy at higher levels to get fewer, more generic clusters, and we can cut it at lower levels to get more specific, more fine-grained clusters.

Hence, hierarchical cluster analysis is useful for studying the hierarchical structure of data, finding natural groupings in the data, and learning about the relationships between the clusters. Due to the quadratic complexity of the data, it is computationally expensive to work with large datasets. Nevertheless,

hierarchical cluster analysis remains a widely used and versatile technique in many fields, including biology, social science, data visualization, etc.

2.2.1.1 Agglomerative Nesting

2.2.1.1.1 Single Linkage

Single Linkage agglomerative grouping is a specific method within hierarchical grouping that merges clusters according to the distance between the nearest members of the cluster. This method results in the formation of dendrography by gradually connecting clusters with the shortest distance between members, resulting in long, and often elongated, clusters. The simplicity of the single linkage approach makes it easy to apply, but it can be prone to noise and out-of-whack clusters. Here is an explanation of single linkage agglomerative grouping and its algorithm.

Single Linkage Agglomerative Clustering Algorithm:

1. Initialization: Begin with each data point as an individual cluster.

2. Pairwise Distance Computation: Calculate the distances between all pairs of data points using a suitable distance metric.

3. Agglomeration: Find the cluster with the shortest pairwise distance from its nearest neighbour. Then group these two clusters together.

4. Update Distance Matrix: Calculate the distance from the new cluster to the remaining clusters. Distance between two clusters is the distance between their nearest members in this context.

5. Iteration: Repeat steps 3 and 4 until all data points are merged into a single cluster or a predetermined number of clusters is reached.

6. Dendrogram Creation: The merging process generates a dendrogram, visually representing the hierarchical structure of clusters and the order in which they are formed.

Single link aggregation clusters tend to be elongated. This is due to the fact that it takes into account the distance between the nearest points in the merging cluster, resulting in clusters that are connected by a few adjacent data points, even though most of the data points are far away. Due to its sensitivity to noise and outlier effects, elongated clusters can sometimes be less cohesive and meaningful.

Single Linkage clusters can be used for a variety of purposes, from biology (taxonomy) to social science (community detection) and document analysis (document cluster). The simplicity of Single Linkage and its capacity to capture

elongated (or chain-like) clusters can be useful when the data exhibits these structures.

However, it is important to be careful when using this approach, especially on datasets where outliers may disproportionately affect the clustering results.

2.2.1.1.2 Complete Linkage

Complete linkage agglomerative grouping is a method of grouping clusters according to the maximum distance of their members. Complete linkage groups together by iteratively connecting clusters with the furthest distance between them. The resulting dendrogram is compact, spherical. Complete linkage grouping can be more resilient to outliers than single linkage grouping. Here is an explanation of complete linkage aggregation grouping and its algorithm:

Complete Linkage Agglomerative Clustering Algorithm:

1. Initialization: Start with each data point as an individual cluster.

2. Pairwise Distance Computation: Compute the distances between all pairs of data points using an appropriate distance metric.

3. Agglomeration: Find the pair of clusters with the maximum pairwise distance between their members. Merge these two clusters into a single cluster.

4. Update Distance Matrix: Calculate the distance from the new cluster to the remaining clusters. Distance is the maximum distance between two clusters.

5. Iteration: Repeat steps 3 and 4 until all data points are merged into a single cluster or a predetermined number of clusters is reached.

6. Dendrogram Creation: As a result of the merging process, the dendrogram is created. This dendrogram shows the hierarchical order of the clusters and how they are formed.

Complete link aggregation clusters tend to be more compact and round in shape compared to single link aggregation clusters. Taking into account the maximum distance of the cluster members reduces the sensitivity to outliers and noise as a few isolated points have less effect on the cluster merger. This can result in more compact and well-defined clusters in cases with outliers.

The use of Complete Linkage clusters in biology (genetic expression analysis) and psychology (Personality trait cluster) as well as in astronomy (star classification) makes it well-suited for situations where data points have varying densities or when outliers must be managed efficiently.

2.2.1.1.3 Average Linkage

Average linkage agglomeration is a hierarchical grouping method that merges clusters according to the average distance of their members. The goal of this method is to find the right balance between complete linkage compactness and single link sensitivity. The dendrogram is formed by iteratively connecting clusters that have the smallest average distance between them. This means that clusters are less affected by outliers as compared to single link. Here is an explanation of AGL Agglomeration along with the algorithm.:

Average Linkage Agglomerative Clustering Algorithm:

1. Initialization: Begin with each data point as an individual cluster.

2. Pairwise Distance Computation: Calculate the distances between all pairs of data points using an appropriate distance metric.

3. Agglomeration: Find the pair of clusters with the smallest average distance between their members. Merge these two clusters into a single cluster.

4. Update Distance Matrix: Calculate the distance from the new cluster to the other clusters. Distance between clusters is the average distance of the members of two clusters.

5. Iteration: Repeat steps 3 and 4 until all data points are merged into a single cluster or a predetermined number of clusters is reached.

6. Dendrogram Creation: As a result of the merging process, a dendrogram is created that shows the hierarchical order of clusters and how they are formed.

Average linkage agglomerative cluster seeks to balance the effects of single and complete linkages. By taking average distances into account, average linkage clusters are less likely to form elongated clusters similar to single linkages and are less sensitive to outliers like complete linkages.

Average linkage clusters are used in biology (Phylogenetic tree constructions), ecology (Community structure analysis) and marketing (Customer segmentation). This balanced approach can be especially useful when working with datasets with varying density, outliers or when we want clusters that are well separated but not overly compact.

2.2.1.2 Divisive Analysis

Divisive hierarchical grouping is a top-down method of grouping data. It starts with a single dataset as a cluster and divides the dataset into subclusters. This

method is different from agglomerative grouping, which starts with a single data point and merges it into clusters.

Divisive hierarchical grouping provides a hierarchical structure, similar to a dendrogram. In this method, clusters are created by dividing the dataset into a series of subclusters.

Below is a brief explanation of Divisive hierarchical grouping along with its algorithm.

Divisive Hierarchical Clustering Algorithm:

1. Initialization: Begin with all data points belonging to a single cluster.

2. Splitting: Identify the cluster with the highest variability or variation within the cluster. Divide the cluster into two subclusters using a selected criterion.

3. Recursion: Repeat the splitting step for each new subcluster formation. Keep repeating until we reach a stopping point, such as the number of clusters we want to have or the granularity we want.

4. Dendrogram Creation: A dendrogram is created by the recursively splitting process. A dendrogram is a graph that shows the hierarchical order of clusters and how they are separated.

Divisive hierarchical grouping can be more computational demanding than agglomerative grouping as it involves dividing the data repeatedly. However, it can provide benefits in cases where the data has a well-defined hierarchy of nested clusters as it can capture complex structures.

Divisive clusters are used in biology (taxonomy) and social science (subgroup analysis) as well as ecology (species classification) as it can uncover finer structures within the data. This makes it suitable for cases where we want to investigate relationships across many scales.

However, it should be noted that divisive clusters may not always be suitable for datasets where clusters are not well-defined or where clusters have varying densities.

2.2.2 Partitional Clustering

Machine learning and data analysis use partitional algorithms to group similar data points together. Partitional algorithms are based on the idea that a dataset should be divided into non-interrelated subsets or clusters. Each data point

belongs to a single cluster. One of the most common partitional algorithms is K-Means.

Let’s take a look at what partitional algorithms are and how K-Means is used.:

1. Partitional Clustering: Partitional grouping is the process of dividing a set of data into a certain number of parts, usually referred to as "k". This number of parts is usually determined by either the user or an automated system, such as the elbow or silhouette analysis.

2. K-Means Algorithm: K-Means is a widely used partitional cluster algorithm that follows the following steps:

a. Initialize the cluster centroids by randomly selecting k centroids from the dataset

b. Assign the nearest centroid to the data points in the dataset using a distance metric, such as Euclidean distance

c. Update centroids: The centroids of the cluster are recalculated as the average of all data points assigned to the cluster.

d. Repeat steps b and c until convergence, which is when the centroids do not change significantly or the maximum number of iterations are reached

e. Result: The dataset is divided into k clusters, with each data point belonging to the cluster that has the centroid closest to it.

While K-Means is straightforward and efficient, it does have some drawbacks. For instance, it is dependent on the initial location of the centroids and can only converge to a local minima. Other partitional algorithms, such as PAM and hierarchical, offer alternatives with varying properties and may be more suitable for particular datasets or use cases.

What is Partitional Clustering with K-Means?

Partitional clustering with an algorithm such as K-Means divides a set of data into k non- overlapping clusters. The data points are then iterated over to the closest centroids and the centroids are updated until convergence. The selection of k and algorithm may depend on the type of data and the particular objectives of the cluster analysis.

2.2.2.1 K-Means

K-Means is a widely used unsupervised learning algorithm that divides a set of data into K different, non-interrelated clusters. Every data point in a set of data is assigned to a cluster based on its closeness to the centroid of the cluster. Here’s how it works:

Initialization:

1. Choose the number of clusters, K, that we want to partition the data into.

2. Randomly select K initial centroids. These centroids are often chosen as random data points from the dataset or using some other initialization strategy.

Assignment:

3. For each data point in the dataset, calculate its distance (usually Euclidean distance) to all K centroids.

4. Assign each data point to the cluster associated with the nearest centroid. In other words, a data point is assigned to the cluster whose centroid it is closest to.

Update Centroids:

5. Recalculate the centroids of each cluster as the mean (average) of all data points assigned to that cluster. This step involves taking the average of the data points' feature values along each dimension.

Repeat:

6. Repeat the assignment and centroid update steps iteratively until one of the stopping criteria is met:

- Convergence: The centroids no longer change significantly (i.e., they remain relatively stable from one iteration to the next).

- Maximum number of iterations is reached: We can set a predefined maximum number of iterations to limit the algorithm's runtime.

Result:

7. After convergence, we have a partitioned dataset with K clusters, and each data point belongs to one of these clusters.

Notes and Considerations:

- The initial centroid selection can have an impact on the final cluster result. Various initialization methods can be applied to improve convergence as well as cluster quality.

- The number of clusters is sensitive to K. Different values of K may need to be tested and techniques such as the elbow method and silhouette analysis may be used to find the optimum number of clusters.

- K-Means tends to converge to local optimization, which means that the final cluster result may not be optimal worldwide. Several runs with different initialization methods can help alleviate this issue.

- If the data has different scales, it is important to normalize or standardize the data.

- Other algorithms may be more suitable for complex cluster shapes such as DBSCAN and hierarchical clustering.

K-Means is one of the simplest and most efficient cluster algorithms, but it is based on the assumption that clusters are spherical and of the same size.

So, in a nutshell, K-Means is a popular partitional algorithm that divides a set of data into K clusters. It divides the data points into the nearest centroid and updates the centroid until convergence.

2.2.2.2 ISODATA

ISODATA (pronounced "Iterative self-organizing data analysis technique") is a data-cluster algorithm used to divide a dataset into different clusters. It is a variation of the K (K-Means) algorithm. Originally, ISODATA was used for remote sensing, image analysis, and data mining, but it has also been used in other areas such as pattern recognition, machine learning, and more.

ISODATA's main goal is to automatically figure out how many clusters there are in the data. This is different from the K (K) algorithm, which requires we to specify how many clusters we want to have in advance.

Here are the main steps involved in the ISODATA algorithm:

Initialization:

1. Start with an initial guess for the number of clusters, K.

2. Randomly select K initial centroids. These centroids are often chosen as random data points from the dataset.

3. Define other parameters such as the maximum number of iterations and a minimum number of data points in a cluster.

Iteration:

4. Assign each data point to the cluster whose centroid it is closest to, typically using Euclidean distance.

5. Calculate the new centroids for each cluster as the mean (average) of all data points assigned to that cluster.

6. Merge clusters that have a small number of data points (below a specified threshold).

7. Split clusters that have a large variance (if the variance exceeds a certain threshold).

8. Repeat steps 4-7 iteratively until one of the stopping criteria is met:

- Convergence: The centroids no longer change significantly (i.e., they remain relatively stable from one iteration to the next).

- Maximum number of iterations is reached: We can set a predefined maximum number of iterations to limit the algorithm's runtime.

- The number of clusters reaches the desired range (this is typically determined by the user or based on certain criteria).

Result:

We now have a cluster-rich dataset after convergence, and our number of clusters can be calculated by the algorithm on the basis of the data properties.

Advantages of ISODATA:

ISODATA has the following features:

- Automatic cluster identification: ISODATA automatically determines the number of clusters.

- Dynamic cluster adjustment: ISODATA dynamically adjusts the number of clusters during the clusterization process.

- Adaptability: ISODATA is able to work with clusters of different sizes and shapes by combining and dividing them.

Robustness: Compared to K-Means algorithms, ISODATA has less sensitivity to initializations.

Drawbacks and Considerations:

- ISODATA can be very demanding on our computer, especially if we have a lot of data points and features to work with.

- The performance of ISODATA depends on the parameters we select, such as the number of clusters we want to start with, the thresholds we want to merge and split, and the stopping criteria we want to use.

- The clusters we get from ISODATA might not match up with what we know about the data, and we may need to fine-tune the parameters.

In short ISODATA is a K-Means-like algorithm that automatically calculates the number of clusters, and it can iteratively merge and split clusters during the.

2.2.2.3 Partitioning around Medoids (PAM)

Partitioning around medoids (or PAM) is an algorithm used to divide a dataset into different clusters. It is similar to other classifying algorithms such as K-Means or ISODATA. However, PAM differs from K-Means in that it uses medoids rather than centroids. This allows it to work with distances other than just Euclidean distances.

PAM is sometimes referred to as k-medoids or k-medoids. It is especially useful in datasets where the average centroid is not a valid representation of the cluster. It is also useful when we want to be robust to noise and outlier.

Here are the key steps of the Partitioning Around Medoids (PAM) algorithm:

Initialization:

1. Choose the number of clusters, K, that we want to partition the data into.

2. Randomly select K initial data points as the initial medoids.

Assignment:

3. Calculate the distance to each K medoid in the dataset using a selected distance metric (Euclidean, Manhattan or other distance measures are popular).

4. Assign each data point to the cluster represented by the nearest medoid.

Update Medoids:

5. For each cluster, compute the total distance between the medoid and all other data points within that cluster.

6. Select the data point within each cluster that minimizes this total distance. This data point becomes the new medoid for that cluster.

Repeat:

7. Repeat steps 3 to 6 iteratively until convergence or a predefined stopping criterion is met. Convergence occurs when the medoids no longer change.

Result:

Once the algorithm converges, we have a partitioned dataset with K clusters, where each data point belongs to the cluster represented by its nearest medoid.

Advantages of Partitioning Around Medoids (PAM):

1. Outlier sensitivity: PAM is more sensitive to outliers than K-Means due to the fact that actual data points are used as medoids instead of means (which can be affected by extreme values).

2. Coverage: PAM can be used with a wide variety of distance metrics, allowing it to be used with a variety of data types.

3. Representation: Medoids are real data points in the dataset, meaning they represent their respective clusters.

Drawbacks and Considerations:

1. Computational complexity: PAM may be more computationally demanding than K-Means, particularly when the data set is large

2. Sensitivity to initial medoid choice: Similar to K-Means the selection of initial medoids may influence the final clustering outcome

3. Scalability: Due to its computational demands, PAM may not be suitable for large datasets. Various initialization methods are available to mitigate this.

In short, partitioning around Medoids is a centroid-like algorithm that uses actual data points (medoids) and various distance metrics to group a set of data into K clusters.

PAM can be used to group datasets that contain outliers or where the mean is not a good fit for a cluster.

2.2.3 Distance Based Clustering

Distance-based cluster, also called proximity-based cluster, similarity-based cluster, or distance-based cluster, is a type of cluster that divides a dataset into groups based on the closeness or distance of the data points. The basic idea behind distance-based cluster is to group close-to-nearest data points together in a near-nearest or far-from metric.

Distance-based cluster methods are used in many different areas, such as data analysis and machine learning, as well as data mining. Below are some of the most popular distance-based cluster methods:

1. Hierarchical Clustering:

- Hierarchical clustering is the process of building a tree-like network of nested clusters (dendrogram). Each data point is treated as a single cluster, and clusters are merged or separated based on their closeness or distance from each other.

- Hierarchical linking methods include single linking (minimum pairwise length), complete linking (maximum pairwise length), and average linking (average pairwise length).

2. Density-Based Clustering (DBSCAN):

- Density-based cluster methods, such as DBSCAN, determine clusters based on the number of data points in a given feature space.

- DBSCAN defines a cluster as a set of core points, defined as having a minimum number of adjacent points within a given radius, and joins them together to form a cluster. Data points that aren't core points, but are adjacent to core points, are considered part of a cluster.

3. OPTICS (Ordering Points to Identify the Clustering Structure):

- Optics is an add-on to DBSCAN that provides a hierarchical grouping result.

- Optics calculates the reachability distance of each data point, revealing hierarchical relationships between clusters or subclusters.

4. Self-Organizing Maps (SOM):

- Self-Organizing Maps are a type of artificial neural network used for clustering and visualization.

- SOMs map high-dimensional data onto a lower-dimensional grid and organize similar data points closer to each other in the grid.

5. Agglomerative Clustering:

- Agglomerative cluster analysis is a top-down hierarchical approach to cluster analysis. Each data point is treated as its own cluster and the closest clusters are grouped together in an iterative manner until a stopping point is reached.

- The grouping of clusters is typically based on distance measurements, such as Euclidean distances or other measures of similarity.

6. K-nearest Neighbours Clustering (KNN):

- K-nearest neighbours clustering assigns each data point to one of K clusters based on the majority class of its K nearest neighbours.

- It is commonly used for density-based clustering in high-dimensional spaces.

7. Fuzzy C-Means (FCM):

- Fuzzy c-means is a variant of K-Means that assigns clusters to data points probabilistically. This allows data points to be part of multiple clusters with different membership levels.

8. Hierarchical Fuzzy Clustering:

- Hierarchical fuzzy grouping is a combination of hierarchical grouping and fuzzy grouping that enables the hierarchical representation of fuzzy groups.

Distance-based methods are flexible and can be used for a wide range of data and use cases. The selection of a particular distance metric or similarity metric depends on the data type and the purpose of the clustered task.

2.2.3.1 Nearest Neighbour Clustering

Nearest Neighbor Search (NNS) or Nearest Neighbor Cluster (NNC) is a way of grouping data points based on how close or how far away they are from each other. The idea behind NNC is to group data points together so that each data point is closer to its nearest neighbour than to data points that are not part of a cluster. NNC is used in many different applications, such as data analytics, recommendation systems, anomaly detection, etc. There are several variations and algorithms associated with NNC.:

1. Single Linkage Clustering:

- In single-link clustering (also known as single-link method), each data point is initially its own cluster. Clusters are then merged iteratively by linking both clusters to the nearest pair of data points.

2. Complete Linkage Clustering:

- Complete-link method, also known as complete-cluster method, begins with every data point as a cluster. The two clusters are connected to the nearest pair of data points.

3. Average Linkage Clustering:

- The average distance between two clusters is determined by the average distance between each pair of data points in the cluster. Clusters are grouped according to the smallest average distance.

4. Centroid-Based Clustering:

- In centroid-based cluster, each cluster represents its data point centroid (the average or centre of its data points). Data points belong to the centroid cluster whose data points are closest to it. Centroid-based cluster is similar to nearest neighbor clustering.

5. Nearest Neighbor Chain Algorithm:

- The nearest neighbor chain algorithm is a method for creating clusters by connecting data points that are nearest neighbours of each other in a specific order.

6. DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- Density-based cluster analysis (DBSCAN) is an algorithm that classifies data points into clusters based on their proximity and density. DBSCAN clusters are defined by having each data point within a certain distance from at least a certain number of data points.

7. OPTICS (Ordering Points to Identify the Clustering Structure):

- OPTICS Computes Reachability Distance Between Data Points to Find Clusters and Subclusters. OPTICS is a DBSCAN extension that generates a hierarchical cluster result.

8. Shared Nearest Neighbor Clustering:

"SNN" stands for "Shared Nearest Neighbor." It’s a method of grouping data points by the number of neighbours they have in common. Data points that have more than one neighbour in common are grouped together.

9. K-Nearest Neighbours Clustering (KNN):

- K-Nearest Neighbourhoods (K-NNBs) Stacking assigns a data point to a K-nearest neighbour cluster based on its majority class of K NNBs. It is most commonly used in density-based clusters in high dimensional spaces.

The closest neighbour algorithm depends on the data type, the desired nearest neighbour structure, and the objectives of the analysis. Each algorithm has its own strengths and may be better suited to different data types and use cases.

2.2.4 Fuzzy Clustering

Fuzzy Clustering is an algorithm that allows a data point to belong to more than one cluster with different levels of membership or the probability of belonging to one cluster. This is different from traditional hard-cluster methods (such as K-Means) where a data point only belongs to one cluster.

FCM is the most widely used fuzzy C-means algorithm. In FCM, each data point has a membership value that indicates how much it belongs to a cluster. This allows for more flexible and more nuanced approach to clusters, especially when data points have ambiguities or overlap.

Here’s how fuzzy clusters work, especially using FCM:

1. Initialization:

- Select the cluster number, K, into which we want to group the data.

- Initialize the cluster centroid(s) for each cluster.

- Select the fuzzy parameter, often called "m," which specifies the fuzzyness of the clustered data. A typical value of m is higher than 1.

2. Membership Assignment:

- Each data point has a membership value for all K clusters that is calculated by comparing its similarity to the centroid of each cluster. Common metrics for this are Euclidean distance and other distance measures.

- For a single data point, the membership value is equal to 1, with higher values indicating a higher degree of membership for a cluster, and lower values indicating a lower degree of membership.

3. Update Cluster Centroids:

- Recalculate cluster centroid values using membership values. Each centroid is recalculated and updated as weighted average data points, with the weights based on membership values.

4. Iteration:

- Repeat the membership assignment steps and the centroid update steps one at a time until convergence, which is reached when there are no significant changes in membership values or centroids, or when a pre-defined stopping condition is met.

5. Result:

- After convergence, we have a partitioned data set with K fuzzy clusters. Each data point has member values that represent its association to each cluster. These member values can be interpreted as a probability or degree of association.

Advantages of fuzzy clustering (FCM) include:

- Flexibility: Data points can belong to multiple clusters at the same time, reflecting the randomness of real-world data

- Robustness: Fuzzy clusters can handle noisy data or data points that are on the boundary of multiple clusters

- Soft boundaries: soft boundaries are used to create soft boundaries between clusters. This is useful when hard boundaries are not suitable.

- Fuzziness Control: The fuzzy parameter (m) can be used to control the level of fuzzyness in the cluster. The level of fuzzyness can range from very sharp (m=1) to very sharp (m > 1).

Applications of fuzzy clustering include image segmentation, pattern recognition, customer segmentation, and any problem where data points may exhibit overlapping characteristics or belong to multiple categories simultaneously.

2.2.4.1 Gustafson Kessel

The GK algorithm is an extension of the fuzzy C-means algorithm, which was first introduced in 1979. It was developed as an improvement to FCM and is used to solve complex data structures. It works well when the data clusters have different shapes and sizes, and when the distribution is not spherical.

The following are some of the key characteristics and concepts related to the GK algorithm:

1. Membership Functions: Like FCM, GK multiplies each data point's membership value by probabilistic values to determine how much it belongs to each cluster.

2. Cluster Covariance Matrices: In FCM, the covariance matrices used to model clusters are spherical or circular. On the other hand, in GK, the cluster shapes can be modelled with any covariance matrix. The flexibility of GK makes it better suited for non-spherical cluster modelling.

3. Initialization: Like any other algorithm, GK needs to be set up from scratch. It begins by defining the cluster number (K) and then randomly initializes the cluster centroids with their corresponding covariance matrices.

4. Membership Update: The membership value for each data point is calculated on the basis of the cluster covariance matrix (Mahanobis distance). The Mahanobis distance takes into account the shape, orientation and size of clusters.

5. Cluster Centroid Update: Once the membership values are updated, cluster centroid and associated covariance matrix are re-weighted using weighted average data points, with the weights being determined by membership values.

6. Iteration: The membership assignment step, cluster centroid step and covariance matrix step are sequential steps until convergence, usually based on a convergence test.

7. Result: After convergence, the fuzzy distribution of the data is provided by the algorithm, indicating to what extent each data point is assigned to each cluster.

Cluster centroids and the covariance matrix describe the properties and properties of the clusters.

When dealing with datasets with different cluster shapes and axes, GK can be used to represent clusters in a more flexible way than FCM. FCM assumes spherical clusters, while GK can represent them in a more flexible manner.

GK can be used for a variety of applications, including:

-image segmentation

-medical data analysis

- pattern recognition

It is important to note, however, that GK is computationally more demanding than FCM. This is due to the fact that GK requires the use of the covariance matrix, which is a type of matrix used to measure the distance between two points.

The proper initialization and setting of the number of GK clusters is essential for achieving meaningful results.

2.2.5 Evolutionary Clustering

Evolutionary Clustering is a group of cluster-based algorithms based on the principles of evolutionary computing. These algorithms employ evolutionary concepts inspired by biology, like natural selection or genetic variation, in order to find optimal cluster solutions. The goal of evolutionary cluster methods is to find the optimal way to divide a set of data into clusters by changing the population of solutions over a number of generations. Let's take a look at how it works.:

1. Initialization: A cluster is a collection of possible cluster solutions. Every solution represents a potential cluster of data, usually by grouping data points together.

2. Evaluation: For each solution in the population, a fitness function is used to measure the fit of clusters in that solution to the data. This fitness function can be parameterized, for example, to minimize intra-cluster distances and maximize inter-cluster distances.

3. Selection: Solutions are chosen from the general population to act as the parents for the generations to come. Solutions with greater fitness levels tend to be chosen more often, similar to natural selection.

4. Crossover (Recombination): Parent solutions are paired to form new parent solutions (offspring). Cross-operators are used to pass data between parent solutions, resulting in potentially better parent cluster solutions.

5. Mutation: Some of these new solutions may have random mutations. Mutations introduce genetic variability into the population and allow for exploration of new areas of the solution domain.

6. Replacement: The new generation's descendants, along with a few legacy solutions, make up the next generation's population. The population size stays the same or is adjusted by a parameter.

7. Termination: Generations of solutions are iterated through by the algorithm. Evaluation, Selection, Crossover, Mutation, and Replacement are repeated. When the process terminates, it is determined by a number of conditions such as the number of generations, or the achievement of a successful clustering solution.

8. Result: The evolutionary clustering algorithm's best-fit solution is the best-fit solution in the last generation.

Key advantages of evolutionary clustering include:

- Evolutionary Clustering Algorithms
- Exploring a large solution space
- Discovering complex cluster structures
- Ability to adapt to various types of cluster criteria or fitness function

- Resilience to poor initializations
- Evolutionary Clustering Algorithms (ECAs)

ECAs can be very computationally demanding and may need to be fine-tuned to get good results. They are often used when traditional cluster methods are not suitable for a large dataset or the desired cluster objective is not clearly defined in advance. Examples of ECAs are:

GAC (Genetic Algorithm-Based Clustering)

PSO (Particle Swarm Optimized)

ACO (Ant Colony Optimized)

2.2.5.1 GA- based clustering

GA-based clustering is a cluster method that uses genetic algorithms (a type of evolutionary algorithm) to find and solve cluster problems for a dataset.

Genetic algorithms are based on natural selection and the principles of genetic evolution. They are commonly used in optimization and search algorithms.

In the context of clusters, the goal of GA-based methods is to find the best way to group data points together by evolving a collection of potential cluster solutions over generations.

Here is how GA-based clusters work:

1. Initialization: A collection of potential cluster solutions is created. Each cluster represents a possible cluster of the data set. Data points are grouped into clusters.

2. Fitness Function: A fitness function evaluates the fit of each cluster solution to the data. The fitness function is typically used to measure the quality of clusters. For example, the fitness function measures the minimum distance between clusters and the maximum distance between clusters. The fitness function helps in the optimization of clusters.

3. Selection: Cluster solutions are chosen to act as parents for the new generation. Solutions with better fitness scores are more likely to be chosen. This is similar to natural selection.

4. Crossover (Recombination): Parent solutions are paired to create new progeny solutions. Cross-over operators are used to transfer genetic information from parent to parent, potentially resulting in better cluster solutions.

5. Mutation: Mutation can lead to changes in some of the offspring solutions, which can bring more genetic diversity to the population and enable researchers to explore new areas of the solution.

6. Replacement: The offspring solutions, in combination with some of the previous generation's solutions, make up the next generation's population. The size of the population is usually fixed or parameterized.

7. Termination: Generations are iterated through the algorithm. Evaluation, Selection, Cross-Genesis, Mutations, and Replacements are repeated. Termination conditions (e.g., maximum number of generations, or satisfactory clustering solution) specify when the process terminates.

8. Result: The end-to-end result of GA-based cluster is the best-fit cluster solution of the past generation.

Key advantages of GA-based clustering include:

- Ability to explore a wide range of solutions, which can help uncover complex cluster structures
- Ability to adapt to various types of cluster criteria or function
- Ability to withstand poor initializations.

GA-based algorithms, on the other hand, require a lot of computing power and may need to be fine-tuned to get the best results, depending on parameters such as population size and mutation rate, as well as crossover operators.

GA-based algorithms are often used when traditional methods may not work well for large datasets, or when the goal of the cluster is not clearly defined beforehand.

2.2.5.2 Variable Length GA

Variable length genetic algorithm (VLGA) is an evolutionary cluster algorithm. VLGA is an extension of traditional genetic algorithms (GAs). The goal of VLGA is to solve cluster problems with variable or unknown cluster size.

In VLGA, not only does the algorithm find the best way to partition the data points into clusters, but it also dynamically determines the number of clusters. This flexibility is very useful in situations where the number of clusters cannot be predicted in advance or when the clusters have variable sizes or shapes.

Here is how VLGA clusters typically work:

1. Initialization: The algorithm begins with an initial set of possible cluster solutions. Each cluster solution represents a possible grouping of the data set. Data points are grouped into clusters.

2. Fitness Function: The fitness function evaluates the quality of each cluster solution. The fitness function evaluates the fit of the clusters to the data, considering both the cluster quality (intrinsic similarity) and the cluster count. The fitness function controls the search for the best clusters and their cluster sizes.

3. Selection: Cluster solutions are chosen to be the parents of the next generation on the basis of their fitness levels. Solutions with a higher fitness level are more likely to be chosen. This is known as natural selection.

4. Crossover (Recombination): Parent solutions are paired to create new progeny solutions. Cross-over operators are used to transfer genetic information from parent to parent, resulting in better cluster solutions and changes to cluster count.

5. Mutation: Certain offspring solutions may be altered randomly, known as mutations. These mutations introduce a variety of genetic elements to the population, which can lead to the development of new ideas, such as different cluster sizes.

6. Variable Length Encoding: The representation of solutions in VLGA clusters is usually encoded using variable-length chromosome. Variable-length encoding enables the algorithm to change the cluster count in every solution. For instance, a variable chromosome might contain information about data points assigned to clusters, clusters centroids and cluster count.

7. Replacement: The offspring solutions and some of the previous generation's solutions make up the next generation's population. The size of the population is usually fixed or parameterized.

8. Termination: Generations are iterated through the algorithm. Evaluation, Selection, Cross-Genesis, Mutations, and Replacements are repeated as the algorithm iterates through generations. When the process terminates, it terminates when the number of generations reaches a maximum or the successful clustering solution is reached.

9. Result: The end-to-end VLGA cluster result is the best-fit VLGA cluster solution of the past generation, including both the assigned clusters and the assigned cluster count.

One of the advantages of VLGA is that it can be adjusted for clusters. This makes VLGA a good choice for applications where we don't want to know how many clusters we'll have in advance.

VLGA also allows us to work with datasets with different cluster sizes and cluster shapes.

However, as with any genetic algorithm, it is important to note that VLGA requires a lot of computing power and may need to be fine-tuned for optimal performance.

2.2.6 Graph Based Clustering

Graph-based methods are used to divide data into meaningful clusters based on relationships or connections between two or more data points. These relationships are often represented as graphs or networks. Graph-based methods are especially useful for data where the pairwise similarity or distance between data points can

be easily found or calculated. The goal of graph-based methods is to find closely related subgraphs within a data graph. These subgraphs correspond to clusters of related data points.

Common Graph-based Clustering Methods and Concepts:

1. Spectral Clustering:

- Spectral cluster is one of the most common graph-based cluster methods. It uses the eigen of the graph's Laplacians to group the data. The process is as follows:

a. Create an affinity matrix (similarity matrix) based on the pairwise distance or similarity between data points

b. Calculate Laplacians of affinity matrix

c. Calculate Eigenvalues and Eigenvectors of Laplacians

d. Use eigen to perform K-Means (or other cluster method) to get the final clusters.

2. Normalized Cut and Ratio Cut:

- Normalized cut and ratio cut are spectral grouping criteria used to divide a graph into groups. The goal is to reduce the number of edges connecting different groups while increasing the number of edges within groups.

3. Minimum Spanning Tree Clustering:

- MST (minimum spanning tree) is a tool that creates a minimum spanning tree for a data graph. It then cuts the tree branches to form clusters.

- Clusters are formed when the edge weights of the branches are greater than a certain threshold.

4. Agglomerative Hierarchical Clustering:

- Hierarchical cluster methods can be used on graphs by gradually combining or separating clusters according to the strength of the connections.

- The result is a hierarchy of clusters that can be represented as dendrogram.

5. Community Detection:

- The goal of community detection algorithms is to detect dense subgraphs within a larger graph, commonly known as communities or modules. Examples of community detection algorithms include Louvain, Girvan-Nedman, and modularity approaches.

6. Density-Based Graph Clustering:

- Using density-based graph classification methods (DBSCAN, OPTICS), these methods extend the principles of graph classification to include graph data, classifying clusters according to the density of correlations between data points.

7. Graph-Cut-Based Clustering:

- Graph cuts are used to divide a graph into at least two sets of disjoint data.

- In this context, the minimum cut algorithm is used, as well as the normalized cut criteria.

Graph-based cluster can be used for a wide range of purposes, such as social network analysis, picture segmentation, document cluster, bioinformatics, etc. It can also capture intricate relationships between data points, and reveal hidden patterns within the data.

However, the selection of the similarity measure, the graph construction method and the clustering criterion can have a significant impact on the results. Furthermore, parameter fine-tuning is often necessary to achieve optimal cluster performance.

2.2.6.1 Minimum Spanning Tree Based Clustering

MST (minimum spanning tree) based clustering is a graph-based method that uses the minimum spanning tree to group a data set into clusters. This method is most commonly used when we have pairwise similarity or distance data points and want to find natural clusters based on relationships between the data points.

MST-based clusters are commonly used in image segmentation, networks analysis, and data

Here's how Minimum Spanning Tree Based Clustering typically works:

1. Construct the Similarity (Distance) Matrix: Determine the pairwise relationship or the distance between two data points based on the selected metric. The data points will be represented by a matrix.

2. Build the Minimum Spanning Tree (MST): Building a Minimum Spacing Tree (MST) from a similarity matrix.

A minimum spanning tree (MST) is a tree like subgraph that links all the data points in a data set while keeping the sum of the edge weights (likelihoods or distances) to a minimum. Various algorithms, including Kruskal’s or Prim’s

3. Threshold the MST: Set a limit value for the distance and similarity of the edges to be used when dividing the MST into groups. Any edges with weights lower than this limit are kept, while any edges higher than this limit are cut off, resulting in disconnected subgraphs.

4. Connected Components: Step 3 separated subgraphs are treated as clusters. Every connected element of MST is a cluster.

5. Result: After grouping, we have a division of the data into sub-divisions, each sub-divided by a sub-divided component of MST.

Key characteristics and considerations of MST-based clustering:

- MST-based cluster method is hierarchical because the threshold in Step 3 can be changed to produce different number of clusters by changing the threshold.

- The choice of similarity metric or distance metric as well as the threshold value have a big impact on the cluster results. Sometimes parameter tuning is required.

- MST-based clusters can be identified in different shapes and sizes.

This makes it suitable for data sets with complex structure.

- It is relatively fast compared to other cluster methods because constructing an MST takes O(E* log(V)).

E = number of edges, and log(V) = number of data points.

- Visualizing the cluster and the data provides insights into the underlying structure of the data.

MST clusters work best when clusters have tight interconnections and weak interconnections between clusters. MST clusters are used in a wide range of applications, from image segmentation to network analysis and biology where the aim is to discover meaningful relationships and patterns in data.

2.2.7 Density Based Clustering

2.2.7.1 DBSCAN

DBSCAN, or density-based special cluster, stands for “Density-based Spatial Cluster of Applications with Noise.” DBSCAN is one of the most widely used density-based classification algorithms. DBSCAN divides a dataset into clusters with different shapes and sizes. DBSCAN is different from other centroid-based classification algorithms such as K-Means because it doesn’t assume that clusters are round or equal in size. DBSCAN can detect clusters with irregular shapes and is especially useful for identifying clusters in noise and outliers’ datasets. The first version of DBSCAN was published in 1996 by Martin Ester (author of the paper on density-based spatial classification), Hans- Peter Kriegel (author of the second version of the paper, “Density based special classification of applications with noise”, Jörg Sunder (author of the third version of the paper), and Xiaoowei Xu (author of the fourth version).

Here are the key concepts and steps of the DBSCAN algorithm:

1. Density-Based Clustering:

Clusters are defined by DBSCAN as clusters of dense regions separated by sparse regions.

A cluster is a collection of data points in which each point is “density-reachable” from all other points in the cluster.

2. Core Points:

- What is a core point? A core point is defined as a data point with at least a minimum number of MinPts (in MinPts) in a defined radius (in Epsilon or in ε). Basically, a core point has enough neighbours to be a cluster member.

3. Border Points:

A boundary point is a point within a core point’s ε radius but doesn’t have enough neighbours to qualify as a core point. A boundary point is on the edge of a cluster and is part of the cluster to which it is connected.

4. Noise (Outliers):

- A noise point (or outlier) is a data point that is not part of a cluster. It does not meet the definition of a core point or boundary point.

5. Algorithm Steps:

- DBSCAN begins with a random selection of unvisited data points. If the data points are core points, a new cluster is created.

- The cluster is then expanded by the algorithm by adding all density-reachable core points and their boundary points.

- The process continues until there are no more core points to add to the cluster and the algorithm starts a new cluster by selecting another unvisited cluster.

- This process continues until all data points are assigned to clusters or are identified as noise.

6. Result:

- Once DBSCAN is executed, we have a collection of clusters, each one represented by a collection of data points, and noise points (outsiders).

Key advantages of DBSCAN include:

- Robustness to noise and outliers.

- Ability to discover clusters of arbitrary shapes and sizes.

- No need to specify the number of clusters in advance.

- Handles varying cluster densities effectively.

However, there are also some drawbacks to DBSCAN, such as sensitivity to parameter selection (ε, MinPts), and challenges when working with high dimensional data. DBSCAN requires parameter fine-tuning and pre-processing to achieve best-effort results. All in all, the DBSCAN cluster algorithm is a useful tool for a variety of applications, especially when working with large and noisy data sets.

2.2.7.2 OPTICS

OPTICS (pronounced “Ordering points to identify the Clustering structure”) is a density-based clustering algorithm that builds on DBSCAN. OPTICS can be used to find clusters in datasets of different densities and shapes. It is especially useful for dealing with noisy data and for finding clusters with irregular shape and size.

Here are the key concepts and steps of the OPTICS clustering algorithm:

1. Core Distance:

In the same way that DBSCAN defines neighbourhood relationships between two data points, OPTICS defines a distance threshold, which can be expressed as either a distance (ε) or an epsonity (Epsilon). For example, a data point, x, has a core distance, which is defined as its distance to its nearest neighbor, i.e., the nearest point of interest (i.e., the k-point). The k-point is a parameter that can be specified by the user, i.e., MinPts. This means that the core distance is the distance from the k-point to x within the parameter ε.

2. Reachability Distance:

- "Reachability distance" is the maximum distance between a data point (x) and another data point (y). It is the distance between the core distance (x) and the distance (y) between two data points (x and y).

- The reachability distance measures how easy it is for a data point (y) to be reached from the data point (x). It measures the distance between data points (x) and other data points (y) while staying within a range of ε.

- Reachability distance takes into account different densities of data points and efficiently handles clusters of various shapes and sizes.

3. Optics Ordering:

- Optics assigns an "ordering" value to each data point based on how far away it is from the center of a cluster. Data points with a lower ordering value are closer to the center of the cluster.

- The ordering values form an ordering plot, also known as a reachability plot, that is used to plot the order of the data points in the dataset.

4. Clustering Extraction:

- Optics does not immediately group data points into clusters.

- Instead, it creates a “reachability” plot that allows us to view the dataset’s cluster structure in a visual way.

- The reachability plot can be extracted from clusters by selecting “valley points” or points were reachability distance increases significantly.

- These valley points indicate the boundaries of clusters.

- The order of valley points in reachability plots shows the hierarchy of clusters.

5. Result:

- Once we have executed OPTICS and extracted the clusters, we will see that we have a hierarchical cluster distribution with different cluster sizes and density.

Key advantages of OPTICS clustering include:

- The ability to discover clusters with varying shapes, sizes, and densities.

- Robustness to noise and outliers.

- The flexibility to extract clusters at different granularity levels from the hierarchical structure.

- The lack of a need to specify the number of clusters in advance.

However, OPTICS may be computationally intensive, especially for large datasets. Careful parameter tuning and interpretation of the reachability plot are often necessary to obtain meaningful clustering results. OPTICS is a valuable tool for data exploration and clustering when dealing with complex and real-world datasets.

2.2.7.3 DENCLUE

DENCLUE is a density-based cluster detection algorithm that can be used to find clusters in high dimensional data sets. It’s an extension of density-based spatial cluster detection with noise (DBSCAN). It’s especially useful for datasets with different cluster density and irregular shape.

DENCLUE works by using a density estimation model to model the data distribution and find clusters by density peaks.

Here are the key concepts and steps of the DENCLUE clustering algorithm:

1. Density Estimation:

- Kernel density estimation is a method used by DENCLUE to estimate the density of each data point. The most common denominator for kernel density estimation is the Gaussian kernel. The density of a data point is the density of the data points around that point.

2. Hill Climbing:

- DENCLUE utilizes hill climbing to identify local density peaks within the density surface estimate.

A density peak is a potential cluster center.

- Hill climbing is the process of iteratively traversing a data point from a higher density neighbor until it reaches a peak.

3. Thresholding:

- A threshold parameter, commonly referred to as epsilon, is used to decide if a density peak is large enough to be classified as a cluster center.

- Peaks with a density lower than epsilon are considered noise. The selection of epsilon has an impact on the number and scale of clusters found.

4. Cluster Assignment:

- Once density peaks have been determined, data points are grouped together based on how close they are to density peaks.

- Data points are grouped around the closest density peak. The grouping is done by spreading assignments to adjacent points.

5. Result:

- After running DENCLUE, we obtain a set of clusters, each represented by a density peak and the data points assigned to it.

Key advantages of DENCLUE clustering include:

- Ability to discover clusters of varying shapes and densities.

- Robustness to noise and outliers.

- The flexibility to handle high-dimensional data.

- No need to specify the number of clusters in advance.

However, because of the density estimation step, it can be quite expensive to compute, especially for large, high-dimensional datasets. To get meaningful clustering results, it is often necessary to fine-tune parameters, such as the kernel and the ε value.

DenCLUE is a useful algorithm for examining large datasets with varying cluster density and shape.

Chapter 3

K-Means Clustering

The purpose of this chapter is to make us understand that cluster analysis is the process of dividing a collection of patterns (typically vectors) in a multidimensional space into clusters, where patterns in one cluster are related in a certain way and patterns in another cluster are different in a similar way. It is assumed that the patterns in the cluster belong to an n-dimensional Euclid's space Rn, and the measure of similarity is the distance between the two clusters.

K-Means clustering is an unsupervised algorithm that sorts the unlabeled data into different clusters. Let the set of patterns M be {x1, x2, x3..…. $x_{i;}$}, where x is the ith pattern vector. The number of clusters is defined by K, i.e., if K = 2, there will be 2 clusters, and if K = 3, there will be 3 clusters, etc. If the clusters are represented as C_1, C_2, C_3....$C_{k,}$ then,

P1 $C_i \neq \Phi$, for i = 1, ... K,

P2 $C_i \cap C_j = \Phi$ for $i \neq j$, and

P3 $\cup^{k}_{i=1} C_i = M$, where Φ represents null set.

There are a number of ways to group a data set. One of the most important ways to group data is by minimizing the sum of the intraclass distances. This principle is shown below.

1. Let be a set of K clusters of M.
2. Let $z_j = \frac{\sum x}{\frac{x \in ci}{\# Cj}}$ for j = 1,2, 3....K

   Where #C_j represents the number of points in C_j.
3. Let f (C_1, C_2, C_3....$C_{k,}$) = $\sum^{k}_{j=1} \sum_{x \in Cj} \| x - z_j \|^2$.
   f (C_1, C_2, C_3....$C_{k,}$) is referred to as the objective function of the clustering C_1, C_2, C_3....$C_{k,}$).
4. Minimize f (C_1, C_2, C_3....$C_{k,}$) overall such that C_1, C_2, C_3....C_k where C_1, C_2, C_3....C_k efficiently. All possible clustering's of M are to be considered to get the optimal C_1, C_2, C_3.... Ck.

This algorithm allows us to group the data into different clusters and provides a convenient way to find the categories of clusters in the unlabeled dataset without any training.

The algorithm is centroid-based, i.e., each cluster is centroid-related. The goal of the algorithm is to reduce the distance between the data points and their corresponding clusters as much as possible.

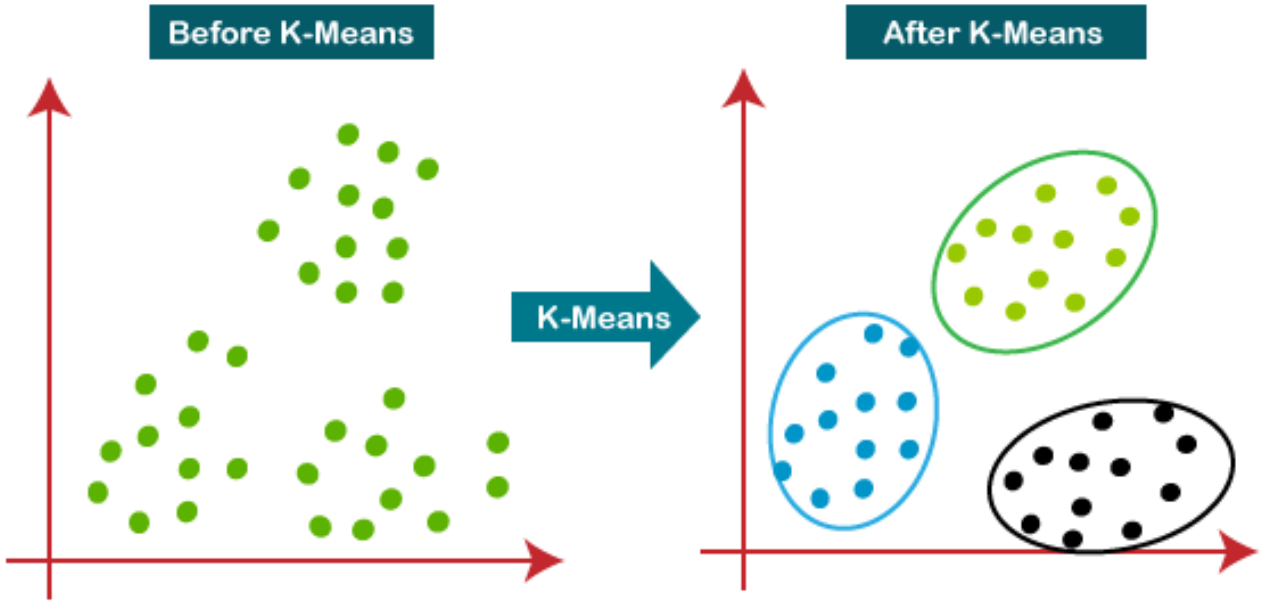
K-Means algorithm takes the unlabeled data as input, divides it into k-n clusters, and iterates until the best clusters are found. The value k should be pre-determined in this algorithm.

The K-Means clustering algorithm mainly performs two tasks:

- Determines the best value for K centre points or centroids by an iterative process.
- Assigns each data point to its closest k-centre. Those data points which are near to the particular k-centre, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-Means Clustering Algorithm:



The K-Means algorithm has been described as below:

Step 1: Select an initial cluster of configurations

Repeat

Step 2: Calculate cluster centres

$z_j j = 1,2,3.....K$, K of the existing groups.

Step 3: Redistribute patterns among clusters utilizing the minimum squared Euclidean distance classifier concept:

$x_i \in C_j$ if $|| x_i - z_j ||^2 < || x_i - z_l ||^2$

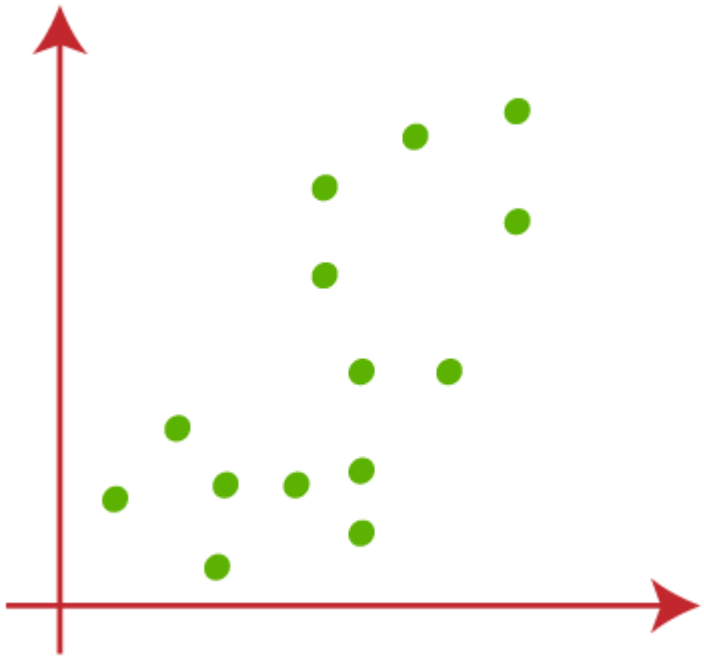
$\forall l \in \{1,2,3 ..... K\}, l \neq j.$

Until (there is no change in the clusters)

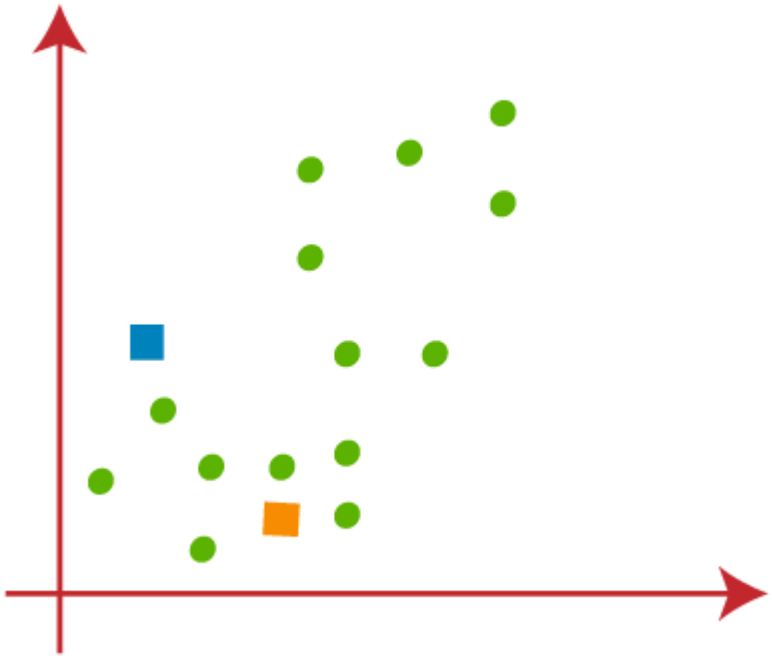
End (KM)

K-Means behaves differently depending on the number of cluster centres we specify, the number of initial cluster centres we select, and the geometric properties of our data.

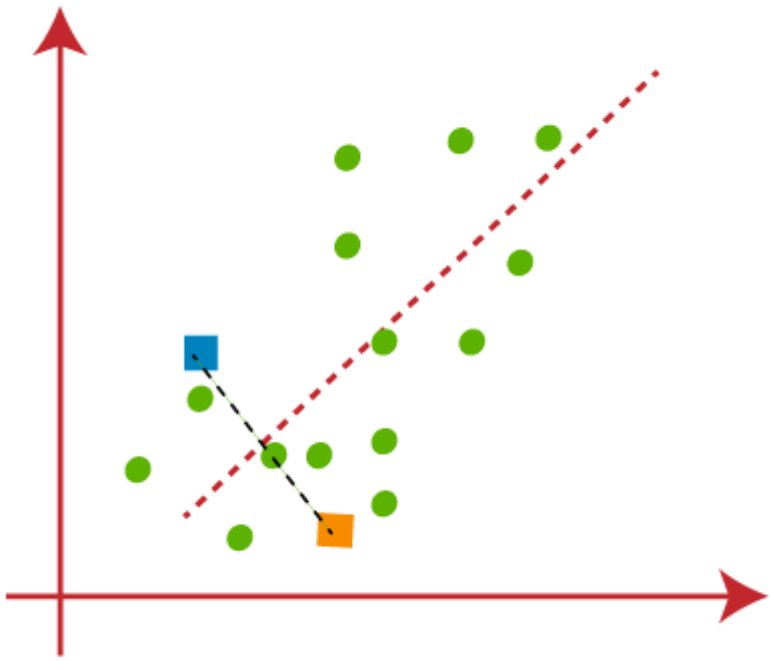
We can expect K-means to produce acceptable results when our data has characteristic pockets that are relatively distant from one another. However, K-MEANS converges when we apply it to multiple data sets from a broad range of applications.



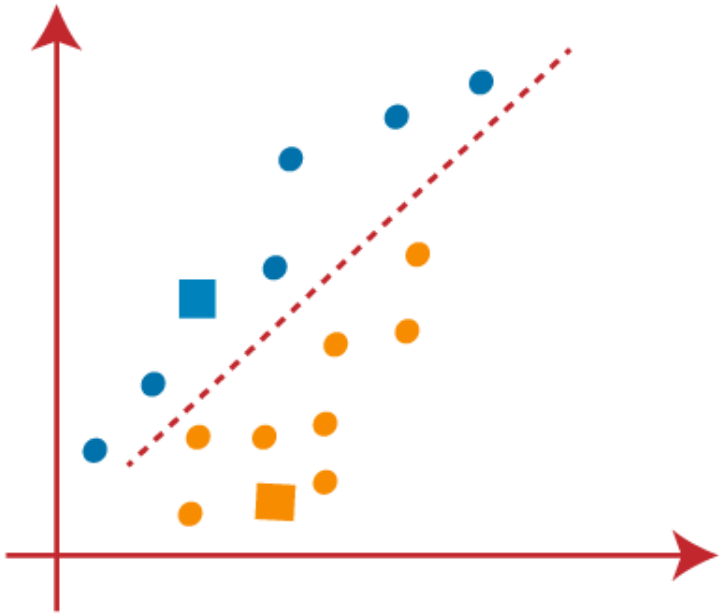
- Let's use the number k of clusters (K = 2) to define the dataset and divide it into different clusters. This means that here we are going to divide the dataset into two different clusters.
- To form the cluster, we need to select some random k points, or centroid. The points can be from the dataset or from any other point. Here, we are choosing the following 2 points as k points. These points are not part of the dataset. Take a look at the following image:



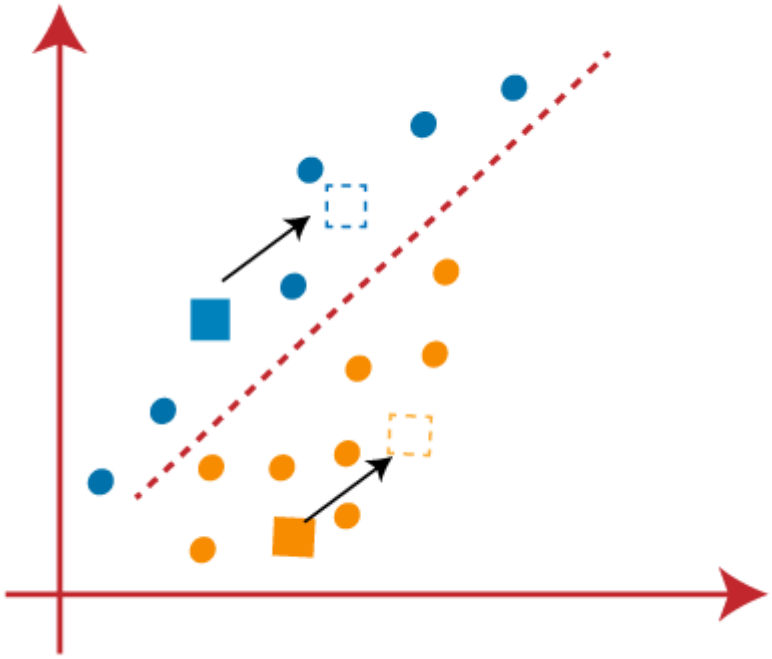
- Let’s now take each data point in the scatter plot and assign it to its nearest K-point (centroid). We’ll use some of the math we’ve learned to figure out the distance between points. Let’s draw a median distance between both centroids, as shown in the image below.:



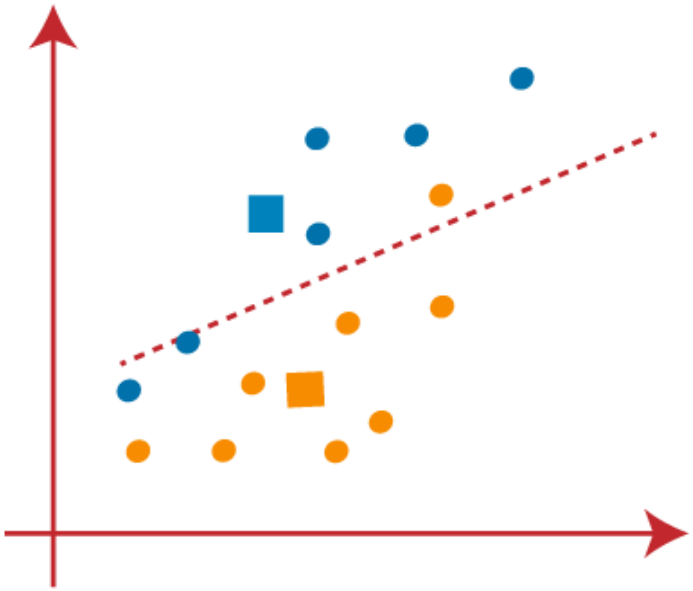
As we can see from the image above, the points on the left-hand side of the line correspond to the K1 (blue) centroid, while the points on the right-hand side correspond to the yellow (yellow) centroid. Let's color them blue and yellow for easy visualization.



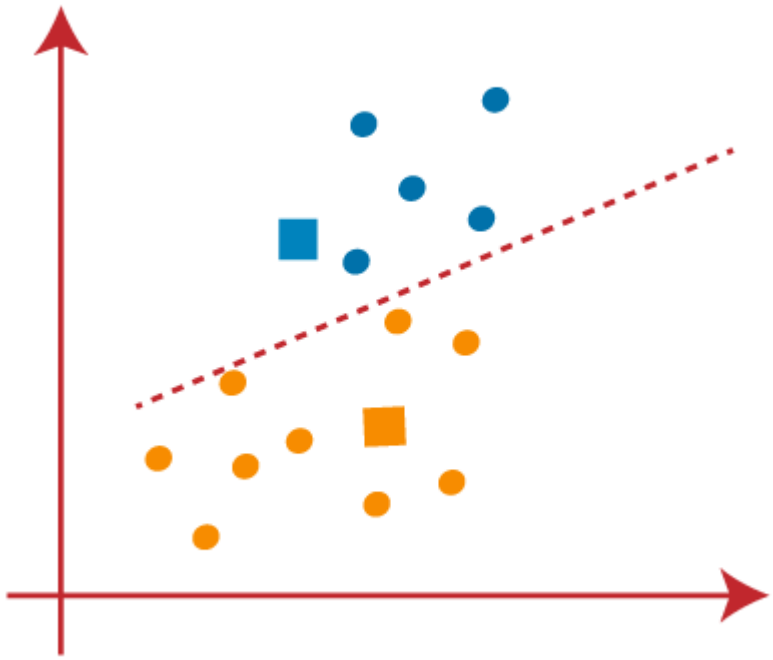
- Since we need to locate the nearest cluster, we will do the same thing again and select a different centroid.
- To select the new centroid(s), we will calculate the center of mass of the centroid(s) we want to select, and then we will find the new centroid as shown below.



- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:



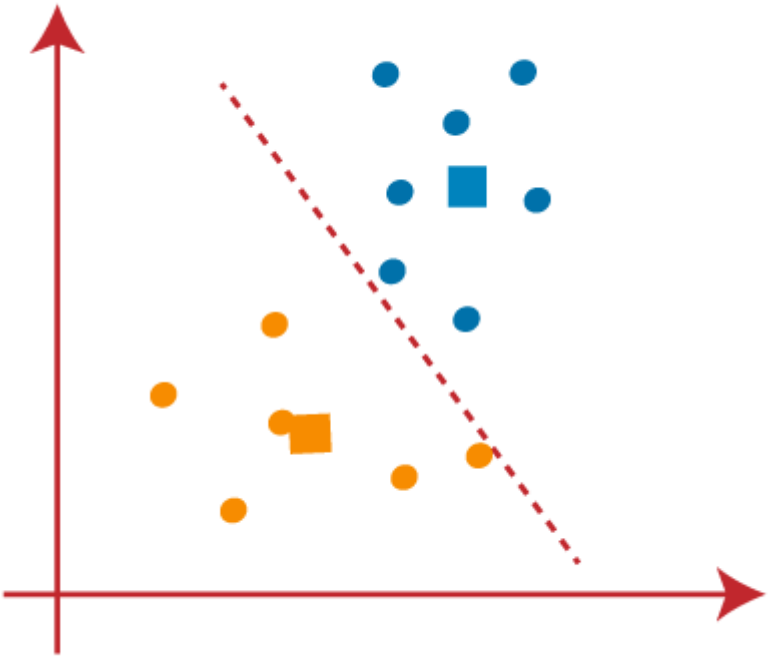
As we can see from the image above, one yellow dot is positioned to the left of the line and two blue dots are positioned to the right of the line. These three dots will be placed on the new centroid.



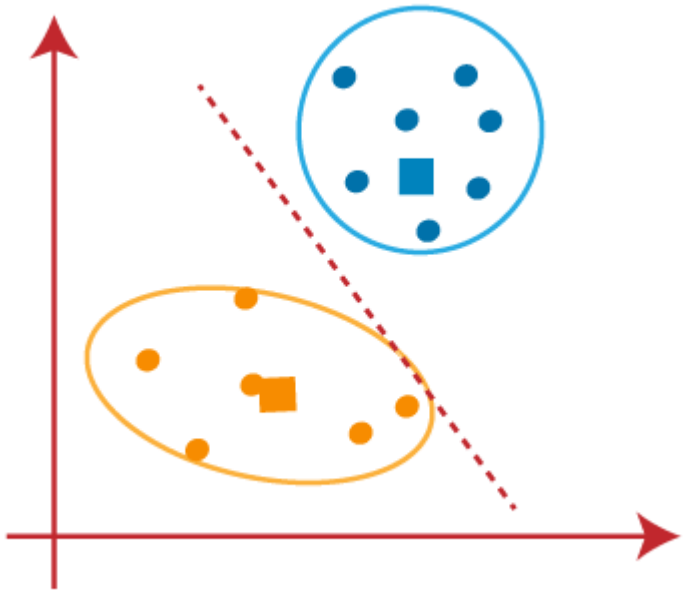
As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

- We will repeat the process by finding the centre of gravity of centroids, so the new centroids will be as shown in the below image:

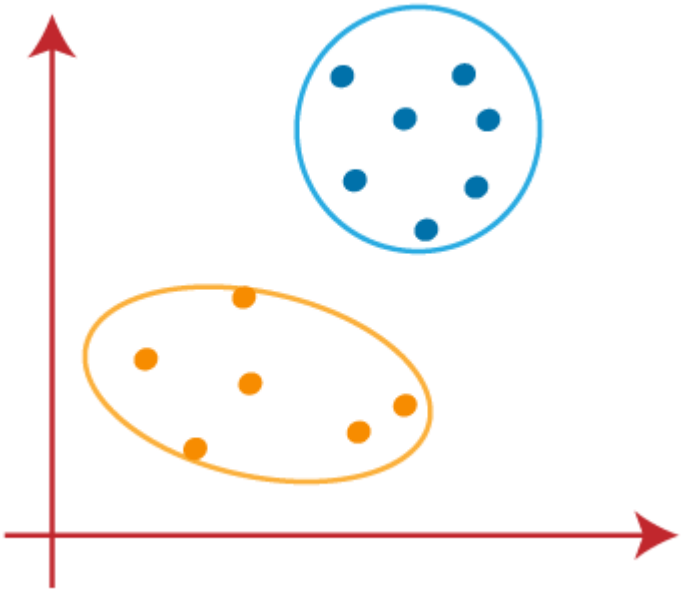
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



The performance of K-Means cluster algorithm depends on the number of highly efficient clusters it generates. However, selecting the best cluster number is a challenge. There are several ways to find the best cluster number, but here we will focus on the Elbow method, which is the most popular method for finding the best cluster number or K value.

The WCSS value is defined by the following formula: WCSS = WCSS (Cluster1Distance(Pi C1), Cluster2Distance(Pi C2), Cluster3Distance(Pi C3).

Here, WCSS is defined as the sum of the squares of each data point's distance to its centroid within cluster1 and the other two terms:

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

The elbow method is used to find the optimal cluster value by following the following steps:

- First, it performs the K-Means cluster distribution on the dataset for various K values (range from 1 to 10).

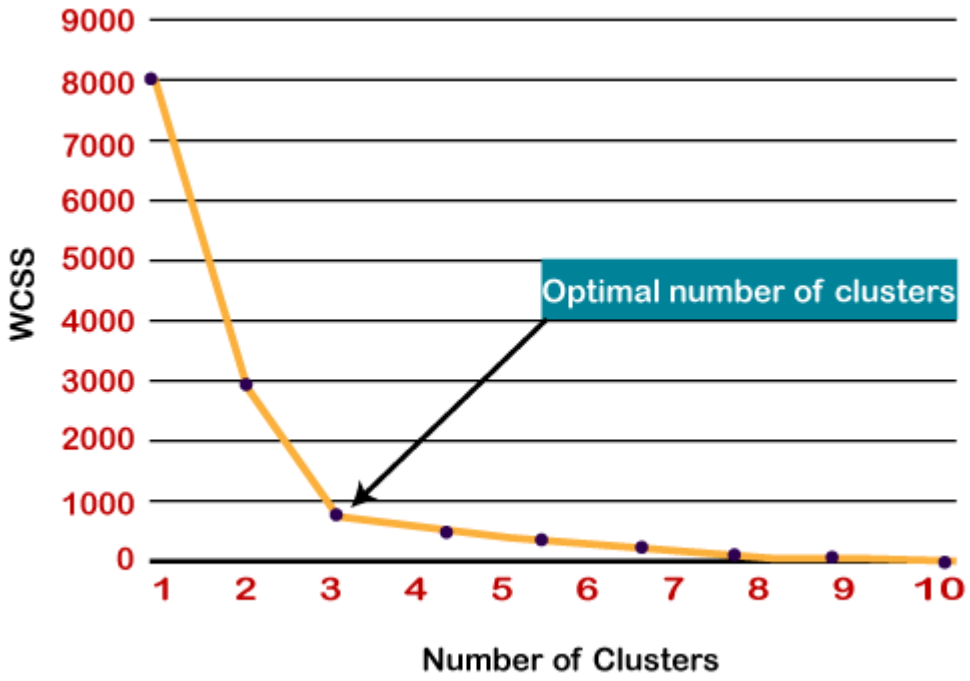
- Then, it calculates WCSS value for each K value.

- Next, it plots a curve between the calculated WCSS value and number of clusters.

- If the curve contains a sharp point that appears to be an elbow, then the elbow point is considered the best value for K.

Because the graph shows a sharp bend, it is called the "elbow" method.

The graph of the elbow method is as follows:



Chapter 4

A modified version of K-Means Clustering Method

This work presents a modified version of K-Means algorithm [1,3] for clustering where we look into the advantages that it has with respect to the traditional K means clustering. The main objective of this study is to show experimentally, that the new algorithm provides almost the same result as that of K-MEANS algorithm, while taking less execution time. In the K-Means algorithm, all patterns must be reassigned in each iteration, resulting in a large execution time for a practical dam set. However, it appears that points with longer distances from their cluster centres are more likely to be re-assigned to nearby clusters. To address this issue, a modified version of the algorithm has been proposed in this project report. This algorithm involves a fraction of points m, whose distances are greater than those of their cluster centres, being reassigned at each iteration, thus reducing the CPU time required for each iteration compared to K-Means. [3] The selection of p is intended to ensure that the cluster resulting from the reassignment of pm points is almost identical to the one obtained with K-Means, and that the number of iterations required for convergence should not be significantly greater than that of the proposed modified algorithm. Various p values have been used for the experiment, but the results of clustering are presented here. We are taking p = 0.6,0.3,0.1.

The project work is done on an algorithm which is a modification to K-Means to reduce the CPU time requirement. The proposed algorithm is referred to as MKM(p) (shortened to MKM(p)) where the value of $0 < p < 1$. After considering various examples, it has been demonstrated that the proposed modification is very similar (in many cases) to K-Means, and the CPU time required for the modification is significantly lower than that of K-Means. Additionally, the proposed modification has been evaluated in a fuzzy set theory clustering technique.[1]

It is important to note that the K-Means algorithm takes into account all the patterns that will be reassigned at every iteration. To achieve this, the distance between all the patterns and the cluster centres must be calculated.[1] However, it is not difficult to infer that patterns with a larger distance from their cluster centre are more likely to be reallocated to different clusters. This concept is employed in this project report to obtain a new algorithm, where only a fraction p ($0 < p < 1$) from the total number of patterns will be reassigned in each iteration.

In each iteration of the algorithm, the distance of each point from its cluster centre is calculated. Then, the pattern is sorted according to the distance. A fraction (p) of the number of patterns that are far from their cluster centres are considered for re-assignment. The re-assignment process is the same as the K-Means algorithm. This algorithm is called “MKM” (modified KM) algorithm.[3]

It is important to note that only MP (0<p<1) patterns are re-assigned in each iteration of this algorithm. If the number of iteration needed to reach the final configuration for K-Means and K-Means algorithm (p) is equal, then K-Means (p) algorithm would be about (1 / p).

A complete description of the new algorithm is given below:

Algorithm MK-MEANS (p) [(MKM (p)]:

Step I: Select an initial cluster configuration and read p (0 <p < 1).

Repeat

Step 2: Calculate cluster centers.

z_j, j = 1, 2, ..., K of the existing K groups.

Step 3: Sort all the patterns according to Euclidean distances from their cluster centers.

Step 4: Using the sorted array created in step 3, redistribute a fraction p of the total number of patterns (having larger distances from. their own cluster centers) among K clusters utilizing the minimum squared Euclidean distance classifier concept:

$X_i \in C_j$ if $\| x_i - z_j \|^2 < \| x_i - z_l \|^2 \ \forall l \in (1,2,3.....,K\}, l \neq j$

If *mp* is not an integer, then, the nearest integer exceeding the value of mp should be considered as the number of patterns to be redistributed.

Until (there is no change in cluster centers,

End (MKM)(p).

It has already been observed that MKM (p) is more rapid than K-Means; however, it is also necessary to compare the cluster generated by K-Means to MKM (p), in order to evaluate the performance. Numerous experiments have been conducted with both artificial and real-world data sets, and the results have been presented. It is pertinent to note that, in the context of this work, K-Means average performance and MK-Means mean performance are discussed on a given dataset.

Experimental Results

To assess the efficiency of the suggested clustering method, experiments have been conducted on synthetic dataset. Two sets of synthetic data have been generated for the experiments.

Experiment 1

Fig 1(a) shows the synthetic data distribution among two clusters having a data distribution of 500 generated randomly. The scattered diagram hence formed is depicted in Fig 1 one in the shape as a circular disc and the other in the form of a square.

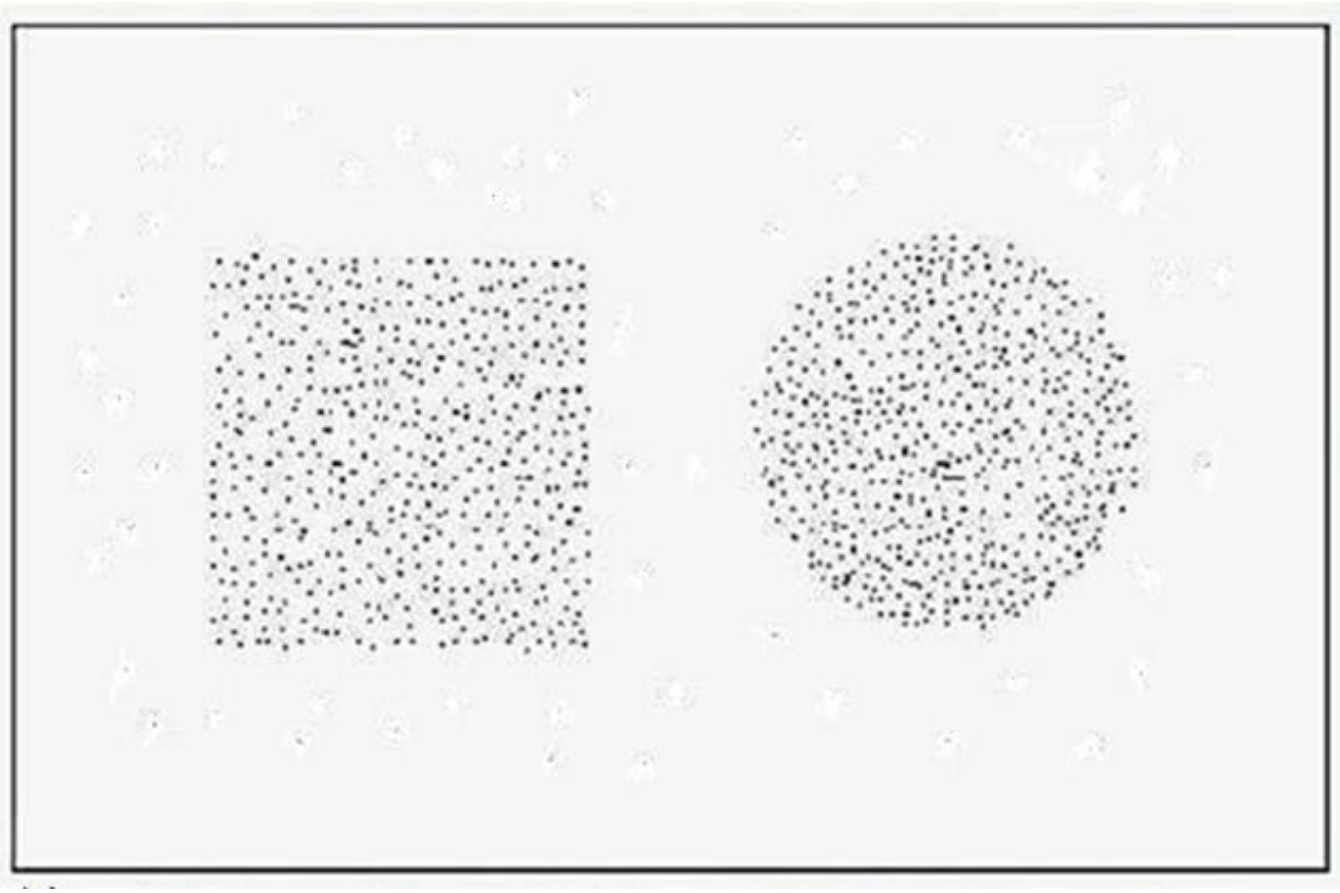


Figure 1(a): Scatter Diagram for Dataset 1

Fig 1(b) shows the clusters formed by the application of K-Means algorithm for the dataset shown in Fig 1(a). It can be seen that the K-Means algorithm has successfully extracted the two clusters present in the dataset.

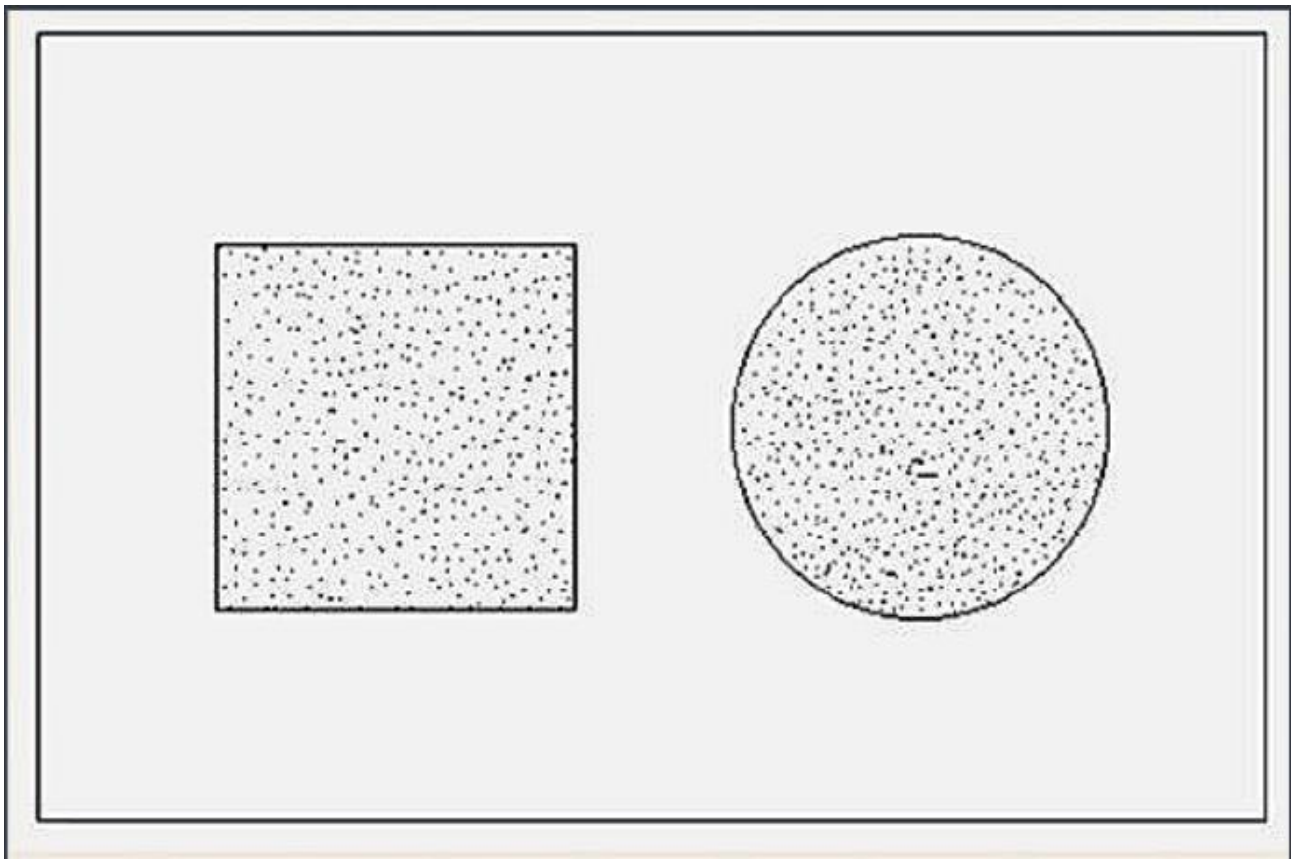


Figure 1(b): Cluster formed using K-Means algorithm

Fig 1(c) shows the clusters formed by the application of modified K-Means algorithm for the dataset shown in Fig 1(a) for all the three values of p, p = 0.6,0.3,0.1. It can be seen that the modified K-Means algorithm is equally capable of extracting the two clusters present in the dataset.

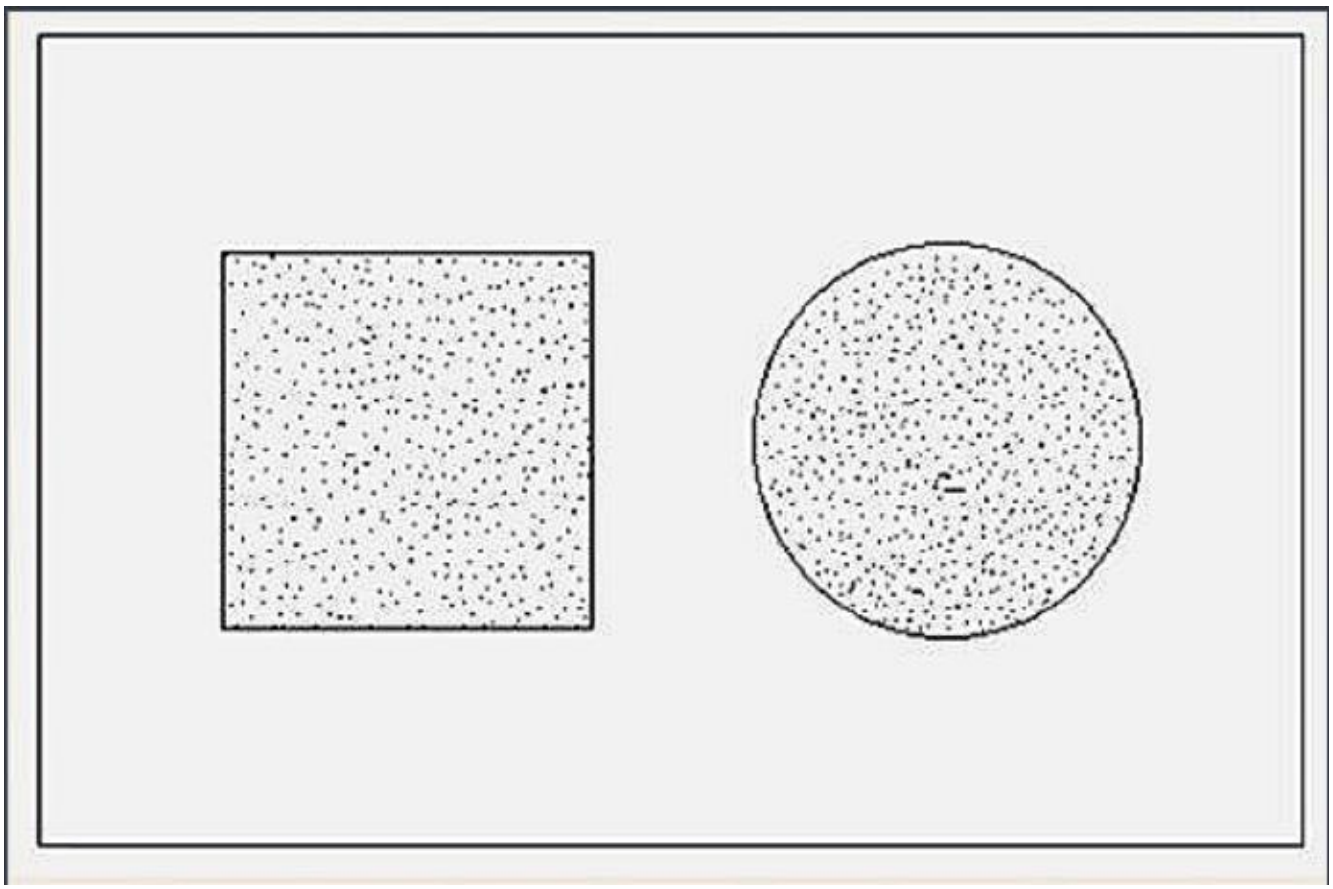


Figure 1(c): Cluster formed using modified K-Means

Experiment 2

Fig 2(a) shows the synthetic data distribution among four clusters having a data distribution of 500 generated randomly. The scattered diagram hence formed is depicted in Fig 2(a) two in the shape as a circular disc and the other two in the form of a square.

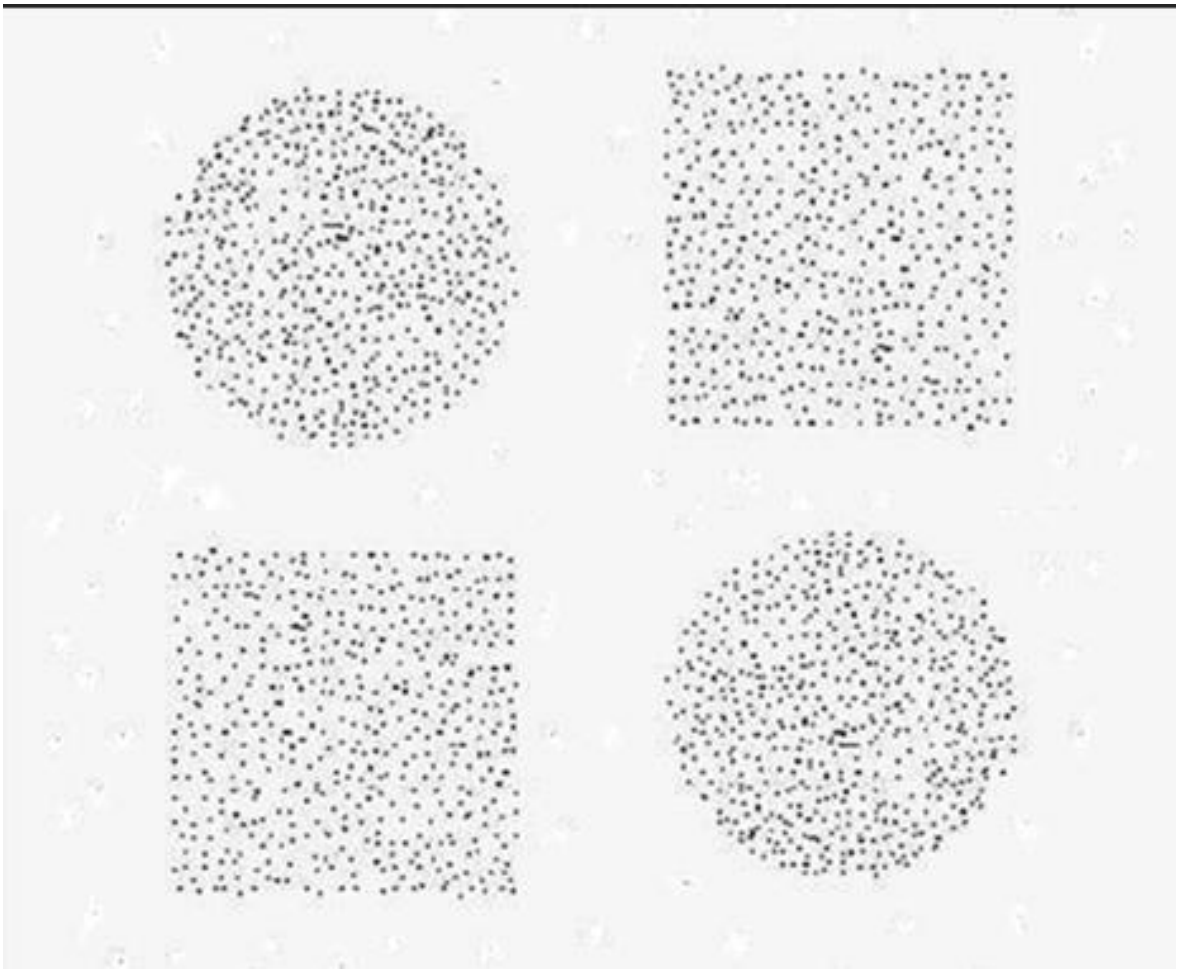


Fig 2(a): Scattered diagram for four clusters

Fig 2(b) shows the clusters formed by the application of K-Means algorithm for the dataset shown in Fig 2(a). It can be seen that the K-Means algorithm has successfully extracted the four clusters present in the dataset.

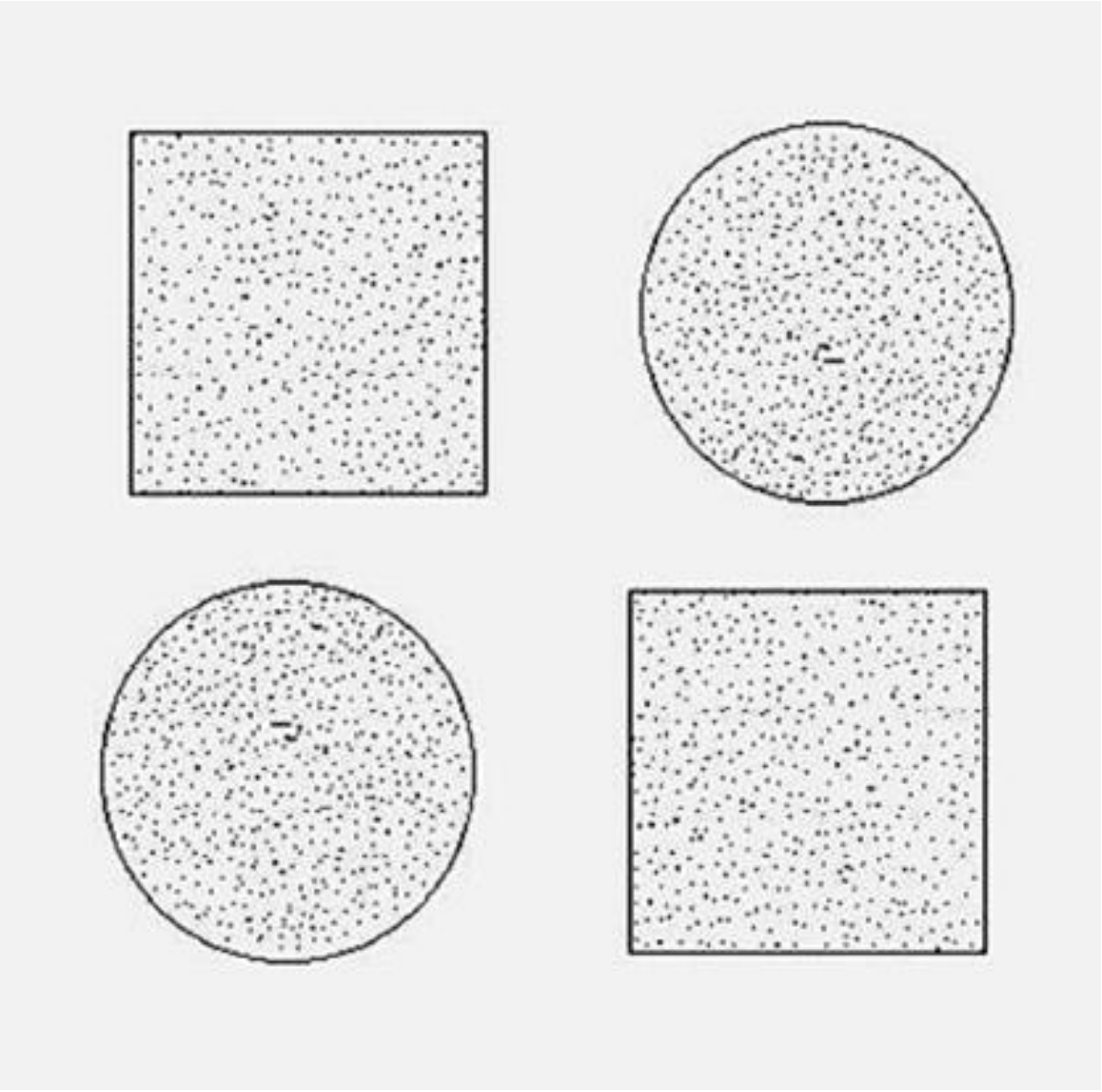


Fig 2(b): Cluster formed using K-Means for p = 0.6,0.3,0.1

Fig 2(c) shows the clusters formed by the application of modified K-Means algorithm for the dataset shown in Fig 2(a) for all the three values of p, p = 0.6,0.3,0.1. It can be seen that the modified K-Means algorithm is equally capable of extracting the four clusters present in the dataset.

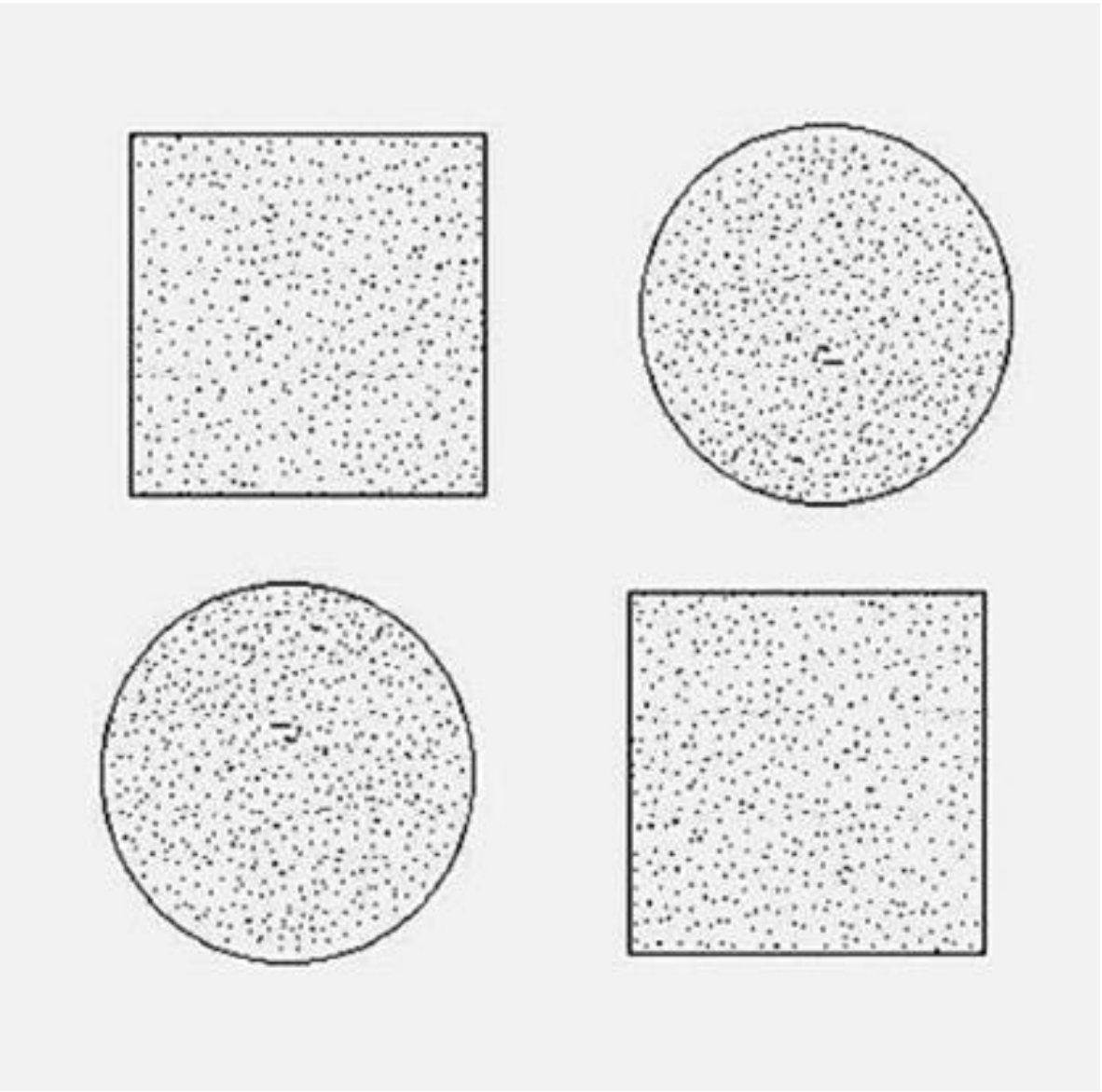


Fig 2(c): Cluster formed using modified K-Means

From the above experiments we can observe that the cluster formed using the modified K-Means produces exactly the same cluster as the K-Means for all the values of P considered for experimentation. The time taken to form the cluster is less in modified K-Means algorithm in comparison to the K-Means algorithm.

Conclusion

The purpose of this thesis is to apply and improve the K-Means cluster algorithm. The main goal was to evaluate the effectiveness of K-Means and its variants in solving real world data cluster problems, to evaluate their strengths and limitations, and to propose improvements where necessary. The execution time required for clustering any practical data using K-Means is quite large. It can be seen from the experimental results that modified K-Means(p) provides almost the same result as that of K-Means in case of synthetic data (for p=0.6, 0.3,0.10) while the latter algorithm is seen to be faster than the former one. The p value would be determined by the number of data points to cluster and the distribution of the data points. The clusters in the synthetic data are not overlapping and the data distribution shows characteristic pockets that are isolated from one another. We have found that the greater the distance between characteristic pockets in the data the lower the p value can be for MKM(p), making it faster while yielding very close results to K-Means.

This modified algorithm can be applied in the below fields:

Machine learning

Data analysis

Image segmentation

Customer segmentation

At the end of this thesis, we recognize that the exploration of clustering algorithms is just beginning. Future research directions will include:

- Advanced Initialization Schema: Developing new initialization methods that balance computational efficiency with convergence robustness
- Hybrid Cluster Approaches: Exploring hybrid cluster techniques that combine K-Means with other algorithm or deep learning model to leverage strengths of multiple approaches
- High-dimensional Data Handling: Reducing the difficulties of high dimensional data clustering by dimensionality reduction or feature engineering or other specialized clustering techniques
- Robustness & Outlier Handling: Finding ways to increase the robustness of clustered algorithms to outliers & noise to provide reliable results in real-world scenarios

This thesis has highlighted the importance of the K-Means algorithm and its variants in clustered applications. By recognizing their strengths, reducing their limitations, and looking for opportunities for improvement, we are contributing

to the continuous development of clustered algorithms, making them more relevant and effective in an era of data driven decision-making.

Bibliography

[1] N. Chowdhury and C. A. Murthy, "Multi-dimensional data clustering using a modified version of K-Means algorithm", *3rd International Conference on Pattern Recognition and Digital Techniques Calcutta*, India, pp. 60 – 67. 1993.

[2] Kodinariya, T.M. and Makwana, P.R., 2013. Review on determining number of Cluster in K-Means Clustering. *International Journal*, *1*(6), pp.90-95.

[3] Chowdhury, Nirmalya & Murthy, Chaitra. (1996). An Efficient Method for Hard and Fuzzy Clustering of Multi-dimensional Data. Journal of The Institution of Engineers. 77. 13 - 19.

[4] Velmurugan, T. and Santhanam, T., 2010. Computational complexity between K-Means and K-medoids clustering algorithms for normal and uniform distributions of data points. *Journal of computer science*, *6*(3), p.363.

[5] Ahmed, M., Seraj, R. and Islam, S.M.S., 2020. The K-Means algorithm: A comprehensive survey and performance evaluation. *Electronics*, *9*(8), p.1295.

[6] Ghosh, S. and Dubey, S.K., 2013. Comparative analysis of K-Means and fuzzy c-Means algorithms. *International Journal of Advanced Computer Science and Applications*, *4*(4).

[7] Coates, A. and Ng, A.Y., 2012. Learning feature representations with K-Means. In *Neural Networks: Tricks of the Trade: Second Edition* (pp. 561-580). Berlin, Heidelberg: Springer Berlin Heidelberg.

[8] Fard, M.M., Thonet, T. and Gaussier, E., 2020. Deep K-Means: Jointly clustering with K-Means and learning representations. *Pattern Recognition Letters*, *138*, pp.185-192.

[9] Yadav, J. and Sharma, M., 2013. A Review of K-mean Algorithm. *Int. J. Eng. Trends Technol*, *4*(7), pp.2972-2976.

[10] Celebi, M.E., Kingravi, H.A. and Vela, P.A., 2013. A comparative study of efficient initialization methods for the K-Means clustering algorithm. *Expert systems with applications*, *40*(1), pp.200-210.

[11] Jain, A.K., 2010. Data clustering: 50 years beyond K-Means. *Pattern recognition letters*, *31*(8), pp.651-666.

[12] Krinidis, S. and Chatzis, V., 2010. A robust fuzzy local information C-Means clustering algorithm. *IEEE transactions on image processing*, *19*(5), pp.1328-1337.

[13] Cao, L., Zhao, Z. and Wang, D., 2023. Clustering algorithms. In *Target Recognition and Tracking for Millimeter Wave Radar in Intelligent Transportation* (pp. 97-122). Singapore: Springer Nature Singapore.

[14] Suganya, R. and Shanthi, R., 2012. Fuzzy c-Means algorithm-a review. *International Journal of Scientific and Research Publications*, *2*(11), p.1.

[15] Fan, J.L., Zhen, W.Z. and Xie, W.X., 2003. Suppressed fuzzy c-Means clustering algorithm. *Pattern Recognition Letters*, *24*(9-10), pp.1607-1612.

[16] Nayak, J., Naik, B. and Behera, H., 2015. Fuzzy C-Means (FCM) clustering algorithm: a decade review from 2000 to 2014. In *Computational Intelligence in Data Mining-Volume 2: Proceedings of the International Conference on CIDM, 20-21 December 2014* (pp. 133-149). Springer India.

[17] Jipkate, B.R. and Gohokar, V.V., 2012. A comparative analysis of fuzzy c-Means clustering and k Means clustering algorithms. *International Journal Of Computational Engineering Research*, *2*(3), pp.737-739.

[18] Hung, M.C. and Yang, D.L., 2001, November. An efficient fuzzy c-Means clustering algorithm. In *Proceedings 2001 IEEE international conference on data mining* (pp. 225-232). IEEE.

[19] Cannon, R.L., Dave, J.V. and Bezdek, J.C., 1986. Efficient implementation of the fuzzy c-Means clustering algorithms. *IEEE transactions on pattern analysis and machine intelligence*, (2), pp.248-255.

[20] Xu, R. and Wunsch, D., 2005. Survey of clustering algorithms. *IEEE Transactions on neural networks*, *16*(3), pp.645-678.

[21] Na, S., Xumin, L. and Yong, G., 2010, April. Research on K-Means clustering algorithm: An improved K-Means clustering algorithm. In *2010 Third International Symposium on intelligent information technology and security informatics* (pp. 63-67). Ieee.

[22] Mohamad, I.B. and Usman, D., 2013. Research Article Standardization and Its Effects on K-Means Clustering Algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, *6*(17), pp.3299-3303.

[23] Madhulatha, T.S., 2012. An overview on clustering methods. *arXiv preprint arXiv:1205.1117*.

[24] Dhanachandra, N., Manglem, K. and Chanu, Y.J., 2015. Image segmentation using K-Means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, *54*, pp.764-771.

[25] Murtagh, F. and Contreras, P., 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), pp.86-97.

[26] Aggarwal, C.C. and Zhai, C., 2012. A survey of text clustering algorithms. *Mining text data*, pp.77-128.

[27] Chang, C.T., Lai, J.Z. and Jeng, M.D., 2011. A fuzzy K-Means clustering algorithm using cluster center displacement. *J. Inf. Sci. Eng.*, *27*(3), pp.995-1009.

[28] *K-Means Clustering Algorithm - JavatPoint*. (n.d.). www.javatpoint.com. https://www.javatpoint.com/K-Means-clustering-algorithm-in-machine-learning

[29] Ecosystem, E. (2022, May 17). Understanding K-Means clustering in machine learning. *Medium*. https://towardsdatascience.com/understanding-K-Means-clustering-in-machine-learning-6a6e67336aa1

[30] Fränti, P., & Sieranoja, S. (2019). How much can K-Means be improved by using better initialization and repeats? *Pattern Recognition*, *93*, 95–112. https://doi.org/10.1016/j.patcog.2019.04.014