An investigation into fuzzy clustering quality and speed: Fuzzy C-means with effective seeding

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Science and Engineering

2016

By

Mihai Adrian Stetco

School of Computer Science
Contents

ABSTRACT.....................................................................................................................10

DECLARATION..............................................................................................................11

COPYRIGHT..................................................................................................................12

ACKNOWLEDGEMENTS ...............................................................................................13

ABBREVIATIONS AND ACRONYMS............................................................................14

CHAPTER 1 INTRODUCTION.........................................................................................15

1.1 Motivation...............................................................................................................15

1.2 Aims and contributions ..........................................................................................18

1.3 Structure ................................................................................................................20

1.4 Publications ............................................................................................................21

CHAPTER 2 BACKGROUND..........................................................................................22

2.1 Introduction..............................................................................................................22

2.2 Cluster Analysis .....................................................................................................22
    2.2.1 Hierarchical algorithms .......................................................................................27
    2.2.2 Partitional algorithms ..........................................................................................29
    2.2.3 Density-based algorithms ..................................................................................30
    2.2.4 Grid-based algorithms ......................................................................................32
    2.2.5 Methods based on co-occurrence of categorical data .......................................33
    2.2.6 Other clustering algorithms ...............................................................................34
### 2.3 Fuzzy C-means

2.3.1 Origins .................................................. 35

2.3.2 Fuzzy C-means: The algorithm ................................................. 36

2.3.3 Improvements of FCM .......................................................... 39

### 2.4 Summary ................................................................. 47

### CHAPTER 3 FCM++: FUZZY C-MEANS WITH EFFECTIVE SEEDING......48

3.1 Introduction ....................................................................... 48

3.2 FCM++ description ........................................................... 50

3.3 Synthetic dataset evaluation .................................................. 52

3.3.1 Equal sized clusters that do not overlap ................................... 54

3.3.2 Equal sized clusters that overlap ........................................... 56

3.3.3 Different sized clusters that do not overlap ................................. 58

3.3.4 Different sized clusters that overlap ........................................ 60

3.4 Real world dataset evaluation ................................................ 62

3.4.1 Iris dataset ...................................................................... 62

3.4.2 Spam dataset .................................................................... 65

3.4.3 Wine dataset .................................................................... 67

3.4.4 Life expectancy dataset ....................................................... 70

2.4.5 Transfusion dataset .......................................................... 72

3.5 Summary ............................................................................ 74

### CHAPTER 4 TIME SERIES CASE STUDIES .................................. 76

4.1 Introduction ....................................................................... 76

4.2 Motivation .......................................................................... 76

4.3 Time series terminology ....................................................... 77
List of Tables

Table 2.1. Similarity measures used in clustering [19] .........................................................23
Table 2.2 Clustering methods classification [12] .....................................................................25
Table 3.1 Iris dataset results ..................................................................................................64
Table 3.2 Spam dataset results ...............................................................................................67
Table 3.3 Wine dataset results ...............................................................................................68
Table 3.4 Life Expectancy dataset results ..............................................................................70
Table 3.5 Transfusion dataset results ......................................................................................72
Table 4.1 Guideline on interpreting correlations between time series [7] .........................78
Table 4.2 FTSE Data .............................................................................................................93
Table 4.3 Sharpe Ratio of the stocks .....................................................................................95
Table 4.4 Returns on investment and Sharpe ratio for each cluster ....................................105
List of Figures

Fig 1.1 A time series switching position [14]..............................................................17
Fig 2.1 The cluster analysis process [19]......................................................................24
Fig 2.2. Hierarchical Clustering: each node represents a cluster and each leaf node represents a singleton cluster [1]. Each tree level represents a segmentation of the data and a clustering is obtained by cutting the tree at the desired level. The correct cutoff level can be determined using either (or both) context information and validity indexes........................................27
Fig 2.3 DBSCAN [28] Density of cluster p (core) is high (MinPts=4) while density of q (border) is low (MinPts<4)..................................................................................30
Fig 2.4. Clusters with respect to different density parameters [42].................................31
Fig 2.5.Two of the subspaces of the data [2]...............................................................32
Fig 2.6 The intersection of the subspaces shown in 3D [2].............................................33
Fig 2.7 An example of graph clustering showing three clusters [3]..............................34
Fig 2.8 Membership function assignment of a person based on age: in red we see a hypothetical membership function called YOUNG while in yellow one named OLD.................................................36
Fig 2.9 FCM Algorithm............................................................................................37
Fig 2.10 Partitional Cluster Analysis...........................................................................38
Fig 2.11 Five clusters of radius λ separated by space of size B [99]..............................40
Fig 2.12 Initialization with random centers (black dots) [99]......................................41
Fig 2.13 Initialization with random centers (black dots) [99]......................................41
Fig 2.14 The Mountain method for center initialization [100]........................................41
Fig 2.15. The mountain function at the first step......................................................42
Fig 2.16. The mountain function after the first cluster is removed [100].....................42
Fig 2.17 Example of Mountain method applied to three clusters positioned in a triangle shape [100]....................................................................................43
Fig 2.18. Subtractive clustering on an artificial 3-dimensional dataset: 4 of the 5 centers were correctly placed (r_a=0.25 with 5 iterations). ..................................................44
Fig 2.19. FCM++ on the artificial 3-dimensional dataset: 5 of the 5 centers were placed closer to the clusters real center.................................................................46
Fig 3.1 FCM++ algorithm..........................................................................................50
Fig 3.2. Equal sized clusters with no overlap. Real cluster centers are marked in magenta, FCM initial clusters are marked in red, while in the yellow and green triangles we have FCM++ (with p=0.5 and p=1.8 respectively). ..................................................54
Fig 3.3 Mean number of iterations on equal sized clusters with no overlap, function of p. ..................................................................................................................55
Fig 3.4 FCM vs FCM++ Partition Entropy Index on equal sized clusters with no overlap........55
Fig 3.5 FCM vs FCM++ Xie Beni Index on equal sized clusters with no overlap............55
Fig 3.6 Equal sized clusters with some overlap. Real clusters are marked in magenta, FCM initial clusters are marked in red, while in the yellow and green triangles we have FCM++(with p=0.5 and p=1.8 respectively)..............................................................................55
Fig 3.7. Mean number of iterations on equal sized clusters with overlap function of p. ......57
Fig 3.8 FCM vs FCM++ Partition Entropy Index on equal sized clusters with some overlap on equal sized clusters with some overlap................................................................57
Fig 3.9 FCM vs FCM++ Xie Beni Index on equal sized clusters with some overlap on equal sized clusters with some overlap..........................................................................................................................57
Fig 3.10. Different sized clusters with no overlap. Real clusters are marked in magenta, FCM initial clusters are marked in red, while in the yellow and green triangles we have FCM++ (with p=0.5 and p=1.8 respectively)..........................................................................................................................58
Fig 3.11. Mean number of iterations on different sized clusters with no overlap, function of p..........................................................................................................................................................................................59
Fig 3.12 FCM vs FCM++ Partition Entropy Index on different sized clusters with no overlap on different sized clusters with no overlap..........................................................................................................................59
Fig 3.13 FCM vs FCM++ Xie Beni Index on different sized clusters with no overlap on different sized clusters with no overlap..........................................................................................................................59
Fig 3.14. Different sized clusters with overlap. Real clusters are marked in magenta, FCM initial clusters are marked in red, while in yellow and green triangles we have FCM++ (with p=0.5 and p=1.8 respectively)..........................................................................................................................60
Fig 3.15. Mean number of iterations on different sized clusters with overlap function of p...61
Fig 3.16 FCM vs FCM++ Partition Entropy on different sized clusters with some overlap on different sized clusters with some overlap..........................................................................................................................61
Fig 3.17 FCM vs FCM++ Xie Beni Index on different sized clusters with some overlap on different sized clusters with some overlap..........................................................................................................................61
Fig 3.18. MDS on Iris Dataset shows a clear separation between setosa in red and the cloud of points made up by virginica (green) and versicolor (blue) ..................................................................................................................................63
Fig 3.19. MDS on Spam dataset where spam massages are shown in black while spam shown in red...........................................................................................................................................................................................................63
Fig 3.20. MDS on Wine Dataset shows an overlap between the clusters.................................66
Fig 3.21. MDS on Life expectancy dataset.....................................................................................71
Fig 3.22. MDS on Transfusion dataset..........................................................................................73
Fig 4.2. Example of correlation between time series........................................................................79
Fig 4.3. Anscombe’s quartet. ..............................................................................................................80
Fig 4.4 Two uncorrelated time series having an upward trend........................................................81
Fig 4.5 The same time series with the trend removed....................................................................81
Fig 4.6 Two correlated time series having different trends............................................................82
Fig 4.7 The same time series having the trend removed..................................................................82
Fig 4.8 Aberdeen Asset Management (AND.L) over 250 days with linear regression in red 83
Fig 4.9 Aberdeen Asset Management (AND.L) with trend removed.............................................83
Fig 4.10 “Augmented Dickey-Fuller Test” on AND.L showing stationarity .................................84
Fig 4.11. Two correlated time series in Cluster 1............................................................................85
Fig 4.12. Two correlated time series in Cluster 10..........................................................................85
Fig 4.13 A sample of 10 out of 1000 synthetically generated time series.......................................85
Fig 4.14 MDS scaled representation of synthetic time series dataset.............................................86
Fig 4.15 Number of iterations FCM (in red) vs FCM++ (with p=1.8 in blue and p=5 in magenta)..........................................................................................................................................................................................87
Fig 4.16 Number of iterations FCM (in red) vs FCM++ (with p=1.8 in blue and p=5 in magenta)..........................................................................................................................................................................................88
Fig 4.17 Xie-Beni validity index: FCM in red, FCM++ (p=5) in blue on the lower representation of the synthetic time series dataset..........................................................................................................................88
Fig 4.18 FTSE companies stock prices over the period 1/1/2015-31/12/2015..............................89
Fig 4.19 Stock prices over the period 1/1/2015-31/12/2-15 with trend removed
Fig 4.20 Differenced Schroder’s vs Standard Life over the year 2015
Fig 4.21 Multi-dimensional scaled time series
Fig 4.22 Worst (GLEN.L in red) versus best (HL.L in blue) performing stocks in terms of Sharpe Ratio
Fig 4.23 FCM (red) vs FCM++(blue, p=1.8) Xie Beni Validity index for different k values
Fig 4.24 FCM (red) vs FCM++(blue with p=1.8) Partition Entropy for different K values
Fig 4.25 FCM (red) vs FCM++ (blue with p=1.8) for k=12 over 10 independent runs
Fig 4.26 FCM (red) vs FCM++ (blue with p=1.8) independent runs for k=12
Fig 4.27 Multi-dimensional scaled time series clustered into k=12 clusters
Fig 4.28 Stock entropy sorted by value
Fig 4.29 Entropy distribution for synthetic (red) and financial (blue)
Fig 4.30 Outliers shown in red
Fig 4.31 Evolution of cluster returns when £1 is invested in each cluster
Fig 5.1 Text Mining Methodology
Fig 5.2 The Guardian website
Fig 5.3 Image in text
Fig 5.4 Top terms in our news data set
Fig 5.5 Terms correlated with “Allardyce”
Fig 5.6 Three documents represented in two dimensional space a red dots. The axes capture the count of either London (X Axis) or Manchester (Y Axis) in the documents
Fig 5.7 Three normalized documents represented in two dimensional space as red dots. The axes capture the count of either London (X Axis) or Manchester (Y Axis) in the documents
Fig 5.8 Un-normalized MDS scaled Guardian news articles represented in three dimensional space. Points in black represent Football news, those in red represent news on immigration, those in green represent news on environment and those in blue represent news on money.
Fig 5.9 Unit Sphere: normalized scaled Guardian news articles represented in three dimensional space. Points in black represent Football news, red represents news on immigration, green represents news on environment and blue represents news on money
Fig 5.10 Using SFCM++ initialization with resulting centers shown in magenta
Fig 5.11 Number of iterations SFCM (red) vs SFCM++(blue)
Fig 5.12 Xie Beni SFCM (red) vs SFCM++(blue)
Fig 5.13 Final cluster centers location
Fig 5.14 Partition Entropy SFCM (red) vs SFCM++(blue)
Fig 6.1 Three Clusters with initial randomly selected point marked in red
Fig 6.2 The distance distribution plot
Fig 6.3 Three Clusters with initial randomly chosen point in red and the rest being picked according to the density plot in Fig 6.2
Fig 6.4 Three Clusters with initial randomly selected point marked in red

Fig 6.5 The distance distribution plot

Fig 6.6 Three clusters showing the potential of each point

Fig 6.7 Euclidean distance between A and B is higher than that between B and C

Fig 6.8 Euclidean distance on autocorrelation coefficients of A and B is lower than that between B and C

Fig 6.9 Euclidean distance between A and B is higher than that between B and C

Fig 6.10 Euclidean distance on autocorrelation coefficients of A and B is lower than that between B and C

Fig 6.11 A Spark Based Distributed Architecture for online FCM++ for time series clustering
Abstract

Cluster analysis, the automatic procedure by which large data sets can be split into similar groups of objects (clusters), has innumerable applications in a wide range of problem domains. Improvements in clustering quality (as captured by internal validation indexes) and speed (number of iterations until cost function convergence), the main focus of this work, have many desirable consequences. They can result, for example, in faster and more precise detection of illness onset based on symptoms or it could provide investors with a rapid detection and visualization of patterns in financial time series and so on. Partitional clustering, one of the most popular ways of doing cluster analysis, can be classified into two main categories: hard (where the clusters discovered are disjoint) and soft (also known as fuzzy; clusters are non-disjoint, or overlapping). In this work we consider how improvements in the speed and solution quality of the soft partitional clustering algorithm Fuzzy C-means (FCM) can be achieved through more careful and informed initialization based on data content. By carefully selecting the cluster centers in a way which disperses the initial cluster centers through the data space, the resulting FCM++ approach samples starting cluster centers during the initialization phase. The cluster centers are well spread in the input space, resulting in both faster convergence times and higher quality solutions. Moreover, we allow the user to specify a parameter indicating how far and apart the cluster centers should be picked in the dataspace right at the beginning of the clustering procedure. We show FCM++’s superior behaviour in both convergence times and quality compared with existing methods, on a wide range of artificially generated and real data sets.

We consider a case study where we propose a methodology based on FCM++ for pattern discovery on synthetic and real world time series data. We discuss a method to utilize both Pearson correlation and Multi-Dimensional Scaling in order to reduce data dimensionality, remove noise and make the dataset easier to interpret and analyse. We show that by using FCM++ we can make an positive impact on the quality (with the Xie Beni index being lower in nine out of ten cases for FCM++) and speed (with on average 6.3 iterations compared with 22.6 iterations) when trying to cluster these lower dimensional, noise reduced, representations of the time series. This methodology provides a clearer picture of the cluster analysis results and helps in detecting similarly behaving time series which could otherwise come from any domain.

Further, we investigate the use of Spherical Fuzzy C-Means (SFCM) with the seeding mechanism used for FCM++ on news text data retrieved from a popular British newspaper. The methodology allows us to visualize and group hundreds of news articles based on the topics discussed within. The positive impact made by SFCM++ translates into a faster process (with on average 12.2 iterations compared with the 16.8 needed by the standard SFCM) and a higher quality solution (with the Xie Beni being lower for SFCM++ in seven out of every ten runs).

Index Terms—Cluster Analysis, Fuzzy C-means clustering, initialization, financial data, text data
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on Presentation of Theses.
Acknowledgements

I owe my sincere gratitude to my supervisors Professor John Keane and Professor Xiao-Jun Zeng for their patience, support and guidance during my research studies.

This thesis would not have been possible without the support of the Engineering and Physical Science Research Council (EPSRC) who funded my research via their doctoral training grant.

Last but not least, I want to thank the researchers and institutions whose open source software packages and tools aided my research studies.
# Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>ADF</td>
<td>Augmented Dickey-Fuller</td>
</tr>
<tr>
<td>akFCM</td>
<td>Approximate Kernel FCM</td>
</tr>
<tr>
<td>brFCM</td>
<td>Bit Reduced Fuzzy C-means</td>
</tr>
<tr>
<td>FCM</td>
<td>Fuzzy C-means</td>
</tr>
<tr>
<td>FCM++</td>
<td>Fuzzy C-means with effective seeding</td>
</tr>
<tr>
<td>FTSE100</td>
<td>Financial Times Stock Exchange 100 Index</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphic Processing Unit</td>
</tr>
<tr>
<td>IDF</td>
<td>Inverse document frequency</td>
</tr>
<tr>
<td>KD</td>
<td>Knowledge discovery</td>
</tr>
<tr>
<td>KM</td>
<td>K-means</td>
</tr>
<tr>
<td>KM++</td>
<td>K-means with effective seeding</td>
</tr>
<tr>
<td>kKM</td>
<td>Kernel K-means</td>
</tr>
<tr>
<td>kFCM</td>
<td>Kernel FCM</td>
</tr>
<tr>
<td>LSE</td>
<td>London Stock Exchange</td>
</tr>
<tr>
<td>MDS</td>
<td>Multi-Dimensional Scaling</td>
</tr>
<tr>
<td>MRI</td>
<td>Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>oFCM</td>
<td>Online FCM</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>PE</td>
<td>Partition Entropy</td>
</tr>
<tr>
<td>rseFCM</td>
<td>Random Sample and Extend Approach FCM</td>
</tr>
<tr>
<td>SFCM</td>
<td>Spherical Fuzzy C-means</td>
</tr>
<tr>
<td>SFCM++</td>
<td>Spherical Fuzzy C-means with effective seeding</td>
</tr>
<tr>
<td>spFCM</td>
<td>Single Pass FCM</td>
</tr>
<tr>
<td>TF</td>
<td>Term frequency</td>
</tr>
<tr>
<td>XB</td>
<td>Xie Beni validity index</td>
</tr>
</tbody>
</table>
1. Introduction

This chapter presents a brief overview of the research contained in this thesis. We begin by outlining the motivations, objectives and contributions and we finish by describing the thesis organization.

1.1 Motivation

Every complex life form, endowed with a system for vision, learns early on how to segment the input coming from its senses (e.g. eyes) into discrete units on which it can mentally operate. These units can be further categorized based on their similarity (in shape, colour, odour etc.) into groups. By cluster analysis we will understand the process of finding these groups in data where objects in a group are similar to each other while objects from different groups are dissimilar [8].

In brains, this process happens in an automatic way requiring little or no deliberate thinking: a child learns to distinguish between cats and dogs, mother and father, windows from doors and so on. Moreover, once these categories are established it enables the brain to assign additional characteristics onto an object that are typical of the object category but perhaps not specific to the individual [9]. For example, a child having learned how to distinguish the class of dogs and having been presented only with cases of friendlier specimens would have, perhaps, a hard time in the company of a more aggressive breed (where he will assume the same level of friendliness). In general, this system of categorization makes life easier: imagine the difficulty of not being able to make any assumptions about the objects that we perceive at any given time.

Cluster analysis is an unsupervised component of learning and learning is a prerequisite for intelligence if by intelligence we understand the ability to solve goals. In contrast to supervised learning, where the learner is presented with labelled data, unsupervised learning is a process by which structures are discovered in unlabelled data. Trying to replicate human learning (either supervised or unsupervised) by machines is a long standing dream of humanity. The advent of computers able to perform accurate computations at a higher speeds than that of humans, lead to new ways of thinking about what is possible with such resources. Arguably, the seeds of Artificial Intelligence were planted at the seminal conference in Dartmouth [10] where its goal was clearly stated: “to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves”. Together with sub-fields such
as brain modelling, vision, natural language processing, expert systems, evolutionary computation and others, cluster analysis, the focus of this thesis, is an integral part of Data Science.

Before the advent of automatic cluster analysis algorithms, researchers have used plots and figures and carried out clustering in a subjective way, relying on perception which is suitable for this type of classification in up to three dimensions. As data keeps accumulating not only in size but also in dimensions, this problem becomes intractable for human minds. Algorithms for cluster analysis became prevalent in the last 30 years with a multitude being proposed to solve clustering problems with a diversity of methods stemming from the problems in defining what a cluster is in the first place [8]: spherical, linear, etc. compounded with the multitude of data types and several ways in which to define what constitutes similarity between objects. These methods can be classified into a few large categories such as: partitional-based, hierarchical-based, density-based, grid-based, methods based on co-occurrence of nominal attributes and others (an overview is given in Chapter 2).

An important category of unsupervised methods is that of partitional clustering, also referred to as partitioning relocation clustering [11] [12]. They use a predefined number of cluster points that are moved around in the data space, eventually settling in the middle of point cloud. The resulting solution is a partitioning of the data based on closeness to these centers points. These algorithms iteratively optimize a cost function that defines what represents desirable properties in clustering solutions. Some of the initial research [13] was concerned with finding hard clusters, i.e. disjoint, non-overlapping groups of objects that share similar features. What constitutes similarity is another aspect that has to be specified by the researcher before commencing the clustering procedure: many times this similarity function being one of the several realizations of Minkowski distance, with the most common being Euclidean distance although more complex function have been proposed.

Many times, the disjoint clusters produced by classical partitional methods such as K-means (KM) do not suffice. For example consider the illustrative case of the time series in Fig. 1 [14] which could represent anything from the observed daily rain amount in a location to the daily stock price of a London Stock Exchange listed company. First of all, we notice the existence of two homogeneous clusters: one marked in black (cluster 1) and one in blue (cluster 2). Arguably, a hard partitional clustering algorithm equipped with a sensible similarity metric would have no problem in discovering the two clusters.
However, the problem arises when a time series such as the one marked with dotted red line is present. This time series behaves as the time series in cluster 1 during the first half of the time and then switches and behaves as those in cluster 2 during the last. A hard partition algorithm would classify this time series randomly to either one of the above clusters which would be a mistake. Ideally we wish the cluster solution to be as informative about the actual behavior of this time series and it should clearly state that the membership of this time series is 50% in cluster 1 and 50% in cluster 2. As we will see, real world datasets, regardless of their type (time series or not), contain many such points that are at the edge of their clusters, requiring a soft clustering approach that is able to discover overlapping groups in the data.

Fuzzy C-means (FCM), the focus of this thesis, is such an approach which is able to discover overlapping clusters at the additional cost of maintaining and computing a membership matrix of the size of the data multiplied by the number of clusters. Each object to be clustered receives a row in this membership matrix with each column value of this row representing the belonging of the object to a particular cluster. The membership degree value varies from zero to one: a degree of zero means the object does not belong to the cluster while a degree of 1 represents total confidence that the object belongs to the cluster.

Similar to other partitional methods, FCM defines what constitutes a good solution in terms of a cost function: the points should be as close to their cluster center as possible (additionally to
KM, factoring the membership to that cluster as well). The algorithm does not guarantee that it will find the global minimum of this cost function and the choice of the initial centers has an important impact on the final solution. This thesis looks at ways in which to pick these initial center in order to make the algorithm faster, requiring fewer iterations and thus fewer membership matrix computations. An additional benefit of the proposed method is the general achievement of better clustering outcomes in terms of quality (defined through validity indices). We present a way in which to use our initialization method together with FCM as components in more elaborate knowledge discovery (KD) processes where other steps such as data preprocessing are required (Chapter 3-5).

1.2 Aims and contributions

The primary aim of this work is to investigate whether methods based on spreading the cluster centers of the soft partitional clustering algorithm FCM are viable and result in improvements over other algorithms, in terms of the number of iterations as well as cluster solution quality (as captured by external validity indexes). Since soft partitional algorithm such as FCM provide a much more realistic and natural clustering solution (that discovers overlapping clusters) than hard partitional ones (discovering disjoined, non-overlapping clusters), an improvement in its speed (number of iterations to convergence to minima of cost function) and quality (as expressed by validity indexes) can have a significant impact on the numerous applications where soft partitional clustering methods are applied. As we will see in Chapter 2 (Literature survey), there are numerous fields that currently use FCM to solve real world challenges: from image segmentation to recommender systems and anomaly detection.

We can split the thesis contributions into three main parts:

1. **A seeding mechanism for FCM** which leads to an improvement in its speed and cluster quality on a wide range of data sets. The new algorithm that we proposed, Fuzzy C-means++ (FCM++), samples starting cluster representatives during the initialization phase in such a way that they are well spread in the input space, resulting in both faster convergence times and higher quality solutions. A similar but simpler method to the one that we propose, entitled K-Means++ [15] (KM++), has already shown its efficiency on a wide range of datasets. Both FCM and KM are partitional clustering algorithms with the difference being that FCM is a more general algorithm, allowing the discovery of
either non-disjoint or overlapping clusters (in its default case) as well as disjoint and non-overlapping ones (when its m parameter is set to 1) resulting in a behavior similar to KM. Unlike KM++, our initialization algorithm allows for the fine tuning of the spreading power through a predefined parameter. The more we increase the value of this parameter, the more the probability distribution is skewed towards picking points that are far from each other. Our method compares favorably to both standard FCM and Subtractive (a popular seeding mechanism). We consider 4 synthetic cases of Gaussian clusters having different sizes and degrees of overlap and we show an improvement of FCM++ over the standard algorithm in terms of iteration counts, Xie-Beni and Partition Entropy external validity indexes. Further, we consider popular UCI [16] real world data sets such as Iris, Wine a Spam and show a similar improvement.

2. **A methodology for clustering financial time series that uses FCM++.** Clustering time series is a more complex task where we need additionally to capture the relationship between consecutive time intervals that can require several steps of preprocessing. Such preprocessing might need to transform the data and make it stationary. Further, we can aim to visualize the correlation between the time series and facilitate the clustering procedure by performing a Multi-Dimensional Scaling (MDS) and reducing data dimensionality.

We look first at synthetic time series and show how FCM++ could make a significant improvement over the standard in terms of iterations (6.3 iterations on average versus 22.6 for the standard). Secondly, we have addressed a more challenging task of grouping similarly fluctuating stocks of leading (in terms of market capitalization) London stock exchange (LSE) companies based on their historical price record. We show the improvement of FCM++ over FCM in terms of speed and quality (external validity index) and finally we analyze each cluster in terms of profitability and risk. Such an approach can have an important impact in investment decisions, in particular portfolio creation and management where the task is to minimize risk and one way to achieve this is by combining uncorrelated assets.

3. **A methodology for clustering text data.** We propose a methodology for visualizing and clustering a pervasive type of data, namely that which comes in textual form. We present a case study involving 368 news articles extracted from a popular British
newspaper web-site. The methodology that we propose performs several preprocessing steps and makes use of the Spherical Fuzzy C-means (SFCM) algorithm, a variant of FCM that is adapted to work with text data. The spreading mechanism that we introduced for FCM is extended to SFCM resulting in the SFCM++ algorithm which picks the cluster centers during initialization closer to their final position, translating into a lower number of iterations count until convergence and a higher quality solution in terms of validity indexes. We show that on average, SFCM++ method needs 12.2 iterations vs 16.8 needed by the standard SFCM (the one with random initialization). Seven out of every ten runs, SFCM++ achieves a lower Xie-Beni index and Partition Entropy value suggesting higher quality solutions.

1.3 Structure

This section provides an overview of the contents of the following chapters. The thesis is organized as follows:

   **Chapter 2:** surveys the existing literature on cluster algorithms in general and soft partitional algorithms in particular. We begin by defining cluster analysis and discuss the steps required for performing a successful analysis. We categorize clustering algorithms into five large types and we briefly talk about their features and some of their representative methods. The FCM algorithm is introduced together with its limitations.

   **Chapter 3:** discusses the main contribution of this thesis, the FCM++ algorithm. We discuss how the initialization algorithm for FCM works and compare the results obtained with it to those obtained by running the standard version of FCM (which initializes cluster centers at random). Improvements in terms of number of iterations and quality solution (captured with popular validity indexes) are discussed on a wide range of data sets including the popular Iris Dataset, Life Expectancy and Transfusion Datasets.

   **Chapter 4:** In this chapter we survey the process of clustering time series data, which has an implicit, temporal, ordering on its features. Unlike standard datasets in Chapter 3 where the use of Euclidean distance might suffice, time series data, especially as used in finance, often requires the use of correlation or other, more complex distances. We propose the use of Pearson Correlation together with Multi-Dimensional Scaling in order to reduce dimensionality, a process which makes the data visualizable in two dimensions and makes the clustering process more accurate. We
evaluate our proposed method and compare it with the standard as part of a methodology for clustering both synthetic time series as well as real financial time series of stock listed at the London Stock Exchange (LSE).

Chapter 5: We introduce a second case study that considers textual data, in particular news articles. We provide a methodology for visualizing and clustering news articles crawled as textual data from the Guardian [137] newspaper website. A total of 368 news articles covering four major topics (football, immigration, environment and finance) are extracted, pre-processed in several steps, visualized and clustered using SFCM++, a FCM++ like algorithm for text data which uses cosine distance for capturing similarity between documents.

Chapter 6: This chapter reviews the thesis key findings and identifies areas of future work in the area of FCM clustering initialization. We discuss the possibility of using distance distributions to further improve the FCM++ algorithm. Point potentials computed as functions of neighbourhood are also discussed. We conclude with a discussion on autocorrelation coefficients and how they could be used in time series clustering with FCM++.

1.4 Publications:

Publications published during the work include:

2. Background

2.1 Introduction

This chapter investigates the current state of cluster analysis. Section 2.2 places the domain of cluster analysis in the context of Artificial intelligence (AI) and surveys the existing literature on cluster algorithms in general. Hierarchical, Partitional, Grid-based and other types of clustering algorithms are presented and discussed. Fuzzy C-Means (FCM) is introduced in Section 2.3 where we present an overview of the history of the method together with its areas of application. Section 2.4 investigates the proposed improvements of FCM throughout the literature.

2.2 Cluster Analysis

AI is a sub field of computer science that originated in 1956 [10] and which is concerned with the scientific goal of understanding real intelligence (analyzing aspects such as knowledge representation and learning) and the engineering goal of solving real world challenges through the application of its methods. The sub-fields of AI are numerous and include brain modelling (neural networks), evolutionary computation (genetic algorithms), vision (object recognition), natural language processing, expert systems, planning and machine learning [17]. Machine learning can be further divided into three main sub fields [18]:

1. **Unsupervised learning** also known as cluster analysis or exploratory data analysis, the main focus of this thesis, has the goal of separating a finite unlabeled dataset into a finite and discrete set of natural hidden data structures [19][20][21].

2. **Supervised learning** aims to build models that approximate functions from a set of labeled examples provided by an expert. Supervised learning algorithms can be classified into categories based on the type of the variable they predict: Regression (Neural Network [22], Linear Regression [23], etc.) and Classification (Naïve Bayes[24], Support Vector Machines [2], etc.).

3. **Semi-Supervised learning** or reinforcement learning [26] aims to build adaptive systems that can learn from their environment and maximize a utility function predefined by an expert.
Cluster analysis algorithms place unlabeled objects into subgroups, also called clusters that are suggested by the data rather than being defined a priori. Objects in a given cluster tend to be similar to each other in some sense, and objects in different clusters tend to be dissimilar [27]. This similarity is subjectively chosen (based on its ability to create “interesting” clusters) and the similarity between objects within a subgroup is greater than the one between objects belonging to different subgroups [28]. This idea of similarity, or closeness, is captured by a specified distance function which has an important impact on the final clustering solution. Table 2.1 presents a few of the most common similarity (distances) used in the literature [19].

| Minkowski based distances | \[ D_{ij} = \left( \sum_{l=1}^{d} \left| x_{il} - x_{jl} \right|^n \right)^{\frac{1}{n}} \] |
|--------------------------|--------------------------------------------------|
| Euclidean Distance       | \[ D_{ij} = \sum_{l=1}^{d} \left| x_{il} - x_{jl} \right|^2 \] |
| City-Block (Manhattan Distance) | \[ D_{ij} = \sum_{l=1}^{d} \left| x_{il} - x_{jl} \right| \] |
| Sup Distance             | \[ D_{ij} = \max_{1 \leq s \leq d} \left| x_{il} - x_{jl} \right| \] |
| Mahalanobis distance     | \[ D_{ij} = (x_i - x_j)^T S^{-1} (x_i - x_j)^T \] |
| Pearson correlation based distance | \[ D_{ij} = 1 - \frac{\sum_{i=1}^{N} (l_i - \bar{l})(J_i - \bar{J})}{\sqrt{\sum_{i=1}^{N} (l_i - \bar{l})^2} \sqrt{\sum_{i=1}^{N} (J_i - \bar{J})^2}} \] |
| Cosine Similarity distance | \[ D_{ij} = \frac{x_i^T x_j}{||x_i|| ||x_j||} \] |
| Point Symmetry distance  | \[ D_{ir} = \max_{j=1..N \text{ and } j \neq i} \frac{|| (x_i - x_r) + (x_j - x_r) ||}{|| (x_i - x_r) || + || (x_j - x_r) ||} \] |

Table 2.1. Similarity measures used in clustering [19]

Euclidean distance, one of the simplest and most intuitive similarity metrics, is the one we have used in the experiments in Chapter 3 (artificial and real world datasets) and Chapter 4 (case
study where we used it as part of the clustering procedures on Multi-dimensional Scaling (MDS) compressed representations of highly dimensional financial time series).

1. The data can be filtered using feature selection (where only the relevant features are retained) or feature extraction (a combination of the features is created) both of which can affect the effectiveness of the clustering solution. This step can have other motivations besides relevancy: it can result in general data reduction (to limit storage requirements), feature set reduction (to save resources for the next step), performance improvement and data understanding [29].

2. We choose an algorithm (Table 2.2) that is appropriate for the type of data distribution we want to cluster. As an example, spherical clusters can be discovered with K-Means (KM) or Fuzzy C-means (FCM) but non-spherical ones might require other methods such as Kernel K-means (KKM) or Kernel Fuzzy C-means (KFCM). The choice of similarity measure (Table 2.1) is also significant: as we will see in Chapter 5, Cosine distance is more appropriate than Euclidean distance when clustering text data. Once the clusters are discovered, the user can choose to display the solution.

3. The user can choose to assess and validate the solution. For this purpose, there are two main types of validation functions: external (where we evaluate the result with respect to a pre-
specified structure, in other words we know the labels beforehand) and internal criteria (where we evaluate the result with respect to information intrinsic to the data alone) [30]. This internal criteria that we want to capture in cluster solutions is based on the idea of compactness (how close the elements of a clusters are) and separability (how separate the clusters are) although other internal criteria have also been used (e.g. Partition Entropy discussed in Chapters 3-5). There are many ways in which the results of clustering can be expressed [31]. For example, the groups identified in the data could be exclusive with each instance belonging to only one group (hard clustering) or they may be overlapping (soft clustering) where instances could fall into all groups in different percentages. Unlike hard clustering, where each data point is assigned to only one cluster, in soft clustering, data elements can belong to more than one cluster.

4. Finally, the results interpretation phase is the ultimate step where meaningful insights are discovered and turned into knowledge that helps the user effectively solve their problems [18]. The problems that can be solved by cluster analysis are numerous and include collaborative filtering, customer segmentation, data summarization, dynamic trend detection, multimedia data analysis, biological data analysis, social network data analysis and so on [32].

<table>
<thead>
<tr>
<th>Hierarchical algorithms</th>
<th>Single and Complete Link, DIANA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning algorithms</td>
<td>KM, K-medoids, FCM, SFCM</td>
</tr>
<tr>
<td>Density based algorithms</td>
<td>DBSCAN, GDBSCAN, OPTICS</td>
</tr>
<tr>
<td>Grid based algorithms</td>
<td>STING, WaveCluster, OptiGrid</td>
</tr>
<tr>
<td>Methods based on co-occurrence of categorical data</td>
<td>ROCK, CACTUS</td>
</tr>
<tr>
<td>Other</td>
<td>Constraint based clustering, Graph Partitioning</td>
</tr>
</tbody>
</table>

Table 2.2 Clustering methods classification [12]
Choosing the right clustering algorithm is the most important part of the clustering process and the following section surveys the main types of cluster algorithms proposed in the literature. Cluster validation and feature selection/extraction will be thoroughly introduced in the following chapters where we will show a method to validate our solutions (using Xie-Beni and Partition Entropy index) and preprocess our the data (MDS scaling). The clustering algorithms can be split into six categories as in Table 2.2 [12]:

1. **Hierarchical methods** build a hierarchy of clusters using either a top-down (divisive) or bottom-up (agglomerative) approach; Some example of algorithms that fall into agglomerative category include: Single and Complete link [33], Group average [34], Wards' method [35], UPGMA [36], etc. and some that fall into divisive methods include: DIANA (divisive analysis), MONA (monothetic analysis) [8] and so on;

2. **Partitional clustering methods**: Also called "centroid based clustering" considers clusters as sets of similar objects closer to the center of a cluster rather than any other center of other clusters. These centers define a partitioning of the data space into regions also known as Voronoi regions. Some example of algorithms that fall into this category include: KM [13], K-medoids [37], K-medians [38], FCM [39];

3. **Density-based clustering methods**: discover, using only one scan, arbitrarily shaped regions of data points that are densely populated; Some example of algorithms that fall into this category include: DBSCAN [40], GDBSCAN [41], OPTICS [42], DENCLUE [43], CLIQUE [44];

4. **Grid based algorithms**: Clustering algorithms based on grids divide up the data space into finite number of cells that form a grid structure and perform clustering on this grid structure [45]. The algorithms in this category are fast as they process the grid structures and not the points directly. Famous grid-based clustering approaches include STING [46], WaveCluster [47], OptiGrid [48] and so on. It is worth knowing that some algorithms such as CLIQUE [44] and DENCLUE [43] fall into both the density and grid based categories.

5. **Methods based on co-occurrence of categorical data**: These methods work with data that may have categorical (nominal) data. This type of data is often related to transactions involving a finite set of items (e.g. market basket) [12]. Some examples of algorithms falling into this category include ROCK (Robust Clustering Algorithm for
Categorical Data) [49] and CACTUS (Clustering Categorical Data Using Summaries) [50].

6. **Others:** This category includes methods based on constraints (such as COP K-means [147] where the solutions need to satisfy certain limitations specified by the user beforehand), graph partitioning algorithms (as those described in [3]) and so on.

The classification above is derived from [12] omitting the categories of scalable and clustering of high dimensional data as most of the algorithms in those categories were already described or can be used as they are on large datasets (e.g. CLIQUE [44], CACTUS [50], etc.) and we will also see in Section 2.4 how existing algorithms such as FCM can be scaled to work with larger datasets.

It is also important to mention at this point that the above specified list is in no way exhaustive: new algorithms emerge constantly in the literature while improvements are made to older, existing algorithms.

### 2. 2.1 Hierarchical algorithms

Fig 2.2. Hierarchical Clustering: each node represents a cluster and each leaf node represents a singleton cluster [1]. Each tree level represents a segmentation of the data and a clustering is
obtained by cutting the tree at the desired level. The correct cutoff level can be determined using either (or both) context information and validity indexes.

Hierarchical clustering methods build a hierarchical tree-like structure either by progressively merging objects into groups (in the case of agglomerative methods) or by dividing large groups into smaller chunks (in the case of divisive methods). Unlike Partitional or density-based clustering, the goal of hierarchical clustering is not to find a single partitioning of the data set but instead find a hierarchy (dendogram) of the data which may reveal interesting patterns at multiple levels of granularity. An important concept in hierarchical clustering, a criteria for differencing between all the proposed sub-methods, is that of the linkage method: or how to link together (agglomerative) or split clusters (divisive). Agglomerative clustering starts by considering each object as belonging to one and only one cluster and then merges these clusters based on a predefined measure of similarity at each step, thus following a bottom-up approach. Divisive clustering works in the opposite fashion (top-down) considering every object as belonging to one large cluster and then splitting recursively.

From [51] we find that because of their speed and easily interpreted results, hierarchical clustering algorithms have made their way into many fields such as biology, social sciences and computer vision. However, the main drawback of these methods lies in their sensitivity to outliers: consider for example the case of complete-linkage criterion where the maximum distances between points in two clusters are computed. Two clusters can be very close together in terms of distance between constituent points but if their furthest away elements are separated by a large distance they would (possibly) not be merged according to an agglomerative method. Some examples of linkage criteria used in hierarchical clustering include [1]:

- Single-linkage (where two clusters separated by the shortest distance are combined at each step).
- Complete-linkage (where two clusters having the two furthest away elements separated by the shortest distance are combined at each step), etc.
- Group average-linkage (where the distance between any of the considered clusters is taken as the mean distance between all of their points taken as pairs).
- Ward criterion (where the pair of clusters that leads to minimum increase in total within-cluster variance after merging are combined together).
2.2.2 Partitional algorithms

Partitional cluster analysis is defined as the problem of partitioning a group of objects into clusters that share similar characteristics. As compared to other techniques, these approaches, also known as objective function-based clustering, assume that the data set can be well represented by finite prototypes [52]. Some of the most well-known and widely used partitional clustering algorithms are KM [13], K-medoids (KMED) [37] and FCM [39].

KM is one of the oldest partitional clustering algorithms. Given a set $X$ of input data of size $N$ and a parameter $K$, the task of KM is to choose $K$ representatives of $X$ such that the distances between any point $x_i$ (where $i \in [1..N]$) and its representative $r_j$ (where $j \in [1..K]$) is minimized.

The set of representatives discovered after running the KM algorithm is enough to define a clustering of the points in the data space (the $j$th cluster being the set of all points in $X$ that are closer to $r_j$ than any other representative). The KM algorithm minimizes the following objective function:

$$J(R) = \sum_{i=1}^{N} \sum_{j=1}^{K} |x_i - r_j|^2$$  \hspace{1cm} (2.1)

Using a series of iterations consisting of

- an assignment step, where all the points in the data set are assigned to the closest representative according to a distance metric (most commonly Euclidean distance);
- an update step, where the representatives are recomputed as the centroids of new groups of points;

The KMED algorithm is related to the KM algorithm in the sense that it is defined in terms of a cost function with both of them trying to minimize the distance between every point in a group to the group representative (center of that cluster). Instead of computing the means during each iteration, a medoid (representative) that minimizes the cost function is chosen from the points in the cluster. The KMED algorithm with its implementations such as PAM [37] or CLARA (a medoids based algorithm for large datasets) [37] try to mitigate one of the problems in classical KM, namely that of outliers distorting the clustering result. For example, PAM uses the most representative objects in the clusters (called medoids) instead of the centroids (as in KM) which deemphasizes the influence of outliers [146].
The FCM algorithm, the subject of this thesis, is a generalisation of the KM algorithm, it was constructed on a fuzzy set framework and allows for overlapping clusters, describing a more realistic solution. The algorithm will be described at length in Section 2.3 and in Chapter 3 we present a method for initializing FCM in such a way as to minimize the number of iterations while boosting cluster solution quality.

2.2.3 Density-based algorithms

Density-based clustering methods consider areas having a large number of data points (dense regions within a specified radius $\varepsilon$) as belonging to clusters while at the same time viewing sparse areas of lower density as noise or borders of the discovered clusters. The main strengths of density-based methods lie in their ability to automatically discover the number of clusters present in the data (unlike partitional KM, for example, where it has to be supplied as a parameter). These algorithms have the capacity to discover arbitrarily shaped clusters and are independent of initial starting conditions unlike the partitional clustering methods.

![Fig 2.3 DBSCAN](image)

Fig 2.3 DBSCAN [40] Density of cluster p (core) is high ($MinPts = 4$) while density of q (border) is low ($MinPts < 4$)
The most representative and popular density-based clustering algorithm is DBSCAN (ranked 41st most popular [55] in terms of publications). This method considers points as core points if they have at least \( MinPts \) neighbours in their region of radius \( \varepsilon \), border points (if they have less than \( MinPts \) in their neighbourhood but are within the \( \varepsilon \) radius of a core point) and finally noise points if they are neither core nor border points. One cluster is formed by points which can be reached from a core point by hoping from one point to another using steps of no more than \( \varepsilon \) in distance.

The shortcomings of DBSCAN are:

- Sensitivity to distance measures: when data is highly dimensional Euclidean distances fail due to the "Curse of Dimensionality" [56]
- Density variability: different regions in space may have different densities so it becomes a problem choosing the right radius parameter
- The necessity to specify at least two parameters: \( \varepsilon \) and \( MinPts \) which may be very hard to determine [57]

The intrinsic structure of most real-world datasets cannot be characterized by global density parameters assumed by the DBSCAN algorithm: the density that defines a cluster in most cases is not constant throughout the data. For example, it is impossible to detect clusters A, B, C1, C2, C3 using one global density parameter Fig. 2.4 from [42].

![Fig 2.4 Clusters with respect to different density parameters](image-url)
Ankerst [42] developed the OPTICS algorithm which addresses the density variability issues of DBSCAN by computing a data structure (called augmented clustering order) of the database objects which provides information equivalent to density-based clustering corresponding to a wide range of parameter settings.

2.2.4 Grid-based clustering

Grid-based clustering algorithms have a lot in common with density-based methods and for illustration let us consider the algorithm CLIQUE (Clustering in QUEst) [44]. It is an algorithm that discretizes a high dimensional data space through a grid and estimates the density by counting the number of each points in a grid.

As an example, let us consider a three-dimensional employment data space containing Salary, Age and Vacation as axes. This data space can be divided into three two-dimensional subspaces: one formed by Salary and Age, one by Age and Vacation and one by Salary and Vacation. Fig 2.5 [2] shows how two of these subspaces may look. One thing we notice is that these subspaces are split into the equal sized number of bins (through the grid).

![Fig 2.5. Two of the subspaces of the data [2]](image)

Dense regions (shown in colour in Fig 2.5) in these subspaces contain a higher amount of points than a pre-specified parameter. A cluster is then defined as a maximal set of connected dense units [58]. Once the subspaces are processed and dense regions are found the algorithm
proceeds to search for dense regions at a higher level, namely the intersection of these dense subspaces which forms the search space in a higher dimension (three in our case, Fig 2.6).

Some of the strengths of this algorithm include its efficiency, scalability (it scales linearly with the size of input), and insensitiveness to the order of records among others. The main drawback of this approach is the accuracy that is degraded at the expense of simplicity [45].

Fig 2.6 The intersection of the subspaces shown in 3D [2]

2. 2.5 Methods based on co-occurrence of categorical data

The algorithms presented so far work well with datasets that have numeric attributes. Most expect a similarity (distance) function (as in Table 2.1) to be specified beforehand and expect numeric attributes. To overcome this limitation, algorithms such as ROCK (Robust Clustering Algorithm for Categorical Data) [49] have been proposed.

Instead of using metric distances the ROCK [49] algorithm performs clustering using a Jaccard coefficient similarity measure which allows for nominal data. It also defines a threshold on this similarity measure to decide whether two or more points are neighbours. By links it is understood the number of neighbours in common and the larger the number of links the more
probable that these points are in the same cluster. The authors found empirically that the algorithm has good scalability properties.

2. 2.6 Other clustering algorithms

There are various clustering algorithms that could fall into this category; we will focus our discussion on two other major types of clustering: constraint-based clustering and graph partitioning clustering.

All algorithms discussed so far are unconstrained algorithms as they do not receive a set of constraints from the user that specify how the solution should look. In constraint-based clustering the user specifies these constraints that can fall into four categories: (1) constraints on objects (which confines the objects to be cluster), (2) obstacle objects as constraints (for example, when geographical regions are to be clustered, mountain, lakes, rivers need to be identified), (3) clustering parameters as constraints or (4) constraints imposed (e.g. on size) on each individual cluster [59].

Graphs are mathematical structures formed by nodes (vertices) and edges (connections between the nodes). The aim of graph clustering is to group nodes of the graph together if they have many edges between them and fewer between clusters containing other nodes [3]. Fig 2.7 from [3] shows an example of a graph clustering solution. A good quality cluster is defined as a subgraph having dense connection such as the left cluster in Fig 2.7 which has many edges between its points and few outside reaching edges. The middle cluster in Fig 2.7 is worse in terms of quality (has multiple outside edges) while the rightmost cluster has fewer outside edges but few inside edges making it a poor solution too.

![Fig 2.7 An example of graph clustering showing three clusters [3]](image)
2.3 Fuzzy C-means

2.3.1 Origins

The FCM clustering algorithm traces its origins in fuzzy set theory introduced by Lotfi Zadeh in 1960 [60]. Fuzzy set theory extends the mathematical theory of sets to allow for objects having a continuum of grades of membership to sets. Such sets are characterized by a membership (characteristic) function which assigns a membership degree from 0 to 1 for each object.

For example, let us consider the set of young people and the set of old people. These, like many other categories we encounter in our daily life (tall versus short, slim versus fat) have imprecise meaning. While it is not a problem to classify a 1 year old to the class of young people and a 100 year old to the class of old people, the problem arises at the boundaries of these sets: how would we classify a 50 year old? It would certainly be wrong to assign him to one and only one of these sets. What about a 25 year old versus a 1 year old? While 25 is young it is not as young as 1 and in classical set theory we have no way in specifying the degree of youthfulness. If instead we define two fuzzy sets YOUNG and OLD with their corresponding membership functions ranging from 0 to 1 then we can obtain a more expressive situation. Membership function of the YOUNG (shown in red in Fig 2.8) set would assign a high value (close to 1) to a 1 year of child, a lower value of 0.75 to a 25 year old, a 0.50 value to a 50 year old and the values keep decreasing, finally outputting a value of 0 for a 100 year old (we are pretty sure he is old). A similar but reversed membership function (shown in magenta in Fig 2.8) could be attached to the OLD set where the function keeps increasing with a 100 year old having a membership degree of 1. We can further extend the concept of fuzzy set theory to two important sets, namely that of TRUE and FALSE sentences and we obtain a new class of logic, that of fuzzy logic where similar generalized deductive operations can be applied and new knowledge discovered.

The attractive property of fuzzy set theory to incorporate vagueness led to its applications in a wide range of field from human sciences [61], industrial process control [62], linguistics [63], soil classification and modelling [64], philosophy [65] and the list goes on. A search for “fuzzy set” on google scholar returns 1,860,000 results as of July 2016.

At this point we can start to understand how fuzzy theory can be incorporated into the problem of clustering. Many times, real life objects that we want to categorize do not belong certainly to one category or another and one of the main reasons for why this happens is the imprecise nature
of perception “reflecting the bounded ability of human sensory organs, and ultimately the brain, to resolve detail and store information” [66].

![Fig 2.8 Membership function assignment of a person based on age: in red we see a hypothetical membership function called YOUNG while in magenta one named OLD](image)

2.3.2 FCM: The algorithm

FCM is a generalization of KM (it can be made to behave similarly and perform hard clustering through the use of its parameter 𝑚 which specifies the fuzziness of the solution). It uses a similar cost function to which it adds the membership degrees (see [39]) and it expects, additionally to the 𝑘 parameter, an 𝑚 parameter (default 𝑚=2) which specify how fuzzy (overlapping) we want the solutions to be.

The application domain of FCM are numerous, being one of the most popular ongoing area of research [67]. The algorithm found its use in video segmentation [68], sonar returns [69], colour clustering [70], real-time applications [71], signal analysis [72], internet portals [73], ECG arrhythrias [74], spike detection [75], biology [76], damage detection [77], gene expression [78], forecasting [79], speech recognition [80], forest fire detection [81], load frequency [82], supply chain management [83], document analysis [84], fluid detection [85], mathematics [86], software engineering [87][88] and communication [89].

The standard version of the FCM algorithm [39] (Fig 2.9) minimizes the following function:

\[
J_m(U, R) = \sum_{i=1}^{N} \sum_{j=1}^{K} \mu_{ij}^m |x_i - r_j|^2
\]

subject to
\[ \mu_{ij} \in [0,1]; \sum_{j=1}^{k} \mu_{ij} = 1 \forall i; \ 0 < \sum_{i=1}^{n} \mu_{ij} < N, \forall N \] 

(2.3)

**Algorithm 1: Fuzzy C-means**

Given
- a set of \( N \) data points \( X = \{x_i\}_{i=1}^{N} \)
- \( k \) representing the clusters number
- \( m \) representing the fuzziness parameter

Return
- a set \( R \) of centers
- a membership matrix \( U \).

1: \( U \) and \( R \) are initialized at random
2: repeat
3: \[ u_{ij} = \frac{1}{\sum_{k=1}^{K} \left( \frac{|x_i - r_j|}{|x_i - r_j|} \right)^{\frac{2}{m-1}}} \]
4: \[ r_j = \frac{1}{\sum_{i=1}^{N} \mu_{ij}^m} \sum_{i=1}^{N} \mu_{ij}^m x_i \]
5: until \( |U^{k+1} - U^k| < \epsilon \)
6: end procedure

Fig 2.9 FCM Algorithm

where \( X = \{x_i\}_{i=1}^{N} \) the set of data points, \( U = \{\mu_{ij}\}_{i,j=1}^{NK} \) the matrix of membership degrees, \( k \in N \) the number of clusters and \( R = \{r_i\}_{i=1}^{k} \) the set of representatives, \( m \) is the fuzzifier parameter which determines the fuzziness of the cluster. At the limit \( m \to 1 \), the method becomes standard KM, here we use a default value of \( \epsilon = 0.00001 \) and the \( || \) operation denotes the Frobenius norm of the difference between the matrices.

In contrast to the hard clustering of KM where each point belongs to one cluster, in FCM each point \( x_i \) in the space belongs to \( r_j, \forall j \in R \) with \( \mu_{ij} \in [0,1] \) defined in the membership matrix (of size \( n \times k \) where \( n \) is the number of points in the data space and \( k \) is the number of representatives). The use of a membership matrix increases the expressiveness of the clustering analysis, arguably presenting a more comprehensive view of relationships present in the data. Further, the hard assignment of the data points by KM is inadequate when the points are equally distanced between
representatives (e.g. point s in Fig 2.10), in which case they will be randomly assigned to one cluster or another. FCM mitigates this problem by assigning equal degrees of belonging through the use of the membership matrix.

Consider for example the two clusters of objects in Fig. 2.10 and a hard clustering scheme. We observe that objects s1...s5 belong to C1, whilst others, s6...s9, belong to C2. When an object that is located at the boundaries of the space covered by the two clusters is considered (in bold), it will be randomly assigned to one of the clusters. By implication it belongs to that cluster as much as any other member does, even though its qualification for membership is much less clear.

The object marked in bold situated halfway between the cluster centers will have, more realistically, a membership degree of 0.5 for each. In contrast, in a soft clustering scheme all objects belong to all clusters but with a different belonging degree specified by a membership function. As a consequence, an object such as s3 will have a high membership value for C1 and a low membership value equal to 1− (its membership in C1) for C2 (because the memberships sum to 1).

This method computes membership degrees matrix U at each iteration, a costly operation that gives a membership degree to a point proportional to its proximity to the cluster representatives. Moreover, the size of this matrix grows as a product of the number of points and clusters, making the algorithm computationally expensive for high values. The following section will discuss some of the improvements that have been made to the standard algorithm.
2. 3.3 Improvements to FCM

Although noted both for its simplicity of implementation and its output validity [70], FCM suffers from high computational cost. For each iteration, the computational complexity of the algorithm is quadratic in the number of clusters $O(NC^2P)$ where $N$ is the number the number of data points, $C$ is the number of clusters and $P$ is the dimension of the data points.

A linear complexity approach $O(NCP)$ that removes the need to store a large matrix during iterations was proposed in [90]. In [91] a method to obtain qualitatively better clusters (as measured using a series of validity indexes) is proposed. This approach uses a weighted Euclidean distance which incorporates feature weights. While this method showed promising results on several UCI databases, it requires a feature weight learning step of complexity $O(N^2CP)$. A hybrid approach proposed in [92] combines particle swarm optimization (a stochastic population-based global optimization technique) with FCM. Although this approach achieved encouraging results in terms of final cost function value, particularly for smaller datasets where it surpassed the standard method, its drawback is the need for an additional fine tuning step on a set of parameters: population size, inertia weight and acceleration coefficients.

An interesting methodology combines the standard version of FCM with an evolutionary algorithm [93] assuming that the distance function should not be predetermined when running the algorithm. The evolutionary algorithm searches for the best distance function (in terms of validity index) to be used for a particular dataset.

Heuristics have also been utilized in [94], where a hybrid method is introduced that utilises evolutionary algorithms to reduce the dimensionality of the space. Their method simultaneously performs feature selection and data clustering yielding fewer misclassifications and better validity index values.

Hathaway and Bezdek [5] investigate the behaviour and generalize the FCM method using different $L_p$ norms for computing the dissimilarity both on object (or feature data sets) and relational (where instead of objects we have relations between objects) datasets. This work shows that for feature data the best norms to use are $L_2$ (the standard norm used for FCM and the one used here) and $L_1$ (or Manhattan distance). The $L_1$ distance between two points is defined as the sum of their absolute differences of their coordinates, a measure which is particularly suitable for noisy data containing outliers. As $p$ in $L_p$ increases, the attraction of the centers to outliers increases, resulting in a less reliable model.
Improvements in terms of reliability to outliers have been proposed through a relaxation of constraints (each element of a membership matrix column, also called typicality, can be any positive number in [0,1] as long as their sum is bigger than zero). The method, termed Possibilistic C-means (PCM), was introduced in [95]. Further, a hybrid method called Possibilistic FCM (PFCM) that combines both FCM and PCM in a variable proportion using two additional parameters was introduced in [96]. This method mitigates both the problems of FCM with outliers and PCM which, although reliable to noisy data, sometimes generates coincident clusters and is highly sensitive to initialization. However, PFCM is a more complex algorithm that uses two matrices and has more parameters that need to be specified. In order to deal with more corrupt noisy data, Kannan and Ramathilagam [97] introduced an initialization scheme for a generalized FCM algorithm termed quadratic entropy-based FCM.

Other improvements to the method have been achieved in the context of image segmentation such as [98] (colour segmentation) where an initialization scheme selects the centers based on the idea that vivid and dominant colours in an image are unlikely to belong to the same cluster. In [68] the objective function of the standard FCM method is modified to incorporate a penalty term (regularizer) which considers the influence of neighbouring pixels over the central ones.

An important aspect that affect performance and final quality of the clustering solution in partitional clustering algorithms is the choice of center initialization. In both KM and FCM, the choice of initial centers matters as it will impact directly the quality of the solution. For example let us consider 5 tight clusters (of radius λ) separated by an empty space of radius B [99]:

![Fig 2.11 Five clusters of radius λ separated by space of size B](image)

The global minimum is achieved when the centers are perfectly placed in the middle of the clusters and it is equal to \(\lambda^{2n}\) (KM) and \(\mu^{n\lambda^{2n}}\) (for FCM). In their standard form both these algorithms initialize the centers at random either by picking from a Gaussian distribution (with a given mean and variance) or selecting from the existing set of points in the dataset. If we let the algorithm pick initial centers at random from the existing points it can start in the following configuration: one center in C2, two centers placed in C3, one in C4 and one in C5 (Fig 2.12).
While the centers of C4 and C5 will move to their correct position in the first iteration, the cluster center of C2 will move in the middle of C1 and C2 and both the centers in C3 will end up sharing the cluster same cluster.

In Figs 2.12-13 we notice the possible impact of choosing the initial clusters at random, often leading to a sub-optimal solution.

To mitigate the problem of random initialization, [100] proposes "The Mountain Method" for estimating good initial center positions together with the number of clusters. The first step consists of discretization of the $R^S$ data space by forming a grid in the $R^S$ space. In Fig. 2.14 we can observe a grid formed in $R^2$. 

![The Mountain method for center initialization](image)
The grid lines intersection forms the set of vertices $V$, which are taken as the primary candidates for cluster centers. The user can determine how fine or coarse the grid is: by increasing the number of grid lines, and in turn the number of vertices, one can increase the quality of the solution at the cost of increasing the computation process. Besides uniform grid lines one could use a more complex setting with variable gridding. After creating the grid, the method computes what is called "the mountain function" which takes as argument the vertices discovered in the previous step (Figs 2.15-16).

![Fig 2.15. The mountain function at the first step [100]](image1)

![Fig 2.16. The mountain function after the first cluster is removed [100]](image2)

For each point $x_i$ in the data set a value is added to a point $v_j$ proportional to the distance from the two points. In this way the mountain function for grid points $v_j$ that contain many close neighbours will receive a higher value than points with fewer neighbours. In this way the function resembles a mountain range and reflects the underlying distribution of the data. After this step, also known as mountain creation, the highest peak is selected as the first center $c_1$. If there is more than one peak having maximal height, one of them is picked at random. It is usually the case that the highest peak is surrounded by grid points which also have high values. The next step involves destroying the mountains: for all the points in $V$, a quantity is subtracted, proportional to their distance from the newly selected center. This ensures that the next peak $c_2$ is not located very near to the previous center. The process continues in this manner until the mountain function is reduced under a certain threshold given at the start of the algorithm. The introduction of the grid lines allows for an approximation to an otherwise continuous optimization problem.
The distance measure \( d(x_k, V_i) \) is used to measure the contribution of point \( x_k \) to the mountain function of grid point \( V_i \).

\[
M_1(V_i) = \sum_{k=1}^{n} e^{-\alpha \, d(x_k, V_i)}
\]

(2.4)

This function computes the value at grid point \( V_i \) as a function of the distance from all other points in the dataset. The contribution of each point to increasing the mountain value decays exponentially with furthest points having minimal contribution. The algorithm is not limited in using a certain type of distance nor is it bounded in using the exponential function as part of (x). Removing the influence of an already picked center is done using the following formula:

\[
M_2(V_i) = M_1(V_i) - \text{Max}(M_1) \sum_{k=1}^{n} e^{-\beta \, d(V_k, V_i)}
\]

(2.5)

The new maximum point is picked as the second center and the process continues. The reduction (2.5) will guarantee that the points closer to the previous center will have a lower chance of being picked.

More generally, at step \( k \), \( M_k(V_i) = M_{k-1}(V_i) - \text{Max}(M_{k-1}) \sum_{k=1}^{n} e^{-\beta \, d(V_k, V_i)} \)

(2.6)

Fig 2.17 Example of Mountain method applied to three clusters positioned in a triangle shape [100]
The process continues until:

\[
\frac{\text{Max}(M_1)}{\text{Max}(M_k)} < \delta
\]  

(2.7)

After creating a grid from the dataspace, the potential of each point is computed based on the number of neighbours in the grid. The more neighbours that surround the point, the greater its potential value of becoming a cluster center. The point with the highest potential value is chosen as the first cluster center. After choosing the first cluster, the potential of all other points is reduced proportionally to their distance to the first cluster (the closer they are to the first center, the higher the reduction in their potential). The point with the highest potential, after potential reduction, is chosen as the second cluster center. The process continues until the potential of any point falls under a given threshold.

Fig 2.18. Subtractive clustering on an artificial 3-dimensional dataset: 4 of the 5 centers were correctly placed \((r_\alpha = 0.25\) with 5 iterations).

This method, although simple to implement and effective for small datasets, is computationally exponential becoming harder to solve as the number of variables and number of
dimensions increases. For example, having 2 variables with each dimension having 10 grid lines will result in \(10^2\) points that need to be evaluated. Besides, the performance of this approach is sensitive to initialization of the \(\alpha\), \(\beta\) and the stopping criterion \(\delta\) parameters.

Rather than evaluate all grid points, Chiu in [101] proposed a modified version of the Mountain Method, called Subtractive Clustering to initialize iterative optimization cluster centers applicable both to soft (FCM) and hard (KM) clustering algorithms.

Instead of computing the potential of every grid point, this method computes only the potential of the data points under consideration. This method does not have to tradeoff between computational complexity and accuracy as with the Mountain Method where users control the accuracy through grid resolution (the more grid lines, the more accurate the algorithm). Rather than using grid points, a potential is computed for each data point as follows:

\[
P_1(x_i) = \sum_{j=1}^{n} e^{-\alpha \|x_i-x_j\|^2}
\]

where \(\alpha\) is defined as positive parameter equal to:

\[
\alpha = \frac{4}{r_{a}^2}
\]

(2.9)

The potential of a point is a function of its distance to all other points in the data and the more neighbours (in a radius of \(r_{a}\)) a point has, the higher its potential. In contrast to the mountain method, the potential is associated here with a data point rather than a grid point and the influence of a neighbouring data point decreases exponentially with the square of the distance rather than the distance itself (which eliminates the square root operation, making the computation faster). After selecting the first cluster center, the one having the highest potential function, the algorithm proceeds in the same manner as the mountain method with a destruction phase in which the influence of the points closer to the first center are greatly reduced using formula (x).

\[
P_i(x_i) = P_i(x_i) - \max(P_1(x_i)) \sum_{j=1}^{n} e^{-\beta \|x_i-x_j\|^2}
\]

(2.10)

where \(\beta\) is defined as:

\[
\beta = \frac{4}{r_{b}^2}
\]

(2.11)
The amount of the reduction in potential is a function of the distance to the previously picked center: the closer a point is to the sampled center, the higher the reduction in its potential as a candidate. Chiu [60] proposes \( r_b = 1.5r_a \) as a good candidate value for the parameter. The criteria for stopping the algorithm can be the same as (eq. 2.7) dependent on the parameter \( \delta \).

Fig 2.19. FCM++ on the artificial 3-dimensional dataset: 5 of the 5 centers were placed closer to the clusters real center

Chiu proposes a more complex criterion for stopping the algorithm which, unfortunately, introduces another parameter that needs careful tuning. In Fig. 2.18 we can observe subtractive clustering in action on a three-dimensional data space containing 5 separate clusters. We note that the algorithm failed to place one cluster center at the bottom right of the hypercube. In contrast, our proposed method, called FCM++ and described in Chapter 3 (Fig. 2.19), needs only one parameter and places the centers close to their actual location resulting in a faster convergence times.
2.4 Summary

In this chapter we presented the process of cluster analysis and discussed its four main steps: feature selection/extraction, cluster algorithm selection, cluster validation and cluster results interpretation. We have surveyed the existing literature and described several ways in which clustering can be performed. The theory of fuzzy sets (an extension of the standard set theory) and its benefits has been described and the FCM algorithm was introduced. Improvements of the standard algorithm, such as the Mountain Method and Subtractive Clustering, were discussed.

The next chapter will introduce an important contribution of this thesis, the FCM++ algorithm for initialization of the FCM. We will show that spreading the centers in the data space can result in faster convergence (with the algorithm needing far fewer iterations to stop at a minima of its cost function) and higher quality solutions as expressed by validity indexes.
3. FCM++: Fuzzy C-Means with effective seeding

3.1 Introduction

We surveyed the literature on cluster analysis in general and Fuzzy C-means (FCM) in special. This chapter introduces the Fuzzy C-means++ (FCM++) algorithm, the main contribution of this thesis. We show that by utilizing the seeding mechanism of the K-means++ (KM++) algorithm we can improve the effectiveness and speed of Fuzzy C-means (FCM). The KM++ method, proposed in [15] initializes the cluster centers of the K-means (KM) algorithm by selecting points in the dataset that are further away from each other in a probabilistic manner. This method both avoids the problems of the standard method and improves speed of convergence, being theoretically guaranteed to be $O(\log k)$, and hence competitive with the optimal solution. To reduce the computational burden of the standard FCM algorithm and at the same time increase its accuracy, we propose an integration of the KM++ seeding algorithm into the standard version of FCM. Compared to other methods used for initializing FCM, our method does not need as many additional parameters that require careful tuning (as in the case of the mountain method [100], subtractive clustering [101] and hybrid evolutionary methods [92]) in order to achieve better results in terms of both fewer iterations and qualitatively superior structures (in terms of validity index). We provide a proof showing that the performance after initialization is $O(\log k)$ competitive.

By careful seeding that disperses the initial cluster centers through the data space, our approach samples starting cluster representatives during the initialization phase. The cluster representatives are spread in the input space and we control the spreading through an added parameter. We will show that if good data separation can be achieved, higher values of this parameter (resulting in higher spread) can translate into much faster convergence times and better clustering solutions in terms of validity indexes than the standard FCM. In contrast, if the data is less separable, containing overlapping clusters, the algorithm still outperforms the standard but a lower value of this parameter might be appropriate.

First, we investigate cluster quality and iteration count as we vary the spreading factor on a series of synthetic data sets. These synthetic datasets were generated as mixtures of Gaussians in two dimensions covering multiple cases:

1. equal sized clusters that do not overlap (Section 3.4.1);
2. equal sized clusters that overlap (Section 3.4.2);
3. different sized clusters that do not overlap (Section 3.4.3), and
4. different sized clusters that overlap (Section 3.4.4).

The choice of two dimensions was made in order to be able to see graphically how these centers are initialized using both the standard FCM as well as FCM++ with different parameter values. It is important to note that while FCM picks all the centers at random, FCM++ still presents some non-determinism because it picks the first center at random. To account for the non-determinism, we record the average of ten runs while choosing different p parameter values in the range of 0.1 to 3. It can be seen that the representatives sampled with our proposed method (Figs. 3.2, 3.6, 3.10, 3.14 yellow and green triangles) start in a much better position, closer to the real center of the clusters (marked in magenta) and thus require fewer steps to converge. Choosing a higher p-parameter value is advisable when the data points show a high spread in the dataspace. As an example, Fig 3.3 shows that a value higher than 1.3 for p results in a general improvement of 3 times fewer iterations over the standard FCM. Conversely, when the clusters are overlapping Fig 3.14-3.15, a high parameter value for p does not necessarily translate into a faster convergence. Overall, a p-value of value 1.8 was chosen as a good compromise choice that works well on all the synthetic datasets under test. The results show that the proposed method gives significant improvement in convergence times (the number of iterations) on the synthetic datasets of up to 40 (2.1 on average) times the standard on synthetic datasets and, in general, an associated lower cost function value and Xie-Beni value and Partition Entropy.

Secondly, we have additionally considered a series of real world-datasets to show how the benefits of initial cluster center choices translate into improvements that have practical relevancy. IRIS, SPAM and WINE and TRANSFUSIONS are four real-world datasets from the UCI Machine Learning Repository [103] that we have examined together with LIFE EXPECTANCY dataset from [104] [105]. By using a p-parameter with value 1.8, which was show to achieve best performance on artificial datasets, and letting the k parameter (number of clusters) run on a range of [2,10], we obtain overall fewer iterations and a higher quality solution in terms of Xie-Beni value and Partition Entropy validity indexes. For our implementation, we used the R programming language [106] together with the e1071 package [107] which, as well as standard clustering algorithms, contains useful cluster validity functions to test the quality of the discovered structures.
The remainder of this chapter is structured as follows: in **Section 3.2** the proposed algorithm is introduced; **Section 3.3** presents a sketch proof that shows the theoretical bounds of the expected cost function; **Section 3.4** presents the datasets and the evaluation procedure used, and compares the proposed scheme with the standard algorithm on synthetic datasets; **Section 3.5** compares the algorithms on real world datasets and **Section 3.6** concludes and summarizes findings.

### 3.2 FCM++ description

<table>
<thead>
<tr>
<th>Algorithm 2: Fuzzy C-means++</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Given</strong></td>
</tr>
<tr>
<td>a set of N data points $X = {x_i}_{i=1}^{N}$</td>
</tr>
<tr>
<td>$k$ representing the clusters number</td>
</tr>
<tr>
<td>$p$ representing the spreading factor</td>
</tr>
<tr>
<td>$m$ representing the fuzziness parameter</td>
</tr>
<tr>
<td><strong>Return</strong></td>
</tr>
<tr>
<td>a set $R$ of representatives</td>
</tr>
<tr>
<td>a membership matrix $U$.</td>
</tr>
</tbody>
</table>

1. $R_i = x_i$ sampled uniformly from $X$
2. $\text{dist} = \text{vector of size } N$
3. for (j in 1 to N): $\text{dis}[j] = \text{distance}(R_i, X_i)$
4. for (j in 2 to k):
   5. $R_j = \text{sample from } X \text{ with } \text{prob} = \text{dist}^p/\text{sum} (\text{dist}^p)$
   6. for (i in 1 to N):
      7. $\text{aux} = \text{distance}(R_j, X_i)$
      8. if (aux < $\text{dis}[j]$), $\text{dis}[j] = \text{aux}$
9. repeat
10. $U_{ij} = \frac{1}{\sum_{j=1}^{k} \left( \frac{1}{|x_i - r_j|} \right)^{m-1}}$
11. $r_j = \frac{1}{\sum_{i=1}^{n} \mu_{ij}^m} \sum_{i=1}^{n} \mu_{ij}^m x_i$
12. until $|U^{k+1} - U^k| < \varepsilon$

Fig 3.1 FCM++ algorithm

Initializing representatives by selecting random points from the input dataset results in a sub-optimal starting strategy for the standard version of the algorithm: consider Figs. 3.2, 3.6, 3.10,
3.14. The idea behind the FCM++ algorithm (Fig 3.1) is to choose points that are spread out in the dataset as representatives and update the membership matrix accordingly before commencing the iterative steps of FCM.

The proposed algorithm (see Fig 3.1: Algorithm 2) adds representatives initialization instructions (Steps 1->8) to the Fuzzy C-means algorithm (Steps 9->12). The algorithm receives as an input a set of data points $X$ (of size $N$), a parameter $k$ specifying the number of representatives, a parameter $p$ representing the spreading factor (or how should we skew the sampling distribution during initialization) and $m$ representing the fuzziness parameter (defining how fuzzy the membership matrix $U$ should be, by default initialized with $m=2$). Algorithm 2 returns a set of final representatives (centers) $R$ and a membership matrix $U$.

At the beginning the set $R$ is an empty matrix of size $k \times \text{dim}(X)$ where $k$ is the input parameter representing the number of clusters and $\text{dim}(X)$ represents the number of columns of $X$ (the number of dimensions of the dataset). The vector “dist” of size $N$ is a placeholder for the shortest distances from points in $X$ to the representatives in $R$ (for example, $\text{dist}[i]$ represents the distance from point $x_i$ to the closest representative in $R$).

At Step 1 a point $x_i$ is sampled uniformly at random from $X$ and added to $R$ as $R_1$. Dist vector of size $N$ is initialized at Step 2 and then the distance from each point in $X$ to $R_1$ is computed at Step 3. Steps 4 to 8 pick the remaining representatives $R_j$ $(j \in [2,k])$ by sampling points $x_i$ in $X$ based on the vector of distances “dist”. This vector stores the closest distances from each point to the centers in $R$ and it defines the probability distribution when sampling is performed at Step 5. Whenever a center $R_j$ is added to $R$, the distance to it from every point is computed at Step 7. Points that are closer to it than any other representative will update their distance entry in dist vector at Step 8 which in turn will lesser their probability of being picked as the next centers.

Using the distance $d^p$ (where $d^p(x, R)$ denotes the distance (raised to power $p$) from a point $x \in X$ to its closest representative in $R$) allows for controlling the spreading factor of the algorithm by the parameter $p$. A small value for $p$ will pick points closer together while a larger $p$ will pick points that are further away as initial points. In the extreme case of $p=0$, each point $x_i$ will have a random chance of being picked next and the method is similar to random initialization. Conversely, if we choose a $p$ that is very large, we would likely pick outliers as starting points. A tradeoff has to be made and the next section considers ways to choose $p$ depending on the data.
At this point it is important to restate the time complexity of a FCM iteration (Step 9-12 in Fig 3.1) which is quadratic in the number of clusters $O(Nk^2P)$ where $P$ is the dimension of the data, $N$ is the size of the dataset $X$ and $k$ is the number of representatives. We note that at Step 10, the computation of the membership matrix is the dominating operation in the repeat cycle. At Step 10, in order to compute an element $u_{ij} \in U$ one has to sum over whole $k$ range the distances from the point $x_i$ to all the representatives (see denominator at Step 10). This operation requires $k$ distance computations, each composed of additional $P$ calculations (for example, for computing Euclidean distance between two vectors one has to subtract each dimensions as a basic step). Thus we have $k*P$ computations for each member of matrix $U$ of size $N \times k$, resulting in a total of $N * k^2 * P$ computations at each iteration of the FCM algorithm. The total number of iterations cannot be predetermined and depends on the datasets under consideration.

Conversely, Steps 1-8 performing the initialization of the representatives using our proposed scheme has a complexity of $O(NkP)$. This happens because of the two main for loops: one which runs over the $k$ range (Step 4), one which runs over the $N$ range (Step 6) coupled with a distance computation which takes $P$ operations. If an iteration of FCM takes a unit of time and is of $O(Nk^2P)$ complexity then the initialization scheme (Steps 1-8) is expected to take $\frac{NkP}{Nk^2P} = \frac{1}{k}$ of the iteration step. We observe that the larger the $k$ parameter, the less time it will take to sample the representatives relative to an iteration step of the FCM algorithm.

On the assumption that a good clustering is well spread out in the data space, the work in [15] has shown theoretically that the method for KM++ not only achieves $O(\log k)$ approximation of the optimal cost but, by placing the representatives further away from each other at the beginning, the algorithm converges in fewer iterations.

### 3.3 Synthetic dataset evaluation

This section considers ways in which clusters can be distributed and evaluates the performance of both the proposed scheme and the standard version of FCM in terms of speed (the number of iterations to convergence), final cost function value and appropriateness of data partitions using both the Xie-Beni validity index and Partition Entropy. Because of the stochastic nature of FCM (which randomly initializes the membership partition and centers) and the FCM++ method (which picks the first point at random from the data set and the others further away in a probabilistic
manner), each test has been run ten times and the averages computed for the number of iterations, the Xie-Beni function value, Partition entropy and the final cost function value.

Besides the cost function, validity indexes have long been used to assess the quality of the final clusters after FCM execution. There are a multitude to choose from, see [107], and they capture all sorts of properties in the solution that are considered desirable. A simple way to assess the quality of a fuzzy clustering solution is to look at the membership matrix output by the algorithm. If the rows show high fuzziness, meaning the objects have similar memberships to many (if not all) clusters, then the solution is considered bad. The opposite, extreme case is when the objects belong to one cluster with a degree of 1, having 0 membership to the others (hard clustering). As we will see in later examples, many datasets do not contain easily separable clusters: there are points for which it cannot be said for certain that they belong to one and only one cluster. It is important to note that the fuzziness can directly be impacted by the m parameter of the FCM algorithm. In all our examples we used m=2 which is the default value used in literature [39].

Partition entropy is a validity index which assesses the quality of a solution in terms of the fuzziness of the membership matrix alone. Contrary to other types of indices, it does not need the data and/or the centers since it only captures the fuzziness of the solution through the entropy \( H(X) \) (3.6) of the rows in the resulting membership matrix.

\[
H(X) = -\sum_{i=1}^{n} p(x_i) \log(p(x_i))
\]  

(3.8)

\[ 0 \leq H(X) \leq \log(n) \]  

(3.9)

The entropy of the whole membership matrix \( U \) is defined as the sum of entropies for each row divided by the number of rows in \( U \):

\[
H(U; k) = \frac{\sum_{i=1}^{n} H(u_{i:j})}{N}
\]  

(3.10)

When the membership matrix shows a hard partitioning, \( H(U; k) \) will be equal to zero. In contrast, the highest possible value for \( H(U; k) \) is achieved when \( H(U; k) = [1/k] \) or the maximum fuzziness [107]. A lower value of the Partition Entropy index is desirable.

Besides partition entropy, which captures the fuzziness of the solution, we wish to investigate other properties such as the tightness (compactness) of the clusters as well as the spread
of the cluster centers in the final solution. Although several validity indexes have been proposed in the literature (such as [108], etc.) the Xie-Beni [109] is used here as it has been shown to be superior and reliable [110] being defined as:

\[
XB = \frac{\sum_{i=1}^{N} \sum_{j=1}^{K} \mu_{ij}^{m} |x_{i} - r_{j}|^2}{N \min_{i \neq j} |r_{i} - r_{j}|^2}
\]  

(3.11)

In (3.11) the numerator is the FCM cost function \(J_m\) and represents the tightness of the discovered clusters (the higher the value, the more cohesive are the clusters). The denominator captures the spread in the clusters by computing the minimum distance between two neighboring clusters (the higher the value, the better).

The performance of FCM++ has been investigated on four artificially generated and three real-world datasets. The artificial datasets consist of five globular clusters (in two dimensions) that fall into the four possible categories listed earlier: clusters of equal size with no overlap, clusters of different sizes with no overlap, clusters of equal size and overlap and clusters of different sizes with overlap (Figs. 3.2, 3.6, 3.10, 3.14) below.

3.3.1 Equal sized clusters that do not overlap

![Equal sized clusters that do not overlap](image)

Fig 3.2 Equal sized clusters with no overlap. Real cluster centers are marked in magenta, FCM initial clusters are marked in red, while in the yellow and green triangles we have FCM++ (with \(p=0.5\) and \(p=1.8\) respectively)
In Fig. 3.4.1 we observe the outcomes of initializing the centers using the standard method. This method is allowed to freely choose $k$ centers at random from the dataset supplied as input. These centers, marked in red, are often picked from the same clusters and hence have to travel
many steps to their final position (magenta). For this type of data, a power $p=0.5$ for FCM++ generates initial cluster centers (yellow triangles) similar to the default method. In fact, looking at Fig. 3.4.2 we notice that in the range $p \in [0.1, 1.3]$, the FCM++ method produces very volatile results in terms of iterations with little improvement over the standard. A value of $p$ parameter larger than 1.3 tends to result in solutions requiring far fewer iterations than the standard (3 vs 9 with the exception $p=2$). If we were to compare the results of FCM and FCM++ (with $p=1.8$) in terms of solution quality by taking the Xie Beni and Partition Entropy index (Fig 3.4 and 3.5) we observe that overall the indices values are lower for the FCM++ algorithm, sometimes at a very high margin (second independent run). Moreover, there is more volatility in the FCM results showing large swings in solution quality (e.g. second vs fifth run in Fig 3.4) for both the two considered indexes.

### 3.3.2 Equal sized clusters that overlap

![Fig 3.6 Equal sized clusters with some overlap. Real clusters are marked in magenta, FCM initial clusters are marked in red, while in the yellow and green triangles we have FCM++ (with $p=0.5$ and $p=1.8$ respectively).](image)


Fig 3.7 Mean number of iterations on equal sized clusters with overlap function of p.

Fig 3.8 FCM vs FCM++ Partition Entropy
Fig 3.9 FCM vs FCM++ Xie Beni Index

In Fig. 3.6 we present the synthetic dataset that has equal sized Gaussians containing some overlap in the bottom left/right clusters. As in the case of Fig 3.2, the centers picked at random by
the FCM algorithm start in a worse position, requiring much more steps to converge to the actual centers coloured in magenta. Comparing FCM++ solutions we note that for this type of data, a power \( p=0.5 \) for FCM++ generates initial cluster centers (yellow triangles) worse than those generated by FCM++ and \( p=1.8 \) (green triangles). The centers are spread further away for \( p=1.8 \) and we do not get the case where a cluster such as the bottom left one receives two initial centers (\( p=0.5 \), yellow triangles). For values of \( p \) larger than 1, we observe noticeable improvements in terms of iteration count for FCM++ (Fig 3.7).

In terms of solution quality we see that FCM++ with \( p=1.8 \) achieves better, less fuzzy partitions (8 out of 10 times, Fig 3.8) and more spread and tighter clusters (8 out of 10 times, Fig 3.8). The volatility in the solution quality can be high especially for FCM and we notice considerable swings in the Partition Entropy and Xie Beni index (Fig 3.8- Fig 3.9). In conclusion, some overlap between equal sized clusters does result in some swings in solution quality if we run the algorithms multiple times.

### 3.3.3 Different sized clusters that do not overlap

![Different sized clusters](image)

Fig 3.10. Different sized clusters with no overlap. Real clusters are marked in magenta, FCM initial clusters are marked in red, while in the yellow and green triangles we have FCM++ (with \( p=0.5 \) and \( p=1.8 \) respectively).
Fig 3.11. Mean number of iterations on different sized clusters with no overlap, function of p

Fig 3.12 FCM vs FCM++ Partition Entropy Index on different sized clusters with no overlap

Fig 3.13 FCM vs FCM++ Xie Beni Index on different sized clusters with no overlap

Fig. 3.10 shows different sized synthetic Gaussian clusters with no overlap and the centers picked by FCM (red) and FCM++ (p=0.5 in shown in yellow triangles and p=1.8 in green triangles). FCM algorithm is known to be sensitive when clusters of unequal size are present [111]. This happens because equal cluster sizes result in smaller objective function values, which
is what the algorithm tries to minimize [11], [112], [113]. We notice that on this dataset FCM++
(with a small p value of 0.5) makes more mistakes than the standard FCM, needing more iterations
to converge (Fig 3.11). As an example, it has mistakenly picked three centers within the top-middle
larger cluster at close proximity (shown in yellow triangles Fig 3.10). A FCM++ with p=1.8
spreads the centers further away, although it makes the mistake of choosing two cluster centers in
the same cluster (Fig 3.10, middle-top cluster). FCM performs better than FCM++ (with p=0.5)
but worse than FCM++ (with p=1.8) picking two initial centers in top-middle cluster and two
initial centers in bottom-left clusters. If we vary the p-parameter (Fig 3.11) we observe that the
solutions tend to be better (with two exceptions) for FCM++ when the parameter becomes larger
than 0.6. The solution quality tends to be very volatile in terms of both Partition Entropy and Xie
Beni (Fig 3.12-3.13), for each FCM and FCM++ due to the sensitiveness of these algorithms on
unequal sized clusters.

3.3.4 Different sized clusters that overlap

Fig 3.14. Different sized clusters with overlap. Real clusters are marked in magenta, FCM initial
clusters are marked in red, while in yellow and green triangles we have FCM++ (with p=0.5 and
p=1.8 respectively)
Fig 3.15. Mean number of iterations on different sized clusters with overlap function of p.

Fig 3.16 FCM vs FCM++ Partition Entropy on different sized clusters with some overlap

Fig 3.17 FCM vs FCM++ Xie Beni Index Index on different sized clusters with some overlap

Fig. 3.14 shows what could be considered a worst case scenario. Here we have five different sized Gaussian clusters that overlap with their true center marked in magenta. The FCM randomly picks the centers in red during initialization which are further away from the true ones (magenta)
than those sampled by FCM++ with a minimum spread \((p=0.5)\) in yellow triangles. FCM++ with \(p=1.8\) has spread the centers (green triangles) a bit further from one another but it fails to get the leftmost cluster. The improvements for FCM++ that were seen on previous synthetic datasets tend to be less present in this case (Fig 3.15). Although it outperforms the standard by a high margin (when \(p\) is chosen on intervals such as 1.8 to 2.2), FCM++ shows a higher volatility than normal. During ten independent runs, the solution quality for FCM is worse in terms of both Partition Entropy (Fig 3.16) and Xie Beni (Fig 3.17) although less volatile than FCM++.

We have investigated the behavior of FCM versus FCM++ on four synthetic data sets shown in Figs. 3.2, 3.6, 3.10 and 3.14. The analysis shows that the improvements of FCM++ in terms of iteration and solution quality tend to be more significant the more separated and equal the cloud of synthetic Gaussians are. A FCM++ parameter of \(p=1.8\) achieved good results on all these datasets under considerations.

### 3.4 Real world dataset evaluation

Four real world datasets are considered: IRIS, SPAM and WINE (from the UCI Machine Learning Repository [103]) and one, Life Expectancy, from [105]. For the FCM++ algorithm we pick a power \(p=1.8\), a value proved to achieve best performance on all the artificial cases considered above: it does not spread the centers too far (which could result in outliers being selected) and it does not make them too close (which would result in performance similar to the standard approach). As in the synthetic datasets case, we will analyze the solution quality using the Partition Entropy and Xie-Beni indices.

#### 3.4.1 Iris dataset

The Iris Dataset consists of 150 instances belonging to one of three classes each referring to a particular type of Iris plant: setosa, versicolor or virginica. There are four features representing sepal width and length, petal length and petal width.

The data, being multidimensional (with more than three dimensions), cannot be viewed directly in one figure and thus we need a method to project the data into a lower dimensional space.
that can be viewed and analyzed. For this task, we use the MDS algorithm [152] which takes the matrix of dissimilarities between points as an input and reconstructs a map that preserves them in a lower space. These dissimilarities do not have to present all the properties associated with metric functions such as symmetry, triangle inequality, etc. and there are non-metric MDS algorithms as well [149].

![MDS on Iris Dataset](image)

Fig 3.18. MDS on Iris Dataset shows a clear separation between setosa in red and the cloud of points made up by virginica (green) and versicolor (blue)

Given a dissimilarity matrix $D = (d_{ij})$, MDS seeks to find $x_1, x_2, x_3 \ldots x_n \in \mathbb{R}^p$ such that $d_{ij} \approx \|x_i - x_j\|^2$ [149] should be as close as possible in $\mathbb{R}^p$ with $p$ usually being defined in a range from 1 to 3 (that can be plotted in one figure). For finding the coordinates of $x_1$ to $x_n$, a loss function (3.12), usually defined as “stress” or “strain function”, is expressed in terms of residual sums of squares.

$$Stress_D(x_1, x_2, \ldots, x_n) = \left( \frac{\sum_{i,j}(d_{ij} - \|x_i - x_j\|^2)^2}{\sum_{i,j}(d_{ij})^2} \right)^{1/2}$$ (3.12)
One way in which (3.12) can be minimized is by using a steepest descent approach, as described in [150]. It is important to note that the objects in the projected lower dimensional space, while preserving the distance as close as possible to the ones in high dimensions, do not have any orientation and the axes value range do not have a particular meaning [151].

Throughout this thesis we have used MDS to visualize and cluster in the lower dimensional space various datasets. The similarity between objects in high dimensions was computed using Euclidean distance on the simpler datasets presented in this chapter and we have used Pearson Correlation for time series in Chapter 4 and Cosine distance for text data in Chapter 5.

By performing an MDS scale on the Iris data (Fig 3.18) we notice the existence of two overlapped clusters of points. These two clusters represent the iris versicolor and virginica and are harder to distinguish. We also note that each cluster has an equal amount of points: 50 observations.

<table>
<thead>
<tr>
<th>IRIS Dataset</th>
<th>Number of iterations</th>
<th>Number of iterations</th>
<th>Within Error</th>
<th>Within Error</th>
<th>Xie Beni</th>
<th>Xie Beni</th>
<th>Partition entropy</th>
<th>Partition entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FCM</td>
<td>FCM++</td>
<td>FCM</td>
<td>FCM++</td>
<td>FCM</td>
<td>FCM++</td>
<td>FCM</td>
<td>FCM++</td>
</tr>
<tr>
<td>K=2</td>
<td>7.9</td>
<td>7.5</td>
<td>0.85929</td>
<td>0.859299</td>
<td>0.000361</td>
<td>0.000361</td>
<td>0.19574</td>
<td>0.19574</td>
</tr>
<tr>
<td>K=3</td>
<td>23.6</td>
<td>16.3</td>
<td>0.40337</td>
<td>0.40337</td>
<td>0.000912</td>
<td>0.000912</td>
<td>0.39549</td>
<td>0.39549</td>
</tr>
<tr>
<td>K=4</td>
<td>28.5</td>
<td>20.9</td>
<td>0.28272</td>
<td>0.293331</td>
<td>0.001593</td>
<td>0.002174</td>
<td>0.56802</td>
<td>0.56572</td>
</tr>
<tr>
<td>K=5</td>
<td>32.1</td>
<td>23.3</td>
<td>0.23509</td>
<td>0.218620</td>
<td>0.003126</td>
<td>0.001886</td>
<td>0.68938</td>
<td>0.68938</td>
</tr>
<tr>
<td>K=6</td>
<td>54.5</td>
<td>43.9</td>
<td>0.18737</td>
<td>0.178870</td>
<td>0.003396</td>
<td>0.002843</td>
<td>0.81549</td>
<td>0.80694</td>
</tr>
<tr>
<td>K=7</td>
<td>48.6</td>
<td>37.3</td>
<td>0.16049</td>
<td>0.149680</td>
<td>0.005690</td>
<td>0.002834</td>
<td>0.9029</td>
<td>0.90036</td>
</tr>
<tr>
<td>K=8</td>
<td>35.7</td>
<td>55.8</td>
<td>0.14239</td>
<td>0.123284</td>
<td>0.006304</td>
<td>0.002155</td>
<td>0.99773</td>
<td>1.00007</td>
</tr>
<tr>
<td>K=9</td>
<td>50.4</td>
<td>47.3</td>
<td>0.11214</td>
<td>0.108072</td>
<td>0.004296</td>
<td>0.003102</td>
<td>1.07967</td>
<td>1.07225</td>
</tr>
<tr>
<td>K=10</td>
<td>37</td>
<td>59.2</td>
<td>0.09999</td>
<td>0.098921</td>
<td>0.005141</td>
<td>0.004666</td>
<td>1.14639</td>
<td>1.13514</td>
</tr>
</tbody>
</table>

Table 3.1. Iris dataset results
In Table 3.1 we observe that FCM++ outperforms the standard in terms of number of iterations in seven out of nine runs (exceptions being k=8 and k=10). For k=3, which is the real number of clusters in the data, FCM++ needs on average 16.3 iterations to converge while the standard needs 23.6. In terms of cost function at the convergence the algorithms have the same values for k=2 and k=3, lower value (better clustering) for FCM on k=4 and lower values for FCM++ for all other k parameters. The same happens to the Xie Beni index which is equal for lower k parameters but better for FCM++ when k is larger than 5.

### 3.4.2 SPAM dataset

The SPAM dataset (4601 instances) contains attributes describing the characteristics of two categories of emails (solicited and unsolicited) where the features (56 continuous real-valued) describe frequencies of certain words, capital letters run length, etc. The MDS projection of the SPAM dataset (Fig 3.19) in two dimensions reveals an overlapping cloud of points. Across the range of k parameters, the FCM++ algorithm with a power $p=1.8$ outperforms the standard approach in some cases by a large factor (needing on average only 84.5 iterations compared with FCM which needs 217.9 for k=10). With the exception of $k=3$, FCM++ achieves lower cost functions values at convergence.

The Xie Beni index indicating cluster quality is lower (better) for FCM++ across the whole range of k parameters. Over all the dataset (except k=2 where the results are equal) FCM++ outperforms FCM in terms of partition entropy, generating fewer fuzzier membership matrices.
Fig 3.19. MDS on Spam dataset where spam messages are shown in black while spam shown in red.
### Table 3.2. Spam dataset results

<table>
<thead>
<tr>
<th>K</th>
<th>Number of iterations FCM</th>
<th>Number of iterations FCM++</th>
<th>Within Error FCM</th>
<th>Within Error FCM++</th>
<th>Xie Beni FCM</th>
<th>Xie Beni FCM++</th>
<th>Partition entropy FCM</th>
<th>Partition entropy FCM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>26.9</td>
<td>23.9</td>
<td>137448.1</td>
<td>137448.1</td>
<td>1.0851</td>
<td>1.084566</td>
<td>0.097</td>
<td>0.097</td>
</tr>
<tr>
<td>3</td>
<td>34.8</td>
<td>23.9</td>
<td>73642.85</td>
<td>81690.52</td>
<td>2.1165</td>
<td>1.292544</td>
<td>0.162</td>
<td>0.152</td>
</tr>
<tr>
<td>4</td>
<td>43.7</td>
<td>31.8</td>
<td>49749.26</td>
<td>49509.64</td>
<td>4.7718</td>
<td>1.864137</td>
<td>0.239</td>
<td>0.202</td>
</tr>
<tr>
<td>5</td>
<td>71.8</td>
<td>37.4</td>
<td>38253.01</td>
<td>31621.59</td>
<td>6.5946</td>
<td>3.75236*</td>
<td>0.302</td>
<td>0.228</td>
</tr>
<tr>
<td>6</td>
<td>145.1</td>
<td>48.4</td>
<td>30797.2</td>
<td>25292.62</td>
<td>6.1485</td>
<td>4.307133</td>
<td>0.321</td>
<td>0.297</td>
</tr>
<tr>
<td>7</td>
<td>76.9</td>
<td>57.7</td>
<td>25475.89</td>
<td>19654.03</td>
<td>0.0001</td>
<td>4.099956</td>
<td>0.409</td>
<td>0.334</td>
</tr>
<tr>
<td>8</td>
<td>145.3</td>
<td>61.3</td>
<td>21962.28</td>
<td>15058.67</td>
<td>0.0001</td>
<td>5.906694</td>
<td>0.432</td>
<td>0.368</td>
</tr>
<tr>
<td>9</td>
<td>99.5</td>
<td>66.7</td>
<td>18817.46</td>
<td>11659</td>
<td>0.0002</td>
<td>7.696938</td>
<td>0.504</td>
<td>0.416</td>
</tr>
<tr>
<td>10</td>
<td>217.9</td>
<td>84.5</td>
<td>17062.23</td>
<td>8919.816</td>
<td>0.0002</td>
<td>7.642981</td>
<td>0.512</td>
<td>0.441</td>
</tr>
</tbody>
</table>

Table 3.2. Spam dataset results

### 3.4.3 Wine dataset

The WINE dataset consists of 178 instances and 13 attributes (integer and real) resulting from chemical analysis of wines derived from three types of cultivars. This analysis reveals properties
in the wines such as alcohol content, color intensity, flavonoids and so on. MDS analysis in two dimensions shows three clouds of points with a degree of overlap between two of them (blue and green). This means that two cultivars have an overlapping offer of wines sharing similar qualities. Table 3.3 shows a significant reduction in the number of iterations until FCM++ converges for this dataset on most of the k range, outperforming the standard in seven out of nine runs, sometimes at a high margin (k=7 needs 63.2 less iterations on average).

<table>
<thead>
<tr>
<th>Wine Dataset</th>
<th>Number of iterations FCM</th>
<th>Number of iterations FCM++</th>
<th>Within Error FCM</th>
<th>Within Error FCM++</th>
<th>Xie Beni FCM</th>
<th>Xie Beni FCM++</th>
<th>Partition Entropy FCM</th>
<th>Partition Entropy FCM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>K=2</td>
<td>11.6</td>
<td>11.9</td>
<td>21006.4044</td>
<td>21006.4044</td>
<td>3.00×10⁻⁴</td>
<td>3.00×10⁻⁴</td>
<td>0.216</td>
<td>0.216</td>
</tr>
<tr>
<td>K=3</td>
<td>28.7</td>
<td>25.8</td>
<td>10090.3527</td>
<td>10090.3527</td>
<td>7.00×10⁻⁴</td>
<td>7.00×10⁻⁴</td>
<td>0.3804</td>
<td>0.3804</td>
</tr>
<tr>
<td>K=4</td>
<td>30.1</td>
<td>25.6</td>
<td>5388.1255</td>
<td>5388.1255</td>
<td>5.00×10⁻⁴</td>
<td>5.00×10⁻⁴</td>
<td>0.4183</td>
<td>0.4183</td>
</tr>
<tr>
<td>K=5</td>
<td>136.2</td>
<td>126.2</td>
<td>3741.8134</td>
<td>3767.0324</td>
<td>6.00×10⁻⁴</td>
<td>5.00×10⁻⁴</td>
<td>0.5098</td>
<td>0.5053</td>
</tr>
<tr>
<td>K=6</td>
<td>54.7</td>
<td>84.1</td>
<td>2647.6292</td>
<td>2591.9996</td>
<td>7.00×10⁻⁴</td>
<td>7.00×10⁻⁴</td>
<td>0.5235</td>
<td>0.5275</td>
</tr>
<tr>
<td>K=7</td>
<td>106.9</td>
<td>43.7</td>
<td>2066.1265</td>
<td>1673.0150</td>
<td>0.001</td>
<td>5.00×10⁻⁴</td>
<td>0.5696</td>
<td>0.5145</td>
</tr>
<tr>
<td>K=8</td>
<td>82.5</td>
<td>63.3</td>
<td>1712.8443</td>
<td>1347.2349</td>
<td>0.0011</td>
<td>6.00×10⁻⁴</td>
<td>0.6209</td>
<td>0.5521</td>
</tr>
<tr>
<td>K=9</td>
<td>87.8</td>
<td>74.8</td>
<td>1283.6281</td>
<td>1058.1556</td>
<td>0.001</td>
<td>7.00×10⁻⁴</td>
<td>0.6299</td>
<td>0.6034</td>
</tr>
<tr>
<td>K=10</td>
<td>90.7</td>
<td>80.3</td>
<td>1054.6496</td>
<td>897.9707</td>
<td>0.0012</td>
<td>7.00×10⁻⁴</td>
<td>0.6502</td>
<td>0.6287</td>
</tr>
</tbody>
</table>

Table 3.3. Wine dataset results
Fig 3.20. MDS on Wine Dataset shows an overlap between the clusters

The cost function, while has equal values for lower k parameters, diverges significantly in favor of FCM++ from k=6 to k=10. We observe the same pattern with the Xie-Beni index which starts equal for both and diverges on higher k values in favor of FCM++. With the exception of k=6 the same holds true for partition entropy index. In three out of ten runs, the algorithms had the same partition entropy index value, the FCM outperformed the standard (k=6) one time and on five occasions FCM++ outperformed the standard (k=5, k=7, k=8, k=9 and k=10).
Table 3.4. Life Expectancy dataset results

<table>
<thead>
<tr>
<th>K=2</th>
<th>Number of iterations FCM</th>
<th>Number of iterations FCM++</th>
<th>Within Error FCM</th>
<th>Within Error FCM++</th>
<th>Xie Beni FCM</th>
<th>Xie Beni FCM++</th>
<th>Partition Entropy FCM</th>
<th>Partition Entropy FCM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.8</td>
<td>20.1</td>
<td>73.493 101</td>
<td>73.49310 1</td>
<td>0.00164 1</td>
<td>0.00164 1</td>
<td>0.205633 0.20560 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K=3</td>
<td>17.2</td>
<td>14.8</td>
<td>35.308 799</td>
<td>35.30879 9</td>
<td>0.00874 0.00874</td>
<td>0.518218 0.51821 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K=4</td>
<td>94.5</td>
<td>81.3</td>
<td>23.837 163</td>
<td>23.83716 3</td>
<td>0.01451 9</td>
<td>0.01451 4</td>
<td>0.785726 0.78573 9</td>
<td></td>
</tr>
<tr>
<td>K=5</td>
<td>76.3</td>
<td>55.0</td>
<td>15.887 455</td>
<td>15.88745 5</td>
<td>0.00637 3</td>
<td>0.00637 4</td>
<td>0.793197 0.79321 9</td>
<td></td>
</tr>
<tr>
<td>K=6</td>
<td>72.9</td>
<td>57.4</td>
<td>11.523 295</td>
<td>11.523295 5</td>
<td>0.01009 3</td>
<td>0.01009 2</td>
<td>0.894872 0.89487 3</td>
<td></td>
</tr>
<tr>
<td>K=7</td>
<td>54.4</td>
<td>56.6</td>
<td>9.4216 88</td>
<td>9.297488 2</td>
<td>0.00895 2</td>
<td>0.00881 7</td>
<td>0.956203 0.95761 1</td>
<td></td>
</tr>
<tr>
<td>K=8</td>
<td>57.8</td>
<td>45.5</td>
<td>7.4610 4</td>
<td>7.551925 9</td>
<td>0.00726 9</td>
<td>0.00718 7</td>
<td>1.007804 1.00075 7</td>
<td></td>
</tr>
<tr>
<td>K=9</td>
<td>52.2</td>
<td>42.3</td>
<td>6.3498 17</td>
<td>6.431745 0.00757</td>
<td>0.00705 1.04923 0.102296 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K=10</td>
<td>56.1</td>
<td>54.3</td>
<td>5.4038 35</td>
<td>5.483383 0.00622</td>
<td>0.00618 1.051219 1.06972 4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.4.4 Life expectancy dataset

The Life expectancy dataset is available as part of package “cluster.datasets” [105] and was first described in [104]. It contains 31 countries and their life expectancy for males and females of different ages. For each sex there are 4 numeric feature vectors representing the life expectancy when a person is born, at 25, 50 and 75 respectively. The total usable number of features is thus 8 and Fig 3.21 shows the scaled dataset (MDS) in two dimensions. We observe the countries with the highest life expectancy being located close to the top right corner while the countries with the
lowest life expectancy are located towards the left. As a comparison, the life expectancy at birth in Cameroon was 34 for males and 38 for females, whilst in the US the whites had a male life expectancy of 68 and females one of 75.

Fig 3.21. MDS on Life expectancy dataset

Countries in Central and South America were situated in the middle having an overall better life expectancy than those in Africa but worse than those in US, Canada, etc. Arguably there could be many factors behind this high disparity but access to high quality medical services is a prominent one. Table 3.4 shows the results of running the FCM and FCM++ (p=1.8) for k in
a range from 2 to 10 on Life Expectancy dataset. Unlike the cases we have seen so far, there is no known number of clusters, in other words we do not know the ground truth beforehand. In terms of number of iterations, FCM++ outperforms the standard for the whole k range. The highest difference in average number of iterations occurs when k=5 where FCM++ needs 22.3 fewer iterations to converge to the local minimum of the cost function. In terms of Partition Entropy and Xie Beni there is no significant difference between the two. The cost function value at the end is approximately equal for both algorithms, indicating discovery of the same cluster centers at convergence.

### 3.4.5 Transfusion dataset

<table>
<thead>
<tr>
<th>Transfusion Dataset</th>
<th>Number of iterations FCM</th>
<th>Number of iterations FCM++</th>
<th>Within Error FCM</th>
<th>Within Error FCM++</th>
<th>Xie Beni FCM</th>
<th>Xie Beni FCM++</th>
<th>Partition Entropy FCM</th>
<th>Partition Entropy FCM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>K=2</td>
<td>22.3</td>
<td>19</td>
<td>602429.607</td>
<td>602429.607</td>
<td>1.00* 10^-4</td>
<td>1.00* 10^-4</td>
<td>0.2016</td>
<td>0.2016</td>
</tr>
<tr>
<td>K=3</td>
<td>86.8</td>
<td>78.5</td>
<td>310162.04</td>
<td>305119.195</td>
<td>2.00* 10^-4</td>
<td>1.00* 10^-4</td>
<td>0.3184</td>
<td>0.2819</td>
</tr>
<tr>
<td>K=4</td>
<td>51.5</td>
<td>31.8</td>
<td>140231.216</td>
<td>140231.216</td>
<td>1.00* 10^-4</td>
<td>1.00* 10^-4</td>
<td>0.3306</td>
<td>0.3306</td>
</tr>
<tr>
<td>K=5</td>
<td>88.6</td>
<td>65.3</td>
<td>87296.498</td>
<td>87296.498</td>
<td>1.00* 10^-4</td>
<td>1.00* 10^-4</td>
<td>0.3954</td>
<td>0.3954</td>
</tr>
<tr>
<td>K=6</td>
<td>188.4</td>
<td>109.3</td>
<td>59302.277</td>
<td>59302.277</td>
<td>1.00* 10^-4</td>
<td>1.00* 10^-4</td>
<td>0.4411</td>
<td>0.4411</td>
</tr>
<tr>
<td>K=7</td>
<td>125.8</td>
<td>94</td>
<td>40811.353</td>
<td>41890.631</td>
<td>2.00* 10^-4</td>
<td>2.00* 10^-4</td>
<td>0.4741</td>
<td>0.4723</td>
</tr>
<tr>
<td>K=8</td>
<td>119.2</td>
<td>66.1</td>
<td>30450.763</td>
<td>30813.347</td>
<td>2.00* 10^-4</td>
<td>2.00* 10^-4</td>
<td>0.4953</td>
<td>0.4937</td>
</tr>
<tr>
<td>K=9</td>
<td>170</td>
<td>69.7</td>
<td>24936.225</td>
<td>25671.997</td>
<td>3.00* 10^-4</td>
<td>2.00* 10^-4</td>
<td>0.5218</td>
<td>0.5018</td>
</tr>
<tr>
<td>K=10</td>
<td>186.7</td>
<td>108.4</td>
<td>20912.087</td>
<td>19942.933</td>
<td>4.00* 10^-4</td>
<td>2.00* 10^-4</td>
<td>0.4113</td>
<td>0.4951</td>
</tr>
</tbody>
</table>

Table 3.5 Transfusion dataset results
The transfusion dataset (from Blood Transfusion Service Center in Hsin-Chu City in Taiwan) [103] contains data about persons who have donated blood. It includes four attributes detailing the number of months since last donation, frequency (or the number of total donations), the total amount of blood donated and the number of months since first donation. There is an additional feature, a binary variable, indicating whether the person donated blood (during March 2007) or not. We take this attribute as the class label and we use only the first four attributes. Fig 3.22 shows the MDS scaled transfusion datasets showing two overlapping clouds of points coloured based on the binary feature.
Table 3.5 shows a large improvement using FCM++ over the whole k range, including the special case k=2 (we know that we can divide the data into two sets: those who donated and those who did not). The results in terms of cost function at convergence are mixed with FCM++ outperforming FCM on some k parameters (k=3, k=10) while FCM outperforms FCM++ on others (k=7, k=8, k=9). The cost function is equal in value for k=2, k=4, k=5 and k=6. The Xie Beni index has better (or equal to FCM) values for FCM++ on the whole range of k parameters and the same holds for Partition entropy with the exception of k=10 where the standard has a better value.

3.5 Summary

This chapter has empirically investigated the effectiveness of introducing the KM++ initialization scheme into the context of FCM. After introducing the FCM++ algorithm, we provided a sketch proof showing the theoretical bounds of the expected cost function.

We have presented case studies on artificial data sets that cover the following range of possibilities: equal sized clusters that do not overlap, equal sized cluster having some overlap, non-equal sized clusters with no overlap and non-equal sized clusters with overlap. The results show that when the clusters are of equal size our proposed method tends to outperform the standard by a high margin for p value parameters greater than 1.3. The sensitivities of FCM to non-equal sized clusters tend to translate into some performance degradation of FCM++ over FCM; while it still outperforms the standard on most p value range the behavior in terms of iteration count is more volatile and erratic. A limitation of the proposed method remains the choice of p parameter representing the spreading factor. A choice of p=1.8 achieved good results on a range of overlapping/non-overlapping, equal/non-equal synthetic datasets as well as on several real world datasets.

On real-world data sets, the algorithm produced encouraging results in terms of both the number of iterations needed to reach convergence and final cluster quality. On each dataset tested, the FCM++ algorithm achieved fewer iterations in general (particularly where k=the real number of clusters) with similar if not significantly better Xie-Beni and Partition entropy values suggesting overall higher quality clusters.

The most impressive results were obtained on the artificially generated medium overlap and equal sized clusters, where FCM++ was on average 2.1 times faster than the standard (40 times
the standard for some particular $p$ parameter values). On real-world datasets we have seen improvements over the standard ranging from 1.12 on the SPAM dataset up to 1.44 on IRIS, when the $k$ parameter equals true number of clusters in the dataset. We also observe improvements in terms of iteration count on the whole range of $k$ values.

The datasets that we have considered so far are typical in the sense that the features that they contain are all numeric and there is no relationship between them. In the next chapter we look at financial data, a special type of data that has temporally ordered features. Instead of using FCM/FCM++ directly (as we did in this chapter), these types of datasets need preprocessing in order to capture higher level concepts such as correlations.
4. Time series case studies

4.1 Introduction

In Chapter 3 we introduced the Fuzzy C-means++ (FCM++) algorithm and evaluated its performance in terms of speed and cluster quality on standard datasets, that is a dataset that has no implicit or explicit order defined between its constituent variables. In this chapter we consider a special type of dataset that has an implicit, temporal, ordering on its features: time series data. Unlike standard datasets where Euclidean distance might suffice for clustering, time series data, especially as used in finance, often requires the use of correlation or other, more complex distances [114]. This chapter considers two types of time series data: synthetically generated and historical financial stock data. We discuss a method that uses correlation in order to reduce the dimensions of the data, remove noise and make the dataset easier to interpret and analyse. We show that FCM++ makes a positive impact on the quality and speed when trying to cluster these lower-dimensional, noise reduced, representations of the time series. The end result is a sub-grouping of the time series into clusters based on similar price evolution. This outcome can be informative in investment decisions where the aim is to minimize risk by picking a diversified set of assets.

4.2 Motivation

Discovering insightful patterns in large sets of time series can be a daunting challenge. For example, it is impossible to discern stocks having a similar price evolution if we look at larger time series sets such as the ones shown in Fig. 4.13 and Fig. 4.18. Multi-Dimensional Scaling (MDS) has been used on financial time series [115] as it provides a convenient way of not only reducing the number of dimensions but also making it easier to plot and visualize the time series. We introduce a methodology that combines MDS with a FCM++ approach and results in an lower average number of iterations as well as higher qualitative solutions (in terms of validity indexes) when compared with the standard approach when performing clustering of time series data. As we shall show, in this lower dimensional space, discovered using MDS, interesting patterns emerge that could provide useful insights into the behaviour of the time series.

Although multiple ways of defining similarities on time series data have been proposed in the literature [114] we focus on Pearson Correlation distance. Unlike other measures of similarity, such as those derived from Minkovski distance (e.g. Manhattan or Euclidean distance), correlation
measures the linear relationship between two time series and has applications in the framework of portfolio construction and management in finance [116] [117]. The approach that we introduce here could have practical relevance in the context of diversified portfolio construction as it detects fuzzy clusters of correlated stocks that have lower inter-cluster correlation. Companies listed on the London Stock Exchange are classified into industries based on their activity, however we provide examples showing that stocks of companies in any given industry do not necessarily perform in a similar way over the time interval considered; thus it may be more valuable and interesting to use an empirical approach where we group stocks based on their actual historical price. After a series of pre-processing steps and visual investigations we compare the FCM against the FCM++ algorithm noting an improvement in terms of iterations and cluster quality on both synthetic and financial datasets. Furthermore we assess the performance of each discovered cluster in terms of its returns and volatility.

The chapter is structured as follows: in Section 4.3 we define terminology on time series (Section 4.3.1), Pearson Correlation (Section 4.3.2) and present some limitations of correlation (Section 4.3.3); in Section 4.4 we contrast the FCM++ algorithm with the standard approaches on synthetically generated time series; in Section 4.5 we contrast the FCM++ algorithm with the standard approaches on real world financial datasets; we conclude this chapter in Section 4.6.

4.3 Time Series terminology

4.3.1 Time series

A time series is a special type of data that can be interpreted as a sequence of measurements of a phenomenon taken during a time interval [118]. These measurements, unlike standard datasets, have an implicit temporal relationship between them, being equally spaced in the time interval. It is common to describe a time series using notation (4.1) although other notations have also been used.

\[ X = \{X_1, X_2, X_3, \ldots, X_n\} \] (4.1)

As we can see in (4.1), a time series X is composed of observations \(X_1 \ldots X_n\) where typically \(n\) is naturally high dimensional. We are interested in comparing multiple time series to find insightful patterns. High dimensionality together with the implicit relationship that may exist between the points make it desirable to use similarity measures between time series which capture
this temporal aspect. We want to discover whether multiple time series grow or decline together over time (share a similar stock price evolution).

### 4.3.2 Pearson Correlation

For our experiments, it is important to introduce a method of comparing related time series. As the time series datasets are naturally high dimensional, Euclidean distance can be a very poor choice in discriminating between many time series as during the execution of a clustering procedure \[56\]. Although the metric is popular and widely used for lower dimensional spaces, it performs poorly on high dimensional data: as dimensions increase, there is less differentiation between different pairs of points \[119\] \[56\] \[120\]. Pearson correlation \[121\], usually denoted with \(r\), is well suited to find linear relationships between time series.

Table 4.1 Guideline on interpreting correlations between time series \[7\].

<table>
<thead>
<tr>
<th>Strength of association</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>0.1 to 0.3</td>
<td>-0.1 to -0.3</td>
</tr>
<tr>
<td>Medium</td>
<td>0.3 to 0.5</td>
<td>-0.3 to -0.5</td>
</tr>
<tr>
<td>Large</td>
<td>0.5 to 1</td>
<td>-0.5 to -1</td>
</tr>
</tbody>
</table>

For example, two time series are anti-correlated (negative \(r\)) if they move in opposite directions, that is, when one tends to grow, the other decreases in value. They are positively correlated if when one grows the other one does too. The closer \(r\) is to -1 or 1 the stronger the correlation between the time series (negative and positive). In Table 4.1 \[7\] we see a guideline for determining the strength of the correlation. Correlation \(r\) between two time series \(X\) and \(Y\) of size \(N\) is computed as follows \[121\]:

\[
r(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{N} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N} (X_i - \bar{X})^2 \sum_{i=1}^{N} (Y_i - \bar{Y})^2}} \tag{4.2}
\]

where by \(\bar{X}\) we understand the mean of \(X\), computed as:

\[
\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{4.3}
\]

and \(\text{Cov}(X, Y)\) is the covariance between two time series, being equivalent to:

\[
\text{Cov}(X, Y) = \sum_{i=1}^{N} (X_i - \bar{X})(Y_i - \bar{Y}) \tag{4.4}
\]

while \(\sigma_X\), the standard deviation of a time series \(X\), is defined as:
\[ \sigma_X = \sqrt{\sum_{i=1}^{N}(X_i - \bar{X})^2} \]  

(4.5)

### 4.3.3 Aspects and limitations affecting Pearson Correlation

At this point, it is important to introduce the limitations of correlation as a similarity measure between two time series. If we were to plot two time series against each other we would get plots similar to Fig 4.2. This figure shows the correlation value between different clouds of points representing on one axis the values of time series X and on the other axis the values of Y.

![Example of correlation between time series](image)

Fig 4.2. Example of correlation between time series [148]

There are three aspects that stand out in Fig.4.2:

1. **Scattering of the points:** The top row shows how dispersion of the points along a line tends to either decrease correlation (if the slope is positive) or increase correlation (on the negative slope).

2. **Slope of the points:** Assuming the points are scattered around a line, with little or no variance the actual slope does not matter (Fig 4.2 middle row). We see multiple positive slopes all having a correlation of \( r = 1 \), while the negative ones have \( r = -1 \).

3. **Linearity of the points:** On the bottom row (Fig 4.2) we notice the limitation of correlation as a similarity measurement: time series that have a non-linear relationship between themselves would result in a correlation \( r = 0 \).
Fig 4.3 shows the Anscombe’s quartet [123]: four scatterplots of eight time series (paired two by two). Each pair of time series has an \( r = 0.816 \) and the time series have a mean \( \bar{X} = 7.501 \) and standard deviation of \( \sigma_X = 2.030 \).

In Example 1, the scatterplot follows what looks like a normal distribution around the mean (regression line through the points). Example 2 has the same correlation although it does not indicate a linear relationship.

Example 3 show an almost perfectly correlated time series except for the outlier which changes \( r \) to 0.816. Again the influence of the outlier is changing \( r \) significantly in Example 4: without the outlier, the correlation would be undefined as one time series is constant. If we computed the linear models for these datasets, all would result in the same \( (y = 3 + 0.5x) \) having the same coefficients, p-values and intercepts. To mitigate these problems, it is important to test for normality assumptions and plot graphs before deciding to use Pearson Correlation. For example, performing a Shapiro-Wilk [124] test for normality on the Anscombe’s quartet would yield four time series that do not meet this requirement.

There are other, correlation measures such as Spearman and Kendal that are less frequently used [118] and more appropriate with nonlinear monotonic relationships between time series that do not follow normal distributions.
Spearman correlation is the rank-based version of the Pearson correlation, its range is defined on the interval [-1, 1] and it can be used to discover monotonic non-linear relationships such as the ones plotted on the bottom row of Fig 4.2.

Kendall’s correlation coefficient is designed to capture the association between two ordinal (not necessarily interval) variables [118]. It too has a range of [-1, 1] where a higher number represents a higher positive relationship.

As our time series follow normal distributions (as shown with Shapiro Wilk tests) and have linear relationships between themselves, we consider Pearson Correlation further.

Another aspect we should look for is the stationarity of the time series [153]. A time series $X$ that is not-stationary has $\bar{X}$ and $\sigma_X$ that change over time and, as a rule, are unpredictable and cannot be modelled or forecast. Time series that are not stationary have trends, cycles, random walks or a combination of both [153].

Fig 4.4 Two uncorrelated time series having an upward trend

Fig 4.5 The same time series with the trend removed.

In Fig 4.4 we have an example of two time series that are not stationary, presenting an upward trend. At first sight, we might conclude that the two signals are positively correlated and computing the correlation would yield a value close to 1. Looking more closely we would notice that it is quite the opposite: when one time series grows, the other falls.
By “differencing” a time series $X$ we remove the trend and stabilize the mean by subtracting each point from the previous one as follows:

$$X_t = X_t - X_{t-1}$$  \hspace{1cm} (4.6)

In Fig 4.5 we applied equation (4.6) on the two original time series using equation (4.6) which reveals the time series to be perfectly anti-correlated, in fact, their correlation yields $r = -1$. In contrast, we might have time series that look negatively correlated (Fig 4.6) because of the trends. Removing the trend by applying (4.6) we obtain Fig. 4.7 and a real correlation equal to 1.

Moreover, time series having different standard deviations $\sigma_X$ over different time intervals can be made stationary by taking the logarithm at each point [6] (together with (4.6) if needed).

In Fig 4.8 we have a real world example of daily stock prices for Aberdeen Asset Management, (ticker symbol ADN.L) showing a downward trend over 261 days (1/1/2015-31/12/2015). By applying (4.6) on it we obtain Fig 4.9, now a stationary time series, ready for further processing.
Fig 4.8 Aberdeen Asset Management (AND.L) over 250 days with linear regression in red

Fig 4.9 Aberdeen Asset Management (AND.L) with trend removed
The Augmented Dickey-Fuller (ADF) [125][126] checks whether the time series have unit root (a property which discriminates stationary time series from non-stationary) and computes a negative statistic: the lower the value, the more sure we are that the time series is stationary (Fig 4.10). We can perform an (ADF) test which results in a p-value of 0.01<0.05, rejecting the hypothesis of non-stationarity of AND.L (Fig 4.10).

\[
\text{data: } \text{diff(AND.L)} \\
\text{Dickey-Fuller = -5.96, Lag order = 6, p-value = 0.01} \\
\text{alternative hypothesis: stationary}
\]

Fig 4.10 “Augmented Dickey-Fuller Test” on AND.L showing stationarity

4.4 FCM and FCM++ on synthetic time series data

In this section we present a methodology to reduce the dimensionality of a larger set of synthetic time series, which allows us to better analyze, view and cluster the data. On this lower dimensional data space we compare the performance of FCM versus FCM++ in terms of average number of iterations as well as quality solution (using Xie Beni validity index [109]).

By using R, we generate 1000 time series: 10 clusters of 100 time series each, with the following properties:

- Each time series has 100 observations;
- Each cluster is a set of time series that have ~0.8 correlation coefficient between themselves;
- Each time series in one cluster has a different correlation coefficient when computed against time series in other clusters;
- Time series do not have trends.

As we know how these time series are distributed, we can for illustration plot two pairs of time series belonging to two clusters: Cluster 1 (Fig 4.11) and Cluster 10 (Fig 4.12).
Fig 4.11. Two correlated time series in Cluster 1

Fig 4.12. Two correlated time series in Cluster 10

Fig 4.13. A sample of 10 out of 1000 synthetically generated time series
Looking at the whole dataset in Fig 4.13, one may find it difficult to identify which series belongs to which cluster as no relevant patterns are discernible. Running an FCM clustering algorithm directly on this dataset will generate poor clustering solutions due to the problem of using Euclidean Distance on high dimensional spaces [120].

Multi-Dimensional Scaling (MDS) is an exploratory data analysis technique “that represents measurements of similarity (or dissimilarity) among pairs of objects as distances between points of a low dimensional multi-dimensional space” [127]. MDS expects a matrix of distances between data points in the higher dimensional space, distances which can be both metric and non-metric. In this case study we are interested in the distance between time series in terms of correlation and we use it as \(d = 1 - r\) with a range of \([0,2]\). This newly defined distance \(d\) meets the metric requirements (such as being positive, symmetric, etc.) and makes the results easily interpretable: if two time series are very similar in behavior they will tend to have a low value of \(d\).

Fig 4.14 MDS scaled representation of synthetic time series dataset
In Fig 4.15 we show a comparison between FCM and FCM++ in terms of the number of iterations needed to converge over ten independent runs. It is important to note that, because of the clear separability of the data in the transformed space (Fig 4.14), a high value for the FCM p parameter achieves better results. As in the case of standard datasets, a quick visual inspection of the data is important in order to choose a suitable parameter value. The standard version of the algorithm takes an average of 22.6 iterations for the algorithm to converge, for FCM with p=1.8 we need an average of 12 and when p = 5 we need an average of 6.3 which is a significant improvement.

Fig. 4.16 shows the outcome of centers initialization with the standard approach (in red), FCM++ with p=1.8 in blue and p=5 in magenta. We observe that picking the points at random would result in a higher number of iterations for the algorithm to converge: the centers would need a greater number of steps to reposition at the centers of the ten clusters. Picking a smaller p value for FCM++ disperses the centers but, while better than the standard, the results are inferior to using a high p value (p=5).
Fig 4.16 Number of iterations FCM (in red) vs FCM++ (with p=1.8 in blue and p=5 in magenta)

Fig 4.17 Xie-Beni validity index: FCM in red, FCM++ (p=5) in blue on the lower representation of the synthetic time series dataset
The Xie-Beni index measures the separation as well as the compactness of the resulting clusters. In Fig 4.17 we observe a comparison between the standard algorithm in red and our proposed methodology initialized with a parameter p=5. With the exception of one independent run out of 10 (run 9), FCM++ with p=5 achieves a superior cluster quality solution in terms of Xie-Beni. Because there is a degree of randomness in all the initialization procedures (including our own) there is a minimal chance of not delivering the exact results every time (e.g. run 9, Fig 4.17). FCM++ (with p=5) on this lower dimensional MDS space almost always (9 out of 10 runs) results in perfect clustered time series as we defined them in the creation process.

4.5 FCM vs FCM++ on financial time series data

Fig 4.18 FTSE companies stock prices over the period 1/1/2015-31/12/2015
Every company listed on the London Stock Exchange (LSE) is classified into an industry sector based on its primary activity, however, it may be both more interesting and valuable to group similarly performing companies based on their historical stock price record over a long period of time [119].

Unlike [119] where PCA was used to remove noise and extract relevant features before running the clustering procedures, in this section we use Multi-Dimensional-scaling (MDS) on the matrix of correlations between time series. By following this approach we integrate the correlation information in the dimensionality reduction algorithm making it easier to visualize and analyze the distance between time series. Using fuzzy clustering analysis on a MDS scaled time series dataset, we obtain a more insightful categorization of the companies into groups with fuzzy boundaries, giving arguably a more realistic and detailed view of their relationships. After a series of preprocessing steps, we visualize the time series in the lower dimensional space and perform cluster analysis using both FCM and FCM++ and compare results. This approach has the potential to be of practical relevance in the context of diversified portfolio construction as it can detect fuzzy clusters of correlated stocks (represented as time series) that have lower inter-cluster correlation, analyze their volatility and sample potentially less risky combination of assets.

The approach that we are using here is based on the following steps:

1. Data is downloaded using the Quantmod library [128]. Out of the 100 constituents of FTSE100 (Financial Times Stock Exchange 100 Index) [129], only 93 were available for download through the Quantmod framework.
2. Missing data is approximated using LOCF (last observation is carried forward)
3. Time series are made stationary by applying equation (4.6)
4. We compute the correlation distance \( r \) between every time series, resulting in a matrix of size 93 x 93
5. Using the Sharpe ratio [130][131] we compute the risk adjusted return for every stock.
6. MDS is performed on the distance matrix in order to reduce the dimensionality of the data, and to visualize and cluster the results
7. Cluster solutions are investigated and a comparison made between FCM and FCM++
8. We provide a method based on entropy for discovering hard to cluster stocks, which are stocks that have a fuzzier membership, being situated at the margin of discovered clusters.

Quantmod provides investors with a rapid prototyping environment which assists traders in building statistical models, testing and deploying statistical models. We use the library for gathering daily adjusted closing prices of companies that are part of the FTSE100 index and are traded on the LSE over one year period (1 January 2015-31 December 2015). As the historical price record for all 100 companies’ part of the index was not fully available over this timeframe, a total of 93 large companies have been investigated. In Table 4.1 the companies considered are identified by their stock symbol; the data that we use here are time series corresponding to adjusted close prices of 93 major companies listed at the LSE over the period 1/1/2015-31/12/2015.

![Figure 4.19](image)

Fig 4.19 Stock prices over the period 1/1/2015-31/12/2-15 with trend removed

As our scaling and clustering algorithms cannot handle missing data, imputation was is needed in 5 time series out of 93 and this missing data was mostly in the form of a few nonconsecutive days with missing prices. A fast, simple and widely useful method to correct this flaw is LOCF which carries previous data forward to impute the missing ones. Other methods worth mentioning here include linear approximations, spline interpolation, and so on [132].

In Fig 4.18 we notice that some of the time series are having trend present, which needs to be removed in order for correlation as a distance metric to work. For this we use differencing (4.6)
followed by ADF tests to ensure all the data are stationary. Fig 4.19 we see the time series after differencing, we note their zero mean and constant variance.

Table 4.2 presents a view of the dataset where the companies are grouped together based on a predefined industry. More details about the companies can be found in the Appendix. We note from Table 4.2 that the companies belong to a wide range of industries from water and utilities to banks and fashion. In order to understand why the classification based on industries (Table 4.2) alone is not always a good informer of stock behavior, let us consider the companies listed in the “Fund management” industry and focus on Schroders (SDR.L) and Standard Life (SL.L) in Fig 4.20.

![Fig 4.20 Differenced Schroder’s vs Standard Life over the year 2015](image)

We note from Fig 4.20 that the stocks have no obvious pattern in common and it would be wrong to assume that just because they are classified together in an industry, they share the same price movement. For this reason, it is insightful to build up a classification in a data-driven fashion, empirically using clustering algorithms rather than using arbitrary definitions. To estimate how aligned the behavior of the stocks is we will first compute the distance matrix using correlation distance. Correlation distance is computed on the differenced time series using the MKmisc [133] library. The end result is a symmetric matrix of size 93x93 representing the distance from each time series to the others.
<table>
<thead>
<tr>
<th>Sector</th>
<th>Tickers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fund management</td>
<td>ADN.L, SDR.L, SL.L</td>
</tr>
<tr>
<td>Insurance</td>
<td>ADM, AV.L, LGEN.L, OML.L, RSA.L</td>
</tr>
<tr>
<td>Generator hire</td>
<td>AGK.L</td>
</tr>
<tr>
<td>Consulting</td>
<td>AMFW.L</td>
</tr>
<tr>
<td>Mining</td>
<td>AAL, ANTO.L, BLT.L, EVR.L, FRES.L, GLEN.L, RRS.L, R</td>
</tr>
<tr>
<td>Engineering</td>
<td>ARM.L, BAB.L, IML.L, MGGT.L, MRO.L, SMIN.L, WEI</td>
</tr>
<tr>
<td>Food</td>
<td>ABF.L, CPG.L, TATE.L</td>
</tr>
<tr>
<td>Pharmaceuticals</td>
<td>AZN.L, GSK.L, SHP.L</td>
</tr>
<tr>
<td>Military</td>
<td>BA.L</td>
</tr>
<tr>
<td>Banking</td>
<td>BARC.L, HSBA.L, LLOY.L, RBS.L, STAN.L</td>
</tr>
<tr>
<td>Oil and gas</td>
<td>BG.L, BP, PFC.L, RDSA.L, TLW.L, WG.L</td>
</tr>
<tr>
<td>Tobacco</td>
<td>BTI, IMT</td>
</tr>
<tr>
<td>Property</td>
<td>BLND.L, HMSO.L, LAND.L</td>
</tr>
<tr>
<td>Media</td>
<td>BSY, ITV.L, WPP.L</td>
</tr>
<tr>
<td>Telecoms</td>
<td>BT-A.L, VOD.L</td>
</tr>
<tr>
<td>Industrial products</td>
<td>BNZL.L</td>
</tr>
<tr>
<td>Fashion</td>
<td>BRBY.L</td>
</tr>
<tr>
<td>Support Services</td>
<td>CPL.L</td>
</tr>
<tr>
<td>Leisure</td>
<td>CUK.L, TUI.L</td>
</tr>
<tr>
<td>Energy</td>
<td>CNA.L, SSE.L, NG.L</td>
</tr>
<tr>
<td>Consumer</td>
<td>CCH</td>
</tr>
<tr>
<td>Building materials</td>
<td>CRH.L, WOS.L</td>
</tr>
<tr>
<td>Chemicals</td>
<td>CRDA.L, JMAT.L</td>
</tr>
<tr>
<td>Beverages</td>
<td>DGE.L, SAB.L</td>
</tr>
<tr>
<td>Information</td>
<td>EXPN.L</td>
</tr>
<tr>
<td>Security</td>
<td>GFS</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>GKN.L, RR.L</td>
</tr>
<tr>
<td>Finance</td>
<td>HLL.L, PRU.L</td>
</tr>
<tr>
<td>Hotels</td>
<td>IHG.L</td>
</tr>
<tr>
<td>Transport air</td>
<td>IAG.L</td>
</tr>
<tr>
<td>Product testing</td>
<td>ITRK.L</td>
</tr>
<tr>
<td>Supermarket</td>
<td>SRBY.L, MRW.L, TSCO.L</td>
</tr>
<tr>
<td>Retail homeware</td>
<td>KGF.L</td>
</tr>
<tr>
<td>Retailer</td>
<td>MKS.L</td>
</tr>
<tr>
<td>Retail clothing</td>
<td>NXT.L</td>
</tr>
<tr>
<td>Education</td>
<td>PSON.L</td>
</tr>
<tr>
<td>Consumer goods</td>
<td>RB.L, ULVR.L</td>
</tr>
<tr>
<td>Publishing</td>
<td>REL.L</td>
</tr>
<tr>
<td>Packaging</td>
<td>REX.L</td>
</tr>
<tr>
<td>IT</td>
<td>SGE.L</td>
</tr>
<tr>
<td>Outsourced services</td>
<td>SRP.L</td>
</tr>
<tr>
<td>Water</td>
<td>SVT.L, UUL.L</td>
</tr>
</tbody>
</table>
As in the case of synthetic data, we perform an MDS procedure on the correlation distance matrix. This method projects the data into two dimensions such that the distances between the stocks are preserved. In Fig. 4.21 we present this projection and we use a different colour for each company stock to indicate its computed risk adjusted return (Sharpe ratio; the redder the better) in the original space. The ratio, also known as the Sharpe index, is a popular way of adjusting return to risk (volatility) is defined as follows [131]:

$$\text{Sharpe}(X) = \frac{\bar{X}_a - \bar{X}_r}{\sqrt{\sigma(X_a - X_r)}}$$  \hspace{1cm} (3.8)
where $X_a$ represents the asset return and $X_r$ the risk free rate. We have used the Sharpe Ratio function in Performance Analytics package [131]. Table 4.3 ranks the stocks in terms of their Sharpe Ratio.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Sharpe Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL.L</td>
<td>0.08591</td>
</tr>
<tr>
<td>REX.L</td>
<td>0.083724</td>
</tr>
<tr>
<td>SGE.L</td>
<td>0.078713</td>
</tr>
<tr>
<td>ITV.L</td>
<td>0.072915</td>
</tr>
<tr>
<td>RB.L</td>
<td>0.069419</td>
</tr>
<tr>
<td>CRH.L</td>
<td>0.058639</td>
</tr>
<tr>
<td>BT-A.L</td>
<td>0.053287</td>
</tr>
<tr>
<td>WPP.L</td>
<td>0.051528</td>
</tr>
<tr>
<td>IAG.L</td>
<td>0.050075</td>
</tr>
<tr>
<td>CRDA.L</td>
<td>0.046345</td>
</tr>
<tr>
<td>SAB.L</td>
<td>0.044684</td>
</tr>
<tr>
<td>ITRK.L</td>
<td>0.043421</td>
</tr>
<tr>
<td>CPI.L</td>
<td>0.042106</td>
</tr>
<tr>
<td>SVT.L</td>
<td>0.041186</td>
</tr>
<tr>
<td>ULVR.L</td>
<td>0.041089</td>
</tr>
<tr>
<td>LGEN.L</td>
<td>0.036928</td>
</tr>
<tr>
<td>REL.L</td>
<td>0.035837</td>
</tr>
<tr>
<td>EXPN.L</td>
<td>0.035625</td>
</tr>
<tr>
<td>TUI.L</td>
<td>0.0348</td>
</tr>
<tr>
<td>BNZL.L</td>
<td>0.033027</td>
</tr>
<tr>
<td>BA.L</td>
<td>0.031863</td>
</tr>
<tr>
<td>AV.L</td>
<td>0.031407</td>
</tr>
<tr>
<td>NXT.L</td>
<td>0.031065</td>
</tr>
<tr>
<td>SDR.L</td>
<td>0.03071</td>
</tr>
<tr>
<td>BTI</td>
<td>0.029985</td>
</tr>
<tr>
<td>CPG.L</td>
<td>0.028646</td>
</tr>
<tr>
<td>NG.L</td>
<td>0.023803</td>
</tr>
<tr>
<td>PFC.L</td>
<td>0.023566</td>
</tr>
<tr>
<td>BG.L</td>
<td>0.023391</td>
</tr>
<tr>
<td>SBY.L</td>
<td>0.018702</td>
</tr>
<tr>
<td>UU.L</td>
<td>0.01834</td>
</tr>
<tr>
<td>ABF.L</td>
<td>0.018282</td>
</tr>
<tr>
<td>GSK.L</td>
<td>0.015531</td>
</tr>
<tr>
<td>AZN.L</td>
<td>0.015083</td>
</tr>
<tr>
<td>BLND.L</td>
<td>0.01421</td>
</tr>
<tr>
<td>ARM.L</td>
<td>0.010498</td>
</tr>
<tr>
<td>MRO.L</td>
<td>0.009747</td>
</tr>
<tr>
<td>SHP.L</td>
<td>0.008179</td>
</tr>
<tr>
<td>HMSO.L</td>
<td>0.00745</td>
</tr>
<tr>
<td>WOS.L</td>
<td>-0.00062</td>
</tr>
<tr>
<td>KGF.L</td>
<td>-0.00124</td>
</tr>
<tr>
<td>LLOY.L</td>
<td>-0.00547</td>
</tr>
<tr>
<td>RSAS.L</td>
<td>-0.00143</td>
</tr>
<tr>
<td>OML.L</td>
<td>-0.00438</td>
</tr>
<tr>
<td>BAB.L</td>
<td>-0.00453</td>
</tr>
<tr>
<td>CUK.L</td>
<td>-0.00519</td>
</tr>
<tr>
<td>LLOY.L</td>
<td>-0.00547</td>
</tr>
<tr>
<td>RBS.L</td>
<td>-0.0061</td>
</tr>
<tr>
<td>RR.L</td>
<td>-0.0070</td>
</tr>
<tr>
<td>RIO.L</td>
<td>-0.006024</td>
</tr>
<tr>
<td>STAN.L</td>
<td>-0.00143</td>
</tr>
<tr>
<td>RDSA</td>
<td>-0.061</td>
</tr>
<tr>
<td>BRBY.L</td>
<td>-0.064</td>
</tr>
<tr>
<td>RBS.L</td>
<td>-0.065</td>
</tr>
<tr>
<td>RR.L</td>
<td>-0.069</td>
</tr>
<tr>
<td>RIO.L</td>
<td>-0.070</td>
</tr>
<tr>
<td>SGE.L</td>
<td>0.078713</td>
</tr>
<tr>
<td>MRO.L</td>
<td>0.009747</td>
</tr>
<tr>
<td>SHP.L</td>
<td>0.008179</td>
</tr>
<tr>
<td>HMSO.L</td>
<td>0.00745</td>
</tr>
<tr>
<td>WOS.L</td>
<td>-0.00062</td>
</tr>
<tr>
<td>KGF.L</td>
<td>-0.00124</td>
</tr>
<tr>
<td>LLOY.L</td>
<td>-0.00547</td>
</tr>
<tr>
<td>RSAS.L</td>
<td>-0.00143</td>
</tr>
<tr>
<td>OML.L</td>
<td>-0.00438</td>
</tr>
<tr>
<td>BAB.L</td>
<td>-0.00453</td>
</tr>
<tr>
<td>CUK.L</td>
<td>-0.00519</td>
</tr>
<tr>
<td>LLOY.L</td>
<td>-0.00547</td>
</tr>
<tr>
<td>RBS.L</td>
<td>-0.0061</td>
</tr>
<tr>
<td>RR.L</td>
<td>-0.0070</td>
</tr>
<tr>
<td>RIO.L</td>
<td>-0.006024</td>
</tr>
<tr>
<td>STAN.L</td>
<td>-0.00143</td>
</tr>
<tr>
<td>RDSA</td>
<td>-0.061</td>
</tr>
<tr>
<td>BRBY.L</td>
<td>-0.064</td>
</tr>
<tr>
<td>RBS.L</td>
<td>-0.065</td>
</tr>
<tr>
<td>RR.L</td>
<td>-0.069</td>
</tr>
<tr>
<td>RIO.L</td>
<td>-0.070</td>
</tr>
</tbody>
</table>

The ranking in Table 4.3 already provides investors with a view of the price evolution in terms of both returns and volatility and so we discard some of the worst performing stocks such as TLW, WEIR, AGK, GLEN. We note from Fig 4.20 that previously hidden patterns start to emerge: companies situated on the center-right hand side of Fig 4.20 tend to have a higher Sharpe Ratio.
Insurance companies such as AV.L (Aviva), LGEN.L (Legal & General) are correlated with financial companies such as PRU.L (Prudential) and SDR.L (Schroders) and have a high Sharpe Ratio indicating an opportunity for good investments. We notice some interesting exceptions such as REX.L (Packaging) and PFC.L (Oil and gas) that are farther (less correlated) from the high yielding cluster yet have high Sharpe ratios.

![Graph showing worst and best performing stocks](image)

**Fig 4.22** Worst (GLEN.L in red) versus best (HL.L in blue) performing stocks in terms of Sharpe Ratio

We observe in Fig. 4.22 the worst performing stock, GLEN.L (Glencore, a mining company) in red compared with the best performing one, HL.L (Hargreaves Lansdown, financial company) in blue. Glencore stock started the year with a stock price of 274.9 and ended the year at a value of 90.4 pounds, while Hargreaves started at a price of 987.6 and ended at 1496.7. Mining companies such as ANTO.L (Antofagasta) and RIO.L (Rio Tinto) situated on the left of Fig 4.21 had a bad year in terms of returns and volatility both having negative Sharpe Ratios (Table 4.3).

A shortcoming of the clustering algorithms employed is the need for proper parameter setting representing the number of clusters in the data. Unlike the generated synthetic data where we knew the number of clusters, here we need to infer the value of this parameter from the available
data. We could run the algorithm multiple times with different k parameter values (representing the number of clusters) and validate it using the Xie-Beni validation index. For each k parameter value we run the two algorithms 10 times and average the results in order to mitigate the random initialization used in both algorithms (FCM++ picks the first point at random). We observe in Fig 4.23 that the lowest Xie Beni index value is achieved for a value of k=12 suggesting that the number of clusters is 12. Clustering 93 companies into k=12 clusters results in an average of 7.75 stocks per cluster, while the predefined industry classification has an average of 2.2 stocks per industry. It is important to note that some of the stocks may be hard to assign to a cluster hence we present a way to detect these outliers based on the notion of entropy.

Fig 4.23 FCM (red) vs FCM++(blue, p=1.8) Xie Beni Validity index for different k values
Fig. 4.24 FCM (red) vs FCM++ (blue with p=1.8) Partition Entropy for different K values

Fig. 4.25 FCM (red) vs FCM++ (blue with p=1.8) for k=12 over 10 independent runs
The entropy of a random variable is defined as [107]:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \ast \log(p(x_i))$$

(4.9)

$$0 \leq H(X) \leq \log(n)$$

(4.10)

and it represents a measure of uncertainty associated with the X variable. The algorithms produce, as well as a set of centers, a membership matrix as an output where on each row we have the clustered objects (here stocks) and each column represents the stocks belonging to a particular cluster out of k=12 clusters. Each row in this matrix sums to 1 and each stock entry in this matrix can be viewed as a probability distribution and thus one can compute its entropy. The less certain the algorithm is about the stock, the fuzzier will be this distribution. For example, let us consider two clusters and a stock Y having a membership matrix entry (0.5, 0.5). A hard K-means algorithm, unlike FCM, would assign this stock randomly to one of the clusters and we may wrongly assume that it behaves very similar to others in that cluster. By providing this more granular view, FCM informs about the true behavior of this stock. If we were to compute the entropy of Y it will have a maximum (the same entropy as a fair coin), \(\log(2)=0.69\). We can compute the total entropy over all stocks using the membership matrix \((U)\) alone:

$$H(U; k) = \frac{\sum_{i=1}^{n} H(u_i)}{N}$$

(4.11)

This entropy-based validity index [112], in contrast to Xie-Beni, does not take into account the data or the centers and looks only at the fuzziness of the membership matrix. Fuzzier clusters provide less useful information about the behavior of the stocks, and thus a low partition entropy representing harder clusters is deemed more appropriate. It should be noted at this point that the \(m\) parameter of the FCM algorithm impacts the fuzziness of the results and in the extreme case of \(m=1\), FCM becomes KM. Throughout our experiments we have used the standard \(m=2\) value. In Fig 4.24 we notice that FCM++ with \(p=1.8\) achieves lower partition entropy over the whole range of Ks and thus harder clusters and better solutions overall. Although this informs about the quality of the solution, partition entropy, unlike Xie Beni, does not provide us with a good way of deciding the value of k-parameter: the more centers, the higher the chance of partitions being fuzzier. Essentially it lacks a way of penalizing for high values of \(k\).
In Fig 4.26 we observe the cost function value at convergence for FCM as compared to FCM++ with $p=1.8$. Seven out of ten runs of FCM++ (in blue) outperform the standard. The number of iterations until convergence was smaller for FCM++ vs FCM seven times out of ten runs (Fig 4.25). On the ninth iteration it required over 3 times less iterations to converge.

Finally, we run FCM++ with $k=12$ and $p=1.8$ with the outcome shown in Fig 4.27. The solution we present in Fig 4.27 is a less conservative one, where the stocks were assigned to the clusters for which they had the highest membership. It is important to note that some stocks may be situated at the edge of these clusters, thus they have a fuzzier membership and may be potential outliers.

To identify these outlier points we propose the use of entropy and quantiles in a two-step process:

1. Entropy is computed for each stock using its entry in the final membership matrix. As $k=12$ the maximum possible entropy is $\log(12) = 2.48$ which occurs when a stock has a value of $1/12$ in each membership matrix column.

2. The entropies are sorted by their values and a cutout point is set at the .90 quantile. The points above this cutout are considered to be outliers.
Fig 4.27 Multi-dimensional scaled time series clustered into k=12 clusters

Investigating entropy values for different stocks as in Fig 4.28 is useful in understanding their behavior. For example, we see that about half of the data has a high entropy of 1.5 or above. We notice that the distribution is not Gaussian by visualizing a density plot (Fig 4.29) and performing a Shaphiro-Wilk test (which rejects the NULL hypothesis of normality with a p-value=1.463e-05). This means we cannot perform outlier identification tests that assume a normal distribution and we use quantiles to identify 10% of the highest entropy stocks.
Fig 4.28 Stock entropy sorted by value
Fig 4.29 Entropy distribution for synthetic (red) and financial (blue)

If we visually compare the MDS scaled synthetic and real world data sets we note patterns emerging. First, we note the nice separability of clusters in Fig 4.14 as compared with Fig 4.21 suggesting real world financial time series form less compact clusters that are harder to identify. Second, we note the entropy distribution (Fig 4.29) of FCM solutions on the two datasets are different. The entropy of time series in the synthetic dataset has a Gaussian distribution confirmed by a Shapiro-Wilk test (resulting in a p-value of 0.00166).

As we note, most of the time series in the synthetic data set has a small entropy of about 0.3 (maximum possible being log(10)=2.3) which means the algorithm has no problem in figuring which time series belongs to which cluster. We also note that the FCM++ by far outperformed the standard FCM algorithm especially when we used a higher $p$ parameter value resulting in centers being picked far apart in the synthetic set. If we compare Fig 4.15 (synthetic) vs Fig 4.25 (real world) we see that the more separated and compact the clusters are the more FCM++ (especially with higher $p$ parameter values) outperforms the standard. A hypothesis that remains to be tested is whether entropy values could be used to determine the best $p$-parameter for spreading the centers at the initialization phase. These entropies can be computed for a FCM solution on a small sample of the whole data. If we find that they are spread over a larger area (as in Fig 4.29 with real world time series entropies in blue) FCM++ with a small $p$ parameter (less spread of the centers) might be appropriate. If the entropies are bunched together with a low mean (relative to the maximum
log (K) entropy such as in Fig 4.29 synthetic time series, with entropies in red) a FCM++ with a larger parameter might be more appropriate.

Fig 4.30 Outliers shown in red

The outliers identified are the top ten stocks in Fig 4.28 representing the upper 10% quantile: AGK.L, BLT.L, BTI, BNZL.L, CRDA.L, IMI.L, ITRK.L, RR.L, RBS.L, and SMIN.L.

If we look at Figs 4.27 and 4.30 we note that the time series in red are separated from the rest (being at the edge of their clusters) and the FCM algorithm assigned them a very high fuzzy membership (and thus high entropy). An investor could exploit this as an opportunity for diversification as these outliers behave differently to the rest (having a high correlation distance from the rest). Their Sharpe Ratio (Table 4.3) knowledge should also be considered.
Fig 4.31 Evolution of cluster returns when £1 is invested in each cluster

In Fig 4.31 we see the evolution of returns for a theoretical investment of £1 in each of the twelve clusters. For computing returns for a cluster we considered only the stocks that have a higher than $\alpha=0.5$ membership degree in that cluster (presented in Table 4.4).

<table>
<thead>
<tr>
<th>Cluster number</th>
<th>Stocks having &gt;0.5 membership to cluster</th>
<th>Return on £1</th>
<th>Sharpe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (black)</td>
<td>BAB.L, CNA.L, SBRY.L, LLOY.L,</td>
<td>£0.885</td>
<td>-0.027</td>
</tr>
<tr>
<td>2 (red)</td>
<td>FRES.L, RRS.L, SRP.L</td>
<td>£0.849</td>
<td>-0.026</td>
</tr>
<tr>
<td>3 (green)</td>
<td>IHG.L, ITV.L, MKS.L, NXT.L, SGE.L,</td>
<td>£1.093</td>
<td>0.037</td>
</tr>
<tr>
<td>4 (dark blue)</td>
<td>AV.L, CUK.L, GSK.L, LGEN.L, OML.L,</td>
<td>£1.037</td>
<td>0.017</td>
</tr>
<tr>
<td>5 (light blue)</td>
<td>AMFW.L, BP, PFC.L, WG.L</td>
<td>£0.844</td>
<td>-0.022</td>
</tr>
<tr>
<td>6 (magenta)</td>
<td>ANTO.L, GLEN.L, RIO.L, STAN.L</td>
<td>£0.507</td>
<td>-0.096</td>
</tr>
<tr>
<td>7 (yellow)</td>
<td>ADM, MRO.L, SL.L</td>
<td>£0.840</td>
<td>-0.043</td>
</tr>
<tr>
<td>8 (gray)</td>
<td>BSY</td>
<td>£0.369</td>
<td>-0.006</td>
</tr>
<tr>
<td>9 (dark gold)</td>
<td>AZN.L, BLND.L, CPG.L, HMSO.L,</td>
<td>£1.065</td>
<td>0.028</td>
</tr>
<tr>
<td>10 (deep pink)</td>
<td>AND.L, BRBY.L, CRH.L, GKN.L, HL.L</td>
<td>£0.985</td>
<td>0.002</td>
</tr>
<tr>
<td>11 (dark orange)</td>
<td>AAL, MGGT.L, PSON.L, SAB.L,</td>
<td>£0.858</td>
<td>-0.045</td>
</tr>
<tr>
<td>12 (light coral)</td>
<td>BA.L, BT-A, KGF.L, NG.L, SVT.L,</td>
<td>£1.074</td>
<td>0.032</td>
</tr>
</tbody>
</table>

Table 4.4 Returns on investment and Sharpe ratio for each cluster

Return at day $t$ for a cluster $C$ is computed as a weighted sum of individual stock returns as follows:
\[ R_t(C) = \sum_{i=1}^{93} (r_t * (U_{iC} / \text{sum}(U_{iC}))) \]  

(4.12)

where \( U_C \) is a column in the membership matrix \( U \) produced by FCM and \( r \) is a vector of returns for all the stocks at time \( t \). In this example, only weights in \( U_C \) that are larger than 0.5 are considered (representing the stocks in Table 4.3). Although the rows of the membership matrix \( U \) sum to one, their columns do not have to and there is no constraint on this in the standard FCM algorithm.

To mitigate this issue, we perform the transformation \( U_{iC} / \text{sum}(U_{iC}) \) in order to make the weights sum to 1.

After computing weighted returns for each cluster and each time \( t \), we can carry forward an initial hypothetical investment of £1 using (3.13) and observe how the investments would have performed (Fig 4.31)

\[ CR_t(C) = CR_{t-1}(C) + CR_{t-1}(C) * R_t(C) \]  

(4.13)

and for \( t=1 \), \( CR_t(C) = 1 \), representing the initial investment, \( CR \) standing for cumulative returns.

In terms of positive results, only clusters 3, 4, 9 and 12 are relevant and achieve positive cumulative returns at the end of the year. We notice that constituents of these highly achieving clusters are part of the central right points in (Fig. 3.21) identified (using colours) as having a higher Sharpe Ratios.

It is important to note that we have used an \( \alpha=0.5 \) membership degree on the FCM++ results in order to select time series in each of these clusters, but one can choose arbitrary values for \( \alpha \). In the extreme case of \( \alpha=0 \), each cluster has all the time series and there is no difference between the clusters. That happens because for FCM, each point belongs to every cluster by a membership degree greater or equal to 0. In the other extreme, \( \alpha=1 \), at most one stock from each cluster will be picked. In our example we could not have stocks with a membership of one in any of the clusters. Essentially an investor needs to decide how conservative he wants to be: choosing a higher \( \alpha \) results in less time series in each cluster while a lower \( \alpha \) results in clusters having many time series.

4. 6 Summary

We have proposed a methodology for clustering time series based on FCM++. In general, time series have particularities, such a non-stationarity, missing data, etc., which requires a more
complex approach when trying to perform data analytics on them. The primary contribution of this chapter is a methodology based on FCM clustering that addresses these challenges and provides the user with a clearer picture of time series patterns.

The first dataset we discussed was synthetic and it contained a larger amount of time series distributed into ten clusters. We reduced the number of dimensions while preserving the correlation structure and then analyzed the results of running FCM and FCM++ on this dataset. The results show that in general FCM++ outperforms FCM in terms of the required number of iterations for the algorithm to converge and also in terms of quality of the solutions. Moreover, we noticed that the more the cloud of points are separated, the higher the better are the FCM++ results if we choose a higher p parameter value.

The second case involved a real world data set of time series representing one year of stock price evolution for FTSE index constituents. By including the correlation information (through correlation distance matrix) in the dimensionality reduction algorithm (MDS) we were able to see how similar their behavior was in two dimensions. Moreover we have compared the results of clustering using FCM and FCM++ on this lower dimensional data: we found a general improvement in partition entropy, Xie Beni validity index and the number of iterations required for convergence when FCM++ was used. We have provided a way to select the number of stocks in each cluster based on their membership degrees and shown a way to identify outliers. The FCM++ methodology proposed for clustering stock time series may offer investors a better way visualizing and picking uncorrelated assets when constructing or managing a portfolio.
5. Text mining case study

5.1 Introduction

In a second case study we have looked at clustering of standard data sets as well as time series using Fuzzy C-means (FCM) utilizing our proposed method of center initialization, showing significantly improved results. In this chapter we consider another type of data coming in textual form. We have developed a web crawler that gathers news article from The Guardian [137] and extracts news articles on four topics (football, immigration, environment and money), performs a series of pre-processing steps, visualizes the data and clusters the news into four categories using a modified version of FCM entitled Spherical Fuzzy C-means (SFCM), available in [138] as an R package. We compare the behaviour of SFCM using our initialization method against the standard SFCM. By using SFCM++ we show that we can make an important impact in terms of the number of iterations needed for the algorithm to converge and the quality of clusters in terms of partition entropy.

5.2 Motivation

Text mining is the application of machine learning and statistical methods to textual data, methods which are able to cope with a wide range of words and structures in natural language with the goal of finding useful patterns. It is estimated that around 85% of business data comes in the form of text [139] and thus it would be very useful to be able to perform analytics and gain valuable business insights.

Traditionally, unsupervised learning methods for textual data have been categorized into topic models and hard clustering algorithms [140]. Hard Clustering of documents has the aim of partitioning the set of documents into non-disjoint subsets that are closely related in terms of the topic discussed. A challenge of hard clustering is to assign documents to single category, given the possibility that a document may contain paragraphs on several topics. Topic models capture topics by using probability distributions. For example, the topic of “football” may be described as a probability distribution that assigns high probabilities to words such as “ball”, “goal” and extremely low probabilities to words such as “biotech”. Topic model algorithms assign membership degrees to topics for each given document. In this sense, they share many similarities with soft (fuzzy) clustering procedures such as FCM.
Soft clustering of documents has many potential uses. We can cluster academic papers based on the topics discussed, website data in order to find similar content, news articles and many more. In this chapter we show a way to gather news articles at scale and perform soft clustering using a modified version of FCM entitled SFCM. Instead of using Euclidean distance, SFCM uses cosine distance and we show why it is a more appropriate measure for textual data. By using the seeding mechanism introduced in this thesis on a low dimensional representation of the text data, we can improve the performance of this algorithm. By spreading the points in the input space SFCM requires fewer iterations to converge to the local minima of its cost function than the standard approach and results in better solutions in terms of partition entropy.

The chapter is structured as follows: in Section 5.3 we introduce the multi-step methodology for visualizing and clustering text data; for gathering news from the Guardian web page [137], a Python [141] crawler based on Scrapy [142] library is discussed in Section 5.4; in Section 5.5 we discuss a series of pre-processing steps to transform the crawled unstructured data into one that is ready for visualization and clustering; in Section 5.6 we provide the intuition behind cosine distance for text documents, show why normalization is useful and contrast the SFCM algorithm with the one initialized with our algorithm SFCM++ in terms of iteration counts until cost function convergence and validity indexes; Section 5.7 concludes this chapter with a summary of key findings.

5.3 The Methodology

Our methodology for news article clustering is composed of five main steps (Fig 5.1): data crawling, data preprocessing, multi-dimensional scaling, 3d visualization and spherical FCM clustering.
Fig 5.1 Text Mining Methodology

1. Crawl Data

   Pre-process:
   1. Strip white space
   2. Convert to lowercase
   3. Remove punctuation
   4. Remove numbers
   5. Remove stop words
   6. Stem words
   7. Compute TF-IDF Matrix
   8. Remove sparse terms

   Multi-Dimensional Scaling and unit normalization:
   1. Compute Distance Matrix
   2. Build MDS model
   3. Normalize vectors to unit length

   Spherical Fuzzy C-means Clustering
   1. Compute SFCM++
   2. Compute SFCM
   3. Compare Iteration Counts
   4. Compare Validity indexes

   3D Visualization
5.4 Data crawling

The Guardian [137] (Fig 5.2) is a popular newspaper from the UK that covers a wide range of topics. A web crawler is a robot which visits websites with the purpose of downloading content in an automatic manner. We have chosen to develop a web crawler in Python [141] with the use of the Scrapy library [142] which is an open source web crawling platform for Python. Scrapy makes crawling easy by providing a framework that utilizes regular expressions to extract relevant data from crawled web pages.

Fig 5.2 The Guardian website

Fig 5.2 shows an example of an immigration news listing page visited by the web crawler. It contains a list of headlines and images, each accompanied by a link to a page that contains the full article. The crawler visits multiple such listing pages on the topics of “football”, “immigration”, “environment” and “money” and saves the links to the actual news articles. Next, it visits each individual news web page and crawls its text section which contains the whole story. We crawled 400 news articles on the four topics mentioned but due to broken links we crawled a total of 368. The crawled news text differs greatly in size and may contain inserted images. For example, in Fig
5.3 we see a football news paragraph beginning with the word “It” where “I” was an actual image inserted in text, the image was not saved by the crawler.

It was incredible to work with Zlatan Ibrahimovic and José Mourinho. They are two winners with great personalities, two real football people. The Manchester United fans can really look forward to having two world-class people at the team, one on the bench and one on the pitch. The two of them can turn Manchester United into winners again.

Fig 5.3 Image in text

5.5 Text clustering

We use the tm package [143] in R for the purpose of data preprocessing. In order to transform this unstructured text data into a structured format that we can further analyze, we perform a series of pre-processing steps as follows:

1. **Stripping white space**: we remove all blank spaces in the document, as they do not carry any informational content;
2. **Converting words to lowercase**: we convert all words to lower case in order to compute term counts, thus words such as “football” and “FoOtball” become equivalent;
3. **Removing punctuation**: punctuation signs do not carry any information so we discard them;
4. **Removing numbers**: we remove numbers as they are very specific in nature when they occur inside paragraphs and do not help our clustering process;
5. **Removing stop words**: we remove conjunctions and prepositions (words such as “the”, “or”) that carry no useful information for clustering;
6. **Stemming words**: for each word in each document we build the basic form of the word, one where endings such as “s” for plurals or “ing” for verbs and other affixes are removed [143].
7. **Compute TF-IDF Matrix**: Each document in our set is converted into a record in a TF (term frequency) - IDF (inverse document frequency) matrix. This matrix is **very large**, containing 368 rows (representing the news we gathered) and 15808 columns (representing all the terms in all the documents crawled). A value in this matrix
represents the term frequency for a term in a particular document, weighted using the IDF scheme. The IDF scheme assigns lower weights for words that are often repeated in English language while assigning higher weights to uncommon words such as “psycholinguistics”;

8. **Remove sparse terms:** After filtering and removing the stop words, numbers, etc. the TF-IDF matrix still contains a large number of terms. We further remove the terms that occur in very few documents. Terms that are at least 99% sparse are removed and we finally obtain a TF-IDF matrix of size 368x3646. Removing even more terms results in less separated data sets in the Multi-dimensional scaling (MDS) space discussed in the later sub-chapters;

Fig 5.4 Top terms in our news data set

Fig 5.4 shows the top most terms, occurring more than 300 times, in the final TF-IDF matrix. We notice the presence of terms that are relevant to each of the four topics. “Football”, “leagu”, “player” are stemmed words occurring in our football news; “migrant”, “asylum” are example of terms in immigrations news’ “energy”, “nuclear” are examples of terms frequent in environment news; and “bank”, “pension” are terms frequent in news about money. Some words such as “benefit” might be present in multiple news categories such as money and immigration news. A peculiar word occurring more than 300 times is the word “Allardyce”, the surname of a football manager mentioned very frequently in the football news. Fig 5.5 shows the terms that are correlated with the name “Allardyce”.

```
account allardyc also around asylum back bank benefit brexit britain british business campaign can car charg charg children clim climat club come community compani cost countri cup custom cut day deal dont doum hous immigr import includ incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incre incr
With the TF-IDF matrix we compute the distance between each pair of documents using the cosine distance. This resulting matrix of size 368x368 is further used in the MDS scaling to reduce the number of dimensions for further visualization and clustering. To understand why cosine distance is more appropriate than Euclidean when dealing with text data, let us consider three simple documents (D1, D2 and D3) and two terms: “London” and “Manchester” that repeat in different proportions throughout the documents. We cannot form real sentences using only these two terms but for the purpose of illustration we keep the number of terms low so that we can visualize how these vectors would look like in two dimensions. The documents are as follows:

- Document 1: London London
- Document 2: London London Manchester Manchester

Looking at Fig 5.6 we observe that in terms of Euclidean distance, D1 and D2 are closer in space than D2 and D3. In fact, Euclidean distance between D1 and D2 is equal to 2, while the Euclidean distance between D2 and D3 is equal to 5.65.

Another thing we notice is that D3 is proportional in term count to document D2: it has 3 times more mentions of London and 3 times more mentions of Manchester. We can think of D3 as D2 having been copy-pasted three times. Coming back to our news data, it could be perhaps that one news article is just a few paragraphs long while another on the same topic is ten times longer. Document D2 and D3 have more in common (they share the two same two terms) than D2 and D1 (which share only one term), yet Euclidean distance misleads us into believing something else.
Instead of capturing the magnitude (count of words) in our distance function, it is more relevant to capture the angle formed by the document vectors. We notice in Fig 5.6 that the angle between D2 and D3 is equal to zero while the one between D2 and D1 is around 45 degrees. There are two ways in which we can address this problem:

1. By normalizing the data (5.1) and then computing Euclidean distance on this new space (Fig 5.7);
2. By using cosine distance (5.3) [144] on the data.

Fig 5.6 Three documents represented in two dimensional space as red dots. The axes capture the count of either London (X Axis) or Manchester (Y Axis) in the documents
Fig 5.7 Three normalized documents represented in two dimensional space as red dots. The axes capture the count of either London (X Axis) or Manchester (Y Axis) in the documents.

We normalize the vectors to unit length by using the following formula:

\[ \text{Norm}(x) = \frac{x}{\|x\|} \quad (5.1) \]

where \( \|x\| \) is the norm (length) of the vector, defined as \( \|x\| = \sqrt{\sum_{i=1,N} x_i^2} \) \quad (5.2)

\[ \text{CosDist}(x, y) = 1 - \frac{x \cdot y}{\|x\| \|y\|} \quad (5.3) \]

where \( x \cdot y \) is the inner product.

In Fig 5.7 we notice that after transforming the vectors D2 and D3 to unit lengths they become closer (overlapping) in the new space. Similarly, our news dataset in Fig 5.8 become much more separated in the new normalized space formed of unit length vectors in Fig 5.9.
Fig 5.8 Un-normalized MDS scaled Guardian news articles represented in three dimensional space. Points in black represent Football news, those in red represent news on immigration, those in green represent news on environment and those in blue represent news on money.

Normalizing vectors to unit lengths puts them on a circle in two dimensions, a sphere in three dimensions and a hypersphere in N dimensions. We would be happy at this point to run and compare our FCM++ to FCM with Euclidean distance on this newly formed, spherical, normalized data. To deal with spherical data, however, the clustering algorithm needs to know that the data lies on a sphere and compute clusters centers accordingly (they have to lie on this sphere as well, in the middle of their clusters).
Fig 5.9 Unit Sphere: normalized scaled Guardian news articles represented in three-dimensional space. Points in black represent Football news, red represents news on immigration, green represents news on environment and blue represents news on money.

SFCM [145] has been proposed to deal with this type of data. Besides using cosine distance in its computation of similarity between documents, it makes sure that at each iteration the new clusters centers are on the sphere, having unit length. Equation 5.4 shows the cost function minimized by SFCM [145].

\[
\text{Cost}(X, Y) = \mu_{i,j} \sum_{i,j} (1 - \cos(x_i, p_j)) = \mu_{i,j} \sum_{i,j} (\text{CosDist}(x_i, p_j))
\]

(5.4)
Fig 5.10 Using SFCM++ initialization with resulting centers shown in magenta

Fig 5.10 shows an example of center initialization using SFCM++ and a spreading factor of $p=3$ which we found the best way to initialize the centers. Fig 5.11 shows the number of iterations needed by the algorithm to converge with SFCM in red and SFCM++ in blue. On average, the SFCM++ method needs 12.2 iterations vs 16.8 needed by the one with random initialization. There is not much difference in terms of cost function at the end of convergence (Fig. 5.12) suggesting both the algorithms end with centers in a similar position.
Fig 5.11 Number of iterations SFCM (red) vs SFCM++ (blue)

Fig 5.12 Xie Beni SFCM (red) vs SFCM++ (blue)
Fig 5.13 shows the centers location at the convergence to the minima of the SFCM cost function (5.4). The Xie-Beni validity index tends to be lower (better) for SFCM++ in seven out of ten occasions (Fig 5.12). A lower Xie Beni suggests tighter clusters with more space in between the clusters. In addition, in seven out of ten occasions (Fig 5.14) the partition entropy validity index is lower for SFCM++. A lower Partition Entropy means that the membership matrix resulting from running SFCM++ has a lower degree of fuzziness.
5.6 Summary

In this chapter we have considered the effectiveness of FCM+ (by a variant, SFCM+) in dealing with textual data and provided a methodology for clustering news data. Through a series of pre-processing steps we have converted unstructured textual data into a matrix of documents and terms. We have further reduced the dimensions of this matrix and visualized the data in three dimensions. The results show nicely separated clusters on which our seeding initialization mechanism requires significantly fewer iterations (an average of 12.2 as opposed to 16.8) until the SFCM algorithm converges to the minima of the cost function. SFCM, an extension of FCM that uses cosine distance during the document distance computations, has been used as it is more appropriate on textual data. In terms of quality expressed by validity indexes, partition entropy and Xie-Beni was lower on SFCM++ compared to the standard SFCM.

Fig 5.14 Partition Entropy SFCM (red) vs SFCM++(blue)
6. Conclusion

6.1 Summary

Cluster Analysis, also known as unsupervised learning, is a domain of central importance in the field of Machine Learning (which is an integral part of Artificial Intelligence). The goal of cluster analysis, is that of separating a finite unlabeled dataset into a set of structures called clusters which contain objects having similar properties. To achieve this objective there have been numerous approaches proposed through literature: hierarchical (building a hierarchy of clusters using either a top-down or bottom-up approach), density-based (scanning for dense structures in the data), grid based (using grids to divide up the data space), methods based on co-occurrence of nominal data and so on.

One important class of clustering algorithms that has been used extensively is that of partitional methods. Also known as centroid based clustering these methods consider clusters as sets of similar objects closer to the center of a cluster rather than any other center of other clusters. This similarity aspect is subjective, the user having to choose a measure that captures his idea of closeness before commencing the clustering procedure. These partitional methods have found a variety of applications ranging from image segmentation, real time applications, signal analysis internet portals to software engineering and communication [83-89]. Partitional clustering algorithms can be further divided into two categories based on how the clusters are constructed. One approach, that of KM (and the related K-medoids, etc.) finds clusters that are disjoint, non-overlapping (also known as hard clusters): an object discovered using these methods can belong to one and only one cluster. On the other hand we have methods such as Fuzzy C-means (FCM), the main subject of this thesis, which are more general and can find either hard or soft (which are allowed to overlap) clusters through the tuning of one of its parameters. As we have seen, the underlying theory of fuzzy sets on which FCM is based has a more sound and realistic foundation: there are few objects in real life that can be categorized for sure into one and only category. The objects in fuzzy clustering are allowed to belong to several clusters at the same time with a membership function specifying the belonging degree into each.

One of the major limitation of the FCM algorithm (present in KM as well) is its reliance on random initialization of the centers that specify the partitioning of the data. These centers do not
start in the right place, in the middle of the clusters in the data, rather, each iteration of the FCM algorithm brings them closer to the correct position in the middle of these clusters. By moving, they minimize a cost function which captures a desirable property of the clustering solution: points in any given cluster should have a short distance to their cluster center. The minimum of this cost function is not guaranteed to be a global minimum, thus the achievement of ideal solution cannot be assumed beforehand. Moreover, the algorithm, besides computing new centers at each iteration, has to compute additionally a (possible large) matrix of membership degrees. One way to make this process faster is by finding good location for these centers right at the beginning before the iterative algorithm commences. From a good starting position, these centers have to travel less to their final position at the minimum of the cost function, which translates into a lower number of iterations and thus less membership matrix computations.

This thesis has investigated empirically the effectiveness of introducing a more general form of KM++ initialization scheme into the context of FCM. By careful choosing the cluster centers in a way which disperses the initial cluster centers through the data space, the resulting FCM++ approach samples starting cluster centers during the initialization phase. The cluster centers are well spread in the input space, resulting in both faster convergence times and higher quality solutions. Moreover, we allow the user to specify a parameter how far and apart the cluster centers should be picked in the dataspace right at the beginning of the clustering procedure.

After introducing the algorithm, we have presented case studies on artificial data sets that cover a wide range of possibilities and discussed how the performance is affected by data size, overlap and the choice of spreading factor parameter $p$ of the proposed algorithm. We have considered clusters of equal sizes with no overlap, clusters of equal sizes with some overlap, clusters of different sizes with no overlap and clusters of equal sizes with some overlap. The most impressive results were obtained on the medium overlap and equal sized clusters, where FCM++ was on average 2.1 times faster than the standard (40 times the standard for some particular $p$ parameter values).

We have also considered real world datasets where we have seen improvements over the standard ranging from dataset 1.12 on the SPAM dataset up to 1.44 on IRIS, when the $k$ parameter equals true number of clusters in the dataset. We also observe improvements in terms of iterations count on the whole range of $k$ values. On real world data sets, the algorithm produced encouraging results in terms of both the number of iterations needed to reach convergence and the final cluster
quality. On each dataset tested, the FCM++ algorithm achieved fewer iterations (particularly where \( k = \) the real number of clusters) with similar if not significantly better Xie–Beni values suggesting overall higher quality clusters.

We have presented a methodology for analyzing time series using FCM++. Time series are a special type of data where the feature values are collected at equal time intervals from a generating process such as the weather, the stock exchange or many other places. Unlike standard datasets (such as those considered in Chapter 3), clustering and analyzing time series requires additional preprocessing steps. We have shown why this is the case when we tried to capture correlations between time series. Preprocessing steps such as differencing (when consecutive data points are subtracted from one another) are necessary and has been used as an important step before dimensionality reduction in financial time series of stock prices. We have shown a way in which to use Multi-Dimensional Scaling with Pearson correlation distance to reduce the dimensions of the data while preserving the correlation information. FCM++ used on this reduced data sets and improves over the standard in terms of iterations and external validity indexes (suggesting better solutions). Further, we have investigated the clusters discovered in stock price data (representing highly correlated stocks) in terms of their profit and volatility.

Further, we have looked at an important clustering case study involving textual data. Unlike other types of data, textual written data is the carrier of people’s thoughts and feelings on various topics and presents special challenges. News articles on various topics were crawled, preprocessed, projected in a lower dimensional space, visualized and clustered using a textual clustering variant of FCM++ entitled Spherical C-means++ (SFCM++). The same spreading mechanism introduced for FCM++ was used in combination with SFCM resulting in a lower average number of iterations (until convergence to the minima of the cost function) and a higher qualitative solution (as expressed by Xie Beni index and Partition Entropy).

### 6.2 Future Work

Let us consider Figure 6.1 and suppose that our algorithm picks the first point at random from the dataset (marked in red). Our proposed algorithm builds a distribution based on the distances from this point to every other point, assigning higher probability of being picked to further away points. While constructing this distribution it might be highly valuable to investigate in more detail the distribution of these distances as they could provide more information that could be used to
enhance the quality of the centers picked. A fitted density plot of the distances from the point marked in red in Fig 6.1 to all other points is shown in Fig 6.2. The actual distances are shown on the X axis and number of points from the one marked in red having same distance to it shown on the Y axis. First thing that we notice is that the density plot contains three peaks which map on to the three clusters that we have. The first peak is made out of short distances from the point marked in red, representing the points close to it in the top right Cluster in Fig 6.1.

Since the cloud of points follows a Gaussian distribution there are more points towards the center of the cluster thus resulting in a higher number of points having a similar distance to the one marked in red, explaining the first peak. The second peak in Fig 6.2 is formed by the multiple similar distances from the point marked in red to the middle cluster in Fig 6.1. Similarly, the third peak represents the distances to the bottom left cluster. KM and FCM algorithms expect the k parameter (number of clusters) to be provided by the user and here we have the peaks in the distance distribution readily informing us about the number of clusters in the dataset.

Moreover, the distance density plot in Fig 6.2 shows a deep valley between the first peak and the second, much larger than the one between the second and third peak which reflects what
happens in Fig 6.1 where the distance between the cluster with the point marked in red and the middle cluster is higher than the one between the middle and bottom left cluster.

Fig 6.3 Three Clusters with initial randomly chosen point in red and the rest being picked according to the density plot in Fig 6.2

As we have seen so far, given just the distances from the point marked in red to all other points we could estimate the number of clusters in the datasets. Moreover, the peaks of the distribution in Fig 6.2 represent the most common distances with the most common being those that lead to the points in the middle of the clusters. In fact, if we pick cluster centers based on these distances we select points that are close to, if not, the real centers of the clouds. In Fig 6.1 we see the original point marked in red and the points picked according to density plot in blue.
Let us consider Fig 6.4 and show the limitation of such an approach. In this case we have three clusters and we pick the point in red at random from the whole data set. We notice that the point is at about the same distance from the two other clusters (bottom right and top left). The distance density plot in Fig 6.5 shows two peaks of different heights. The first one represents distances to the points closer to the red point in the bottom left cluster. Since the distances from this red point to the bottom right and top left cluster are similar they get mixed together in the second peak in Fig 6.5. This peak is about twice the size of the first one, confirming the mixing problem. This problem prevents us from determining the number of clusters directly from this density distance distribution. Moreover it does not help us in picking the next centers either (as in Fig 6.3).

Further research could focus on these problems perhaps by choosing points that are not situated at equal distances to other clusters. These points do not necessarily have to be come from a cloud of points or the other, they could be generated in the dataspace in such a way as to provide favorable distance distributions. Instead of fitting density distributions as in Fig 6.2 and Fig. 6.5 we could analyze histograms instead and the associated problem of bin partitioning.

Although KM and FCM are well suited for detecting Gaussian clusters of equal size, it would be interesting to investigate this approach on data departing from these assumptions.
Another avenue for research includes the hybridization of the spreading mechanism of FCM++ together with point’s potential value computation (as in Subtractive clustering). In Fig 6.6 we have three clusters and for each point a label is included showing its potential, a value that is higher for points having many neighbours. We could compute these values in the beginning, pick a point with high potential value and then use the spreading mechanism of FCM++ to pick the others. Spreading here should take in consideration the potentials as well: we pick further away points that have higher potential values.

One important challenge here would be in deciding the neighborhood function parameter which specifies the distance radius for computing the potentials. Another challenge would be to decide how much control to allocate to the spreading aspect and how much to potentials, these two being contradictory objectives. The more control we allocate to the potentials aspect the more likely the outcome of picking points in the middle of clusters, points that have little spread between them. For example, let us say we decided to pick $k=3$ points from the clusters in Fig 6.6 during
initialization. If we care just about the potentials of these points, it is likely that the three points will be picked from only one of the clusters which is not an optimal solution.

When it comes to time series clustering, there is another approach that we could investigate in combination with FCM++ in the future. Instead of using MDS on correlation matrix to reduce the number of dimensions we could perhaps transform each of the time series into a series of autocorrelation coefficients. Instead of computing correlation between two series one can compute the correlation of a series with itself at different lags [134]. The autocorrelation reveals information about the inner structure of a time series and, for example, it can tell us whether a day of growth is closely followed by another one (autocorrelation of lag 1). The choice of using autocorrelation was proposed in [14] where an autocorrelation-based FCM was introduced.

Fig 6.7 Euclidean distance between A and B is higher than that between B and C
There are a few properties (Figs 6.7-6.10) of an autocorrelation function which makes the use of autocorrelation coefficients desirable when the task is to group similar shaped time series. For example, Euclidean distance fails to capture the shape of the data when used directly between time series (see Fig 6.7). In this case, we have time series A, B, C for which Euclid (A, B) > Euclid (B, C) although in terms of shape A, B are more similar than B, C. We can resolve this by computing the autocorrelations of each time series and we have the scenario in Fig. 6.8, where Euclid (A, B) < Euclid (B, C) and Euclid (A, B) < Euclid (A, C). Thus, similar shaped time series are grouped together regardless of their amplitude in their raw state.

Fig 6.8 Euclidean distance on autocorrelation coefficients of A and B is lower than that between B and C

Fig 6.9 Euclidean distance between A and B is higher than that between B and C
Similarly, in the case shown in Fig. 6.9, we have time series A, B, C for which Euclid (A, B) > Euclid (B, C) although in terms of shape A, B are more similar than B, C. The only difference is that B is shifted on the X axis. Computing the autocorrelations we have the case shown in Fig. 6.10 where Euclid (A, B) < Euclid (B, C), and Euclid (A, B) < Euclid (A, C). A similar case was presented in [135].

![Fig 6.10 Euclidean distance on autocorrelation coefficients of A and B is lower than that between B and C](image)

An important question that needs to be addressed is how many autocorrelation coefficients to consider, which not a trivial question. Besides, some time series might have more statistically significant coefficients than others and the question raises whether we would then take the least amount to cover all the cases or not (case in which to few will be picked).

Another future work will look at developing a scalable, online parallel implementation of FCM++ for clustering large time series on distributed clusters. The implementation could use parallel pre-processing to capture important features in the time series, making the subsequent clustering not only faster but more accurate. We can cluster incoming time series data of large sizes using the Spark distributed architecture [136] which enables improved speed performance and scalability on large data sets thanks to its caching mechanism which makes use of resilient distributed data sets in fast memory.
Fig 6.11 shows how the architecture of such a distributed system could look like. The master allocates the work to N slaves that all perform a time series reduction in parallel (in this case through autocorrelation computation but it can be any dimension reduction as well). Then, each slave computes FCM on local data (possible with FCM++ initialization) followed by sending the k*N centers to the master which clusters k*N points into k. The slaves receive the k centers, representing the centers of the whole data and they iteratively compute the membership of each local point to each center. These memberships together with the centers can then be written down to files.

Fig 6.11 A Spark Based Distributed Architecture for online FCM++ for time series clustering
Bibliography


[59] A. Tung, J. Han, L. Lakshmanan, and R. Ng, “Constraint-Based Clustering in Large Databases,” in *Database Theory — ICDT 2001*, 2001, pp. 405–419.


observation weighting and context information for reverberant blind speech separation,”

enhanced fuzzy c-means clustering model: Application in the case of forest fires,”

[82] K. R. Sudha, Y. Butchi Raju, and A. Chandra Sekhar, “Fuzzy C-Means clustering for
robust decentralized load frequency control of interconnected power system with
Generation Rate Constraint,” *Int. J. Electr. Power Energy Syst.*, vol. 37, no. 1, pp. 58–66,
2012.

approach to the clustering of supply chain,” *Comput. Ind. Eng.*, vol. 66, no. 4, pp. 768–
780, 2013.


Clustering algorithm and its application on carbonate fluid identification,” *J. Appl.

strategies using Lp norm distances,” *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 576–582,
2000.

[87] A. Idri, A. Zakrani, M. Elkoutbi, and A. Abran, “Fuzzy Radial Basis Function Neural
Networks for Web Applications Cost Estimation,” in *Innovations in Information


markets/stocks/indices/summary/summary-indices-constituents.html?index=UKX.


Appendix

The companies under consideration are summarized below:

- **AMEC.L (AMEC plc.):** provides consultancy, engineering, and project management services to the oil and gas, mining, clean energy, and environment and infrastructure markets worldwide;
- **ARM.L (ARM Holdings):** together with its subsidiaries, designs microprocessors, physical intellectual property (IP), and related technology and software. It also sells development tools that enhance the performance of embedded applications;
- **BAB.L (Babcock International Group plc):** provides engineering support services for defense, energy, emergency services, transport, education, and telecommunications sectors;
- **BATS.L (British American Tobacco plc):** through its subsidiaries, manufactures tobacco and nicotine products. It provides cigarettes, roll-your-own and make-your-own tobacco, cigars, cigarillos, pipe tobacco, and snus, as well as nicotine inhalation, heat-not-burn devices, and electronic cigarettes;
- **BG.L (BG Group plc):** operates as an exploration and production, and LNG company worldwide. Its Upstream segment is engaged in the exploration, development, production, liquefaction, and marketing of hydrocarbons;
- **BLT.L (BHP Billiton plc):** together with its subsidiaries, operates as a diversified natural resources company worldwide;
- **BNZL.L (Bunzl plc.):** distributes non-food consumable products primarily in the Americas, Europe, and Australasia.
- **BRBY.L (Burberry Group plc):** sources, designs, manufactures, and sells luxury apparel and accessories for men, women, and children primarily in Europe, the Americas, and the Asia Pacific region;
- **DGL.L (Diageo plc):** produces, distills, brews, bottles, packages, and distributes spirits, beer, wine, and ready to drink beverages.
- **EXPN.L (Experian plc):** an information services company, provides data and analytical tools to organizations worldwide.
• GFS.L (G4S plc): together with its subsidiaries, provides various security products, services, and solutions;
• GKN.L (GKN plc): is engaged in engineering business worldwide;
• HMSO.L (Hammerson plc): is a publicly owned real estate investment trust. The firm engages in investing, developing, and managing retail properties;
• IMI.L (IMI plc): through its subsidiaries, provides engineering solutions for the control and movement of fluids in critical applications worldwide;
• IMT.L (Imperial tobacco plc): together with its subsidiaries, manufactures, markets, and sells tobacco and tobacco-related products;
• ITV.L (ITV plc): operates as a commercial television network in the United Kingdom;
• KGF.L (Kingfisher plc.): its subsidiaries, operates as a home improvement retailer;
• LAND.L (Land securities group plc.): a real estate investment trust, engages in the ownership, development, and management of commercial properties primarily in the United Kingdom;
• LLOY (Lloyds Banking Group plc.): provides a range of banking and financial services to individuals and businesses in the United Kingdom and internationally;
• LSE.L (London stock exchange group plc.): together with its subsidiaries, engages in the admission of securities to trading; delivery and operation of trading systems;
• MGGT (Meggitt plc.): designs and manufactures components and sub-systems for aerospace, defense, energy, medical, industrial, test, and automotive markets in the United Kingdom and internationally;
• MRW.L (Wm. Morrison Supermarkets plc): operates as a food retailer in the United Kingdom;
• NXT.L (Next plc.): a multi-channel retailer, provides clothing, footwear, accessories, and home products for men, women, and children;
• PFC.L (Petrofac limited): an oilfield service company, provides facilities solutions to the oil and gas production and processing industry worldwide;
• PRU.L (Prudential plc.): together with its subsidiaries, provides retail financial products and services, and asset management services to individuals and businesses;
• PSN.L (Persimmon plc.): through its subsidiaries, operates as a home builder in the United Kingdom;
• RDSB.L (Royal Dutch Shell plc.): operates as an independent oil and gas company worldwide;
• REL.L (Reed Elsevier plc): provides professional information solutions worldwide;
• RIO.L (Rio Tinto plc.): is engaged in finding, mining, and processing mineral resources worldwide;
• RR.L (Rolls Royce Holdings plc.): together with its subsidiaries, provides integrated power solutions for customers in civil and defence aerospace, marine, and energy markets worldwide;
• RRS.L (Randgold Resources Limited plc.): explores and develops gold deposits in Sub-Saharan Africa;
• SAB.L (SABMiller plc): through its subsidiaries, engages in the manufacture, distribution, and sale of beverages;
• SBRY.L (J Sainsbury plc.): together with its subsidiaries, is engaged in the grocery and related retailing activities in the United Kingdom;
• SGE.L (Sage group plc.): together with its subsidiaries, engages in the development, distribution, and support of business management software and related products and services worldwide;
• SN.L (Smith and Nephew plc.): develops, manufactures, markets, and sells medical devices in the advanced surgical devices and advanced wound management sectors worldwide;
• STAN.L (Standard Chartered plc.): provides consumer and wholesale banking products and services primarily in Asia, Africa, and the Middle East;
• SVT.L (Severn Trent plc.): engages in the treatment and provision of water;