Universitat Oberta de Catalunya

uoc.edu

Fuzzy C-means and clustering algorithms: a comparative study

Victor Garcia Domingo
Grau en Enginyeria Informàtica
Intel·ligència Artificial

Dr. Joan M. Nuñez Do Rio
Dr. Carles Ventura Royo

4 juny 2019

FITXA DEL TREBALL FINAL

Títol del treball:	*Fuzzy C-means and clustering algorithms: a comparative study*
Nom de l'autor:	*Victor Garcia Domingo*
Nom del consultor/a:	*Joan M. Nuñez Do Rio*
Nom del PRA:	*Carles Ventura Royo*
Data de lliurament (mm/aaaa):	*06/2019*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Intel·ligència Articial*
Idioma del treball:	*Anglès*
Paraules clau	*Clustering, Fuzzy C-Means, overlapping clusters*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

La clusterització de dades és una tècnica que agrupa les observacions d'un conjunt de dades en funció de la distància al centre dels clústers. Un dels primers algoritmes de clusterització va ser el K-Means (KM), que és especialment acurat per a reconèixer grups de clústers separats. El Fuzzy C-Means (FCM) va ser formulat per a millorar la precisió del KM amb clústers superposats. S'han desenvolupat altres algorismes derivats del FCM per a millorar-lo: el Gustafson Kessel Fuzzy C-Means (GKFCM), per a clústers no esfèrics, el Fuzzy C-Means++ (FCM++) i el Suppressed Fuzzy C-Means (S-FCM), més eficients, i el Possibilistic C-Means (PCM), més precís per a observacions atípiques. En aquest projecte, he comparat el KM, el FCM, el GKFCM, el FCM++, el S-FCM i el PCM i les millores dels nous respecte als predecessors, centrant-ho al voltant del FCM. He validat paràmetres com l'eficiència computacional, el rendiment i la precisió. He trobat que, d'entre tots els algorismes, el FCM té el millor rendiment per a conjunts de dades amb clústers superposats i el KM és l'algorisme més eficient. El GKFCM funciona bé amb clústers no esfèrics, però no és del tot precís. Finalment, el PCM no ha mostrat cap avantatge respecte al FCM. Aquest projecte és un punt de partida per a futures investigacions sobre els algorismes de clusterització, ja que la majoria dels conjunts de dades utilitzats aquí són conjunts de dades sintètics, basats en característiques ideals. i s'espera que els conjunts de dades reals tinguin estructures més complexes.

Abstract (in English, 250 words or less):

Clustering is a technique that groups observations in a dataset based on the distance to the centre of the clusters. One of the first clustering algorithms was K-Means (KM), which is especially accurate at recognising well-separated clusters. Afterwards, Fuzzy C-Means (FCM) was formulated to improve the accuracy of KM with datasets containing overlapping clusters. Since then, other derivatives of FCM have been developed to improve it: Gustafson Kessel Fuzzy C-Means (GKFCM) performs better for non-spherical clusters, Fuzzy C-Means++ (FCM++) and Suppressed-Fuzzy C-Means (S-FCM) improve FCM's efficiency and Possibilistic C-Means (PCM) is more accurate for datasets with noise and outliers. In this project, I have compared KM, FCM, GKFCM, FCM++, S-FCM and PCM to check how each evolution has improved its predecessor. This comparison is centralised around FCM. I have validated parameters such as computational efficiency, performance and accuracy. I have found that, among all the algorithms, FCM has the best performance for datasets with overlapping clusters, even though S-FCM improves its computational efficiency. Also, KM is the most efficient algorithm and GKFCM performs well with non-spherical clusters. However, it is less accurate. Finally, PCM has not shown any advantage over FCM. This project is a starter point for future investigations of the conditions under which every algorithm works better. Most of the datasets used here are synthetic datasets, based on near-ideal characteristics. Nevertheless, real-world datasets are expected to have more complex structures for which the choice of algorithms require a more thorough investigation.

Acknowledgements

I would like to express my infinite gratitude to my family, Llucià, Dolors and Eric, for their unconditional support throughout all these years of hard work and difficulties.

I would also like to express my warm thanks to Jalpa for her motivation and patience.

Finally, my sincere thanks to Dr. Joan M. Nuñez Do Rio for the fantastic supervising job he has done.

It is because of all of them that this project exists.

Index

1 Introduction 7
1.1 Context and justification 7
1.2 Objectives 10
1.3 Method and Approach 11
1.4 Planification 13
1.5 Short summary of the products 14
1.6 Short description of the rest of the chapters 14
2 Algorithms 15
2.1 Introduction 15
2.2 Algorithms 15
2.2.1 K-Means 15
2.2.2 Fuzzy C-Means 17
2.2.3 Fuzzy C-Means++ 19
2.2.4 Suppressed-Fuzzy C-Means 21
2.2.5 Gustafson-Kessel Fuzzy C-Means 23
2.2.6 Possibilistic C-Means 25
3 Datasets and validation methods 27
3.1 Introduction 27
3.2 Datasets 27
3.2.1 Synthetic dataset 1 (SD1) 27
3.2.2 Synthetic dataset 2 (SD2) 28
3.2.3 Synthetic dataset 3 (SD3) 29
3.2.4 Synthetic dataset 4 (SD4) 30
3.2.5 Synthetic dataset 5 (SD5) 30
3.2.6 Synthetic dataset 6 (SD6) 31
3.2.7 Synthetic dataset 7 (SD7) 32
3.2.8 Synthetic dataset 8 (SD8) 33
3.2.9 Synthetic dataset 9 (SD9) 33
3.2.10 Synthetic dataset 10 (SD10) 34
3.2.11 Real-world dataset 1 (RWD1) 35
3.3 Validation methods 36
3.3.1 Internal methods 36
3.3.2 External methods 37
4 Experiments 38
4.1 Introduction 38
4.2 Groups of experiments 39
4.2.1 K-Means and Fuzzy C-Means 39
4.2.2 Fuzzy C-Means, Fuzzy C-Means ++ and Supressed-Fuzzy C-Means 41
4.2.3 Fuzzy C-Means and Gustafson-Kessel Fuzzy C-Means 46
4.2.4 Fuzzy C-Means and Possibilistic C-Means 55
4.3 Discussion 61
5 Conclusions 63
6 Bibliography 65

List of figures

Figure 1. SD1 28
Figure 2. SD2 29
Figure 3. SD3 29
Figure 4. SD4 30
Figure 5. SD5 31
Figure 6. SD6 32
Figure 7. SD7 32
Figure 8. SD8 33
Figure 9. SD9 34
Figure 10. SD10 35
Figure 11. RWD1 36
Figure 12. Efficiency of FCM, FCM++ and S-FCM for SD1 42
Figure 13. Performance (Xie-Beni) of FCM, FCM++ and S-FCM for SD1 42
Figure 14. Performance (Silhouette) of FCM, FCM++ and S-FCM for SD1 43
Figure 15. Efficiency of FCM, FCM++ and S-FCM for RWD1 44
Figure 16. Performance (Xie-Beni) of FCM, FCM++ and S-FCM for RWD1 44
Figure 17. Performance (Silhouette) of FCM, FCM++ and S-FCM for RWD1 45
Figure 18. Efficiency of FCM, and GKFCM for SD3 47
Figure 19. Performance (Xie-Beni) of FCM, and GKFCM for SD3 47
Figure 20. Performance (Silhouette) of FCM and GKFCM for SD3 48
Figure 21. Clusters of SD3 recognised by FCM 48
Figure 22. Clusters of SD3 recognised by GKFCM 48
Figure 23. Clusters of SD7 recognised by GKFCM 49
Figure 24. Clusters of SD7 recognised by FCM 50
Figure 25. Clusters of SD8 recognised by GKFCM 51
Figure 26. Clusters of SD8 recognised by FCM 52
Figure 27. Clusters of SD9 recognised by GKFCM 53
Figure 28. Clusters of SD9 recognised by FCM 53
Figure 29. Clusters of SD10 recognised by FCM 54
Figure 30. Clusters of SD10 recognised by GKFCM 54
Figure 31. Clusters of SD4 recognised by PCM 55
Figure 32. Clusters of SD4 recognised by FCM 56
Figure 33. Original classes in SD4 vs PCM and FCM results 56
Figure 34. Clusters of SD5 recognised by PCM 57
Figure 35. Clusters of SD5 recognised by FCM 57
Figure 36. Clusters of SD6 recognised by FCM 58
Figure 37. Clusters of SD6 recognised by PCM 58
Figure 38. Clusters of SD9 recognised by FCM 59
Figure 39. Clusters of SD9 recognised by PCM 59
Figure 40. Clusters of SD10 recognised by FCM 60
Figure 41. Clusters of SD10 recognised by PCM 60

1 Introduction

1.1 Context and justification

Clustering[1] was originally formulated by Linnaeus, a Swedish physician and botanist from the eighteenth century. He provided a hierarchical structure that helped to understand the roles and interactions of different botanical species [1]. Since its development, the concept of clustering has become a common technique for statistical analysis and classification of data. It is at the heart of exploratory data mining and has been extensively used in many fields, such as image analysis, pattern recognition, information retrieval, bioinformatics, data compression, computer graphics, and most recently in machine learning and artificial intelligence.

Clustering is not defined by one specific algorithm and therefore it is not an automatic task, but it is a method of information extraction by iterative process making it a very robust technique for statistical analysis. More specifically, clustering is a method that aims at “classifying observations from a dataset based on the notions of similarity, distance, or indistinguishability” [1]. The goal of clustering analysis is to find clusters in a dataset by grouping similar observations in the same group based on a distance function. There are different distance measures, such as the Euclidian, the Manhattan or the Chebyshev, depending on the dataset of interest. It has been shown by several studies that the Euclidian distance is a fitting choice for most datasets.

As mentioned above, clustering cannot be defined by one specific algorithm, which is one of the reasons for the development of multiple algorithms for clustering. For a group of datasets, there can be different cluster models, and furthermore, many different algorithms have been developed for each of the cluster models. Therefore, there are different types of clustering algorithms based on different ways to classify them. Among the existing clustering algorithms, there are two important classification criteria to distinguish between them. One of the criteria is to distinguish between “hierarchical clusters, non-hierarchical (flat) or mixture techniques” [2]. The other criterion distinguishes between hard clustering and soft clustering.

The fact that there are possibly over 100 clustering algorithms in existence indicates that there is no objectively “correct” clustering algorithm, rather the choice of an algorithm depends on the dataset in question. This choice is often

[1] Linnaeus referred to this method as classification. However, I will refer to it as clustering, since classification is nowadays a different technique.

based on experimental selection unless there is a mathematical reason to prefer one algorithm over another.

Soft clustering, especially fuzzy clustering, has several advantages compared to hard clustering. For example, membership restriction of each observation in hard clustering is quite unrealistic [11] for real-world datasets, and it would be more natural to assign a degree of membership to each observation. Also, fuzzy clustering categorises observation-outliers more accurately than hard clustering, because of the degree of membership assigned to each of the observations. This means that, for example, if an observation does not clearly belong to one cluster or another, a hard-clustering algorithm is more likely to mis-categorise it, as it must choose only one of the clusters. On the other hand, a fuzzy algorithm will be more precise in categorising such observations because the assigned degree of membership will exactly identify its parent cluster, even if by a small percentage of difference.

In the 1960's and 1970's, cluster analysis became a big topic in statistics, data analysis and applications when Sokal and Sneath published the monograph 'Principles of numerical taxonomy' in 1963. [4] One of the first clustering algorithms, the K-Means, is based on the sum-of-squares criterion. This criterion was first formulated by Dalenius (1950) and Dalenius and Gurney (1951) to estimate "the expectation $\mu = [E]$ of a real-valued random variable *X* with distribution density *f(x)*." [4] However, the first who formulated the K-Means algorithm was Steinhaus in 1956, then Forgy proposed it for clustering data in 1965 and the name K-Means was used for the first time by MacQueen in 1967. Since then, the algorithm "became a standard procedure in clustering". [4]

Even though K-Means performs well on datasets with well-separated clusters, it is not very accurate with overlapping ones. [2] This is because K-Means works with the notion of hard sets, which is not appropriate for overlapping clusters. The concept of fuzzy sets was first introduced by Zadeh in 1965, as an extension of the notion of hard sets. This laid the foundation for later works in fuzzy clustering. In 1969, Ruspini generalized the fuzzy partition of Zadeh, based on which Dunn addressed the problem of K-Means in 1973. However, Dunn's idea was an extension of K-Means and not a completely new algorithm. It was Bezdek who connected K-Means and Dunn's generalization in the same year to develop the Fuzzy C-Means (FCM) algorithm. [1]

Fuzzy C-Means works well with overlapping clusters, but it forces all the clusters in a dataset to have approximately the same shape. In 1978, Gustafson and Kessel addressed this problem with a new algorithm, the Gustafson Kessel Fuzzy C-Means (GKFCM), that adapted to the shape of each cluster in a dataset. [1] GKFCM is also important because it was the first algorithm derived from Fuzzy C-Means which tried to improve some of its defects. Since then,

several other modifications have been applied to address other limitations of the original FCM. Some of the latest ones are Fuzzy C-Means++ (FCM++) and Suppressed-Fuzzy C-Means (S-FCM).

In 1993, Krishnapuram and Keller introduced a new algorithm, the Possibilistic C-Means (PCM). This algorithm was based on a different mathematical concept than FCM, which is based on the probabilistic idea that each observation belongs to all the clusters in some degree. In FCM, the sum of all these degrees of belongingness must be equal to 1, so even the outliers can be part of a cluster even if it is obvious that they do not. The theory behind the PCM, on the contrary, does not have this constraint. Instead, the PCM measures “how close the point is to each cluster prototype”. [1] Thus, for example, an outlier may not belong to any of the clusters, whereas a point in a vector may belong to more than one cluster along this vector in the same degree. This gives an advantage to PCM for datasets with noise and outliers.

In this project, I will compare some of the most important clustering algorithms that have been developed so far. This is important because not all algorithms are best for all kinds of datasets. Therefore, I want to check in which situations the choice of an algorithm is more appropriate over another.

1.2 Objectives

The main objective of this project is to validate how every new algorithm have improved the previous one by comparing their computational efficiency, performance and accuracy on different types of datasets.

I will also show the advantages and disadvantages of fuzzy clustering compared with hard clustering. Also, within soft clustering algorithms, I will investigate which ones perform better under which conditions.

First, I want to perform a series of experiments to validate that Fuzzy C-Means is more accurate than K-Means when there are overlapping clusters in a dataset. These experiments will also validate that K-Means is generally more efficient than Fuzzy C-Means, since the convergence speed is much slower for the latter. [2]

Second, I will compare FCM with FCM++ and S-FCM. I want to conduct a series of experiments that will validate whether both FCM++ and S-FCM are more computationally efficient and perform better than FCM.

Third, I will compare FCM with GKFCM to validate that the latter adapts to the shape of each cluster and thus performs better than FCM with differently shaped clusters.

Finally, I will perform a series of experiments to validate that PCM performs better than FCM with differently shaped clusters and outliers.

This project aims at being a starting point for further investigations about which conditions make one algorithm a better choice.

1.3 Method and Approach

I intend to combine internal and external validation methods “to validate the goodness of partitions after clustering”. [18]

Internal validation is a way to measure the goodness of partitions, relying only on information present in the data. It can be used either to choose the best clustering algorithm or to find the optimal cluster number [23]. Internal validation can be performed through indexes. Even though there some research studies about mixed indexes, these are still not very stable, Thus, these indexes are used to compare only hard clustering methods or only soft clustering methods. Among the most used soft-clustering indexes are Xie-Beni and Silhouette. Computational time through number of iterations is another internal validation method.

On the contrary, external validation is a way to measure the goodness of partitions relying on information external to the data. To use external validation methods, it is important to know in advance the correct class labels. Some of the external validation methods are accuracy, entropy or purity. [25] External validation is a good validation method when the aim is to compare algorithms hard and soft clustering algorithms, since, as explained above, these algorithms cannot be compared with indexing methods. [22]

I will perform different groups of experiments. In the first group of experiments, I will perform internal and external validation of K-Means and Fuzzy C-Means on two datasets, one with overlapping clusters and another with both overlapping and well-separated clusters. I will specifically validate the accuracy and the computational efficiency of both algorithms. This will allow me to check whether Fuzzy C-Means is more accurate than K-Means for overlapping clusters.

In the second group of experiments, I will perform internal validation of Fuzzy C-Means, Fuzzy C-Means++ and Suppressed-Fuzzy C-Means on two datasets. One of the datasets contains overlapping clusters and the other one both overlapping and well-separated clusters. I will validate the performance through the Xie-Beni and Silhouette indexes and the efficiency of each clustering method. This will allow me to check whether Fuzzy C-Means++ and Suppressed-Fuzzy C-Means are more computationally efficient than Fuzzy C-Means.

The third group of experiments will consist in the internal validation of Fuzzy C-Means and Gustafson Kessel Fuzzy C-Means on five different datasets. Three of the datasets contain non-spherical clusters, one contains spherical clusters, and the last one has both spherical clusters. I will validate the performance of each algorithm on each dataset through the Xie-Beni and the Silhouette indexes. I will also validate their efficiency. This will allow me to check whether

the Gustafson Kessel Fuzzy C-Means performs better than Fuzzy C-Means on datasets with different-shaped clusters.

Finally, in the fourth group of experiments I will compare Fuzzy C-Means and Possibilistic C-Means through internal and external validation on five different datasets. Four of the datasets contain spherical clusters with outliers and different levels of overlapping and one of the datasets contain both spherical and non-spherical clusters. I will validate both algorithms by comparing their accuracy and efficiency. This will allow me to check whether Possibilistic C-Means is more accurate than Fuzzy C-Means on datasets with outliers.

1.4 Planification

Associated tasks			Timeline											
			March				April				May			
Phase 1	Comparing K-Means with Fuzzy C-Means and its derivative algorithms													
Hypothesis 1														
A 1.1	K-Means algorithm													
	SD1	datasets without noise and clear clusters												
	SD2	datasets without noise and unclear clusters												
	RWD1	Real-world dataset												
A 1.2	Fuzzy C-Means algorithm													
	SD1	datasets without noise and clear clusters												
	SD2	datasets without noise and unclear clusters												
	RWD1	Real-world dataset												
Hypothesis 2														
A 2.1	K-Means algorithm													
	SD1	datasets without noise and clear clusters												
	SD2	datasets without noise and unclear clusters												
	RWD1	Real-world dataset												
A 2.2	Fuzzy C-Means algorithm													
	SD1	datasets without noise and clear clusters												
	SD2	datasets without noise and unclear clusters												
	RWD1	Real-world dataset												
Hypothesis 3														
A 3.1	Fuzzy C-Means algorithm													
	SD1	datasets without noise and clear clusters												
	RWD1	Real-world dataset												
A 3.2	Fuzzy C-Means++ algorithm													
	SD1	datasets without noise and clear clusters												
	RWD1	Real-world dataset												
A 3.3	Suppressed Fuzzy C-Means algorithm													
	SD1	datasets without noise and clear clusters												
	RWD1	Real-world dataset												
Phase 2	Investigating derivatives of Fuzzy C-Means algorithms for possible improvements													
Hypothesis 4														
A 2.2	Fuzzy C-Means algorithm													
	SD3, SD7, SD8	datasets with non-spherical clusters												
	SD9	datasets with spherical clusters												
	SD10	dataset with spherical and non-spherical clusters												
A 2.4	Gustafson and Kessel fuzzy clustering algorithm													
	SD3, SD7, SD8	datasets with non-spherical clusters												
	SD9	datasets with spherical clusters												
	SD10	dataset with spherical and non-spherical clusters												
Hypothesis 5														
A 2.2	Fuzzy C-Means algorithm													
	SD4, SD5, SD6, SD9	datasets with spherical clusters												
	SD10	dataset with spherical and non-spherical clusters												
A 2.1	Possibilistic C-Means algorithm													
	SD4, SD5, SD6, SD9	datasets with spherical clusters												
	SD10	dataset with spherical and non-spherical clusters												
Project Monitoring														
Solving unexpected problems														
Reviewing analysis														

1.5 Short summary of the products

This project aims at being a starting point to understand the evolution of clustering, the main algorithms and under which conditions it is better to use one or another.

1.6 Short description of the rest of the chapters

In Chapter 2, I introduce each one of the six algorithms and I explain how to computationally compute them.

In Chapter 3, I introduce each one of the eleven datasets. I also introduce the internal and external validation methods that I will use.

In Chapter 4, the results and the discussion of four groups of experiments are presented.

Finally, Chapter 5 contains the general conclusions of the project.

2 Algorithms

2.1 Introduction

In this chapter, I will introduce with more detail the six algorithms that I will later compare. For each algorithm, I will develop a short introduction and then I will explain how they work in technical details.

2.2 Algorithms

2.2.1 K-Means

The most famous algorithm in hard clustering (HC) is K-Means. It was first explicitly proposed by Steinhaus [12], although the name K-Means was first used by MacQueen [4, 5].

K-means algorithm works the following way. “It iteratively computes cluster centroids for each distance measure to minimize the sum with respect to the specified measure” [2]. The goal is to minimize this objective function, that is, “the sum of the squared error over all K clusters”, [6] as seen in the equation (1):

$$J_{KM}(C_k) = \sum_i^C \sum_j^n D_{ij}^2 \quad (1)$$

Where C is the cluster, n is the observation and D_{ij}^2 is the Euclidean distance, which is the one used in most of cases because of its good results.

The algorithm has the following constraints:

- A set $X = \{x_1, x_2, x_3, \dots x_n\}$ of *N* points with *d* dimensions each.
- A number of $C = \{c_1, \dots c_k\}, C \geq 2$ clusters.
- A distance $D_{ij}^2, 1 \leq i \leq k, 1 \leq j \leq n$, usually the Euclidean distance. [2]

The steps of the algorithm are:

1. Select the number of clusters.
2. The initial C centroids are chosen randomly among the N dimensions of the dataset.
3. For 1 to C centroids, calculate the distance of each observation to each cluster to find which one it belongs to. If the Euclidean distance is used, if the observations have two attributes, then the formula is (2). This formula subtracts each attribute from one observation to the correspondent

attribute of another observation and it squares it. Then, all the results are summed up and the square root of the final sum is calculated.

$$D_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \cdots} \quad (2)$$

4. The centroids are recalculated using the formula (3), where the value of first attribute of each dataset is summed up and divided by the number of observations, then the second and so on.

$$C_k = (\frac{x_1 + x_{2\pm\ldots+} x_n}{n}, \frac{y_1 + y_{2\pm\ldots+} y_n}{n}, \ldots) \quad (3)$$

5. When the there is new clusters are not assigned to any data point, then stop the iteration, else, go back to step 2 and repeat.

2.2.2 Fuzzy C-Means

Fuzzy C-Means was developed by Bezdek in 1981 and it “is still the most popular classical fuzzy clustering technique” [6]. It is used in fields such as image processing, data analysis, and construction of models [7].

It minimizes the objective function of the equation (4). *X* indicates the dataset, *U* the membership matrix and *V*, the prototypes matrix. The membership matrix has size *C x N*, where *C* is the cluster and *N* is the data point, where each observation has assigned a degree of belongingness to every cluster.

$$J_{FCM}(X, U, V) = \sum_i^C \sum_j^n (u_{ij})^q D_{ij}^2 \quad (4)$$

The popularity of FCM is partly due to its flexible mathematical foundations, which lets to incorporate image feature information such as pixel location, pixel intensity, and combination of location and intensity.

Its performance is supposedly better than any hard-clustering algorithm because of the fuzzifier parameter and the membership value. However, its convergence speed is lower than hard-clustering algorithms.

Otherwise, one of the parameters used in the calculation of the membership matrix is the fuzzifier *m*. If the value of the *m* is large (*m* > 2), it increases the gap between the membership values, leading to a decrease of the overall segmentation performance. If *m* is close to 1, then the algorithm becomes the K-Means.

Some studies show that “high dimensions seem to have a devastating effect on the FCM algorithm” [8]. Specifically, FCM works well until five dimensions. If the data set contains more than that, then it must be started with well initialised prototypes, and if there are more than 20 dimensions, the exact position of the clusters must be known in advance, which means that random initialisation cannot be used at all to find cluster centres. According to [8], this can be solved by initialising the prototypes “very close to the cluster centres”.

It is also worth noticing that the “computational complexity of FCM is quadratic in the number of clusters $O(NC^2P)$, where N is the number of data points, C is the number of clusters and P is the dimension of the data points” [13].

Finally, even though one advantage of fuzzy clustering when compared to hard clustering algorithms is good performance in front of outliers, FCM can also be too sensitive to them. This is because of the membership matrix, “since even a noise point has to be considered to have a higher membership value in a particular cluster” [14].

The constraints of the algorithm are the following:

- A set $X = \{x_1, x_2, \dots, x_i, \dots x_n\}$ of *N* points with *d* dimensions each.
- A number C of clusters where $C \geq 2$
- A confusion matrix $U = [u_{ij}]\, \epsilon[0,1]; 1 \leq i \leq C;\ 1 \leq j \leq N$ where *i* is the index of the data point and *j* is the cluster.
- A prototype matrix $V = [v_1, v_2, \dots, v_j, \dots, v_c]$.
- A distance $D_{ij}^2, 1 \leq i \leq C, 1 \leq j \leq C$.
- A parameter *m* representing the fuzzifier

The steps of the algorithm are:

1. Select the number of clusters.
2. Randomly initialise the membership matrix considering the constraint of (5), that is, the sum of the degree of belongingness of each data point to every cluster must be 1.

$$\sum_1^j u_{ij} = 1; i = 1, 2, \dots, N \tag{5}$$

3. Calculate the prototypes using the equation (6), where the sum of each degree of belongingness to the power of the fuzzifier *m* multiplied by the observation is divided by the sum of the degrees of belongingness of each data point to the power of *m*. The value of *m* is usually 2.

$$V_i = \frac{\sum_j^n {u_{ij}}^m x_j}{\sum_j^n {u_{ij}}^m} \tag{6}$$

4. Calculate the distance to each observation to every prototype using the formula (2).
5. Update the membership matrix using the formula (7). In the formula, in the numerator, 1 is divided by the distance and then raised to the power of 1 divided by the fuzzifier minus 1. The same is performed in the denominator, but for each degree of belongingness of each data point to every cluster, and then it all summed up for each observation.

$$U_{ij} = \frac{(\frac{1}{d_{ij}})^{\frac{1}{m-1}}}{\sum_{k=1}^{C} (\frac{1}{d_{ik}})^{\frac{1}{m-1}}} \tag{7}$$

6. If the result of formula (8) is true, then stop, else, repeat from step 2. The value of ϵ is usually very low, 0.01 or 0.001.

$$\|U^{t+1} - U^t\| < \epsilon \tag{8}$$

2.2.3 Fuzzy C-Means++

One of the problems of FCM is its low convergence speed. There have been many attempts to improve its efficiency, such as Fuzzy C-Means or Suppressed-Fuzzy C-Means. Fuzzy C-means++, which was introduced in 2015 by Stetco, Zeng and Keane [13].

As Stetco et al. explain, Fuzzy C-means++ utilises the seeding mechanism of the K-means++ algorithm to improve the effectiveness and speed of FCM. The idea is to choose points that are spread out in the dataset as representatives and update the membership matrix accordingly before starting.

The first representative is randomly chosen from the dataset and added to the prototype's matrix, renamed by the authors as *R*. This point determines a probability distribution for each other point r_i in the dataset. The bigger the distance from r_1 to r_i, the higher the chance of r_i being picked as the next center.

On synthetic datasets with moderately overlapping clusters, FCM++ was on average 2.1 times faster than FCM.

Fuzzy C-Means++ is not a completely different algorithm. Instead, it is just a set of preparations that are done before starting the actual Fuzzy C-Means algorithm. As Stetco et al. explain [13], the idea is to choose points in the dataset as representatives and then update the membership matrix accordingly before starting FCM. This allows FCM to start in a better position, which is closer to the real centre of the clusters. Hence, the algorithm requires less steps to converge.

The constraints of FCM++ are:

- A set $X = \{x_1, x_2, \dots, x_i, \dots x_n\}$ of *N* points with *d* dimensions each.
- A number C of clusters where $C \geq 2$
- A confusion matrix $U = \left[u_{ij}\right] \epsilon[0,1]; 1 \leq i \leq C;\ 1 \leq j \leq N$ where *i* is the index of the data point and *j* is the cluster.
- A prototype matrix $V = \left[v_1, v_2, \dots, v_j, \dots, v_c\right]$.
- A distance $D_{ij}^2, 1 \leq i \leq C, 1 \leq j \leq C$.
- A parameter *m* representing the fuzzifier
- A vector $dist = D_{ij}^2$ with the distance of every data point to the last centroid calculated.
- A parameter *p* representing the spreading factor

Before executing the FCM algorithm, the FCM++ algorithm performs the following steps:

1. Randomly initialise the first row of the matrix *R* by choosing a random point from the dataset (9).
$$R_1 = x_i \tag{9}$$
2. Calculate the distance of every data point to the first centroid and story it in the vector of distances. Find then the probability of belongingness of each dataset to the first centroid and add it to the first column of the membership matrix U (10).
$$For\ 1\ to\ N: dist_i = distance(R_1, x_i) \tag{10}$$
3. $For\ 2\ to\ C$:
   Find the value of the next centroid R_j through the probability distribution given by $\frac{dist^p}{sum(dist^p)}$. The value chosen will be the one with the highest probability.
4. $For\ 1\ to\ N$:
   Update the vector of distances to the next cluster
5. Repeat from step 3 until the membership matrix U and the set of representatives R are filled.

2.2.4 Suppressed-Fuzzy C-Means

Introduction:

Before Suppressed-Fuzzy C-Means, Rival Checked Fuzzy C-means was introduced to address the problems of FCM. It is based on competitive learning, meaning that it magnifies the largest membership value (u_{pj}) and suppresses the second largest (u_{sj}) membership value through a value α [6], as seen in (11) and (12).

$$u_{pj} = u_{pj} + (1 - \alpha)u_{sj} \tag{11}$$

$$u_{sj} = \alpha u_{sj} \tag{12}$$

The main problem is that it only pays attention to the largest membership values, so if the choice of α is not suitable, it can distort the membership values by making the second largest smaller than others. This may lead to a non-convergence of the algorithm.

To overcome this, a new algorithm was introduced by Fan et al [9]. This algorithm is Suppressed-Fuzzy C-Means. S-FCM magnifies only the largest membership value and supresses the rest, as seen in equations (13) and (14).

$$u_{pj} = 1 - \alpha \sum_{i \neq p} u_{ij} = 1 - \alpha + \alpha u_{pj} \tag{13}$$

$$u_{ij} = \alpha u_{ij}, \;\; i \neq p \tag{14}$$

Where *p* indicates the cluster where the value *u* of the membership is bigger. If there exist more than two biggest membership values, then it takes one of them randomly [9]. Thus, this prizes the biggest membership.

This modification does not disturb the original order and forces the convergence of the algorithm. Also, when α=0, the algorithm is equal to hard-clustering algorithms, and when α=1, it becomes FCM, that is, it establishes a more natural relationship between both hard and soft clustering algorithms.

S-FCM supposedly integrates the advantages of both HC (i.e. it is more efficient) and FCM (i.e. it provides a better clustering performance), and it is not sensitive to the fuzzy factor *m*.

Finally, it is worth noticing that one of the uses of S-FCM is to segment objects having similar surface variations (SSV) precisely because of its insensitiveness to the fuzzy factor *m* and because it “prizes the biggest membership values and suppresses the others”. [10]

The constraints of the algorithm are:

- A set $X = \{x_1, x_2, \dots, x_i, \dots x_n\}$ of *N* points with *d* dimensions.

- A number C of clusters where $C \geq 2$
- A confusion matrix $U = [u_{ij}] \epsilon [0,1]; 1 \leq i \leq C;\ 1 \leq j \leq N$, where i is the index of the data point and j is the cluster.
- A centroids matrix $V = [v_1, v_2, \ldots, v_j, \ldots, v_c]$.
- A distance $D_{ij}^2, 1 \leq i \leq n, 1 \leq j \leq k$.

The steps are the same as FCM, although the formulas (13) and (14) are calculated after step 5.

2.2.5 Gustafson-Kessel Fuzzy C-Means

GKFCM was first developed in 1978 by Donald E. Gustafson and William E. Kessel on his famous article “Fuzzy Clustering With A Fuzzy Covariance Matrix” [11]. It was designed to find non-spherical clusters, since FCM was made to discover spherical clusters with equal volumes and density. This is important because in many cases, datasets have clusters with different shapes [16].

Hence, GKFCM adapts the distance metric to the shape of the cluster. This is done through an “adaptative distance norm unique for every cluster as the norm-inducing matrix A, which is calculated by estimates of the data covariance” [1, 16], following the formula (15).

$$A_i = [\rho_i \det(F_i)]^{1/n} F_i^{-1} \tag{15}$$

Another advantage of the GKFCM is that while FCM needs a good initialization of the partition matrix to return good results, GKFCM is insensitive to the data scaling of that initialization. Also, GKFCM performs very well for large datasets and it allows to integrate generic shape information into the clustering framework, which means that it is a good tool for image processing.

One disadvantage of GKFCM is that, even though it is computationally more efficient than FCM [21], its performance is not the best when the datasets are small or there are linearly correlated datapoints [6], since the covariance matrix becomes singular.

The steps of the algorithm are [24]:

$\boldsymbol{Repeat\ for\ l = 1, 2, \ldots}$

1. Compute the cluster prototypes with the formula (6).
2. Compute the cluster covariance matricx using the formula (16).

$$\boldsymbol{F_i} = \frac{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^m (Z_k - V_i^l)(Z_k - V_i^l)^T}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, 1 \le i \le K \tag{16}$$

3. Compute the distances with (17), which uses the covariance matrix from (16) and the covariance of (15).

$$\boldsymbol{D_{ikA_i}^2} = \left(Z_k - V_i^l\right)^T \left[\rho_i det(F_i)^{\frac{1}{n}} F_i^{-1}\right] \left(Z_k - V_i^l\right), 1 \le i \le K, 1 \le k \le N \tag{17}$$

4. Update the partition matrix iteratively:

   $\boldsymbol{for}\ 1 \le k \le N$

$\boldsymbol{if}\ D_{ikA_i} > 0\ for\ 1 \leq i \leq K\ \boldsymbol{then}$

$$\boldsymbol{\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{K}\left(D_{ikA_i}/D_{jkA_j}\right)^{2/(m-1)}}}$$

$\boldsymbol{else}$

$\boldsymbol{if}\ D_{ikA_i} > 0, and\ \mu_{ik}^{(l)} \in [0,1]\ \boldsymbol{then}$

$\mu_{ik}^{(l)} = 0$

$\boldsymbol{else}$

$\sum_{i=1}^{K} \mu_{ik}^{(l)} = 1$

$\boldsymbol{Until}\ \left\|\boldsymbol{U^{l} - U^{l-1}}\right\| < \boldsymbol{\epsilon}$

2.2.6 Possibilistic C-Means

Fuzzy clustering algorithms are based on the probabilistic theory, but there are other soft clustering algorithms based on the possibilistic theory. In fact, FCM uses a probabilistic constraint to make the sum of all the memberships of one observation across all the clusters be 1. Thus, the degree of membership of one data point to one cluster, represents the degree of sharing, i.e. an arbitrary division of data, but not the actual degree of belongingness. Possibilistic algorithms such as Possibilistic C-Means were introduced to address this issue [6].

Possibilistic C-Means was defined by Krishnapuram in 1993 [17] to improve some of the flaws of FCM. For example, PCM is said to be better in finding the correct clusters in noisy datasets thanks to a less strict degree of belongingness to every cluster. While FCM's memberships represent probabilities, PCM's memberships represent typicality or an elastic constraint.

Thus, in PCM, the probabilistic constraint (see equation below) is eliminated, and every cluster is independent of the other clusters, as shown by (18), and minimises the objective function (19).

$$\sum_{i=1}^{C} u_{ij} = 1, j\epsilon\{1, \dots, C\} \tag{18}$$

$$J_{PCM}(X, U, V) = \sum_{i=1}^{C}\sum_{j=1}^{n}(u_{ij})^q D_{ij}^2 + \sum_{i=1}^{C}\eta_i \sum_{j=1}^{n}(1-u_{ij})^q \tag{19}$$

The PCM algorithm is applied twice. If the value of *q* is equal to 1, then the PCM becomes crisp.

The main advantage of PCM is that, as it gives more emphasis to typicality, it can visually separate distinctive objects very well. The main disadvantage is that, when objects are not visually different, it produces poorer segmentation performance.

The constraints of the algorithm are:

- A set $X = \{x_1, x_2, \dots, x_i, \dots x_n\}$ of *N* points with *d* dimensions.
- A number C of clusters where $C \geq 2$
- A confusion matrix $U = \left[u_{ij}\right] \epsilon[0,1]; 1 \leq i \leq c;\ 1 \leq j \leq n$ where *i* is the index of the data point and *j* is the cluster.
- A vector $V = \left[v_1, v_2, \dots, v_j, \dots, v_c\right]$ of centroids.
- A distance $D_{ij}^2, 1 \leq i \leq n, 1 \leq j \leq k$, usually the Euclidean distance.

The steps of PCM are [17]:

1. Randomly initialise the possibilistic C-partition $\boldsymbol{U}^{(o)}$
2. Estimate the value of the parameter $\boldsymbol{\eta_i}$ by using the formula (20).

$$\eta_i = K \frac{\sum_{j=1}^{N} u_{ij}^m u_{ij}^2}{\sum_{j=1}^{N} u_{ij}^m} \tag{20}$$

3. Until (21) is true, repeat steps 4 and 5.

$$\left\| U^{(l-1)} - U^{(l)} \right\| < \epsilon \tag{21}$$

4. Update the prototypes using (22).

$$u_{ij} = \frac{1}{1+(\frac{d_{ij}^2}{\eta_i})^{\frac{1}{m-1}}} \tag{22}$$

3 Datasets and validation methods

3.1 Introduction

In this section, I will introduce the datasets and the validation methods I will use in Section 4. There are eleven datasets, ten synthetically generated and one from a real-world case, which is the Iris dataset. As for the validation methods, they are divided in internal and external validation methods. The internal methods are the performance, which will be measured through two indexes, the Xie-Beni and the Silhouette, and the computational efficiency, which will be measure through the number of iterations. Otherwise, I am going to use one the external method, the accuracy.

3.2 Datasets

3.2.1 Synthetic dataset 1 (SD1)

I will use the Synthetic Dataset 1 (SD1) with the objective to compare KM with FCM and FCM with FCM++ and S-FCM. Based on this aim, I have synthetically generated this dataset with 1000 observations and three clusters, two of them overlapping and one well-separated. This is a convenient number of observations and clusters in order to reduce the number of iterations required by the fuzzy clustering algorithms.

This dataset will allow me to check how much more accurate is the FCM compared to the KM and how much more efficient are FCM++ and S-FCM confronted to FCM. The pictorial representation of the dataset is given in Figure 1.

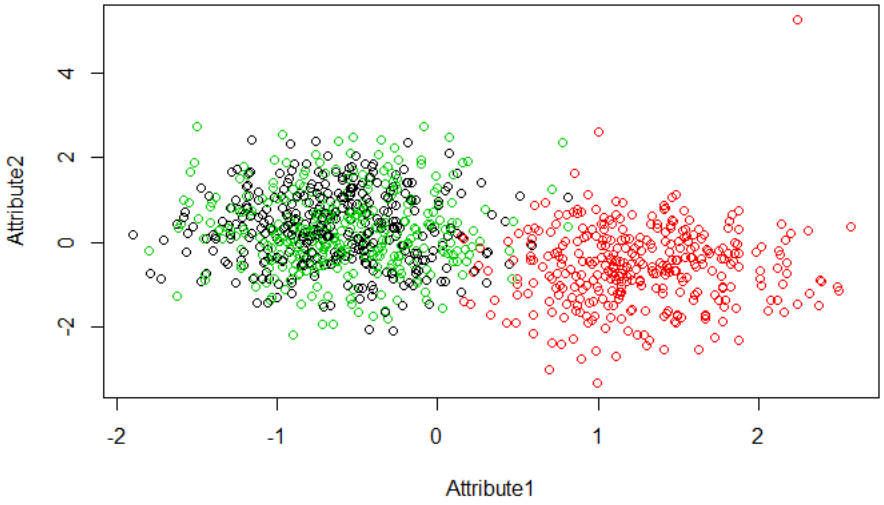

Figure 1. SD1

3.2.2 Synthetic dataset 2 (SD2)

I will use the Synthetic Dataset 2 (SD2) with the objective to compare KM with FCM and FCM with FCM++ and S-FCM. Based on this, I have synthetically generated this dataset with 1000 observations and three well-separated clusters. This is a convenient number of observations and clusters in order to reduce the number of iterations required by the fuzzy clustering algorithms.

This dataset will allow me to check how much more accurate is the KM compared to the FCM in the event of well-separated clusters and how much more efficient are FCM++ and S-FCM confronted to FCM. The representation of the dataset is given in Figure 2.

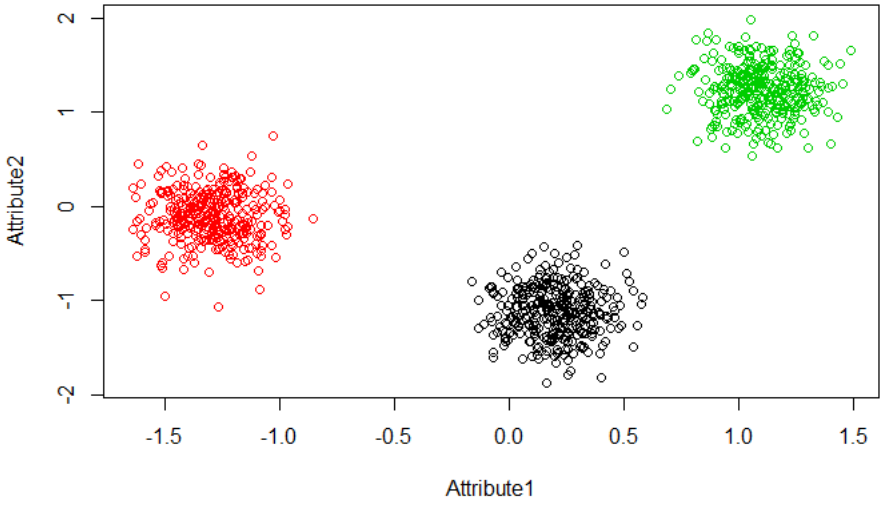

Figure 2. SD2

3.2.3 Synthetic dataset 3 (SD3)

I will use the Synthetic Dataset 3 (SD3) with the objective to compare FCM and GKFCM. Based on this, I have synthetically generated this dataset with 240 observations and three non-spherical and well-separated clusters. This is a convenient number of observations and clusters in order to reduce the number of iterations required by the fuzzy clustering algorithms.

This dataset will allow me to check how much more accurate is the GKFCM compared to the FCM in the event of non-spherical and well-separated clusters. The representation of the dataset is given in Figure 3.

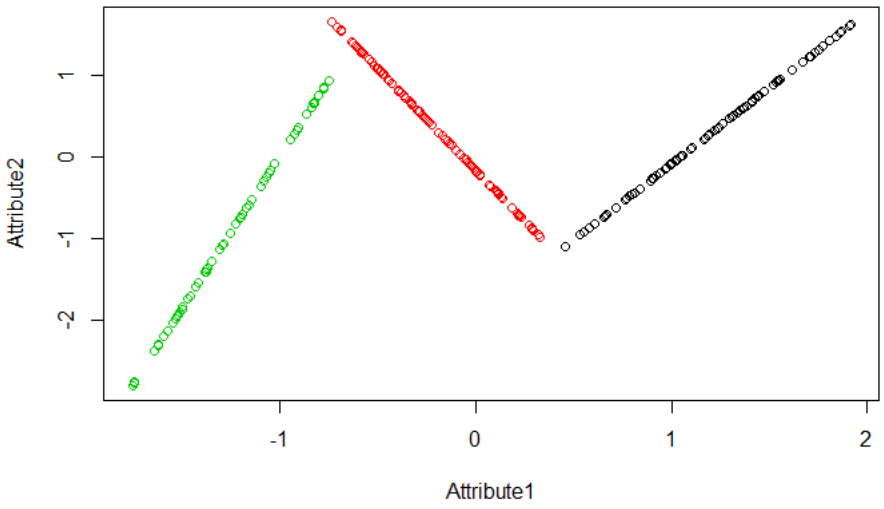

Figure 3. SD3

3.2.4 Synthetic dataset 4 (SD4)

I will use the Synthetic Dataset 4 (SD4) with the objective to compare FCM and PCM. Based on this, I have synthetically generated this dataset with 1000 observations and two well-separated clusters with outliers. This is a convenient number of observations and clusters in order to reduce the number of iterations required by the fuzzy clustering algorithms.

This dataset will allow me to check how much more accurate is the PCM compared to the FCM in the event of outliers. The representation of the dataset is given in Figure 4.

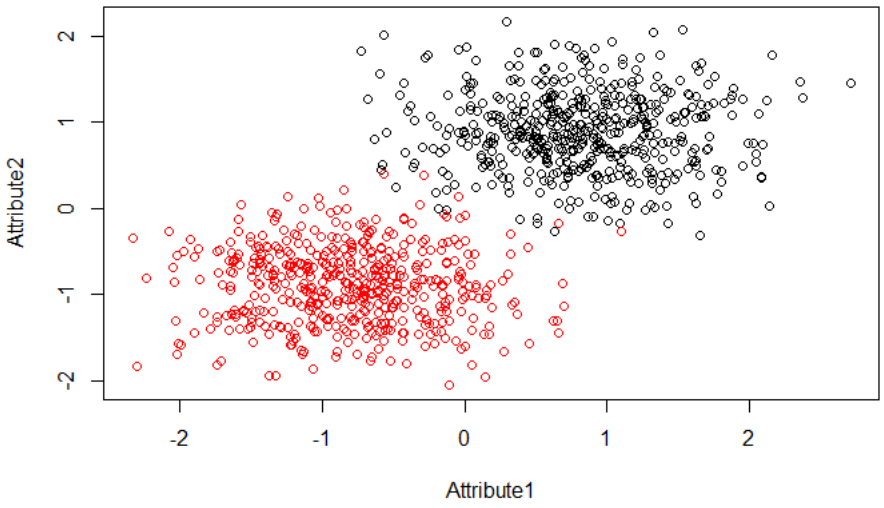

Figure 4. SD4

3.2.5 Synthetic dataset 5 (SD5)

I will use the Synthetic Dataset 5 (SD5) with the objective to compare FCM and PCM. Based on this, I have synthetically generated this dataset with 1000 observations, 3 well-separated clusters with outliers and two attributes. This is a convenient number of observations, clusters and attributes in order to reduce the number of iterations required by the fuzzy clustering algorithms and to make easier its representation. Also, it will allow me to see the behaviour of PCM under a different number of clusters.

This dataset will allow me to check how much more accurate is the PCM compared to the FCM in the event of outliers. The representation of the dataset is given in Figure 5.

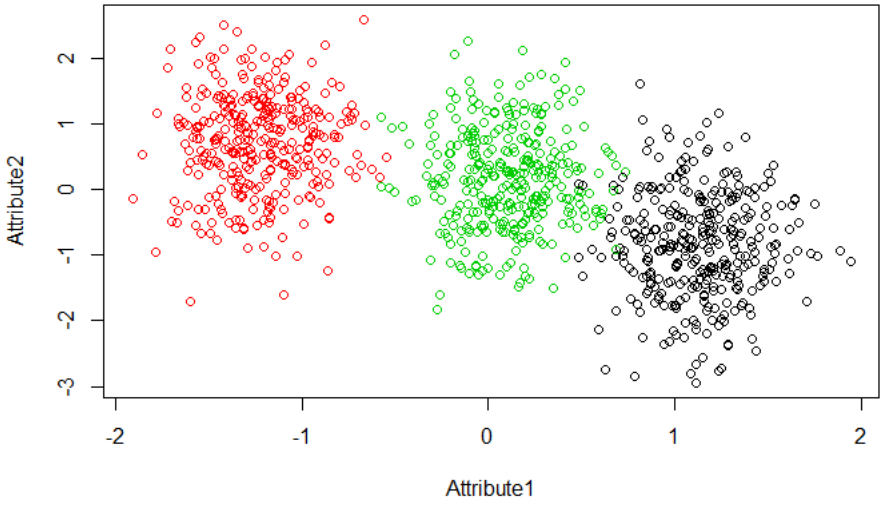

Figure 5. SD5

3.2.6 Synthetic dataset 6 (SD6)

I will use the Synthetic Dataset 6 (SD6) with the objective to compare FCM and PCM. Based on this, I have selected this dataset from [26] which contains 3100 observations, 31 clusters with some noise and two features. This is a convenient number of observations, clusters and attributes because it will allow me to see the behaviour of PCM under an extreme number of clusters.

This dataset will allow me to check how much more accurate is the PCM compared to the FCM in the event of outliers and many clusters. The representation of the dataset is given in Figure 6.

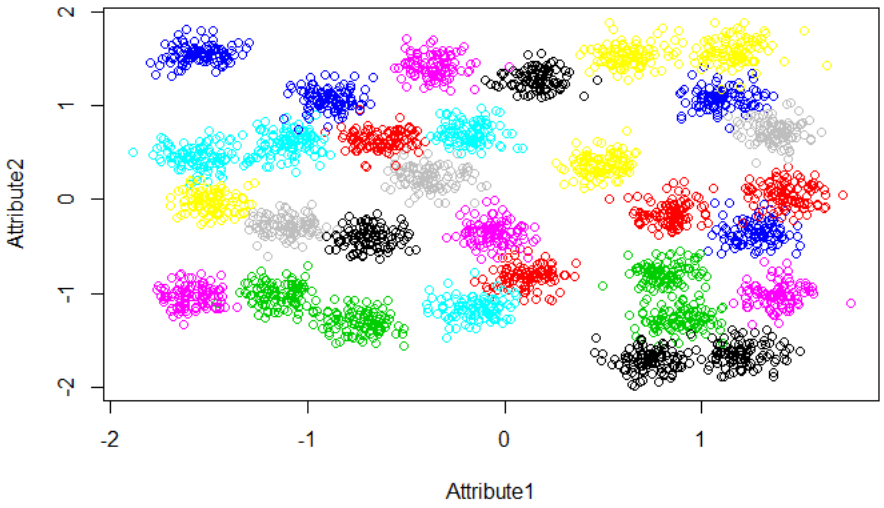

Figure 6. SD6

3.2.7 Synthetic dataset 7 (SD7)

I will use the Synthetic Dataset 7 (SD7) with the objective to compare FCM and GKFCM. Based on this, I have selected this dataset from [28] which contains 312 observations, 3 non-spherical clusters and two features. This is a convenient number of observations, clusters and attributes because it will allow me to see the behaviour of GKFCM under non-spherical and curved clusters.

Thus, this dataset will allow me to check how much more accurate is the GKFCM compared to the FCM in the event of non-spherical clusters. The representation of the dataset is given in Figure 7.

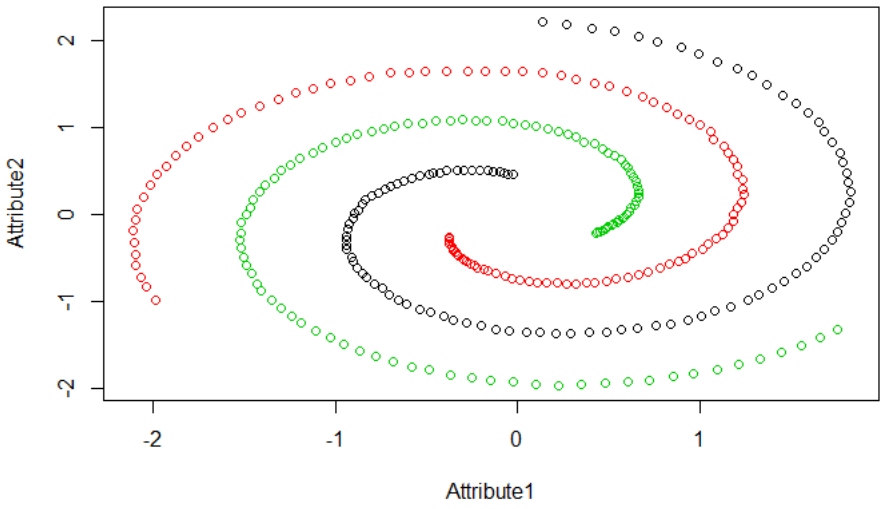

Figure 7. SD7

3.2.8 Synthetic dataset 8 (SD8)

I will use the Synthetic Dataset 8 (SD8) with the objective to compare FCM and GKFCM. Based on this, I have selected this dataset from [28] which contains 373 observations, two non-spherical and fuzzy clusters and two features. This is a convenient number of observations, clusters and attributes because it will allow me to see the behaviour of GKFCM under non-spherical, fuzzy and curved clusters.

Thus, this dataset will allow me to check how much more accurate is the GKFCM compared to the FCM in the event of this specific dataset. The representation of the dataset is given in Figure 8.

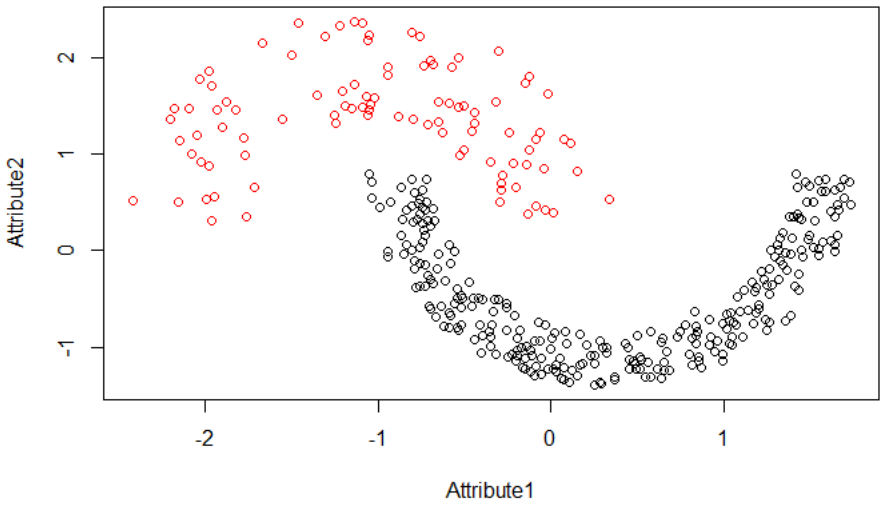

Figure 8. SD8

3.2.9 Synthetic dataset 9 (SD9)

I will use the Synthetic Dataset 9 (SD9) with the objective to compare FCM and GKFCM and FCM and PCM. Based on this, I have selected this dataset from [26] which contains 600 observations, 15 spherical and well-separated clusters and two features. This is a convenient number of observations, clusters and attributes because it will allow me to see the behaviour of FCM, GKFCM and PCM under the conditions several well-separated clusters.

Thus, this dataset will allow me to check how much more accurate is the FCM compared to the GKFCM and PCM in the event of this specific dataset. The representation of the dataset is given in Figure 9.

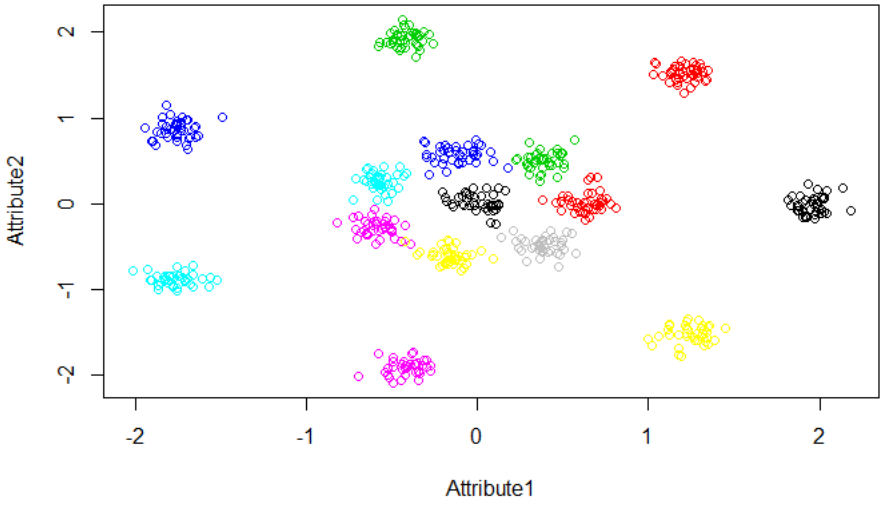

Figure 9. SD9

3.2.10 Synthetic dataset 10 (SD10)

I will use the Synthetic Dataset 10 (SD10) with the objective to compare FCM and GKFCM and FCM and PCM. Based on this, I have selected this dataset from [28] which contains 300 observations, 3 clusters, two spherical and one non-spherical, and two features. This is a convenient number of observations, clusters and attributes because it will allow me to see the behaviour of FCM, GKFCM and PCM under the conditions mixed spherical and non-spherical clusters.

Thus, this dataset will allow me to check how much more accurate is the FCM compared to the GKFCM and PCM and vice-versa in the event of this specific dataset. The representation of the dataset is given in Figure 10.

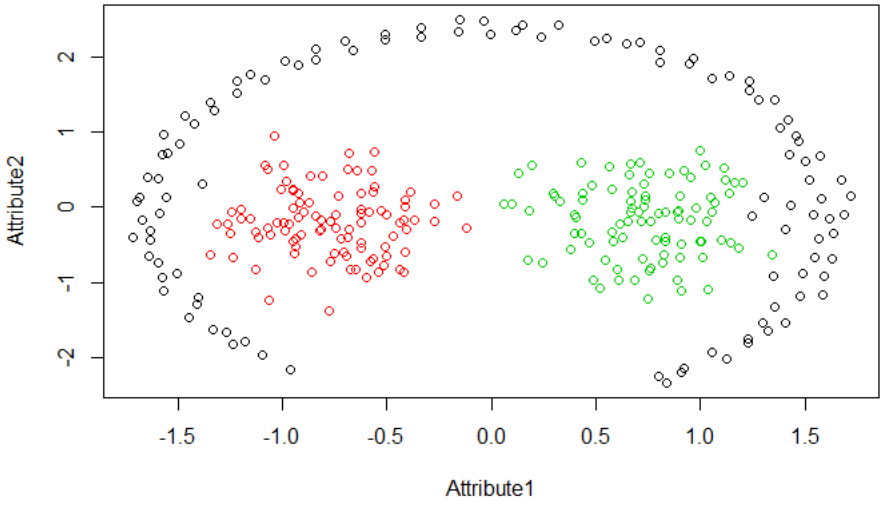

Figure 10. SD10

3.2.11 Real-world dataset 1 (RWD1)

I will use the Real-world Dataset (RWD1) with the objective to compare KM with FCM and FCM with FCM++ and S-FCM. Based on this, I have selected this dataset which contains 150 observations, 3 well-separated and overlapping clusters, and four features. This is a convenient number of observations, clusters and attributes because it will allow me to compare KM, FCM, FCM++ and S-FCM within a reasonable amount of iterations.

Thus, this dataset will allow me to check how much better performance and efficiency has FCM confronted to KM and vice-versa, as well as FCM++ and S-FCM compared with FCM. The representation of the dataset is given in Figure 11, which shows the four attributes of the dataset in a confusion matrix.

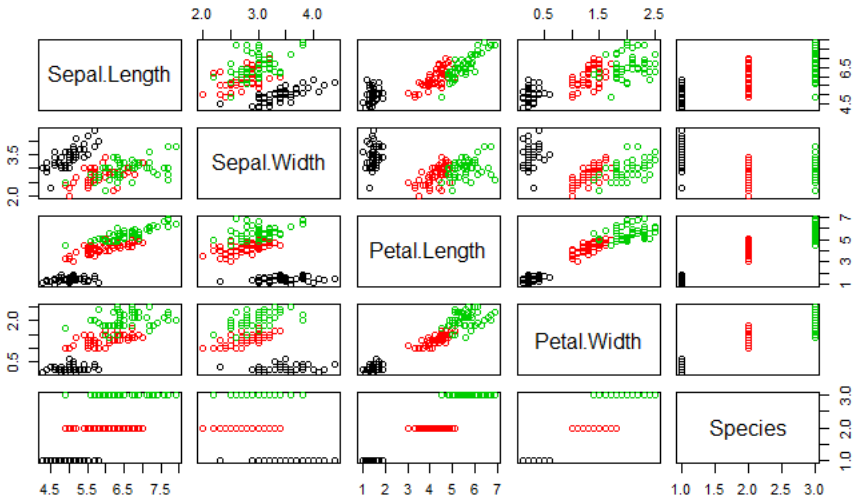

Figure 11. RWD1

3.3 Validation methods

3.3.1 Internal methods

I will use two internal validation methods, performance and efficiency. I will measure performance through two indexes, the Xie-Beni and the Silhouette.

Equation (23) shows how the Xie-Beni index works. It defines the separation between the clusters, which is the minimum square distance that exists between the cluster centres and the compactness within each cluster, which is the average square distance existing between every observation and its cluster center. [23] The smaller the value of this index is, the more optimal the number of clusters is.

Otherwise, equation (24) shows how the Silhouette index works. It "validates the clustering performance based on the pairwise difference of between and within-cluster distances". [23] The larger the value of this index is, the more optimal the number of clusters is.

$$[\sum_i \sum_{x \epsilon C_i} d^2(x, C_i)]/[n \cdot min_{i,j \neq i} d^2(x, C_i)] \quad (23)$$

$$\{\frac{1}{NC} \sum_i \{\frac{1}{n_i} \sum_{x \epsilon C_i} \frac{b(x) - a(x)}{\max[b(x), a(x)]}\} \quad (24)$$

On the other hand, I will measure efficiency through the number of iterations.

3.3.2 External methods

I will use one external measure, accuracy. I will validate accuracy by finding the correct number of well-clustered observations, based on the original clusters, and dividing it by the total number of observations.

4 Experiments

4.1 Introduction

I have established four groups of experiments which concern six algorithms. I will perform fifteen experiments to do the validation, using eleven different datasets. The datasets contain different kind of clusters such as overlapping or non-overlapping clusters with and without noise. Some of them are spherical and some other contain non-spherical geometrical shapes. I will explain the relationship between hypotheses, experiments and datasets below.

The first group of experiments concern the K-Means and the Fuzzy C-Means algorithms as well as the datasets SD1, SD2 and RWD1. The details of these datasets are given in Section 3. I want to validate that FCM is more accurate than KM when the dataset contains overlapping clusters. I also want to validate that KM is always more computationally efficient than FCM. The parameters for these experiments are efficiency and accuracy.

The second group of experiments concerns Fuzzy C-Means, Fuzzy C-Means++ and Suppressed-Fuzzy C-Means. For this group, I will use the datasets SD1 and RWD1, two datasets that contain both overlapping and well-separated fuzzy clusters. I want to validate that FCM++ and S-FCM perform better than FCM when the number of clusters set in the algorithm is exactly or closer to the actual number, and they perform worse otherwise. The parameters used to validate these experiments are efficiency and performance.

The third group of experiments involves Fuzzy C-Means and Gustafson Kessel Fuzzy C-Means. For this group, I will use five datasets, SD3, SD7, SD8, SD9 and SD10, which contain different kind of datasets, such as spherical and non-spherical clusters. I will check under which conditions GKFCM performs better than FCM, and vice-versa. The parameters used to validate these experiments are efficiency and accuracy.

Finally, in the fourth group of experiments I will compare Fuzzy C-Means and Possibilistic C-Means on the datasets SD4, SD5, SD6, SD9 and SD10. These datasets contain clusters with different degrees of outliers and shapes. I want to validate that, for clusters with different degrees of outliers and shapes, PCM is more accurate than FCM. The parameters used for the validation are efficiency and accuracy.

The following sections will provide the results of each group of experiments, a discussion at the end of each one and a final discussion involving all of the experiments.

4.2 Groups of experiments

4.2.1 K-Means and Fuzzy C-Means

The following group of experiments is based on three experiments and validations involving K-Means and Fuzzy C-Means.

The main advantage of K-Means over Fuzzy C-Means is that it is a faster algorithm, which means that it takes less iterations to converge. Otherwise, the main advantage of FCM over KM is that is more accurate when clusters are overlapping. I am going to validate this by measuring the accuracy and the efficiency of each algorithm and by comparing the results.

I will use SD1 and SD2 to validate that FCM is more accurate than KM for overlapping and fuzzy clusters and that KM is always more computationally efficient. I will use a RWD with both overlapping and well-separated clusters to do the validation under realistic observations with unexpected situations.

The results of the validation are summarised in Table 1. The table shows that when clusters are clear and separated and for over ten executions of the algorithm, KM requires an average of 1.89 iterations to converge, whereas FCM requires 10.82 iterations. In both cases, the accuracy is the same, 100%. Then, for clear and well-separated clusters, either KM or FCM are equally accurate finding the actual clusters. In the specific case of KM, it is clearly faster than FCM.

Iterations K-Means	Iterations Fuzzy C-Means	Accuracy K-Means	Accuracy Fuzzy C-Means
1.89	10.82	100%	100%

Table 1: Validation for Hypothesis 1 using SD2

Hence, the numbers show that, for datasets with well-separated clusters, both KM and FCM have the same accuracy, although KM is converges faster than FCM.

Table 2 shows the results of the validation on a dataset with three fuzzy clusters where two of them are overlapping overlap (SD1). Over ten executions of the algorithm, KM has performed an average of 2.89 iterations, whereas FCM has performed an average of 63.68 iterations. The average accuracy of KM has been 29.8%, and the average accuracy of FCM, 36.6%.

Based on the results, for overlapping and fuzzy clusters, FCM is more accurate than KM, even though KM is clearly more computationally efficient than FCM.

Iterations K-Means	Iterations Fuzzy C-Means	Accuracy K-Means	Accuracy Fuzzy C-Means
2.89	63.68	29.8	36.5

Table 2: Validation for Hypothesis 2 with SD1

So far, I have used synthetic datasets specifically built for the validation. In real-world, though, the observations in a dataset are not so optimal to validate the hypotheses, since the observations are unexpected. Hence, I want to do the validation under the circumstances of unexpected conditions. For that objective, I am going to use a real-world dataset (RWD1). This dataset contains one clear cluster and two overlapping clusters.

Table 3 shows the results of the validation with RWD1. After ten executions of both KM and FCM, the average iterations of KM are 2.14, whereas the average iterations for FCM are 47.17. The average accuracy of KM is 29.5% and the average accuracy of FCM is 28.7%. Therefore, although the KM is slightly more accurate than FCM, the difference is not significative. Also, the KM is much more efficient than FCM.

Iterations K-Means	Iterations Fuzzy C-Means	Accuracy K-Means	Accuracy Fuzzy C-Means
2.14	47.17	29.5	28.7

Table 3: Validation for Hypothesis 2 using RWD1

In conclusion, FCM provides with more information about the class every observation belongs to through a probability. This allows to get more accurately the class of each data point, but also, in the cases when it is not clear where do they belong to, it indicates it is possibly an outlier. However, FCM takes increasingly longer time to converge as the clusters get more overlapping in nature. In datasets with clear and separated clusters, KM is more accurate than FCM. Finally, in all cases, FCM has been less computationally efficient than KM, which is a disadvantage for large datasets.

4.2.2 Fuzzy C-Means, Fuzzy C-Means ++ and Supressed-Fuzzy C-Means

In this group of experiments, I want to investigate how good the performance (internal validation) and the efficiency (external validation) of FCM++ and SFCM algorithms is on datasets containing fuzzy and overlapping clusters. The datasets used for the validation are SD1 and RWD1. I will execute the algorithm ten times and then I will calculate the average number of iterations and the average performance.

Even though the actual number of clusters is always $k = 3$, I will also perform the validation for $k = 2, 3, 4, 5$. This will help me determine whether the algorithms actually find the best results for the actual number of clusters of the datasets or for another number of clusters. It will show which number of clusters is most suitable for these soft algorithms, effectively determining their capacity to perform well with overlapping clusters.

In order to carry the performance validation, I have chosen the indexes Xie-Beni and Silhouette because they are two of the most widely used indexes to validate fuzzy algorithms.

My third hypothesis is that modified Fuzzy C-Means algorithms, such as Fuzzy C-Means++ and Suppressed Fuzzy C-Means, increase the computational efficiency compared to Fuzzy C-Means and they improve the performance.

The results of the validation are in Table 4, Figure 12, Figure 13 and Figure 14. Table 4 illustrates the efficiency and performance of FCM, FCM++ and S-FCM for $k = 2, 3, 4, 5$. The efficiency is measured through the iterations and the performance through the Xie-Beni and Silhouette indexes.

Table 4 shows that for actual number of clusters $k=3$, S-FCM takes 20 iterations to converge while FCM and FCM++ take between 64 and 66. Otherwise, the XB index is lower for FCM and FCM++, 0.288, than for S-FCM, 0.577, but the Silhouette index is similar in all cases. For the rest of the values of k, it is worth it to note that the number of iterations of S-FCM increase slightly as k increases, while in the case of FCM and FCM++, it increases significantly. Finally, for FCM and FCM++, the values of XB are always lower than S-FCM, and the values of Silhouette are similar in all cases.

Num. of clusters (k)	It. FCM	It. FCM++	It. S-FCM	XB FCM	XB FCM++	XB S-FCM	Sil. FCM	Sil. FCM++	Sil. S-FCM
2	35.1	37.3	11.7	0.18	0.18	0.455	0.717	0.717	0.678
3	64.2	66.6	20	0.228	0.228	0.577	0.675	0.675	0.616
4	245.3	186.3	26.3	0.207	0.192	0.574	0.644	0.673	0.565
5	101.1	102.7	25.9	0.265	0.265	0.657	0.608	0.608	0.523

Table 4: Validation for Hypothesis 3 using XB and Sil. Indexes for SD1

Therefore, S-FCM is always more computationally efficient than FCM and FCM++, but the Xie-Beni index shows that FCM and FCM++ perform better than S-FCM. Also, there is not much difference between FCM and FCM++ in efficiency and performance.

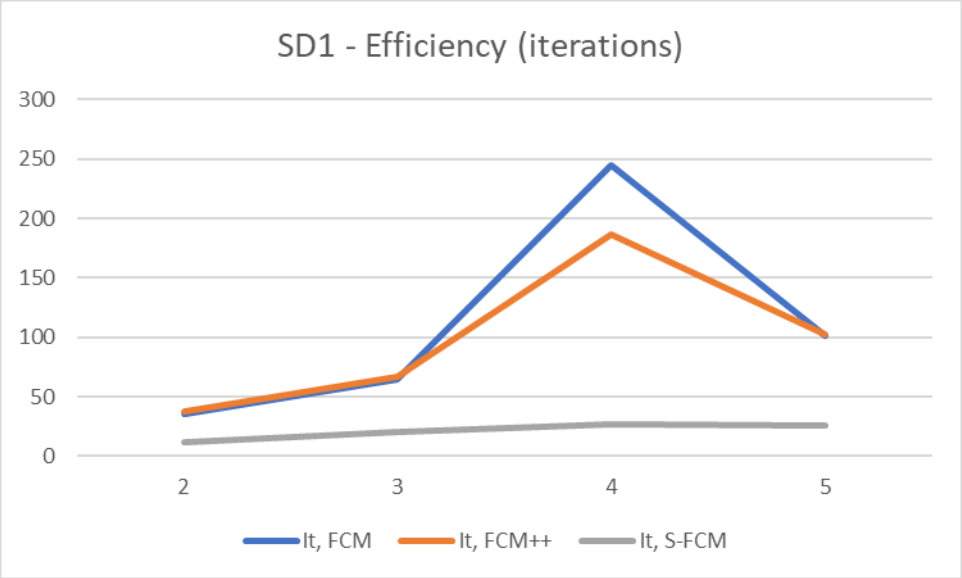

Figure 12. Efficiency of FCM, FCM++ and S-FCM for SD1

Figure 12 shows that the efficiency of FCM and FCM++ is very similar, the efficiency of S-FCM is better, and that the efficiency of SFCM is more consistent than FCM and FCM++ for all values of *k*.

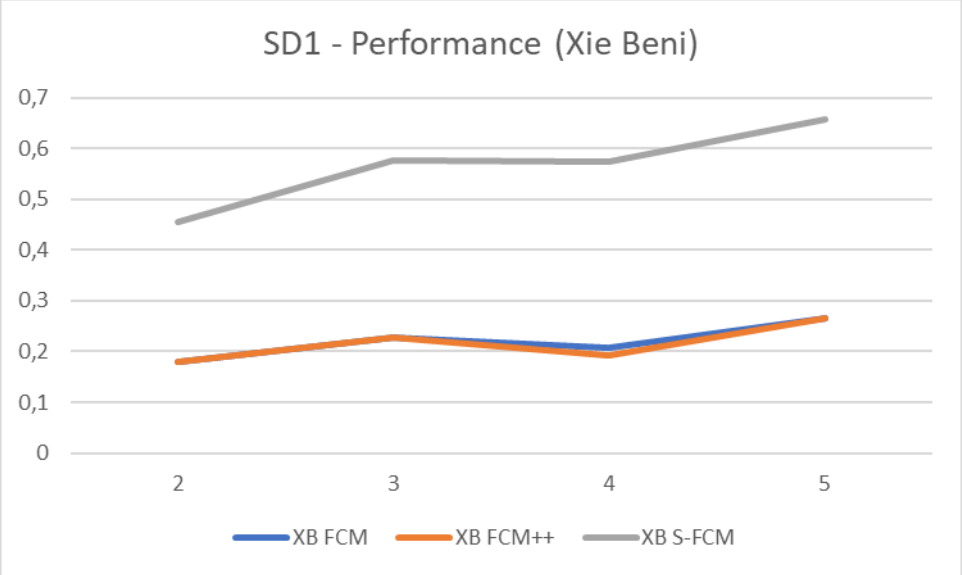

Figure 13. Performance (Xie-Beni) of FCM, FCM++ and S-FCM for SD1

Figure 13 shows that FCM and FCM++ always performs better than S-FCM, as their values are lower.

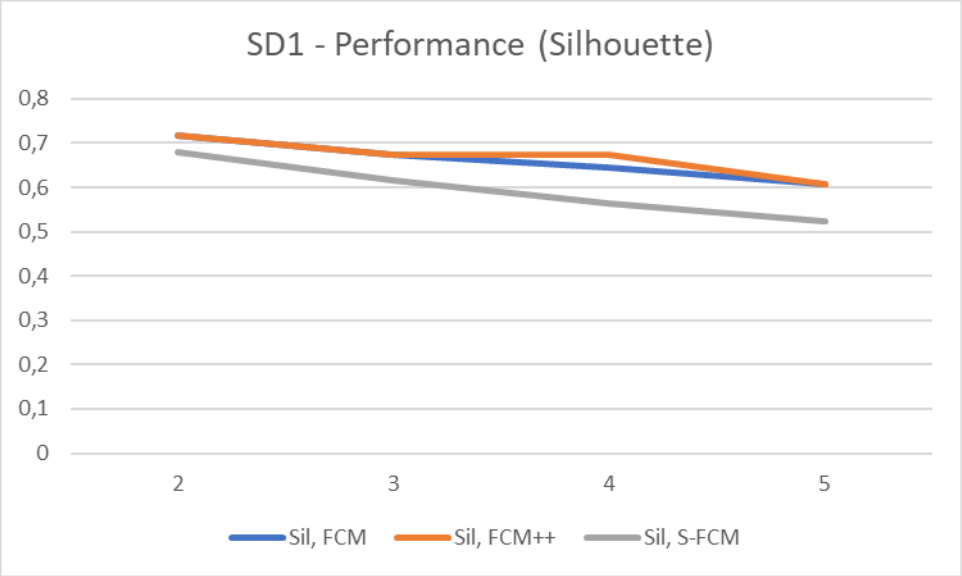

Figure 14. Performance (Silhouette) of FCM, FCM++ and S-FCM for SD1

Figure 14 shows that FCM and FCM++ always performs better than S-FCM especially for lower values of *k*.

Table 5 and in Figures 15, 16 and 17 are the results of the validation using a real-world data set (RWD1) with less overlapping clusters than SD1. The number of iterations is like the one for SD1, showing that, as the value of *k* increases, so does the number of iterations. On the contrary, the number of iterations of S-FCM is lower than those of FCM and FCM++ being always very similar for all values of *k*.

The Xie-Beni indexes are quite low for FCM and FCM++, being the lowest for *k=2*, followed by *k=3*. On the contrary, the S-FCM shows higher values of XB compared to FCM and FCM++, especially for higher values of *k*, although the difference between the three algorithms is smaller than that of SD1.

On the other hand, the Silhouette index shows higher values for FCM and FCM++, which are very similar. The highest value is for *k=2*. The values for S-FCM are more like those of FCM and FCM++ than in the case of SD1. The highest value is also for *k=2*. Therefore, the performance is slightly better in the case of FCM and FCM++.

Number of clusters (k)	It. FCM	It. FCM++	It. S-FCM	XB FCM	XB FCM++	XB S-FCM	Sil. FCM	Sil. FCM++	Sil. S-FCM
2	23.4	22.8	10.5	0.113	0.113	0.253	0.807	0.807	0.792
3	45.7	48.4	9.8	0.222	0.222	0.59	0.729	0.729	0.69
4	68.9	73.9	13.4	0.272	0.272	0.835	0.669	0.668	0.596
5	107.5	73	14.3	0.367	0.359	1.68	0.609	0.658	0.524

Table 5: Validation for Hypothesis 3 using XB and Sil. Indexes using RWD1

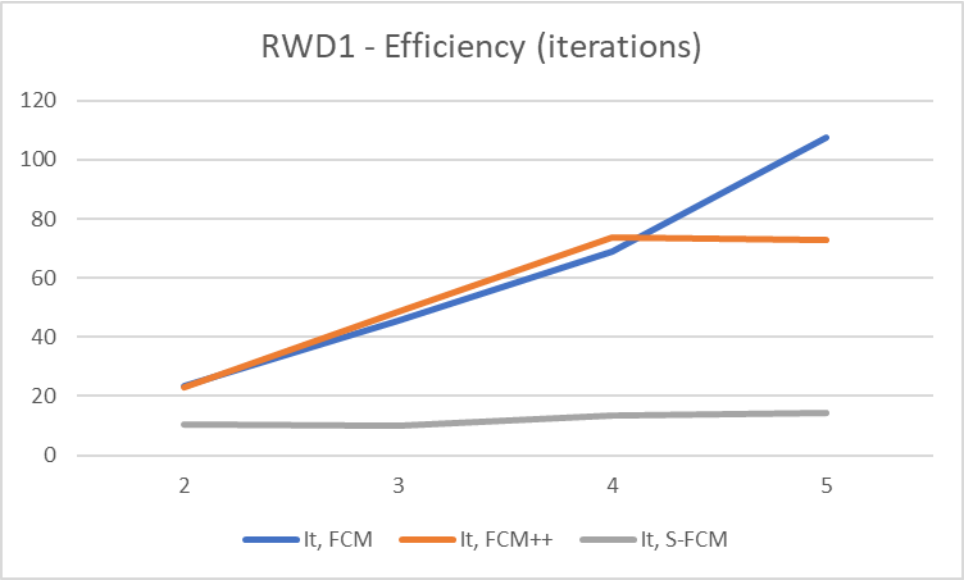

Figure 15. Efficiency of FCM, FCM++ and S-FCM for RWD1

Table 15 shows that the number of iterations of FCM++ increase as *k* increases, but it becomes stable for *k* = *4, 5* and the number of iterations of S-FCM remains low and stable for every value of *k*.

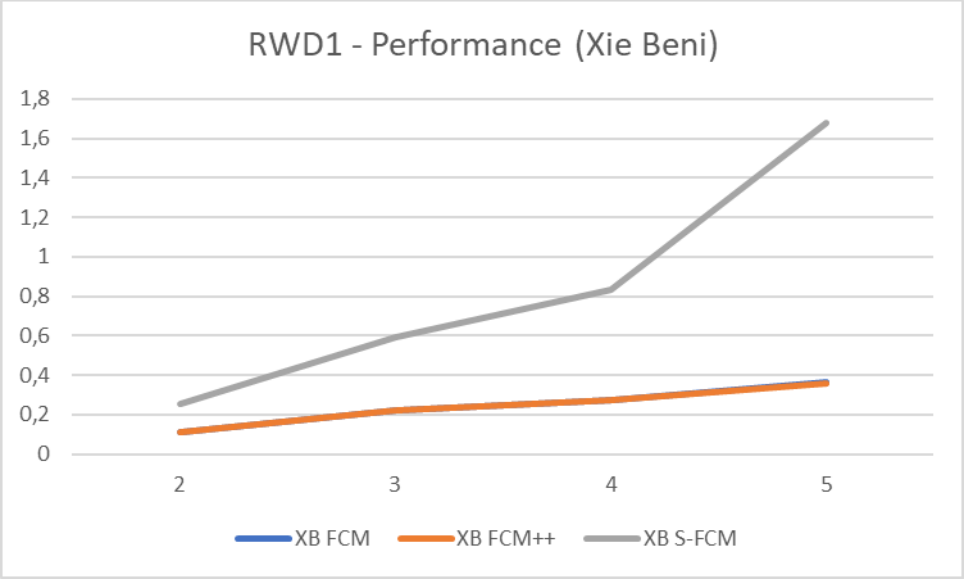

Figure 16. Performance (Xie-Beni) of FCM, FCM++ and S-FCM for RWD1

Table 16 shows that the XB index is better on FCM and FCM++, as it is lower, although in the case of S-FCM performs better for lower values of *k*, which are closer to the actual number of clusters, and worse for higher values of *k*. On the contrary, XB remains lower and stable for FCM and FCM++. It is also worthy to note that FCM and FCM++ have similar values.

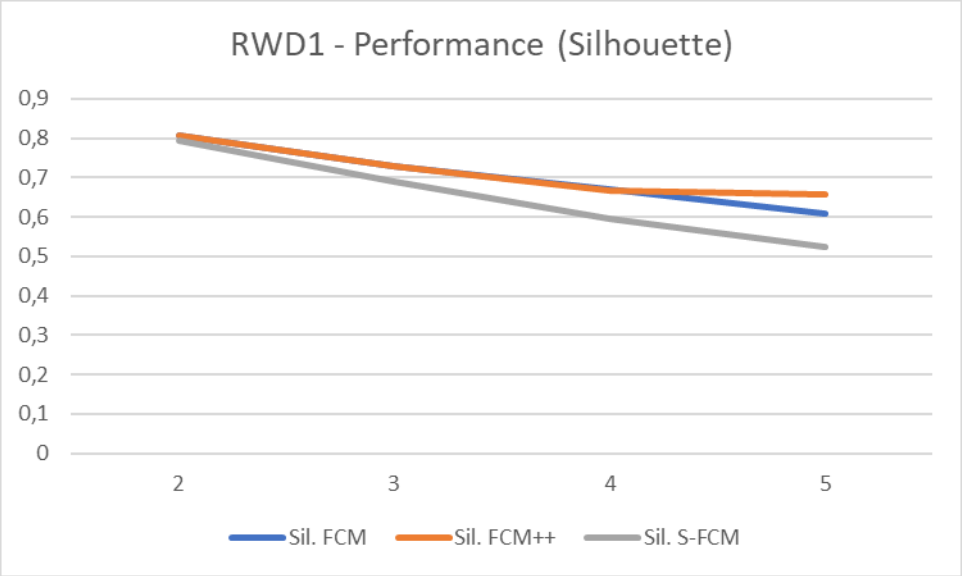

Figure 17. Performance (Silhouette) of FCM, FCM++ and S-FCM for RWD1

In Table 17, the Silhouette index is better on FCM and FCM++, even though the values are similar for the three algorithms.

In conclusion, either on the synthetic or the real-world dataset with overlapping and well-separated clusters (SD1 and RWD1), both FCM and FCM++ perform better than S-FCM, even though their efficiency is worse. Therefore, S-FCM is more computationally efficient than FCM and FCM++: the efficiency of these two algorithms decreases as the number of *k* increases, whereas for S-FCM the efficiency remains always stable. However, the performance of S-FCM is a bit better for less overlapping clusters. Hence the less the clusters overlap on a dataset, the better the S-FCM performs, and the more the clusters overlap, the better the FCM and FCM++ perform. In any case, if efficiency is important and the clusters do not overlap so much, the best choice is S-FCM, whereas if efficiency is not important, either FCM or FCM++ are a good option.

Finally, it is interesting to note that, for the datasets analysed, FCM++ does not seem to improve the efficiency of FCM, as it is claimed.

4.2.3 Fuzzy C-Means and Gustafson-Kessel Fuzzy C-Means

Gustafson-Kessel Fuzzy C-Means is an algorithm designed to improve FCM's efficiency and to work better with non-spherical clusters. Next, I am going to introduce a group of experiments on five datasets to validate these assumptions. Three of the datasets contain non-spherical clusters and different levels of fuzziness (SD3, SD7 and SD8), one contains spherical clusters (SD9) and another has a mix of spherical and non-spherical clusters (SD10). They will let me check under which conditions GKFCM performs better than FCM and vice-versa.

Table 6 shows the results of the validation for SD3. The number of iterations of FCM increases as the value of *k* increases, except for *k=4*, which is the lowest value. The number of iterations of GKFCM, instead, significantly decreases when *k > 2*, and is very high for *k=2*. This shows that, in general, GKFCM is much more computationally efficient than FCM, keeping the number of iterations very stable for higher values of *k*. On the contrary, and as seen in previous validations of FCM, it is not very efficient as the number of clusters increase.

The XB index is always lower for FCM compared to GKFCM, showing that the former performs better, especially for *k=3* and *k=4*. The lowest values of XB in GKFCM are those for *k=2* and *k=4*.

As mentioned in Sections 4.3, the higher values of the Silhouette index compliment the results of XB index, that is, that FCM performs better in general. However, Figure 21 and Figure 22 show that GKFCM has found the correct clusters, whereas FCM has not, although the indexes values indicate the other way. This may be because these indexes are ideally defined for spherical clusters, and since the dataset SD3 has non-spherical clusters, the index values do not correctly indicate the performance of the GKFCM algorithm.

Number of clusters (k)	It. FCM	It. GKFCM	XB FCM	XB GKFCM	Sil. FCM	Sil. GKFCM
2	71	161	0.208	0.388	0.713	0.492
3	113	13	0.179	0.416	0.73	0.497
4	45	14	0.085	0.359	0.814	0.397
5	684	14	0.222	1.088	0.752	0.155

Table 6: Validation for Hypothesis 4 using SD3

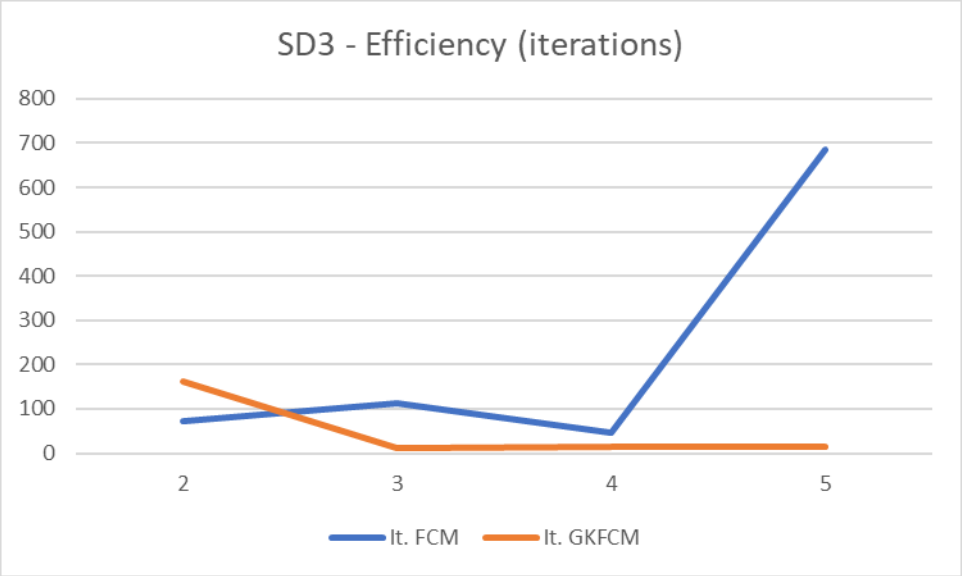

Figure 18. Efficiency of FCM, and GKFCM for SD3

Figure 18 shows that FCM is less efficient than GKFCM for *k = 3, 4, 5*, but more efficient for *k = 2*. The number of iterations for GKFCM are similar for all values of *k*, whereas they are very different in the case of FCM.

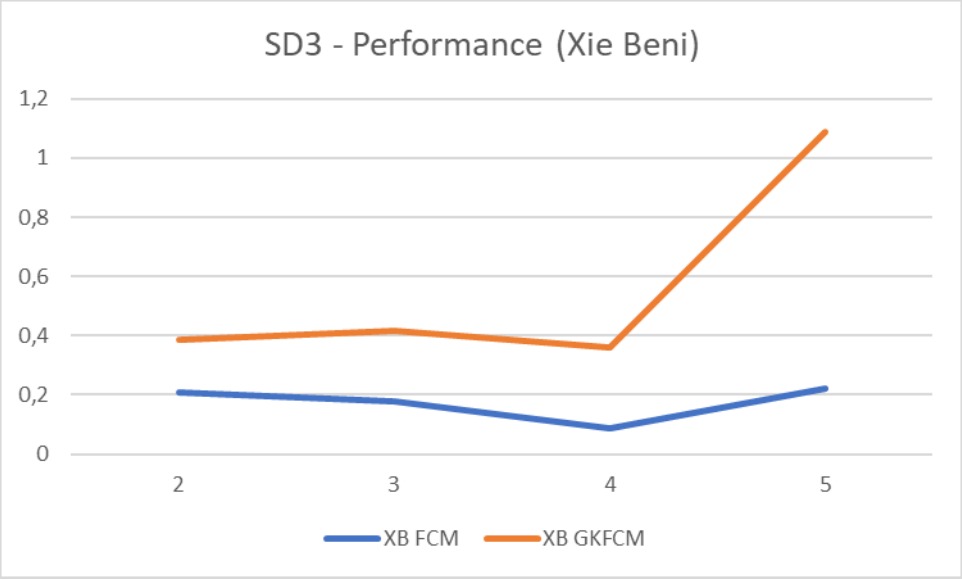

Figure 19. Performance (Xie-Beni) of FCM, and GKFCM for SD3

Figure 19 shows that FCM always performs better than GKFCM, especially for *k = 3* (the actual number of clusters) and *k = 4*.

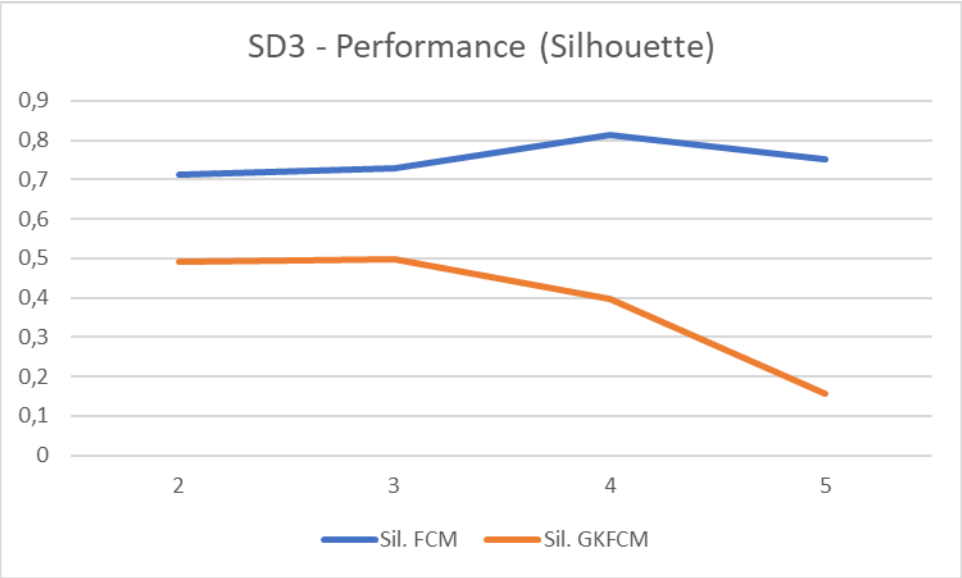

Figure 20. Performance (Silhouette) of FCM and GKFCM for SD3

Figure 20 demonstrates that FCM always performs better than GKFCM, according to Silhouette's index, especially for higher values of *k*.

Observe from Figure 21 and Figure 22 that GKFCM is very good at finding the actual clusters when they are non-spherical. The accuracy is much better for GKFCM than for FCM. On the contrary, FCM does not perform well for non-spherical clusters and tries to fit them based on the assumption that they are overlapping, resulting in inaccurate detection of clusters.

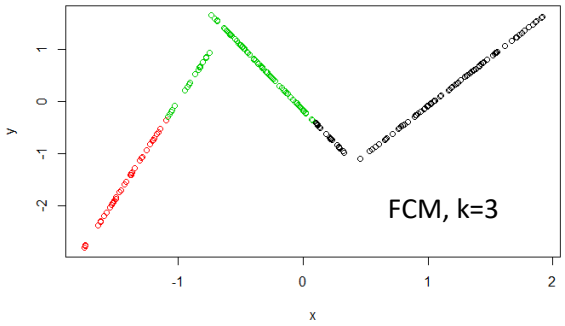

Figure 21. Clusters of SD3 recognised by FCM

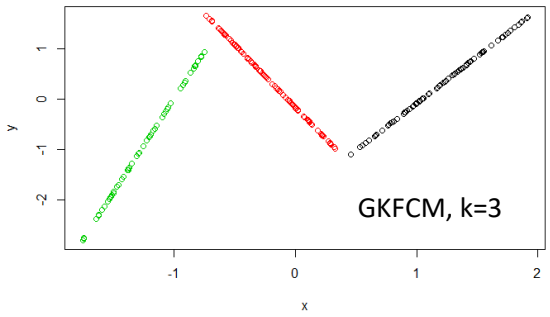

Figure 22. Clusters of SD3 recognised by GKFCM

Table 7 shows the results of the validation on the dataset SD7, which contains non-spherical clusters with noise. As seen on the table, the computational efficiency of GKFCM and FCM do not seem to follow a pattern. This may be because the clusters are not spherical. FCM is not very efficient for *k=2, k=4, and k=5*, whereas its efficiency gets better for the actual number of clusters, *k=3*. Otherwise, GKFCM seems to be more efficient than FCM for all the values of *k*, except for *k=3*.

The XB index indicates that FCM performs better higher values of *k* and GKFCM for *k=3* and *k=4*. Otherwise, in the case of GKFCM, the Silhouette index is better for *k*=3 whereas for FCM, it performs similarly for higher values of *k*. In general, the XB and the Silhouette indexes show that the performance of GKFCM is a bit better for non-spherical and curved clusters Figure 23 and Figure 24 show that both algorithms have found similar clusters.

Therefore, for datasets with non-spherical and curved clusters, GKFCM performs a bit better than FCM while having a similar computational efficiency, but the clusters found by both algorithms are similar.

Number of clusters (k)	It. FCM	It. GKFCM	XB FCM	XB GKFCM	Sil. FCM	Sil. GKFCM
2	1000	668	0.37	0.327	0.577	0.56
3	210	319	0.152	0.181	0.657	0.612
4	1000	1000	0.17	0.168	0.632	0.581
5	1000	356	0.154	0.496	0.619	0.353

Table 7: Validation of FCM and GKFCM using SD7

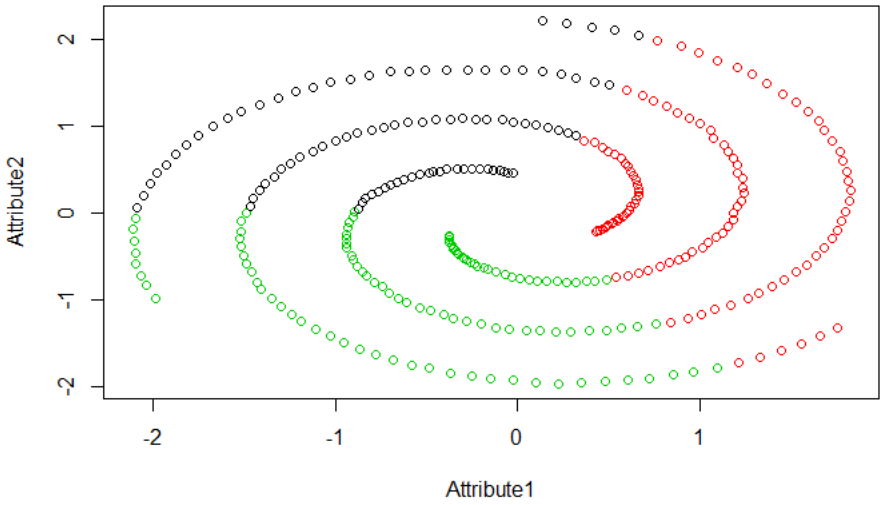

Figure 23. Clusters of SD7 recognised by GKFCM

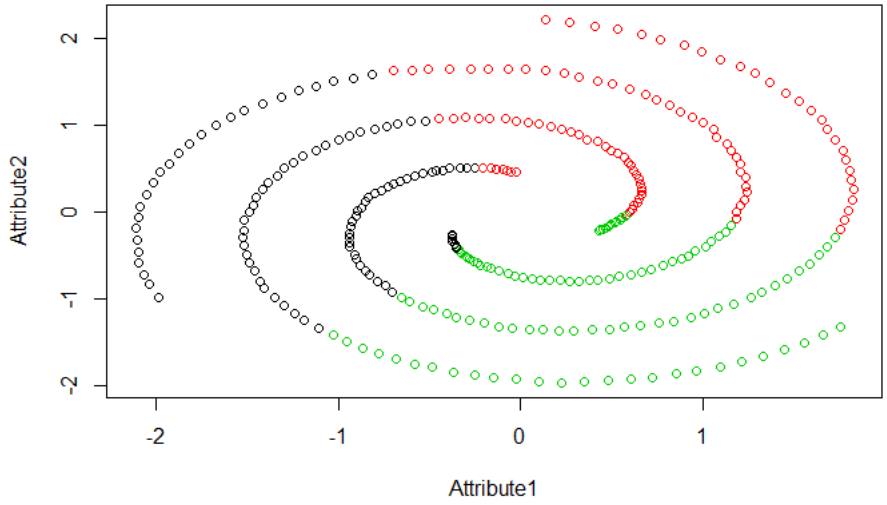

Figure 24. Clusters of SD7 recognised by FCM

Figure 23 and Figure 24 display the results of the GKFCM and FCM algorithms respectively. The clusters found by both algorithms are very similar.

The next experiment involves the dataset SD8, which also contains non-spherical clusters. These clusters, however, are made of more fuzzy observations than the previous two clusters. It allows to validate how good the performance of GKFCM is when the shape of the clusters is made out of a cloud of scattered observations. It will also reveal the performance of FCM under the circumstances of fuzzy non-spherical clusters.

Table 8 shows the results of the validation. The computational efficiency of FCM is similar to that of datasets SD3 and SD7, since it gets worse as the value of *k* increases. In the case of GKFCM, the computational efficiency does not follow a pattern for the values of *k*, although it is better for *k=3*.

The XB index is similar for both FCM and GKFCM when *k=2* and *k=3*. In the case of GKFCM, though, the value of the index increases significantly for *k=3* and *k=4*. Something similar occurs for the Silhouette index. For FCM, it is very similar for all values of *k*, whereas for GKFCM, the highest value is that of the actual number of clusters, *k=2*, and performs worse and worse the higher the value of *k*. This shows that GKFCM works better for the actual number of clusters, when these are non-spherical, and worse for the rest. FCM, instead, works the same for all values of *k*.

However, Figures 25 and Figures 26 show that both algorithms have found similar clusters, so it is likely that the Xie-Beni index and the Silhouette index are not the best tool to validate fuzzy clustering on non-spherical clusters.

Number of clusters (k)	It. FCM	It. GKFCM	XB FCM	XB GKFCM	Sil. FCM	Sil. GKFCM
2	35	380	0.142	0.139	0.744	0.734
3	88	72	0.152	0.186	0.759	0.524
4	91	195	0.105	0.204	0.754	0.56
5	152	405	0.153	0.787	0.734	0.471

Table 8: Validation for Hypothesis 4 using SD8

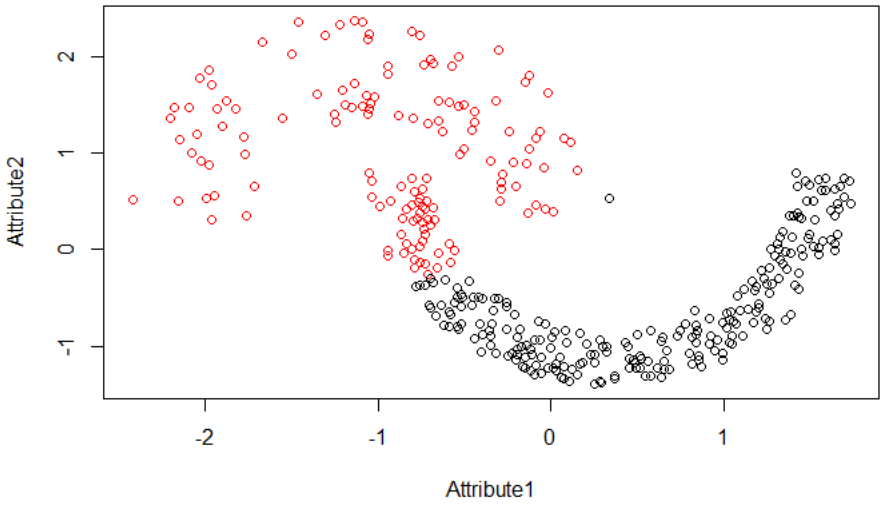

Figure 25. Clusters of SD8 recognised by GKFCM

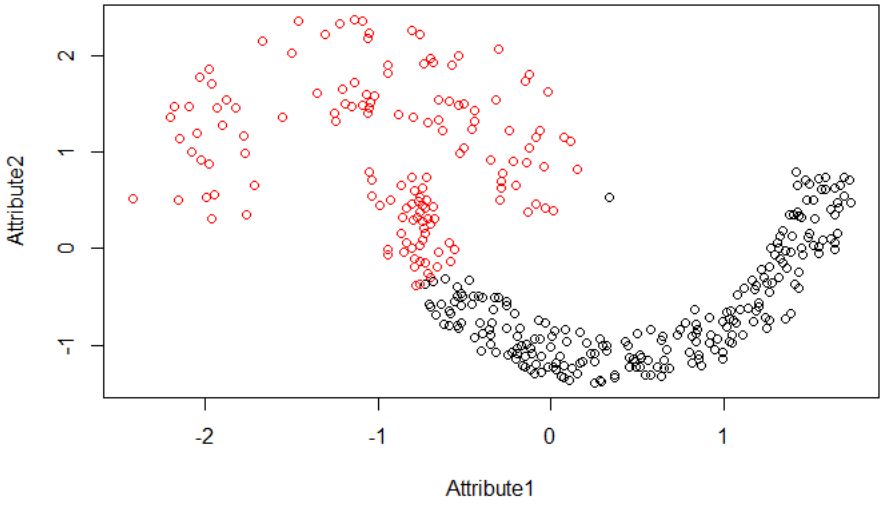

Figure 26. Clusters of SD8 recognised by FCM

The next dataset is SD9, which contains fifteen clusters with outliers. Here I want to see the compare the performance and efficiency of both algorithms on a dataset with spherical clusters.

Table 9 shows the results of the validation, in this case, only for *k=15* because the computational efficiency of the GKFCM working with this dataset is very low. For example, it has taken GKFCM 818 iterations to converge, and only 70 for FCM. Nevertheless, XB index shows that FCM performs worse than, although the Silhouette index is similar in both cases.

Hence, on datasets with several clusters with outliers, GKFCM performs a bit better than FCM, but FCM is much more efficient than GKFCM.

Number of clusters (k)	It. FCM	It. GKFCM	XB FCM	XB GKFCM	Sil. FCM	Sil. GKFCM
15	70	818	1	0.844	0.818	0.85

Table 9: Validation of PCM and GKFCM using SD9

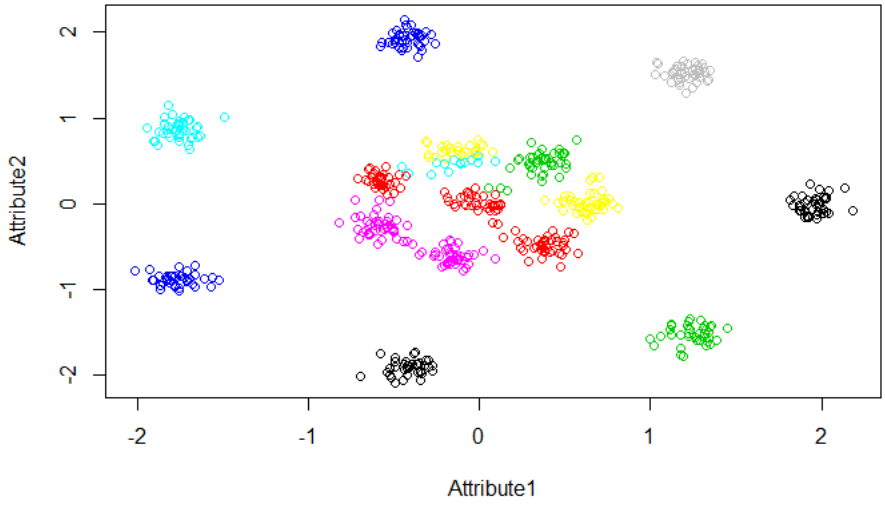

Figure 27. Clusters of SD9 recognised by GKFCM

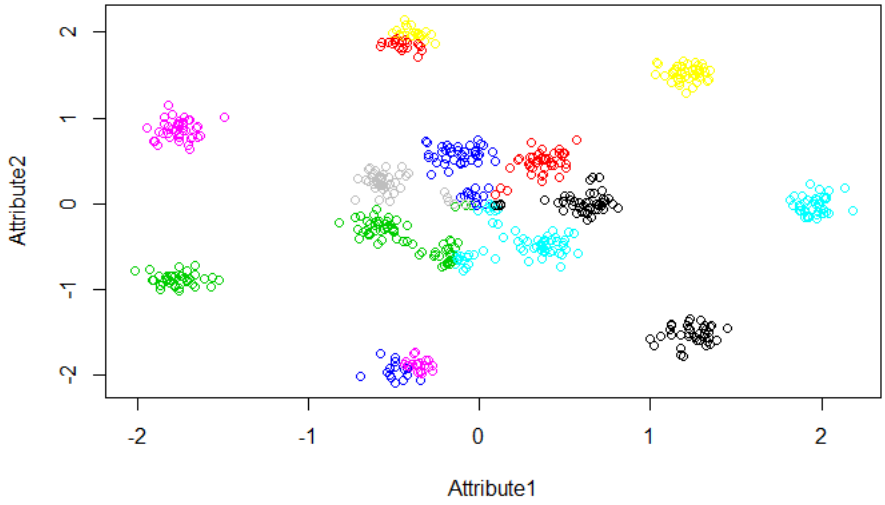

Figure 28. Clusters of SD9 recognised by FCM

Finally, the last validation of this section is on the dataset SD10, which contains three spherical and non-spherical fuzzy clusters. This is an interesting validation to check the performance and the efficiency of both algorithms when the dataset has different kinds of clusters.

Table 10 shows that, for all cases, the computational efficiency decreases as the value of *k* increases, although the efficiency of GKFCM gets much worse. Otherwise, the XB index is similar in both cases, being always lower when *k=3*, the real number of clusters. In the case of the Silhouette index, the values of both algorithms are also similar for all values of *k*. In the case of FCM, however, the bigger value is when *k=3*, and for GKFCM, when *k=4*.

Consequently, for datasets containing both spherical and non-spherical clusters, FCM is much more efficient than GKFCM and performs slightly better. Nonetheless, Figure 29 and Figure 30 show that both algorithms have failed in finding the actual clusters, even though FCM is closer to the real ones, since it has classified better more data points from the non-spherical cluster.

	It. FCM	It. GKFCM	XB FCM	XB GKFCM	Sil. FCM	Sil. GKFCM
2	44	45	0.277	0.257	0.65	0.633
3	61	291	0.14	0.171	0.751	0.668
4	72	707	0.28	0.176	0.71	0.7
5	113	588	0.354	0.357	0.625	0.569

Table 10: Validation of FCM and GKFCM using SD10

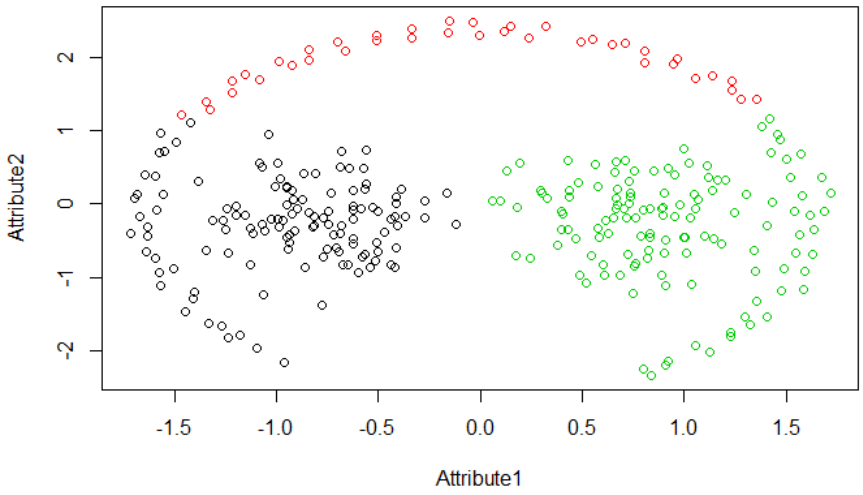

Figure 29. Clusters of SD10 recognised by FCM

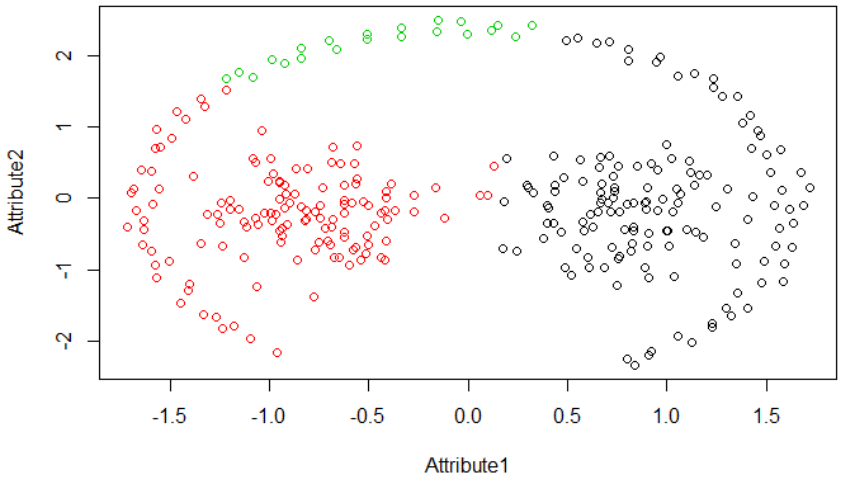

Figure 30. Clusters of SD10 recognised by GKFCM

4.2.4 Fuzzy C-Means and Possibilistic C-Means

Possibilistic C-Means algorithm is based on the possibilistic theory instead of the probabilistic theory of FCM and its derivatives. This algorithm allows to incorporate the noise and outliers in the dataset when the clusters are separated. For that reason, it is supposed to be more accurate than FCM in such cases.

In this group of experiments, I am going to validate this on five different datasets: SD4, SD5, SD6, SD9 and SD10. The first and the second datasets contain two and three well-separated clusters with outliers between them respectively. They will be used to validate the performance of PCM when there are outliers. The third and the fourth dataset contain 31 and 15 clusters with outliers. They will allow me to check if the number of clusters affects the performance of both algorithms. Finally, the last dataset contains two well-separated and spherical clusters with noise and one non-spherical cluster to validate the performance of FCM and PCM under unexpected clusters.

Table 4 shows the efficiency and the accuracy of both FCM and PCM when *k=2*. Notice that the accuracy of PCM is slightly better than that of FCM. However, the number of iterations of FCM are half as many as PCM.

It. FCM	It. PCM	Accuracy FCM	Accuracy PCM
16	38	0,99	0,991

Table 11: Validation for Hypothesis 5 using SD4

Therefore, while the PCM is 1% more accurate than FCM, this different is not significant to say that PCM is better than FCM in the event of outliers. Nevertheless, FCM is much more efficient than PCM.

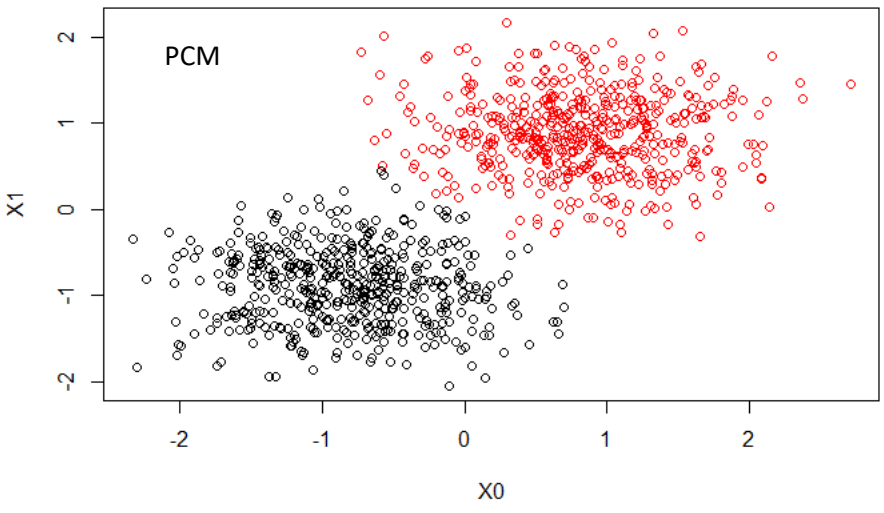

Figure 31. Clusters of SD4 recognised by PCM

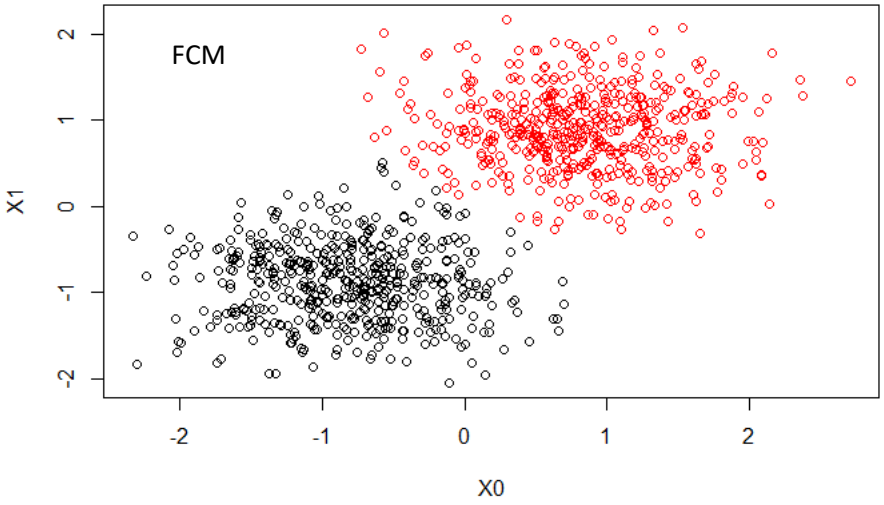

Figure 32. Clusters of SD4 recognised by FCM

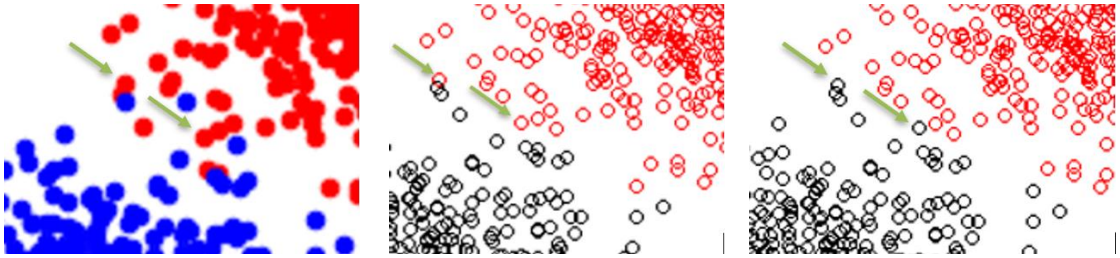

Figure 33. Original classes in SD4 vs PCM and FCM results

Figure 33 shows the magnified views of the clusters found in Figure 31 and Figure 32. The left panel shows the noisy area of the actual dataset and the centre and right panel shows the same area analysed using PCM and FCM, respectively. Observe that on the noisy area, PCM is slightly more accurate than FCM finding the data points belonging to the actual clusters of the dataset, although the difference is not significant. However, the computational efficiency of PCM declines by more than half of the efficiency of FCM. This means that PCM may be a better choice of algorithm when accuracy is more important than efficiency. On the other hand, FCM can be more useful for faster analysis because it is computationally more efficient (less than half the number of iterations compared to PCM) if the accuracy is of less importance.

Table 12 shows the efficiency and the accuracy of both algorithms for the dataset SD5, which has three clusters with outliers in the regions in between the clusters. The FCM has done 49 iterations and the PCM, 194. On the other hand, the accuracy of FCM is much higher than that of PCM. In fact, Figure 34 shows that the majority of observations have been classified in one of the clusters, while a second cluster is made only out of one single green observation in the middle of the figure.

It. FCM	It. PCM	Accuracy FCM	Accuracy PCM
49	194	88.7%	33.5%

Table 12: Validation of PCM and FCM using SD5

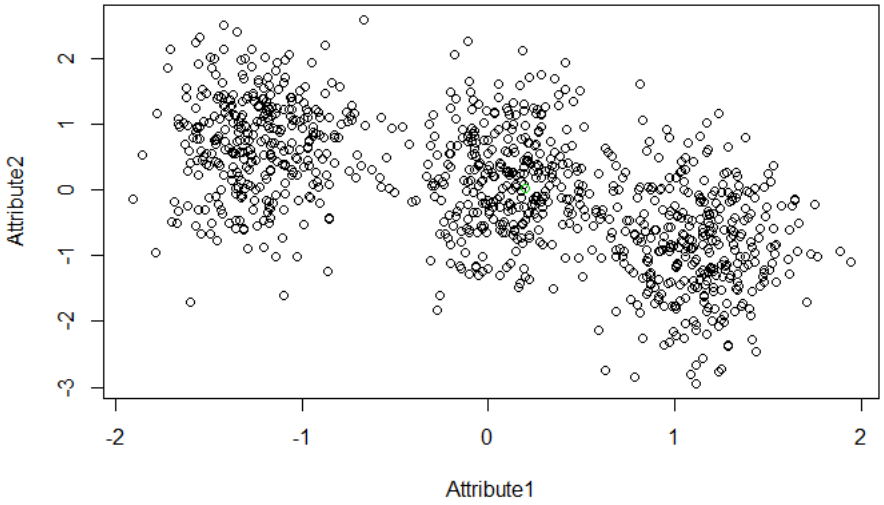

Figure 34. Clusters of SD5 recognised by PCM

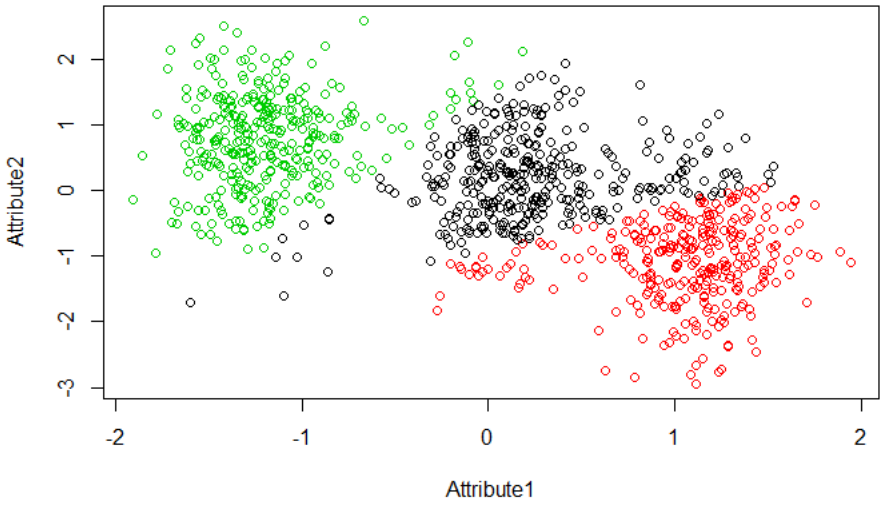

Figure 35. Clusters of SD5 recognised by FCM

Otherwise, Figure 35 shows that FCM has found better clusters than PCM. In conclusion, in the event of datasets with three spherical clusters and outliers, FCM is more efficient and accurate than PCM.

Table 13 shows the validation on the dataset SD6, which contains 31 well-separated spherical clusters with some noise. In this case, FCM has done more than twice as

many iterations as PCM, whereas its accuracy has been 89.7% and 87.7% for PCM, so the difference is not significant.

It. FCM	It. PCM	Accuracy FCM	Accuracy PCM
234	81	89.7%	87,7%

Table 13: Validation using SD6

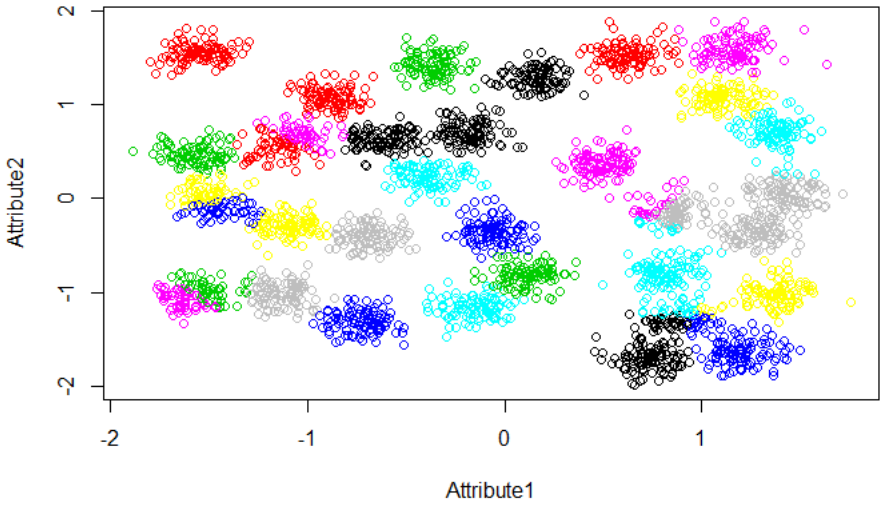

Figure 36. Clusters of SD6 recognised by FCM

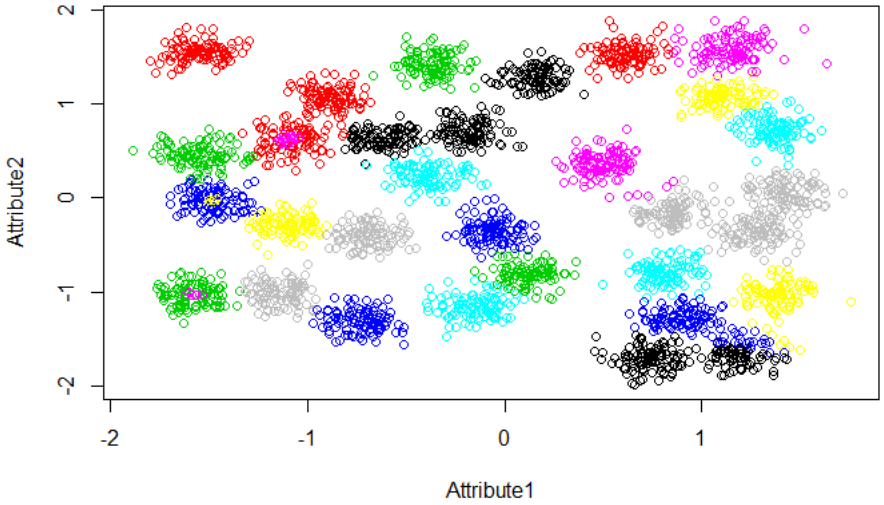

Figure 37. Clusters of SD6 recognised by PCM

Figure 36 and Figure 37 represent the cluster identification of dataset SD6 with FCM and PCM, respectively. Observe that, in general, the PCM is better at identifying clusters in noisy area compared to FCM.

In conclusion, for datasets with spherical and well-separated clusters with outliers, the accuracy of both algorithms is the same, but PCM is significantly more efficient than FCM.

Table 14 shows the results of the validation for the SD9, which has 15 spherical and well-separated clusters with some noise. The accuracy of both algorithms is the same, 99,67%, but FCM takes almost half the iterations to converge compared to PCM.

It. FCM	It. PCM	Accuracy FCM	Accuracy PCM
32	56	99,67%	99,67%

Table 14: Validation using SD9

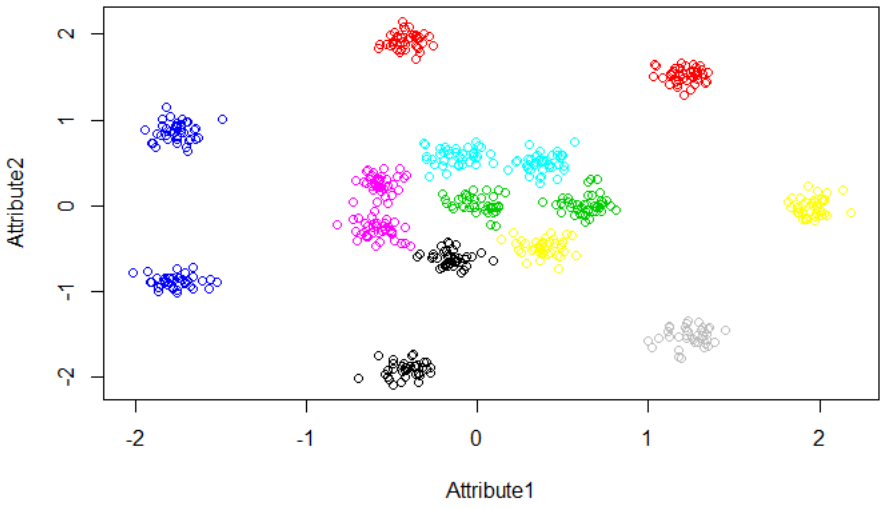

Figure 38. Clusters of SD9 recognised by FCM

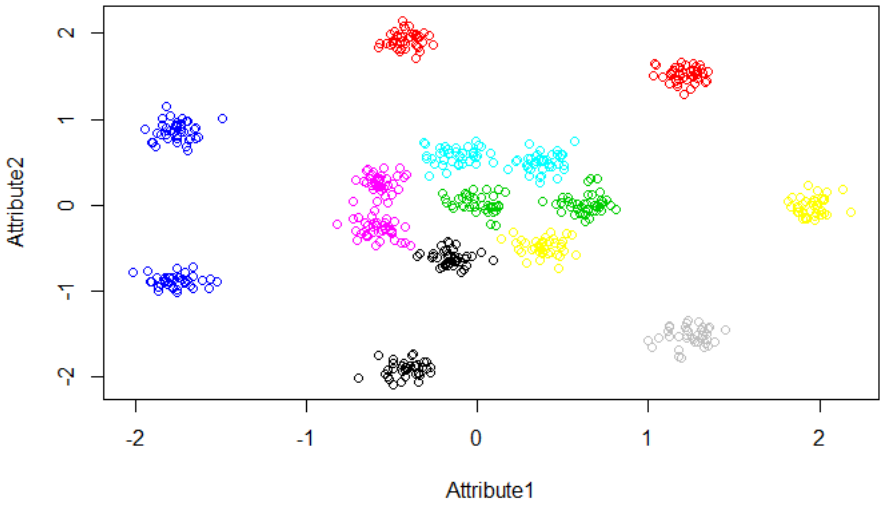

Figure 39. Clusters of SD9 recognised by PCM

Figure 38 and Figure 39 show the cluster identification by FCM and PCM, respectively. As evident by the efficiency, the figures compliment the observation that both algorithms have the same accuracy in finding clusters. Also, since the FCM is computationally more efficient than PCM, the former would be a better choice of an algorithm for this type of datasets.

Finally, Table 15 shows the results of the validation for the dataset SD10, which contains a mixture of spherical and non-spherical clusters with noise. FCM has taken 56 iterations to converge and PCM, 96. On the other hand, the accuracy of FCM has been 76.3% and that of PCM, 65%.

It. FCM	It. PCM	Accuracy FCM	Accuracy PCM
56	96	76.3%	65%

Table 15: Validation using SD10

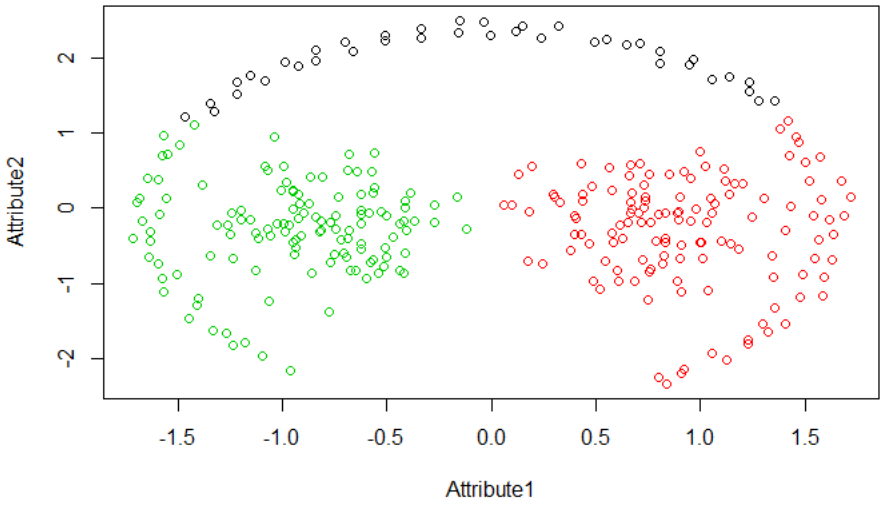

Figure 40. Clusters of SD10 recognised by FCM

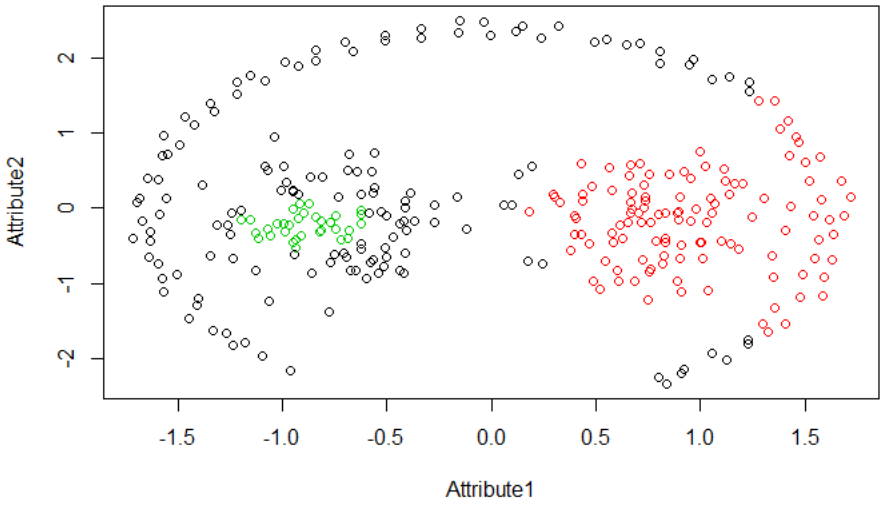

Figure 41. Clusters of SD10 recognised by PCM

Therefore, for datasets with both spherical and non-spherical clusters, FCM is more efficient more accurate than PCM. The latter requires twice as many iterations to converge as FCM.

However, notice from Figure 40 and 41 that neither FCM nor PCM can correctly identify the clusters present in the dataset. On more detail observation, although it appears that FCM does well to identify the spherical clusters present in the dataset, it fails for the non-spherical cluster and tries to find overlapping clusters instead. On the other hand, PCM does not identify either the spherical or the non-spherical clusters of the dataset correctly.

This shows that for datasets with mixed clusters, both FCM and PCM fails to do correct analysis, although PCM fails relatively more miserably than FCM as the latter at least finds the spherical clusters to some extent.

4.3 Discussion

In this section, I have compared different algorithms to investigate the evolution of hard and soft clustering techniques in terms of efficiency, accuracy and performance on datasets with differently shaped clusters.

I have found that KM is definitely a better choice of algorithm for datasets with clearly defined clusters and has higher computational efficiency. FCM also performs as good as KM in the same cases in terms of accuracy, however, its computational efficiency is considerably lower compared to that of KM, making KM a better choice. On the other hand, FCM is generally a better algorithm for most cases involving datasets with overlapping clusters. Also, I have found that KM does perform well when overlapping clusters are present, and its computational efficiency is still better than that of FCM. This leads me to question whether the little improvement of accuracy offered by FCM for overlapping clusters is worth the significant reduction of computational efficiency. Of course, this requires more comprehensive study of both these algorithms, which is beyond the scope of this project.

As for the comparison between FCM and its derivatives FCM++ and S-FCM, the performance of the first two improves when clusters overlap more. However, their computation efficiency gets worse as the number of clusters increase. It is interesting to note that the performance of the three algorithms become similar when there is less overlap between clusters, however, SFCM is always computationally more efficient compared to the other two. Therefore, when the clusters do not strongly overlap then SFCM is a better choice because of its highly invariable computational efficiency with number of clusters in the dataset.

For the comparison of the GKFCM and FCM, I have used five different datasets: three non-spherical, one spherical and one mixed. The dataset with non-spherical and clear clusters, the GKFCM is much more computationally efficient than FCM, whereas FCM has performed better. However, the accuracy of GKFCM is found to be much better than that of FCM. This may be because the indexes used to validate this dataset are defined for spherical datasets, which may not be ideal for this case. On the other hand,

both algorithms show similar performance for datasets with non-spherical and noisy clusters, although GKFCM is slightly more efficient. It is important to note that they both fail to find the actual clusters. Finally, for non-spherical and fuzzy clusters, GKFCM is clearly less efficient than FCM, whereas its performance is better than that of FCM. Once again, they both fail to find the actual clusters, reconfirming the scepticism over the use of the indexes to validate non-spherical clusters. On the contrary, for spherical clusters with noise, GKFCM is significantly less efficient than FCM, although its performance in general is better. They both have similar accuracy in finding clusters. Therefore, FCM is a better choice because is more computationally efficient. At last, for mixed clusters, GKFCM is less efficient than FCM, yet the performance is similar in both cases. However, FCM seems to be more accurate in finding the actual clusters. In conclusion, GKFCM is a better choice for non-spherical and clear clusters while FCM is better for spherical clusters.

Finally, between PCM and FCM for non-overlapping and noisy clusters, I did not find any conclusive answer about the better choice between the two. In general, the accuracy of PCM is slightly better than FCM, but its computational efficiency is significantly lower. For the case of a dataset with three separated and noisy clusters, the PCM completely fails to identify correct cluster, whereas the FCM succeeds contrary to expectation. However, PCM is considerably better than FCM in terms of computational efficiency when the dataset contains 31 clusters by keeping the same accuracy as FCM. However, for the dataset with 15 clusters, FCM has been more efficient and accurate than PCM, so this needs further investigation in order to find out which is the ideal number of clusters under which PCM is better than FCM. Finally, for mixed clusters, FCM outperforms PCM in both accuracy and computational efficiency, although they both fail to correctly identify the actual clusters. In conclusion, the choice between FCM and PCM is not straightforward and appears to depend on the case by case basis.

5 Conclusions

During this project, I have reviewed the history of clustering and the main contributions to the field in order to follow what the improvements of each algorithm has been over the rest. I have validated these improvements in a series of experiments on six clustering algorithms, K-Means, Fuzzy C-Means, Fuzzy C-Means++, Suppressed-Fuzzy C-Means, Gustafson Kessel Fuzzy C-Means and Possibilistic C-Means.

K-Means was one of the first clustering algorithms. It was first formulated in 1956 and its aim was to become a tool to find groups in data based on the sum-of squares criterion. Even though KM is accurate in finding groups when the data contains well-separated clusters, it is not so much when they are overlapping.

To solve this problem, Bezdek defined the Fuzzy C-Means algorithm in 1973 based on the KM. FCM was able to tell the degree of belongingness of each observation to every cluster, thus being more accurate with overlapping clusters. However, FCM does not perform well on datasets with different shapes, because it assumes that they all are spherical. Also, the convergence speed of FCM is very slow compared to KM.

Since then, other authors have formulated several algorithms in order to improve FCM. Some of the most important contributions are the Gustafson Kessel Fuzzy C-Means and the Possibilistic C-Means.

Gustafson and Kessel created GKFCM in 1978, being the first one explicitly created to improve FCM. Unlike FCM, GKFCM adapts to the shape of every cluster in a dataset, and it is more accurate especially when clusters are non-spherical.

Otherwise, PCM was defined by Krishnapuram and Keller in 1993. Unlike FCM, where the sum of all the degrees of belongingness of an observation to all the clusters must be equal to 1, PCM does not have this constraint. In PCM, a degree is assigned to each data point based on how close it is to each cluster so, for example, the outliers may not belong to any of them. This makes PCM a better choice for clusters containing noise and outliers.

More recently, there have been some other attempts to improve the FCM's computational efficiency. I have chosen two of the more recent, which are Fuzzy C-Means++ and Suppressed-Fuzzy C-Means.

Throughout this project, I have designed a series of experiments to validate the improvements of each algorithm over another. I have found that FCM is more accurate than KM for datasets with overlapping clusters, although the difference is not very high. Also, KM is significantly more computationally efficient than FCM in all cases. Moreover, I have discovered that FCM is still the fuzzy algorithm that performs better for spherical clusters, although S-FCM improves its efficiency. On the contrary, FCM++ does not show any significant advantage over FCM. On the other hand, I have validated that, in some cases, GKFCM performs better than FCM when clusters are non-spherical. Finally, PCM has been more accurate than FCM when the number of clusters is very high, but it has not shown any advantage when the number of clusters is low.

This work aims at being a starting point for further investigations about clustering to better define when it is adequate to choose one algorithm or another. Although I have mainly used synthetic datasets, further studies are required with real-world datasets to test the algorithms under unexpected situations. Also, it will be interesting to use other parameters for the validations.

6 Bibliography

[1] Ruspini EH, Bezdek JC, Keller JM (2019) ‘Fuzzy Clustering A Historical Perspective’, IEEE Comp. Int. Mag. 14(1): 45-55

[2] Cebeci Z, Yildiz F (2015) ‘Comparison of K-Means and Fuzzy C-Means Algorithms on Different Cluster Structures’, Journal of Agricultural Informatics. Vol. 6, No. 3:13-23

[3] Pakhira MK, Bandyopadhyay S, Maulik U (2004) ‘Validity index for crisp and fuzzy clusters’, Pattern Recognition 37(3):487-501

[4] Bock (2017) Clustering Methods: A History of k-Means Algorithms, in Studies in Classification, Data Analysis, and Knowledge Organization, pp. 161-172

[5] MacQueen, JB (1967) ‘Some Methods for Classification and Analysis of Multivariate Observations’. Proc. of 5th Berkeley Symp. on Mathematical Statistics and Probability, Berkeley, University of California Press, 281-297.

[6] Ali, MA, Karmakar, GC & Dooley, LS (2008) ‘Review on Fuzzy Clustering Algorithms’. IETECH Journal of Advanced Computations, vol. 2, no. 3, 169 – 181

[7] Suganya, R & Shanthi, R (2012) ‘Fuzzy C- Means Algorithm - A Review’. Int. J. of Scientific and Research Publications, vol. 2, no. 11, 1-3

[8] Winkler R, Klawonn, F, Kruse R (2010) ‘Fuzzy c-means in high dimensional spaces’, International Journal of Fuzzy System Applications, 1(1), 1-16

[9] Fan JL., Zhen WZ, Xie WX (2003) ‘Suppressed Fuzzy c-means Clustering Algorithm’, Pattern Recognition Letters 24, 1607–1612

[10] Ali A, Karmakar GC and Dooley LS (2004) ‘Fuzzy Image Segmentation using Suppressed Fuzzy C-Means Clustering’. In: 7th International Conference on Computer and Information Technology, 26-28 Dec 2004, Dhaka, Bangladesh

[11] Gustafson DE and Kessel WC (1978). ‘Fuzzy clustering with a Fuzzy covariance matrix’. IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes, 17, 761–766

[12] Steinhaus H (1956), ‘Sur la division des corps matériels en parties. Bulletin de l’Académie Polonaise des Sciences’, Classe III, vol. IV, no. 12, 801-804.

[13] Stetco A, Zeng X-J, and Keane J (2015) ‘Fuzzy C-means++: Fuzzy C-means with effective seeding initialization’. Expert Systems with Applications. 42(21): p. 7541-7548

[14] Siddique et al (2018) ‘Implementation of Fuzzy C-Means and Possibilistic C-Means Clustering Algorithms, Cluster Tendency Analysis and Cluster Validation’, arXiv:1809.08417v2 [cs.LG]

[15] Gan G, Lan Q, and Sima S (2016) ‘Scalable Clustering By Truncated Fuzzy C-Means’, Big Data and Information Analytics, American Institute of Mathematical Sciences, Volume 1, Number 2&3, pp. 247–259

[16] Georgieva O and Filev D (2009) ‘Gustafson-Kessel Algorithm for Evolving Data Stream Clustering’, International Conference on Computer Systems and Technologies, CompSysTech’09

[17] Krishnapuram R and Keller J (1993) ‘A possibilistic approach to clustering’, IEEE Trans. Fuzzy Syst., vol. 1, no. 2, pp. 98–110.

[18] Liu Y et al (2010) ‘Understanding of Internal Clustering Validation Measures’, 2010 IEEE International Conference on Data Mining, pp. 911-916

[19] Kannan SR, Ramathilagam S, and Chung PC (2012) ‘Effective fuzzy c-means clustering algorithms for data clustering problems’, Expert Systems with Applications 39 (2012) 6292–6300

[20] Jain, AK (2009) ‘Data clustering 50 years beyond K-means’, Pattern Recognition Letters, Volume 31, Issue 8, 1 June 2010, Pages 651-666

[21] Krishna Priya CB, Venkateswari S (2017) ‘Application of Gustafson-Kessel-like clustering algorithm in Delineation of management Zones in precision Agriculture’, International Journal of Applied Agricultural Research, Volume 12, Number 3 pp. 279-293

[22] Cebeci Z, Kavlak AT, Yildiz F (2017) ‘Validation of Fuzzy and Possibilistic Clustering Results’, International Artificial Intelligence and Data Processing Symposium (IDAP)

[23] Liu et al (2010) ‘Understanding of Internal Clustering Validation Measures’, ICDM '10 Proceedings of the 2010 IEEE International Conference on Data Mining, Pages 911-916

[24] Babuska R, Veen PJ, Kaymak U (2002) ‘Improved covariance estimation for Gustafson-Kessel clustering’, in Proceedings of the 2002 IEEE international conference on fuzzy systems (pp. 1081-1085)

[25] Rendón E et al. (2011) ‘A comparison of internal and external cluster validation indexes’, Applications of Mathematics and Computer Engineering

[26] Veenman CJ, Reinders MJT and Backer E (2002) A maximum variance cluster algorithm. IEEE Trans. Pattern Analysis and Machine Intelligence. 24(9): p. 1273-1280.

[27] Jain A and Law M (2005) Data clustering: A user's dilemma. Lecture Notes in Computer Science. 3776: p. 1-10.

[28] Chang H and Yeung DY (2008) Robust path-based spectral clustering. Pattern Recognition, 2008. 41(1): p. 191-203.