Fuzzy Clustering. Criterion to Class Depuration.

Celso Alberto Rojas Pukall
Facultad Politécnica, Universidad Nacional del Este
Ciudad del Este - Paraguay
albertorp@politec.une.edu.py

y

Benjamín Barán Cegla
Centro Nacional de Computación, Universidad Nacional de Asunción
Asunción - Paraguay
bbaran@sce.cnc.una.py

ABSTRACT

This paper proposes an approach to fuzzy clustering that ameliorates the data sample considered for clustering in the sense that it first removes some few atypical sparse items that are not relevant to the whole set topology, but that may severely interfere, specially when trying to find out the optimal number of clusters in the set. Basically, the approach consists of successive employment of a fuzzy equivalence relation-based hierarchical clustering method for the purpose of pruning the sample from spurious data, followed by a c-means clustering method based on an objective function optimization. The latter method is applied to get the objective function performance for several probable cluster numbers. By analyzing the objective function behavior and by some other consideration regarding the fuzzy equivalence relation-induced partition spectrum on the data set, one can achieve the optimal cluster quantity.
Keywords: fcm, hierarchical clustering, pattern recognition, pruning, parallel processing.

1 INTRODUCTION

The objective of cluster analysis is to group a set of objects into clusters such that items within the same cluster have a high degree of similarity, while objects belonging to different clusters have a high degree of dissimilarity. Associated with clustering problem, there is the specification of *c*, the most appropriate number of clusters in *X*, the set of *n* data to be grouped which represent the studied objects [5, 8].

Cluster analysis is intrinsically related to pattern recognition in data structure. There are three fundamental problems in pattern recognition. The first one is concerned with the representation of input data obtained by measurements on objects that are to be recognized: the *sensing problem*. In general, each object is represented by a vector of measured values of *s* variables, $\mathbf{x} = [x_1, x_2, ..., x_s]$ this vector is usually called a *pattern vector*. The second problem concerns the extraction of characteristic features from the input data in terms of which the dimensionality of pattern vectors $\mathbf{x}_i$ can be reduced: the *feature extraction problem*. Valid features must characterize attributes by which the given pattern classes are well discriminated. The third problem involves the determination of optimal decision procedures to classify given patterns, this is usually done by defining an appropriate discrimination function that assigns a real number to each pattern vector; the pattern vector is then classified into the class whose discrimination function yields the largest value [8].

In classical cluster analysis, clusters are built around *c* class centers, each defining the relevant classes in *X*, these classes are required to form a *partition* of *X*. However, this requirement is to strong in many practical applications, and it is thus desirable to replace it with a weaker requirement[8]. When the crisp partition requirement is replaced with a weaker requirement of a fuzzy *pseudopartition* of *X*, one enters the problem area of *fuzzy clustering*. Fuzzy pseudopartitions are often called *fuzzy c-partition*.

Fuzzy clustering can be approached by two basic methods: hierarchical and non-hierarchical methods, among the hierarchical ones there is the *fuzzy equivalence relation-based hierarchical clustering method*. The underlying idea in non-hierarchical methods consists in choosing some initial fuzzy *c*-partition by assigning membership values to data, relating to the *c* classes and then alter these values in a way to obtain better partitions of *X* according to some predefined appropriate objective function.

Although the analyst is the irreplaceable responsible for getting a valid clustering, almost each usual method are relying on computers to process enormous quantity of real problem data. In this sense, either method has its own assets and weaknesses. Hierarchical methods do not require predefinition of *c*, but some of the drawbacks using them are related to memory size and per iteration complexity of updating the membership matrix [5], in fact, note that hard *c* partition space M_c is finite but quite large for all but trivial values of *c* and *n* [2]:

$$|M_c| = (1/c!)\left[\sum_{j=1}^{c}\binom{c}{j}(-1)^{c-j}\, j^n\right] \quad (1)$$

is the number of distinct ways to partition *X* into *c* nonempty subsets. Besides, fuzzy equivalence relation-based hierarchical method suits better to sets with chain structure, linking data according to nearest-neighbour distance. On the other side, fuzzy partition methods based on objective function (*fuzzy c-means*) tend to associate data according to their proximity to class centers, this may be desirable but only works well with hyperspherical, roughly equal proportioned and well separated clusters; and need *c* to be predefined, apart from finding a suitable J_m, the objective function [2].

In a way, the present work combines both approaches, besides, it employs parallel processing, in order to profit from each method goodness for getting an optimal fuzzy clustering without having to pay a high price for it in terms of memory size and computing time. The following sections, briefly present the fuzzy equivalence relation-based hierarchical method, the fuzzy c-means method and the proposal of a combined use of them. Then the employed algorithms are described; both: the sequential and the parallel versions, the results are shown an finally the conclusions are presented.

2 FUZZY EQUIVALENCE RELATION-BASED HIERARCHICAL CLUSTERING.

A relation *R*(*X*, *X*) is *reflexive* if and only if

$$r_{ii} = 1, \quad 1 \le i \le n \quad (2)$$

where *r* represents the relation defined on *X* it is *symmetric* if and only if

$$r_{ij} = r_{ji}, \quad 1 \le i \ne j \le n \quad (3)$$

and , if the relation is fuzzy, it is *max-min transitive* if and only if

$$r_{ik} \ge \max_{\forall j \in n}\{\min[r_{ij}, r_{jk}]\} \quad \forall\, r_{ik} \in R \quad (4)$$

A fuzzy relation that is reflexive according to (2), symmetric according to (3) and transitive according to (4), is known as a *fuzzy equivalence relation* or *similarity relation* [8,9].

In a fuzzy set *A*, their elements generally have membership levels in the real interval [0, 1], zero meaning non membership; one, absolute membership, and intermediate values, partial membership. There is a way of restringing membership level that is particularly important. It is the restriction of membership degrees that are greater than or equal to some chosen value α in [0, 1]. When this restriction is applied to a fuzzy set *A*, one obtain a crisp subset ${}^{\alpha}A$ of *X* which is called an α*-cut* of *A*. Formally [9], ${}^{\alpha}A = \{\mathbf{x} \in X \mid A(\mathbf{x}) \ge \alpha\}$ for any $\alpha \in$ [0, 1], where *A*(**x**) denotes the membership degree of **x** in *A*.

Fuzzy relations are also fuzzy sets, and as such, they have all general properties defined in fuzzy set theory. An operation defined on binary fuzzy relations (binary because it relates elements from two sets) that turns to be particularly pertinent to this work is the *max-min composition* that is following defined. Let *P*(*X*,*Y*) and *Q*(*Y*,*Z*) be two binary fuzzy relations with set *Y* in common, the *standard composition* or *max.min composition* of these two relations, denoted by *P*(*X*,*Y*) °*Q*(*Y*,*Z*), produces a binary relation *R*(*X*,,*Z*) on the cartesian product *X* x *Z* defined by

$$R(x, z) = (P \circ Q)(x, z) = \max_{y \in Y}(\min(P(x, y), Q(y, z))) \quad (5)$$

In this context, each fuzzy equivalence relation induces a crisp partition in each of its α-cuts. So, fuzzy clustering problem can be envisaged as the problem of identifying an appropriate fuzzy equivalence relation on the considered data. Although this cannot usually be done directly, it is possible to readily determine a fuzzy compatibility relation (that is, a relation that is only reflexive and symmetric, but not transitive) in terms of an appropriate distance function applied to given data. Then, a meaningful fuzzy equivalence relation is defined as the *transitive closure* of this compatibility relation (the relation that is transitive, contains *R*(*X*, *Y*), and has the fewest possible members).

Given a set of data *X* (*n s*-tuples of $\Re^s$), let a fuzzy compatibility relation, *R*, on *X* be defined in terms of an appropriate distance function of the Minkowski class by the formula

$$r_{ik} = R(\mathbf{x}_i, \mathbf{x}_k) = 1 - \delta(\sum_{j=1}^{s} | \mathbf{x}_{ij} - \mathbf{x}_{kj} |^q)^{1/q} \quad (6)$$

for all pairs $<\mathbf{x}_i, \mathbf{x}_k> \in X$, where $q \in \Re^+$, and δ is the inverse value of the largest distance in X, a constant that ensures that $R(\mathbf{x}_i, \mathbf{x}_k) \in [0, 1]$.

In general, R defined by (6) is a fuzzy compatibility relation, but not necessarily a fuzzy equivalence relation. Hence, it is usually required to determine the transitive closure of R. This can be done by a simple algorithm based on the following theorem [8]:

THEOREM 1 TRANSITIVE CLOSURE.
Let R be a fuzzy compatibility relation on a finite universal set X with $|X| = n$. Then, the max-min transitive closure of R is the relation $R^{(n-1)}$.

ALGORITHM 1 TRANSITIVE CLOSURE

Calculate the sequence of relations

$$R^{(2)} = R \circ R, \qquad R^{(4)} = R^{(2)} \circ R^{(2)}, \ldots \qquad R^{(2^k)} = R^{(2^{k-1})} \circ R^{(2^{k-1})} \quad (7)$$

until no new relation is produced or $2^{(k)} \geq$ n-1. End.

3 FUZZY CLUSTERING BASED ON OBJECTIVE FUNCTION (FCM)

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be a set of given data. A fuzzy pseudopartition or fuzzy c-partition of X is a family of fuzzy subsets of X, $\wp = \{A_1, A_2, ...A_c\}$, which satisfies

$$\sum_{i=1}^{c} A_i(x_k) = 1, \qquad 1 \leq k \leq n \quad (8)$$

and

$$0 < \sum_{k=1}^{n} A_i(x_k) < n, \qquad 1 \leq i \leq c. \quad (9)$$

The problem of fuzzy clustering is to find a fuzzy pseudopartition and the associated cluster centers by which the structure of the data is represented as best as possible. This requires some criterion expressing the general idea that associations be strong within clusters and weak between clusters. To solve the problem of fuzzy clustering, it is necessary to formulate this criterion in terms of a *performance index*. Given a pseudopartition $\wp$, the c cluster centers $V = \{\mathbf{v}_1, \mathbf{v}_2, ... \mathbf{v}_c\}$ associated with the partition are calculated by the formula

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} A_i[(\mathbf{x}_k)]^m \mathbf{x}_k}{\sum_{k=1}^{n} A_i[(\mathbf{x}_k)]^m}, \qquad 1 \leq i \leq n \quad (10)$$

where m>1 is a real number that governs the influence of membership grades[2, 8].
The performance index of a fuzzy pseudopartition $\wp$, $J_m(\wp, V)$, is then defined in terms of the cluster centers by

$$J_m(\wp, V) = \sum_{k=1}^{n} \sum_{i=1}^{c} [A_i(\mathbf{x}_k)]^m \| \mathbf{x}_k - \mathbf{v}_i \|^2, \quad (11)$$

where $\| . \|$ is some inner product-induced norm in space $\Re^s$ and $\| \mathbf{x}_k - \mathbf{v}_i \|^2$ represents the distance between $\mathbf{x}_k$ y $\mathbf{v}_i$. This objective function measures the weighted sum of distances between cluster centers and elements in the corresponding fuzzy clusters. In this work, $\| . \|$ represents the Euclidean distance in all cases. Clearly, the smaller the value of J_m, the better the fuzzy pseudopartition $\wp$. Therefore, the goal of the fuzzy c-means clustering method is to find a fuzzy pseudopartition $\wp$ that minimizes J_m. The usual algorithm for this method was developed by Bezdek [1981] which is described below [2, 8]. The algorithm is based on the assumption that the desired number of clusters c is given and, in addition, a particular distance, a real number $m \in (1, \infty)$, and a small positive number ε, serving as a stopping criterion, are chosen.

ALGORITHM 2.FCM.

Step 1. Let $t = 0$. Select values for c, m, ε, norm $\| . \|$ and an initial fuzzy c-partition $\wp^{(0)}$ that satisfies (8)-(9).
Step 2. Calculate the set $V^{(t)}$ of the c cluster centers with (10) for $\wp^{(t)}$ and the chosen value of m.

Step 3. Update $\wp^{(t+1)}$ by the following procedure: For each $\mathbf{x}_k \in X$, if $\| \mathbf{x}_k - \mathbf{v}_i \|^2 > 0$, $\forall\, i,\ 1 \le i \le c$, then define

$$A_i^{(t+1)}(\mathbf{x}_k) = \left[\sum_{j=1}^{c} \left(\frac{\| \mathbf{x}_k - \mathbf{v}_i^{(t)} \|^2}{| \mathbf{x}_k - \mathbf{v}_j^{(t)} \|^2} \right)^{\frac{1}{m-1}} \right]^{-1} ; \qquad (12)$$

if $\| \mathbf{x}_k - \mathbf{v}_i \|^2 = 0$ for some $i \in I \subseteq N_c$, $N_c = \{1, 2, ...c\}$, then define $A_i^{t+1}(\mathbf{x}_k)\ \ \forall i \in I$ by any nonnegative real number satisfying

$$\sum_{i \in I} A_i^{(t+1)}(\mathbf{x}_k) = 1, \qquad (13)$$

and define $A_i^{(t+1)}(\mathbf{x}_k) = 0 \quad \forall i \in N_c - I \qquad (14)$

Step 4. Compare $\wp^{(t)}$ with $\wp^{(t+1)}$. If $|\wp^{(t+1)} - \wp^{(t)}| \le \boldsymbol{\varepsilon}$, then stop; otherwise, increase t by one and return to **Step 2**. End.

In this work, distance $|\wp^{(t+1)} - \wp^{(t)}|$ in step 4 is taken as

$$| \wp^{(t-1)} - \wp^{(t)} | = \max_{i \in N_c, k \in N_N} | A_i^{(t-1)}(\mathbf{x}_k) - A_i^{(t)}(\mathbf{x}_k) | \qquad (15)$$

In algorithm 2, m is chosen according to the case; when $m \to 1$, the process converges to a "generalized" classical c means. When $m \to \infty$, all cluster centers tend towards the whole data set center of gravity. That is, the partition becomes fuzzier with increasing m.
If m=1, and a crisp partition of X is chosen so that, instead of (8)-(9),

$$\bigcup_{i=1}^{c} A_i = X \qquad (16)$$

$$A_i \cap A_j = \varnothing, \quad 1 \le i \ne j \le c \qquad (17)$$

$$\varnothing \subset A_i \subset X, \quad 1 \le i \le c \quad , \qquad (18)$$

then method *fuzzy c-means* or *fcm* becomes the classical *hard c-means* or *hcm* and it is required to redefine (12)-(14) as

$$A_i^{(t+1)}(\mathbf{x}_k) = \begin{cases} 1, & d_{ik}^{(t)} = \min\limits_{1 \le j \le c} \{d_{jk}^{(t)}\} \\ 0, & \textbf{\textit{otherwise}} \end{cases} \qquad (19)$$

where $d_{ik} = \| \mathbf{x}_k - \mathbf{x}_i \|$. So modified, algorithm 2, becomes Duda and Hart hcm algorithm [2], in fact, it precedes the fcm algorithm, which is a kind of generalisation of hcm. If in (19) it happens to be two or more minimum distances, then such singularity can be solved by assigning $\mathbf{x}_k$.to the first closest center.

4 PROPOSED CRITERION TO CLASS DEPURATION

Most practical applications of cluster analysis combine several techniques to overcome problems any one method presents. Neither mathematics or the art of intuitive judgement produce alone what is readily achievable with their combination [10]. According to appointments made in section 1, fuzzy equivalence relation-based hierarchical clustering method could complement to fuzzy *c*-means method, since it lends itself to deal with data that have pseudolinear structure, while the latter turns out to be appropriate for clustering data with structure of hyperspherical, equal proportioned and well separated clusters.

As to *c value*, while *fcm* method needs it to be predefined, the fuzzy equivalence-based hierarchical method is able to find all partitions induced by the r_{ik} calculated with (6), which are elements of matrix R_{cxn}, which represents the studied relation. This equivalence levels, r_{ik} determine natural α-cuts for those partitions such that each of them intrinsically define a particular value of *c*, the number of classes at that relation level. Then, unfortunatelly, in real problems, it is rather hard to find chain topologies where the fuzzy equivalence-based hierarchical method work, not to mention the disturbance often introduced by some atypical or spurious data that could damage severely the clustering result.

This work proposes an approach to finding the optimal *c* by a successive employment of the fuzzy equivalence relation-based hierarchical clustering method for the purpose of pruning the sample from spurious data, followed by a c-means clustering method based on an objective function optimization. With the first method it is possible to detect data that are

so isolated from the rest forming alone their own class (simpletons) or sharing the class with other few data, even at low α-cuts. When applying the pruning criterion to remove these spurious data from set X, it is required to define a threshold value T such that $\eta \geq T$, where η is the number of elements of a significant class for the clustering effect. The idea that lies under pruning X from "noise data" is to stress the cluster limits by slightly increasing the intercluster distance. Thus, the remaining set after the pruning process was applied, would increase its probably to show the desired characteristics that might favour a valid c-means clustering of it.

In this work, the T value is chosen together with the selection of an α value that limits the acceptable membership degree inside a cluster, a trade-off must be assumed here: the higher the α value chosen (that is, the higher the intracluster cohesion), the bigger the number of cut off data. In the example presented in this paper, α was chosen equal to 0.8.

Optimal *c* calculation

The uncertainty about X topology renders impossible the calculation of the optimal number of clusters right from the analysis of the set partition induced by some chosen α in the transitive closure of the matrix relation. Instead, it is preferable to apply the *c*-means objective function-based method to the data sample once it has been pruned from non representative data.

After the sample depuration, the fuzzy *c*-means (*fcm)* method is applied for several values of *m*, optimizing the objective function (11) with algorithm 2 if $m>1$ or using its variant (*hcm*) if $m=1$. Next, corresponding values of $J_m(c)$ are plotted versus c values in order to obtain its behavior as a function of the number of clusters in a range that one judges must contain the optimal value for c.

With the set of data pairs (c, J_m) and some appropriate numerical method, for example linear or polynomial regression [4], the best representative curve for the employed method can be obtained in order to have a reference to which the raw data curve is contrasted and the relative errors e are calculated for each c value as

$$e = \frac{| J_m(c) - \overline{J_m(c)} |}{\overline{J_m}(c)} \tag{20}$$

where J_m is the unadjusted objective function and $\overline{J_m}$ is the same function previously adjusted. The value of c for which e maximizes, is then chosen as the optimal one.

In the example described in section 5, instead of using (11) for the objective function, a reformulation of it was taken, where J_m appears as a function only of V but is completely equivalent in terms of results [6, 7].

For $m=1$, the classical hcm method, the reformulated objective function is

$$R_1(V) = \sum_{k=1}^{n} \min_{i \in N_c} \{d_{ik}^2\} \tag{21}$$

where $d_{ik} = \| \mathbf{x}_k - \mathbf{x}_i \|$.

And for $m>1$, the fcm method, the reformulated objective function is [3, 6]

$$R_m(V) = \sum_{k=1}^{n} \left(\sum_{j=1}^{c} \left(d_{jk}^2 \right)^{\frac{1}{1-m}} \right)^{1-m} \tag{22}$$

So, in the example it is R_m that is optimized instead of J_m, since the reformulation theorem demonstrated in [7] provides theoretical justification for this approach.

Although the procedure described here cannot completely guarantee the arrival to the best value of c, nor even the validity of X classification, it yields however an appreciable aid for it permits a better cluster limit definition as proved in the implementation described below. A question to be taken into account regarding the proposed method applicability is the greatest increment between two consecutive α values in (0, 1) from the transitive closure matrix R, if the analyst considers that greatest increment too large compared to the rest in the whole α spectrum of R, and besides, if the induced partition by the larger α value of the two values forming that greatest increment renders a c value different from that of the method proposed here, then probably, further analysis must be made in order to elucidate the optimal value for c.

Memory size for matrix R would normally be $O(n^2)$, but one can profit from the symmetry and reflexivity properties of R in order to save the whole lower triangle matrix space with the diagonal included. Thus, only $O(n(n-1)/2)$ matrix space is occupied, that is, less than $O(n^2/2)$.

Matrix processing time is frequently quite long for real values of n. The transitive closure calculation described in algorithm 1 is very time demanding as verified in the example presented where n=2000. In order to improve processing time and to save memory space, in the following, the proposed algorithm is presented in a sequential version that implements matrix R in a one-dimensional array (algorithm 3), and a parallel version of it (algorithm 4) that speeds up the heaviest parts of the process.

For the example problem implementation, the hardware utilized consisted of Pentium II processors, each with 400 Mhz clock and 32 MB memory. A LINUX network of six machines was used for the parallel version, with functions for message communication from the MPICH library and all the code in C language.

Utilized Algorithms

ALGORITHM 3. SEQUENCIAL PROCESSING

Step 1. Choose values of c, m, α, $\boldsymbol{\varepsilon}$, T and norm $\| \cdot \|$ for measuring distances in data vector space $\Re^s$. Input data to memory and save them in matrix X_{nxs}.
Step 2. Calculate the value of δ for (6).
Step 3. Form strictly triangular upper matrix R, which represents the compatibility relation defined by (6). Implement it in a unidimensional array so as to minimize memory use.
Step 4. Find the transitive closure of matrix R (algorithm 1) through max-min composition operations defined by (5)
Step 5. Run R row by row in search of classes with member number equal to or greater than T at level α. If found, save their members, one by one, in a file of depured data.
Step 6. Apply algorithm 2 to the n' data from depured data file.
Step 7. Enter depured data in another matrix $X_{n'x\,s}$ and apply steps 2 to 4 to them.
Step 8. Sort found α values in transitive closure matrix $X_{n'x\,s}$.
Step 9. Find greatest increment between two consecutive values of α, excluding unity. End.

As mentioned before, parallel processing was carried out with message passing interface MPICHC, which demand the storage of the whole data set in each processor independently. The employed algorithm for parallel processing is the following one:

ALGORITHM 4. PARALLEL PROCESSING

Step 1. Choose values of c, m, α, $\boldsymbol{\varepsilon}$, T and norm $\| \cdot \|$ to measure distances in $\Re^s$. *Each process*: after being spawned and identified, input data sample and save them in matrix X_{nxs}.
Step 2. *Each process*: divide the data in equal shares so as to work only on its corresponding share. Each share is identified by its lower and upper row and column limits in the R matrix to be built.
Step 3. *Each process*: calculate the δ value with (6) on its share data.
Step 4. *Each process*: (except the master): send to the master the found value of δ. *The master*: receive the δ values of each processor and calculate the final value as the minimum of them.
Step 5. *The master*: send to all other processes the final value of δ, then build its share of matrix R. *Each process*: (except the master): receive the final δ value, then build its share of matrix R. R must be a strictly triangular upper matrix implemented in a one-dimensional array to minimize memory use
Step 6. *Each process*: (except the master): Send to the master its share of matrix R. *The master*: receive other processes share of matrix R and complete matrix it.
Step 7. *The master*: send to each process the whole matrix R, except for the process share of it, which is already updated. *Each process*: (except the master): Receive the updated part of R and complete it.
Transitive closure iteration start.
Step 8. *Each process*: Find the transitive closure of R (algorithm 1) with max-min composition defined by (5) in an iterative way on its share of matrix R. Verify whether it still changes.
Step 9. *Each process*: (except the master): send to the master its updated share of matrix R and the change status *The master*: receive the R updated share an the information on change status. Then complete matrix R with the updated shares received and decide whether to stop or not on change status.

Step 10. *The master*: send to each process the whole matrix *R*, except for the process share of it, which is already updated and the information whether to stop or not. Then, if more iteration is required, go to step 8, otherwise escape the iterative process. *Each process* (except the master): receive from the master the whole matrix *R*, except for its already updated share, and the information whether to stop or not. Then, if more iteration is required, go to step 8, otherwise escape the iterative process.
Transitive closure end.
Step 11. *The master*: run matrix *R* row by row in search of classes with member number equal to or greater than *T* at level α, if found, save its members sequentially one after another in a depured data file.
Step 12. *The master*: apply algorithm 2 to data from depured data file.
Step 13. *Each processes* (*one as master*) : enter depured data into matrix $X_{n' \times s}$ and apply steps 2 through 10 to them.
Step 14. *The master*: sort found α values in the new transitive closure matrix *R*.
Step 15. *The master*: find the larger increment between two adjacent α excluding unity. End.

In algorithm 4, sections corresponding to algorithm 2 and the sort procedure were not parallelized since they turned out to be fast to process.

5 APPLICATION EXAMPLE WITH REAL DATA

Next is the proposed method example: to cluster data originated in a 1996 household survey by the paraguayan national statistics institut (DGEEC) and converted to a set *X* with *n* = 2000 and *s* = 19, they describe to welfare/poverty level of paraguayan households [1].

Parameters Selection:

Number of data in set *X* *n* = 2000,
Vector space dimensionality: *s* = 19.
Norm for measuring distances in vector space $\Re^s$: || . || = Euclidean distance.
Membership degree at which pruning is applied: α = 0.800.
Pruning threshold: *T* = 20.
Fuzziness Parameter: *m* = 1.0, 1.5 and 2.0
Domain for *c*: N_c ={1, 2, 3, 4, 5, 6}.

Results:

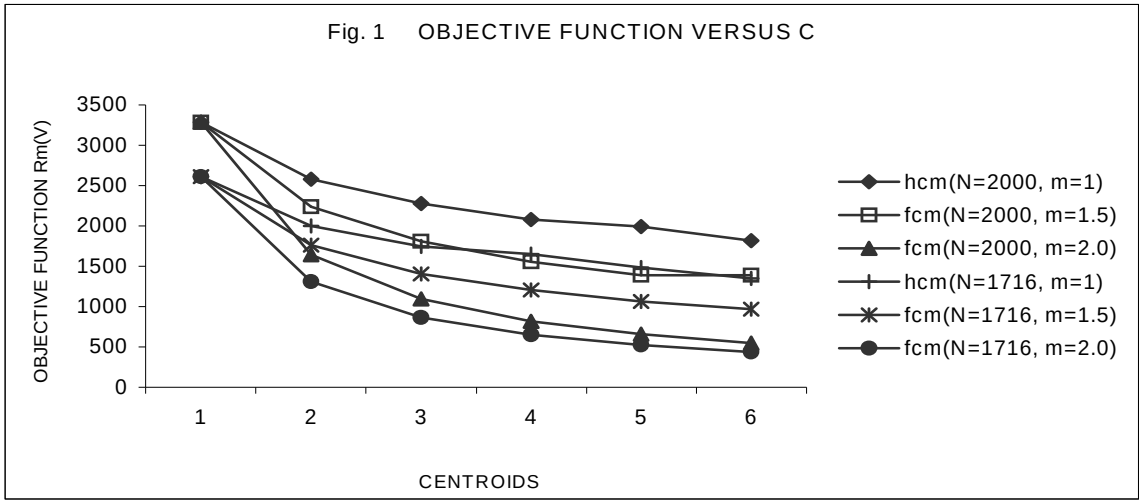
After depuration of classes with less than 20 data to an α level equal to 0.800, the number of data fell from n = 2000 to *n'* = 1716 depured data. Table 1 shows the objective function before and after depuration. These values were plotted as function of the number of clusters; such curves are shown in figure 1.

In the three columns that show ratios of objective functions (R_m(1716)/ R_m(2000)) one can appreciate the relative gains obtained with depuration. Comparing these values to the relative amount of remaining data: 1716/2000 = 0.858 one can conclude that in all cases, there is gain not less than 6,29 %.

CLASSES	Rm		Rm(1716)/ Rm(2000)	Rm		Rm(1716)/ Rm(2000)	Rm		Rm(1716)/ Rm(2000)
C	FCM N=1716 m=2.0	FCM N=2000 m=2.0		FCM N=1716 m=1.5	FCM N=2000 m=1.5		HCM N=1716 m=1	HCM N=2000 m=1	
1	2611.37793	3284.38599	0.79509	2611.37793	3284.38599	0.79509	2611.37793	3284.38599	0.79509
2	1305.68506	1642.19214	0.79509	1760.34705	2240.12427	0.78583	2001.88696	2577.9729	0.77654
3	868.354675	1094.79456	0.79317	1408.27417	1806.38074	0.77961	1742.86304	2279.17798	0.76469
4	652.842529	821.091675	0.79509	1204.25012	1555.86182	0.77401	1650.30066	2080.76978	0.79313
5	520.302673	656.876709	0.79209	1065.15515	1385.37195	0.76886	1481.51685	1988.79163	0.74493
6	435.226807	547.395081	0.79509	965.527588	1259.49756	0.76660	1347.32446	1817.5	0.74131

TablE 1. OPTIMIZED OBJECTIVE FUNCTION VERSUS NUMBER OF CLASSES

Fig. 1 OBJECTIVE FUNCTION VERSUS C

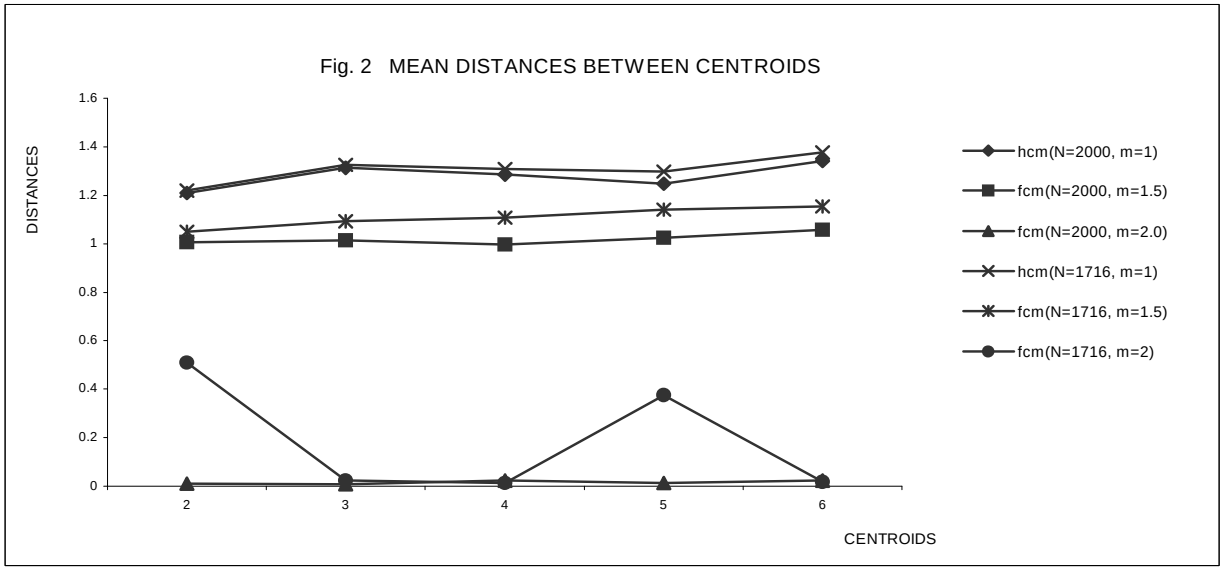


The improvement with the depured data can also be appreciated by analyzing the (center) intercluster distance to see whether they increased or not. Table 2 shows these data before and after pruning. Here again appears an appreciable improvement in the depured data structure. The brusque jumps for *m*=2 are because the centroids tend to converge towards the gravity center of the whole system as *m* increases. Even so, when dealing with depured data, for c=2 and 5, the centroids are still distinguishable.

MEAN DISTANCES BETWEEN DATA CENTERS						c
Fcm (N=1716, m=2.0)	Fcm (N=1716, m=1.5)	Hcm (N=1716, m=1)	Fcm (N=2000, m=2.0)	Fcm (N=2000, m=1.5)	Hcm (N=2000, m=1)	
0.509541	1.049221	1.218446	0.011008	1.006381	1.209035	2
0.023163667	1.093870667	1.327383667	0.006606	1.014772667	1.312597667	3
0.011550667	1.107226833	1.307179333	0.022484833	0.996500667	1.284681167	4
0.375676	1.1410182	1.297077167	0.0133548	1.024883	1.2474199	5
0.018964133	1.153019867	1.377146733	0.022904467	1.056879733	1.342468867	6

Tabla. 2 MEAN DISTANCES BETWEEN CENTROIDS VERSUS NUMBER OF CLASSES

Fig. 2 MEAN DISTANCES BETWEEN CENTROIDS



Optimal c value proceeds from data shown in tables 3 and 4. Table 3 shows the natural logarithms of R_m before pruning, with 2000 data; and after pruning, with 1716 data. Table 4 however show the relative errors calculated with (20). There one can note in all cases, the maximum value with respect to the best representative curve drawn by linear regression with least squares is obtained when c = 3, indicating that three is the optimum value of c. It is interesting to verify that

among all α values from the spectrum obtained, in the interval (0, 1), the greatest jump happens between $\alpha_3 = 0.791544$ and $\alpha_2 = 0.725409$, and the number of classes induced for an α-cut in the interval (0.725409, 0.791544] over *X*, ¡turns out to be three!

NATURAL LOGARITHMS OF Rm						c
Fcm(N=1716, m=2.0)	Fcm(N=1716, m=1.5)	Hcm(N=1716, m=1)	Fcm(N=2000, m=2.0)	Fcm(N=2000, m=1.5)	Hcm(N=2000, m=1)	
7.8676333	7.8676333	7.8676333	8.096935	8.096935	8.096935	1
7.17448313	7.47326625	7.6018455	7.4037873	7.71428662	7.85475867	2
6.76660024	7.25012024	7.46328446	6.998322	7.49908053	7.73157012	3
6.48133595	7.09361235	7.40871277	6.71063477	7.34978489	7.64049319	4
6.25441071	6.97087575	7.30082174	6.48749634	7.23372394	7.59528251	5
6.07586729	6.87267468	7.20587603	6.30517081	7.13846816	7.50521721	6

Table 3 OBJECTIVE FUNCTION NATURAL LOGARITHMS VERSUS NUMBER OF CLASSES

DEVIATIONS OF LN(RM) WITH RESPECT TO THE ADJUSTMENT VERSUS C						c
fcm(N=1716, m=2.0)	fcm(N=1716, m=1.5)	hcm(N=1716, m=1)	fcm(N=2000, m=2.0)	fcm(N=2000, m=1.5)	hcm(N=2000, m=1)	
0.00490882	0.00297064	0.00333274	0.00484857	0.00288372	0.00277708	1
0.04164153	0.02369716	0.01525417	0.04035615	0.02242803	0.01385772	2
0.0527268	0.02878357	0.01769208	0.05074232	0.02721679	0.01580916	3
0.04689882	0.02497572	0.00897467	0.04538686	0.02348052	0.01367035	4
0.03141711	0.01619785	0.00721874	0.0298295	0.01503335	0.00546449	5
0.00628562	0.00337917	0.00361361	0.00615821	0.0032509	0.00297883	6

Tabla 4 RELATIVE ERRORS OF NATURAL LOGARITHMS OF Rm VERSUS NUMBER OF CLASSES

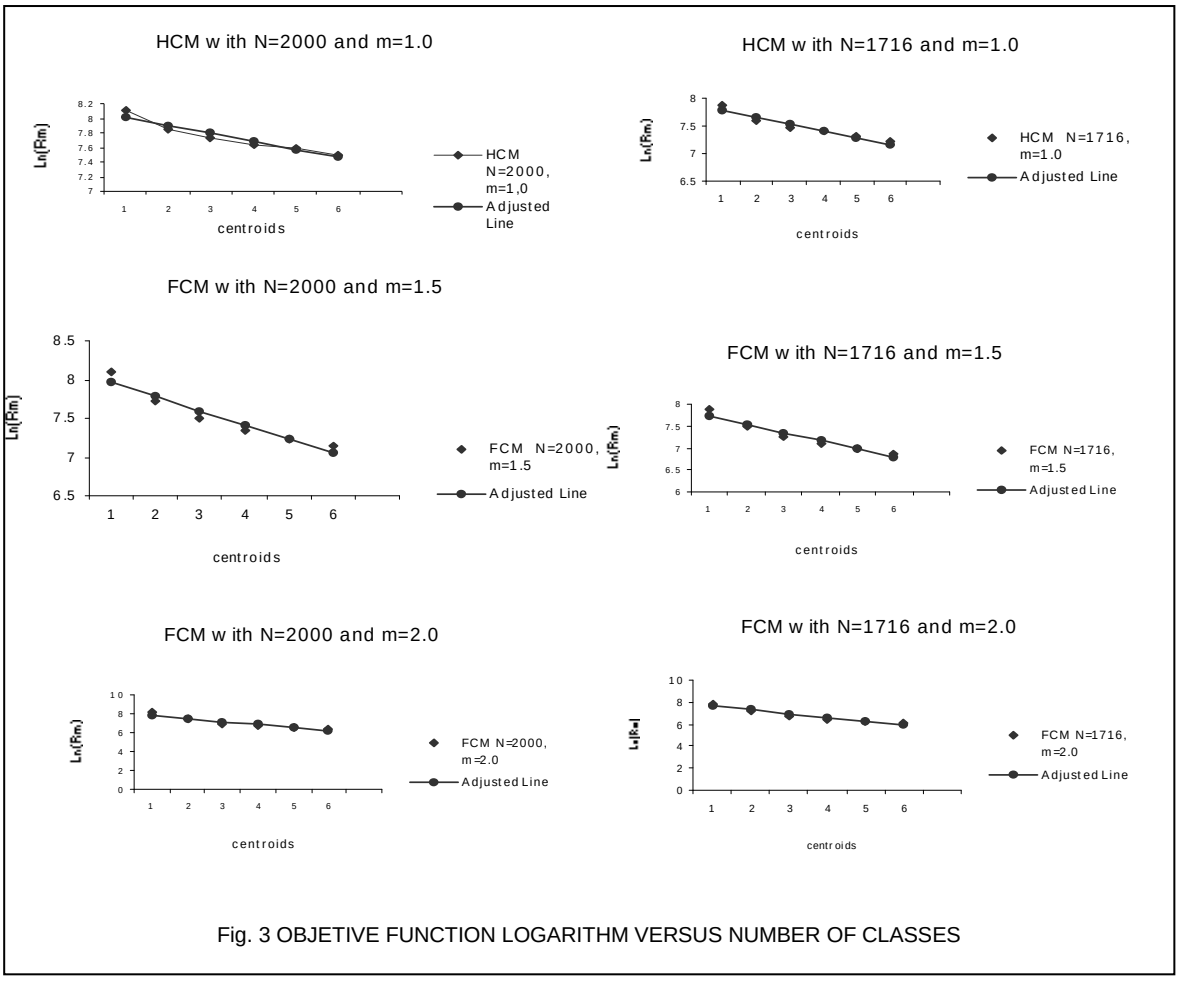


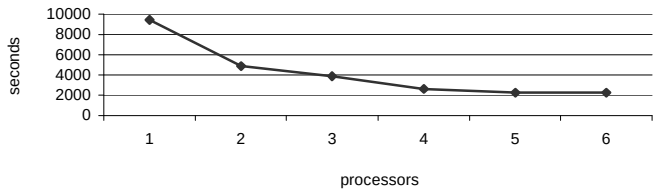
Fig. 3 OBJETIVE FUNCTION LOGARITHM VERSUS NUMBER OF CLASSES

The speedup for the parallelized version is shown in table 5. Up to six processors were tried, when the additional time gain became absolutely marginal showing a poor scalability as communication time takes over. However, from one to two processors, there is a great reduction which approaches linear speedup. Figure 4 shows this trend.

PROCESSORS	TIME (seg)
1	9425
2	4877.5
3	3885
4	2620
5	2230
6	2224

Table 5 TIME VERSUS PROCESSORS

Fig 4. TIME UNTIL TRANSITIVE CLOSURE



6 CONCLUSION

The example presented verifies the applicability of the method proposed. It was found that the optimal value for the number of paraguayan household classes (the economical classes) is three: high (rich) , medium and low (poor) classes.

The result with real data allows to affirm that the proposed method inded helps in getting additional information on the sample topology, that ameliorates the classification obtained. And the price paid for it although may vary, can be considered quite affordable.

The applicability scope may be very vast. So this work shows a way to consider for cluster analysis and pattern recognition applied to more complex problems.

REFERENCES

[1] Barán, B. et al, "Measurement and Analysis of Poverty and Welfare Using Fuzzy Sets", ", SCI'99/ISAS'99. Orlando, 1999.

[2] Bezdek, J. C., *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.

[3] Bezdek, J. C., "A Physical Interpretation of Fuzzy ISODATA", *IEEE Trans. Syst. Man Cybern.*, vol. SMC-6, pp. 387-389, 1976.

[4] Chapra, S. C., and Canale, R. P., *Métodos Numéricos para Ingenieros*, McGraw-Hill, México, 1988.

[5] El-Sonbaty, Y., and Ismail, M. A., "Fuzzy Symbolic K-Means", SCI'98/ISAS'98. Orlando, 1998.

[6] Hall, L. O., Özyurt, I. B., and Bezdek, J. C., "Clustering with a Genetically Optimized Approach", *IEEE Trans. Evol. Comput.*, vol. 3, NO. 2, pp. 103-112, 1999.

[7] Hathaway, R. J. and Bezdek, J. C., "Optimization of Clustering Criteria by Reformulation", *IEEE Trans. Fuzzy Syst.*, vol. 3, NO. 2, pp. 241-245, 1995.

[8] Klir, G. J. and Yuan B., *Fuzzy Sets and Fuzzy Logic. Theory and Applications*. Prentice Hall PTR, Upper Saddle River, 1995.

[9] Klir, G. J., St.Clair, U. H. and Yuan B., *Fuzzy Set Theory. Foundations and Applications*. Prentice Hall PTR, Upper Saddle River, 1997.

[10] Sirgany, W., "An Introduction to the Art and Mathematics of Cluster Analysis", available at http://bass.gmu.edu/~mazel/cluster/clus.htm