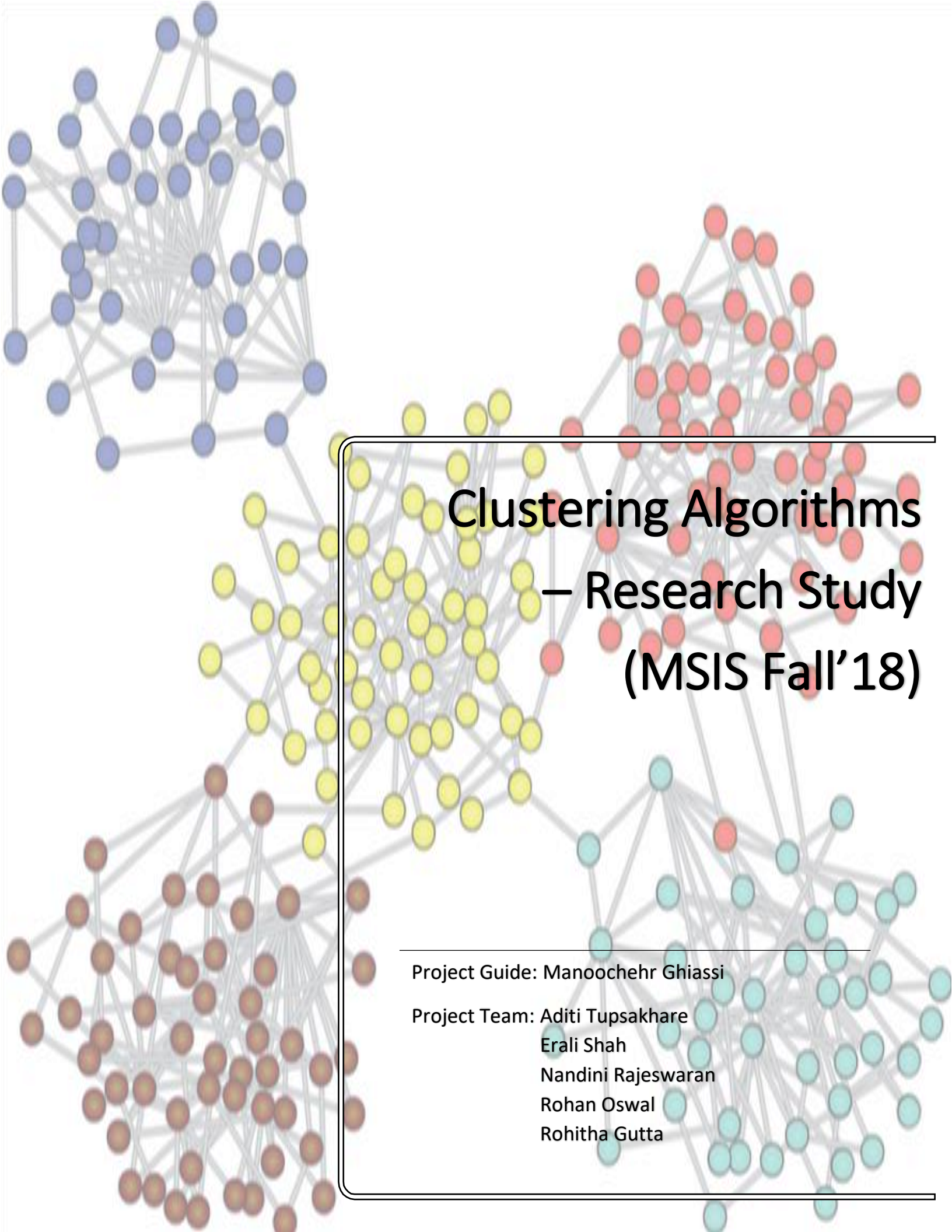


Clustering Algorithms – Research Study (MSIS Fall'18)

Project Guide: Manoochehr Ghiassi

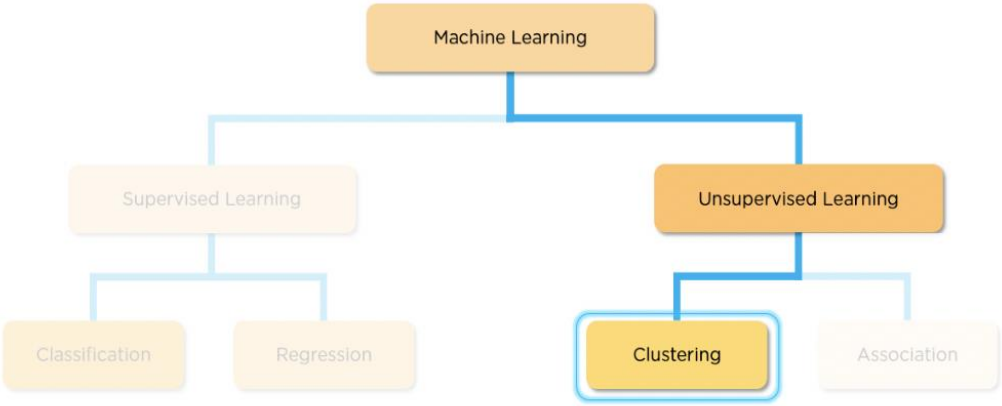
Project Team: Aditi Tupsakhare
Erali Shah
Nandini Rajeswaran
Rohan Oswal
Rohitha Gutta

Table of Contents

1. INTRODUCTION

Big Data is and has always been core to any business. There is a saying; you cannot improve what cannot be measured. Big data analytics helps organizations track their data and identify new opportunities for smarter business moves. So, to extract value from large sets of structured and unstructured data the critical element of Data analysis is to partition the data into logical groups before attempting to analyze it. Clustering plays a significant role here in data analysis for portioning the dataset into subsets, called clusters within which the elements have similar behavior.

Clustering is an unsupervised learning task which draws references from the input datasets without having any labeled responses. There are different categories of clustering, and no clustering technique is universally applicable to any the variety of data sets. Each clustering algorithms works differently for different datasets. It can confuse a user attempting to select the best algorithm suitable the problem at his hand. So, there is a need to identify the best algorithms for each variety of dataset. When a user wants to create a new clustering algorithm, he needs to identify what's the baseline, what's the best algorithm and the drawback of it in order to develop a better algorithm further. Let's dive in deep to understand the concept of clustering.



2. CLUSTERING

Clustering divides the data points within a dataset into the number of different groups such that every data point in respective groups are more similar to that data points in the same and are dissimilar with the data points in the other groups. In simple terms, clustering is a collection of objects arranged into groups depending on the similarity and dissimilarity between them. Clustering uses the input data to understand the pattern, anomalies or dissimilarities in it and segregate them into groups with similar traits. Clustering is significant as it establishes intrinsic grouping in the unlabeled data. There is no

condition or criteria for a clustering technique to be perfect. It is dependent on the user's need to choose on the criteria that will satisfy the need. A good algorithm aims at creating clusters whose - intra-cluster similarity is high and inter-cluster similarity is less.

3. CLUSTERING APPLICATIONS

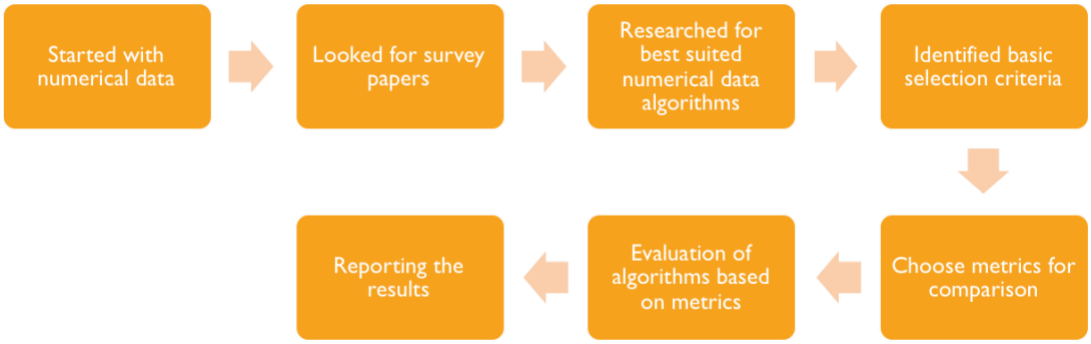
- Clustering analysis is a key intermediate step for other data mining tasks which include
    - Generating a compact summary of data for classification, pattern discovery, hypothesis generation, testing, etc.
    - Outlier detection – to identify the data points far away from any cluster
- Data summarization, compression and reduction. Example - Image Processing
- Collaborative filtering, recommendation systems, or customer segmentation – helps to identify like-minded users or similar products
- Dynamic trend detection – Cluster stream data to detect trends and pattern
- Multimedia data analysis, biological data analysis and social network analysis

4. CLUSTERING ALGORITHMS

In this study, we have considered the following characteristics as the selection criteria –

- *Size of Dataset:* (Large) The size of the dataset has a significant effect on the clustering quality. So, all these clustering methods are more efficient when the data size is large
- *Type of Dataset:* All these clustering algorithms work effectively on numeric data
- *Popularity:* These algorithms are well-known and well-established clustering algorithms
- *Flexibility:* High flexibility concerning the number of training data samples, the dimensionality of each sample vector, number of clusters, different applications
- *Applicability* to the dataset chosen for the study and *availability* of code

Clustering techniques need to be implemented and analyzed for one of the three domains: Numerical, Text or Image. We decided to move forward with Numerical Data. Researched for relevant survey papers and identified 2-3 algorithms in each category which were designed to perform the best for numerical data. Identified and applied the selection criteria to narrow down one algorithm from each of the five clustering techniques. After shortlisting one algorithm from each category, looked for implementation and applied the evaluation metrics to all the five algorithms. Performed test runs on all the shortlisted algorithms to evaluate them based on the metrics considered. Finally, we combined the results to suggest the best algorithm for our set of inputs(datasets).



Let's now discuss the types of clustering algorithms taken into consideration for the research study –

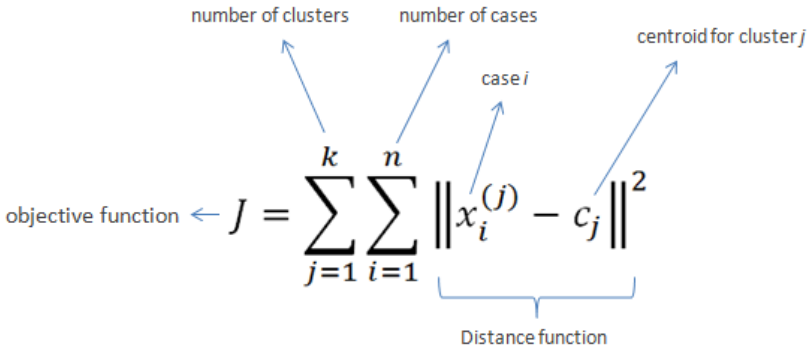
4.1. Partitioning Based Clustering

They are the most popular class of clustering algorithms also known as the iterative relocation algorithm. Partitioning based clustering considers Euclidean distance, i.e. sum of squared errors, while forming clusters. This technique Iteratively relocates data points between clusters until optimal partition attained. For instance, consider n x k, if there are n data points and k cluster Euclidian distance will be calculated between each point and the cluster to identify the optimal partition. It is simply a division of the set of data objects into non-overlapping clusters such that each object is in exactly one subset. Initialization is a crucial step in this clustering as the criterion function (in this case, Euclidean distance) has a non-convex nature, i.e. it can have multiple local optimal solutions. Local optimum is the best solution to a problem while considering a small dataset or subset of dataset for possible solutions. This concept contrasts with the global optimum, which is the optimal solution when every possible solution is considered (in any case). Due to this nature, the quality of the clustering depends on the initial partition. There are two approaches for initialization in partitioning-based technique. One is to run and analyze the algorithm by starting with several initial conditions. The other is to apply some heuristics in advance and find the most optimal initialization. Here, the number of clusters have to be known in advance, thus naming of partitioning based clustering algorithms is usually of the form "k-estimates" (k-means , k-medoids) or "c-estimate"(in case of fuzzy clustering algorithm). It is also due to the tendency to partition any dataset into K fixed number of clusters. Some other advance algorithms under this category are the CLARA and CLARANS.

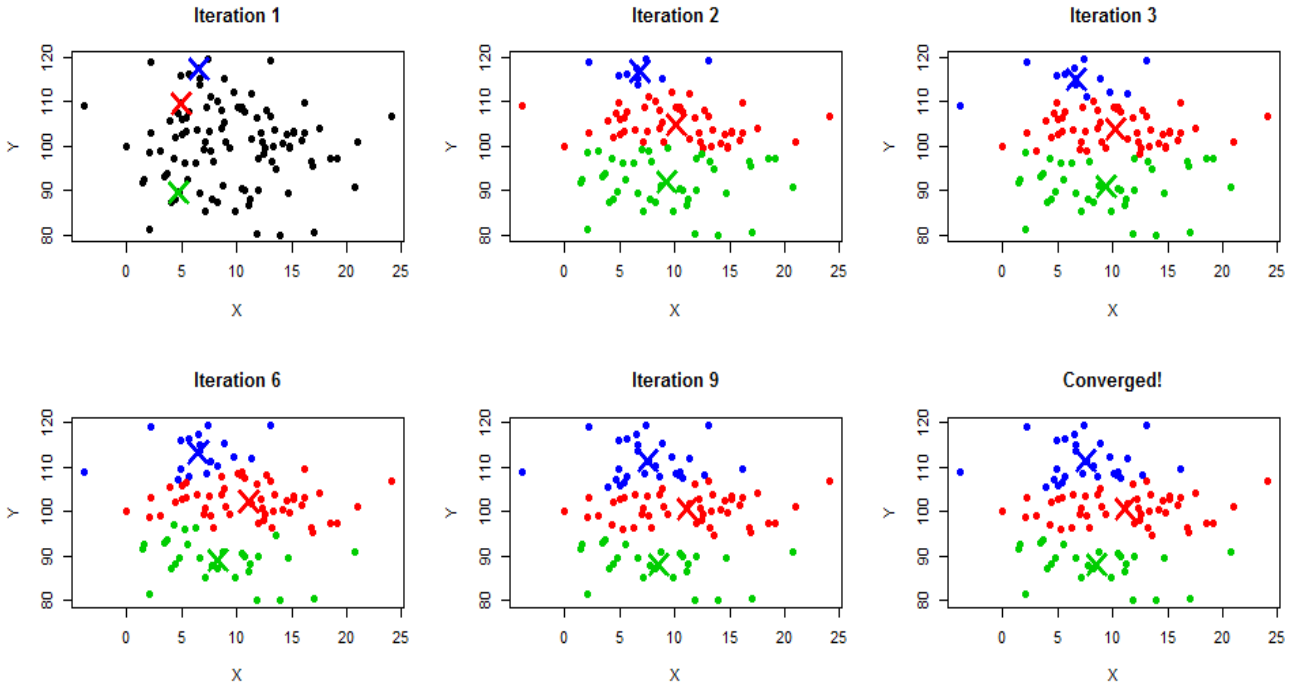
K Means Clustering

K means clustering is one of the simplest unsupervised learning algorithms. K-Means is an exploratory data analysis technique i.e. it will analyze and explore the complete dataset and not just part of it. It won't

follow any particular format of hierarchical methods like agglomerative, but it will take the datasets(objects) as they are coming and will group them together into K clusters. The chief idea is to define K centroids, one for each cluster. The location of centroid in a cluster can influence the clustering significantly. K- Means like any other partitioning based technique determines centroid using Euclidean Distance. The idea of K-Means clustering is to minimize total intra-cluster(within a cluster) variance, or, the squared error function. So, it calculates the centroid using the Euclidean method.



After finding the Euclidean distance, objects are grouped into clusters based on minimum distance obtained.



As seen in the Fig., there are 50 data points with 3 centroids randomly initiated. Iteraation 2 depicts new centroid location. In iteration 3, the number of blue points has increased. Iteration 6 shows centroid for red points moving rightwards. Iteration 9 depicts green cluster points have decreased in size while

comparing it to iteration 2 and blue data points have taken over while thinning the red centroid compared to the one in iteration 6. As per this analysis the results of the 8th and 9th iteration was same. So, we can say that it has converged.

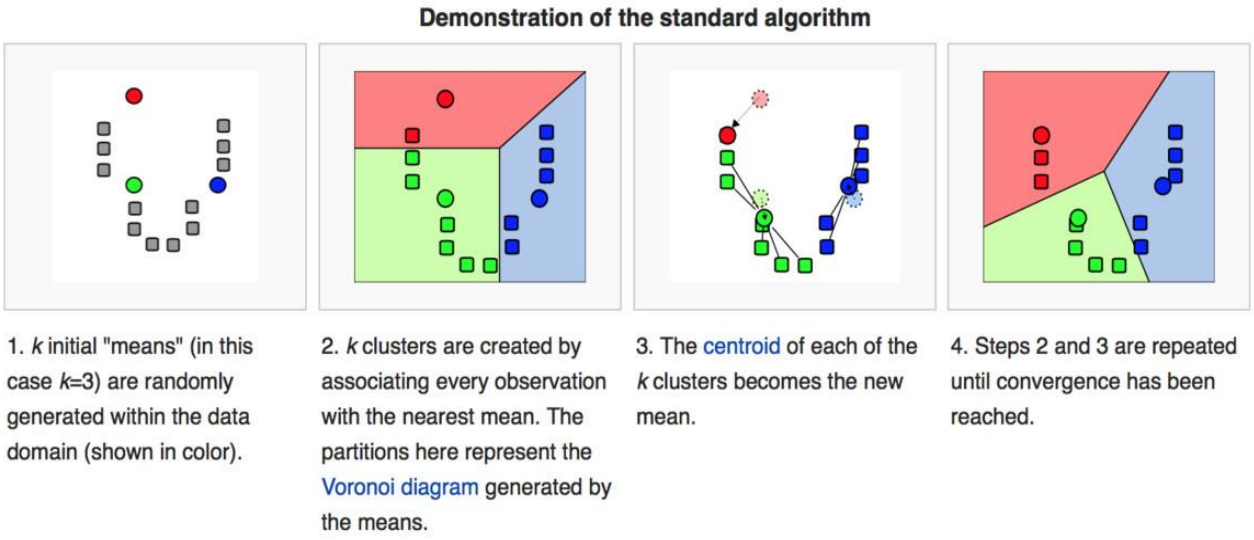
Thus, it shows that this algorithm separates the data into their best suited group based on the information it has. Data is separated into k clusters, and these are usually chosen to be far from each other spatially, (i.e. in Euclidean distance) so that a better result and separation can be obtained.

Each of the cluster has a centroid (a point at the center of the cluster, or the most representative point in the cluster) which can be randomly determined, and data point is clustered into a particular cluster based on the similarity of the data point features to the cluster.

K-means algorithm iteratively minimizes the distances between every data point and its centroid in order to find the most optimal solution for all the data points.

Finding the minimal distances between all the points implies that data points have been separated to form the most compact clusters possible, with the least variance within them. That is, no other iteration could have a lower average distance between the centroids and the data points found within them.

4.1.1. K-Means Algorithm



Steps in algorithm:

1. K random points of the dataset are chosen to be the centroids.
2. Distances between the centroids and each and every data point is calculated and stored.
3. Based on calculated distance, each point is assigned to the nearest/closest cluster

4. New cluster centroid positions are updated: similar to computing new mean in the point locations
5. If the centroid locations changed, the process repeats from step 2, until the calculated new center or the centroid stays the same, which indicates that the clusters' members and centroids are now set.

Advantages

- **Simplest unsupervised learning algorithm:** It follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed in advance.
- **Find pure sub clusters:** It can find pure sub clusters when large number of clusters are specified**.** A large *K* probably decreases the error, but it might increase the risk of overfitting.
- **Tends to produce compact clusters:** a compact cluster has a "ball-like" shape. A set of clusters are called well separated when any two points in a cluster are closer than the shortest distance between two points in different clusters.

Disadvantages

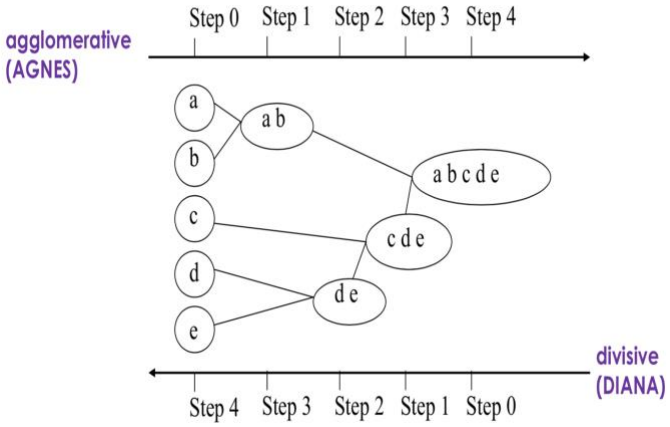
- **Sensitivity to initial configuration:** Since the k means algorithm follows local optimum solution, it adheres to the initial configuration set, till the end. Thus, we cannot obtain a globally optimum solution/partition.
- **Lack of Robustness (firmness) of the entire estimation:** Robust, here means how stable the structure stays with even small changes in data points or any other factors taken into consideration. Breakdown point here is 0 which indicates that if there is an error in placing the data point in one of the clusters, it might distort the estimate. Furthermore, this algorithm is not good with detecting outliers, so presence of outliers in cluster can change the centroid positions and the cluster shape significantly.
- **Only spherical clusters: This algorithm is suited for spherical clusters only.** By use of the Euclidean distance K-means treats the data space as isotropic (i.e. distances unchanged by translations and rotations). This means that data points in each cluster are modeled as lying within a sphere around the cluster centroid. Increasingly, as clusters are modeled only by the position of their centroids, K-means covertly assumes all clusters have the same radius.
- **Unknown number of clusters: There is no clear** boundary for separation, so it is difficult to identify the clusters.

4.2. Hierarchical Clustering

The idea is to build a binary tree of the data that successively merges similar groups of points. We divide the large clusters into small clusters or combine small clusters of large clusters respectively. The distance between each and every point in data set is calculated to evaluate minimum distance, maximum distance, average distance, and center distance. There are two classifications for Hierarchical Algorithm –

Agglomerative: It is bottom-up approach. The steps of this method are: Initially all the objects are clusters i.e. leaf. It recursively merges the nodes (clusters) that have the maximum similarity between them. Two points with minimum distance is taken and combined to form a single cluster. This process will be continued till all points are combined to form a single cluster. At the end of the process all the nodes belong to the same cluster i.e. known as the root of the tree structure.

Divisive: It is top-down approach. Start with assumption that all objects are grouped into single cluster (root). Which is then split into two recursively until each group consist of single object (leaf) on its own.



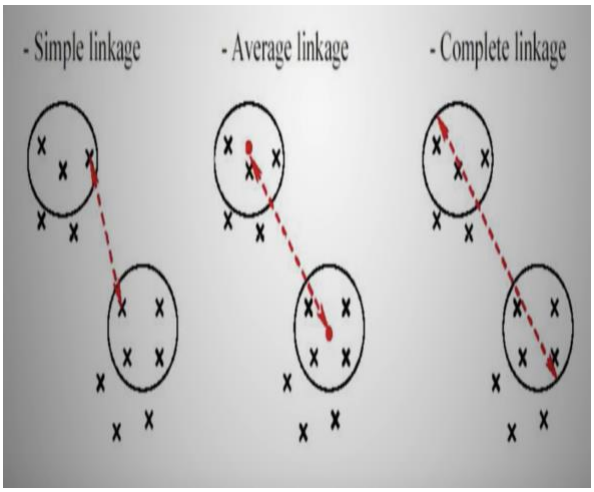
4.2.1. Agglomerative Algorithm

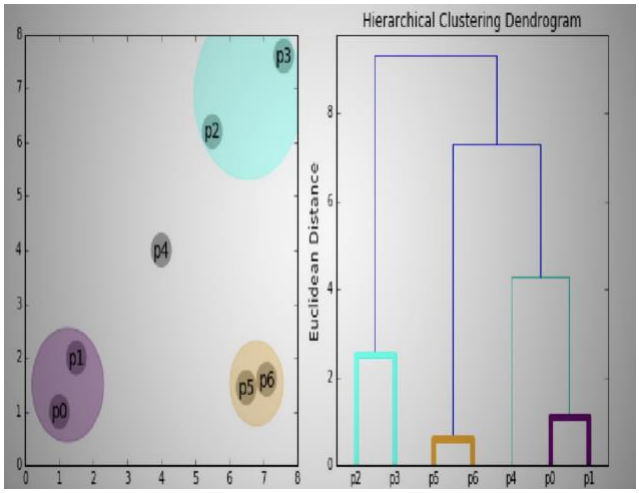
Agglomerative Clustering starts with each data point as its own cluster and find a best way to merge into a new one. Repeat the process until all clusters are fused. Dissimilarity between the data points is identified by knowing the distance of each data point to the other. General practice to identify the number of clusters is to use the midpoint of the longest branch.

There are different methods in respect to how they define proximity between any two clusters.

- *Simple Linkage* defines the distance between two clusters as the minimum distance between any single data point in the first cluster and any single data point in the second cluster. On the basis of this definition of distance between clusters, at each stage of the process we combine the two clusters with the smallest single linkage distance.

- *Average linkage* defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster. On the basis of this definition of distance between clusters, at each stage of the process we combine the two clusters that have the smallest average linkage distance. This method is less affected by outliers.
- *Complete linkage* defines the distance between two clusters to be the maximum distance between any single data point in the first cluster and any single data point in the second cluster. On the basis of this definition of distance between clusters, at each stage of the process we combine the two clusters that have the smallest complete linkage distance. Tries to minimize the diameter of the spherical cluster for every iteration.
- *Centroid method* defines the distance between two clusters as distance between the two mean vectors of the clusters. At each stage of the process we combine the two clusters that have the smallest centroid distance





- *Wards method:* Median or equilibrium centroid method.

  It will start out at the leaves and work its way to the trunk, so to speak. It looks for groups of leaves that form into branches, the branches into limbs and eventually into the trunk. Ward's method starts out with n clusters of size 1 and continues until all the observations are included into one cluster.

  This method is most appropriate for quantitative variables, and not binary variables.

  Based on the notion that clusters of multivariate observations should be approximately elliptical in shape, we assume that the data from each of the clusters have been realized in a multivariate distribution. Therefore, it would follow that they would fall into an elliptical shape when plotted in a p-dimensional scatter plot.

  Let Xijk denote the value for variable k in observation j belonging to cluster i. Furthermore, we define

Error Sum of Squares (ESS): We sum over all variables, and all of the units within each cluster. We compare individual observations for each variable against the cluster means for that variable. Note that when the Error Sum of Squares is small, it suggests that our data are close to their cluster means, implying that we have a cluster of like units.

$$\mathrm{ESS} = \sum_i \sum_j \sum_k |X_{ijk} - \bar{x}_{i \cdot k}|^2$$

Total Sum of Squares (TSS): The total sums of squares are defined the same as always. Here we compare the individual observations for each variable against the grand mean for that variable.

$$\mathrm{TSS} = \sum_i \sum_j \sum_k |X_{ijk} - \bar{x}_{\cdot\cdot k}|^2$$

R-Square:

$$r^2 = \frac{\text{TSS-ESS}}{\text{TSS}}$$

This r2 value is interpreted as the proportion of variation explained by a particular clustering of the observations.

Using Ward's Method, we start out with all sample units in *n* clusters of size 1 each. In the first step of the algorithm, *n* - 1 clusters are formed, one of size two and the remaining of size 1. The error sum of squares and r^2 values are then computed. The pair of sample units that yield the smallest error sum of squares, or equivalently, the largest r^2 value will form the first cluster. Then, in the second step of the algorithm, *n* - 2 clusters are formed from that *n* - 1 clusters defined in step 2. These may include two clusters of size 2, or a single cluster of size 3 including the two items clustered in step 1. Again, the value of r^2 is maximized. Thus, at each step of the algorithm, clusters or observations are combined in such a way as to minimize the results of error from the squares or alternatively maximize the r^2 value. The algorithm stops when all sample units are combined into a single large cluster of size *n*.

You need these linkage methods along with distance metrics to identify the similar data points and group based on how far/ near each clustering groups are. Finally, it forms the dendrogram pattern as in the figure.

Advantages

Hierarchical clustering does not require us to specify the number of clusters and we can even select which number of clusters looks best since we are building a tree. Additionally, the algorithm is not sensitive to the choice of distance metric (Euclidean distance, Manhattan distance, etc.); all of them tend to work equally well whereas with other clustering algorithms, the choice of distance metric is critical. A

particularly good use case of hierarchical clustering methods is when the underlying data has a hierarchical structure and you want to recover the hierarchy; other clustering algorithms can't do this. These advantages of hierarchical clustering come at the cost of lower efficiency, as it has a time complexity of $O(n^3)$, unlike the linear complexity of K-Means and GMM.

4.2.2. BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) Algorithm

BIRCH is an agglomerative hierarchical based clustering algorithm. It is an unsupervised algorithm particularly suitable for large datasets. The algorithm starts with single point clusters which means every point in dataset is considered as a cluster. Then it recursively groups the closest points into separate clusters, until only one cluster remains.

Birch algorithm uses two main concepts for the general cluster description: the clustering features (Clustering Feature, CF) and cluster feature tree (CF Tree).

Clustering Feature (CF)

Clustering Feature is a node in the BIRCH tree. It summarizes the information of underlying cluster of one or many points. The idea is to always consider the points that are close enough a group. Clustering Features provide this level of abstraction. Clustering Features are stored as a vector of three values:

$$CF = (N; LS; SS)$$

The linear sum (LS), the square sum (SS), and the number of points it encloses (N). All of these metrics can be calculated using only basic math:

$$\overrightarrow{LS} = \sum_{i=1}^{N} \overrightarrow{X_i}$$

$$SS = \sum_{i=1}^{N} \vec{X}_i^2$$

Any Clustering Feature in the tree can be calculated by adding its child Clustering Features which is called CF Additivity Theorem:

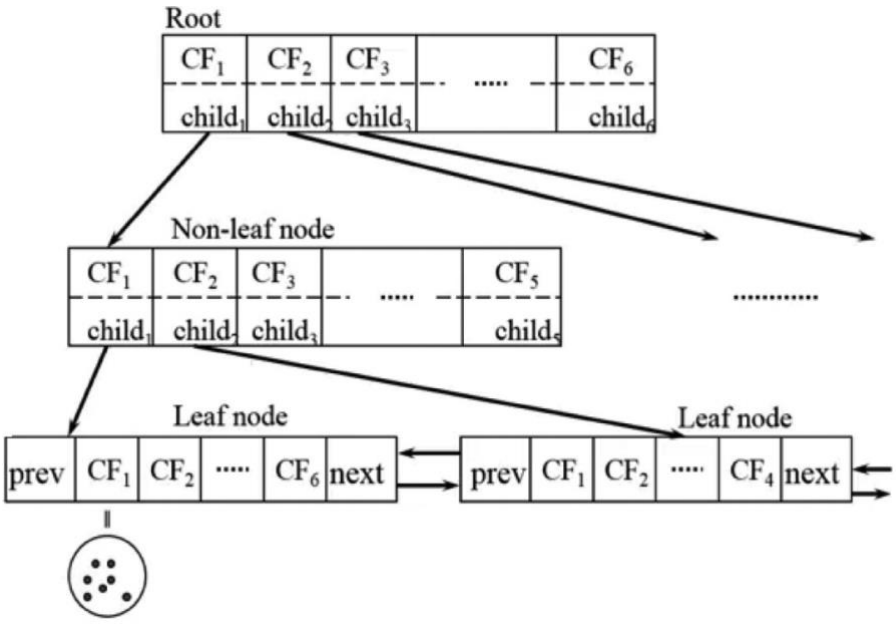
$$CF_1 + CF_2 = (N_1 + N_2, \overrightarrow{LS_1} + \overrightarrow{LS_{12}}, SS_1 + SS_2)$$

From the CF definition and additivity theorem, we know that the CF vectors of clusters can be stored and calculated and merged. A CF entry has sufficient information to calculate the centroid, radius, diameter and many other distance measures.

Cluster Feature Tree (CF Tree)

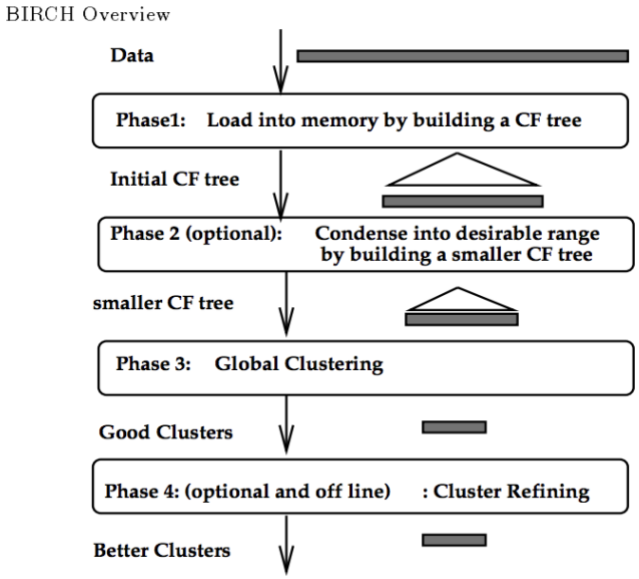
Clustering feature tree (CF Tree) is an alternative representation of data set. Clustering features are organized in a CF tree. It is will be built dynamically as new data objects are inserted

It is a height-balanced tree which is based on two parameters: branching factor and threshold. The branching factor is maximum number of children for non-Leaf node which is denoted as B and for Leaf node which is denoted as L. Threshold is the maximum diameter of sub-clusters stored at the leaf node. The tree size is a function of Threshold (T). The larger T is, the smaller the tree is.



BIRCH Overview

The figure below shows the overview of BIRCH. It consists of mainly four phases: (1) Loading, (2) Optional Condensing, (3) Global Clustering, and (4) Optional Refining.



The main task of Phase 1 is to scan all data and build an initial in-memory CF-tree using the given amount of memory and recycling space on disk. After Phase 1, subsequent computations in later phases will be:

(1) Fast because

(a) No I/O operations are needed, and

(b) The problem of clustering the original data is reduced to a smaller problem of clustering the sub clusters in the leaf entries;

(2) Accurate because

(a) Outliers can be eliminated, and

(b) The remaining data is described at the finest granularity that can be achieved given the available memory;

(3) Less order sensitive because the leaf entries of the initial tree form an input order containing better data locality compared with the arbitrary original data input order.

The phase 2 is optional where it rebuilds the CF-tree with a larger T. In phase 3, it uses a (selected) clustering algorithm to cluster the leaf nodes of the CF tree, which removes sparse clusters as outliers and groups dense clusters into larger ones. In phase 4, it does additional passes over the dataset & reassign data points to the closest centroid from phase 3.

Advantages

An advantage of BIRCH is its ability to incrementally and dynamically cluster incoming, multi-dimensional metric data points in an attempt to produce the best quality clustering for a given set of resources (memory and time constraints). In most cases, BIRCH only requires a single scan of the database as it scales linearly, which means a single scan helps in finding good clustering, and little more additional scans help in improving the quality. It is the first clustering algorithm proposed in the database area to handle 'noise' effectively. It makes full use of available memory to derive the finest possible sub-clusters while minimizing I/O costs. It is also an incremental method that does not require the whole data set in advance.

Disadvantages

It has relatively high in time complexity and proper parameter setting is important to BIRCH's efficiency. BIRCH may not work well when clusters are not spherical because it uses the concept

of radius or diameter to control the boundary of a cluster. In addition, it is order-sensitive and may generate different clusters for different orders of the same input data.

4.3. Model Based Clustering

In this category of algorithms, a fit between a mathematical model and the underlying data is optimized. It relies on the assumption that the data is a mixture of multiple probability distributions. Each component

or cluster is described by a density function and has a weight associated w.r.t. the mixture (Public, n.d.). Model we choose helps in defining our clusters and assignment of points to our cluster. Maximum likelihood is a common criterion for estimating model parameters. It gives us a framework to use our domain knowledge while offering more flexibility. Model selection criterions like Bayesian Information Criterion or integrated complete-data likelihood criterion are used to select number of mixture components and parameterization of component covariance matrices (Luca Scrucca, 2015). The variance between the clusters is maximized and within the cluster is minimized. Model based algorithms use two underlying approaches which are statistical and neural networks. The statistical approach can be used to find number of clusters and take outliers into account thus adding to its robustness. MCLUST, Expectation Maximization (EM), Self-organizing maps (SOMs) and conceptual clustering (COBWEB) are some commonly used and developed examples of model-based algorithms. In statistical approach, probabilistic measures are used to represent each derived concept (ADIL FAHAD, 2014). Neural network approach uses a set of connected input/output units where, connections have a weight associated with them (ADIL FAHAD, 2014).

Gaussian Mixture Models (GMM)

Before understanding what a gaussian mixture model is, a quick understanding of Gaussian distributions would be helpful. A gaussian distribution also known as a normal distribution is a distribution of a random variable taken over a large population. The values in a normal distribution are towards the center which means that the mean, median and mode are same. The central limit theorem is of importance here, it states that averages of samples of observations of random variables independently drawn from independent distributions converge to the normal, that is, they become normally distributed when the number of observations is sufficiently large (Wikipedia, n.d.). Thus, gaussian mixture models represent probabilistic representation of normally distributed subpopulations over a population. Points get assigned to subpopulations automatically, hence the technique is an unsupervised way of doing clustering. Unlike K-means which allows only spherical clusters and hard assignment to clusters, GMM allow probabilistic assignment with flexibility for elliptical or oblong clusters. GMM can be represented by parameters like mean vectors, covariance matrices and mixture weights from all component densities. A hint to using such a mixture model for data is when underlying data has multiple peaks (multimodal) in the distribution of data. Fitting it to a unimodal distribution would give a poor fit, so the data needs to be a mixture of multiple Gaussian distributions for the model to be applied effectively (John McGonagle, n.d.). GMM

parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori (MAP) estimation from a well-trained prior model (Reynolds). In case of GMM, when number of components K is known we can use the expectation maximization technique to estimate the mixture model's parameters.

4.3.1. Expectation Maximization (EM) Algorithm

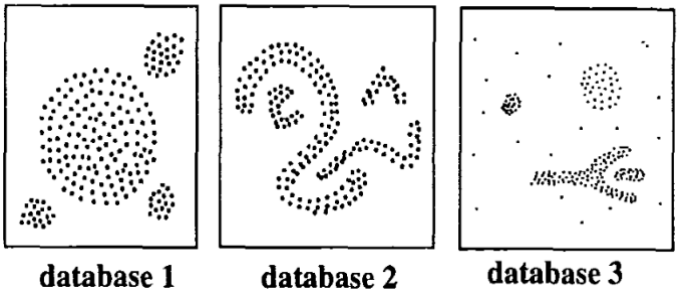
The expectation maximization algorithm is a numerical and iterative way to find maximum likelihood estimates for model parameters when the data is incomplete, missing or has some hidden parameters. The hidden parameters here are basically latent variables i.e. variables which are inferred from the data instead of being explicitly a part of the data. The maximum likelihood of data increases with every iteration till each reaches its own local maximum. It involves a process in which the expectation and maximization steps are iterated over till maxima is reached. The first step is initialization in which the values for model parameters μ, ϕ and σ are initialized usually in a random way or based on inputs from some other clustering results. Here for a model with k components μ represents the mean and σ represents the variance, while ϕ represents the component weights in the mixture. Samples from dataset are randomly assigned to mean estimates, while component variance estimates to sample variance based on sample mean (John McGonagle, n.d.). The second step is expectation in which the values of latent variables is calculated represented by Υ. In the maximization step, using the latent variables Υ the new values of parameters for the model are calculated. The model repeats expectation and maximization steps till all parameters converge to a certain decided value. In cases where k is not known a trade-off between fit for data and guessed number of components is optimized. The likelihood here is non-convex, so we're only guaranteed to reach a local maximum, hence using multiple initial values will give us a better coverage.

Limitations to GMM/EM

The most basic limitation to GMM is that the number of input components are required as an input to the model. Another limitation is the singularity issue, when we try to fit the model to a single point making the Gaussian a spike which collapses at that point. The variance in this case becomes zero which causes likelihood to be infinity making the model overfitted. Variance and mean adjustment are one way of resolving this issue, using MAP instead of MLE is another. Usually when a model is applied to more than one points, we can avoid the issue completely.

4.4. Density Based Clustering Algorithm

Density Based Algorithms locates regions of high density that are separated from one another by regions of low density.



Most of density algorithms use two main parameters:

Eps: Maximum radius of the neighborhood for a point

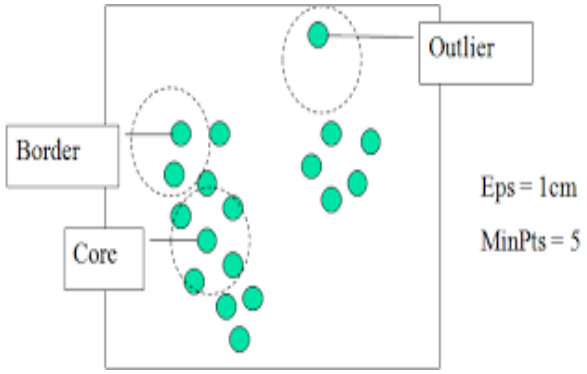
MinPts: Minimum number of points in an Eps neighborhood of a point

Based on Eps and MinPts objects are categorized into three exclusive groups:

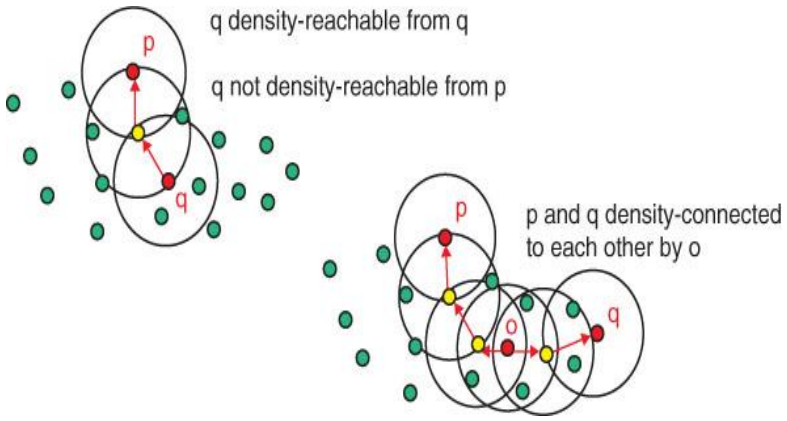
Core Points: A point is *a core point* if it has more than a specified number of points (MinPts) within Eps distance. These are points that are at the interior of a cluster. All the core points are connected based on the densities to form cluster.

Border points: A *border point* has fewer than MinPts within Eps distance but is in the neighborhood of a core point.

Noise points: A *noise point* is any point that is not a core point nor a border point.



Few key terms used in Density Based Clustering which are explained below referring to the diagram –



Directly density-reachable:

A point p is directly density-reachable from a point q w.r.t. Eps, MinPts if:

1. p ∈ NEps(q) and
2. |NEps(q)| ≥ MinPts

Density-Reachable:

A point p is density-reachable from a point q w.r.t. Eps, MinPts if there is a chain of points p1, ..., pn, with p1 = q, pn = p, such that, pi+1 is directly density reachable from pi

Density-Connected:

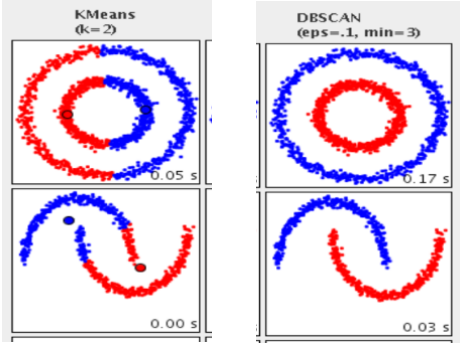
A point p is density-connected from a point q w.r.t. Eps, MinPts if there is a point o. Such that, p and q are density-reachable from o w.r.t. Eps and MinPts

Two points fall in same cluster if they are directly density-reachable or density reachable or density-connected. Examples: DBSCAN, OPTICS, DBCLASD, GDBSCAN, DENCLU and SUBCLU. DBSCAN is a base algorithm for density-based clustering.

4.4.1. DBSCAN Algorithm

Steps in algorithm: For DBSCAN, the parameters ε(Eps) and minPts are needed.

1. Find the points in the ε (Eps) neighborhood of every point and identify the core points with more than minPts neighbors.
2. Find the connected components of core points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an ε (Eps) neighbor, otherwise assign it to noise.



Parameter Estimation

MinPts: As a rule of thumb, a minimum minPts can be derived from the number of dimensions D in the data set, as minPts ≥ D + 1. The low value of minPts = 1 does not make sense, as then every point on its own will already be a cluster. With minPts ≤ 2, the result will be the same as of hierarchical clustering with the single link metric, with the dendrogram cut at height ε. Therefore, minPts must be chosen at least 3. However, larger values are usually better for data sets with noise and will yield more significant clusters. As a rule of thumb, minPts = 2·dim can be used, but it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates.

E(Eps): The value for ε can then be chosen by using a k-distance graph, plotting the distance to the k = minPts-1 nearest neighbor ordered from the largest to the smallest value. If ε is chosen much too small, a large part of the data will not be clustered; whereas for a too high value of ε, clusters will merge, and the majority of objects will be in the same cluster. In general, small values of ε are preferable, and as a rule of thumb only a small fraction of points should be within this distance of each other.

Distance function: The choice of distance function is tightly coupled to the choice of ε(Eps) and has a major impact on the results. In general, it will be necessary to first identify a reasonable measure of similarity for the data set, before the parameter ε can be chosen. There is no estimation for this parameter, but the distance functions need to be chosen appropriately for the data set. For example, on geographic data, the great-circle distance is often a good choice.

Advantages

- DBSCAN algorithm does not require one to specify the number of clusters.
- DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.
- It can detect noise and is robust to outliers.
- DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database.

Disadvantages

- It is sensitive to parameters eps and minPts and it is difficult to determine the correct set of parameters for a particular dataset.
- It is not suitable for high dimensional as mostly distance measure used is Euclidean distance. As for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse

of dimensionality", making it difficult to find an appropriate value for ε. This effect, however, is also present in any other algorithm based on Euclidean distance.

- DBSCAN is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data are processed. For most data sets and domains, this situation does not arise often and has little impact on the clustering result: both on core points and noise points, DBSCAN is deterministic.
- DBSCAN is not suitable for dataset with varying densities, since the minPts and Eps combination cannot then be chosen appropriately for all clusters.
- If the data and scale are not well understood, choosing a meaningful distance threshold Eps can be difficult.

5. DATASET DESCRIPTION

To begin with we used numerical data for our analysis and our choice of algorithms was based on the type of our data. For standardization purposed we used datasets which were already benchmarked for clustering. We used the data set provided by P. Fränti and S. Sierano. The first dataset that we used is the DIM-set(low), which is a synthetic dataset with Gaussian clusters and has 9 components and a 2-15 range of dimensions. The second dataset we used is the S1 dataset from S-Sets available from the same source. This one too has 15 Gaussian clusters with different degree of overlapping.

6. EVALUATION METRICS

We used the following four metrics for analyzing clustering as the first 3 metrics can be used to evaluate the model when the ground truth labels are not known, and they tell us how dense and well separated clusters are generated using any clustering algorithm:

1. *Silhouette Score:* Tells about cluster belonging
2. *Calinski Harabaz Score:* Tells about cluster definition
3. *Davies Bouldin Score:* Tells about cluster separation
4. *Time:* Execution time of a clustering algorithm

6.1. Silhouette Score

The Silhouette score reflects how similar a point is to the cluster it is associated with/, that is, for each point it computes the average distance of the point from the points in the nearest cluster minus

the average distance of the point from the points in its own cluster divided by the maximum between those distances. The overall score is the average of the score per point.

The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster. Score closer to 1 indicates better clustering algorithm for a given dataset.

Calculating Silhouette Score:

The Silhouette Coefficient is defined for each sample and is composed of two scores:

a: The mean distance between a sample and all other points in the same class.

b: The mean distance between a sample and all other points in the *next nearest cluster*.

$$s = \frac{b - a}{max(a, b)}$$

6.2. Calinkski Harabaz Score

- Ratio between the within-cluster dispersion and the between-cluster dispersion.
- If the ground truth labels are not known, the Calinski-Harabaz score- also known as the Variance Ratio Criterion - can be used to evaluate the model, where a higher Calinski-Harabaz score relates to a model with better defined clusters.
- The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.
- The score is fast to compute.
- The Calinski-Harabaz score is given as the ratio of the between-clusters dispersion mean and the within-cluster dispersion.
- Higher scores indicate better clustering algorithm for a given dataset.

Calculating Calinkski Harabaz Score:

For k clusters, the Calinski-Harabaz score s is given as the ratio of the between-clusters dispersion mean and the within-cluster dispersion:

$$s(k) = \frac{\mathrm{Tr}(B_k)}{\mathrm{Tr}(W_k)} \times \frac{N - k}{k - 1}$$

where B_K is the between group dispersion matrix and W_K is the within-cluster dispersion matrix,

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q} n_q (c_q - c)(c_q - c)^T$$

with N be the number of points in our data, Cq be the set of points in cluster q, cq be the center of cluster q, c be the center of E, nq be the number of points in cluster q.

6.3. Davies Bouldin Score

- The index is defined as the average similarity between each cluster (Ci for i=1,...,k) and its most similar one (Cj). In the context of this index, similarity is defined as a measure that trades off : the average distance between each point of cluster i and the centroid of that cluster – also know as cluster diameter and the distance between cluster centroids.
- The computation of Davies-Bouldin is simpler than that of Silhouette scores.
- The usage of centroid distance limits the distance metric to Euclidean space.
- Lower scores indicate better clustering algorithm for a given dataset.

Calculating Davies Bouldin Score:

The index is defined as the average similarity between each cluster Ci for i=1,..., k and its most similar one C_j. In the context of this index, similarity is defined as a measure R_{ij} that trades off:

s_i - average distance between each point of cluster i and the centroid of that cluster (cluster diameter).

d_{ij} - distance between cluster centroids i and j.

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

Then the Davies-Bouldin index is defined as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

Zero is the lowest possible score. Values closer to zero indicate a better partition.

6.4. Time

Besides these three metrics we have also considered time required for execution by a clustering algorithm for a particular dataset to evaluate the performance of that algorithm in comparison with other clustering algorithms.

7. IMPLEMENTATION

We used the chosen algorithms and did test runs on the data mentioned above. For our test runs we made use of Jupyter notebooks in which we used python codes for actual implementation. We used source codes available as a part of python's scikit learn package for clustering for our specific algorithms (Scikit, n.d.). To measure the clustering quality, we compared all the algorithms on our chosen metrics. For getting the effect of changing data sizes and dimensions we used our multidimensional data and divided into two sizes. We then checked performance at different data sizes and different dimensions for the same. To study the effect of changing cluster sizes we used dataset S1 and measures various metrics for it by changing number of components or clusters in every run. We have articulated the outcomes of these runs in our observations section.

8. OBSERVATIONS

For multi-dimensional (DIM) dataset - Here are the results from our runs,

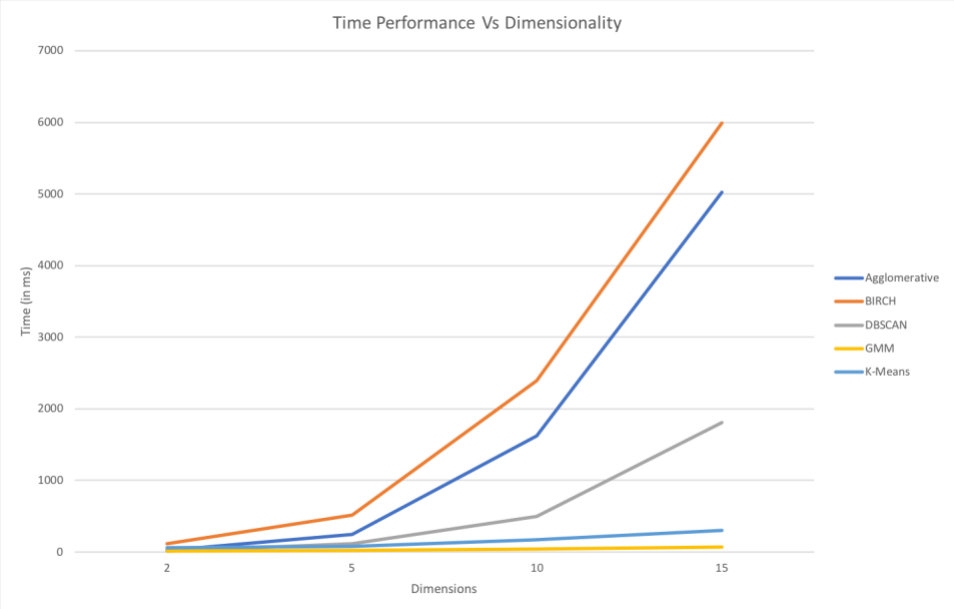
Algorithms		Full Data (n=9)				Half Data (n=5)			
	Dimensions	2	5	10	15	2	5	10	15
	Dataset length	1351	3376	6751	10126	675	1688	3375	5063
K-means Algorithm	Silhouette Coefficient	0.95	0.92	0.92	0.92	0.95	0.94	0.92	0.72
	Calinski-Harabaz Index	126599.70	137552.50	278319.70	302436.40	49626.98	103364.18	157444.82	10145.59
	Davies-Bouldin Index	0.07	0.11	0.11	0.12	0.06	0.08	0.12	0.58
	Time Performance (ms)	59	80	175	299	28	41	94	149
BIRCH	Silhouette Coefficient	0.95	0.92	0.92	0.92	0.95	0.94	0.92	0.72
	Calinski-Harabaz Index	126599.75	137552.46	278319.72	302436.37	49626.99	103364.18	157444.82	10145.60
	Davies-Bouldin Index	0.07	0.11	0.11	0.12	0.06	0.08	0.12	0.58
	Time Performance (ms)	118	515	2390	5990	46	166	528	2530
Agglomerative Clustering	Silhouette Coefficient	0.95	0.92	0.92	0.92	0.95	0.94	0.92	0.72
	Calinski-Harabaz Index	126599.75	137552.46	278319.71	302436.36	49626.98	103364.18	157444.82	10145.59
	Davies-Bouldin Index	0.07	0.11	0.11	0.12	0.06	0.08	0.11	0.58
	Time Performance (ms)	20	250	1620	5027	10	75	469	2076
Gaussian Mixture Models	Silhouette Coefficient	0.95	0.92	0.92	0.92	0.95	0.94	0.92	0.72
	Calinski-Harabaz Index	126649.83	137537.36	222516.92	262894.62	49471.60	103158.00	104448.00	10077.20
	Davies-Bouldin Index	0.07	0.11	0.11	0.12	0.06	0.08	0.12	0.58
	Time Performance (ms)	15	24	42	67	6	11	21	33
DBSCAN	Silhouette Coefficient	0.95	0.92	-0.49	0.18	0.95	0.93	-0.18	0.12
	Calinski-Harabaz Index	269419.65	63224.01	224.35	1650.13	147299.21	43193.00	383.58	871.60
	Davies-Bouldin Index	0.06	1.16	1.14	1.21	0.05	1.15	1.12	1.24
	Time Performance (ms)	19	114	497	1807	9	50	227	822

Following are the trends that are observed from the above table,

	K-Means		Birch		Agglomerative		Gaussian Mixture		DBSCAN	
	Increasing Data	Increasing Dims	Increasing Data	Increasing Dims	Increasing Data	Increasing Dims	Increasing Data	Increasing Dims	Increasing Data	Increasing Dims
Silhouette -Score	Marginal difference	Worsens	Marginal difference	Worsens	Marginal difference	Worsens	Marginal difference	Worsens	Random	Random
Calinski-Harabaz Score	Improves	Improves	Improves	Improves	Improves	Improves	Improves	Improves	Improves	Worsens
Davies-Bouldin Score	Improves	Worsens	Random	Worsens	Random	Worsens	Worsens	Worsens	Worsens	Worsens drastically
Time	2x	~2x	2-5x	2-5x	2-4x	4-10x	2x	~2x	2x	3-5x

Key Observations

- The silhouette score for all algorithms has a marginal difference when it comes to changing data sizes, which means cluster belonging isn't affected much. But with increasing dimensions the cluster belonging spoils as an effect.
- Calinski-Harabaz score increases with increasing data and dimensions, which shows that with a greater number of points we get a better cluster definition.
- Davies-Bouldin score usually worsens with increasing data and increasing dimensions, showing degrading cluster separation.
- The time performance for GMM and K-Means is best due to their linear complexity and worst for Agglomerative and BIRCH due to their quadratic complexity

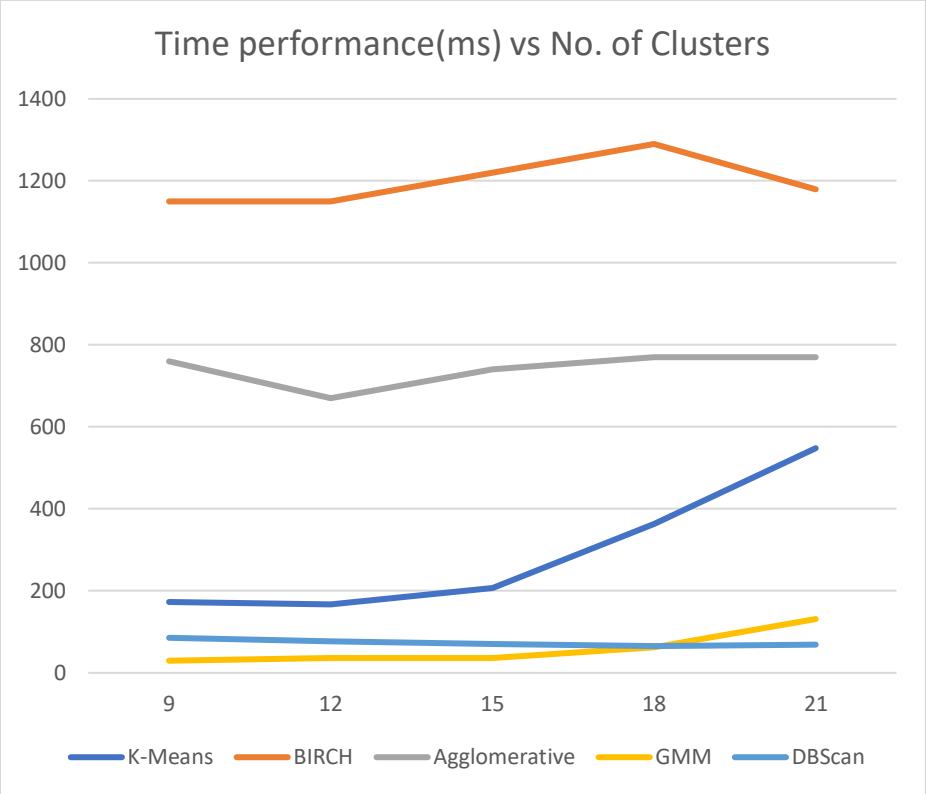


For S1 dataset, here are the results from our runs -

Algorithm	K-Means			
Cluster size	Silhouette Coefficient	Calinski-Harabaz Index	Davies-Bouldin Index	Time Performance (ms)
9	0.582	8277.11	0.59616	173
12	0.631	10453.20	0.49524	167
15	0.711	22679.70	0.36652	207
18	0.637	20255.79	0.60309	363
21	0.549	18646.98	0.84813	548
Algorithm	BIRCH			
Cluster size	Silhouette Coefficient	Calinski-Harabaz Index	Davies-Bouldin Index	Time Performance (ms)
9	0.583	8216.20	0.603322095	1150
12	0.636	10741.59	0.49519	1150
15	0.709	22330.52	0.36589	1220
18	0.634	20056.93	0.57986	1290
21	0.556	18384.64	0.78718	1180
Algorithm	Agglomerative Clustering			
Cluster size	Silhouette Coefficient	Calinski-Harabaz Index	Davies-Bouldin Index	Time Performance (ms)
9	0.583	8216.20	0.603	760
12	0.636	10741.59	0.495	670
15	0.709	22330.52	0.366	740
18	0.634	20056.93	0.58	770
21	0.556	18384.64	0.787	770
Algorithm	Gaussian Mixture Model			
Cluster size	Silhouette Coefficient	Calinski-Harabaz Index	Davies-Bouldin Index	Time Performance (ms)
9	0.579	8012.76	0.608	30
12	0.641	10014.29	0.473	36
15	0.711	22612.36	0.366	36
18	0.632	19599.44	0.610	63
21	0.555	16124.06	4.964	131
Algorithm	DBSCAN			
Cluster size	Silhouette Coefficient	Calinski-Harabaz Index	Davies-Bouldin Index	Time Performance (ms)
9	0.431	2842.71	1.644	86
12	0.474	3888.13	1.596	77
15	0.629	5849.44	1.442	71
18	0.614	5138.77	1.441	65
21	0.544	4136.91	1.351	69

Key Observations

- The algorithms perform similarly against each other in terms of metrics, only time performance of GMM is better than K-Means and DBScan.
- Metrics value for all algorithms are best at cluster size 15 which corresponds to the ideal value in the benchmarks



9. CONCLUSION

For the bench mark dataset that we have used the best results were obtained for Gaussian Mixture Model (GMM). The possible reasons for GMM doing better would be its advantages with numerical data and working well with missing parameters and data. Also, it has a linear time complexity as compared to other models.

While developing a new algorithm, we would suggest the users can take advantage of the well-developed mathematical models to work on underlying numeric data. There are various models available and the best fit can be chosen, to leverage our existing domain knowledge and apply it to machine learning. In some cases where we don't know number of components and where the data follows a hierarchical structure, we can make use of agglomerative based clustering models.

In general, there is not one thing fits all solution in clustering as each model depends on underlying data size, type and other factors associated with it while clustering. So, we need to understand when to apply which algorithm based on its features and strengths

10. FUTURE SCOPE

We would like to explore more on Entropy based Expectation Maximization / K-means Clustering - a density based model. Perform different implementation of these algorithms that we have so far used. Also, performance tuning the parameters of these algorithms and implement them on various datasets as well.

11. REFERENCES

https://ieeexplore.ieee.org/document/1427769

https://www.cc.gatech.edu/~isbell/reading/papers/berkhin02survey.pdf

https://searchdomino.techtarget.com/definition/application-clustering

https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

http://www.cs.uvm.edu/~xwu/kdd/BIRCH.pdf

https://www.cs.sfu.ca/CourseCentral/459/han/papers/zhang96.pdf

https://www.cse.buffalo.edu/faculty/azhang/cse601/Birch.ppt

A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis - ADIL FAHAD

A Study of Hierarchical Clustering Algorithm Yogita Rani1 and Dr. Harish Rohil

https://www.computerhope.com/jargon/l/local-optimum.htm

https://brilliant.org/wiki/k-means-clustering/

https://www.researchgate.net/figure/An-Example-of-compact-well-separated-clusters_fig1_242269657

https://scialert.net/fulltextmobile/?doi=itj.2011.478.484

Image: https://www.datascience.com/blog/k-means-alternatives

https://www.saedsayad.com/clustering_kmeans.htm

https://blog.easysol.net/machine-learning-algorithms-3/

http://users.jyu.fi/~samiayr/pdf/introtoclustering_report.pdf

John McGonagle, G. P. (n.d.). Gaussian Mixture Model. Retrieved from Brilliant: https://brilliant.org/wiki/gaussian-mixture-model/

Luca Scrucca, A. E. (2015). Improved initialisation of model-based clustering using Gaussian hierarchical partitions. Adv Data Anal Classif .

P. Fränti, S. S. (n.d.). Clustering Benchmark Datasets. Retrieved from http://cs.joensuu.fi/sipu/datasets/

Public. (n.d.). Retrieved from http://www.public.iastate.edu/~maitra/stat501/lectures/ModelBasedClustering.pdf

Reynolds, D. (n.d.). Gaussian Mixture Models.

Scikit. (n.d.). Scikit Learn. Retrieved from https://scikit-learn.org/stable/

Wikipedia. (n.d.). Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Normal_distribution