Parallel Implementation of Fuzzy Minimals Clustering Algorithm

Isabel Timón, Jesús Soto*, Horacio Pérez-Sánchez, José M. Cecilia

Bioinformatics and High Performance Computing Research Group (BIO-HPC), Computer Science Department, Universidad Católica San Antonio de Murcia (UCAM), Spain

Abstract

Clustering aims to classify different patterns into groups called *clusters*. Many algorithms for both hard and fuzzy clustering have been developed to deal with exploratory data analysis in many contexts such as image processing, pattern recognition, etc. However, we are witnessing the era of big data computing where computing resources are becoming the main bottleneck to deal with those large datasets. In this context, sequential algorithms need to be redesigned and even rethought to fully leverage the emergent massively parallel architectures. In this paper, we propose a parallel implementation of the fuzzy minimals clustering algorithm called *Parallel Fuzzy Minimal* (PFM). Our experimental results reveal linear speed-up of PFM when compared to the sequential counterpart version, keeping very good classification quality.

Keywords: Parallel fuzzy clustering, fuzzy clustering, fuzzy minimals

1. Introduction

We are witnessing the consequences of Web 2.0 where huge amounts of data are continuously generated. Sources of data such as commercial interactions, including financial transactions, search histories, and product information; public

*Corresponding author

Email addresses: `imtimon@alu.ucam.edu` (Isabel Timón), `jsoto@ucam.edu` (Jesús Soto), `hperez@ucam.edu` (Horacio Pérez-Sánchez), `jmcecilia@ucam.edu` (José M. Cecilia)

The authors declare that there is no conflict of interests regarding the publication of this article.

agencies that contribute with medical records, population databases; or scientists that routinely undertake large-scale simulations for weather prediction, drug discovery and so on, are only some examples of this landscape of data deluge. These progressively generated big datasets can be considered as valuable resources, since they can provide with key insights into human behavior, market trends, diseases, engineering safety, environmental change, etc (Duranton et al., 2013; Manyika et al., 2011).

Clustering is a data-analysis technique that aims to organize a collection of patterns (usually represented as a vector of measurements, or a point in a multidimensional space) into groups (or clusters) based on a similarity metric (namely Euclidean, city-block, Mahalanobis, etc) (Tan et al., 2006). As expected, patterns that belong to the same cluster are more similar to each other than they are to a pattern belonging to a different cluster (Jain et al., 1999). Clustering has been successfully applied to the analysis of datasets from several fields such as image processing, pattern recognition, analysis of microarray data in bioinformatics, etc, in order to provide valuable knowledge within these fields (de Hoon et al., 2004; Bezdek, 1981; Wu & Leahy, 1993; Agrawal et al., 1998).

Many data clustering algorithms have been proposed in the literature for many different scientific application (we refer the reader to (Jain, 2010) for a review). A good data clustering algorithm should have the following characteristics: (1) *scalability* i.e. the ability to handle a growing amount of objects and attributes in a capable manner, (2) *adaptability* to determine clusters of different shape or size, (3) *self-driven* as it should require minimum knowledge of the problem domain (e.g., number of clusters, thresholds, termination condition parameters), (4) *stability* as it should remain stable in the presence of noise and outliers, and finally (5) *data-independency* as it should be insensitive to the way the objects are organized in the dataset (Han & Kamber, 2006).

Most of current clustering algorithms depend on iterative procedures in order to find local or global optimal solutions in high-dimensional datasets. The capability to find these solutions usually requires to perform many experiments with different algorithms and to study the influence of different dataset features. Hence, clustering algorithms have a high intrinsic time complexity. For example, the classic *k-means* method is NP-hard even when $k = 2$ (Tong & Kang, 2013). Therefore, the parallelization of clustering algorithms becomes mandatory in the era of big data.

The first parallelization proposals for a data-clustering algorithm were based on *k-means* (Dhillon & Modha, 2000; Nagesh et al., 2000). However, *k-means* provides a hard-partition scheme; i.e. each data point belongs to exactly one cluster. Of particular interest to us are those clustering algorithms that provide multiple and non-dichotomous cluster memberships; i.e *fuzzy* clustering. One of the most widely used fuzzy clustering methods is the fuzzy c-means (FCM) algorithm (Bezdek et al., 1984). Some parallelization efforts have been done in the literature for FCM algorithm to deal with large datasets (Kwok et al., 2002; Ravi et al., 2012; Rahimi et al., 2004; Modenesi et al., 2007; Havens et al., 2012). However, the FCM's execution time grows exponentially with the problem size as it needs prior knowledge about the number of clusters to generate, and therefore, several executions should be done to find out the optimal number of clusters.

In this article, we propose a parallel version of the Fuzzy Minimals (FM) clustering algorithm, which was proposed first by (Flores-Sintas et al., 1998) and modified by (Soto et al., 2008). This algorithm does not need prior knowledge about the number of clusters and presents the advantage that the clusters do not need to be CWS (compact well-separated), being this feature a requirement for parallelization. Our performance evaluation focuses on speed-up and scalability factors, and it reveals that our approach obtains lineal speed-up while clustering quality remains stable compared to the sequential counterpart version. The rest of the paper is structure as follows: Section 2 shows a brief outline of the FM and *Parallel Fuzzy Minimals* (PFM) clustering algorithms. Section 3 describes the experimental results of PFM before we conclude the paper with the main conclusions obtained from our findings and some directions for future work.

2. Methods

2.1. The Fuzzy Minimals Algorithm

The Fuzzy Minimals (FM) algorithm was initially proposed by Flores-Sintas et. al where authors demonstrate that FM algorithm fulfills the expected characteristics of a classification algorithm; i.e. scalability, adaptability, self-driven, stability and data-independent. In what follows, we reprise FM's description and we refer the reader to (Flores-Sintas et al., 1998, 2001; Soto et al., 2008) for insights in the definition and demonstration of FM algorithm.

In general, fuzzy clustering techniques like Fuzzy C-Means (FCM) algorithm (Bezdek et al., 1984) minimize an objective function that determines the prototypes of each cluster. Let X be a set of n data points,

$$X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^F.$$

where F is the dimension of the vector space. FM algorithm uses the objective function given by Equation (1):

$$J_{(v)} = \sum_{x \in X} \frac{d_{xv}^2}{1 + r^2 d_{xv}^2}, \tag{1}$$

where d_{xv}^2 is the Euclidean norm that determines the distance between two points in the dataset. The factor r measures the *isotropy* in the dataset. The use of the Euclidean distance implies that we are assuming the homogeneity and isotropy of the feature space. Whenever the homogeneity and isotropy are broken, clusters are created in the features space. The factor r measures the disruption of the homogeneity and isotropy of the sample by a set of factors affecting the Euclidean distance in each group (Flores-Sintas et al., 2001). Equation (2) shows the factor r calculation in a non-linear expression

$$\frac{\sqrt{|C^{-1}|}}{nr^F} \sum_{x \in X} \frac{1}{1 + r^2 d_{xm}^2} = 1, \tag{2}$$

where $|C^{-1}|$ is the determinant of the inverse of the covariance matrix. m is the mean of the sample X, d_{xm} is the Euclidean distance between x and m, and n is the number of elements of the sample.

In the FM algorithm, the objective function presented in equation 1 is reformulated as shown in Equation (3)

$$J_{(v)} = \sum_{x \in X} \mu_{xv} \cdot d_{xv}^2, \tag{3}$$

where

$$\mu_{xv} = \frac{1}{1 + r^2 \cdot d_{xv}^2}, \tag{4}$$

Equation (4) is the membership function that measures the degree of membership for a given element x to the cluster where v is the prototype. The FM algorithm is an iterative procedure that minimizes the objective function through Equation (5), giving the prototypes that represents each cluster.

$$v = \frac{\sum_{x \in X} \mu_{xv}^2 \cdot x}{\sum_{x \in X} \mu_{xv}^2} \tag{5}$$

Algorithm 1 shows the FM algorithm. It is an iterative process where two standard values are included in the computation. ε_1 establishes the error degree committed in the minimum estimation; and ε_2 shows the difference between potential minimums.

As a fuzzy classification algorithm, the FM computation is similar than well-known FCM algorithm. However, FM algorithm minimizes a different objective function than FCM. The minimum values of the objective function are the prototypes that represent the clusters obtained by the classification process. FM algorithm does not need prior knowledge about the number of prototypes the user wants to identify in the dataset as it is the case of FCM algorithm. FM finds the number of prototypes according to the data-structure which is actually modeled by *factor* r in FM's objective function. Those prototypes are actually the output obtained from FM. Moreover, the clusters do not need to be CWS (compact well separated) in the FM algorithm which is also an important issue in many datasets.

Algorithm 1 Fuzzy Minimals algorithm, where n is the size of the dataset. V is the algorithm output that contains the prototypes found by the clustering process. F is the dimension of the vector space.

```
1: Choose $\varepsilon_1$ and $\varepsilon_2$ standard parameters.
2: Initialize $V = \{\ \} \subset \mathbb{R}^F$.
3: Estimate factor $r$.
4: for $k = 1; k < n; k = k + 1$ do
5:     $v_{(0)} = x_k$, $t = 0$, $E_{(0)} = 1$
6:     while $E_{(t)} \geq \varepsilon_1$ do
7:         $t = t + 1$
8:         $\mu_{xv} = \frac{1}{1+r^2 \cdot d_{xv}^2}$, using $v_{(t-1)}$
9:         $v_{(t)} = \frac{\sum_{x \in X} \left(\mu_{xv}^{(t)}\right)^2 \cdot x}{\left(\mu_{xv}^{(t)}\right)^2}$
10:        $E_{(t)} = \sum_{\alpha=1}^{F} \left(v_{(t)}^{\alpha} - v_{(t-1)}^{\alpha}\right)$
11:    end while
12:    if $\sum_{\alpha}^{F} (v^\alpha - w^\alpha) > \varepsilon_2, \forall w \in V$ then
13:        $V \equiv V + \{v\}$.
14:    end if
15: end for
```

2.2. The Parallel Fuzzy Minimals Algorithm

This section introduces our proposal, the *Parallel Fuzzy Minimal* (PFM) algorithm, which is designed to enhance the execution of the FM clustering algorithm previously explained. FM algorithm does not need to compare clusters to minimize the objective function like FCM does. PFM benefits from this property to divide the original dataset into different *data-partitions* where FM will be applied to. This partition of data does not lose information about global properties for the classification, since each subset will be classified using an objective function that includes factor r, which has information about the overall data structure. Indeed, we consider the factor r as a global parameter that contains information about the whole dataset, and we use it in each of those data-partitions that will run in parallel, maintaining the main clustering characteristics of FM unaltered.

Algorithm 2 Parallel Fuzzy Minimals pseudocode where n is the total number of elements, p is number of processors, V is the set of prototypes, C is the set of cluster obtained by hierarchical clustering, X is the targeted dataset and r is the factor r that measures isotropy and homogeneity of the whole dataset.

```
1: Read X $= \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^F$
2: Initialize $V = \{\ \} \subset \mathbb{R}^F$.
3: Initialize $C = \{\ \} \subset \mathbb{R}^F$.
4: pid = 0
5: Estimate factor $r$
6: for $k = 0; k < n; k = h + (n/p)$ do
7:     pid = pid +1
8:     Send (pid, &X[k], $n/p$, r)
9: end for
10: (All processors) Execute FM (X, $n/p$, r)
11: for $pid = 0; pid < p; pid = pid + 1$ do
12:     (Receive processor) $V \equiv V + Receive(pid)$
13: end for
14: (Master processor) C = Hierarchical clustering (V)
```

The algorithm 2 shows the PFM algorithm. The Master processor (master) prepares the execution and sends the data to each processor involved in the computation. First of all, the master reads the dataset to be classified (X), initializes some structures to store prototypes (V) and clusters of prototypes (C). Then, it computes the factor r, using the whole dataset (X). Next, the master divides the dataset equally among the processors so that each processor handles n/p data points (being n the total number of elements and p the number of processors involved in the computation). Once the different processors receive the information, they proceed with the FM clustering algorithm over the n/p data assigned to it and also with the r factor previously calculated by the master.

Codename	Number of points	Dimension	Size	Source
Iris Data	150	$\mathbb{R}^4$	Small	(Anderson E. , 1934)
Ellipsoids	5.000	$\mathbb{R}^3$	Medium	Synthetically developed
USA Hospitals	15.506	$\mathbb{R}^2$	Large	(NGA , 2015)

Table 1: Benchmarks overview.

Finally, the master gathers all prototypes into an unique group (V) before it proceeds with a hierarchical clustering to determine the final clustering result.

If the data-partition was performed based on a data order, then the classification algorithm would give us completely independent prototypes and thus the hierarchical clustering would not be needed as a final step. However, we do not assume this criteria and data-partitions are performed randomly. Therefore, the prototypes can be very close to each other and PFM needs to check whether these prototypes are actually representative or not. Hierarchical clustering groups data over a variety of scales by creating a cluster tree or dendrogram. The tree is not a single set of clusters, but rather a multilevel hierarchy, where clusters at one level are joined as clusters at the next level. This allows you to decide the level or scale of clustering that is most appropriate for your application.

3. Results and discussion

This section shows the experimental results obtained with the *Parallel Fuzzy Minimal* (PFM) clustering algorithm. We describe the datasets used for the evaluation before we show the experimental results. PFM is first validated by comparing its results with FCM on the well-known Anderson's Iris Data (Anderson E. , 1934). Then, the PFM's performance scalability and clustering quality are shown. Finally, we discuss the benefits of our proposal. We refer the reader to (Flores-Sintas et al., 1998) for a comparison between FCM and FM.

3.1. Datasets

We test our algorithms using a set of benchmark instances that have different sizes and characteristics (see Table 1). Our first dataset is the well-known Andersons Iris Data (Anderson E. , 1934), which consists of the sepal length, sepal width, petal length and petal width for each of 150 irises. The first 50 plants, according to Andersons ordering, are Iris Setosa; the second 50 are Iris Versicolor and the last 50 are Iris Virginica. Iris Versicolor is a hybrid of Iris Setosa and Iris Virginica but it is much more similar to the latter. Consequently, Iris Setosa is easily identified. From the other side, it is very difficult to separate the other two groups.

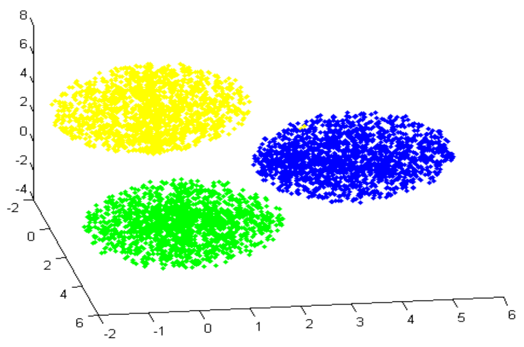


Figure 1: Synthetically developed benchmark that stands for three well-separated ellipsoids.

Our second dataset is synthetically developed for testing the PFM's scalability. It consists of 5000 3-Dimensional points which represent three well-separated ellipsoids (see Figure 1). It is selected to ensure a representative sample, i.e. these ellipsoids are close enough to each other in order to test the quality of our clustering.

Our last benchmark is a large-dataset that has up to 15.506 geographical coordinates of USA hospitals obtained from USA National Geospatial-Intelligence Agency (NGA) (NGA , 2015). It provides 2-dimensional points that are spread out among different states of USA (see Figure 2). This benchmark has been disorganized randomly in order to avoid any correlation.

3.2. *Experimental results*

This Section analyzes Parallel Fuzzy Minimals (PFM) algorithm. We focus on the computational features of the Fuzzy Minimals (FM) algorithm and how it can be designed on parallel systems. To guarantee the correctness of our algorithms, a quality comparison between the results obtained by FCM, FM and PFM is also provided. First of all, we compare PFM to the well-known FCM, provided by Bezdek in (Bezdek, 1981) on the Anderson's Iris dataset. Then, we focus on performance evaluation of PFM algorithm using two different benchmarks previously described. The experiments are developed on a Windows-based laptop machine with 4 GB DDR3 memory and Intel Core I5 1,6 Mhz processor using Matlab 6.0 release 12.

3.3. *FCM Vs. PFM*

As previously mentioned, Anderson's Iris data is a small benchmark which is structured into three main groups, where one of them, Iris Setosa, it is eas-

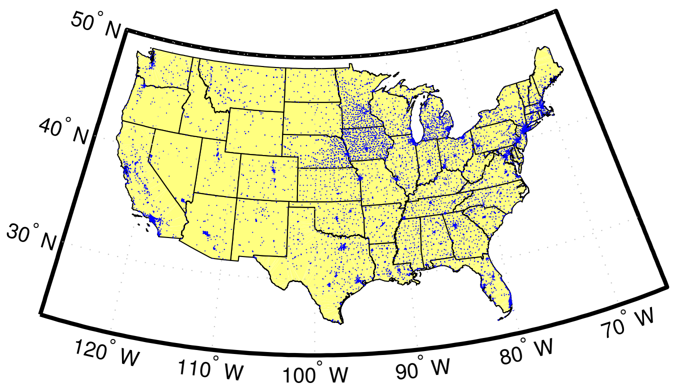


Figure 2: 15.506 geographical coordinates of USA hospitals (source NGA).

ily identified but the others two are very difficult to separate. We divide the Iris dataset randomly into two data-partitions for setting up the PFM algorithm. We use only two data-partitions to make those partitions representative enough. Actually, PFM is designed to deal with large datasets where many data-partitions can be obtained without losing generality. Figure 3 shows the dendogram generated by the PFM's final step; i.e. the hierarchical clustering. It shows three main clusters are found as expected.

Figure 4 shows a quality comparison between FCM and PFM. The x-axis

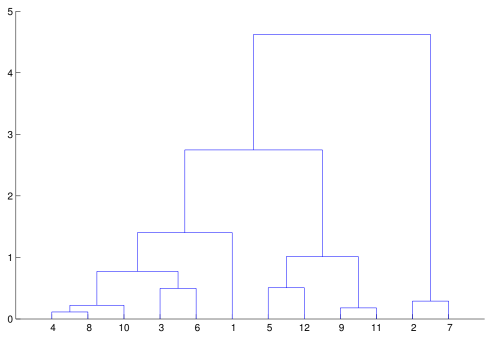


Figure 3: Hierarchical clustering of Iris Data, showing a dendogram that divides the data into three main clusters.

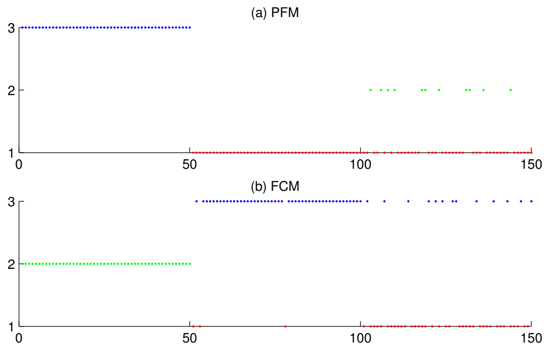


Figure 4: Quality comparison between PFM And FCM on Iris dataset. In x-axis the first 50 plants are Iris Setosa; the second 50 are Iris Versicolor and the last 50 are Iris Virginica. The y-axis shows different clusters where the data points are included.

shows the Iris data according to Andersons ordering where the first 50 plants are Iris Setosa; the second 50 are Iris Versicolor and the last 50 are Iris Virginica. The y-axis shows the cluster that belongs each data. The ideal classification is given by equation $f(x_i) = m_i,\ x_i \in [1, 150], m_i = \{1, 2, 3\}$, then $f(x_i) = f(x_j) \forall i, j \in [1 + 50(k-1), 50k]$, with $k = \{1, 2, 3\}$. Figure 4 also shows that both FM and FCM find three main clusters. Both of them clearly identify Iris Setosa data ant they have some troubles to identify the other two clusters. Those results can be improved after refinement in both algorithms but this is out of the paper's scope.

3.4. PFM Vs. FM

3.4.1. Ellipsoids dataset

Our experimental environment for this benchmark uses five different test cases. They are selected according to the number of data-partitions in which we divide the original dataset (see Figure 1). First of all, we run the PFM algorithm with only one data-partition, which actually represents the full dataset; i.e. we apply FM clustering to the whole dataset. Henceforth, we proceed with the PFM clustering algorithm previously explained in Section 2.2. Four different test cases are executed with 2, 4, 5 and 8 data-partitions of the whole dataset respectively. We ensure that each of these data-partitions contains above 10% of the original dataset as we empirically demonstrate this is the minimum percentage of data, for this particular case, that each subset should have in order to be considered as representative.

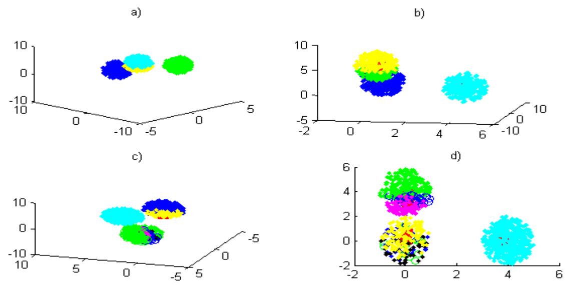
Table 2 shows the execution time in seconds for each of these test cases. The number of subsets, and thus the number of independent processes we execute, are shown on the left-hand side of this table. Then, we show the execution time for each independent process which is basically the execution time to run steps 2 and 3 of PFM algorithm. Finally, we summarize the total execution time, that also includes steps 1 and 4, consisting of both: compute the factor r for the full dataset ($2, 20$ seconds), gathering all prototypes into an unique group and proceed with a hierarchical clustering for this group to eventually deter-

Table 2: Execution times (in seconds) for our Parallel Fuzzy Minimal Algorithm on different data-partition schemes. The number of processes are increased according to the number of subsets in which the original dataset has been divided.

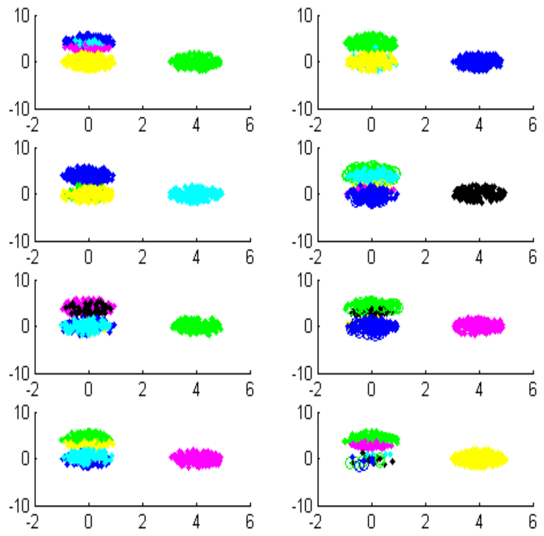
Number of subsets (processes)	Execution time (seconds)	Total execution time (seconds)
1 subset	16.455,22	16.455,22
2 subset	4.438,90	8.206,04
	3.764,93	
4 subset	692,96	3.216,04
	760,61	
	907,39	
	855,08	
5 subset	466,55	2.778,98
	436,97	
	634,87	
	647,85	
	590,52	
8 subset	225,12	1.786,89
	177,94	
	213,95	
	279,42	
	193,60	
	291,01	
	213,02	
	190,63	

mine the number of clusters $(0,57$ seconds). We show a super lineal scalability along with the number of data-partitions (processes). The main computation of PFM is the calculation of the matrix inverse, so if the matrix dimensions are reduced, the computational cost of calculating the inverse of the matrix is reduced substantially. However, there is a bottleneck in the number of subsets we can divide original dataset. As previously explained, subsets should contain above 10% of the original dataset. The data structure is very important for this classification technique as Flores-Sintas stated in (Flores-Sintas et al., 1998). If the original dataset is divided into many data-partitions, the intrinsic properties of the original dataset could be affected and therefore the factor r that measures the isotropy of the whole dataset could not be representative.

Figures 5.a and 5.b show the classification for four and eight data-partitions respectively. They contain up to 4 (or 8) subfigures within them, showing the PFM's classification result for each data-partition based on prototypes selection



(a) Data points distribution for four data-partitions.



(b) Data points distribution for eight data-partitions.

Figure 5: Data points distribution for 4 and 8 data-partitions test cases. Each color represents a different set of points that belong to the same prototype that has been found by PFM's.

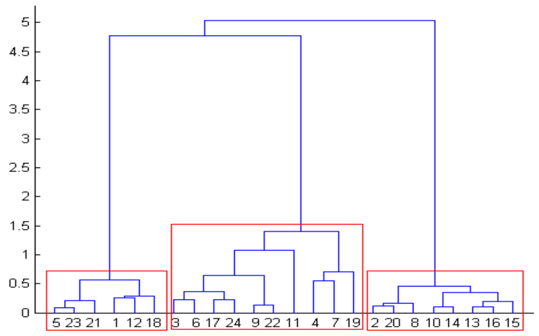
(PFM's step 3 output). Data points in each subfigure represent the data points within each data-partition that belongs to each prototype found by PFM. The number of prototypes found by PFM in each data-partition is the number of different colors drawn in the chart. The Ellipsoids dataset contains up to three different clusters (see Figure 1). However, at this stage we find many prototypes in each subset before performing the step 4 of PFM algorithm.

Figures 6.a and 6.b show the result of PFM's step four; i.e. the hierarchical clustering as applied to all prototypes found in the PFM's step 3. We use the Matlab dendogram plot of the hierarchical cluster tree to illustrate this. A dendogram consists of several U-shaped lines that connect data points in a hierarchical tree. The height of each U represents the distance between the two prototypes previously found. We use the distances among all prototypes found in the data-partition to determine the U. These figures clearly show three main clusters are found in both cases which is exactly the same than the initial dataset shown in Figure 1.

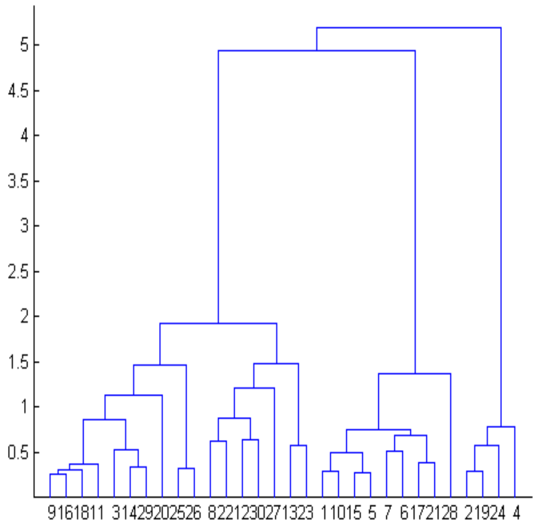
3.4.2. USA Hospitals dataset

Table 3 shows the execution time in seconds for our biggest benchmark; the USA hospitals dataset. On the left-hand side we show the number of data-partitions, and thus the number of processes. The execution time for each independent process and the total execution time is also shown. In this case, the computation of the factor r on the full dataset takes $18,60$ seconds and the hierarchical clustering takes $0,67$ seconds. The performance results for this benchmark also show good scalability along with the number of processors, obtaining up to 10x speed up factor for 15 data-partitions. In this case, we use 8 (PFM8) and 15 (PFM15) data-partitions as we empirically demonstrate they provide very good results. Indeed, this is a tuning process that can be automatized by means of autotuning techniques.

Figure 7 shows the prototypes found by FM algorithm. Up to four clusters are found after hierarchical clustering; they are represented in different colors. Prototypes obtained with PFM8 and PFM15 are also represented in Figures 8



(a) Hierarchical clustering for protoypes found by PFM on Ellipsoids dataset using four data-partitions.



(b) Hierarchical clustering for protoypes found by PFM on Ellipsoids dataset using eight data-partitions

Figure 6: Hierarchical clustering for protoypes found by PFM on Ellipsoids dataset. The dendogram clearly shows three main clusters.

Table 3: Execution times (in seconds) for our Parallel Fuzzy Minimal Algorithm on different data-partition schemes. The number of processes is increased according to the number of subsets in which the original dataset is divided.

Number of subsets (processes)	Execution time (seconds)	Total execution time (seconds)
1 subset	87.461,13	87.461,13
8 subset	1.824,80	19.687,04
	2.619,19	
	2.360,45	
	3.425,11	
	2.183,70	
	2.070,93	
	2.268,95	
	2.915,31	
15 subset	639,56	9.095
	656,79	
	740,30	
	816,33	
	679,47	
	669,71	
	779,61	
	604,17	
	959,73	
	696,19	
	524,15	
	669,39	
	659,12	
	690,40	
	678,01	

and 9 respectively. As expected, the number of prototypes is higher as some of them are actually the same prototype. After hierarchical clustering PFM8 and PFM15 also identifies four clusters.

Finally, Figure 10 shows the mean of all prototypes found within each cluster for each algorithm. Blue points represents FM candidates, green points PFM8 and red points PFM15. The prototypes distribution of FM, PFM8 and PFM 15 are very similar to each other. The main difference can be found on Hospitals at the west coast. In this part of the dataset, the number of correlations is very high compared to others. FM algorithm is highly influenced by correlations since the determinant of covariance matrix would be zero, and therefore there would not be matrix inverse. PFM, however, divides the dataset into data-partitions, and thus it is likely to have less correlations as they are randomly distributed among data-partitions. Nevertheless, we could not make sure correlations are uniformly distributed and thus the PFM's user have the opportunity to analyze the dataset and distribute the correlations among different data-partitions.

4. Conclusions

Big Data has the potential to transform development and accelerate social progress all over the world. There are several issues surrounding this new era, though, that should be addressed before taking a step further. Among them, we may highlight the lack of computational resources available to deal with this data deluge. Nowadays, the computational field is massively parallel as

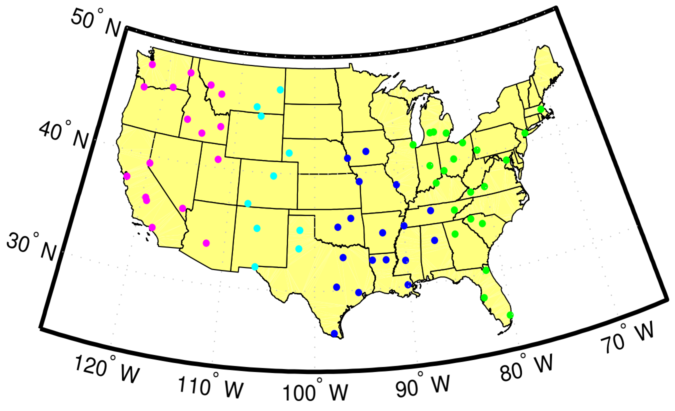


Figure 7: FM's hierarchical clustering representation. Four clusters are found; each cluster is represented by a different color.

a consequence of the new trend in developing new microprocessors. This fact makes mandatory the redefinition of our algorithms to fully leverage the new massively parallel platforms. In this paper, we redefine a clustering technique called Fuzzy Minimals, to enhance the classification of large datasets. Our experimental results reveal linear speed-up along with the number of parallel processes launched while the classification quality is still good enough with our initial assumptions. Moreover, we have noticed a bottleneck in the data division as the subsets created from the initial dataset should contain at least above 10% of the total data in the targeted dataset. Actually, we are working on establishing a general rule that defines the maximum number of subsets that a given data-partition could have in order to achieve peak performance. We are also working on executing our algorithm in a massively parallel environment such a supercomputer. To do so, we will migrate our Matlab implementation to the C programming language and MPI interface.

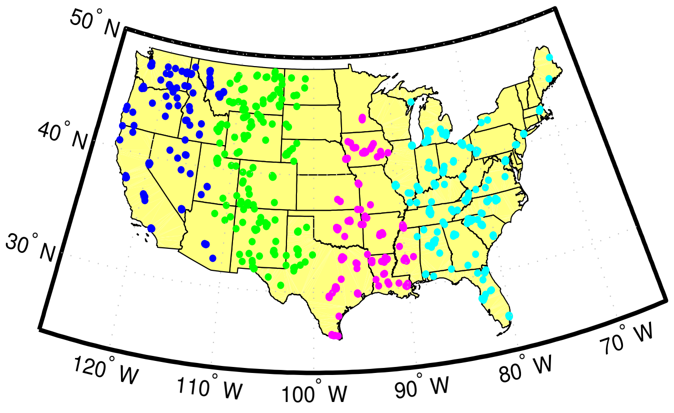


Figure 8: PFM8's hierarchical clustering representation. Four clusters are found; each cluster is represented by a different color.

Acknowledgements

This research was supported by the Fundación Séneca (Agencia Regional de Ciencia y Tecnología de la Región de Murcia) under grant 18946/JLI/13 and by the Nils Coordinated Mobility under grant 012-ABEL-CM-2014A.We also thank Nvidia for the hardware donation under GPU research and Educational center. We want to thank anonymous reviewers for the valuable suggestions they have made to improve the overall quality of this paper. Finally, we warmly thank Antonio Flores-Sintas, who died while the submission process, for his contributions to this work.

References

Anderson E. (1934). The Irises in the Gaspe Peninsula . *Bulletin of the American Iris society*, *59*, 2–5.

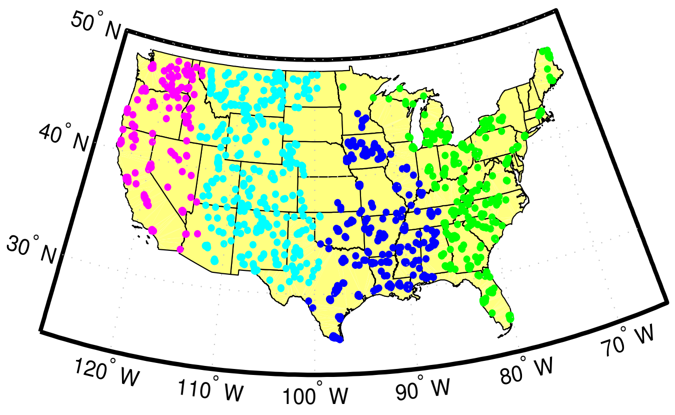


Figure 9: PFM8's hierarchical clustering representation.Four clusters are found; each cluster is represented by a different color.

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications *SIGMOD Rec.*,*27*,94–105.

320 Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers.

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, *10*, 191–203.

Dhillon, I. S., & Modha, D. S. (2000). *A data-clustering algorithm on distributed memory multiprocessors*. Springer-Verlag.

Duranton, M., Black-Schaffer, D., De Bosschere, K., & Maebe, J. (2013). *The hipeac vision for advanced computing in horizon 2020*. HiPEAC High-Performance Embedded Architecture and Compilation.

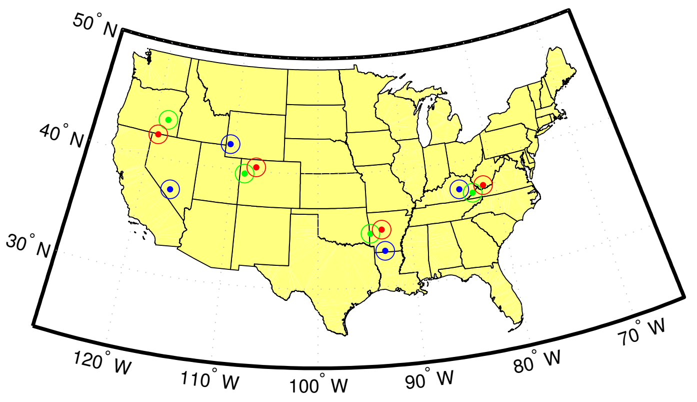


Figure 10: Means for clustering in 4 group of total set of prototypes. Blue points are FM candidates, Green points are PFM8 candidates and red for PFM15.

Flores-Sintas, A., Cadenas, J., & Martin, F. (1998). A local geometrical properties application to fuzzy clustering. *Fuzzy Sets and Systems*, *100*, 245–256.

Flores-Sintas, A., M Cadenas, J., & Martin, F. (2001). Detecting homogeneous groups in clustering using the euclidean distance. *Fuzzy Sets and Systems*, *120*, 213–225.

Han, J., & Kamber, M. (2006). *Data Mining, Southeast Asia Edition: Concepts and Techniques*. Morgan kaufmann.

Havens, T.C., & Bezdek, J.C., & Leckie, C., & Hall, L.O., & Palaniswami, M. (2012). Fuzzy c-Means Algorithms for Very Large Data. *Fuzzy Systems, IEEE Transactions on*, *20*, 1130-1146.

de Hoon, M. J., Imoto, S., Nolan, J., & Miyano, S. (2004). Open source clustering software. *Bioinformatics*, *20*, 1453–1454.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, *31*, 651–666.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*, 264–323.

Kwok, T., Smith, K., Lozano, S., & Taniar, D. (2002). Parallel fuzzy c-means clustering for large data sets. *Euro-Par 2002 Parallel Processing* (pp. 365–374). Springer Berlin Heidelberg.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute.

Modenesi, M. V., Costa, M. C., Evsukoff, A. G., & Ebecken, N. F. (2007). *Parallel fuzzy c-means cluster analysis*. Springer Berlin Heidelberg.

Nagesh, H. S., Goil, S., & Choudhary, A. (2000). *A scalable parallel subspace clustering algorithm for massive datasets*. In *Parallel Processing, 2000. Proceedings. 2000 International Conference on* (pp. 477–484). IEEE.

National Geospatial-Intelligence Agency. `https://www.nga.mil/Pages/Default.aspx` (last accessed Oct 27, 2016).

Rahimi, S., Zargham, M., Thakre, A., & Chhillar, D. (2004). A parallel fuzzy c-mean algorithm for image segmentation. In *Fuzzy Information, 2004. Processing NAFIPS'04. IEEE Annual Meeting of the* (pp. 234–237). IEEE.

Ravi, A., Suvarna, A., DSouza, A., Reddy, G. R. M. et al. (2012). A parallel fuzzy c means algorithm for brain tumor segmentation on multiple mri images. In *Proceedings of International Conference on Advances in Computing* (pp. 787–794). Springer India.

Soto, J., Flores-Sintas, A., & Palarea-Albaladejo, J. (2008). Improving probabilities in a fuzzy clustering partition. *Fuzzy Sets and Systems*, *159*, 406–421.

Tan, P., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining*. Addison Wesley.

Tong, H., & Kang, U. (2013). Big data clustering. In Charu C. Aggarwal, Chandan K. Reddy (Eds.), *Data Clustering: Algorithms and Applications*. CRC Press.

Tsao, E. C.-K., Bezdek, J. C., & Pal, N. R. (1994). Fuzzy kohonen clustering networks. *Pattern recognition*, *27*, 757–764.

Wu, Z., & Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *15*, 1101–1113.