Analyzing clustering performance using visualization
MSc thesis

Ashfaq Afzal

Master's thesis

UNIVERSITY OF EASTERN FINLAND

School of Computing

Computer Science

July 2021

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu
School of Computing
Computer Science

Ashfaq Afzal: Analyzing clustering performance using visualization
Master’s Thesis, 59 p.

Supervisors of the Master’s Thesis: Professor Pasi Fränti and Dr. Radu Mariescu-Istodor

July 2021

Abstract: Visualizations have been proved to be more effective in communicating ideas for a very long time. They can help us understand something easily and effectively. In this study, we try to learn about the performance of clustering algorithms through visualization. Clustering is a machine learning technique that groups data having similar features together and vice versa. K-means is a very popular clustering algorithm and here we try to learn about the performance of k-means and few other clustering algorithms that use k-means as an extension or are a variant of k-means. Different visualization techniques, their advantages, drawbacks, and the analysis of these algorithms have been discussed and documented.

Keywords: Clustering algorithms, K-means, Random Swap, Gradual k-means, Weighted k-means, Convex hull.

Acknowledgments

First and foremost, I would like to extend my sincere gratitude towards the University of Eastern Finland and its faculty members. I would like to thank my supervisors Prof. Pasi Fränti and Dr. Radu Mariescu-Istodor for guiding me throughout my thesis and motivating me to produce satisfactory results. It has been a great pleasure for me to have them as my instructors. I am thankful to them for all the guidance that they provided me during my studies not just thesis but also in other courses and career-related matters.

I express my humblest gratitude to my friends and family for providing me with their support and motivation that helped me complete these studies. This accomplishment would not have been possible without them.

Last but not the least, I would like to thank my parents and remember my grandparents because of whose blessings I believe, I got the opportunity to study in Finland.

May God shower the above-cited personalities with success and honor in their life.

List of abbreviations

SSE	Sum of Squared Error
BIC	Bayesian information criterion
GMM	Gaussian Mixture models
MSE	Mean squared error

Contents

Contents

Table of Figures

Table of Tables

1 Introduction

Clustering is a grouping of objects so that a similar set of objects are in one group than those in the other groups. Clustering can be achieved by various algorithms that have some differences in understanding what constitutes a cluster and how the clusters are found. It is challenging for a human to categorize and process large-scale data into separate categories.

Selecting the best method to measure the distance between objects is important for the clustering results. In clustering software, *Euclidean distance* is the default distance measure method. In Euclidean distance, objects that have low values are clustered together.

When we have a large set of data, like thousands and millions of data points, we need tools and techniques to visualize the data. There are lots of tools available for visualization. Today, the best-used tools for data visualization are *Google charts*, *Grafana*, *Chartist.js*, *Fusion charts*, *Infogram*, *D3.js*, *Chart blocks*, and *Fusion charts*. The graphical representation of data is called *data visualization*. Some other commonly used tools are *flash player*, *Seaborn*, *Matlab*, *matplotlib.py*. In this thesis, HTML5 with JavaScript is used to visualize the clustering results. Visualizing data is an important step to draw conclusions from large data sets.

Flash player was used for animations, games, and visualizing algorithms in the past. Same goes for Java applets. Nowadays HTML improved and does not need (potentially dangerous) external plugins etc. When doing data visualization, the process of creation of visualization is usually wanted automated. The goal of this thesis is to study different ways of visualization to study the performance of several clustering algorithms.

1.1 Objective

In this thesis, we will discuss some visualization techniques by using a clustering algorithm that helps to find similar objects by splitting the observation into some subsets. When objects are split, similar objects go to the same group, and different objects are in different groups. We focus on the visualization of some clustering algorithms to see how they work and how we can learn more about the behavior of those algorithms on different datasets. We also try to interact with those visualizations to check that where (in which kind of dataset) a specific algorithm lacks, and others perform well.

1.2 Motivation

Clustering has a lot of importance in data mining and data analysis applications. The set of objects is the primary clustering task where similar objects are going into the same group. Identifying the distinct groups in customer base clustering has a significant role in organizations. In many applications, clustering is mesmerizing, like pattern recognition, market research, image processing, and data analysis. Using clustering in data visualization, one can get a clearer idea about information means by giving it visual context through graphs or maps.

Moreover, Clustering algorithms are NP-hard because clustering problems are polynomial-time reducible even though they may not be NP itself (Vattani 2009). and exact algorithms are not useful. Heuristic algorithms are efficient, but their behavior is hard to understand at times. Visualizers/animators help scientists to better understand algorithm behavior in real instances.

2 Clustering Problem

Clustering provides an abstraction from individual data objects to clusters in which those data objects reside. Clustering aims to find usefulness defined by the goals of data analysis. Clustering is a fundamental exploratory analysis used in many scientific domains. Various clustering algorithms with various cost functions produce solutions, and there is no single best algorithm for all possible data sets. A primary task is to find the best possible clustering for the given data set. Other challenges in clustering are:

1. Determining the number of clusters in the dataset.
2. Detecting outliers as they may disturb the clustering process

Most papers focus on convex clustering techniques. Some Bayesian approaches like Dirichlet Process Mixture Models help by directly estimating the number of clusters. This approach is by no means a panacea. Other hyperparameters still tuned these models, and these models do not always estimate the correct number of clusters accurately (Cheng et al 2018).

According to David Robinson (Robinson, D. J. et al 2013) the shortcoming of the most famous clustering algorithm, k-means, is that even when the human eye can easily separate data into groups, the k-means algorithm can still fail (Robinson 2013). It aligns with that analysis, but it is entirely reasonable from an algorithm perspective (Zhao, Q. 2012).

According to (Zhao & Fränti 2014), finding the correct number of the clusters, clustering algorithm, and cluster validity indexes based on the sum of squares, works very well for Gaussian type data. Most of them cannot provide a global maximum or minimum point for the correct cluster number. WB index works slightly better than the other indices. It provides a more stable result than the min-max function (Zhao & Fränti 2014) This is the standard way to find cluster variations but not enough to determine the value of *k* reliably in all cases.

Another method is finding knee (or elbow) point is SSE curve for various k-values. *Bayesian information criterion* (BIC) was reformulated in partitioning-based clustering showing the perspective of finding several clusters (Zhao et al 2008). This method decides the local maximum but not always accurately. To improve BIC, it still needs functions that provide min (or max) at this point. When the data set becomes more arduous, it becomes more complex, so the knee point is just an intuitive idea, and one needs a concrete function to implement it. Interestingly, BIC was initially designed for Gaussian Mixture models (GMM) within the EM algorithm. By imposing multivariate Gaussian assumptions on observations, a closed form of BIC is provided (Teklehaymanot et al 2018)

GMM itself is worth mentioning in the cost function part. While k-means uses *sum-of-squared errors* (SSE) it has only centroid and binary (hard) partition, *Gaussian mixture model* (GMM) has also the variance of each cluster. Either a so-called diagonal matrix so that one variance for each attribute (commonly used) or a full matrix so that co-variances of individual attributes are also modeled (rarely used). While it is more complex than SSE, it can potentially find clusters of elliptical shape and invariant to their size. SSE tends to make all clusters of similar size: tiny dense points will make one cluster, but a sizeable wide cluster may be split into several smaller.

2.1 Distance Function

In our work, the centroid is used as the prototype because it is the most common choice for this. These centroids are calculated as the average of the objects in the cluster using Euclidean distance. The choice of distance function is an important step. To measure the distance, there are some classical methods like *Euclidean Distance* and *Manhattan distance*. Euclidean distance is measured using the formula:

$$\boldsymbol{d_{euc}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}} \quad \textbf{(1)}$$

Manhattan Distance is measured using the formula:

$$\boldsymbol{d_{man}(x,y) = \sum_{i=1}^{n} |(x_i - y_i)|} \tag{2}$$

Some correlation-based distance measures also exist to measure gene expression data analysis. Pearson Correlation distance measure is one example:

$$\boldsymbol{d_{cor}(x,y) = 1 - \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 (y_i - \bar{y})^2}}} \tag{3}$$

Eisen Cosine correlation distance measured is another:

$$\boldsymbol{d_{eisen}(x,y) = 1 - \frac{|\sum_{i=1}^{n} x_i y_i|}{\sqrt{\sum_{i=1}^{n} x_i{}^2 \sum_{i=1}^{n} y_i{}^2}}} \tag{4}$$

Spearman Correlation distance measure computes the correlation of centroid c and y as:

$$\boldsymbol{d_{spear}(x,y) = 1 - \frac{\sum_{i=1}^{n}(x'_i - \bar{x}')(y'_i - \bar{y}')}{\sqrt{\sum_{i=1}^{n}(x'_i - \bar{x}')^2 (y'_i - \bar{y}')^2}}} \tag{5}$$

Kendall correlation distance is non-parametric perform rand-based correlation analysis measured as:

$$\boldsymbol{d_{kend}(x,y) = 1 - \frac{n_c - n_d}{\frac{1}{2} n(n-1)}} \tag{6}$$

Select the best distance measure is important. In many clustering software, Euclidean distance is the default distance measure method. In Euclidean distance, objects with low values should be clustered together.

2.2 Objective Function

Clustering goals are typically expressed by an objective function that depends on points to one another and the cluster centroids. For example, to minimize the sum of the squared distance of each point of the cluster to its closest cluster centroid. By using some examples, we use this goal. The critical point in this study is that when we have specified distance measure and objective function, then centroid can often be determined mathematically. The primary objective behind all this is to improve the performance of the algorithm. The objective function is to minimize the square distance of every point to its nearest centroid.

2.3 Data Processing

Some issues appear while gathering the data. The first issue is setting boundaries for objects. Some researchers clearly delineate the focus of their study and include objects within the boundary. The second issue is identifying the objects representing that domain which is most critical because if the domain is incomplete then a further solution can be distorted, and it generates a partial picture of the cluster in visualization (Roskos-Ewoldsen 2008). The following procedure is followed for the functionality of the clustering visualizer (or clustering animator).

The first step in visualization is to show the dataset on the screen (denoted as *canvas* in the following) so we start by loading the dataset. Dataset is loaded and normalized (between 0-1 values) to unify all computations. Then to show the data points on the canvas, values are rescaled to the size of the canvas and then plotted on the canvas. Rescaling the data set according to the canvas size is required; otherwise, our plot might have the size of one pixel and be practically invisible. Rescaling the data set is necessary also for the algorithms because it helps to calculate the distances among the data. If scaling of data sets is not done, then the features with higher value ranges start to dominate when we calculate distance. The scaling of data points makes a difference between a better and a weaker algorithm.

2.3.1 Normalization and standardization

The most commonly used technique for scaling is normalization and standardization as in figure 1. The process of *normalization* is to bound the data values between two numbers, minimum and maximum (Roskos-Ewoldsen 2008). The *standardization* process transforms that data to have zero mean and variance of 1., as shown in Figure 1.

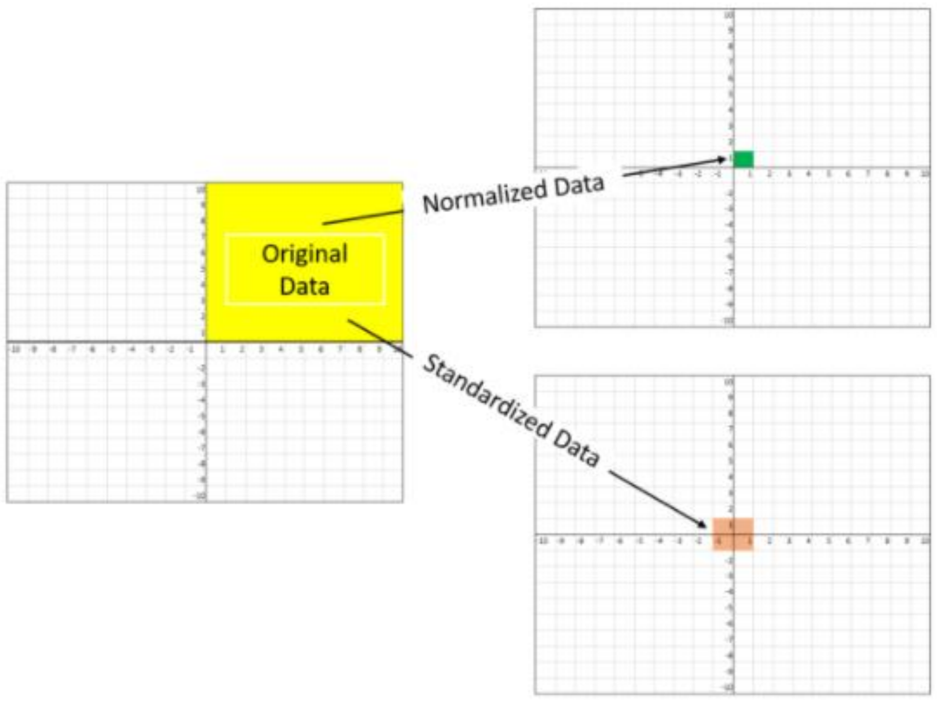


Figure 1. Data points scaling (Roskos-Ewoldsen, 2006)

Data from multiple features can be transformed through normalization, so normalization is well suited for processing data distribution known as Gaussian distribution (Murtagh 2017). Normalization needs less data for calculation compared to Quantiles. The zero scores of normalized data are:

$$x' = (x - \mu)/\sigma$$
$$\mu = \text{mean}$$
$$\sigma = \text{standard deviation}$$

We can see the difference between normalized and raw data in Figure 2.

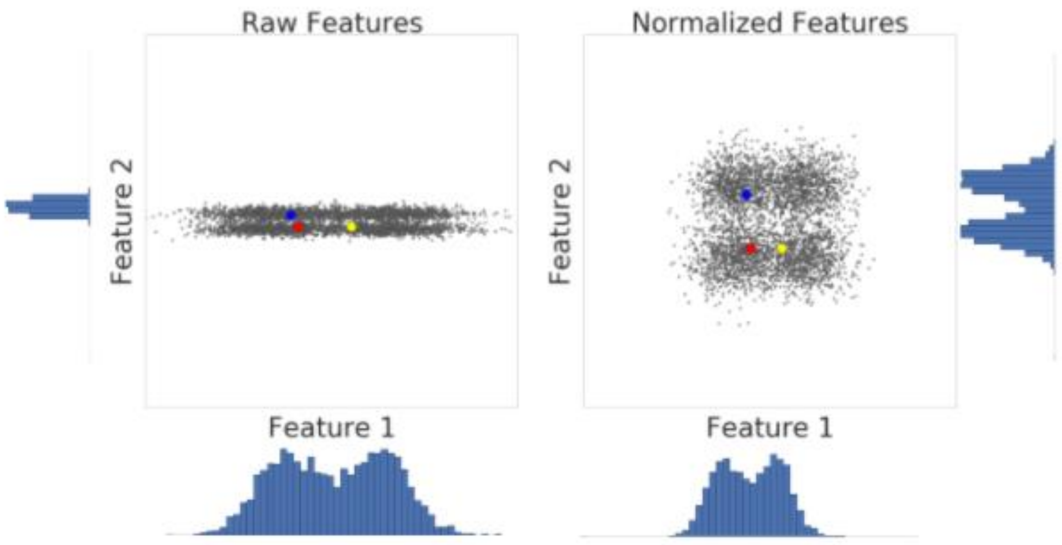


Figure 2. Data before and after normalization

2.3.2 Log Transform

In *Log Transform*, the data set conforms to power distribution by clumping the data at the lower end. In Figure 3, the red color is closer to yellow. However, when the power-law distribution is performed by log transform, it creates smoother distribution, and the red color becomes closer to the blue color (Shah 2017).

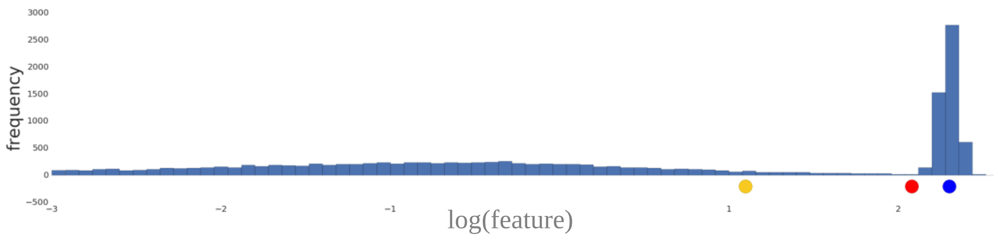


Figure 3. Power-law distribution

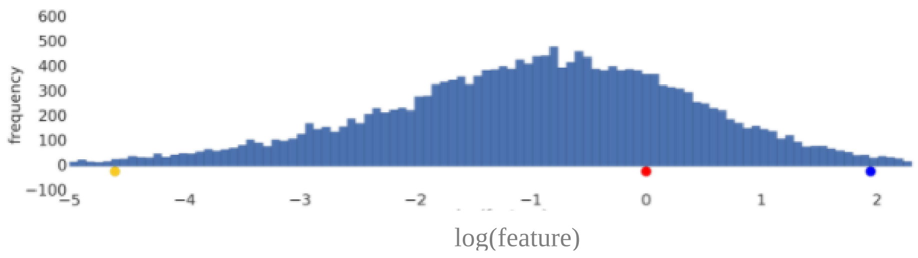


Figure 4. Gaussian distribution (Shah 2017)

2.3.3 Quantiles

Quintiles are a technique of making distributions identical in properties. To quantile two or more distributions to each other with reference distribution as before and then setting average of these distributions. So, the similarity between the examples decreases when the number of examples among them increases. Data normalization reproduces data distribution, so applying log transformation does not reflect intuition on how similarity works. Instead of the data divided into intervals where every interval has an equal number of data objects, these boundaries are called *quantiles*. Converting data into quantiles is done by the following steps (Murtagh 2017). Firstly, several intervals are decided; each interval is defined and every value is replaced by the index of the interval it falls into. Then indexes are converted into predefined ranges defined by their features by scaling the values of the index. Quantiles are sometimes a good choice to distribute the data more evenly. Two Quantiles are taken here as an example:

- $sim(A, B) \approx 1 - |\text{prob}[x > A] - \text{prob}[x > B]|$
- $sim(A, B) \approx 1 - |\text{quantile}(A) - \text{quantile}(B)|$

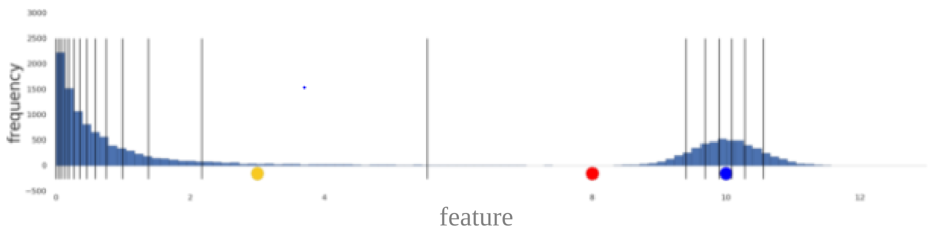


Figure 5. Quantiles (Murtagh 2017)

Researchers identify different methods of scaling and clustering with several goals. Our goal is to simplify the data and describe it in an organized way to visualize the clusters. Scaling methods help to reduce the complexities of the dataset by identifying its underlying data set structure. Some researchers use multidimensional scaling by discovering four-dimensional situations like:

1. Cooperative and friendly vs. competitive and hostile
2. Equal and unequal
3. Informal or social-emotional vs. task-oriented or formal
4. Superficial vs. intensive

2.4 Algorithms

Clustering can be defined as a problem and a clustering algorithm that can be solved using various algorithms that are different in their characteristics and cluster constitution and how efficiently we can find them. The critical challenges in clustering are what is a cluster, how to measure goodness of clusters, and the variance inside the cluster. For example, in centroid-based clustering, clusters are primary vectors, not necessarily a data set member. It is necessary to modify data until the results are according to the desired properties. When clusters are fixed to k, then k means clustering is performed, which gives the formation optimization problem, finding the cluster center and assigning objects to the nearest center of a cluster like square distance from the cluster.

In clustering, outlier detection and removal is considered a preprocessing step, so the idea here is to clean the data from outliers that can affect the cluster quality analysis. The approach to performing clustering first and label the objects as outliers that fail to fit into any cluster. Removing outliers improves the clustering, and performing clustering first facilitates outlier detection. Outlier detection consists of two steps known as scoring and thresholding. Computing outliers relies on reference choice in the dataset. *K* nearest-based outlier detector uses neighbor objects and tunes their size by setting *k* value. When the outlier number is significant, the most traditional detector becomes ineffective (Yang & Rahardja 2021).

3 Clustering Algorithms

Clustering an unsupervised machine learning technique splits the data points into subsets (called clusters) based on similarity (data objects in the same cluster are similar to each other in some sense) basis. Clustering is also known as *aggregation* (Wu, O. et al 2012), whose result does not have any, predefined labels. It is one of the most common data analysis techniques used to get insights into the data structure and identify the subgroups as shown in figure 6. Data points in the same subset are similar and data points in different subgroups are different (Malinen 2014). We can say that data points in each cluster would be identical as possible according to the similarity like Euclidean distance to find the homogeneity of the subgroup. These similarities depend on the decision.

The similarity of data points is measured based on their characteristics, where similar values are grouped called clusters. Every cluster has some data points that share common properties, so all collections of clusters compose a clustering.

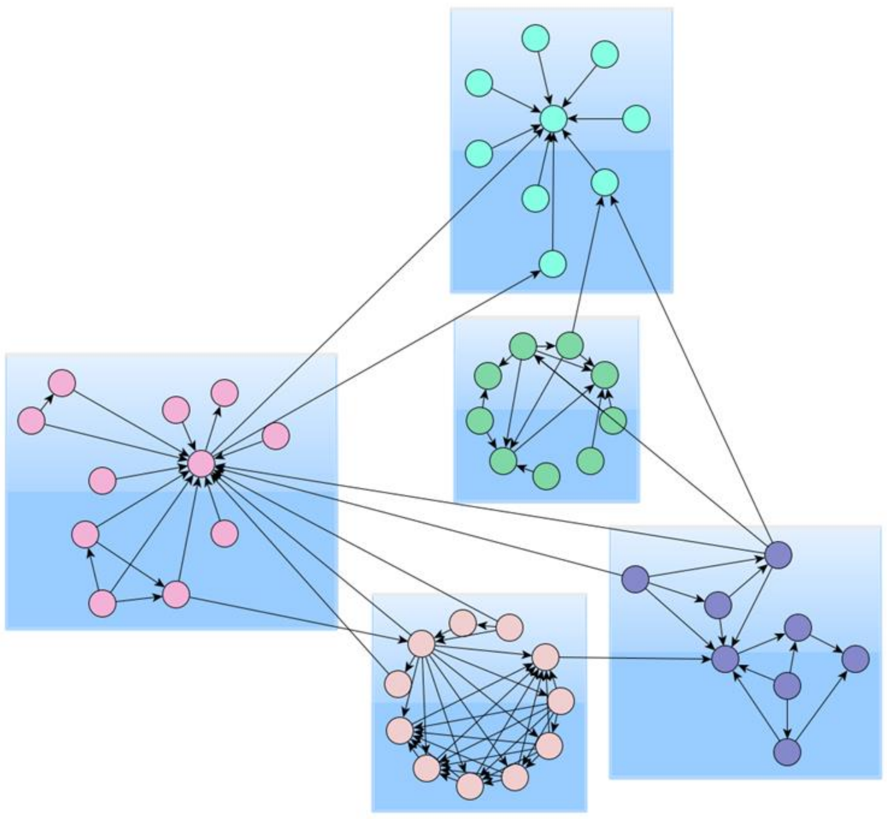


Figure 6. Clustering (Kassambara 2017)

When we try to find sample subgroups, feature-based clustering analysis is done to find samples based on their features. Clustering is unlike supervised learning, so considered an unsupervised learning method. In a supervised learning algorithm, the algorithm learns the data sets labeled and provides them with an answer key so that the algorithm can evaluate the accuracy of training data. At the same time, unsupervised learning algorithms give unlabeled data that try to make sense by extracting the patterns and features independently (Saraswat 2017), as seen in the figure 6.

Most of the clustering algorithm works on similarity computing basis means their runtime increase with square numbers of n example as O(n^2) with complexity notation. The algorithm with complexity O(n^2) is no practical, so the focus of this work is k-means has complexity O(n) means the algorithm proceeds linearly with n. To reach the clustering, several approaches exist, as researched by Hagan Murphy (2018) in his comprehensive review of clustering algorithms.

Objective function

The clustering goal is expressed as an objective function that depends on proximities of cluster centroids and points to one another. The objective function is to minimize the square distance of every point to its nearest centroid to improve the clustering performance.

3.1 Clustering approaches

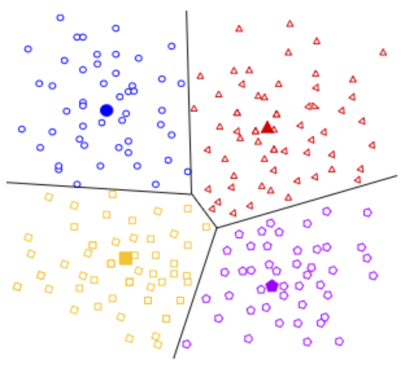
Centroid-based clustering algorithms are efficient but very sensitive to initial conditions and outliers. For centroid-based clustering, k means is the most widely used example (Kassambara 2017).

Density-based clustering connects high-density areas into clusters and allows arbitrary-shaped distribution to extended dense connected areas. Outliers are not assigned in this algorithm and have difficulty for varying high density and dimensions.

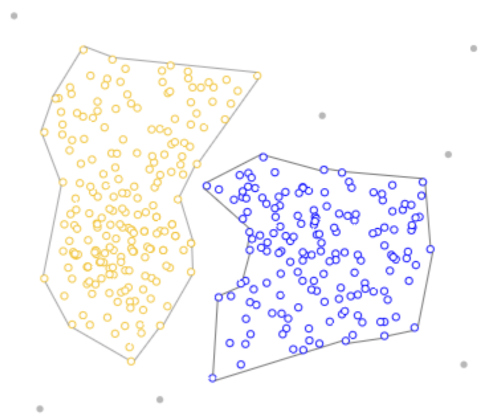
Distribution-based clustering has just like Gaussian distributions. As Gaussian distance increases, probability belongs to distribution decrease (Zhang, Z. et al 2019).

Hierarchical clustering creates a tree of clusters with well-suited hierarchal data like taxonomies, as discussed by Lukjancenko, O., Wassenaar, T. M., & Ussery, D. W. (2010).

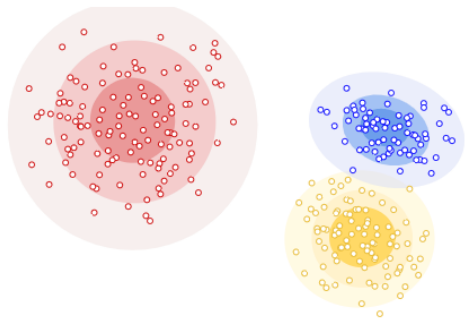
Centroid based



Density-based



Distribution-based



Hierarchical

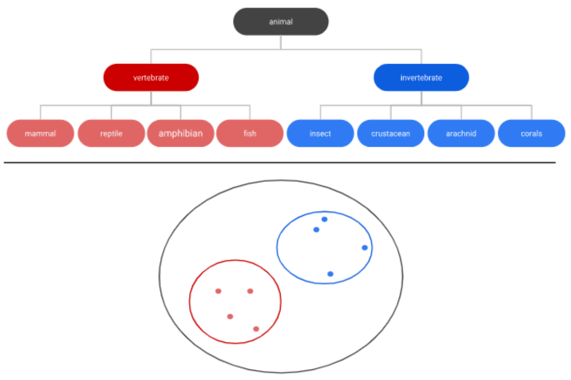


Figure 7. Examples for different clustering approaches

Table 1 Algorithms Objective Functions

Clustering Algorithm	**Research**	**Clustering Approach**	**Properties**
K-means	Irawan (2019) Zhang et al, (2018)	Centroid based Clustering	• Help to scale a large set of data
Random swap	Fränti, (2018) Fränti et al, (2008)	Hierarchical clustering	• Proved to be very efficient • can drive expected time complexity
Gradual K-means algorithm	Malinen et al (2014)	Hierarchal clustering	• Have better performance when mutation decreases to some point • Computes faster if variables are enormous.
Density Weighted K-means	Goceri et al, (2018) Kerdprasop et al, (2005)	Density based clustering	• Improved version of the k-means algorithm • Improved scalability.

3.2 K-Means Algorithm

The clustering K-means algorithm is a method of vector quantization that aims to partition the n observations into k clusters. The iterative algorithm tries to partition the sets of data into *k* predefined non-overlapping subgroups (see figure 8). The data points belong to each group and try to make cluster data that would be possibly similar or different (Goel 2020). The data points assigned to the cluster would be the sum of squared distance between cluster centroid, and data points are at a minimum. If the data points' variations are low, then data points become more homogenous within that cluster. The K-means algorithm's primary use is to get meaningful data structure intuition through which we are dealing with and predict where different models will build for other subgroups. K-means algorithm identifies the no of k centroids and allocates every data point to its nearest cluster by keeping centroids small as possible (Malinen 2014).

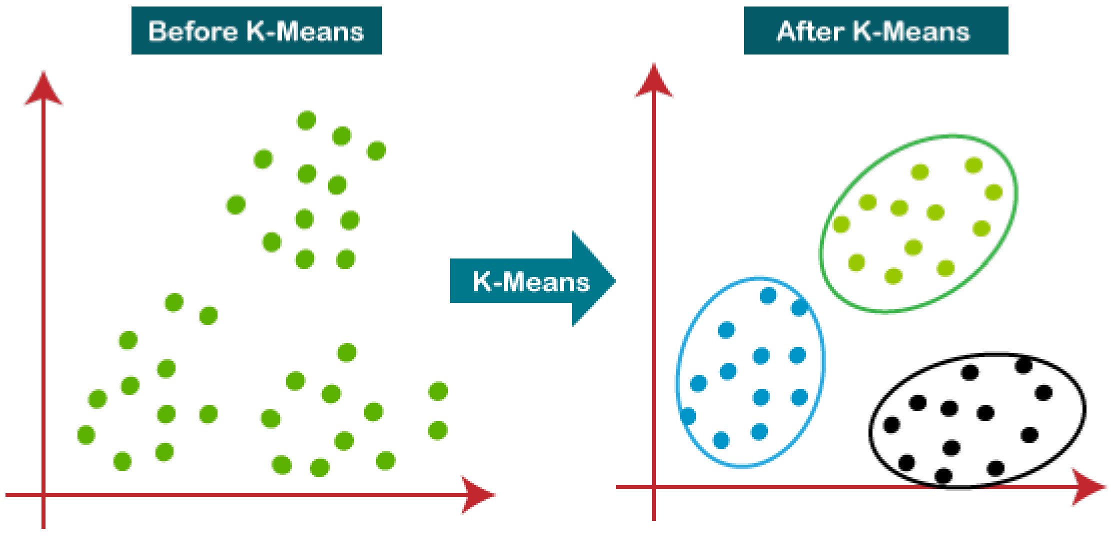


Figure 8. K-means (Malinen 2018)

It starts working by initially selecting the k number of random data points called centroids. Each data point is assigned to its nearest centroid (to measure distance, some method is used, e.g., Euclidean distance or haversine distance). In the next step, every centroid is moved to the average position of all the data points assigned to it (the middle part of that cluster). And then, these steps are kept repeating until centroids no longer change (then the algorithm is said to be converged) (Khanmohammadi, 2017). The distance-based approach is used to calculate the data points within a fixed neighborhood. The algorithm calculates the data points and neighborhood approach find the distance define by a k-nearest neighbor where k is the input parameter (Fränti, P. & Sieranoja 2018).

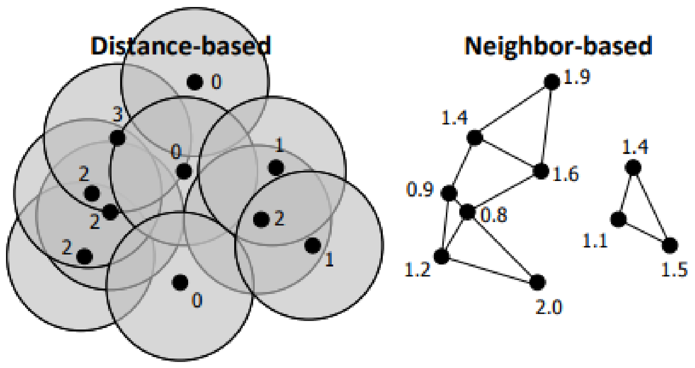


Figure 9. Distance-based and neighbor based (Fränti, P. & Sieranoja 2018)

The K-means clustering algorithm finds out characteristics and similarities of the grouped data and classifies them according to their properties or similarities. K-means algorithms can label each data point itself according to attributes as shown in figure 10.

The following results are obtained with the use of k-means clustering:

- Centroids k clusters are used to label new data.
- Labeling to train the data like each data point is assigned to every cluster.
- There is no such method that determines K's exact value, but in this article, we determine the accurate estimate by using the metrics technique. One metric is used to compare the results of the importance of K, which means the difference between data points in centroids.

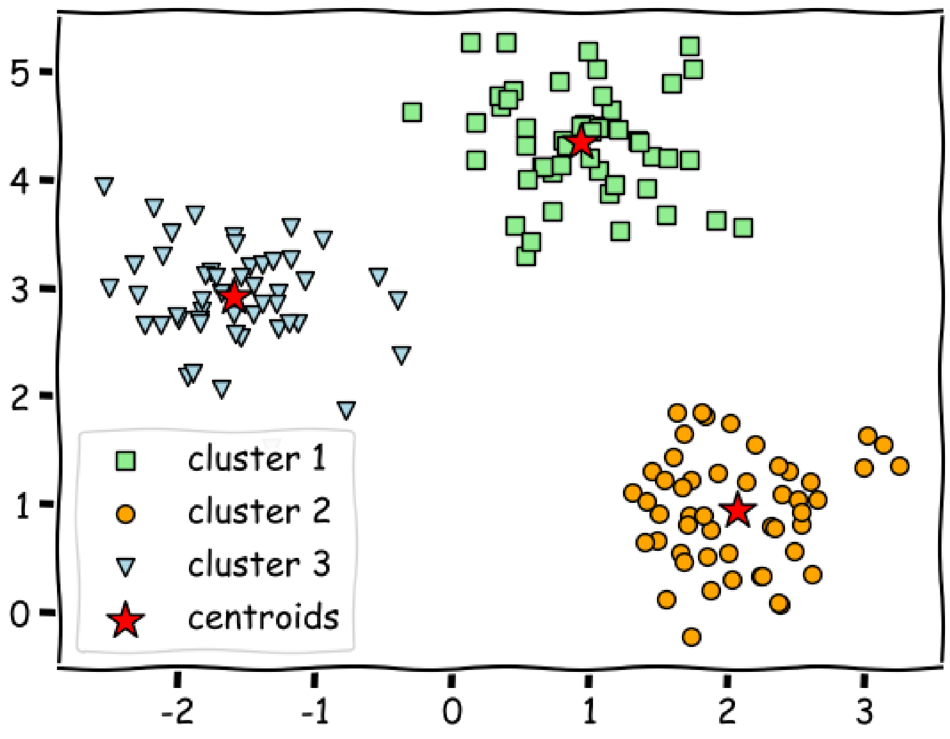


Figure 10. K-means clustering (Zhang 2018)

The purpose of the K-means algorithm is to set every data point which is closest to the centroid. K-means compute the centroid and iterate it until it finds the optimal centroid. Data points in the algorithm are assigned to cluster in such a way that the sum of squared distance between data points and centroids would be minimum. Clusters are formed by minimizing the mean square error by looking at the fixed number of *k* of the cluster in the dataset. Mean squared error (MSE) measures the average squares of the errors, the average squared difference between the estimated values and the actual value. The aim is to find the set of *k* clusters such that every data point is assigned to the closest center, and the sum of the distances of all such assignments is minimized.

Algorithm 1 k-means algorithm

```
1: Specify the number $k$ of clusters to assign.
2: Randomly initialize $k$ centroids.
3: repeat
4:     expectation: Assign each point to its closest centroid.
5:     maximization: Compute the new centroid (mean) of each cluster.
6: until The centroid positions do not change.
```

When we give the number of sets k or select any collection of data, a group of clusters is generated at a specific distance. These data points are selected randomly from datasets and called clusters. We can say that the distance between them depends on the similarity and characteristics of the data points. If two data points are close to one another, we can say that clusters are similar or have similar properties. When the iterative approach is implemented, all the data sets go to their position; a central or average point shows a particular cluster's location. So we can say that the selection of nearby data points updates the average location. When the mean position is updated, the position of the centroid changes as well. When the position of the centroid changes, then again algorithm will iterate to see which data points are near to one another, and iterations continue in this way (Khanmohammadi 2017).

The objective of the k-means algorithm is to improve the performance of clustering. For this purpose, we measure SSE, a Statistical method for measuring absolute difference from the actual value to the achieved weight. Euclidean distance helps to find the nearest centroid to compute the sum of square error. For example, with a given set of the cluster, two clusters produce two different K-means runs, so we prefer small square error, which means the prototype of clustering has a better representation of points in clustering.

$$\boldsymbol{SSE} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{K}} \sum_{\boldsymbol{x \in C_i}} \boldsymbol{dist(c_i, x)^2} \tag{7}$$

SSE value can be minimized directly, which is our objective function, by assigning a point to the nearest set of centroids. D is the distance between the cluster center and data. The formula sum of square error SSE is used to measure performance among obtained data by the prediction that has been done previously. SSE is a research reference for determining optimal clusters. Through K-means, it is better for finding the optimum cluster center by modifying the K-means clustering algorithm, and it is based on the minimum sum of a squared error value. As K increases, SSE decreases, and distortion will be small. K-means algorithm can be improved by considering the initialization better and repeating k-means several times using the different initial solutions. As we know that K-means is NP-hard so it does not offer to achieve the exact optimal value of SSE.

3.3 Random Swap

K-means clustering is perfect when we want to capture the data clusters' insights if clusters are spherical. It will try to construct a perfectly spherical shape around the centroid, which means when the cluster has a complex geometric shape, K-means does not perform its jobs well in clustering data (Saraswat 2017). The First K-means algorithm does not let the data points far away from each cluster's share through which they even belong to the same cluster. Second, the data generated from a multivariate normal distribution having different means and having standard deviations.

K-means algorithm has many weaknesses, and many variations have been introduced to overcome these weaknesses. One of the failings of the k-means algorithm is, it may get trapped in the local optimum. Apart from the *k* value of the k-means algorithm is very sensitive to the initial conditions, so that we can run this algorithm several times with the same value of *k*. Then, we can put each instance in each unique cluster with the use of the consequence function. To overcome this issue, the Random Swap k-means algorithm has been introduced as shown in figure 11.

Random Swap k-means algorithm is based on k-means, where it takes *N* number of data positioned into *k* clusters and uses centroids. The functions used are SSE the same as k-means because k-means do not go ahead further if local minima reach and stops, which may lead to a wrong solution. That is why to overcome this problem random swap algorithm is developed. Firstly, a centroid is selected randomly and swapped in a random location. K-mean algorithm runs after random swap with a pre-decided number of times. These steps are repeated with a pre-decided number of times. If these new solutions have lower SSE at each iteration after swapping and k-means, then the new solution is evaluated. Otherwise, some other centroids are chosen randomly and swap again (Fränti 2018).

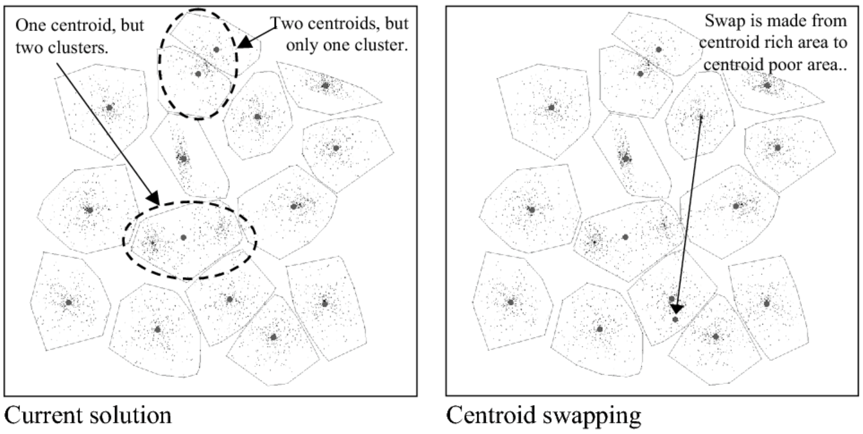

One centroid, but two clusters.
Two centroids, but only one cluster.
Swap is made from centroid rich area to centroid poor area..
Current solution
Centroid swapping

Figure 11. Random swap clustering (Pham 2020)

The benefit of using a random swap algorithm is it is straightforward to implement and has a minimal extension. The primary purpose of using a random swap algorithm is to efficiently solve the swapping prototype's clustering sequence and fine-tuning their exact location using k-means. Using a randomized search strategy proved very simple and efficient; the process goes to have a perfect quality clustering. If it would no longer be iterated, they have a high probability of finding correct clustering. The most crucial observation in using that algorithm is that it is unnecessary to swap one redundant prototype. Still, it would be easy to remove any prototype in the middle of the neighborhood, which is enough since k-means to fine-tune the exact location locally (Hong 2020).

Two steps are involved in the random swapping algorithm in which one of which is known as run-k-mean, and the second is the centroid step. Only one additional step known as prototype swap is included when implementing the random swap algorithm in the k-means algorithm. This step does not depend on the data and the functional objectivity, so it is trivial to implement to make it highly useful for the practice. After implementing a random swap, we find the correct location of the cluster every time. Random collection removes the existing cluster and creates new data space. Alternative swap implementation creates a new cluster by choosing an existing random cluster and selecting a random data vector within this cluster. After the swap, local repetition is done to update the partition, so all the process in a single iteration depends on implementation with following steps.

1. Prototype swapping
2. Old cluster removal
3. The new cluster is created
4. Update the affected prototype
5. K-means iteration

During iteration, three different swaps are done: one is known as a trial, the other is accepted, and the last one is successful. During the trial swap, every iteration improves the objective function known as an accepted swap. If swap becomes successful can be understood as successful (Fränti 2018).

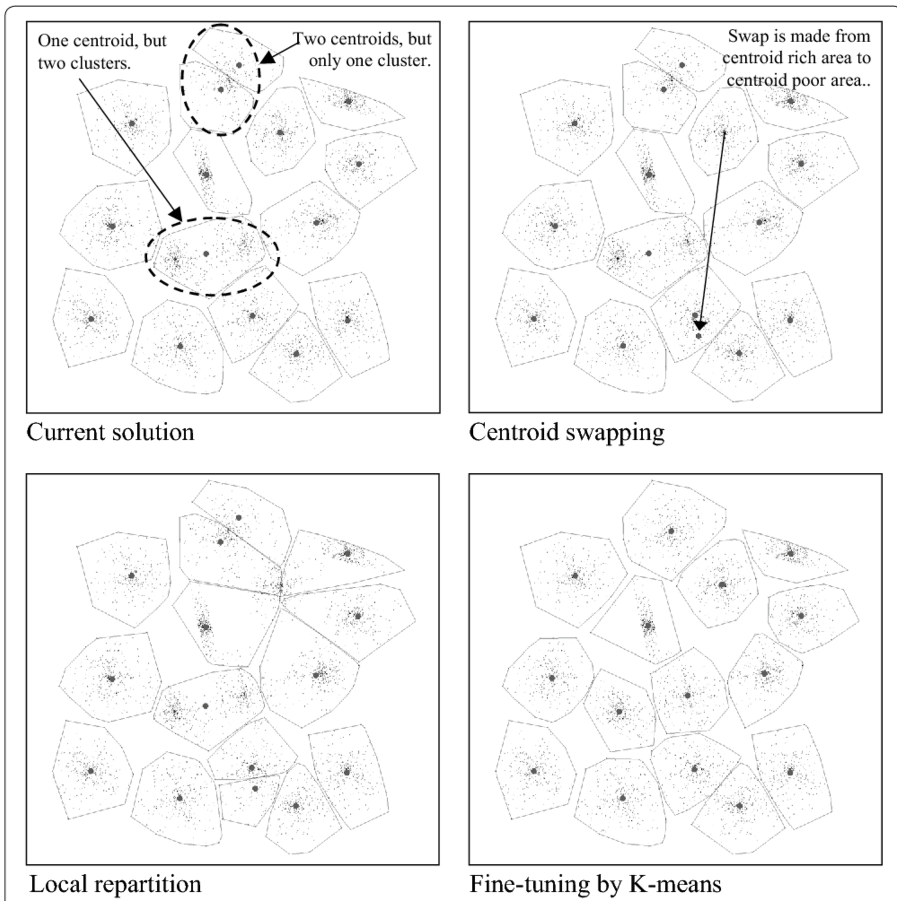

One centroid, but two clusters.
Two centroids, but only one cluster.
Swap is made from centroid rich area to centroid poor area..
Current solution
Centroid swapping
Local repartition
Fine-tuning by K-means

Figure 12. Random swap clustering (Cho 2020)

At the initial stage, the centroid is chosen randomly at a random location. The k-means algorithm runs a pre-decided number of times. Steps are repeated according to the pre-decided number of times. If any new solution after swapping and k-means with lower SSE exist than earlier iteration, then the new solution is accepted. Otherwise, other centroids are chosen randomly and then swapped again. Some steps are involved, which are:

6. Random swapping centroids
7. Run k-mean

The whole pseudo-code for the implementation of random swap algorithm is:

Algorithm 2

```
**RandomSwap**($X$, $C$, $P$): updates $C$ and $P$
    Repeat $T1$ times
        $C_{new}$ = $SwapCentroid(X, C)$;
        $P_{new}$ = $Partitioning(X, C)$;
        Reapeat $T2$ times
            $(C_{new}, P_{new}) = KMeans(X, P_{new}, C_{new})$;
        if($SSE(C_{new}, P_{new}) < SSE(C, P)$)
            $(C, P) = (C_{new}, P_{new})$;
```

If we explain the K-means algorithm, then the distance among two centroids is the inter-cluster distance, and that inter-cluster distance is measured in many ways, like the maximum distance of two vectors in the *k* cluster. Inter-clusters are maximized to achieve cluster in inter-cluster, and then cluster distance could be minimized. The correlation exists between clusters and SSE when SSE decreases the number of clusters increase. The objective of this function is to help the quality cluster understanding in k-means iterations, so the value of the objective function help in understanding the compactness of the cluster. There are many objective functions like graph cut objective, ratio cut, WWS, and Dunn index to minimize and maximize the objective function. The objective function is selected not with a trivial job, but it depends on the nature of the data set and clustering algorithm.

All the process of random swap is just the variation in K-means. Every time two iterations run and check any random centroid point. Mean squared value is found out by swapping the arbitrary values and this process is done when the mean square value of that data point is less than the first data point value. If the means square error MSE value becomes low, then the swapping process is done; otherwise, there is no change. At each iteration of the swap, the value of MSE becomes better. If the MSE value becomes low gradually, we can say that swap becomes successful, and the algorithm continues (Bouzos 2018).

3.4 Gradual K-Means

Gradual K-means algorithm was also introduced to overcome the problem of solving the global allocation of clusters. Compared to traditional approaches, where different methods are proposed to fit the clustering model to data to solve correct global allocation, data is fitted to an optimal clustering structure as represented in figure 13. First, an artificial dataset of identical size and dimensions as input is generated so that the data vectors are divided into *k* perfectly separated clusters without any variation. Bijective mapping of the input data to the artificial data is then performed (M.I. Malinen 2014).

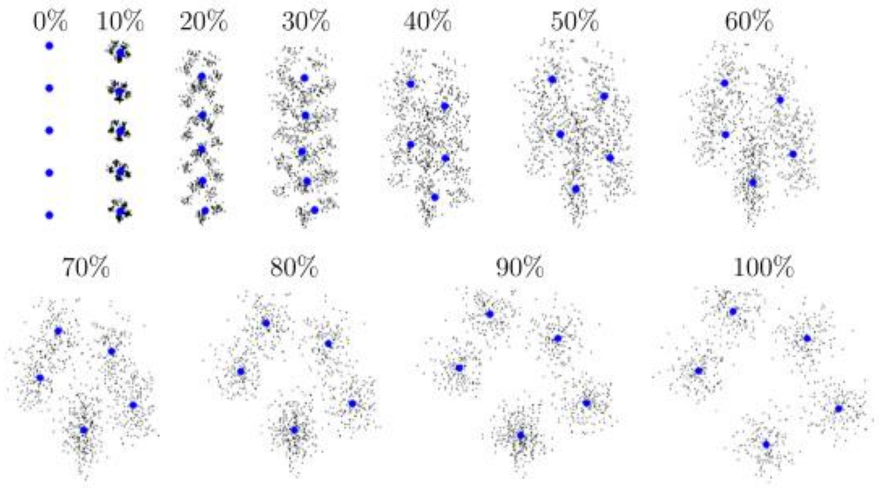


Figure 13. Gradual k-means clustering (Malinen 2014)

k-means is a very traditional approach to fit a model for the data given, like the prototype, and overcome the optimum local problem. In this algorithm, an opposite approach is taken to fit the data into a given cluster, which proved optimal for similar pathological data with equal size and dimensions. The algorithm is as follows:

Algorithm 3

```
Input: dataset $X_1$, number of clusters $k$, $steps$,
Output: Codebook $C$.

$n \leftarrow size(X_1)$
$[X_2, C] \leftarrow Initialize()$
for repeats = 1 to steps do
   for i = 1 to n do
      $X_{3,i} \leftarrow X_{2,i} + (repeats/steps) * (X_{1,i} - X_{2,i})$
   end for
   $C \leftarrow kmeans(X_3, k, C)$
end for
output $C$
```

The inverse transformation of data is used to solve the K-means associated problems. The gradual K-means algorithm is a little bit different as compared to the other two algorithms. In these algorithms, we try to set the centroids according to the data sets to control the centroids at a proper location where similarity can be found. In the gradual K-means algorithm, we perform the reverse of it. Here we are not controlling the centroids within data points, instead we are trying to fit the centroids into the data points. Firstly, in the implementation process, we take a central point as an initial point where we can initialize our centroids. Suppose we bring all the data set at a time and then transform it into multidimensional plan data. After transferring all data, we combine all the data at the location of centroids. We can say that we locate the data here in a compressed form. After following step by step guide, we send data back to its original position (Wang 2020).

3.5 Density Weighted K-Means

The Weighted K-means algorithm is also known as W-K-Means that calculates the weights automatically. The weights of variables produced using the algorithm measure the variable importance in clustering as shown in figure 14 (Guérin, J. et al 2017).

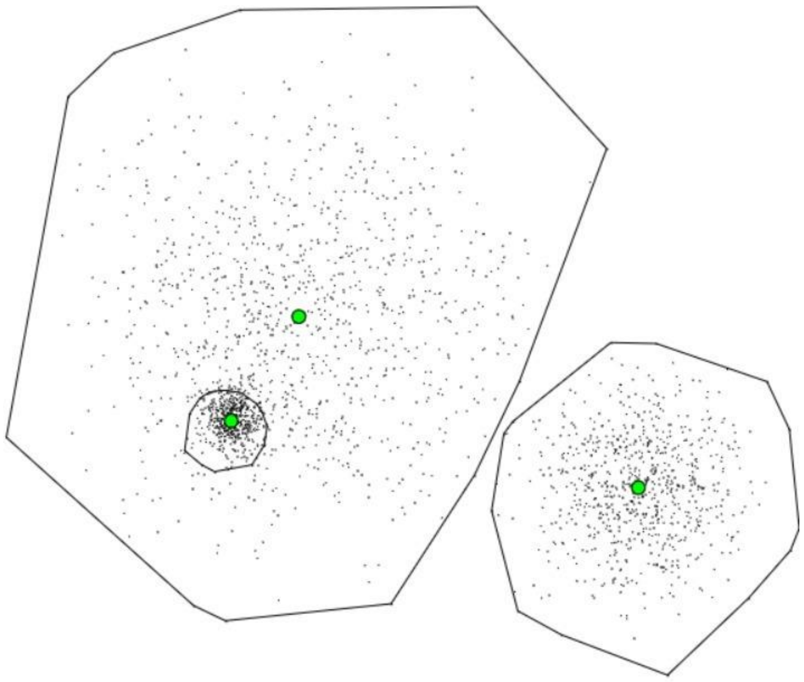


Figure 14. Weighted k-means

This animation uses the weighted algorithm in data mining selection applications when we have large and complex data, so selecting higher weighted variables and removing the lower weighted variables produced good clustering results. For example, we have:

$W_f = \{w_1, w_2, w_3, \ldots, w_m\}$ these are the weights of m number of variables.

The following algorithm is used for weighted K-means clustering:

```
CalculateNewWeights($X, C, P, W$): updates $W$
    $density\_sum = 0;$
    for $i = 1$ to $K$
        $md_i = CalculateMeanDistance(X, C_i, P_i);$
        $d_i = CalculateDensity(md_i, P_i);$
        $density\_sum = density\_sum + d_i;$
    for $i = 1$ to $K$
            $W_i = CalculateClusterWeight(d_i, density\_sum);$
```

We have a set of clusters with data points X, but we want it to stipulate that some points are more critical than others, so we encounter this problem. We use a weighted algorithm to decide where to put data points that have particular importance. We chose some convenient centers for each data point for how much that data point has to travel from one point to the cluster center represented in figure 15. The natural way to record the importance of different data points is to assign a weight to each one. The problem becomes a simple case where each point was given the same weight. Additionally, the K-means algorithm handles when some data points are so important that they pull the cluster close to the gravitational attraction (Liu, H. & Wu 2017).

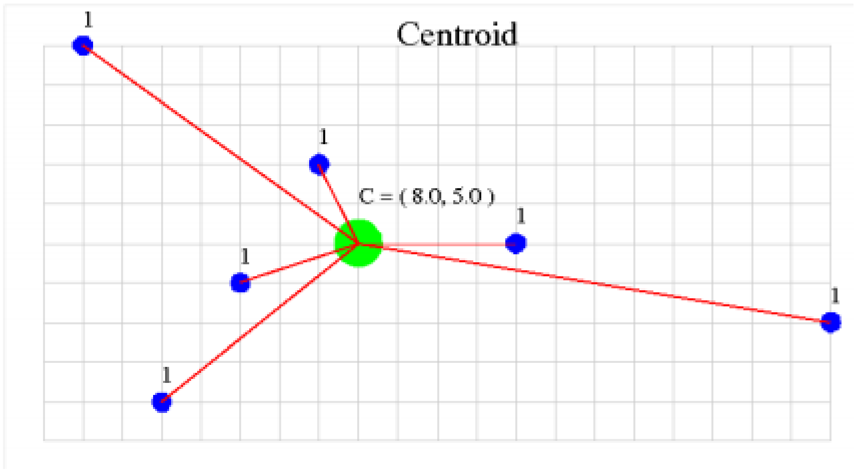


Figure 15. Centroid location

For weighted clusters, we define weighted cluster variance in the form of the following equation.

$$\boldsymbol{var}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{c}) = \sum_{j=1}^{k} \boldsymbol{var}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{c_j}) = \sum_{j=1}^{k} \frac{\sum_{x_i \in c_j} \boldsymbol{w_i}||\boldsymbol{x_i}-\boldsymbol{c_j}||^2}{\sum_{x_i \in c_j} \boldsymbol{w_i}} \qquad \textbf{(8)}$$

C_j is the center of the cluster. At each step, weighted K-means that the algorithm reduces the cluster variance weight, and best clustering minimizes the quantity. While using weighted K-means, some changes occur, which are of two types (Sane 2020)

- Center update: mass can be used instead of centroid
- Variance: computation of weighted variance sum

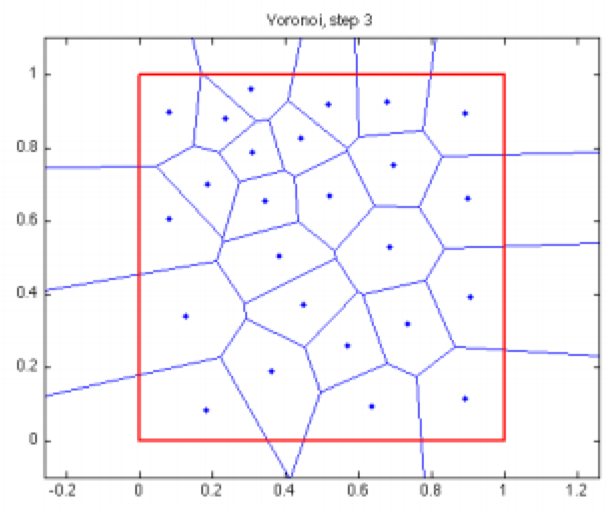


Figure 16. Center and cluster

Weighting allows it to be more sophisticated grouping by considering the importance of things to be more valuable and have more information.

The objective is to obtain a Sum of Square Error SSE, which refers to the derivation among data points from its centroids. Therefore, SSE value is obtained by taking the sum of all square errors at all data points. Declining the value of SSE cannot be longer significant. To improve the algorithm's performance, it would be better to replicate the whole algorithm many times and select a result with the best replication that gives the smallest SSE value. The objective function is defined as follows:

$$\arg\min_{S} \sum_{i=1}^{k} \left(\sum_{x_j \in S_i} ||x_j - \mu_i||^2 \right) \qquad \textbf{(9)}$$

Which tries to minimize the sum of all squared distances among clusters for all available clusters.

4 Algorithm visualization and animation

Visualization can be defined as any technique to create images, diagrams, or any kind of animation to convey a message but as we are focusing on visualization of algorithms, for us the term visualization will refer to animations as we are going to show the work process of algorithms and showing that in a series of images (animation) would be efficient.

For Visualization of algorithms, we have developed a web-based application called clustering animator and the focus would be to discuss and document that why specific methods and techniques have used, what were other possibilities and which method is applicable on a specific algorithm, and where it would not be a good thing to do so. And those algorithms are specifically used for clustering, so all of the algorithms share many common characteristics, and components used for visualizations of those algorithms are also common among all algorithms. Some of those common components are:

- Centroids and Clusters
- Convex Hull
- Voronoi Diagram

In clustering algorithms tend to split the given dataset into different or same groups or categories based on some common characteristics, those groups are called clusters. And we call the average or mean position of the cluster a centroid. In simple words, a centroid is the center position of a cluster as shown in the figure 15.

In computational geometry, many algorithms are used, which are known for computing the convex Hull with the finite number of data points and other geometric objects. The reason behind implementing a convex hull means that constructing an unambiguous and efficient representation of the convex shape that is shown in the figure 17 as outlines.

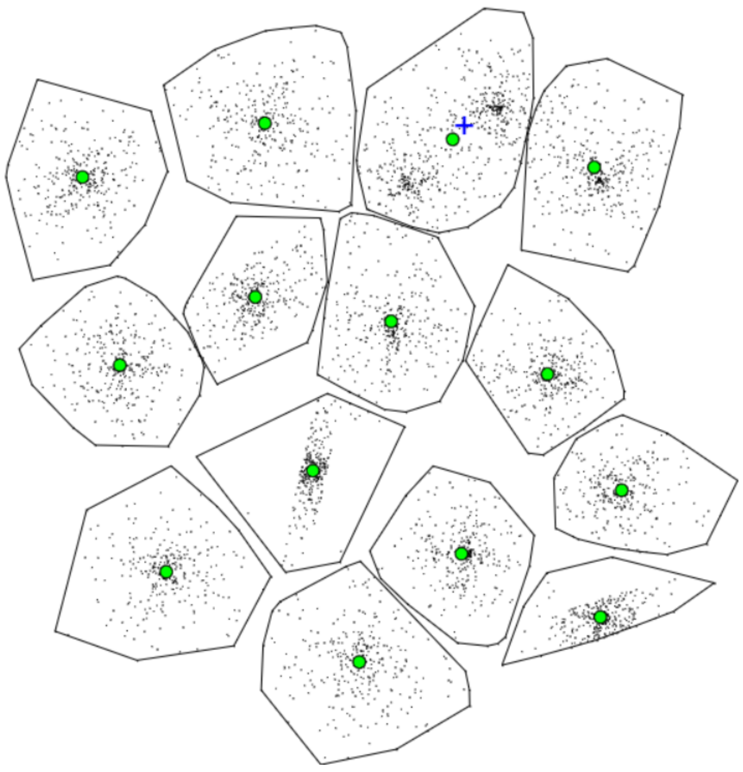


Figure 17. Convex hull

Algorithm

Convex Hull produces convex shape spherical clusters, which are convex shaped, so the output representation considered for the convex Hull of data points includes the list of inequalities that describe the hull facets, an undirected graph with facets and adjacencies, and full-face lattice hull. For two convex Hull, two or three dimensions corresponding to complexities and algorithm is usually estimated in n terms of several inputs and h number of data points in the convex Hull which are significantly smaller than n. for getting higher dimensions faces dimension come into an analysis where Graham scan computes the n points of the convex hull in-plane with time complexity of O(*n log n*).

As its name represent convex Hull having various objects with a broad range of application in computer science. Convex Hull means a non-ambiguous and efficient representation of convex shape constructed. The complexity found in the evaluated algorithm is estimated in terms of number *n*, input point, *h,* and points in the convex hull. With the time of development and varying the time complexity, various algorithms are stated: input points and hull point *h*. Gift wrapping is one of them, which is a most straightforward algorithm and created independently by Chand. The complexity of the Gift-Wrapping algorithm is O(n^2). Complexity can be measured in O(*nh*), where *h* is hull size, and *n* is input size. Graham Scan is another convex hull algorithm that is more sophisticated but more efficient. Its points are sorted by their coordinates or by the angle at a fixed-sized vector. The time complexity for Graham's algorithm is O (*n log n*). By analyzing the complexity, we perform sorting in the form of O (*n log n*), and the time is:

$$\mathbf{time} = \sum_{i=1}^{n} (\boldsymbol{D_i} + \mathbf{1}) = \boldsymbol{n} + \sum_{i=1}^{n} \boldsymbol{D_i} \qquad \mathbf{(10)}$$

D_i is the number of points on processing P_i, and at each point is pushed on stack only once. When one point is popped, it cannot be popped again.

Quickhull Algorithm

Quickhull is another convex hull algorithm that has time complexity O(*n log n*). Another algorithm is Monotone Chain which can be seen as a Graham scan sorting the points lexicographically by its coordinates. Kirkpatrick Seidel algorithm is the more sensitive output generated algorithm that modifies the divide and conquers algorithm using the marriage technique before the conquest. Chan algorithm is a simpler optimal output generator algorithm that combines gift wrapping with O(*n log n*) execution on small subsets of inputs (Zhang 2020). In this process, the Graham scan algorithm is used to find the convex hull algorithm in O(*n log n*) time. Graham's scan algorithm is a very efficient algorithm and helpful in finding the convex Hull with a finite set of data points with O's time complexity (*n log n*). Through this algorithm, all the convex hull vertices are ordered along the boundary (Alshamrani 2020).

The bottom-most point is to find out after comparing coordinates at all points having two issues with the same y value, then point it with a small coordinate name as x considered. Consider the remaining points n-1 and sorting them clockwise around their points. The nearest point is first if two points of polar angle are the same. After sorting, if two points have the same angle, then the same point angle is removed except the farthest point. If the value of m is less, then three convex hulls are not possible. An empty stack is created at point 0, and 1 and 2 to S. processing starts at the reaming points one by one.

The Voronoi diagram is basically a partitioning method (see in figure 18) that separates the data points at a point distance but does not position it. Three basic components are considered important for Voronoi diagrams are vertices, edges, and generators.

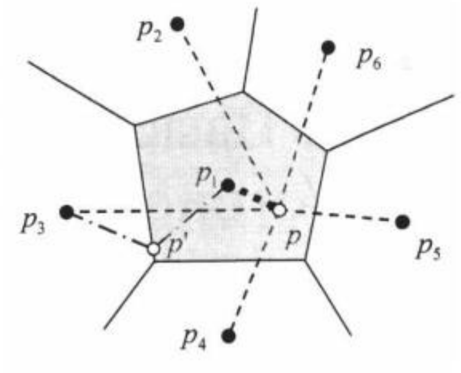


Figure 18. Basic Voronoi diagram **(**Kang 2008)

If we define the Voronoi diagram in two dimensional way then we can formally define it in the ordinary form. The first location of point p_i is donated and its corresponding vector is x. Let

$P = \{p_1, p_2, .., p_n\} \in \mathbb{R}^2$ where $2 \le n < \infty$ and $p_i \ne p_j, i \ne j$ and $\forall\ i, j = 1,2, \dots, n$ the set of generated points and generators so we can call the particular regions given by

$$V(p_i) = \{\vec{x} |\ ||\vec{x} - \vec{x_i}|| \le ||\vec{x} - \vec{x_j}||\ \forall_j \ni i \ne j\} \quad \textbf{(11)}$$

The Voronoi region p_i at $\|\bullet\|$ is Euclidean distance. $V(p_i)$ is referred as V_i. The Voronoi regions are connected and become convex. The evaluated set is represented as

$V = \{V(p_1), V(p_2), \ldots, V(p_n)\}$ Voronoi diagram of P. For V different notations is $\bigcup_{i=1}^{n} V_i$.

As long as we develop the distance function of our problem, we can make a Voronoi diagram; otherwise, we can make a Voronoi diagram without visuals. We would stick and would not get precious medium reads (Sane 2020).

- Voronoi diagrams are easy to understand and apply but cannot deal with real-time distances.
- Apply on the overall complexity of the algorithm
- Recourse intensive
- Naïve implementation is complex
- Fortune algorithm is complex

Weighted Voronoi Diagram

If generated points along their location have equal weight and value, assigning the distinct weight to the generated point will be more beneficial than uniformly points in scenarios. Weighted generator points are more applicable. From the basic Voronoi diagram definition, the region Vi at intersection dominates Pi over every P point generator. The weighted distance between two points like x and y is:

$$Dom_w(p_i, p_j) = \{p | d_w(p, p_i) \leq d_w(p, p_j)\} \quad \textbf{(12)}$$

$\boldsymbol{V}_w(p_i), \boldsymbol{or}\ \boldsymbol{V}_w(\boldsymbol{i})$ is weighted voronoi region and $V_w = \{V_w(p_1), V_w(p_2), \ldots, V_w(p_n)\}$ is weighted voronoi diagram.

There are many ways for the Voronoi diagram construction (Kang 2008).

- Plane Sweep Method
- Tree Expansion Method

In the Plan Sweep method, we draw Voronoi vertices, Voronoi edges, and generated points at the non-cartesian plane. A one-to-one correspondence exists between the cartesian plane and the non-cartesian plane at initial assumptions.

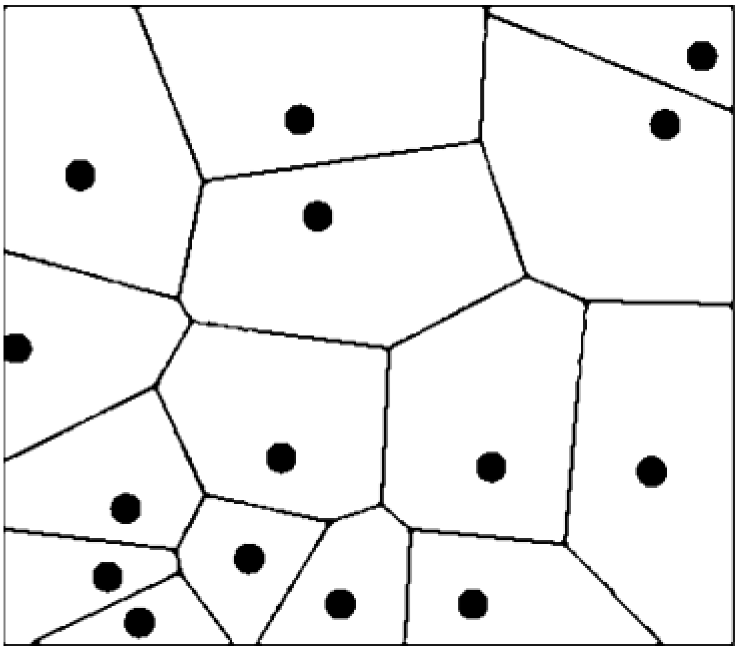


Figure 18. Voronoi diagram (Roskos-Ewoldsen, 2020)

The expansion and deletion method is a good system for constructing the Voronoi diagram and a programmable way to do it. Just like the Plan Sweep method, some assumptions are also made in this method. All polygons and vertices are labeled undecided and non-incidentally.

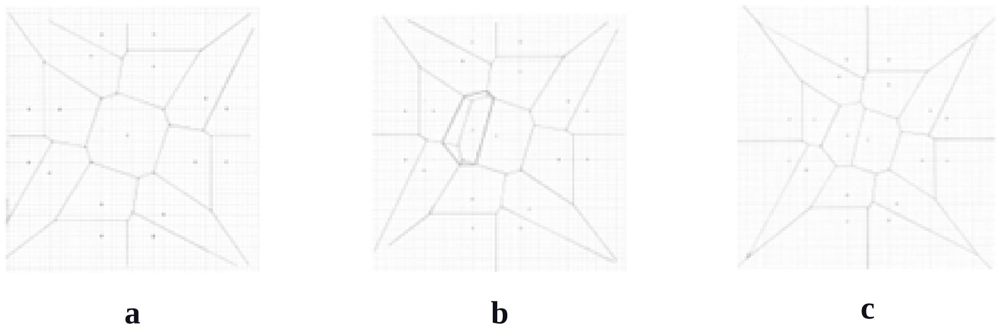


Figure 19. Implementation of tree expansion and deletion algorithm (Kang 2008)

4.1 Clustering Animator

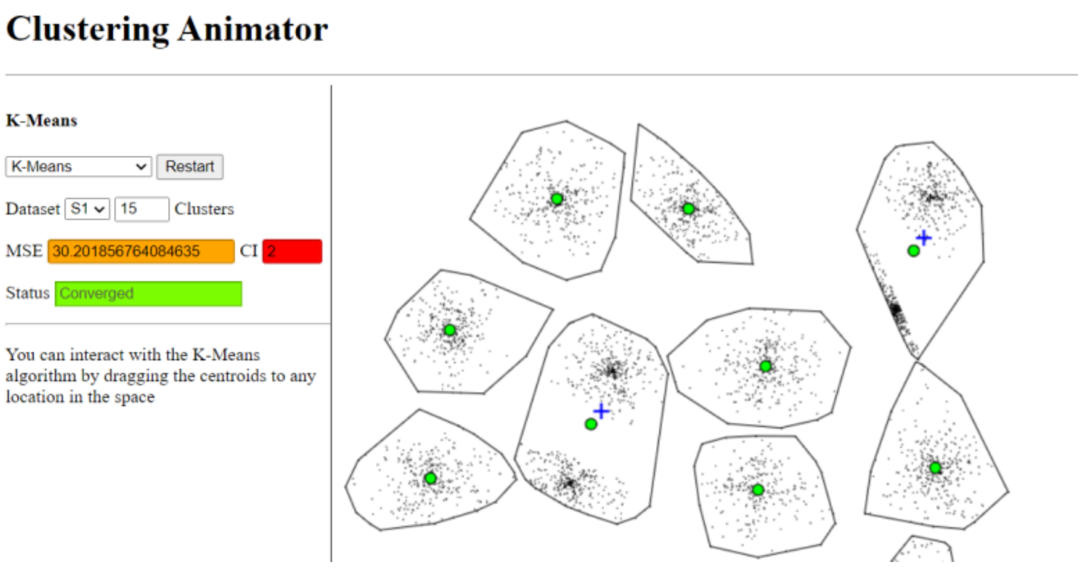
We created a clustering visualization tool for the web called clustering animator that shows us how various clustering algorithms work and lets us interact with it to understand and learn about the processing of those algorithms in a better way and to some extent also gives us the idea of where some algorithms perform better and where they lack.

To develop the clustering animator, we used *HTML5* canvas with *JavaScript*. The following procedure is followed for the functionality of the clustering visualizer (or clustering animator).

The first step in visualization is to show the dataset on the canvas so, we start by loading the dataset. Dataset is loaded and normalized (between 0-1 values) to make computations faster (Scaling techniques have been mentioned above). Then to show the data points on the canvas, values are rescaled to the size of the canvas and then plotted on the canvas. Rescaling data set according to the canvas size is required otherwise our plot will look like one single pixel. Scaling dataset does not only provide the ability to perform visualizations faster, but it also provides us the ability to adapt to the screen size as we rescale data when we plot it on the canvas.

After plotting the dataset on the canvas, we start our animation for the algorithm. Canvas is updated after each iteration of the algorithm, but to make the animation a little slower (so that the user can actually see points changing location) for the human eye to catch up we run each iteration after taking a slight moment break, for example, the canvas is updated after every half of a second.

Here is the basic overview of our animator:


Clustering Animator

K-Means

K-Means Restart

Dataset S1 15 Clusters

MSE 30.201856764084635 CI 2

Status Converged

You can interact with the K-Means algorithm by dragging the centroids to any location in the space

Figure 20. Clustering animator

In the figure 20, we have a menu on the left side for controls like selecting algorithm, dataset, the number of clusters *(k)*, SSE (sum of squared error), CI (centroid index), and state information of the animator (e.g., stopped processing algorithm, processing or restarting, etc.) and on the right side, we have our canvas where we can see our animation showing the process going on in the algorithm.

Animator Design

As visualization is the crucial part of an animator so we followed some UI-friendly concepts to make it easy to interact with. The clustering animator can only show k-means, random swap, weighted k-means, and gradual k-means algorithm at the time of writing this thesis. So, the visualization concepts that we will discuss will mostly be applicable to these algorithms for example, why we chose a certain way to display that specific thing and what were the possibilities to do it, and how it would have affected the user.

Visualization Concepts Followed

Let us discuss visualization methods, in our canvas we have:

1. Datapoints (in small black dots)
2. Centroids (as green circles)
3. Centroid Index (as plus (+) or minus (-) signs)
4. Clusters (Convex hull - a line polygon covering a cluster)

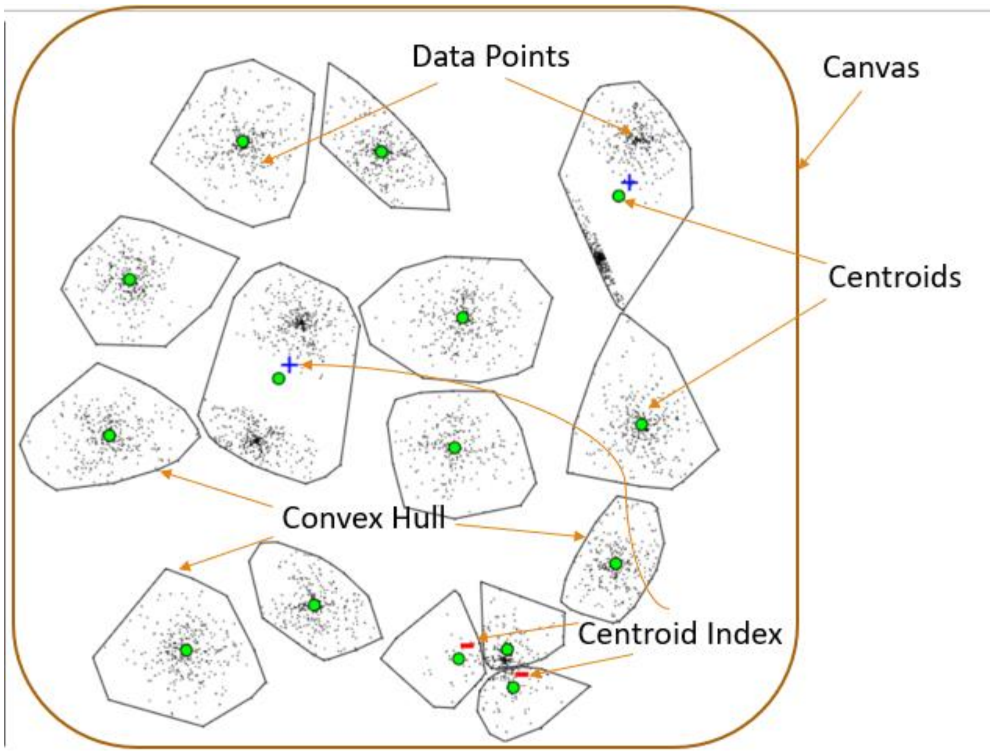


Figure 21. Animator Design

Datapoints

We used black dots as in figure 22 to represent our data points in the canvas, but we also have some other possibilities in terms of visualization. For example, black transparent dots can help indicate density in a region with many overlapping points. With transparency, a dense area will be darker than a region where data points are not overlapping each other so it will provide a better overview of data points.

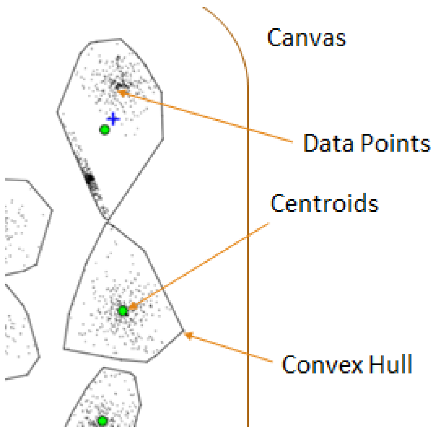


Figure 22. Datapoints

Centroids

Centroids are displayed as a green circle as shown in figure 23 occupying a slightly larger area than our data points.

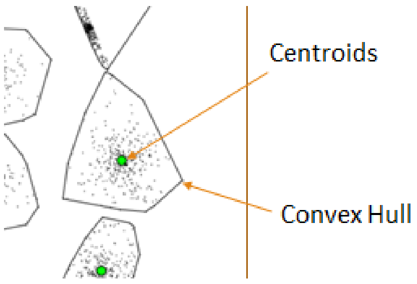


Figure 23. Centroids

On the web they look good but what if we print it on paper? As it is a light color so if we print it in gray ink on paper the centroids will have less intensity than the data points. A better way to represent that could be to have gray-colored data points and black-colored centroids so they stay more intense even if we print these on paper in black and white.

Centroid Index

Centroid index is the similarity measure between two solutions or in simple words, Centroid index tells us where we have more centroids (clusters) than needed as shown in figure 24 and where we need more clusters by comparing our solution with the ground truth values of that specific dataset.

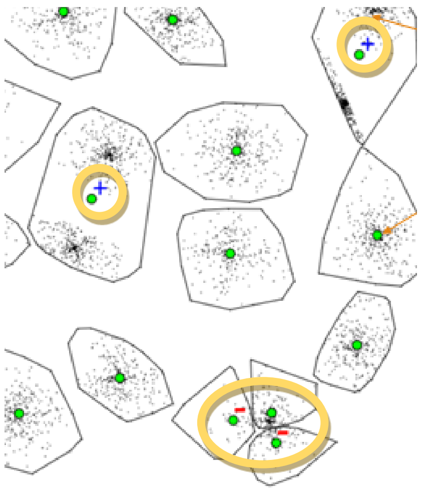


Figure 244. Centroid index

In the figure above, plus (+) signs in blue show that we could have more clusters in this region, and the minus (-) sign in red shows that we have excessive clusters in this region. What are the other possibilities to visualize these values in a better way? Maybe show the signs the other way around to convey the message, plus showing that we have more centroids here and minus showing here are fewer centroids than needed for an optimal solution.

The representation of clusters has been done using the convex hull. The convex hull is the smallest set of convex points that contains it as we discussed in detail earlier. In terms of visualization, there are many possibilities that could be applied for various reasons to represent different clusters such as:

1. Different colors per cluster
2. Different gray intensity per cluster
3. Convex hull

Different colors per cluster

The first problem is that different colors for each cluster could be used but for a huge set of data, it might not be a good choice because it creates difficulty to differentiate similar colors. Or our users might have the problem of color blindness which is difficult for them to differentiate colors. Another problem might be to differentiate clusters on a printed page in black and white as two totally different shades might have almost the same grayscale intensity and they might look like the same color in grayscale.

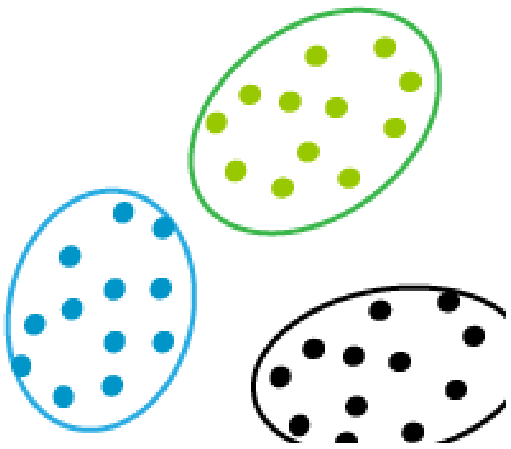


Figure 25. Different color for each cluster

Different gray intensity per cluster

Using different gray intensity per cluster is another choice to represent clusters and it would look good on both screen and paper but if we have a very big dataset with many clusters then it can get difficult to differentiate between different clusters as intensity difference will not be much.

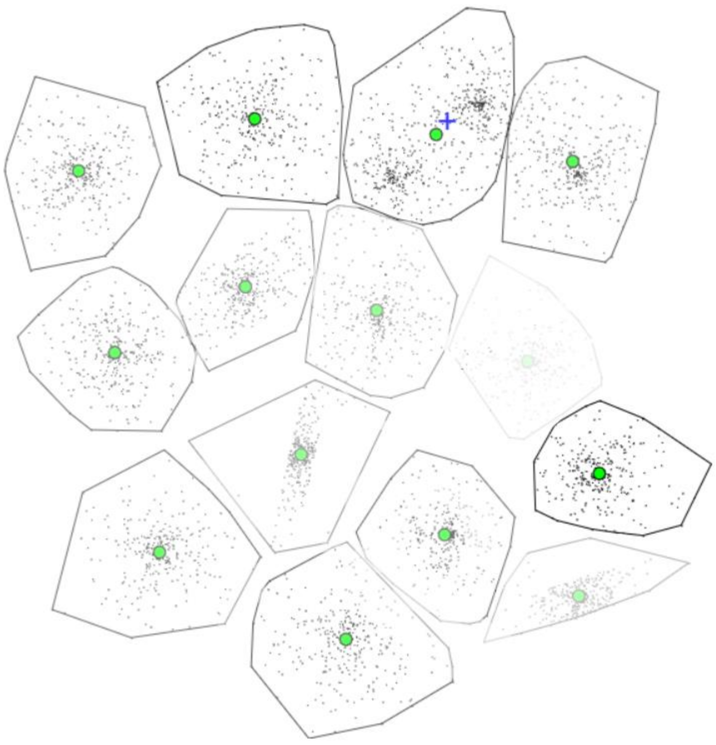


Figure 26. Different color intensity for each cluster

In the figure 26 every cluster has a significant change in intensity of color but for us (human eyes) there is not any difference in most of the clusters above. We can differentiate between them based on the polygon line wrapping each cluster.

Convex hull

The convex hull is a nice option to wrap up clusters for differentiating as shown in figure 27 and it would look nice where clustering methods produce spherically shaped clusters and do not overlap each other like in K-means.

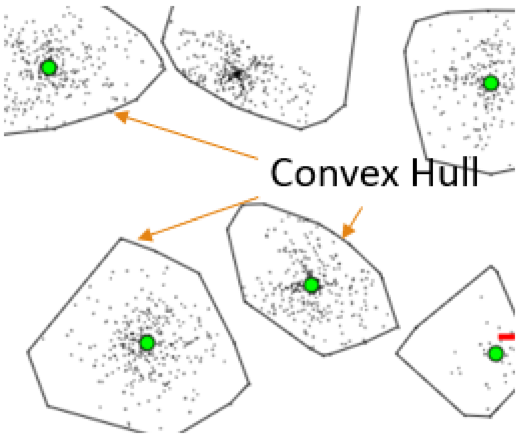


Figure 27. Convex hull

5 Analyzing Clustering Performance

The performance of the algorithm is the most important part of any algorithm that it is being designed for. Almost every algorithm has some limitations and issues so here we are using a clustering animator to monitor the clustering as it progresses and learn more about when they perform better and where they lack in performance. The objective is to interact with the algorithms through visualization and try to analyze their performance in different situations.

5.1 K-means

K-means algorithm has many advantages like:

- simple to understand and fast to cluster,
- scales to large datasets,
- always guarantees convergence,
- easily adapts to new examples.

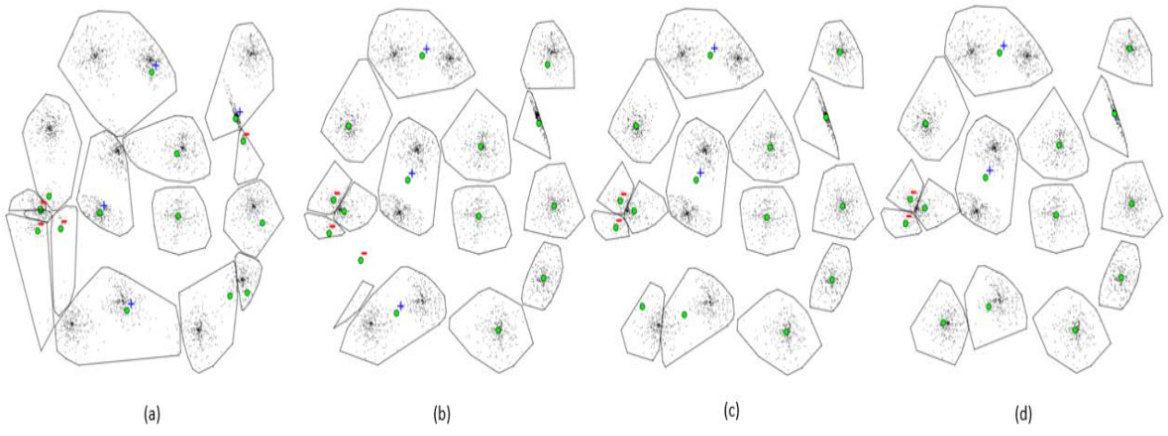


Figure 28. k-means process

Figure 28 has 4 frames captured from the animator while processing the k-means algorithm. As we see there is a major difference in frames (a) and (b). So, we can assume that how fast its clustering can be, so it can easily scale to large datasets. And another advantage is it always yields a result (but it may be a disadvantage as well because sometimes results can be deceiving).

At the same time, it also has some disadvantages like:

- need to select *k* (number of clusters),
- sensitive to outliers,
- sensitive to initialization,
- produces spherical solutions.

The *k* in k-means refers to the number of clusters and this value has to be given as input. Selecting *k* is a hot topic in research and there have been many methods introduced to overcome this problem like the elbow or knee method. The standard solution is to run clustering for several values of *k*, but using a cost function that allows comparing clustering results with different values of *k* (see figure 29).

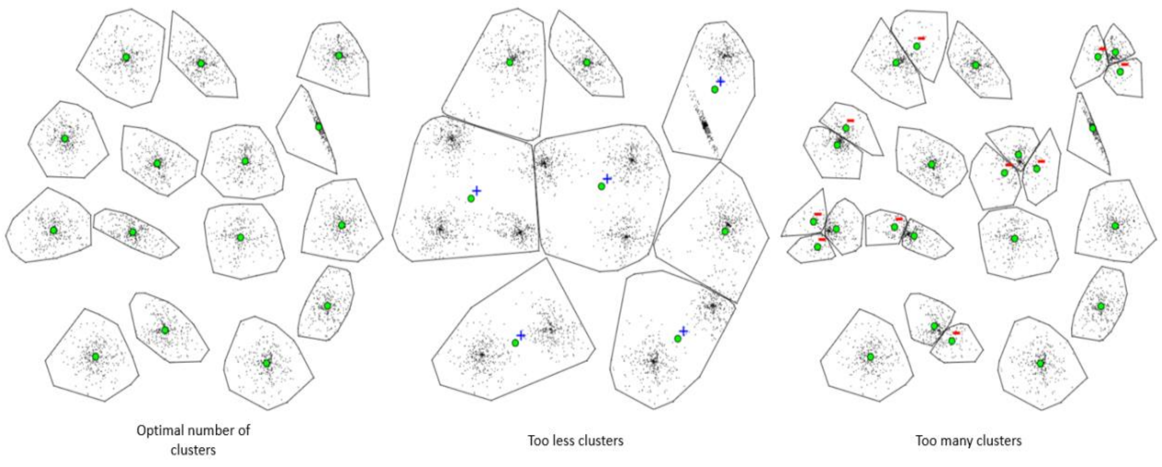


Figure 29. Number of clusters

K-means is also sensitive to outliers that means that if your dataset has one point far away from the rest of the data, then k-means will always try to include in a separate cluster or make it a part of another cluster (in the second case the centroid position of the cluster would change drastically just because of that anomaly).

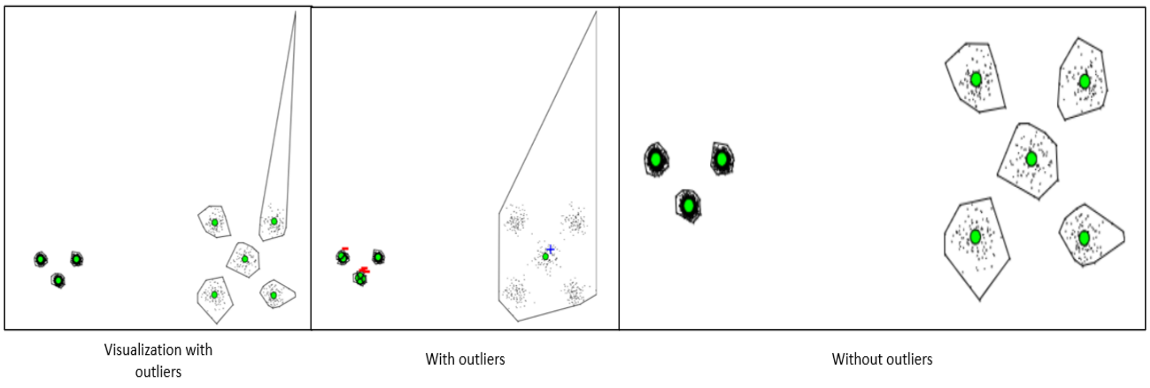


Figure 30. Sensitivity to outliers

Its sensitivity to initial seeds for centroids also plays a vital role in the clustering results. For example, if the initial positions for centroids were congested in one area but not in some parts then clustering results would not be good enough as they will get stuck in local optima. On the other hand, if initial positions for centroids were well diverse throughout the whole dataset, then it would produce good clustering results. A research paper "*How much k-means can be improved by using better initialization and repeats?*" by Fränti, P. and Sieranoja, S. (2019) provides and extensive study of different initialization methods and the effects.

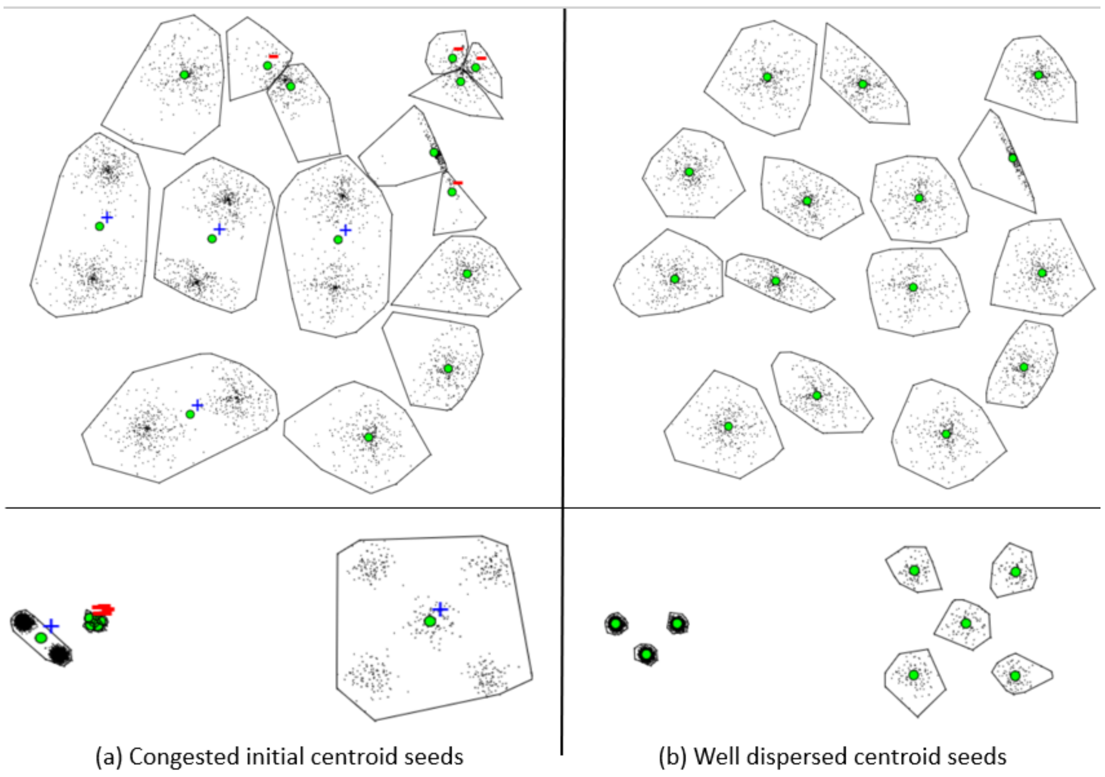

(a) Congested initial centroid seeds
(b) Well dispersed centroid seeds

Figure 31. Sensitivity to centroid initialization

K-means produces spherical solutions means that in a 2D plane it would produce clusters that would more look like circles rather than elliptic shapes. The reason is that we are using the Euclidean distance from the centroid. This is also why outliers are such a big issue for k-means. See example in figure 32.

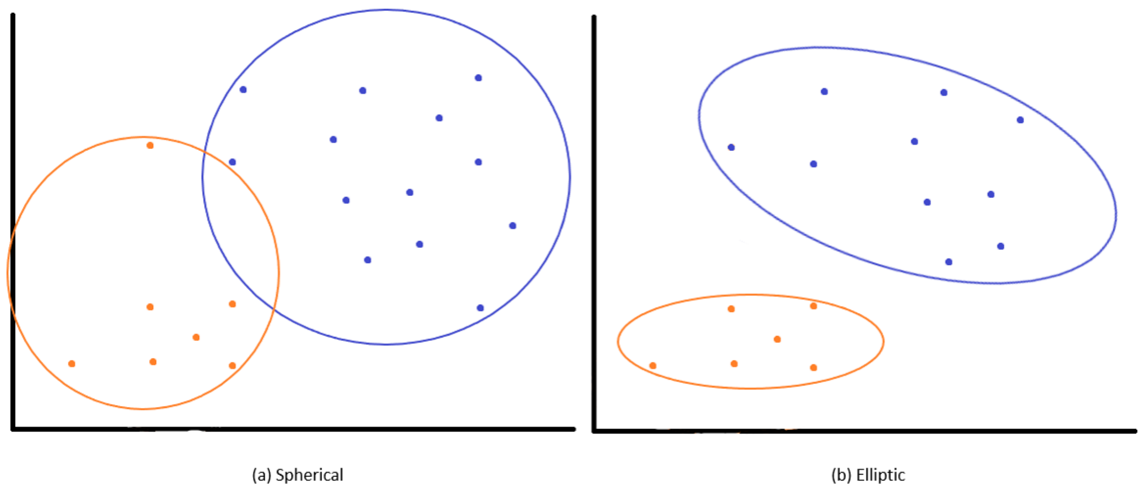


Figure 32. Spherical clusters

5.2 Random Swap

K-means algorithm is suitable for fine-tuning clusters boundaries, but it cannot solve clusters globally (Fränti 2018) whereas, random swap clustering is more efficient as compared to k-means algorithm as it aims at solving clustering by swapping centroids and by fine-tuning their exact location using k-means. Some of its advantages are:

- Easy to implement,
- More efficient as compared to K-means,
- If iterated longer it finds correct clustering,
- Almost never gets stuck in the local minimum.

The major drawbacks of random swap algorithm are:

- Their computational complexity,
- Lack of theoretical results of its properties,
- No precise rule like how the algorithm should be iterated,
- The parameter needs to be selected experimentally.

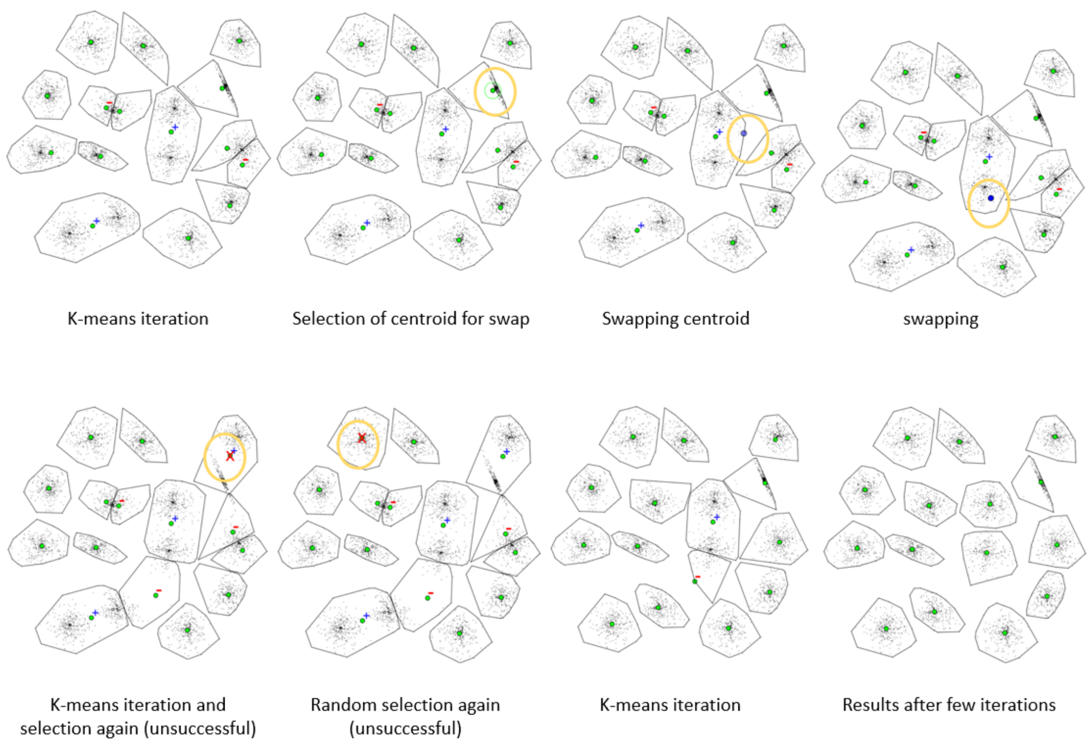

K-means iteration
Selection of centroid for swap
Swapping centroid
swapping
K-means iteration and selection again (unsuccessful)
Random selection again (unsuccessful)
K-means iteration
Results after few iterations

Figure 33. Random Swap iterative steps

The figure 33 shows few iterations of random swap clustering. And if we compare the k-means algorithm and random swap algorithm, it is evaluated that the random swap algorithm is straightforward and is a small extension when k-means can be implemented. The K-means algorithm has two steps, while the random swap algorithm has one additional step and that is prototype swap.

It is seen from the above diagram that in most cases, this additional step is independent of data and the objective function so trivial to implement, which makes it worthwhile for practical implementations of applications. And it also gives us the advantage over k-means as it overcomes the sensitivity to initialization problem that we had in k-means.

5.3 Gradual K-means

Gradual K-means is an alternative approach for clustering that fits the data into an optimal clustering model instead of fitting the cluster model to the dataset. The gradual K-means algorithm gives the following advantages in this study which are:

- Efficient evaluation of objective function avoids illegal string elimination,
- Convergence of global optimum,
- Calculate objective value like mean square error incrementally.

With all these advantages, there are observed some disadvantages with gradual k-means algorithm implementation, which are:

- Predicting the value of k is very difficult as in other clustering algorithms,
- There are some pathological cases where it fails,
- If initial partitions are different, then the final clusters will be different.

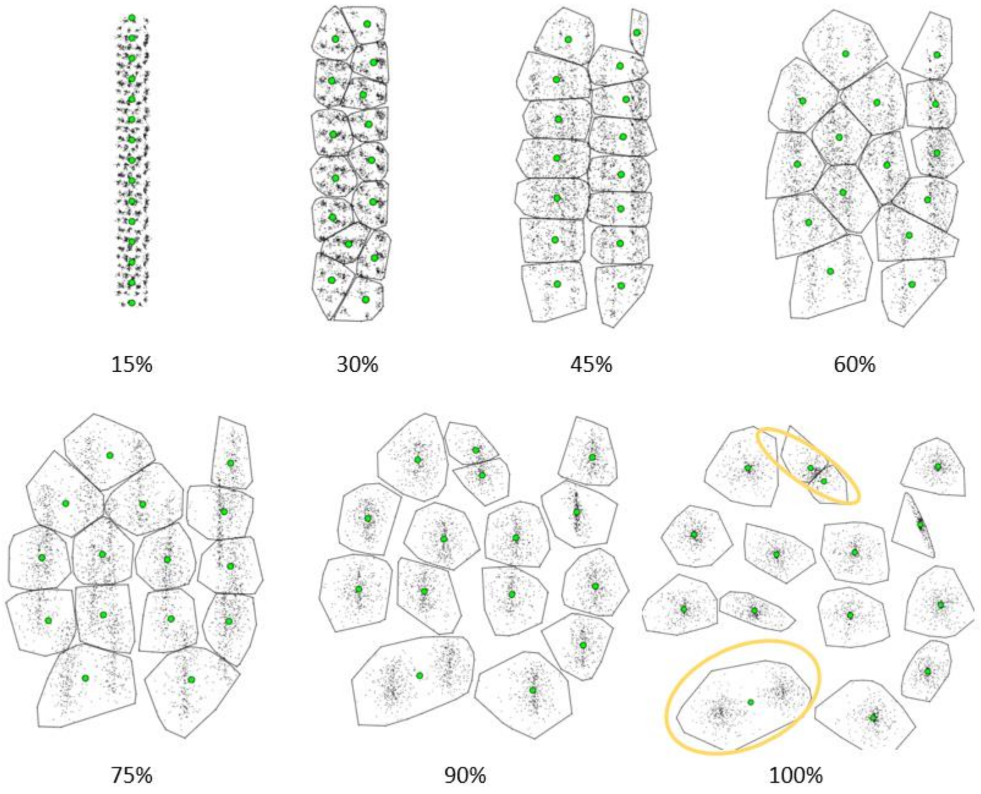


Figure 34. Example showing a complete cycle of gradual k-means.

It can be seen in the figure 34 that gradual k-means overcomes the locality problem of k-means really well and converges globally. But it does not provide optimal results every time in the last step of the diagram it has been pointed out that it failed to correctly allocate one centroid.

5.4 Weighted k-means

Weighted k-means is an extension to the k-means algorithm by adding a new step to assigning weights to in the iterative process that results in improved clustering scalability. Advantages that the weighted k-means algorithm provide are the following:

- It improves clustering scalability,

- Works well with nested and non-nested datasets,
- Initial weights for centroids are equal.

With all the advantages, there are some disadvantages observed while running the algorithm:

- Complexity increases when calculating the weights

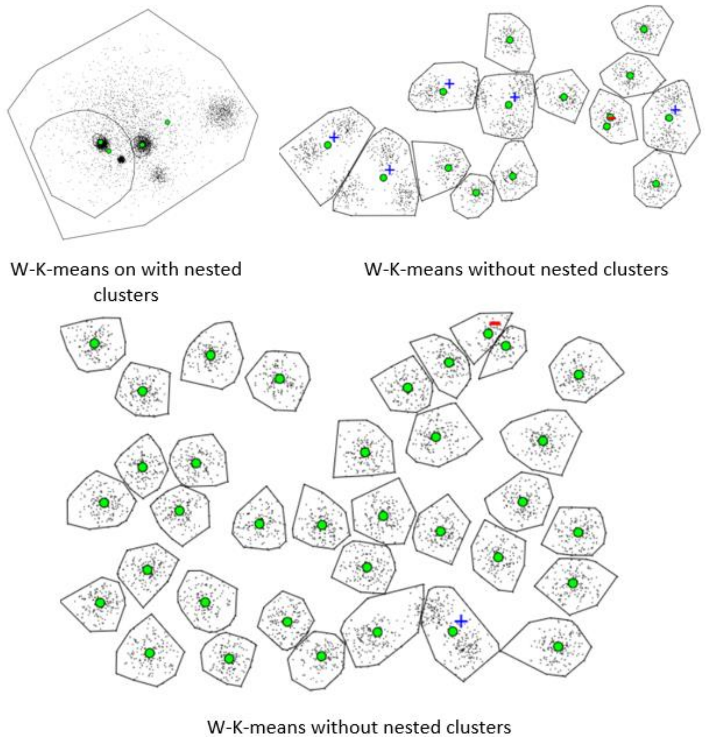


Figure 35. Example showing results of weighted k-means clustering on different datasets.

One of the potential drawbacks might be that it allocates equal weights to all the centroids at the start, so the first iteration of the algorithm works like normal k-means. The weighted k-means algorithm is also known as W-K-Means that calculates the weights automatically and then organizes the cluster according to its weights (Bhatia 2010).

6 Conclusion

In this study, a detailed explanation of the clustering algorithms, explanation of its components that helps to visualize clusters in a more better, a discussion on how some components can be visualized for better representation, and what are possible ways to visualize some components in different situations, and the performance of clustering algorithms is analyzed through visualization. The clustering algorithms included in this study are k-means, random swap, weighted k-means and gradual k-means algorithm. A web-based animator was developed to learn about these algorithms. Each algorithm is implemented individually so that particularities of each algorithm could be emphasized. Every clustering algorithm has its advantages as well as some disadvantages.

Visualization of algorithms can help us learn more about their working. This learning process becomes even more efficient by interacting with the algorithms as we are able to try some extreme cases that may not happen itself during a normal animation cycle.

7 References

Alshamrani, R., Alshehri, F., & Kurdi, H. (2020). A Preprocessing Technique for Fast Convex Hull Computation. *Procedia Computer Science*, *170*, 317-324.

Bouzos, O., Andreadis, I., & Mitianoudis, N. (2019). Conditional random field model for robust multi-focus image fusion. *IEEE Transactions on Image Processing*, *28*(11), 5636-5648.

Cheng, D., Zhu, Q., Huang, J., Wu, Q., & Yang, L. (2018). A novel cluster validity index based on local cores. *IEEE transactions on neural networks and learning systems*, *30*(4), 985-999.

Cho, S. M., Hong, E., & Seo, S. H. (2020). Random Number Generator Using Sensors for Drone. *IEEE Access*, *8*, 30343-30354.

Fränti, P. (2018). Efficiency of random swap clustering. *Journal of Big Data*, *5*(1), 13.

Fränti, P., & Sieranoja, S. (2018, June). Dimensionally distributed density estimation. In *International Conference on Artificial Intelligence and Soft Computing* (pp. 343-353). Springer, Cham.

Fränti, P., Virmajoki, O., & Hautamaki, V. (2008, December). Probabilistic clustering by random swap algorithm. In *2008 19th International Conference on Pattern Recognition* (pp. 1-4). IEEE.

Goceri, E., & Dura, E. (2018). Comparison of weighted k-means clustering approaches. In *Proc. Int. Conf. Math*.

Goel, S., & Tushir, M. (2020). A new iterative fuzzy clustering approach for incomplete data. *Journal of Statistics and Management Systems*, *23*(1), 91-102.

Guérin, J., Gibaru, O., Thiery, S., & Nyiri, E. (2017). Clustering for different scales of measurement-the gap-ratio weighted k-means algorithm. *arXiv preprint arXiv:1703.07625*.

Irawan, Y. (2019). Implementation Of Data Mining For Determining Majors Using K-Means Algorithm In Students Of SMA Negeri 1 Pangkalan Kerinci. *Journal of Applied Engineering and Technological Science (JAETS)*, *1*(1), 17-29.

Kang, J. M. (2008). Voronoi Diagram.

Kassambara, A. (2017). *Practical guide to cluster analysis in R: Unsupervised machine learning* (Vol. 1). Sthda.

Kerdprasop, K., Kerdprasop, N., & Sattayatham, P. (2005, August). Weighted k-means for density-biased clustering. In *International Conference on Data Warehousing and Knowledge Discovery* (pp. 488-497). Springer, Berlin, Heidelberg.

Khanmohammadi, S., Adibeig, N., & Shanehbandy, S. (2017). An improved overlapping k-means clustering method for medical applications. *Expert Systems with Applications*, *67*, 12-18.

Liu, H., Wu, J., Liu, T., Tao, D., & Fu, Y. (2017). Spectral ensemble clustering via weighted k-means: Theoretical and practical evidence. *IEEE transactions on knowledge and data engineering*, *29*(5), 1129-1143.

Lukjancenko, O., Wassenaar, T. M., & Ussery, D. W. (2010). Comparison of 61 sequenced Escherichia coli genomes. *Microbial ecology*, *60*(4), 708-720.

Ma, K., Hao, P., Zhang, H., & Yang, C. (2020, January). A strip steel wave-edge detection algorithm based on convex hull detection. In *Eleventh International Conference on Graphics and Image Processing (ICGIP 2019)* (Vol. 11373, p. 113730N). International Society for Optics and Photonics.

Malinen, M. I., Mariescu-Istodor, R., & Fränti, P. (2014). K-means$*$: Clustering by gradual data transformation. *Pattern Recognition*, *47*(10), 3376-3386.

Murtagh, F., & Contreras, P. (2017). Clustering through high dimensional data scaling: applications and implementations. *Archives of Data Science, Series A (Online First)*, *2*(1), 16.

O'Hagan, A., Murphy, T. B., Scrucca, L., & Gormley, I. C. (2019). Investigation of parameter uncertainty in clustering using a Gaussian mixture model via jackknife, bootstrap, and weighted likelihood bootstrap. *Computational Statistics*, *34*(4), 1779-1813.

Pal, R., & Saraswat, M. (2017, August). Data clustering using enhanced biogeography-based optimization. In *2017 Tenth International Conference on Contemporary Computing (IC3)* (pp. 1-6). IEEE.

Pham, D. T., Dimov, S. S., & Nguyen, C. D. (2004). An incremental K-means algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *218*(7), 783-795.

Robinson, D. J., Sambridge, M., Snieder, R., & Hauser, J. (2013). Relocating a cluster of earthquakes using a single seismic station. *Bulletin of the Seismological Society of America*, *103*(6), 3057-3072.

Roskos-Ewoldsen, D. R., & Roskos-Ewoldsen, B. (2008). Scaling and cluster analysis. *The Sage sourcebook of advanced data analysis methods for communication research*, 275-310.

Sane, R. (2020). *Using densities to detect nested clusters* (Master's thesis, University of Eastern Finland).

Shah, S. A., & Koltun, V. (2017). Robust continuous clustering. *Proceedings of the National Academy of Sciences*, *114*(37), 9814-9819.

Teklehaymanot, F. K., Muma, M., & Zoubir, A. M. (2018). Bayesian cluster enumeration criterion for unsupervised learning. *IEEE Transactions on Signal Processing*, *66*(20), 5392-5406.

Vattani, A. (2009). The hardness of k-means clustering in the plane. Manuscript, accessible at http://cseweb.Ucsd.edu/avattani/papers/kmeans_hardness.pdf, 617.

Wang, Z., Xu, A., Zhang, Z., Wang, C., Liu, A., & Hu, X. (2020, August). The parallelization and optimization of K-means algorithm based on spark. In *2020 15th International Conference on Computer Science & Education (ICCSE)* (pp. 457-462). IEEE.

Wu, O., Hu, W., Maybank, S. J., Zhu, M., & Li, B. (2012). Efficient clustering aggregation based on data fragments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*(3), 913-926.

Yang, J., Rahardja, S., & Fränti, P. (2021). Mean-shift outlier detection and filtering. *Pattern Recognition*, *115*, 107874.

Zhang, G., Zhang, C., & Zhang, H. (2018). Improved K-means algorithm based on density Canopy. *Knowledge-based systems*, *145*, 289-297.

Zhang, Z., Yu, W., Martínez, L., & Gao, Y. (2019). Managing multigranular unbalanced hesitant fuzzy linguistic information in multiattribute large-scale group decision making: A linguistic distribution-based approach. *IEEE Transactions on Fuzzy Systems*, *28*(11), 2875-2889.

Zhao, Q., & Fränti, P. (2014). WB-index: A sum-of-squares based index for cluster validity. *Data & Knowledge Engineering*, *92*, 77-89.

Zhao, Q., Hautamäki, V., Kärkkäinen, I., & Fränti, P. (2012). Random swap EM algorithm for Gaussian mixture models. *Pattern Recognition Letters*, *33*(16), 2120-2126.