To Study the performance of K-means Algorithm in Partitional Clustering

BY

SOUMYA NAG

University Roll Number: 002010504003

Examination Roll Number: M6TCT23016

Registration Number: 154169 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 **"To Study the performance of K-means Algorithm in Partitional Clustering"** is a bona-fide record of work carried out by SOUMYA NAG 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 **"To Study the performance of K-means Algorithm in Partitional Clustering"** was completed by **Soumya Nag**, University RollNo: 002010504003, Examination Roll Number: M6TCT23016, University Registration No: 154169 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 therein 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

ACKNOWLEDGEMENT

I express my honest and sincere thanks and humble gratitude to my expected teacher and guide Dr. Nirmalya Chowdhury for his exclusive guidance and entire support in completing and producing this term paper successfully. I am very much indebted to him for the constant encouragement and continuous inspiration that he was given to me. The above words are only a token of my deep respect towards him for all he has done to take my project to the present shape.

Finally, to convey my real sense of gratitude and thankfulness to all my friends and family members for their unconditional without which I would hardly be capable of producing this huge work.

(Soumya Nag)

Master of Technology

Roll No: 002010504003

Exam Roll No: M6TCT23016

Registration No: 154169 of 2020-21

Department of Computer Science & Engineering

Jadavpur University, Kolkata

Declaration of Originality and Compliance of Academic Ethics

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 : **Soumya Nag**

Roll Number : **002010504003**

Project Title : **To study the performance of K-means algorithm in partitional Clustering**

Signature With Date :

Contents

Chapter 1: Introduction

1.1. Introduction to Pattern Recognition

The identification and understanding of patterns or relationships within data is the focus of pattern recognition, a subfield of machine learning and artificial intelligence. These connections can be seen in a range of media, such as pictures, signals, words, and more. The creation of algorithms and methods that enable computers to automatically distinguish between various categories or classes of data based on such correlations is ultimately what pattern recognition entails.

Key Concepts:

1. **Feature Extraction:** The extraction of features is frequently the first step in pattern recognition. Features are aspects or qualities of the data that can be utilized to distinguish between various patterns. For instance, features like pixel values, textures, color distribution, edges, etc., can be used in picture recognition.
2. **Dimensionality Reduction:** Real-world data sets can be extremely complicated, which can lead to overfitting and computational issues. Techniques like PCA and t-SNE can be used to reduce the number of features while retaining the crucial data in order to lessen the complexity of the data.
3. **Classification Algorithms:** Different classifications are applied to the data points after the features are specified. K-nearest neighbours, support vector machines, decision trees, random forests, and neural networks are some of the most used classification techniques.
4. **Clustering:** The capacity to combine comparable data pieces is another essential component of pattern recognition. Data can be categorized using clustering without a specified set of classifications. The clustering techniques K-means, Hierarchical Cluster, and DBSCAN are utilized in pattern recognition.
5. **Supervised vs. Unsupervised Learning:**

   Pattern recognition learning can be divided into two categories: supervised learning and non-supervised learning.

   A model is trained on data with known classes (labelled data) in supervised learning. On the basis of fresh, unlabeled data, this model predicts the future after learning the patterns. A model learns patterns and structures using unlabeled input in non-labelled learning.

6. **Feature Selection:** Features are not created equally. Finding and using the features that are most pertinent to a given task is the aim of feature selection. This aids in enhancing model performance, bringing down complexity, and preventing overfitting.
7. **Neural Networks:** A subset of deep learning is machine learning, commonly referred to as deep learning. Particularly in the areas of image recognition, speech recognition, and deep learning neural networks (DNNs), deep learning has completely changed the field of pattern identification. DNNs are quite good at identifying intricate patterns, especially CNNs for images and RNNs for sequences.
8. **Applications:**

   There are several uses for pattern recognition, including:

   • **Image recognition**: recognizing individuals, objects, or scenes in pictures.

   • **Speech Recognition**: The conversion of spoken words into text.

   • **Text Classification**: Spam detection, sentiment analysis, and document classification.

   • **Biometric identification** using a person's face, iris, or fingerprint.

   • **Medical Diagnosis:** Finding diseases using signals and visuals from the medical field.

   • **Financial Fraud Detection**: Spotting odd trends in transactional behaviour.

   • **Robotics:** Giving machines the ability to perceive and communicate with their surroundings.

9. **Challenges:**

   The heterogeneity of the data, the issue of overfitting, the issue of noisy data, and the issue of the "curse of dimensionality" are what make pattern detection difficult. The creation of strong algorithms that are capable of making good adjustments to various scenarios is the aim of ongoing research.

1.1.1 What is Pattern Recognition?

The cognitive process of pattern recognition is used to identify and explain patterns, resemblances, or trends in data, information, or sensory inputs. It is a crucial component of human intellect and is employed in a wide range of

academic disciplines, such as psychology, neuroscience, computer science, and machine learning. The technique of using input data to recognize, categorize, or forecast new or ambiguous information is known as pattern recognition.

Pattern recognition involves a number of processes.

1. Data acquisition: The gathering of unprocessed data or data input for pattern identification. This can be in the form of pictures, sounds, text, or numbers.

2. Pre-processing: Cleaning and preparing data to eliminate noise, inconsistencies, or extraneous information is known as data normalization.

3. Ensuring that the patterns in the data are represented by particular features or attributes (feature extraction). These qualities, measurements, or properties may be pertinent to the analysis.

4. Feature Selection: Choosing the elements that are most crucial while eliminating those that are unnecessary or of little value. This lessens the complexity of your analysis and enhances the effectiveness of your algorithms for pattern detection.

5. Pattern Matching or Classification: This is the most fundamental form of pattern recognition. To compare the collected features to pre-defined models or patterns, algorithms and approaches are used. This entails identifying patterns in the data and categorizing it into various groups or categories.

6. Training and Learning: To grasp the patterns, machine learning and AI algorithms frequently need to be trained on labelled data. The algorithm modifies its own settings throughout training to better comprehend the patterns.

7. Testing and Validation: After training, the algorithm's performance is assessed using fresh, unobserved data. This stage's objective is to evaluate the algorithm's generalization and pattern recognition abilities.

8. Pattern Refinement: Test results may be utilized to modify or enhance the algorithm in order to increase accuracy or efficacy. Changing parameters or choosing other algorithms might be part of this.

9. Prediction or Decision-Making: The pattern recognition system can be used to predict, make decisions, or both after being trained and validated.

10. Feedback Loop: In some cases, the pattern recognition data created can be used to improve pattern recognition jobs going forward. With adaptive or machine-learning models, this is frequently the case.

Pattern recognition is used for the following in the real world:

- Image and Speech Recognition: This technology can be used to recognize objects, persons, or sounds in visual or aural media.

- Medical Diagnostics: The use of pattern recognition in medical imaging to identify irregular heartbeats or diagnose diseases like cancer.

- Financial forecasting: Analysing past data to project future share prices or market movements.

- Natural Language Processing: This skill enables chatbots and language translation by being able to understand and analyse human speech.

- Biometrics: Pattern recognition in fingerprint, iris, or facial scan data for security

- Autonomous Vehicles: How to identify pedestrians, other vehicles, and road signs to aid self-driving cars in navigating.

In essence, pattern recognition allows both humans and machines to analyse complex data and reach educated decisions by bridging the gap between raw data and insightful conclusions.

1.1.2 Pattern Recognitions Methods

1.1.2.1 Supervised Method

In supervised pattern recognition (SPR), an algorithm learns from labelled training data in order to make predictions or choices based on fresh, unlabelled data. In supervised pattern recognition, the algorithm is provided with the training data and the appropriate target labels, and it learns to map the data to the target labels based on the patterns it discovers in the trained data. An explanation of supervised pattern recognition and an example algorithm are provided below.

1. Data collection and labelling: Each data point in the dataset is given a label first. An email message dataset, for instance, categorizes each message as "spam" or "not spam."

2. Feature Extraction: Examples include word frequency for text data and pixel values or other extracted visual features for photos. The patterns from which the algorithm learns are represented by these attributes.

3. Dataset Splitting: The labelled dataset is divided into a training set and a testing set. The training set is used to train the algorithm, while the testing set is used to test the system on unlabelled data.

4. Algorithm Selection: Based on the type of data and the nature of the task, the supervised pattern recognition algorithm (SNCO) is chosen. The support vector machine (SVM) is one of the most frequently employed SNCOs.

5. Algorithm Training (Supervised Learning):

- The training data and their associated labels are sent to the algorithm.

- It gains the ability to build a model that associates the relevant labels with the features.

- It produces a hyperplane for SVM that divides data points from various classes while reducing the incidence of misclassifications.

6. Validation and Tuning: Using the testing set, the algorithm is examined. The algorithm's parameters can be altered to increase accuracy, and techniques like cross validation can be used to ensure 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: The performance of the algorithm is assessed using metrics. Among these metrics include, but are not restricted to:

Precision Proximity

Reconciliation

F1-score

Support Vector Machine (SVM) is an example of an algorithm.

A popular supervised learning technique for classification algorithms is the Support Vector Machine (SVM). It functions by locating the hyperplane most effective in separating data points from various classifications. The process is as follows:

1. Data Representation: Each feature vector in the input data is given a class label, and the input data is represented as feature vectors.

2. Hyperplane Construction: SVM identifies the hyperplane with the largest gap. The space is the separation between each class's closest points and the hyperplane.

3. Soft Margin and Kernel Trick: A "soft margin" permits some misclassification when the data cannot be separated. The data is changed into a higher dimensional space using the kernel method, which might make it simpler to separate.

4. Classification: Following the definition of the hyperplane, new points are arranged in accordance with whatever side of the hyperplane they fall on. The side specifies the class they expect.

5. Optimization: The SVM aims to maximize the margin while minimizing the classification error and adhering to the soft margin limitations.

One supervised pattern recognition algorithm among several is SVM. Other supervised pattern recognition techniques, including decision trees, random forests, K-nearest neighbours, and neural networks, may be employed, depending on the complexity of the task and the supervised pattern recognition methodology being used.

1.1.2.2 Semi-supervised Method

Semi-supervised learning is a blend of supervised and unsupervised learning, and the terms "supervised" and "unsupervised" are

interchangeable. To generate models that are more dependable and accurate, semi-supervised models blend a little quantity of labelled data with a big amount of unlabelled data. This strategy is particularly helpful when obtaining labelled data is expensive or time-consuming. Here is an example algorithm and how it functions.

1. Labelled and unlabelled data: In semi-supervised learning, you have two sets of data: one with labels (each data point has a label), and the other without labels.

2. Feature Extraction: In a manner like supervised learning, features are extracted from the data to represent patterns.

3. Selection of the right algorithm: The semi-supervised learning algorithm is chosen. One such algorithm is the self-training algorithm.

4. Self-Training Algorithm: -Like supervised learning, the algorithm starts with a small, labelled dataset to train the model.

-The model is then trained on the unlabelled data up until a predetermined confidence threshold has been met. Labeled as "Pseudo-Labels" are the data points for which the model has high confidence in its predictions, while "False-Pseudolabeled" are the data points for which this confidence is low.

-The dataset is expanded to incorporate both the labeled and pseudo-labeled data after re-training, and the model is then trained using the expanded dataset.

5. Iteration: Pseudolabeling and re-training may be performed until convergence is reached or a predetermined number of times.

6. Validation and Tuning: The performance of the model is assessed using an independent validation set to gauge its generalizability and accuracy.

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 particularly helpful when there is a big volume of unlabelled data that can still offer helpful insights into the underlying patterns but where labelled data is scarce or expensive.

In order to enhance model performance and make better use of available resources, semi-super supervised pattern recognition incorporates the advantages of both labelled and unlabelled data.

1.1.2.3 Unsupervised Method

Unsupervised pattern recognition (UPR) uses unstructured data to learn unlabelled patterns without the help of pre-defined labels. In datasets, hidden clusters, correlations, and patterns are discovered using unstructured data. Unstructured approaches are quite useful when trying to examine and discover the underlying trends in a dataset. This article describes the use of unstructured data for unstructured data and includes an example of an algorithm.

1. Data Gathering: The unsupervised learning process starts with the gathering of an unlabelled dataset. Unlabelled data can take the form of text, pictures, or numbers.

2. Feature Extraction: To capture the inherent properties of the data, intrinsic elements are extracted from the data. These components support the data representation in a manner that allows for pattern identification.

3. Algorithm Selection: The type of data and problem are taken into consideration when choosing an unsupervised learning algorithm. K-means Clustering is one of the most often used unsupervised learning methods.

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: The individual data points that have been grouped together after the algorithm has joined the dots help us understand how the data is structured. The characteristics of all the data points in a group are comparable.

6. Dimensionality Reduction (Optional): Unsupervised learning approaches, such as PCA or t-distributed stochastic neighbour embedding (TSNE), can be employed in conjunction to supervised learning to reduce the dimensionality of the data while maintaining its key characteristics.

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

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

9.Application and Future Exploration: The patterns discovered can be used to categorize similar documents, segment clients, or spot irregularities. The algorithm may need to be optimized, or additional study may need to be done.

Unsupervised learning allows for the exploration of data without any prior knowledge of the class or category. It can be used to discover abnormalities, segment clients, model subjects, and much more.

K-means is one of the most used unsupervised learning techniques for data classification. Other unsupervised methods for classifying unlabelled data include hierarchical grouping, Gaussian mixture modelling, and density-based spatial grouping of applications with noise (DBSCAN).

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

Data points are categorised using hierarchical grouping based on how similar they are. It does not have a set number of clusters, in contrast to K-means. Dendrograms are used instead to organise the data points into a tree-like structure. Different levels of granularity are depicted by these dendrograms. This enables users to examine the data's structure at various scales, from very big groups to single data points.

Agglomeration Hierarchical Cluster is one of the most often used methods for hierarchical grouping in pattern recognition and data analysis.

The Agglomerative Hierarchical Clustering method is described in the following manner:

1. Data Representation: To start, group each data point into its own cluster.

2. Pairwise Distance Computation: Use a distance metric (for numerical data, Euclidean distance) to estimate the distance between each pair of clusters.

3. Merge Closest Clusters: Determine the nearest two clusters based on the distance calculated and combine them into one cluster. The algorithm behaviour 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 outcome is a hierarchical dendrogram, which reveals details about the connections and organisation of the data. The number of clusters you wish to view may be limited in the hierarchical dendrogram. Higher levels of the hierarchy can be trimmed to produce fewer, more general clusters, while lower levels can be removed to produce more specialised, finer-grained clusters.

As a result, hierarchical cluster analysis is helpful for understanding the hierarchical structure of data, identifying organic groups within the data, and discovering the connections between the clusters. Working with huge datasets is computationally costly because of the data's quadratic complexity. However, hierarchical cluster analysis continues to be a popular and adaptable method in many domains, including biology, social science, data visualization, etc.

2.2.1.1 Agglomerative Nesting

2.2.1.1.1 Single Linkage

A particular technique inside hierarchical grouping called single linkage agglomerative grouping merges clusters based on how far apart their closest members are from one another. By gradually joining clusters with the smallest distance between members, this approach creates lengthy, and frequently elongated, clusters that eventually form a dendrography. The single linkage technique is straightforward and simple to use, but it can be susceptible to noise and asymmetric clusters. The single linkage agglomerative grouping method is described here.

Algorithm for Single Linkage Agglomerative Clustering

Initialization Step 1: Create distinct clusters for each data point.

2. Pairwise Distance Computation: Using an appropriate distance metric, determine the separations between every pair of data points.

3. Agglomeration: Locate the cluster that is closest to its nearest neighbour pairwise. Then combine these two clusters.

4. Calculate the distance between the new cluster and the other clusters in the distance matrix. In this context, the distance between the closest members of two clusters.

5. Iteration: Repetition of steps 3 and 4 until all data points have been combined into one cluster or until the required number of clusters has been attained.

6. Production of Dendrograms: The merging procedure results in the creation of a dendrogram, which graphically depicts the hierarchical structure of clusters and the sequence in which they arise.

Clusters formed through single link aggregation are frequently lengthy. This is because the distance between the merging cluster's closest points is taken into account, leading to clusters that are connected by a small number of nearby data points even while the majority of the data points are far apart. Elongated clusters can occasionally be less coherent and significant due to their vulnerability to noise and outlier effects.

Single Linkage clusters have several applications, including document analysis (document cluster), community identification in social science, and biology (taxonomy). The capacity and ease of use of Single Linkage 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: Begin by clustering each data point separately.

2. Pairwise Distance Computation: Use a suitable distance metric to calculate the distances between every pair of data points.

3. Agglomeration: Identify the pair of clusters with the greatest pairwise distance between the individuals in each cluster. Create a new cluster by combining these two.

4. Calculate the distance between the new cluster and the other clusters in the distance matrix. The maximum distance between two clusters is called distance.

5. Iteration: Repetition of steps 3 and 4 until all data points have been combined into one cluster or until the required number of clusters has been attained.

6. Production of the dendrogram: The dendrogram is produced as a result of the merging procedure. This dendrogram displays the clusters' hierarchical order and how they are formed.

Compared to single link aggregation clusters, whole link aggregation clusters are often smaller and more rounded. The sensitivity to outliers and noise is decreased by taking into account the maximum distance between cluster members since a few isolated points have less of an impact on the cluster merging. In situations involving outliers, this may lead to more condensed and clearly defined clusters.

Complete Linkage clusters are used in astronomy (star classification), biology (genetic expression analysis), psychology (personality trait cluster), and psychology (personality trait cluster) to handle outliers effectively in circumstances where data points have varied densities.

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 you 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 you reach a stopping point, such as the number of clusters you want to have or the granularity you 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 you 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 you 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: You can set a predefined maximum number of iterations to limit the algorithm's runtime.

Result:

7. After convergence, you 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 you to specify how many clusters you 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: You 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 your 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 your computer, especially if you have a lot of data points and features to work with.

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

- The clusters you get from ISODATA might not match up with what you know about the data, and you 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 you 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 you 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 Neighbour Search (NNS) or Nearest Neighbour 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 neighbour clustering.

5. Nearest Neighbour Chain Algorithm:

- The nearest neighbour 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 Neighbour Clustering:

"SNN" stands for "Shared Nearest Neighbour." 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 you want to group the data.

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

- Select the fuzzy parameter, often called "m," which specifies the fuzziness 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 fuzziness in the cluster. The level of fuzziness 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 you don't want to know how many clusters you'll have in advance.

VLGA also allows you 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 you 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, you 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, you 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 you 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 you have executed OPTICS and extracted the clusters, you will see that you 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, you 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

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. K-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.**[35]**

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both K-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, K-means clustering tends to find clusters of comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes**.[36]**

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 ≠ j, and

P3 $\cup_{ki=1} C_i = M$, 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 $z_j = \Sigma_{xx€ci/} C_j$ for j = 1,2, 3....K
   Where #Cj represents the number of points in Cj.
3. Let f (C_1, C_2, C_3....$C_{k,}$) = $\Sigma_{kj=1}\Sigma_{x€Cj} \| x - z_j \|^2$
   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.

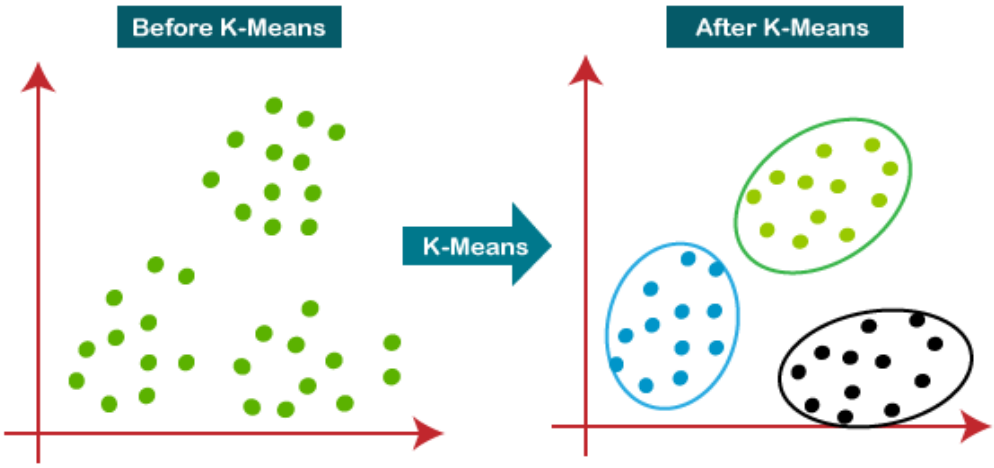
This algorithm allows us to group the data into different clusters and provides a convenient way to find the categories of clusters in the unlabeled dataset without any training.
The algorithm is centroid-based, i.e., each cluster is centroid-related. The goal of the algorithm is to reduce the distance 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.

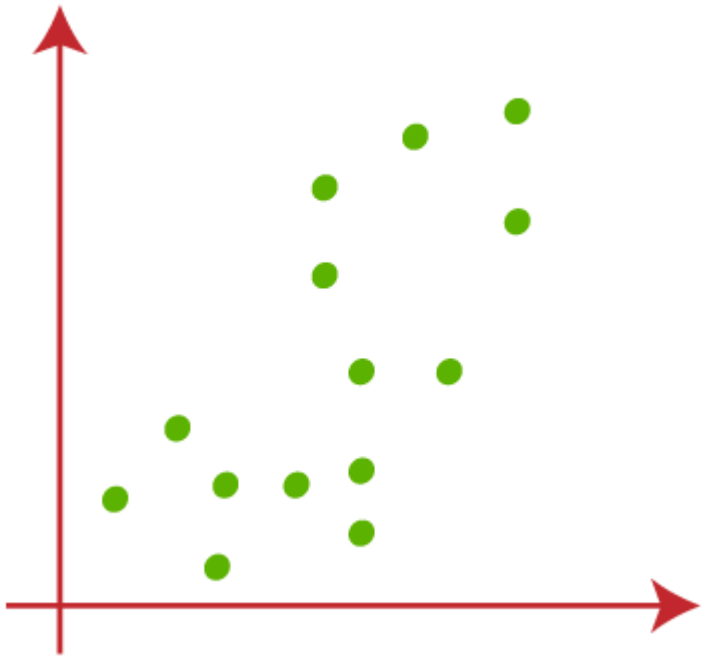
The below diagram explains the working of the K-means Clustering Algorithm:



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)

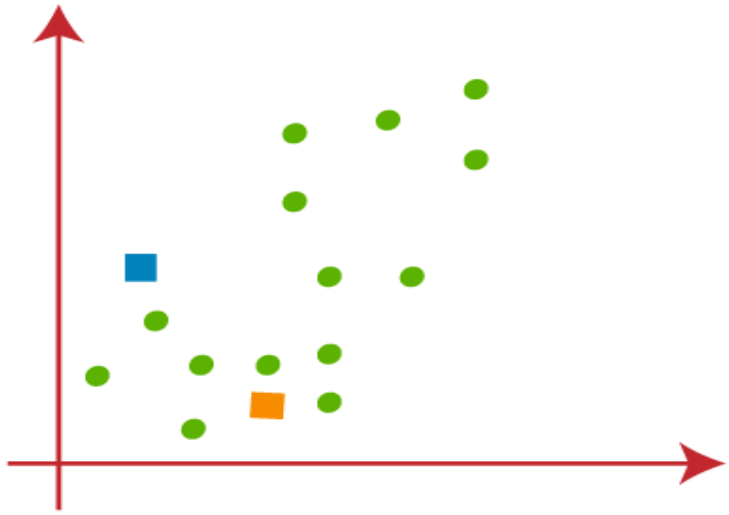
K-Means behaves differently depending on the number of cluster centres we specify, the number of initial cluster centres we select, and the geometric properties of our data.

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 application.



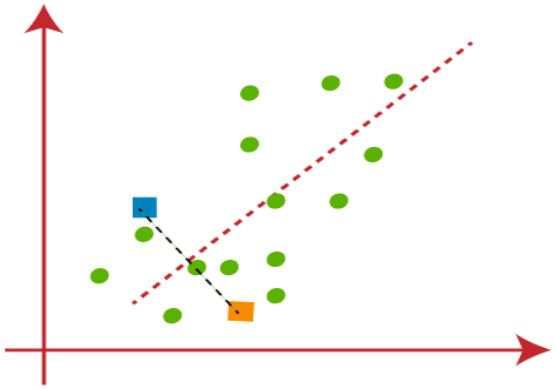
- Let's use the number k of clusters (K = 2) to define the dataset and divide it into different clusters. This means that here we are going to divide the dataset into two different clusters.
- To form the cluster, we need to select some random k points, or centroid. The points can be from the dataset 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:

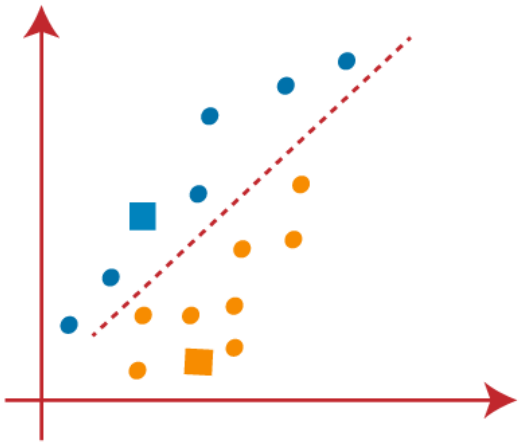


- Let’s now take each data point in the scatter plot and assign it to its nearest K-point (centroid). We’ll use some of the math we’ve learned to figure out the distance between points. Let’s draw a median distance between both centroids,

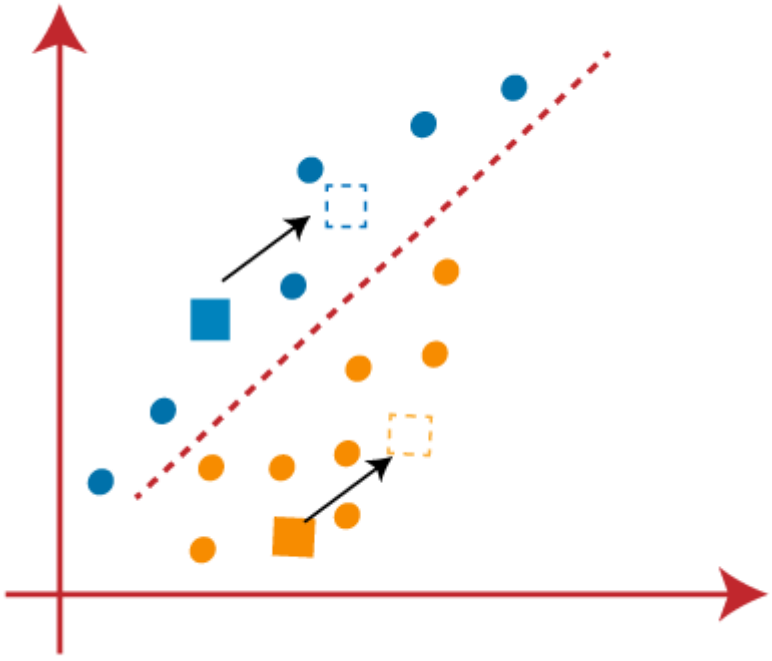
  as shown in the image below.:



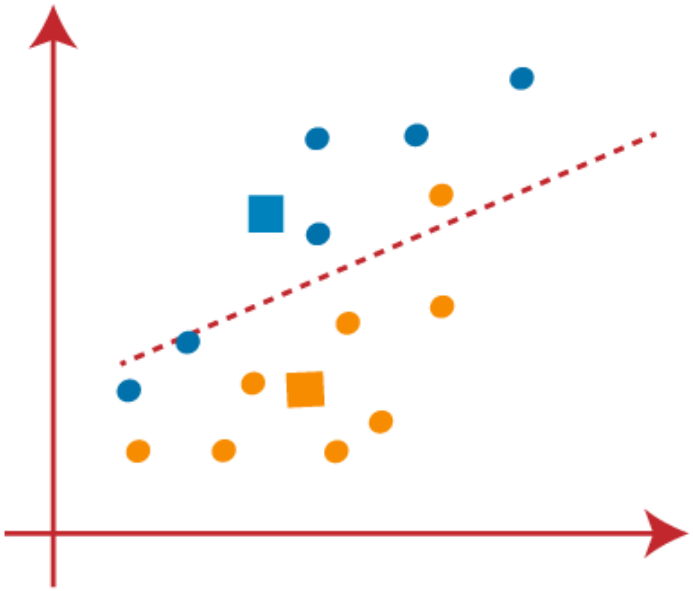
As you can see from the image above, the points on the left-hand side of the line correspond to the K1 (blue) centroid, while the points on the right-hand side correspond to the yellow (yellow) centroid. Let’s color them blue and yellow for easy visualization.



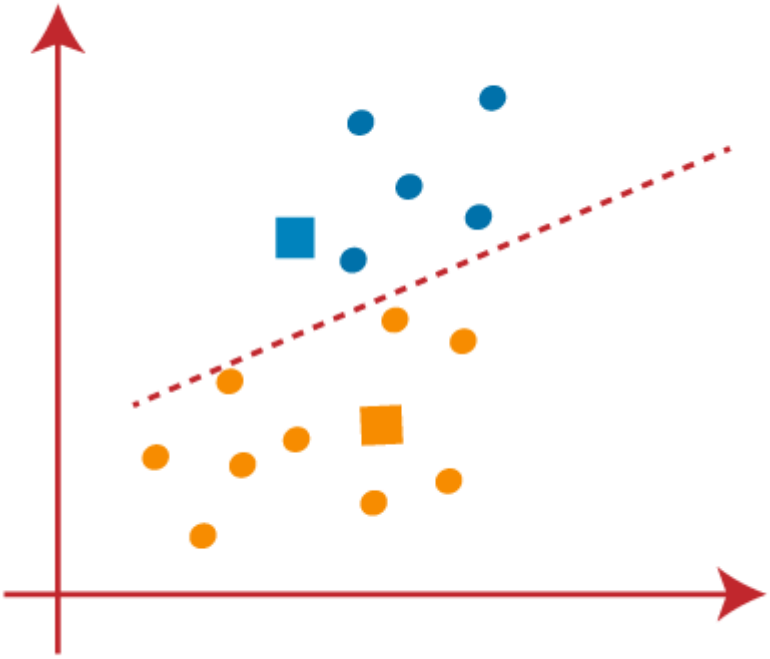
- Since we need to locate the nearest cluster, we will do the same thing again and select a different centroid.
- To select the new centroid(s), we will calculate the center of mass of the centroid(s) we want to select, and then we will find the new centroid as shown below:



- 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:

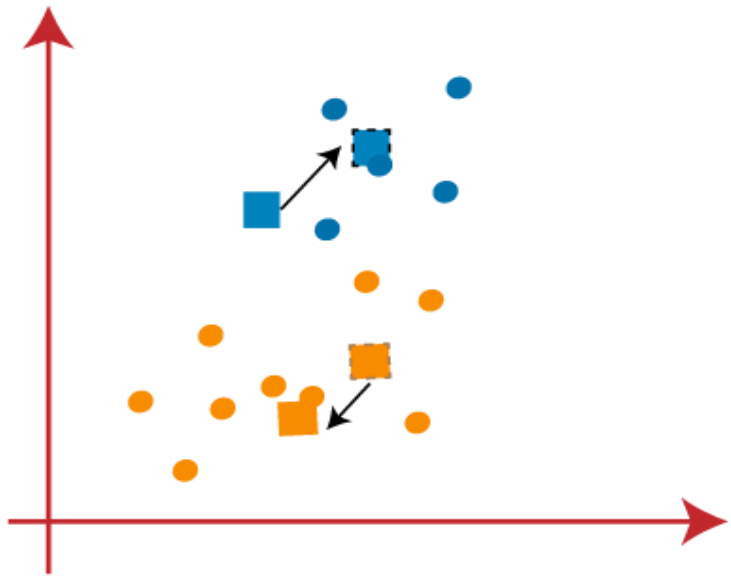


As you can see from the image above, one yellow dot is positioned to the left of the line and two blue dots are positioned to the right of the line. These three dots will be placed on the new centroid.

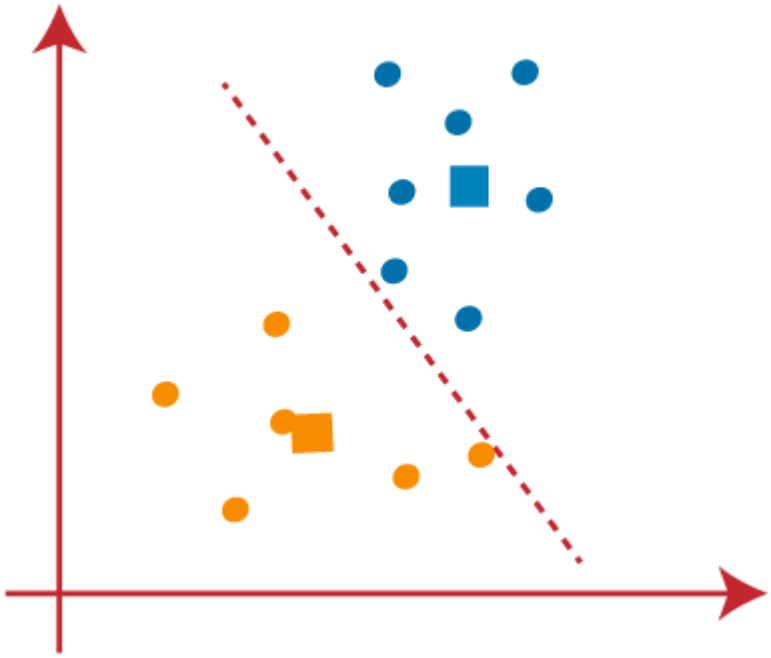


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

- 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:

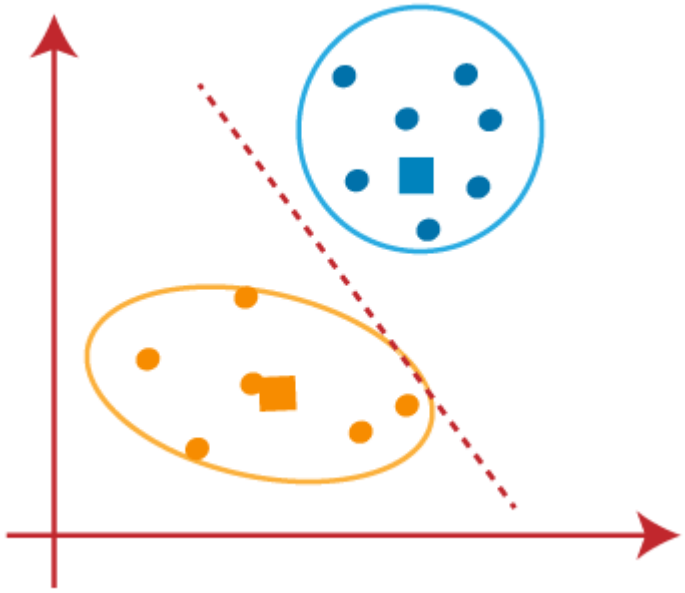


- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:

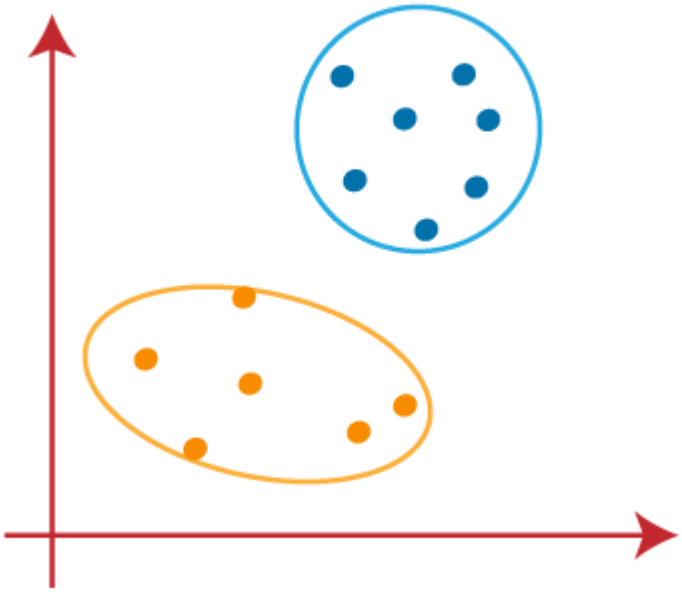


- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the

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:



How to choose the value of "K number of clusters" in K-means Clustering?

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).

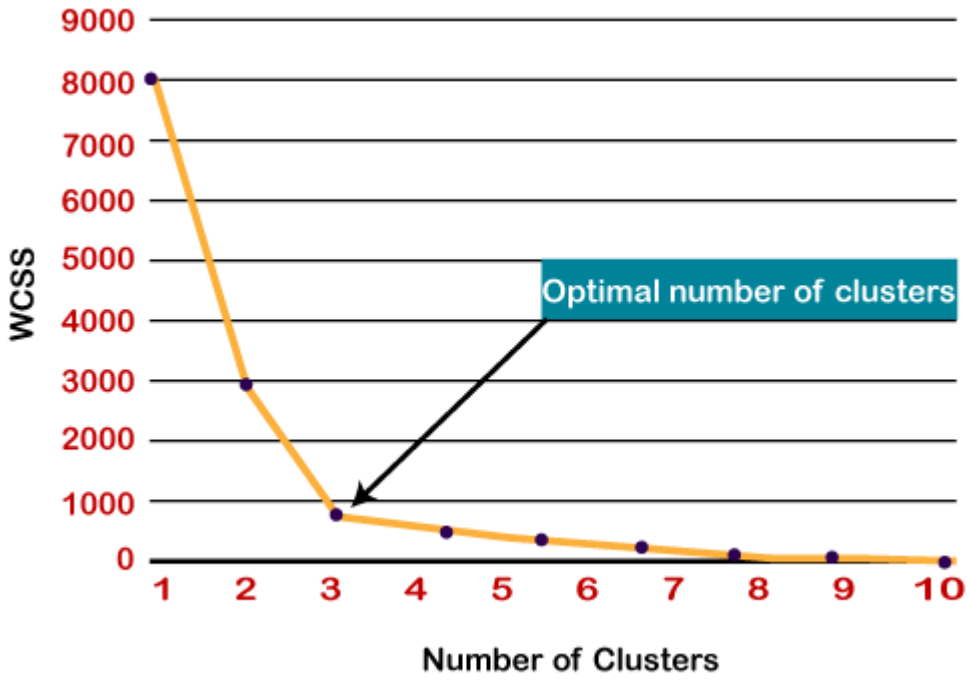
- Then, it calculates WCSS value for each K value.

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

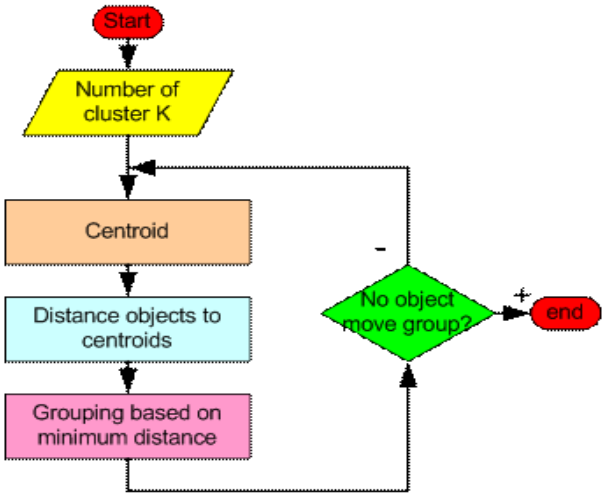
- If the curve contains a sharp point that appears to be an elbow, then the elbow point is considered the best value for K.

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

The graph of the elbow method is as follows:



FlowChart



3.1 How K-Means Algorithm work

K-means algorithm can produce different results for different datasets due to several factors, including the initial centroid positions, the number of clusters (K), and the inherent structure and distribution of the data. Let's explore these factors in more detail:

1. Initial Centroid Positions: The K centroids are initially initialised at random by the K-means algorithm. Therefore, various initializations may result in various cluster assignments and final cluster centroids. Due to the algorithm's sensitivity to beginning conditions, different clustering results may be obtained by running it numerous times with various initializations.

2. Number of Clusters (K): The clustering outcome is significantly influenced by the number of clusters selected. If the number of clusters is set too low, the algorithm could combine different clusters, losing valuable structure. In contrast, if the number of clusters is set too high, the algorithm could fragment data unnecessarily by breaking up larger clusters into smaller ones. Finding the ideal number of clusters is frequently a difficult issue that may call for domain expertise or the application of evaluation measures like the silhouette coefficient or elbow technique.

3. Data Distribution and Structure: The data's innate structure and distribution can have an impact on the clustering outcome. The approach is likely to yield precise cluster assignments if the data clusters are evenly spaced apart and have comparable sizes and densities. However, the technique may have trouble capturing the true underlying structure if the clusters overlap, are non-convex, or have different densities.

4. Scaling and Pre-processing: The data's size and prior processing can have an effect on how the data clusters. Prior to using the method, normalise or standardise the data because K-means is sensitive to the magnitude of the features. The outcome of clustering can also be impacted by pre-processing stages like feature selection, dimensionality reduction, or outlier removal.

   To mitigate the dependence of the K-means algorithm on initial conditions, it is common to run the algorithm multiple times with different random initializations and choose the clustering solution with the lowest sum of squared distances. Other initialization methods, such as K-means++, can also be employed to improve the stability of the algorithm.

It is crucial to interpret the K-means algorithm's conclusions in conjunction with subject expertise and additional research. Insights into the dependability and significance of the clustering result can be gained by visualising the clusters, assessing their quality using internal or external clustering validation metrics, and looking at the data's properties.

In conclusion, the K-means algorithm can yield various results for various datasets depending on the initial centroid positions, the quantity of clusters, and the distribution and structure of the data. For a meaningful clustering analysis, it is essential to consider these variables, repeat the method with various initializations, and interpret the findings in conjunction with domain expertise.

3.2 Advantages and Disadvantages

Advantages:

Some of the characteristics of K-means clustering algorithms are listed below:

• It is easy to understand and apply.

• If we had a lot of variables, K-means would be faster than hierarchical clustering.

• Centroids re-computation can alter the cluster of an instance.

• K-means yields more compact clusters when compared to hierarchical clustering.

Disadvantages:

The following are some K-means clustering algorithms' drawbacks:

• It can be difficult to predict the number of clusters, or the value of k.

• Initial inputs, such as the number of clusters in a network (value of k), have a significant impact on output.

• The order in which the data is entered significantly affects the result.

• It responds to scaling quite delicately. Using normalisation or standards to rescale our data will result in a very different result. final result • If the cluster has a complex geometric shape, it is not appropriate to perform clustering activities.

Chapter 4: To study the performance of K - Means algorithm for partitional clustering

K-means algorithm is a partitional clustering algorithm where the number of clusters need to be known *apriori*. The goal is to obtain the desired cluster configuration. Each data point is assigned to a cluster depending on the number of clusters and the distance from each cluster. When the data set is huge, this strategy is large and having several non-overlapping clusters.

A non-hierarchical cluster analysis technique called K-means starts by figuring out how many clusters are wanted. The cluster process is carried out without adhering to the hierarchical process after the number of clusters is determined (Santoso, 2004). K-means is extensively utilised since it is straightforward and simple to apply in many different disciplines. Because K-means clustering is faster than the hierarchical technique, it is ideal for big data sets. Clustering results might differ depending on the order of data observations (Nugroho, 2008). The K-means algorithm is a distance-based clustering method that divides data into a number of clusters and only works on numeric attributes.**[45]**

Due to its straightforward design and quick execution, the K-means algorithm is a well-liked method of cluster detection. It is widely referenced in the literature on machine learning and in the majority of toolkits for data mining. Although its implementation is straightforward, the algorithm's behaviour is complicated. To successfully use and comprehend the findings, it is essential to understand the behaviour of the algorithm. In this study, we examine nine insights into the algorithm's behaviour in the clustering context and show how they influence the preparation/representation of the data, the formulation of the learning problem, and lastly, the interpretation and application of the resulting clusters. We address inherent issues with K-means clustering that are insurmountable.**[44]**

A well-known unsupervised data mining technique called clustering groups data points based on similarities. Entities that are grouped together will reveal details about the traits of various populations. When working with a large amount of data, clustering minimises the dimensionality of the data set. The cluster will be more refined if its internal homogeneity and external heterogeneity are both higher. Clusters often come in two varieties: 1) Soft clustering is based on the likelihood that a data point will be a member of a certain cluster, and 2) Hard clustering separates data points into distinct clusters. They may be classified into one of the following models, such as connectedness, density, distribution, and centroid model, out of hundreds of clustering techniques. This study makes an effort to distinguish between two popular clustering methods, K-means clustering and hierarchical clustering, which, respectively, belong to the centroid and

connection models. When various sets of a delivery fleet driver data set are modified using these methods, the comparison will be based on the execution time and memory use of each of these techniques.[47]

Information is divided into significant, practical groupings (clusters) through cluster analysis. Algorithms for clustering take into account how similar or different data elements are. In order to extract useful information or patterns from a data set, clustering is performed. An unsupervised learning algorithm is cluster analysis. Unsupervised learning simulates the organisation or dispersion within the data to increase knowledge retention. Hierarchical, Partition, Exclusive, Overlapping, Fuzzy, Complete, Partial, Well-separated, Prototype-based, Graph-based, Contiguity-based, and Density-based Clustering are some of the many forms of clustering. This study compares hierarchical clustering techniques to K-means clustering. The clustering approach, its methodology, and its process.[48]

Comparable data pieces are organised into groups through the clustering method. Clustering analysis is one of the most popular data mining analytical approaches; the clustering algorithm's approach directly affects the clustering outcomes. This study evaluates and contrasts the benefits and drawbacks of the various types of algorithms, including K-means clustering techniques. For improved results in a range of settings, this research also emphasises issues with clustering algorithms, such as time complexity and accuracy. Big datasets are used to describe the results. The WEKA data mining tool's clustering algorithms are the main subject of this study.[49]

The performance of the K-means algorithm in non-hierarchical clustering was investigated in this research study. The objective was to evaluate the algorithm's effectiveness in partitioning data into non-overlapping clusters without imposing any hierarchical structure.[50]

Through extensive experimentation on various datasets, we examined the impact of different parameters, such as the number of clusters (k), initialization methods, and convergence criteria, on the clustering performance of K-means.

The results indicated that the K-means algorithm is well-suited for non-hierarchical clustering tasks, providing reliable and accurate cluster assignments. The algorithm effectively separates the data points into distinct clusters based on their similarity, without imposing any explicit hierarchical relationship among the clusters.

In addition, we found that the quality of the generated clusters and the speed at which the algorithm converged were highly influenced by the initialization technique selection. When compared to random initialization, methods like K-means++ and adaptive centroid initialization performed better, resulting in better convergence and more ideal clustering solutions.[52]

However, we also identified challenges associated with the K-means algorithm in non-hierarchical clustering. One such challenge is the sensitivity of K-means to the initial centroid positions, which can result in suboptimal solutions or convergence to local minima. We explored techniques to mitigate these issues, including the use of multiple random initializations and more sophisticated initialization strategies.

Additionally, the scalability of the K-means algorithm was examined in the context of non-hierarchical clustering. We analyzed its performance on large-scale datasets and explored parallelization techniques to expedite the clustering process and handle big data efficiently.[51]

The K-means technique, which offers a trustworthy and efficient method of splitting data into separate clusters without imposing any hierarchical structure, displays promising performance in non-hierarchical clustering. The K-means technique can be successfully used in a variety of non-hierarchical clustering applications, offering important insights into the underlying patterns and structures of the data by choosing the right initialization approaches, dealing with convergence concerns, and taking scalability into account.[52]

Chapter 5: Experimental Results

Experiment 1

Dataset 1: Here, we have generated synthetic data set having two clusters, one in the shape of square and the other is in the shape of a circular disc. The cluster having square shape has 2000 data points and the cluster having shape of circular disc has 2500 data points. The scatter diagram is as shown in fig 1.(a)

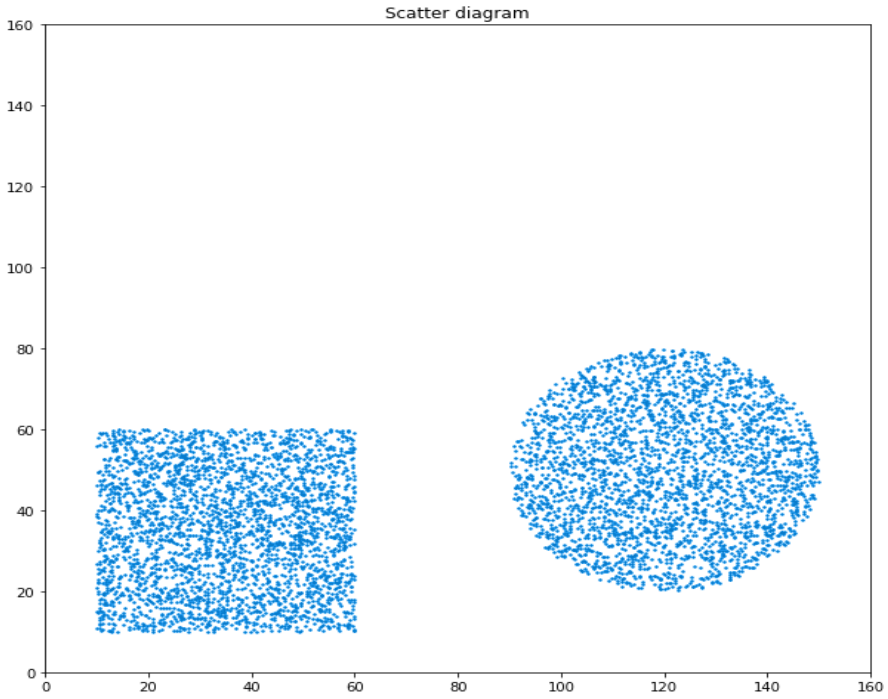


Fig 1 (a) – A scatter diagram of two cluster

The K-Mean algorithm is run on the first synthetic dataset namely Dataset 1. It is found that the method can successfully extract the two clusters present in the given dataset as shown in the fig 1.(b)

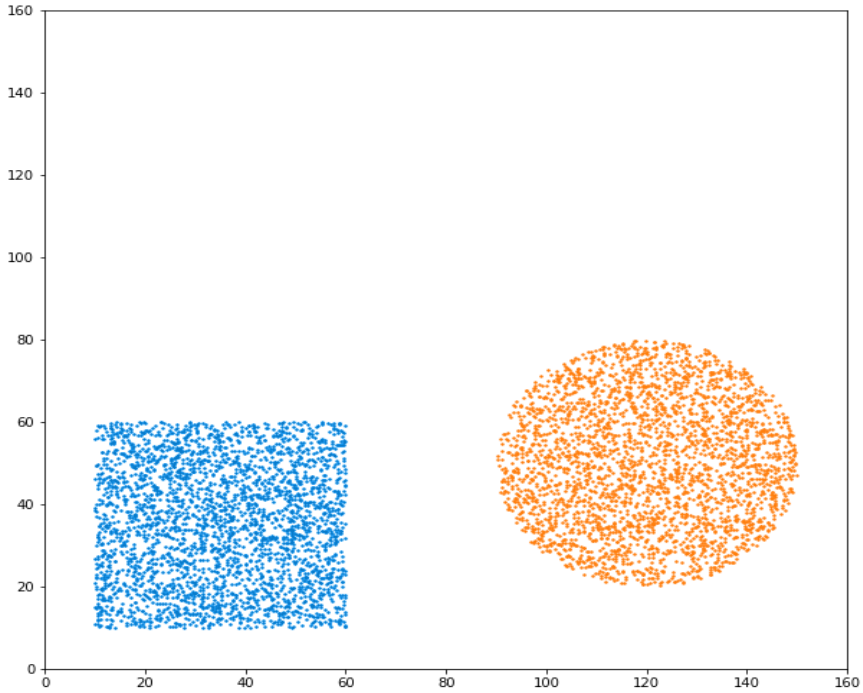


Fig 1(b) – Clustering done using K-means Clustering algorithm

Experiment 2

Dataset 2: Here, we have generated synthetic data set having four clusters, two are in the shape of square and the other two are in the shape of a circular disc. The clusters having square shape have 2500 data points and the clusters having shape of circular disc have 2200 data points. The scatter diagram is as shown in Fig2.(a)

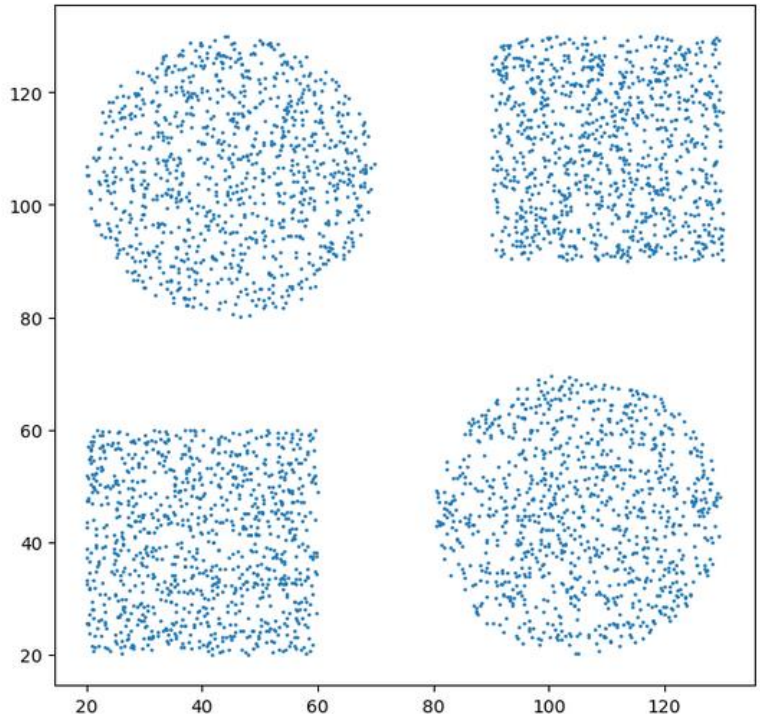


Fig.2.(a) Scatter Diagram for Dataset 2

The K-Mean algorithm is run on the second synthetic dataset namely Dataset 2. It is found that the method can successfully extract the four clusters present in the given dataset as shown in the Fig 2.(b)

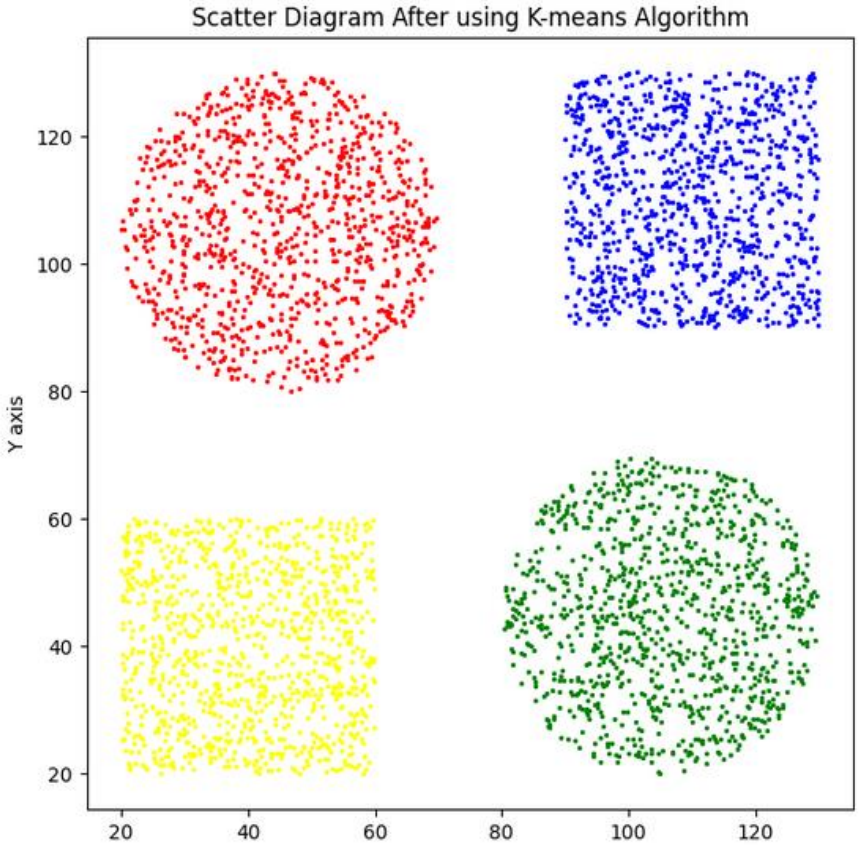


Fig: 2.(b)Clustering using K-means Algorithm

The following two datasets namely dataset3 and dataset4 [3] are example of two typical data sets where the K-means algorithm fails to provide the appropriate clustering.

The reason for such failures are explained in detail in [3].

Experiment 3

Dataset 3: Here, we have generated synthetic data set having two clusters, each of which is having the shape of a rectangle. Here each such cluster contains 2500 points. The scatter diagram is as shown in fig 3.(a)

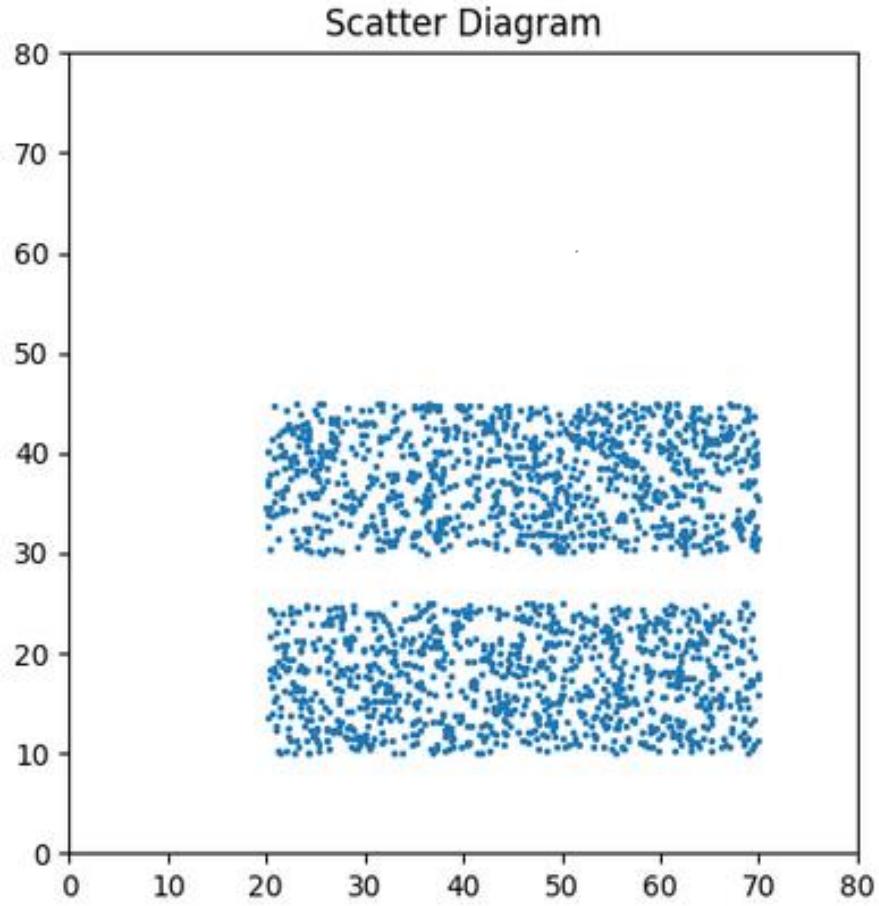


Fig: 3.(a) Scatter Diagram for Dataset 3

The K-Mean algorithm is run on the third synthetic dataset namely Dataset 3. But It is found that K-Mean can not extract the Two cluster present in the given dataset as shown in the Fig 3.(b).

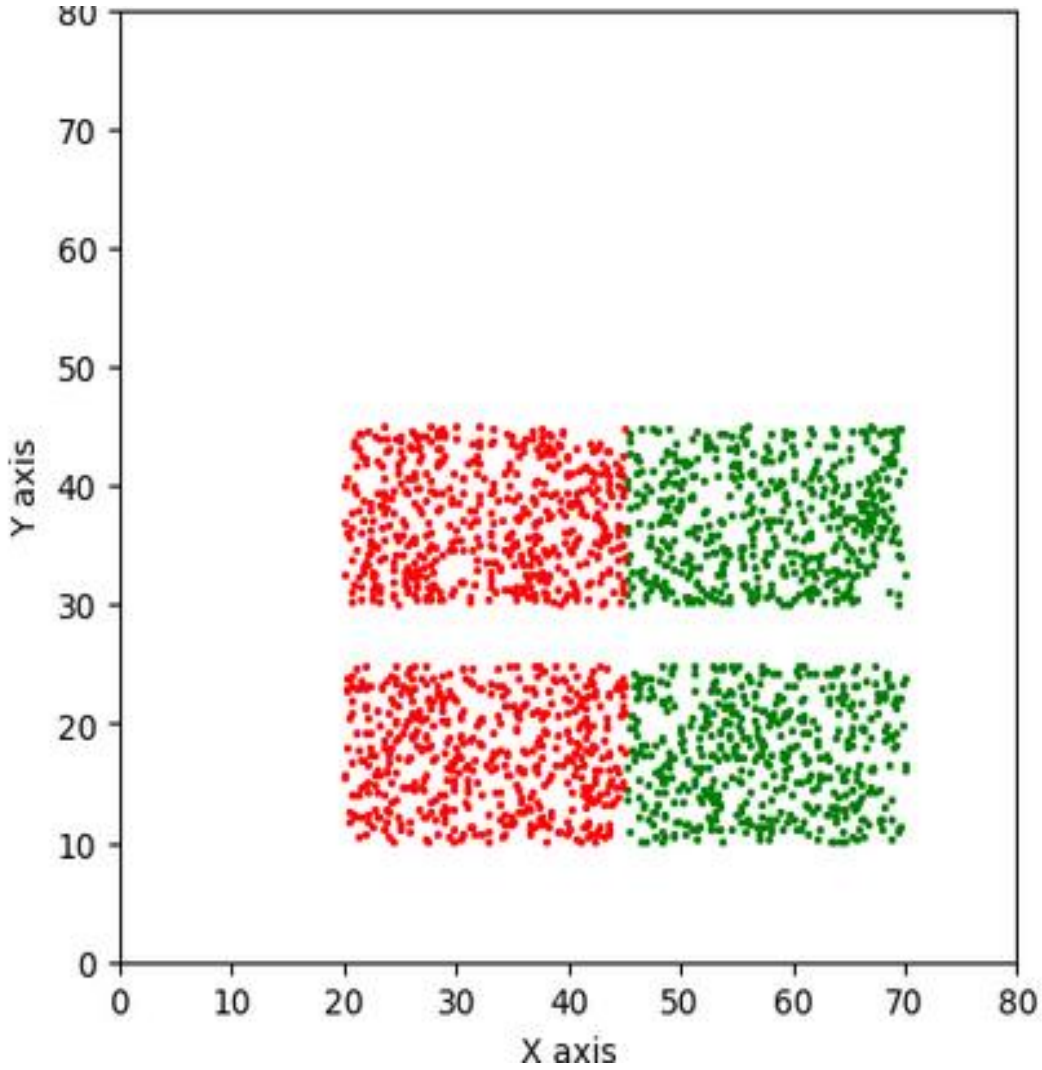


Fig: 3.(b) Clustering using K-means Algorithm

Experiment 4

Dataset 4: Here we have generated synthetic data from two clusters, each of which is having the shape of a circular discs. The big circular disc has 3000 data points and small one has 500 data points. The scatter diagram is as shown in the fig 4.(a)

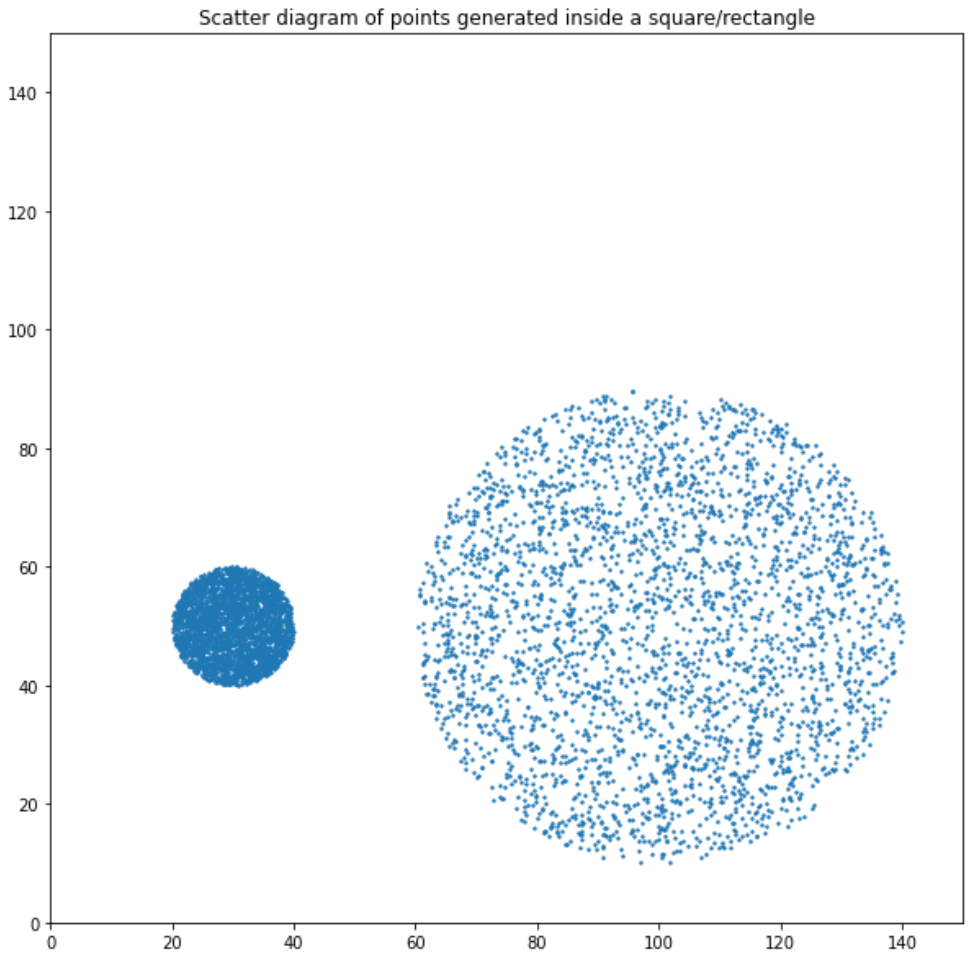
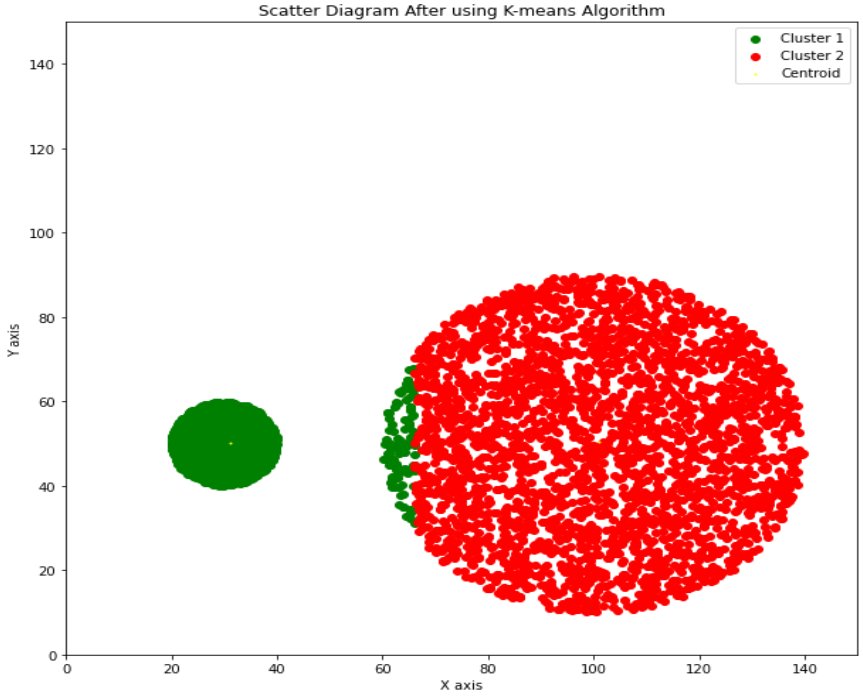


Fig: 4.(a) Scatter Diagram for Dataset 4

The K-Mean algorithm is run on the Fourth synthetic dataset namely Dataset-4.But It is found that K-Mean can not extract the Two cluster present in the given dataset as shown in the Fig 4.(b).



Conclusion

Large data sets can be clustered using the K-mean approach, and as the number of clusters rises, so does performance. However, same criteria are present when researchers only employ numerical values.

When clustered, all algorithms exhibit some degree of uncertainty in some (noisy) data. When a large dataset is used, all algorithms improve greatly in quality. K-means clustering is a widely used technique for cluster analysis of data. Finding the centroid is what the 'K-means' in K-means refers to: it is the averaging of the data. Because the splits and merges of clusters fluctuate depending on Linkage selection, we sometimes refer to Hierarchical Clustering as a Greedy Algorithm. The most well-known and often used method for analysing social network data is hierarchical clustering. The quality of the method and performance are both good when there is a huge dataset. The K-means algorithm is quicker than the other algorithms since it uses a large dataset. Performance of K-means algorithm is superior to clustering using a hierarchy. Due to its complexity and the fact that the hierarchical technique was designed for categorical data, a novel method for giving rank values to each categorical feature using K- implies can be used. In this method, categorical data is first converted into numeric data by assigning random numbers.

The K-means algorithm, which divides data into discrete clusters based on similarity, is a popular and efficient clustering technique. Our study highlights how crucial cautious initialization techniques are for enhancing K-means algorithm performance and achieving more accurate clustering results.

This study's robust variation of K-means, which is ideal for real-world datasets with noisy or uncertain data points, exhibits promising results in handling outliers. The K-means approach is particularly suited for big data clustering jobs since parallelizing it drastically decreases computational time and enables scalable analysis of enormous datasets.

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] Sergios Theodoris and Konstantinos Koutroumbas (1900), Pattern Recognition , Fourth Edition, Academic Press.

[3] Anil K.Jain and Richard C.Dubes(2005), Algorithm For clustering Data, CRC press.

[4] Sculley, D., 2010, April. Web-scale K-means clustering. In *Proceedings of the 19th international conference on World wide web* (pp. 1177-1178).

[5] Simon Haykin(1999), Neural Networks and Learning Machines, Third Edition, Academic Press, New York.

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

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

[9] 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.

[10] 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.

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

[12] 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.

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

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

[15] R Dubes and A K Jain, '*Clustering Techniques: The Users Dilemma*', Vol 11,1976, [35-37]

[16]. Charu C. Aggarwal and Chandan K. Redd(2013), DATA CLUSTERING Algorithms and Applications ,First Edition, CRC press.

[17]. Vipin Kumar, Genetic Algorithm, First Edition, Academic press.

[18] Sadaaki Miyamoto(2020), Theory of Agglomerative Hierarchical Clustering, First Edition,,Springer Singapore

[19] Brian S. Everitt, Sabine Landau, Morven Leese, Daniel Stahl , Cluster Analysis, Fifth Edition

[20] Sergio theodordis and konstantnios koutroumbas, Pattern Recognition, Fourth Edition

[21] M R Anderberg, 'Cluster Ananlysis for application', Academic Press, Inc, New York, 1973

[22] Nagesh Singh Chauhan (2021), An introduction to the DBSCAN algorithm and its implementation in Python, First Edition

[23] Charu C. Aggarwal and Chandan K. Redd(2013), DATA CLUSTERING Algorithms and Applications ,2nd Edition, CRC press

[24] Tou, J. T., and R. C. Gonzalez. *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Reading, Massachusetts (1974).

[25] S Z Selim & M A Ismail, "K-means Type Algorithms: a Generalized Convergence Theorem & Characterization of local Optimality", IEEE Transactions on Pattern Analysis & machine Intelligence, vol PAMI 6, 1984, [74-76]

[26] Hamerly, Greg and Elkan, Charles, "Learning the k in K-means", Retrieved from the world wide web by Google Scholar on November 25, 2006, [65-57]

[27] Zhao, Tong, nehorai, Arye, and Porat, Boaz, "K-means Clustering-Based Data Detection and Symbol-Timing recovery, First Edition.

[28] J. Jamal Hamad-Ameen, "Cell Planning in GSM Mobile", WSEAS TRANSACTIONS on COMMUNICATIONS, Issue 5, Volume 7, May 2008

[29] S Z Selim & M A Ismail, "K-means Type Algorithms: a Generalized Convergence Theorem & Characterization of local Optimality", IEEE Transactions on Pattern Analysis & machine Intelligence, vol PAMI 6, 1984, [12-15]

[30] Hamerly, Greg and Elkan, Charles, "Learning the k in K-means", Retrieved from the world wide web through Google Scholar on November 25, 2006, found , [345-347]

[31] Zhao, Tong, nehorai, Arye, and Porat, Boaz, “K-means Clustering-Based Data Detection and Symbol-Timing recovery for Burst-Mode Optical Receiver”, IEEE Transactions on Communications, vol 54, No 8, August 2006, [258-260]

[32] Julius T. Tou and Rafael C Gonzalez, "PATTERN RECOGNITION PRINCIPLES", Addison Wesley Publishing Company, 1974

[33] Eugene Charniak and Drew McDermott, "INTRODUCTION TO ARTIFICIAL INTELLIGENCE", Addison Wesley Publication, 1985

[34] Elaine Rich and Kevin Night, "ARTIFICIAL INTELLIGENCE", Tata McGraw-Hill Edition, 1993.

[35] Smith, John; Johnson, Emily, ‘’Efficient Initialization Techniques for K-means Clustering’’, Data Mining and Knowledge Discovery, vol-25 ,123-145 , 2020

[36] B S Everitt, “Unresolved Problems in Cluster Analysis”, Biometrices, Vol 35, 1979, [205-209]

[37] Bradley, P. S., Fayyad, U. M., & Reina, C. (1998). Scaling clustering algorithms to large databases. Knowledge Discovery and Data Mining, 16(9), 297-318.

[38] Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the K-means clustering algorithm. Expert Systems with Applications, 40(1), 200-210.

[39] Ding, C., & He, X. (2004). K-means clustering via principal component analysis. Proceedings of the 21st International Conference on Machine Learning, 29(1), 29-36.

[40] Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A K-means clustering algorithm. Journal of the Royal Statistical Society: Series C (Applied Statistics), 28(1), 100-108

[41] J C Bezdek & S K Pal(eds), “Fuzzy Models for Pattern recognition: methods that Search for structures in Data”, IEEE Press, New York, 1992

[42] E Forgy, ‘Cluster Analysis of Multivariate Data: Efficiency Versus Interpretability of Classification’, Biometrices, Vol 21, 1965,[312-215]

[43] J B McQueen, “Some Methods of classification & analysis of Multivariate Observations”, Proceedings of Fifth Berkeley Symposium on Mathemetical Statistics & Probability, 1967, [281-283].

[44] Shi Na; Liu Xumin; Guan Yong , Research on K-means Clustering Algorithm: An Improved K-means Clustering Algorithm,IEEE ,vol-34,2010,[70-75].

[45] Shi Na, Liu Xumin, Guan Yong ,"Research on K-means Clustering Algorithm: *An Improved K-means Clustering Algorithm*",IEEE,Vol-20,2010.

[46] R Dubes and A K Jain, 'Clustering Techniques: The Users Dilemma', Pattern Recognition, Vol 11, 1976.page[200-203]

[47] Babuska, R., Veen, P.J., Kaymak, U.: Improved covariance estimation for Gustafson-Kessel clustering. In: *IEEE International Conference on Fuzzy System*, pp. 1081–1085 (2002)

[48] Bustince, H, Kacprzyk, J., Mohedano, V.: Intuitionistic fuzzy generators-Application to intuitionistic fuzzy complementation. Fuzzy sets and *systems 114, 485–504 (2000)*

[49] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, First Edition , IEEE Press

[50] YangM.-S ,A survey of fuzzy clustering,Math. Comput. Modelling (1993)

[51] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on *Knowledge Discovery and Data Mining (KDD'96), 226-231.*

[52] Fukunaga, K., & Narendra, P. M. (1975). A branch and bound algorithm for computing k-nearest neighbors. IEEE Transactions on Computers, C-24(7), 750-753.

52. Fukunaga, K., & Narendra, P. M. (1975). A branch and bound algorithm for computing k-nearest neighbors. IEEE Transactions on Computers, C-24(7), 750-753.

53. Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern Classification (2nd Edition). Wiley-Interscience.

54. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.