UNIVERSITY OF GHANA, LEGON
DEPARTMENT OF MATHEMATICS

SPATIAL CLUSTERING

A STUDY OF KMEANS AND DBSCAN AS CLUSTERING METHODS

BY

EMMANUEL KWAME AYANFUL
10658737

December, 2021

A dissertation submitted to the Department of Mathematics, University of Ghana, in partial fulfillment of the requirements for Bachelor of Arts degree.

Declaration

I declare that this dissertation is my original work. Except where due acknowledgement has been provided, this dissertation contains no material previously published by any other individual. To the best of my knowledge, there is no content in this dissertation that has been accepted as part of any other academic degree requirements at the University of Ghana or any other university.

This dissertation was carried out in the department of Mathematics, University of Ghana in partial fulfilment of Bachelor Arts Degree under the supervision of Dr. Eyram Kojo Amenuveve Schwinger.

Student's Name: Mr. Emmanuel Kwame Ayanful.

Signature: _____

Date: _____

Supervisor's Name: Dr. Eyram Kojo Amenuveve Schwinger.

Signature: _____

Date: _____

Abstract

Clustering algorithms are of significant importance when it comes to working with spatial databases. In the world of machine learning and data science, clustering is the most used technique as most data around the world are large, unlabled and without meaning. This dissertation outlines the various methods used in spatial clustering, taking a critical look into Kmeans and Density Based Spatial Clustering of Application and Noise (DBSCAN).

This dissertation explores the mathematics behind these methods, toppled with a python program to implement them.

Acknowledgements

First and foremost, I am extremely grateful to the almighty God for the breath of life and the strength to carry out this dissertation. I also want to express my gratitude to my supervisor, Dr. Eyram Kojo Amenuveve Schwinger, for his tremendous guidance, unwavering support, and patience during this dissertation. His expertise and breadth of experience kept me going throughout this project, and his insightful comments pushed me to bring the best out of this project. I would also like to thank my family, especially my mom for their support both emotional and financial throughout my 4-year studies at the University of Ghana.

I say thank you to everybody who contributed, one way or the other to this dissertation.

Contents

Chapter 1

Introduction

Machine learning is a core branch of Artificial Intelligence that gives computers the ability to learn without being explicitly programmed[Samuel, 1959] with many subfields and applications including neural networks, statistical learning methods, image recognition, genetic algorithms, instance-based learning, reinforcement learning, natural language processing, inductive logic programming and computational learning theory.

Machine learning is a subfield of computer science that differs from standard computational approaches in many ways. In traditional computing, an algorithm is a sequence of instructions that have been carefully designed to achieve a certain result. However, machine learning approaches allow computers to learn by developing outcomes from data inputs that are included within a certain range using statistical analysis[Lisa, 2017].

When exposed to data, a typical machine learning model first learns the information from the data and then utilizes this knowledge to make predictions on future data. The capacity to explore and discover patterns in vast amounts of data without a priori knowledge is a specific advantage of machine learning techniques. As an example, machine learning software will be used to find patterns in huge volumes of electronic health record information by detecting anomalous subsets of data records.

Generally speaking, machine-learning activities may be grouped together. Each of these categories is reliant on the mode of receiving learning or how feedback on learning is provided to the technique designer. Machine learning approaches that are commonly used include supervised and unsupervised learning.

1.1 Supervised Learning

As part of supervised learning, inputs that have been tagged with their expected outcomes is provided to the computer. With this technique, the algorithm learns by comparing real outputs with learned outputs in order to detect mistakes and adjust the model as necessary. It utilizes trends in data to predict label values for a new data without labels.

A typical use of supervised learning is to statistically predict probable future events using past data. Using past stock market data, Supervised learning can predict forthcoming fluctuations, and it can be also used to filter out spam email messages. It is possible to categorize untagged images of dogs by using tagged images of dogs as input data.

As an example, an algorithm is fed with data consisting of photos of lizards identified as reptiles, and photos of dogs tagged as mammals, using supervised learning. The supervised learning system is expected to recognize unlabeled dog photos as mammals and unlabeled lizard images as reptiles after being trained on this data[Lisa, 2017].

1.1.1 Regression

Regression analysis consist of a set of machine learning models that enables us to predict an outcome variable (y) based on the value of one or more predictor variables (x). Models under regression aim to provide a mathematical equation that relates the variables x to the outcome, or y. The result variable (y) is then predicted using this mathematical equation and updated values for the predictor variables (x). A such use case is estimating a home's price based on factors like square footage.

There are four(4) methods used in regression analysis namely:

- Simple Linear Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression

1.1.2 Classification

Classification focuses on predicting qualitative responses by analysing data and recognizing patterns. It explicitly states or specify the class to which a data point belongs and is best used when the outcome has finite discrete values. Some use cases will be classifying or predicting if a credit card transaction is fraudulent or not based on past transactions. Another use case will be identifying whether an email received is a spam or ham. Some models used in classification techniques are:

- Logistic Regression, which uses probabilistic approaches to determine if an event exist or not. It calculates the probability of a given outcome in a binary 0, 1 model, for instance the probability of being classified as rich, passing an exam, being sick and many more.
- Support Vector Machines (SVMs), which are used to classify data by recognizing hyperplanes.

- The K-Nearest Neighbor algorithm, which assigns a data point to a class based on similarity, evaluates the labels of a selected number (k) of data points around a particular data point in order to create a prediction about the class to which the data point belongs.

1.2 Unsupervised Learning

Unsupervised learning refers to a set of challenges in which a model is built to characterize or extract correlations or patterns from data. Unsupervised learning, as contrast to supervised learning, focuses entirely on the input data, with no output or target variables[Jason, 2019]. Customer segmentation is one example of a use case, which entails dividing customers into smaller groups based on similar characteristics, and is commonly used in marketing and other business related activities. **Clustering** is the model used in unsupervised learning.

1.2.1 Clustering

Exploratory data analysis techniques, such as clustering, are often employed to gain an intuitive understanding of the data structure. The task of finding subgroups of data in such a way that data points in the same subgroup (cluster) are extremely similar while data points in other clusters are substantially different is known as data clustering. As a result, we want to use a similarity measure like Euclidean distance to locate homogenous groupings of data points that are as similar as possible. The similarity metric to use is determined by the algorithm's application.[Imad, 2018].

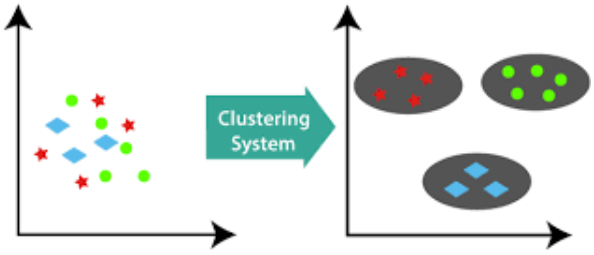


Figure 1.1: A figure showing how clustering works.

There are five (5) methods used in clustering namely:

- Partitional clustering which involves putting observations within a dataset into multiple groups based on their similarities. The number of groups used in this method is pre-specified by the analyst. Under partitional clustering we use the following techniques:
  - K-means Clustering which puts a given dataset into k predefined non-overlapping sub-groups based on similarity of points in dataset.

  - K-medoids Clustering is more robust to noise as compared to K-means since k-medoids minimizes a sum of pairwise dissimilarity instead of the sum of squared euclidean distances.

- Hierarchical Clustering is a method for finding groups in a dataset that is different from partitional clustering. Unlike partitional clustering, it does not need us to define the number of clusters to be produced in advance. The outcome is a dendogram, which is a tree-based representation of data points. The following are the two approaches that are used:

  - Agglomerative hierarchical Clustering which starts by treating each observation as a cluster on its own and iteratively identify and merge two clusters that are closest together.
  - Divisive Hierarchical Clustering which starts by treating the whole dataset as one cluster and iterative identify and split dissimilar points from the big cluster to form other smaller clusters.

- Fuzzy clustering is a type of clustering in which each data point can be assigned to many clusters depending on a membership value. It is referred to as soft clustering due to its nature of clustering.

- Density Based Clustering attempts to identify clusters by detecting regions where points are concentrated and where they are sparse (empty). Points that do not belong to a cluster are labeled as noise.

  - Density-Based Spatial Clustering of Applications with Noise (DBSCAN)classifies points into core points, border points and noise based on some specified parameters such as min points(minimum number of points needed to form a cluster) and epsilon (a small neighborhood around any given point).

- Model Based Clustering assumes that a model created the dataset and attempts to reconstruct the original model from the data. A common criterion used in the estimation of model parameters is maximum-likelihood. One technique used in Model Based Clustering is:

  - The Gaussian Mixture Model which seeks to discover a combination of multi-dimensional Gaussian probability distributions that best models the input dataset.

Chapter 2

Proximity Measure

Proximity measurements are mostly mathematical approaches for calculating data point similarity and dissimilarity. Typically, proximity is evaluated in terms of similarity or dissimilarity, or how similar two things are. This dissertation outlines two of such methods discussed bellow:

2.1 Euclidean Norm

Given a vector $x \in R^n$, the Euclidean norm is defined as the square root of the sum of squares of its elements. Mathematically, the Euclidean norm on a vector x is represented as:

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

2.2 Euclidean Distance

Given two vectors $p, q \in R^n$. The Euclidean distance from p to q is defined as the Euclidean norm on the difference between p and q expressed mathematically as:

$$d = \|p - q\|_2 = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

In this dissertation, the Euclidean distance is employed under Density Based Clustering with Application and Noise (DBSCAN).

2.3 Squared Euclidean Distance

The squared Euclidean distance is the square of the Euclidean distance formula. That is to say that the squared Euclidean distance is the same as the Euclidean distance formula but does not take the square root. It is expresed mathematically as:

$$d = \|x - y\|_2^2 = \sum_{i=1}^{n} (x_i - y_i)^2$$

Under Kmeans clustering, the squared Euclidean distance is employed in the clustering process.

Chapter 3

KMeans Clustering

K-means clustering is a partitional clustering method that attempts to partition or divide the datasets into a number of pre-defined distinct non-overlapping subgroups where each data point belongs to one and only one group[Imad, 2018].

3.1 The K-Means Clustering Process

We illustrate the K-means clustering process by considering the dataset below:

Observation A	1	1
Observation B	2	1
Observation C	4	3
Observation D	5	4
Observation E	1	2
Observation F	4	4

Table 3.1: A table showing a sample data

The tuples **A(1 , 1)**, **B(2 , 1)**, **C(4 , 3)**, **D(5 , 4)**, **E(1 , 2)**, **F(4 , 4)** can be generated from Table 3.1

1. *Specify number of clusters* k *and Choose* k *initial centroids.*
   Here we want two(2) clusters so we choose the points for Observation A and Observation B as our initial centroids. That is C^1(1,1) and C^2(2,1).

2. *Find the sum of the squared distances between the data points and the centroids..*

	X	$\|X-(1,1)\|$	$\|X-(2,1)\|$
A	(1,1)	$(1-1)^2+(1-1)^2=0$	$(1-2)^2+(1-1)^2=1$
B	(2,1)	$(2-1)^2+(1-1)^2=1$	$(2-2)^2+(1-1)^2=0$
C	(4,3)	$(4-1)^2+(3-1)^2=13$	$(4-2)^2+(3-1)^2=8$
D	(5,4)	$(5-1)^2+(4-1)^2=25$	$(5-2)^2+(4-1)^2=18$
E	(1,2)	$(1-1)^2+(2-1)^2=1$	$(1-2)^2+(2-1)^2=2$
F	(4,4)	$(4-1)^2+(4-1)^2=18$	$(4-2)^2+(4-1)^2=13$

3. *Assign each observation to the closest cluster on the basis of smallest distance to cluster's centroid.*

	X	$\|X-(1,1)\|$	$\|X-(2,1)\|$	Cluster
A	(1,1)	0	1	C_1
B	(2,1)	1	0	C_2
C	(4,3)	13	8	C_2
D	(5,4)	25	18	C_2
E	(1,2)	1	2	C_1
F	(4,4)	18	13	C_2

4. *Recompute the center of each cluster by taking the aritmetic mean of all observations belonging to a specified cluster.*

$$C^1_{new} = \frac{A+E}{2} = \frac{(1,1)+(1,2)}{2} = \frac{(2,3)}{2} = (1,1.5)$$

$$C^2_{new} = \frac{B+C+D+F}{4} = \frac{(2,1)+(4,3)+(5,4)+(4,4)}{4} = \frac{(15,12)}{4} = (3.75,3)$$

5. *Repeat step 2, 3, and 4 until no data point changes cluster or centroids do not change values.*

	X	$\|X-(1,1.5)\|$	$\|X-(3.75,3)\|$
A	(1,1)	$(1-1)^2+(1-1.5)^2$=0.25	$(1-3.75)^2+(1-3)^2$=11.5625
B	(2,1)	$(2-1)^2+(1-1.5)^2$=1.25	$(2-3.75)^2+(1-3)^2$=7.0625
C	(4,3)	$(4-1)^2+(3-1.5)^2$=11.25	$(4-3.75)^2+(3-3)^2$=0.0625
D	(5,4)	$(5-1)^2+(4-1.5)^2$=22.25	$(5-3.75)^2+(4-3)^2$=2.5625
E	(1,2)	$(1-1)^2+(2-1.5)^2$=0.25	$(1-3.75)^2+(2-3)^2$=8.5625
F	(4,4)	$(4-1)^2+(4-1.5)^2$=15.25	$(4-3.75)^2+(4-3)^2$=1.0625

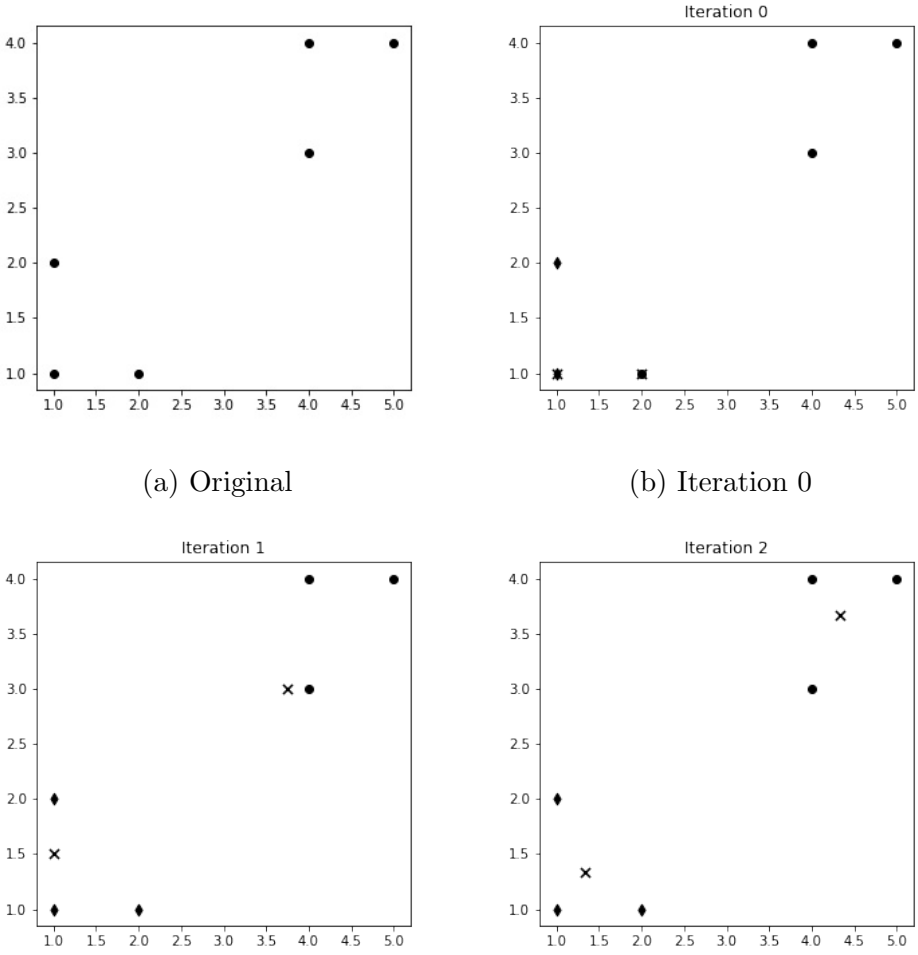
	X	$\|X-(1,1)\|$	$\|X-(2,1)\|$	Cluster
A	(1,1)	0.25	11.562	C_1
B	(2,1)	1.25	7.0625	C_1
C	(4,3)	11.25	0.0625	C_2
D	(5,4)	22.25	2.5625	C_2
E	(1,2)	0.25	8.5625	C_1
F	(4,4)	15.25	1.0625	C_2

$$C^1_{new} = \frac{A+B+E}{3} = \frac{(1,1)+(2,1)+(1,2)}{2} = \frac{(4,4)}{3} = (1.33, 1.33)$$
$$C^2_{new} = \frac{C+D+F}{3} = \frac{(4,3)+(5,4)+(4,4)}{3} = \frac{(13,11)}{3} = (4.33, 3.67)$$

	X	$\lvert X-(1.33,1.33)\rvert$	$\lvert X-(4.33,3.67)\rvert$
A	(1,1)	$(1-1.33)^2+(1-1.33)^2$=0.2178	$(1-4.33)^2+(1-3.67)^2$=18.2178
B	(2,1)	$(2-1.33)^2+(1-1.33)^2$=0.5578	$(2-4.33)^2+(1-3.67)^2$=12.5578
C	(4,3)	$(4-1.33)^2+(3-1.33)^2$=9.9178	$(4-4.33)^2+(3-3.67)^2$=0.5578
D	(5,4)	$(5-1.33)^2+(4-1.33)^2$=20.5978	$(5-4.33)^2+(4-3.67)^2$=0.5576
E	(1,2)	$(1-1.33)^2+(2-1.33)^2$=0.5578	$(1-4.33)^2+(2-3.67)^2$=13.8778
F	(4,4)	$(4-1.33)^2+(4-1.33)^2$=14.2578	$(4-4.33)^2+(4-3.67)^2$=0.2178

	X	$\lvert X-(1.33,1.33)\rvert$	$\lvert X-(4.33,3.67)\rvert$	Cluster
A	(1,1)	0.2178	18.2178	C_1
B	(2,1)	0.5578	12.5578	C_1
C	(4,3)	9.9178	0.5578	C_2
D	(5,4)	20.5978	0.5776	C_2
E	(1,2)	0.5578	13.8778	C_1
F	(4,4)	14.2578	0.2178	C_2

$$C^1_{new} = \frac{A+B+E}{3} = \frac{(1,1)+(2,1)+(1,2)}{2} = \frac{(4,4)}{3} = (1.33, 1.33)$$
$$C^2_{new} = \frac{C+D+F}{3} = \frac{(4,3)+(5,4)+(4,4)}{3} = \frac{(13,11)}{3} = (4.33, 3.67)$$



(a) Original

(b) Iteration 0

(c) Iteration 1

(d) Iteration 2

Figure 3.1: Plots showing the iteration process of the Kmeans algorithm.

We see that the iteration process of the Kmeans algorithm came to a stop after the second iteration as the algorithm converged. That is to say that the respective cluster centroids did not change values and also no point changed cluster after the second iteration.

3.2 K-means As An Optimization Problem

3.2.1 The Objective Function

The main objective function of the K-Means algorithm is given by:

$$J = \sum_{j=1}^{k} \sum_{i:x_i \in j} \|x_i - \mu_j\|$$
$$= \sum_{j=1}^{K} \sum_{i=1}^{n} w_{ij} \|x_i - \mu_j\|$$

where x_i is the i^{th} data point, μ_j is the center of the j^{th} cluster and
$w_{ij} = \begin{cases} 1, & \text{if } x_i \text{ is assigned to cluster } j. \\ 0, & \text{otherwise.} \end{cases}$

The problem here is a minimization problem in two parts. We first want to minimize J w.r.t w_{ij} by treating μ_j as a constant. Then we minimize J w.r.t μ_j by treating w_{ij} as a constant. We want to choose the optimal weights w_{ij} for a fixed center μ_j. We call this step the expectation step (E-step). Also, we want to choose the optimal center μ_j for a fixed weight w_{ij}. We call this step the maximization step (M-step).

3.2.2 The Expectation Step

Here, we minimize J by holding μ_k constant and optimizing w_{ij}

$$w_{ij} = \begin{cases} 1, & \text{if } j = \arg\min_l \|x_i - \mu_l\|^2. \\ 0, & \text{otherwise.} \end{cases}$$

That is, the data point x_n is assigned to the closest cluster with centroid μ_k with respect to the sum of squared Euclidean distance.

3.2.3 The Maximization Step

We continue by taking the partial derivative of J with respect to μ_j given as:

$$\frac{\partial J}{\partial \mu_j} = \frac{\partial \sum_{i=1}^{n} w_{ij} \|x_i - \mu_j\|}{\partial \mu_j}$$

But

$$\begin{aligned}\|x_i - \mu_j\| &= (x_i - \mu_j)^T(x_i - \mu_j) \\ &= x_i^T x_i - x_i^T \mu_j - \mu_j^T x_i + \mu_j^T \mu_j \\ &= x_i^T x_i - 2x_i^T \mu_j + \mu_j^T \mu_j\end{aligned}$$

So

$$\begin{aligned}\frac{\partial J}{\partial \mu_j} &= \frac{\partial \sum_{i=1}^{n} w_{ij}(x_i^T x_i - 2x_i^T \mu_j + \mu_j^T \mu_j)}{\partial \mu_j} \\ &= \sum_{i=1}^{n} w_{ij}(\frac{\partial x_i^T x_i}{\partial \mu_j} - 2\frac{\partial x_i^T \mu_j}{\partial \mu_j} + \frac{\partial \mu_j^T \mu_j}{\partial \mu_j}) \\ &= \sum_{i=1}^{n} w_{ij}(-2x_i + 2\mu_j) \\ &= -2\sum_{i=1}^{n} w_{ij}x_i + 2\mu_j \sum_{i=1}^{n} w_{ij}\end{aligned}$$

Setting $\frac{\partial J}{\partial \mu_j} = 0$

$$\begin{aligned}&\Rightarrow \quad -2\sum_{i=1}^{n} w_{ij}x_i + 2\mu_j \sum_{i=1}^{n} w_{ij} = 0 \\ &\Rightarrow \quad -2\sum_{i}^{n} w_{ij}x_i = -2\mu_j \sum_{i}^{n} w_{ij} \\ &\Rightarrow \quad \mu_j = \frac{\sum_i w_{ij}x_i}{\sum_i w_{ij}}\end{aligned}$$

Now we let

$$\sum_{i}^{n} w_{ij} = n_j$$

Then

$$\mu_j = \frac{\sum_{i:x_i \in j} x_i}{n_j}$$

For each k in j, the matrix of second derivatives is given as:

$$\begin{aligned}\frac{\partial^2 J}{\partial \mu_j^2} &= \frac{\partial \sum_i w_{ij}(-2x_i + 2\mu_j)}{\partial \mu_j} \\ &= 2\sum_{i} w_{ij} I > 0\end{aligned}$$

3.3 Pros And Cons of the Kmeans Algorithm

3.3.1 PROS

Some advatages of Kmeans algorithm are outlined below:

- The Kmeans algorithm is relatively simple to implement as compared to other methods.
- K-Means produces clusters that are simple to understand and even visualize. Because of its simplicity, it may be quite beneficial in situations when you require a rapid overview of the data parts.
- The K-Means approach has a linear time complexity and can be easily applied to huge datasets.

3.3.2 CONS

Some disadvatages of Kmeans algorithm are outlined below:

- The Kmeans algorithm requires the user to specify the optimal number of clusters. This can be a pain if no domain expertice or knowledge of dataset is present.
- The Kmeans algorithm is extremely sensitive to the initialization of centroids. As a result the presence of a noise point or outlier can lead to poor clustering, since outliers are not detected and eliminated in Kmeans.
- The Kmeans algorithm uses a distance-based similarity measure which converges to a constant value between any given example with increasing number of dimensions.
- The Kmeans algorithm performs poorly on clusters of different densities and sizes. As a result a gfeneralized kmeans is adopted to ensure good clustering.

3.4 Elbow Method

In k-means clustering, the elbow method is a parameter tunnin technique used to estimate the ideal number of clusters. The elbow method plots the inertia produced by different values of k. As k increases, inertia falls, each cluster has fewer constituent instances, and the instances are more closer to their respective centroids. As k gets bigger, however, the improvements in inertia slows. The elbow is that specific value of k at which the improvement in inertia diminishes the most, and at which we should cease splitting the data into more clusters.

3.4.1 Inertia

Inertia tells us how far away data points are within a cluster. It is computed by finding the sum of squared deviations from each observation and its cluster centroid. Mathematically inertia is calculated as:

$$wcss = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\| \tag{3.1}$$

An excellent choice of model is one with a low value of inertia as well as a small number of clusters (K). However, when K increases, inertia reduces, thus a tradeoff. For any given dataset, we attempt using the Elbow method, to discover the optimal value for K by identify the point where the drop in inertia begins to slow.

- We will pick a range of numbers as respective candidates of the optimal clusters. Here we pick [1,2,3,4,5].
- Compute within-cluster sum of squares for each case using equation (3.1).

Case 1: *Number of clusters = 1*

In this case all observations belongs to the same cluster with center (2.8333, 2.5)

	X	$\|X - (2.8333, 2.5)\|$	Cluster
A	(1,1)	6.1099	C_1
B	(2,1)	2.9439	C_1
C	(4,3)	1.6119	C_1
D	(5,4)	6.9459	C_1
E	(1,2)	3.6099	C_1
F	(4,4)	3.6119	C_1
Total		**24.3334**	

So wcss = **24.3334**

Case 2: *Number of clusters = 2*

In this case we have some observation A, B and E belonging to cluster 1 with center (1.3333, 1.3333) and observation C, D and F in cluster 2 with center (4.3333, 3.6667)

	X	Cluster	$\lvert X-(1.3333,1.3333)\rvert$	$\lvert X-(4.3333,3.6667)\rvert$
A	(1,1)	C_1	0.222	
B	(2,1)	C_1	0.5554	
C	(4,3)	C_2		0.5556
D	(5,4)	C_2		0.5556
E	(1,2)	C_1	0.556	
F	(4,4)	C_2		0.2222
Total			**1.3334**	**1.3334**

That is wcss = 1.3334 + 1.3334 = **2.6668**

Case 3: *Number of clusters = 3*

Here, we have observation A and B in cluster 1 with center (1.5, 1), observation C, D and F in cluster 2 with center (4.3333, 3.6667) and finally observation E in cluster 3 with center (1, 2).

cluster 1 has inertia of 0.5, cluster 2 has inertia of 1.3333 and cluster 3 has inertia of 0.

Together, wcss = 0.5 + 1.3333 + 0 = **1.8333**

Case 4: *Number of clusters = 4*

Here, we have observations A and B in cluster 1 with center (1.5, 1), observations C and F in cluster 2 with center (4, 3.5), observation D in cluster 3 with center (5, 4) and finally observation E in cluster 3 with center (1, 2).

cluster 1 has inertia of 0.5, cluster 2 has inertia of 0.5, cluster 3 has inertia of 0 and cluster 4 has inertia of 0.

Together, wcss = 0.5 + 0.5 + 0 + 0 = **1**

Case 5:*Number of clusters = 5*

Here, we have observations A and B in cluster 1 with center (1.5, 1), observation F in cluster 2 with center (4, 4), observation C in cluster 3 with center (4, 3), observation D in cluster 4 with center (5,4) and finally observation E in cluster 5 with center (1, 2).

cluster 1 has inertia of 0.5, cluster 2 has inertia of 0, cluster 3 has inertia of 0, cluster 4 has inertia of 0 and cluster 5 has inertia of 0.

Together, wcss = 0.5 + 0 + 0 + 0 + 0 = **0.5**

- Plot the respective number of clusters against their inertia.

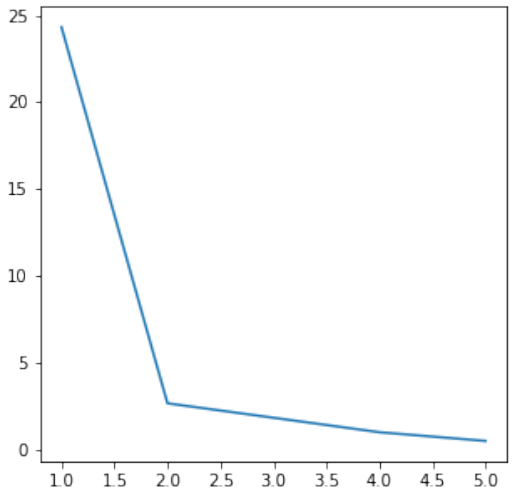


Figure 3.2: A figure showing a plot of the dataset above using Elbow method.

We see from Figure 3.2 above that an elbow is formed when number of clusters is equal to two. That is to say that the optimal number of clusters is two (2) and anything other than that will result to bad clustering. In a scenario where datasets are large, the technique discussed below is employed in identifying the elbow point.

3.4.2 Shortest Distance Between a point and a line

The shortest distance between a point P and a line l_0 is measured along the line that passes through the point and meet the original line l_0 at right angles. Mathematically, the shortest distance between a point (x, y) and a line passing through $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ is given as

$$d(P_1, P_2, (x, y)) = \frac{|(x_2 - x_1)(y_1 - y) - (x_1 - x)(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{3.2}$$

or

$$d(ax + by + c, (x_1, y_1)) = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}$$

Consider the tables below showing the wcss of the respective clusters:

No.of clusters	WCSS
1	24.3334
2	2.6668
3	1.8333
4	1
5	0.5

From Figure 3.2, we want to compute the perpendicular distance between the tuple (x, y) from the set $\{A(1, 24.3334), B(2, 2.6668), C(3, 1.8333), D(4, 1), E(5, 0.5)\}$ and the line (l_0) passing through the points $A(1, 24.3334)$ and $E(5, 0.5)$ using equation (3.2).

$$d(l_0, A) = \frac{|(5-1)(24.3334-24.3334)-(1-1)(0.5-24.3334)|}{\sqrt{(5-1)^2+(24.3334-0.5)^2}} = 0.0000$$

$$d(l_0, B) = \frac{|(5-1)(24.3334-2.6668)-(1-2)(0.5-24.3334)|}{\sqrt{(5-1)^2+(24.3334-0.5)^2}} = 2.6100$$

$$d(l_0, C) = \frac{|(5-1)(24.3334-1.8333)-(1-3)(0.5-24.3334)|}{\sqrt{(5-1)^2+(24.3334-0.5)^2}} = 1.7517$$

$$d(l_0, D) = \frac{|(5-1)(24.3334-1)-(1-4)(0.5-24.3334)|}{\sqrt{(5-1)^2+(24.3334-0.5)^2}} = 0.9035$$

$$d(l_0, E) = \frac{|(5-1)(24.3334-0.5)-(1-5)(0.5-24.3334)|}{\sqrt{(5-1)^2+(24.3334-0.5)^2}} = 0.0000$$

In a tabular form we have:

No.of clusters	WCSS	Shortest Distance
1	24.3334	0
2	2.6668	**2.6100**
3	1.8333	1.7517
4	1	0.9035
5	0.5	0

Thus since working with 2 clusters produce the largest shortest distance we see that an elbow is formed at No. of clusters = 2, so we choose k=2 as our optimal number of clusters.

3.5 The Silhouette Score Analysis

The silhouette score is an alternative to the Elbow method used to evaluate the quality of clusters created. To calculate the silhouette score, the following distances must be computed.

- mean intra-cluster distance
- mean nearest-cluster distance

3.5.1 Mean Intra-Cluster Distance

This measures the mean distance between an observation and all other data points in the same cluster. This distance is denoted by $\boldsymbol{a}$.

For any data point $i \in C_i$,

$$\mathbf{a(i)} = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} |i - j|^2 \tag{3.3}$$

- C_i is the ith cluster.
- Divide by $|C_i| - 1$ because we do not include the data point i in the computation.
- The result measures how well the data point i is assigned to its own cluster.

3.5.2 Mean Nearest-Cluster Distance

This measures the mean distance between an observation and all other data points of the next cluster. This distance is denoted by $\boldsymbol{b}$.

For each data point $i \in C_i$, define

$$\mathbf{b(i)} = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} |i - j|^2 \tag{3.4}$$

- minimum because we seek to compare with the closest neighboring cluster.
- measures the dissimilarity of i to all data points in another cluster of which i is not a member of

3.5.3 The Silhouette value of A Point

We define the silhouette value of a data point i as follows:

- if $|C_i| > 1$

$$\mathbf{S(i)} = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

- if $|C_i| = 1$

$$\mathbf{S(i)} = 0$$

- $\mathbf{S(i)} = \begin{cases} 1 - \frac{a(i)}{b(i)}, & \text{if } a(i) < b(i). \\ 0, & \text{if } a(i) = b(i). \\ \frac{b(i)}{a(i)} - 1, & \text{if } a(i) > b(i) \end{cases}$

- We conclude that the silhouette value of a data point i ranges as $-1 < S(i) < 1$
- If the score is 1, then the cluster is dense and well separated from the neighboring clusters.
- A value closer to 0 represents overlapping clusters with observations very close to decision boundary of neighboring clusters.
- A negative score indicate that observations are wrongly assigned to the respective clusters.

Consider the table below:

Observation A	(1,1)	C_1
Observation B	(2,1)	C_1
Observation C	(4,3)	C_2
Observation D	(5,4)	C_2
Observation E	(1,2)	C_1
Observation F	(4,4)	C_2

Table 3.2: A table showing points and their respective clusters

C_1 is the set $\{(1,1),(2,1),(1,2)\}$ and C_2 is the set $\{(4,3),(5,4),(4,4)\}$

We will calculate the silhoutte score for A(1,1) using equations (3.3) and (3.4).

	(2,1)	(4,3)	(5,4)	(1,2)	(4,4)	
(1,1)	1	13	25	1	18	
a(1,1)	0.5			0.5		1
b(1,1)		4.3333	8.3333		6	18.6667
S(1,1)						0.9464

where

$$\begin{aligned} \mathbf{S((1,1))} &= \frac{b(1,1) - a(1,1)}{max\{a(1,1), b(1,1)\}} \\ &= \frac{18.6667 - 1}{18.6667} \\ &= \frac{17.6667}{18.6667} \\ &= 0.9464 \end{aligned}$$

Thus the silhouette score of the data point (1,1) is 0.9464 which signifies that the data point (1,1) is correctly classified.

3.6 The K-means ++ Algorithm

The K-means method has the drawback of being sensitive to the initialization of the centroids or mean points, which makes it difficult to use. A centroid that is initialized as a "far-off" point may end up with no points connected with it, while at the same time more than one cluster may end up associated with a single centroid. Several more centroids may be initialized into the same cluster, resulting in erroneous clustering. This is as a result of the randomness in initializing our centroids.

To overcome the aforementioned drawback, we use K-means++ to attempt to eliminate the randomness in initializing our cluster centers. The cluster centers are better initialized as a result of this method, and the clustering is improved. Except for initialization, the rest of the technique is identical to the standard K-means algorithm. As a consequence, K-means++ combines the traditional K-means approach with improved centroid initialization[Savya, 2018].

- Take a center c^1, randomly chosen from the dataset.
- Calculate $D(x)$, the distance between x and the nearest center that has already been picked, for each data point x that has not been chosen yet.
- Using a weighted probability distribution, choose one new data point at random as a new center, with a probability proportional to the distance squared (D(x)2).
- Repeat Steps 2 and 3 until k centers have been chosen.

Chapter 4

Density-Based Spatial Clustering Of Applications With Noise (DBSCAN)

When it comes to unsupervised clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular technique that was first presented in 1996. DBSCAN does not require the number of clusters to be specified, unlike the K-mean. Based on your supplied data and parameters, it can automatically determine the number of clusters.The parameters used are ϵ and minimum points (MinPts). Furthermore, DBSCAN is able to detect clusters of any shape that k-means cannot(consider a cluster that is enclosed by another cluster).

4.1 Definitions

4.1.1 Minimum Number of Samples (MinPts)

The minimum number of samples is the smallest number of points that is required for the formation of a cluster.

4.1.2 Epsilon (ϵ)

Epsilon defines the maximum distance any two points or samples can be from each other while still belonging to the same cluster.

4.1.3 Core Point

A point p $\in$ D is a core point if $N_\epsilon(p)$ covers at least MinPts points of D (including p itself).

4.1.4 Boundary Point

A point p is said to be a boundary point if $N_\epsilon(p)$ covers points less than MinPts

4.1.5 ϵ-Neighborhood Of A Point

The ϵ-neighborhood of a point p, denoted by $N_\epsilon(P)$, is defined by $N_\epsilon(P) = \{q \in D | dist(p,q) < \epsilon\}$. An ϵ-neighborhood of a boundary point often comprises fewer points than an ϵ-neighborhood of a core point.

4.1.6 Directly Density-Reachable

A point q is directly density reachable from a point p if p is a core point and $q \in N_\epsilon$. That is to say that for some ϵ and MinPts;

1. $q \in N_\epsilon(p)$
2. $|N_\epsilon(p)| > Minpts$ (core point condition)[Ester et al., 1996]

4.1.7 Density-Reachable

A point q is density reachable from a point p with respect to ϵ and MinPts if there is a series of points $P_1, ..., P_n$, $P_1 = p, P_n = q$ such that P_{i+1} is directly density-reachable from P_i[Ester et al., 1996]. In simpler terms, a point q is density reachable from another point p if they are connected through a series of core points.

4.1.8 Density-Connected

A point p is density connected to a point q with respect to ϵ and MinPts if there is a point r such that both, p and q are density-reachable from r with respect to ϵ and MinPts.

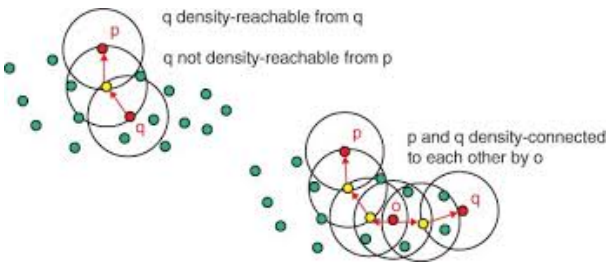


Figure 4.1: Density-reachability and Density-connectivity

4.1.9 Cluster

Let D be a set of data points. A cluster C with respect to ϵ and MinPts is a non-empty subset of D satisfying the conditions of maximality and connectivity; that is is to say that:

1. (Maximality) If a core point p $\in$ C, then all the points density-reachable from p also belong to C.
2. (Connectivity) $\forall p, q \in C, \exists r : p, q$ are density-reachable from r [Ester et al., 1996].

with respect to ϵ and MinPts

4.1.10 Noise

Let $C_1, ..., C_k$ be the clusters of the database D with respect to parameters ϵ_i and $MinPts_i$, i = 1, ..., k. Then we define noise as the set of points in the database D not belonging to any cluster Ci, that is to say that noise = $\{p \in D | \forall i : p \notin C_i\}$ [Ester et al., 1996].

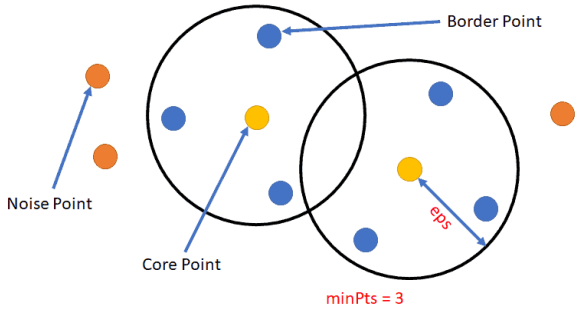


Figure 4.2: Core points, Border points and Noise

4.2 The Algorithm

A starting point is chosen at random from its immediate neighborhood, which is defined by radius epsilon (ϵ). The point is tagged as a core point and a cluster formation begins if there are at least minPts number of points in the neighborhood. If this is not the case, the point is labeled as noise. When a cluster formation, say cluster A begins, all points in the immediate neighborhood of the initial point is added to cluster A. If these new points are also core points, the points in their immediate neighborhood are added to cluster A as well.

A point labeled as noise may be revisited and found to be part of a cluster. The next stage is to pick another spot at random from the ones that haven't been visited yet in the previous steps. Then the same procedure is repeated for all points which have not been

visited. When all the points have been visited, the procedure is complete. The steps are outlined below:

1. Select an ϵ and MinPts value.
2. Calculate the distance between a data point (x) and the other data points.
3. Find all of x's neighbors that are inside the circle of radius ϵ.
4. Treat x as visited, and a core point if the number of nearby points is higher than or equal to MinPts. If x isn't assigned to any cluster, create a new one and assign it to that cluster.
5. Treat x as a border point if the number of neighborhood points around it is fewer than MinPts and it has a core point in its neighborhood.
6. As a single cluster, combine all the density-connected points.
7. Find all core, boundary, and outlier points in the data set by repeating the preceding procedures for every unvisited point in the data set.

Consider the dataset consisting of thirteen (13) points:

X	Y
1	2.5
2.5	4.5
1	3.5
2.5	5.5
2.3	5
2	5
0.7	3
0.5	2.5
0.5	5.5
0.5	3.5
4.5	6.5
3.5	3.5
2.8	5

Table 4.1: This is a sample data consisting of thirteen points.

We use the DBSCAN technique to discover clusters in the aforementioned dataset. First, we must choose the values for eps and MinPts. Let's use eps = 0.6 and MinPts = 4 as our parameters. Let's look at the first data point in the dataset (1,2) and see how far it is from the remaining data points in the set. The following are the calculated values:

X	Y	$\|(X,Y)-(1,2.5)\|$
1	2.5	0.0000
2.5	4.5	2.5000
1	3.5	1.0000
2.5	5.5	3.3541
2.3	5	2.8178
2	5	2.6926
0.7	3	0.5831
0.5	2.5	0.5000
0.5	5.5	3.0414
0.5	3.5	1.1180
4.5	6.5	5.3151
3.5	3.5	2.6926
2.8	5	3.0806

Table 4.2: A table showing the euclidean distances of the point (1,2.5) and the dataset.

From the table above, the point (1, 2.5) has only two additional points in its immediate neighborhood i.e (0, 7.3) and (0.5, 2.5) for the assumed value of ϵ; nonetheless, because it is smaller than MinPts, we cannot tag it as a core point. Repeating the technique above for each point in the dataset to determine its neighbors, we obtain the following results:

Point	Points in Neighborhood
(1,2.5)	(0.7,3) (0.5,2.5)
(2.5,4.5)	
(1,3.5)	
(2.5,5.5)	
(2.3,5)	
(2,5)	
(0.7,3)	(1,2.5)
(0.5,2.5)	(1,2.5)
(0.5,5.5)	
(0.5,3.5)	
(4.5,6.5)	
(3.5,3.5)	
(2.8,5)	

Table 4.3: Table showing points in the neighbourhood of respective points in the dataset after comparing the euclidean distances with ϵ.

Point	Points in Neighborhood
(1,2.5)	(0.7,3) (0.5,2.5)
(2.5,4.5)	(2.3,5) (2,5) (2.8,5)
(1,3.5)	(0.7,3) (0.5,3.5)
(2.5,5.5)	(2.3,5) (2,5) (2.8,5)
(2.3,5)	(2.5,4.5) (2.3,3.5) (2,5) (2.8,5)
(2,5)	(2.3,5) (2.5,4.5)
(0.7,3)	(1,2.5) (1,3.5) (0.5,2.5) (0.5,3.5)
(0.5,2.5)	(1,2.5) (0.7,3)
(0.5,5.5)	
(0.5,3.5)	(1,3.5) (0.7,3)
(4.5,6.5)	
(3.5,3.5)	
(2.8,5)	(2.5,4.5) (2.5,5.5) (2.3,5)

Table 4.4: A table showing all the neighbors of the various points in the dataset

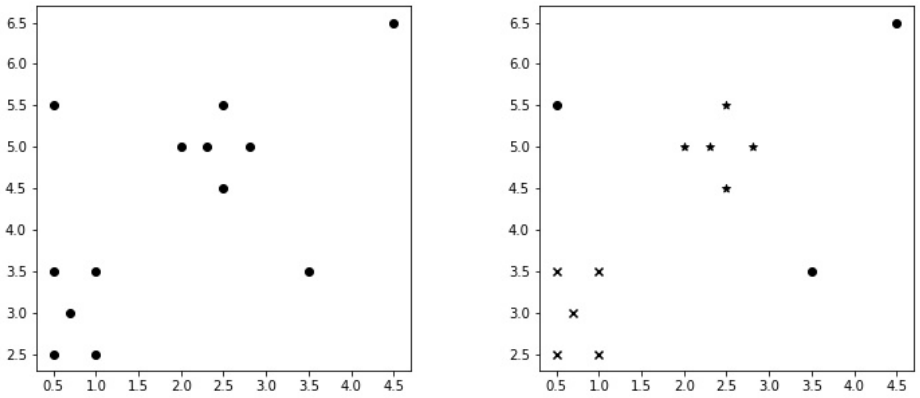
Point	Points in Neighborhood	Type of point
(1,2.5)	2	Border point
(2.5,4.5)	3	Border point
(1,3.5)	2	Border point
(2.5,5.5)	3	Border point
(2.3,5)	**4**	**Core point**
(2,5)	2	Bordejr point
(0.7,3)	**4**	**Core point**
(0.5,2.5)	2	Border point
(0.5,5.5)	0	Outlier
(0.5,3.5)	2	Border point
(4.5,6.5)	0	Outlier
(3.5,3.5)	0	Outlier
(2.8,5)	3	Border point

Table 4.5: A table showing the classification of the various points in the dataset based on the number of neighbors

The DBSCAN algorithm searches the dataset for three types of points: core, boundary, and outliers. Unless some core points share neighboring points, in which case they will be clustered together in the same cluster, each core point will be allocated to a different cluster. Every border point will be allocated to a cluster based on its neighborhood's core point; for example, the first point (1, 2.5) is a border point with a neighborhood core point (0.7, 3) that is included in Cluster 1, hence the point (1, 2.5) will likewise be included in Cluster 1. The categorisation is summarized as follows:

Cluster 1	Cluster 2	Noise
(0.7,3)	**(2.3,5)**	(0.5,5.5)
(1,3.5)	(2.5,5.5)	(4.5,6.5)
(1,2.5)	(2.3,5)	(3.5,3.5)
(0.5,2.5)	(2.5,4.5)	
(0.5,3.5)	(2.8,5)	

Table 4.6: A table showing the categorization of the dataset into respective cluster groups



(a) Original (b) Iteration 0

Figure 4.3: A figure showing the plot of the above dataset 4.1 before and after clustering with DBSCAN.

4.3 Parameter Tuning

DBSCAN as discussed above takes two critical parameters which are epsilon (ϵ) and minimum number of samples (MinPts). Finding the right value of these two parameters can be a pain given large datasets. A good clustering is achieved with a good choice of (ϵ) and minimum number of samples (MinPts). A number of scholars have proposed variety of methods all poised in estimating values for ϵ and MinPts. A guideline is provided below for estimating Minimum number of samples and epsilon repectively:

4.3.1 Estimating the value of Minimum Samples (MinPts)

Usually, there is no laid down method to determine the MinPts value automatically. As a result, in determining the value of MinPts, domain expertise and experience with dataset is employed[Tara, 2020]. There following serves as a guideline in estimating MinPts:

- A large value for MinPts should be chosen if a large datset is being used.

- A large value of MinPts should be chosen if we have a noisier datset.
- Given the dimentionality (d) of a dataset, MinPts should be chosen such a way that the value of MinPts is greater than or equal to the dimensionality of the datset[Tara, 2020].
- if the dataset has 2 or more dimensions, we choose MinPts to be $2 \times d$ [Ester et al., 1996][Sander et al., 1998]

4.3.2 Estimating the value of Epsilon (ϵ)

Once the value for minimum number of samples have been determined, we proceed to estimate the value for ϵ using the estimated value of minimum samples and the K-nearest neighbour algorithm from scikit learn. This method determines the average distance between each point and its k closest neighbors, where k equals the MinPts number chosen. On a k-distance graph, the average k-distances are then shown in ascending order. At the point of maximal curvature, we get the best value for ϵ.

4.3.3 Implementing With the Dataset Above

We saw this dataset from previous sections of Chapter 4:

X	Y
1	2.5
2.5	4.5
1	3.5
2.5	5.5
2.3	5
2	5
0.7	3
0.5	2.5
0.5	5.5
0.5	3.5
4.5	6.5
3.5	3.5
2.8	5

Given the dataset above we will implement the whole idea behind parameter tunning in DBSCAN.

- Since the dataset is 2-dimensional we will choose minimum number of samples (MinPts) to be $2 \times d = 2 \times 2 = 4$.
- With MinPts chosen we proceed to estimate ϵ. Using the K-Nearest Neighbor algoritm we have the following table:

Point	4 Cosest Neighbours
(1,2.5)	(1,2.5) (0.5,2.5) (0.7,3) (1,3.5)
(2.5,4.5)	(2.5,4.5) (2.3,5) (2.8,5) (2,5)
(1,3.5)	(1,3.5) (0.5,3.5) (0.7,3) (1,2.5)
(2.5,5.5)	(2.5,5.5) (2.3,5) (2.8,5) (2,5)
(2.3,5)	(2.3,5) (2,5) (2.8,5) (2.5,4.5)
(2,5)	(2,5) (2.3,5) (2.5,5.5) (2.5, 4.5)
(0.7,3)	(0.7,3) (0.5,3.5) (0.5,2.5) (1,3.5)
(0.5,2.5)	(0.5,2.5) (1,2.5) (0.7,3) (0.5,3.5)
(0.5,5.5)	(0.5,5.5) (2,5) (2.3,5) (2.5,5.5)
(0.5,3.5)	(0.5,3.5) (1,3.5) (0.7,3) (0.5,2.5)
(4.5,6.5)	(4.5,6.5) (2.5,5.5) (2.8,5) (2.3,5)
(3.5,3.5)	(3.5,3.5) (2.5,4.5) (2.8,5) (2.3,5)
(2.8,5)	(2.8,5) (2.3,5) (2.5,5.5) (2.5,4.5)

Table 4.7: A table showing all the neighbors of the various points in the dataset

Point	Distances from respective neighbors			
(1,2.5)	0	0.5	0.58309519	1
(2.5,4.5)	0	0.53851648	0.58309519	0.70710678
(1,3.5)	0	0.5	0.58309519	1
(2.5,5.5)	0	0.53851648	0.58309519	0.70710678
(2.3,5)	0	0.3	0.5	0.53851648
(2,5)	0	0.3	0.70710678	0.70710678
(0.7,3)	0	0.53851648	0.53851648	0.58309519
(0.5,2.5)	0	0.5	0.53851648	1
(0.5,5.5)	0	1.58113883	1.86815417	2
(0.5,3.5)	0	0.5	0.53851648	1
(4.5,6.5)	0	2.23606798	2.26715681	2.66270539
(3.5,3.5)	0	1.41421356	1.65529454	1.92093727
(2.8,5)	0	0.5	0.58309519	0.58309519

Table 4.8: A table showing the sorted distances of between a point and its neighbors.

- Select the smallest distance greater than zero for each point and plot on a k-distance graph.

Index	Sorted Distance
0	0.3
1	0.3
2	0.5
3	0.5
4	0.5
5	0.5
6	0.5
7	0.53851648
8	0.53851648
9	0.53851648
10	1.41421356
11	1.58113883
12	2.23606798

Table 4.9: A table showing the sorted distances of between a point and its neighbors.

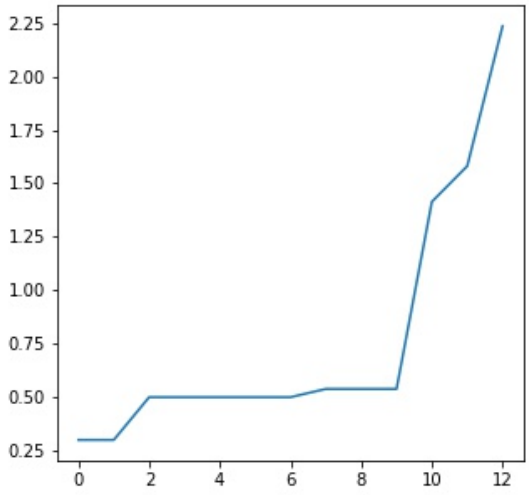


Figure 4.4: A figure showing the k-distance elbow plot.

Clearly, we see from Figure 4.4 that an elbow is formed at the point (9,0.53851648). This means that $\epsilon = 0.53851648$ is a good choice given minimun number of samples as 4.

- Using $\epsilon = 0.53851648$ and $MinPts = 4$, we cluster our datset using DBSCAN. The result is given below:

Cluster 1	Cluster 2	Noise
(0.7,3)	**(2.3,5)**	(0.5,5.5)
(1,3.5)	(2.5,5.5)	(4.5,6.5)
(1,2.5)	(2.3,5)	(3.5,3.5)
(0.5,2.5)	(2.5,4.5)	
(0.5,3.5)	(2.8,5)	

Table 4.10: A table showing the categorization of the dataset into respective cluster groups using our tunned parameters.

Comparing Table 4.6 and Table 4.10, we see that the two tables poduces the same results.

4.4 Pros And Cons of DBSCAN

4.4.1 Pros

Some advantages of the DBSCAN algorithm are outlined below:

- DBSCAN does not require that the number of clusters be specified unlike Kmeans.
- DBSCAN can identify clusters of any shape unlike Kmeans which performs well on spherical datsets.
- DBSCAN is more robust to noise unlike Kmeans. It can identify noise in datsets and eliminates them from clusters in cluster formation.

4.4.2 Cons

Some disadvantages of the DBSCAN algorithm are outlined below:

- The DBSCAN algorithm fails in clustering formation if there no density drop between clusters in dataset. As a result MinPts and ϵ combination cannot be chosen properly for all clusters in dataset.
- The algorithm is very sensitive to parameters. As a result an excellent domain expertise or knowledge of dataset is required to estimate these parameters to ensure good clustering.
- The algorithm usually does not perform well on higher dimentional datasets.

Conclusion

Clustering algorithms are employed when it comes to the task of class identification in spatial databases. The clustering algorithms Kmeans and DBSCAN were presented in this dissertation. The number of clusters, denoted by the letter k, was the only input parameter used by Kmeans. We also saw that DBSCAN uses a density-notion of clustering, and only requires two input parameters: epsilon (ϵ) and the minimum number of samples (MinPts). We saw that both Kmeans and DBSCAN are sensitive to parameter initialization, this was fixed by methods proposed by experts and discussed in this paper.

I look forward to study other methods such as Gauusian Mixture Model, Fuzzy C-means and Hierarchical clustering in future work.

A Python Program for Kmeans

```
import numpy as np
import matplotlib.pyplot as plt

class KMeans():
  def __init__(self, n_clusters=3, init='k-means++',
      max_iter=300, random_state=None):
    self.n_clusters = n_clusters
    self.init = init
    self.max_iter = max_iter
    self.random_state = random_state
    self.cluster_centers_ = None
    self.cluster_labels_ = None
    self.inertia_ = None

  def random_centroids(self, data):
    points = data.copy()
    np.random.shuffle(points)
    return points[:self.n_clusters]

  def _init_kmeans(self, data):
    points = data.copy()
    np.random.shuffle(points)
    centroids = [points[0]]
    while len(centroids) < self.n_clusters:
      distance = np.min([np.linalg.norm(data-c, axis=1)**2
          for c in centroids], axis=0)
      prob = distance/(distance.sum())
      cum_prob = prob.cumsum()
      r_0_1 = np.random.random()
      idx = np.where(cum_prob > r_0_1)
      centroids.append(data[idx[0][0]])
    return np.array(centroids)

  def closest_centroid(self, data, centroids):
    #loop over each centroid and compute the squared distance from point
    distances = np.array([np.linalg.norm(data - c, axis=1)**2
        for c in centroids])
    return np.argmin(distances, axis=0)

  def get_labels(self, data, centroids):
    return self.closest_centroid(data, centroids)

  def update_centroids(self, data, labels):
    #returns the new centroids assigned from the points closest to them
    return np.array([data[labels==k].mean(axis=0)
```

```
46          for k in range(self.n_clusters)])
47
48    def is_converged(self, prev_centroids, cur_centroids):
49      # distances between each old and new centroids, fol all centroids
50      distances = [np.linalg.norm(prev_centroids - cur_centroids, axis=1)]
51      return np.array(distances).sum() == 0
52
53    def fit(self, data):
54      # initialize centroids as k random samples from X
55      if self.init == 'k-means++':
56        centroids = self._init_kmeans(data)
57      elif self.init == 'random':
58        centroids = self.random_centroids(data)
59
60      for _ in range(self.max_iter):
61        #distances = self.closest_centroid(data, centroids)
62        prev_centroids = centroids
63        labels = self.get_labels(data, centroids)
64        centroids = self.update_centroids(data, labels)
65        if self.is_converged(prev_centroids, centroids):
66          break
67      self.cluster_centers_ = centroids
68      self.cluster_labels_ = labels
69      self.inertia_ = sum(np.linalg.norm(self.cluster_centers_[c] - sample)**2
70          for c, sample in zip(self.cluster_labels_, data))
71
72    def fit_predict(self, data):
73      self.fit(data)
74      return self.cluster_labels_
75
76    def predict(self, data):
77      return self.cluster_labels_
```

A Python Program for DBSCAN

```
class Dbscan():
  def __init__(self, eps, MinPts):
    self.eps = eps
    self.MinPts = MinPts

  def getNeighbors(self,data, point):
    neighbors=np.where(np.linalg.norm(data-data[point], axis=1)<self.eps)
    return list(neighbors[0])

  def getClusterLabels(self,data, clusters):
    current = -1
    for i in range(0, data.shape[0]):
      if clusters[i] != None:
        continue
      neighbors = self.getNeighbors(data, i)
      if len(neighbors) < self.MinPts:
        clusters[i] = -1
      else:
        current += 1
        self.expandCluster(data, clusters, i, neighbors, current)
    return clusters

  def expandCluster(self,data, clusters, point, neighbors, current):
    clusters[point] = current
    i = 0
    while i < len(neighbors):
      nextPoint = neighbors[i]
      if clusters[nextPoint] == -1:
        clusters[nextPoint] = current
      elif clusters[nextPoint] == None:
        clusters[nextPoint] = current
        nextNeighbors = self.getNeighbors(data, nextPoint)
        if len(nextNeighbors) >= self.MinPts:
          neighbors = neighbors + nextNeighbors
      i += 1

  def fit_predict(self, data):
    clusters = np.array([None]* data.shape[0])
    self.getClusterLabels(data, clusters)
    return clusters
```

Bibliography

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

[Imad, 2018] Imad, D. (2018). K-means clustering: Algorithm, applications, evaluation methods, and drawbacks. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644 Accessed: 2021-11-13.

[Jason, 2019] Jason, B. (2019). 14 different types of learning in machine learning. https://machinelearningmastery.com/types-of-learning-in-machine-learning/. Accessed: 2021-11-13.

[Lisa, 2017] Lisa, T. (2017). An introduction to machine learning. https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning. Accessed: 2021-11-13.

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

[Sander et al., 1998] Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2(2):169–194.

[Savya, 2018] Savya, K. (2018). K-means++ algorithm. https://www.geeksforgeeks.org/ml-k-means-algorithm/. Accessed: 2021-11-13.

[Tara, 2020] Tara, N. (2020). Dbscan parameter estimation using python. https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd. Accessed: 2021-11-13.