axioms

MDPI

Article

A New Criterion for Improving Convergence of Fuzzy C-Means Clustering

Joaquín Pérez-Ortega [1,*], Carlos Fernando Moreno-Calderón [1], Sandra Silvia Roblero-Aguilar [2], Nelva Nely Almanza-Ortega [3], Juan Frausto-Solís [4,*], Rodolfo Pazos-Rangel [4] and José María Rodríguez-Lelis [1]

1 Tecnológico Nacional de México/Cenidet, Cuernavaca 62490, Mexico; d19ce059@cenidet.tecnm.mx (C.F.M.-C.); jose.rl@cenidet.tecnm.mx (J.M.R.-L.)
2 Tecnológico Nacional de México/IT Tlalnepantla, Tlalnepantla 54070, Mexico; sandra.ra@tlalnepantla.tecnm.mx
3 Consejo Nacional de Humanidades, Ciencia y Tecnología/Conahcyt, Ciudad de México 03100, Mexico; nelva.almanza@conahcyt.mx
4 Tecnológico Nacional de México/IT Cd. Madero, Madero 89440, Mexico; rodolfo.pr@cdmadero.tecnm.mx
* Correspondence: joaquin.po@cenidet.tecnm.mx (J.P.-O.); juan.frausto@itcm.edu.mx (J.F.-S.)

Abstract: One of the most used algorithms to solve the fuzzy clustering problem is Fuzzy C-Means; however, one of its main limitations is its high computational complexity. It is known that the efficiency of an algorithm depends, among other factors, on the strategies for its initialization and convergence. In this research, a new convergence strategy is proposed, which is based on the difference of the objective function values, in two consecutive iterations, expressed as a percentage of its value in the next to the last one. Additionally, a new method is proposed to optimize the selection of values of the convergence or stop threshold of the algorithm, which is based on the Pareto principle. To validate our approach, a collection of real datasets was solved, and a significant reduction in the number of iterations was observed, without affecting significantly the solution quality. Based on the proposed method and the experiments carried out, we found it is convenient to use threshold values equal to 0.73 and 0.35 if a decrease in the number of iterations of approximately 75.2% and 64.56%, respectively, is wanted, at the expense of a reduction in solution quality of 2% and 1%, respectively. It is worth mentioning that, as the size of the datasets is increased, the proposed approach tends to obtain better results, and therefore, its use is suggested for datasets found in Big Data and Data Science.

Keywords: big data; clustering algorithm; convergence; data science; Fuzzy C-Means

MSC: 62H30; 68W40; 90C70; 91C20

Citation: Pérez-Ortega, J.; Moreno-Calderón, C.F.; Roblero-Aguilar, S.S.; Almanza-Ortega, N.N.; Frausto-Solís, J.; Pazos-Rangel, R.; Rodríguez-Lelis, J.M. A New Criterion for Improving Convergence of Fuzzy C-Means Clustering. *Axioms* **2024**, *13*, 35. https://doi.org/10.3390/axioms13010035

Academic Editors: Joao Paulo Carvalho and Hsien-Chung Wu

Received: 28 November 2023
Revised: 25 December 2023
Accepted: 28 December 2023
Published: 2 January 2024

1. Introduction

Technological development has generated an exponential increase in the generation and storage of data at public and private institutions. It is known that, in those large data repositories, potentially useful information for institutions might be distilled. Therefore, there exists a well justified interest in the extraction of that useful knowledge from those large volumes of data. The knowledge obtained in this way could help make better decisions or improve our comprehension of the environment. Some disciplines that deal with this problem are Data Science, Data Mining, and Data Analytics [1,2]. In turn, these disciplines rely on several techniques; one of them is clustering algorithms.

Data clustering algorithms have been used in different areas. Some of them are the following: pattern recognition, image segmentation, data mining, medicine, taxonomy, and business, among others [3,4]. However, one of the limitations of data clustering is its high computational cost [5,6].

Traditionally, clustering algorithms are divided into hierarchical and partitional [7]. In turn, partitional algorithms are divided into two: hard and soft [8–10]; their main exponents are K-Means [11] and Fuzzy C-Means (FCM) [12], respectively. For the rest of this paper, when we speak of the FCM algorithm, we mean the Bezdek algorithm proposed in [13].

The main characteristic of K-Means is that it generates hard partitions (i.e., each object belongs to a single cluster); in contrast, FCM generates a fuzzy partition, where objects have some degree of membership in each cluster. For this reason, one of the advantages of FCM over K-Means is the quality of results when the interpretation of the results has a gradated interpretation, which is related to the fuzzy partition concept. However, the computational complexity of FCM is higher than that of K-Means. Article [14] mentions that the time complexity of FCM is $O(ndc^2t)$, where n is the number of objects, d is the number of dimensions or objects attributes, c is the number of clusters, and t is the number of iterations. The clustering problem can be described as follows: let $X = \{x_1, \ldots, x_n\}$ be the dataset of n objects to be partitioned, where $x_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$, and c is the number of clusters where $2 \leq c \leq n$.

The fuzzy clustering problem was formulated initially as an optimization problem that consisted of the minimization of an objective function [15,16]. Later, this formulation was improved by Dunn and Bezdek. Next, the formulation by Bezdek [12] is presented. Expression (1) shows the objective function. The decision variables are the matrix of membership degrees U and the set of centroids V.

$$P \ : minimize\ J_m\ (\mathcal{U}, \mathcal{V}) = \sum_{i=1}^{\mathrm{n}} \sum_{j=1}^{c} u_{ij}{}^{\mathrm{m}}\ (x_i - v_j)^2 \tag{1}$$

subject to

$$\mathcal{U}_{ij} \in\ [0,1]; 1 \leq i \leq n,\ 1 \leq j \leq c, \tag{2}$$

$$\sum_{j=1}^{c} \mathcal{U}_{ij} = 1, \quad 1 \leq i \leq n, \tag{3}$$

$$\mathcal{U}_{ij} = 1 \quad if x_i = v_j, \tag{4}$$

$$0 < \sum_{i=1}^{n} \mathcal{U}_{ij} < n, \quad 1 \leq j \leq c, \tag{5}$$

Notation

In this section, the notation needed in this paper is defined:

m: weighting exponent or fuzzy factor, $m > 1$;
c: number of clusters in X;
n: number of objects in X;
ε: threshold value;
i: index of objects;
j: index of cluster;
t: iteration count;
T: maximum number of iterations;
$X = \{x_1, \ldots, x_n\}$: set of n objects to be partitioned according to a similarity criterion;
$U = \{u_{ij}\}$: membership degree of object i to cluster j;
$V = \{v_1, \ldots, v_c\}$: set of centroids, where v_j is the centroid of cluster j;
$\| x_i - v_j \|^2$: distance from object x_i to centroid v_j using the Euclidean distance;
ΔU: convergence criterion (the difference of the membership degrees values in two consecutive iterations);
ΔV: convergence criterion (the difference of the centroids values in two consecutive iterations);
$\Delta_{old} J_m$: convergence criterion (the difference of the objective function values in two consecutive iterations);
$\Delta_{new} J_m$: new convergence criterion (the difference of the objective function values, in two consecutive iterations, expressed as a percentage of its value in the next to the last one).

Expression (2) indicates that the membership degree of object i to cluster j must be between 0 and 1. Expression (3) declares that the sum of the membership degrees of any

object i to all the clusters must equal 1. Expression (4) states that the membership degree of an object i to a cluster j must equal 1 if the object x_i lies at the same position of centroid v_j. Expression (5) declares that for each cluster j, the sum of the membership degrees of all the objects to cluster j must be greater than 0 and less than n (i.e., there may not be any centroid v_j such that the sum of the membership degrees of all the objects to v_j equals 0).

The solution method mentioned by Bezdek in [12] consists of performing the calculation of Expressions (6) and (7) alternately, until a convergence criterion is satisfied.

$$v_j = \frac{\sum_{i=1}^{n} \left(u_{ij}\right)^m x_i}{\sum_{i=1}^{n} \left(u_{ij}\right)^m}, \quad 1 \leq j \leq c, \tag{6}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left(x_i - v_j\right)^2}{\left(x_i - v_k\right)^2} \right)^{2/(m-1)}}, \quad 1 \leq i \leq n,\ 1 \leq j \leq c. \tag{7}$$

Expressions (8) and (9) show two convergence criteria [12,13], whose values are used to stop the algorithm when a stop condition is met. Usually, the threshold values are denoted by symbol ε.

$$\Delta U^{(t)} = \max \left(abs\left({u_{ij}}^{(t-1)} - {u_{ij}}^{(t)} \right) \right) \tag{8}$$

$$\Delta V^{(t)} = \max \left(abs\left({v_{ij}}^{(t-1)} - {v_{ij}}^{(t)} \right) \right) \tag{9}$$

The pseudocode of the FCM Algorithm 1 described next is based on article [13].

Algorithm 1: FCM

Input: dataset X, c, m, ε, T
Output: V, U
1 **Initialization:**
2 t: = 0;
3 ε: = 0.01;
4 $U^{(t)}$: = $\{u_{11}, \ldots, u_{ij}\}$ is randomly generated;
5 **Calculate centroids:**
6 Calculate the centroids using Equation (6);
7 **Calculate membership matrix:**
8 Update and calculate the membership matrix using Equation (7);
9 **Convergence:**
10 **If** $\Delta U^{(t)} < \varepsilon$ or $t = T$:
11 Stop the algorithm;
12 **Otherwise:**
13 $U^{(t)}$: = $U^{(t+1)}$ and t: = $t + 1$;
14 Go to **Calculate centroids**
15 **End of algorithm**

According to Algorithm 1, the input parameters are five: dataset X, the number of clusters c, the weighting exponent m, the threshold value ε, and the maximum number of iterations T. The output parameters or decision variables are the matrix of membership degrees U and the set of centroids V. In the initialization phase, the iteration count t is set to 0, the threshold value ε is set to 0.01, and the initial matrix of membership degrees is randomly generated. The cycle starts at line 5, where centroids are calculated, and at line 7, the calculation of the membership matrix is performed. Notice that line 10 defines a convergence criterion constituted of two stop conditions, such that the algorithm stops when any one is met.

It is known that the efficiency of the FCM algorithm depends, among other factors, on the initialization and convergence strategies. In this research, a new convergence strategy is proposed, which is based on a new indicator defined by the difference of the objective

function values, in two consecutive iterations, expressed as a percentage of its value in the next to the last one. Additionally, a new method is proposed to optimize the selection of values for the stop threshold based on the Pareto principle.

This paper is organized as follows: Section 2 presents related work. Section 3 shows the improvement proposal. Section 4 reports the results obtained. Finally, conclusions are presented in Section 5.

2. Related Work

The FCM algorithm consists of four phases: initialization, centroids calculation, membership matrix calculation, and convergence [13]. In this article, we will focus on the last phase.

2.1. Convergence

In the convergence phase of the FCM algorithm, the stop criterion may include one or two stop conditions. Four stop conditions have been considered: (i) a maximum number of iterations (T); (ii) the value of indicator ΔU is smaller than a threshold ε; (iii) the value of indicator ΔV is smaller than a threshold ε; and (iv) the value of indicator $\Delta_{old} J_m$ (see Expression (10)) is smaller than a threshold ε. In those cases where the convergence criterion consists of two conditions, the algorithm stops when one of them is met.

$$\Delta_{old} {J_m}^{(t)} = ({J_m}^{(t-1)} - {J_m}^{(t)}) < \varepsilon \tag{10}$$

Next, on a timeline, the main projects on the convergence phase are briefly described. In its origins, in 1969–1970, Ruspini used ΔU as convergence indicator [15,16]. However, he did not propose a value for threshold ε. It is important to mention that Ruspini was the first to formulate the fuzzy clustering problem as an optimization problem including an objective function. The solution method that he used was the gradient method. Later, in 1973, Dunn improved the Ruspini model with some changes in the objective function [17]. The solution method used by Dunn was the Lagrange multipliers, and the convergence indicator was ΔV. The value for the threshold was 0.00001. Following this timeline, Bezdek, in his PhD dissertation, continued Dunn's research [12]. In particular, he incorporated variable m in the term $(u_{ij})^m$, where m is the m-th power of the membership degree of the i-th object to cluster j.

Table 1 shows the different stop criteria proposed by Ruspini, Dunn, and Bezdek. The structure of Table 1 is the following: column one indicates the reference to the related publication, column two presents the author name, column three shows the publication year, column four presents the threshold values used, column five indicates if a maximum number of iterations (T) was utilized as a stop condition, column six specifies if the indicator ΔU was used, and column seven indicates if the convergence indicator ΔV was utilized.

Table 1. Threshold values proposed by Ruspini, Dunn, and Bezdek.

Ref.	Author	Year	ε	Convergence Criterion		
				T	ΔU	ΔV
[15,16]	Ruspini	1969–1970	—		✓	
[17]	Dunn	1973	0.00001			✓
[12]	Bezdek	1973	0.0001		✓	✓
[18,19]	Bezdek	1974	0.0001		✓	
[20]	Bezdek	1975	0.001		✓	✓
[21]	Bezdek	1976	0.01		✓	
[22]	Bezdek	1981	0.01		✓	
[13]	Bezdek	1984	0.01	✓	✓	
[23]	Bezdek	1986	0.001	✓	✓	

In subsequent years and according to the specialized literature, several threshold values were proposed: for example, 0.1 [24], 0.01 [13,22,23], 0.001 [5,10,23,25,26], 0.0001 [27], 0.00001 [28–30], 0.000001 [31,32], 0.000000001 [33], 0.03 [34]. In some publications, the authors mention that they obtained good results with the proposed threshold values. However, to the knowledge of the authors of this article, none of the proposals for threshold values presents quantitative arguments that relate the quality of the solution obtained to the threshold values. In this sense, this research proposes a new approach to quantitatively choose the best threshold values to obtain a good-quality solution.

2.2. Improvements of the FCM Algorithm That Modify the Objective Function

In the specialized literature, there exist several publications that propose FCM variants of the objective function. A generalization of the objective function is shown in [35]. Other changes in the objective function are oriented to assign importance weights or weighting factors, for example, to the Euclidean distance [36,37], objects and clusters [30,38,39], and attributes [30,40]. These variants include as the convergence indicator the difference of the objective function values in two consecutive iterations, which is denoted by $\Delta_{old}J_m$; see Expression (10). It is important to mention that, in the aforesaid variants, the stop criterion consists of two conditions: one is when a maximum number of iterations is reached, and the other is when $\Delta_{old}J_m$ is smaller than a threshold. Table 2 shows the improvements to the FCM algorithm that include the convergence indicator $\Delta_{old}J_m$. The structure of Table 2 is the following: column one indicates the reference to the publication, column two shows the publication year, and column three presents the threshold value ε.

Table 2. Improvements to the FCM algorithm that include the convergence indicator $\Delta_{old}J_m$.

Ref.	Year	ε
[36]	2004	—
[38]	2008	—
[37]	2008	—
[39]	2016	—
[30]	2019	0.00001
[40]	2023	0.00001

Notice that in Table 2, for publications [36–39], the value of parameter ε is not explicitly mentioned.

It is important to mention that, in our proposed convergence indicator $\Delta_{new}J_m$ (see Expression (11)), as well as for indicator $\Delta_{old}J_m$, the objective values for consecutive iterations are used. However, an important limitation of indicator $\Delta_{old}J_m$ is the difficulty selecting the adequate threshold values. Basically, to define an efficient threshold value, it is necessary to know the range of values of the objective function prior to the conclusion of the algorithm. In this sense, our proposal for indicator $\Delta_{new}J_m$ is independent of the range of values that the objective function may have, since the threshold value is defined as a percentage of the objective function value. A detailed description of our indicator is presented in the next section.

3. Proposal

Our improvement proposal for FCM consists of two parts: a new convergence indicator and an improved selection of the stop thresholds for the algorithm through the Pareto principle.

3.1. Proposal for a New Convergence Indicator

The FCM algorithm uses ΔU as convergence indicator, in particular, the largest change in the value of elements u_{ij} in two iterations. One of the limitations of this approach is that the indicator value is detached from the values of the objective function. This hinders the selection of threshold values that allow for providing a balance between the

number of iterations and the solution quality. In this sense, a new convergence indicator is proposed, which is based on the difference of the objective function values, in two consecutive iterations, expressed as a percentage of its value in the next to the last one; see Expression (11).

$$\Delta_{new}{J_m}^{(t)} = 100 * \frac{\left({J_m}^{(t-1)} - {J_m}^{(t)}\right)}{{J_m}^{(t-1)}} \tag{11}$$

In order to compare the values of indicators $\Delta_{new}{J_m}^{(t)}$ and $\Delta U^{(t)}$, a dataset was solved and, in general, we noticed that they were highly correlated, and in every case, the correlation was larger than 0.9. The correlation calculation was carried out using Expressions (12) and (13), which are related to the Pearson correlation. In particular, when solving the real *Abalone* dataset [41], a correlation of 0.94 was found. Such a dataset consists of 4177 objects with seven dimensions.

$$p(\Delta J_m, \Delta U) = \frac{\sum_{t=2}^{T}\left[\left(\Delta {J_m}^{(t)} - \Upsilon\right)\left(\Delta U^{(t)} - \nu\right)\right]}{\sqrt{\sum_{t=2}^{T}\left(\Delta {J_m}^{(t)} - \Upsilon\right)^2}\sqrt{\sum_{t=2}^{T}\left(\Delta U^{(t)} - \nu\right)^2}} = 0.94, \tag{12}$$

where:

$$\Upsilon = \frac{1}{t}\sum_{t=2}^{T}\Delta {J_m}^{(t)} \quad \text{and} \quad \upsilon = \frac{1}{t}\sum_{t=2}^{T}\Delta U^{(t)}. \tag{13}$$

3.2. Method to Determine Threshold Values

It is known that the Pareto principle [42–45] allows for determining the optimal relation between effort and benefit when solving a dataset [46,47]. In this case, the effort is measured by the number of iterations or computer time, and the benefit is the value of the objective function when the algorithm stops. To apply the Pareto principle, it is necessary to solve at least one dataset with the same characteristics as those of the dataset that it is intended to be solved. Once we have the intermediate results of each iteration, the final value of the objective function and the number of iterations, a series of steps are applied to obtain the optimal relation between the number of iterations and the solution quality. In particular, the selection of the optimal relationship is guided by the minimum distance between the current solution and the ideal solution (0 iterations, 100% efficiency). From the determination of the optimal relation, the threshold values are established. A detailed description on the use of the Pareto principle can be found in [46].

In order to illustrate the process we followed to calculate thresholds, the rest of this section shows the results from applying the Pareto principle to the results obtained from solving the real Abalone dataset, using FCM and 0.01 as the threshold. It is important to remark that we chose the value for $\varepsilon = 0.01$ because it is the least restrictive of the values proposed in Tables 1 and 2. More restrictive threshold values cause the algorithm to generate more iterations.

Table 3 shows the relation that exists between computational effort and solution quality effort and solution quality. Column one indicates the iteration count; column two shows the number of iterations carried out A^t (expressed as a percentage of the total number of iterations), where $A^t = 100 * (t/T)$; column three presents the reduction of the objective function value in two consecutive iterations, B^t, expressed as a percentage of the decrease of the objective function from the 1st iteration to the final one, where $B^t = 100 * ({J_m}^{(t-1)} - {J_m}^{(t)})/({J_m}^1 - {J_m}^{\mathrm{T}})$; column four shows the value of the Euclidean distance D^t between points (A^t, C^t) and (0, 100), where $D^t = \sqrt{(0 - A^t)^2 + (100 - C^t)^2}$; column five presents the cumulative value of the reduction percentages of the objective function values, where $C^t = C^{t-1} + B^t$; and column six shows the value of the convergence indicator $\Delta_{new}J_m$ for each iteration.

Table 3. Some intermediate and final results from FCM when solving Abalone.

Iteration (t)	A^t	B^t	D^t	C^t	$\Delta_{new}J_m{}^{(t)}$
13	4.45	0.59	5.81	96.25	2.36
14	**4.79**	**0.66**	**5.69**	**96.92**	**2.71**
15	5.13	0.56	5.72	97.48	2.34
...	...	...	...	...	...
17	5.82	0.21	6.14	98.03	0.90
...	...	...	...	...	...
29	9.93	0.03	9.98	99.01	0.15
...	...	...	...	...	...
292	100	0.00	100	100	0.0024

Figure 1 shows the Pareto diagram for points (A^t, C^t); see columns two and five of Table 3. The A-axis indicates the percentages of the total number of iterations, and the C-axis indicates the cumulative value of the reduction percentages of the objective function. Therefore, point G with coordinates (4.79, 96.92) means that, with 4.79% of iterations, 96.92% of the cumulative value of reductions of the objective function was attained. In case the algorithm is stopped at this point, there will only be a quality decrease of 3.08% of the objective function.

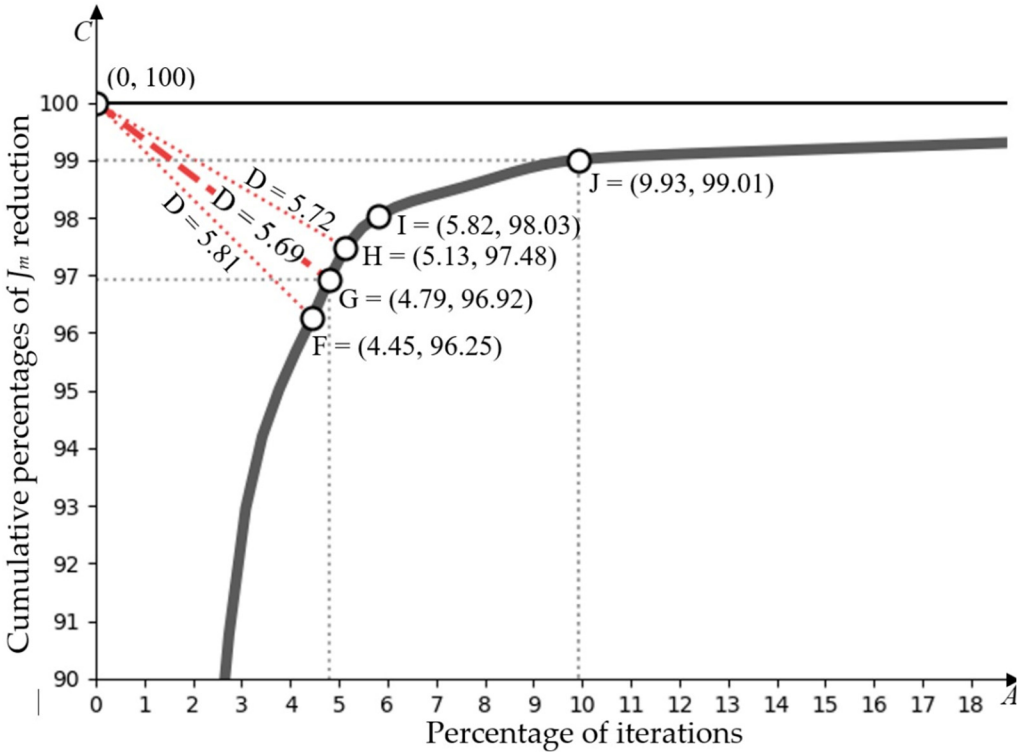

Figure 1. Pareto optimality for Abalone, ε = 0.01.

Notice that point G is the one with the smallest distance D^t to point (0, 100), located in the upper left corner of the diagram. According to the Pareto principle, the aforesaid point is the one with an optimal relation between effort and benefit. As can be seen in row two of Table 3, the coordinates of point G correspond to iteration 14 at the intersection with columns two and five. To determine the optimal value of the threshold, it is necessary to look at the crossing of row two and column six of the said table, and it is found that the value of $\Delta_{new}J_m{}^{(t)}$ is 2.71%. This means that, when the difference of the objective function in two consecutive iterations is less than 2.71% of the value of the objective function in the preceding iteration, the algorithm stops. It is worth mentioning that, if the algorithm stopped at this iteration, 95.21% of the iterations could be avoided, and 96.92% of the

cumulative value of the reduction percentages of the objective function would be obtained (see the intersection of row two and column five). The quality decrease of the objective function would only be 3.08%.

Notice that points F, G, H, I, and J have cumulative values of the reduction percentages of the objective function equal to 96.25, 96.92, 97.48, 98.03, and 99.01, respectively, and they correspond to iterations 13, 14, 15, 17, and 29, respectively. As can be observed in Figure 1, the relation between computational effort and solution quality is not linear. See also that, as one aims to increase the solution quality, the number of iterations required grows in a larger proportion.

Table 4 shows some intermediate results when solving the Abalone dataset using FCM in order to compare values of the solution quality and the computational effort required. The values in each row correspond to the iteration indicated in column one. Column one indicates the iteration number, column two shows the percentage of the total number of iterations that would be reduced if the algorithm stops in the current iteration, column three presents the cumulative value of the reduction percentages of the objective function up to this iteration, column four shows the percentage of quality decrease of the objective function in case the algorithm stops in this iteration, column five presents the value of the objective function calculated in the iteration, column six shows the value of the proposed convergence indicator for the iteration, and column seven presents the value of the traditional convergence indicator of FCM.

Table 4. Results of interest from FCM when solving *Abalone*.

Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new} J_m$	ΔU
1	99.65	--	--	28.20	--	0.99
--	--	--	--	--	--	--
14	**95.20**	**96.92**	**3.08**	**5.56**	**2.71**	**0.44**
15	94.86	97.48	2.52	5.43	2.34	0.42
--	--	--	--	--	--	--
17	94.18	98.03	1.96	5.30	0.90	0.22
--	--	--	--	--	--	--
29	90.07	99.01	0.99	5.07	0.15	0.06
--	--	--	--	--	--	--
292	0	100	0	4.84	0.0024	0.0094

The rows of Table 4 show the values for some iterations of interest. Notice that the values in column three are approximately 96, 97, 98, 99, and 100 percent and correspond to iterations 14, 15, 17, 29, and 292, respectively. As can be observed, for improving quality from 96 to 97 percent, only 1 iteration was needed; however, for improving from 99 to 100 percent, 263 iterations were required. The last case shows a possible inefficiency of the algorithm because a minimal improvement was obtained at the cost of a high computational cost. In contrast, according to the Pareto principle, the optimal relation between effort and benefit occurs in iteration 14. Therefore, the value 2.71 is suggested as the stop threshold using our proposed indicator (column six) instead of the value 0.44 when using the traditional convergence indicator of FCM (column seven). It is remarkable that the values of our indicator are more intuitive because they are defined as a percentage of the objective function value, which does not happen with the traditional indicator of FCM.

It is predictable that each application requires different degrees of solution quality and, therefore, different convergence thresholds. Thus, for example, if some user needs a solution quality of 97.5%, a value of 2.34% for the stop threshold is suggested for our indicator. With this value, there would be reductions of 94.86% in the number of iterations and only 2.52% in the solution quality.

Algorithm 2, shown next, integrates our proposed improvement to the FCM algorithm, which we call Pareto Fuzzy C-Means (P-FCM). In particular, we want to call attention to line 9, where the new convergence indicator is integrated.

Algorithm 2: P-FCM

Input: Dataset X, V, c, m, ε
Output: V, U
Initialization:
  t: = 0;
  $U^{(t)}$: = $\{u_{11}, \ldots, u_{ij}\}$ is randomly generated;
Calculate centroids:
  Calculate the centroids using Equation (6);
Calculate membership matrix:
  Update and calculate the membership matrix using Equation (7);
Convergence:
  If $\Delta_{new}J_m^{(t)} \leq \varepsilon$**:**
    Stop the algorithm;
  Otherwise:
    $U^{(t)}$: = $U^{(t+1)}$ and t: = $t + 1$;
    Go to **Classification**
End of algorithm

As a summary, this section shows the steps of the proposed method to obtain the thresholds for the FCM algorithm:

Figure 2 shows a diagram of the method to obtain the threshold.

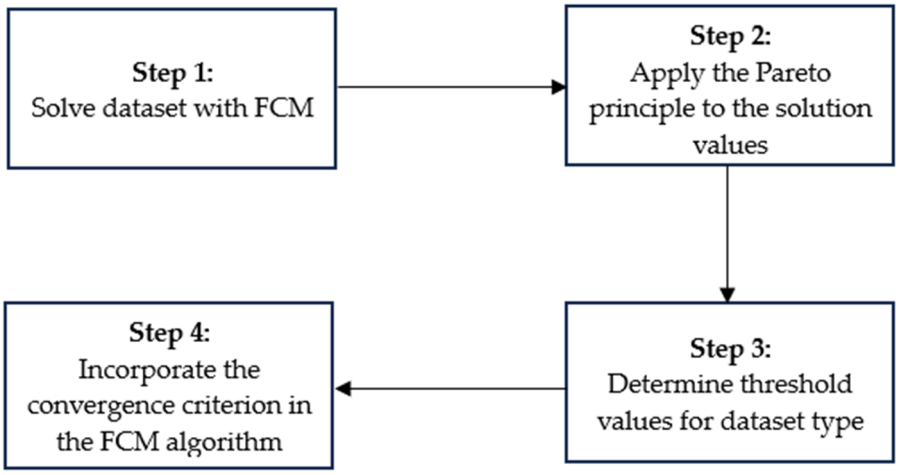

Figure 2. Diagram to obtain thresholds for FCM.

Step 1: Solve a dataset with parameter values similar to those of the types of datasets that will be solved.

Step 2: Apply the Pareto principle to the solution results to determine the effort–quality relationship. It is necessary to store the intermediate and final data of the algorithm execution.

Step 3: Determine threshold values for dataset type. If necessary, select those thresholds that meet the required solution quality.

Step 4: Incorporate the convergence criterion in the FCM algorithm and solve the datasets of the selected dataset type.

4. Results

For evaluating our proposed approach, three experiments were conducted, where three real datasets obtained from the UCI repository were used [41]. In the implementation of FCM, the computation and storage of the values of indicators $\Delta_{new}J_m$ and ΔU for each iteration were included in order to report their values at each iteration.

4.1. Experimental Environment

Algorithm FCM was implemented in the C language and compiled using GCC 7.4.0. The experiments were carried out on an Acer Nitro 5 computer with the Windows 11 operating system, an Intel® Core™ i7-11800H processor at 2.30 GHz, 16 GB of RAM, a 512 GB SSD, and a 1 TB HDD.

4.2. Description of the Datasets

The datasets were obtained from the UCI Machine Learning Repository [41]. Table 5 shows the characteristics of the datasets. The structure of Table 5 is the following: column one presents the name of the dataset; columns two and three show the quantity of data and dimensions, respectively; and column four shows the dataset size, which is the product of columns two and three.

Table 5. Real datasets.

Name	n	d	$n \times d$
Abalone	4177	7	29,239
Wine	4898	11	53,878
Urban	360,177	2	720,354

4.3. Test Cases

Test cases are designed for showing how different threshold values can generate different reductions in the number of iterations and solution quality.

The parameters used for executing the FCM algorithm were the same in all experiments. In particular, the values of $m = 2$ and $\varepsilon = 0.01$ were selected based on article [13], which mentions that they are reasonable values. The value of $c = 50$ was chosen because it is the highest reported in the literature [48]. Furthermore, because the focus of this research is aimed at solving large datasets, it is foreseeable that a larger number of clusters will also be required.

4.3.1. Description of Experiment I

The first experiment consisted of solving Abalone five times using FCM. Table 4 shows the execution from which the best efficiency was obtained.

In Table 4, notice that the optimal result, with respect to effort and solution quality, occurs in iteration 14 with a threshold value of 2.71%. With this value, a decrease of 95.20% in the number of iterations and a quality reduction of 3.08% were obtained.

4.3.2. Description of Experiment II

The second experiment consisted of solving Wine five times using FCM. Table 6 shows the execution from which the best efficiency was obtained.

Table 6 shows that the optimal result with respect to effort and solution quality occurs in iteration 15 with a threshold value of 1.77%. Notice that, for this value, a decrease of 89.78% in the number of iterations and a reduction of 5.98% of the solution quality were obtained.

4.3.3. Description of Experiment III

The third experiment consisted of solving Urban five times using FCM. Table 7 shows the execution from which the best efficiency was obtained.

Table 7 shows that the optimal result with respect to effort and solution quality occurs in iteration 17, where an 83% reduction in iterations and a 9.32% loss of quality were obtained. The value obtained for the threshold in this iteration was 5.51.

Table 6. Results of interest from FCM when solving Wine.

Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	99.27	--	--	207,288.90	--	0.15
--	--	--	--	--	--	--
15	**89.78**	**94.02**	**5.98**	**82,439.54**	**1.77**	**0.33**
--	--	--	--	--	--	--
16	88.33	95.77	4.23	80,114.034	1.31	0.34
17	87.59	96.44	3.55	79,223.43	1.11	0.29
--	--	--	--	--	--	--
19	86.13	97.36	2.64	77,997.14	0.67	0.27
--	--	--	--	--	--	--
22	83.94	98.17	1.83	76,925.08	0.39	0.16
--	--	--	--	--	--	--
27	80.29	99.01	0.99	75,819.27	0.26	0.12
--	--	--	--	--	--	--
137	0	100	0	74,500.37	0.0007	0.0098

Table 7. Results of interest from FCM when solving Urban.

Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	99	--	--	26,283.06	--	0.56
--	--	--	--	--	--	--
17	**83**	**90.69**	**9.32**	**7637.21**	**5.51**	**0.97**
18	82	91.49	8.51	7473.50	2.14	0.54
--	--	--	--	--	--	--
20	80	92.16	7.84	7335.26	0.66	0.23
--	--	--	--	--	--	--
22	78	93.01	6.99	7161.84	1.57	0.33
--	--	--	--	--	--	--
24	76	94.32	5.68	6891.88	1.33	0.28
--	--	--	--	--	--	--
26	74	95.50	4.49	6648.03	2.01	0.42
--	--	--	--	--	--	--
28	72	96.30	3.69	6483.56	1.1688	0.39
--	--	--	--	--	--	--
31	69	97.27	2.72	6284.52	1.0735	0.29
--	--	--	--	--	--	--
43	67	98.04	1.96	6127.57	0.15	0.32
--	--	--	--	--	--	--
73	27	99.26	0.73	5876.32	1.04	0.32
--	--	--	--	--	--	--
100	0	100	0	5724.92	0.0001	0.0087

4.4. Experiment Analysis

In this section, we analyze two types of threshold values. The first is the optimal threshold that is obtained by applying the Pareto principle. Tables 8–10 present the results obtained by solving five times the Abalone, Wine, and Urban datasets, respectively. Column one indicates the execution number, column two indicates the iteration number, column three shows the percentage of the total number of iterations that would be reduced if the algorithm stops in the current iteration, column four presents the cumulative value of the reduction percentages of the objective function up to this iteration, column five shows the percentage of quality decrease of the objective function in case the algorithm stops in this iteration, column six presents the value of the objective function calculated in the iteration, column seven shows the value of the proposed convergence indicator for the iteration, and column eight presents the value of the traditional convergence indicator of FCM. The second type of threshold is the one whose values generate a solution quality of

99%. Tables 11–13 show the results from solving s times the Abalone, Wine, and Urban datasets, respectively.

Table 8. Optimal threshold values from Abalone.

Execution	Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	11	93.03	96.18	3.82	5.70	4.46	0.73
2	14	95.21	96.92	3.08	5.56	2.71	0.44
3	15	95.57	96.52	3.48	5.61	2.03	0.48
4	14	95.64	97.04	2.96	5.52	2.33	0.67
5	16	95.69	96.98	3.02	5.54	1.90	0.62
		Average			5.58	2.68	0.58

Table 9. Optimal threshold values from *Wine*.

Execution	Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	14	89.79	94.02	5.98	82,439.54	1.77	0.33
2	13	89.17	92.75	7.25	84,076.47	2.33	0.38
3	10	87.35	88.94	11.06	89,294.37	2.57	0.43
4	10	86.31	89.22	10.78	88,774.58	2.73	0.44
5	13	88.89	93.31	6.69	83,474.63	2.04	0.35
		Average			85,611.91	2.28	0.38

Table 10. Optimal threshold values from *Urban*.

Execution	Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	17	83.00	90.69	9.31	7637.21	5.51	0.97
2	15	83.15	88.81	11.19	8518.01	8.66	0.77
3	14	77.05	84.30	15.70	9100.69	10.29	0.79
4	15	77.28	87.70	12.30	8421.65	8.40	0.81
5	16	80.25	90.79	9.21	7598.11	6.30	0.51
		Average			8255.13	7.83	0.77

Table 11. Threshold values from Abalone for a quality of 99%.

Execution	Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	17	89.25	99.06	0.94	5.04	0.82	0.15
2	29	90.07	99.01	0.99	5.07	0.15	0.06
3	52	84.62	99.01	0.99	5.06	0.13	0.08
4	31	**90.35**	99.01	0.99	5.06	0.16	0.11
5	51	86.29	99.02	0.98	5.06	0.14	0.09
		Average			5.05	0.28	0.09

Table 12. Threshold values from *Wine* for a quality of 99%.

Execution	Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	27	80.29	99.01	0.99	75,819.27	0.25	0.12
2	29	75.84	99.01	0.99	75,739.98	0.21	0.12
3	24	69.63	99.04	0.96	75,860.19	0.44	0.16
4	26	64.39	99.03	0.97	75,734.01	0.34	0.13
5	25	78.64	99.01	0.99	75,903.72	0.26	0.12
Average					75,811.43	0.30	0.13

Table 13. Threshold values from *Urban* for a quality of 99%.

Execution	Iteration	% Reduction Iterations	C^t	% Reduction Quality	J_m	$\Delta_{new}J_m$	ΔU
1	73	27.00	99.26	0.74	5876.32	1.04	0.32
2	63	29.22	99.04	0.96	6472.90	0.14	0.24
3	39	36.07	99.12	0.88	6079.84	0.48	0.52
4	51	**22.73**	99.13	0.87	6094.44	0.60	0.33
5	50	38.28	99.01	0.99	5906.22	0.23	0.09
Average					6085.94	0.49	0.30

As can be observed in Tables 8 and 9, $\Delta_{new}J_m$ has similar average values. The average for dataset Abalone is 2.68 and for dataset Wine is 2.28. However, this value is very different for Urban, which has an average of 7.83.

The values of $\Delta_{new}J_m$ are a good indicator for stopping the algorithm because its behavior with the Pareto principle shows that it is convenient to stop the algorithm in the first iterations. This reduces the computational effort, and additionally, it obtains a good result for the objective function. In the best case, a quality decrease of 2.96% was obtained, and in the worst case, the quality reduction was 15.70%. Furthermore, a decrease of up to 95.69% of the iterations was observed in the best case and a reduction of 77.05% in the worst case.

Column seven of Tables 11–13 shows the threshold values that deliver more than 99% of the solution quality. Notice that the largest reduction in the number of iterations was 90.35% in the fourth execution for *Abalone*, and the smallest decease was 22.73% in the fourth execution for *Urban*.

5. Conclusions

In this article, the problem addressed is how to increase the efficiency of the FCM algorithm by modifying the convergence phase. The novel contributions of the proposed approach is described in the following three paragraphs.

The first is a new convergence indicator based on the values of the objective function in two consecutive iterations. It is known that, since the pioneering works on FCM, a convergence indicator has been used that is based on differences of the membership matrix in two consecutive iterations. We carried out numerous correlation calculations between the values of both indicators, and the correlation turned out to be very high in every case; therefore, it is possible to substitute the original indicator with the one we propose without affecting the algorithm behavior. Obviously, the threshold values of our indicator are different, though equivalent to those of the original indicator. In this way, we suggest using a threshold of 0.90 to obtain an approximate value of 98% of the solution quality and a 94.18% reduction in iterations, or a threshold of 0.15 for an approximate value of 99% of the solution quality and a 90.07% decrease in iterations. Though it is possible to use any of the two indicators (new or traditional) in FCM, our proposal has two advantages. The first one is that it relates the value of the indicator to the values of the objective function, thus

intuitively allowing for performing adjustments of the threshold values. The second one is that, because the indicator is expressed as a percentage of change, it is independent of the value scale that the objective function may have during the executions of the algorithm.

The second is an innovative method to select stop threshold values that allows relating the solution quality to the computational effort. Virtually since the origin of the Fuzzy C-Means algorithm, 0.00001 and 0.01 were proposed as threshold values; however, no quantitative arguments were presented on the efficiency of the algorithm with these values. For understanding the behavior of the FCM algorithm, three datasets were solved using a threshold of 0.01. In general, we found that the algorithm has inefficient behavior in more than half of the iterations: for reducing the value of the objective function, each time it needs a larger number of iterations. Thus, for example, in some executions of the algorithm, it achieved 96.92% of the solution quality in iteration 14, 99% of the quality in iteration 29, and 100% of the quality in iteration 292. In this example, if the algorithm would have been stopped in iteration 29, the quality would have been reduced by 1%; however, the iterations would have been reduced by 90.07%. As an aspect of the analysis of the experimental results, it was observed that, generally, the value 0.01 is over-dimensioned and negatively affects the algorithm efficiency. In this sense, we propose new general-purpose threshold values. We suggest using a threshold value of 0.22 if a value of approximately 98% of the solution quality and an approximately 94.18% reduction in iterations are wanted, or a threshold value of 0.06 if a quality of 99% and a 90.07% decrease in iterations are needed.

Another contribution worth mentioning is the method for obtaining the threshold values based on the Pareto principle. This approach quantitatively allows for relating the computational effort to the solution quality in such a way that their optimal relation can be obtained.

To continue the line of research of this article, the authors consider the use of variants of the weighted Euclidean distance for different values of parameter weights as a direction for future research.

Finally, we consider that our proposal contributes to the convergence phase of FCM and has no conflict with improvements in its other phases, which allows its integration in other variants of the algorithm.

Author Contributions: Conceptualization, J.P.-O., C.F.M.-C. and S.S.R.-A.; methodology, J.P.-O., N.N.A.-O. and C.F.M.-C.; software, C.F.M.-C. and S.S.R.-A.; validation, J.P.-O., C.F.M.-C., S.S.R.-A., N.N.A.-O. and J.M.R.-L.; formal analysis, J.P.-O., J.M.R.-L. and R.P.-R.; investigation, J.P.-O., C.F.M.-C. and S.S.R.-A.; resources, S.S.R.-A., N.N.A.-O., J.M.R.-L. and J.F.-S.; data curation, S.S.R.-A., N.N.A.-O., R.P.-R. and C.F.M.-C.; writing—original draft preparation, J.P.-O., C.F.M.-C. and S.S.R.-A.; writing—review and editing, J.P.-O., C.F.M.-C., S.S.R.-A., N.N.A.-O., J.F.-S. and R.P.-R.; visualization, C.F.M.-C.; supervision, J.P.-O.; project administration, J.P.-O.; funding acquisition, J.P.-O., J.F.-S. and J.M.R.-L. All authors have read and agreed to the published version of the manuscript.

Funding: The student Carlos Fernando Moreno Calderón acknowledges the scholarship (grantee No. 1000864) given to him by the Consejo Nacional de Humanidades, Ciencia y Tecnología, Mexico.

Data Availability Statement: The real datasets used were obtained from the UCI machine learning repository at https://archive.ics.uci.edu/ml/index.php (accessed on 22 October 2023).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ajin, V.W.; Kumar, L.D. Big Data and Clustering Algorithms. In Proceedings of the 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), Bangalore, India, 6–7 May 2016.
2. Giordani, P.; Ferraro, M.B.; Martella, F. Big data and clustering. In *An Introduction to Clustering with R*; Springer: Singapore, 2020; pp. 111–121.
3. Nayak, J.; Naik, B.; Behera, H.S. Fuzzy C-Means (FCM) Clustering Algorithm: A Decade Review from 2000 to 2014. In Proceedings of the Computational Intelligence in Data Mining, New Delhi, India, 20–21 December 2014.
4. Shukla, A.K.; Muhuri, P.K. Big-data clustering with interval type-2 fuzzy uncertainty modeling in gene expression datasets. *Eng. Appl. Artif. Intell.* **2019**, *77*, 268–282. [CrossRef]

5. Pérez, J.; Roblero, S.S.; Almanza, N.N.; Solís, J.F.; Zavala, C.; Hernández, Y.; Landero, V. Hybrid Fuzzy C-Means clustering algorithm oriented to big data realms. *Axioms* **2022**, *11*, 377. [CrossRef]
6. Pérez, J.; Rey, C.D.; Roblero, S.S.; Almanza, N.N.; Zavala, C.; García, S.; Landero, V. POFCM: A parallel fuzzy clustering algorithm for large datasets. *Mathematics* **2023**, *11*, 1920. [CrossRef]
7. Ezugwu, A.E.; Ikotun, A.M.; Oyelade, O.O.; Abualigah, L.; Agushaka, J.O.; Eke, C.I.; Akinyelu, A.A. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Eng. Appl. Artif. Intell.* **2022**, *110*, 104743. [CrossRef]
8. Miyamoto, S.; Ichihashi, H.; Honda, K. *Algorithms for Fuzzy Clustering Methods in C-Means Clustering with Applications*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 229.
9. Atiyah, I.; Mohammadpour, A.; Taheri, S. KC-Means: A fast fuzzy clustering. *Hindawi Adv. Fuzzy Syst.* **2018**, *2018*, 34861. [CrossRef]
10. Bezdek, J.C. *Elementary Cluster Analysis: Four Basic Methods That (Usually) Work*; River Publishers: Gistrup, Denmark, 2022.
11. MacQueen, J. Some Methods for Classification and Analysis of Multivariate Observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 21 June–18 July 1965.
12. Bezdek, J.C. Fuzzy Mathematics in Pattern Classification. Ph.D. Thesis, Cornell University, Ithaca, NY, USA, 1973.
13. Bezdek, J.C.; Ehrlich, R.; Full, W.E. FCM: The Fuzzy C-Means clustering algorithm. *Comput. Geosci.* **1984**, *10*, 191–203. [CrossRef]
14. Ghosh, S.; Kumar, S. Comparative analysis of K-Means and Fuzzy C-Means algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2013**, *4*, 35–39. [CrossRef]
15. Ruspini, E.H. A new approach to clustering. *Inf. Control* **1969**, *15*, 22–32. [CrossRef]
16. Ruspini, E.H. Numerical methods for fuzzy clustering. *Inf. Sci.* **1970**, *2*, 319–350. [CrossRef]
17. Dunn, J.C. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *J. Cybern.* **1973**, *3*, 32–57. [CrossRef]
18. Bezdek, J.C. Numerical taxonomy with fuzzy sets. *J. Math. Biol.* **1974**, *1*, 57–71. [CrossRef]
19. Bezdek, J.C. Cluster validity with fuzzy sets. *J. Cybern.* **1974**, *3*, 58–73. [CrossRef]
20. Bezdek, J.C.; Dunn, J. Optimal fuzzy partitions: A heuristic for estimating the parameters in a mixture of normal distributions. *IEEE Trans. Comput.* **1975**, *24*, 835–838. [CrossRef]
21. Bezdek, J.C. Feature Selection for Binary Data: Medical Diagnosis with Fuzzy Sets. In Proceedings of the AFIPS '76: National Computer Conference and Exposition, New York, NY, USA, 7–10 June 1976.
22. Bezdek, J.C. *Pattern Recognition with Fuzzy Objective Function Algorithms*; Plenum Press: New York, NY, USA, 1981.
23. Cannon, R.L.; Dave, J.V.; Bezdek, J.C. Efficient implementation of the Fuzzy C-Means clustering algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 248–255. [CrossRef] [PubMed]
24. Song, X.; Shi, M.; Wu, J.; Sun, W. A new Fuzzy C-Means clustering-based time series segmentation approach and its application on tunnel boring machine analysis. *Mech. Syst. Signal Process.* **2019**, *133*, 106279. [CrossRef]
25. Ramze, M.; Lelieveldt, B.P.F.; Reiber, J.H.C. A new cluster validity index for the Fuzzy C-Mean. *Pattern Recognit. Lett.* **1998**, *19*, 237–246. [CrossRef]
26. Shirkhorshidi, A.S.; Aghabozorgi, S.; Wah, T.Y.; Herawan, T. Big Data Clustering: A Review. In Proceedings of the Computational Science and Its Applications—ICCSA 2014, Guimarães, Portugal, 30 June–3 July 2014.
27. Singh, C.; Bala, A. A transform-based fast Fuzzy C-Means approach for high brain MRI segmentation accuracy. *Appl. Soft Comput.* **2019**, *76*, 156–173. [CrossRef]
28. Pal, N.R.; Bezdek, J.C. On cluster validity for the Fuzzy C-Means model. *IEEE Trans. Fuzzy Syst.* **1995**, *3*, 370–379. [CrossRef]
29. Stetco, A.; Zeng, X.J.; Keane, J. Fuzzy C-Means++: Fuzzy C-Means with effective seeding initialization. *Expert Syst. Appl.* **2015**, *42*, 7541–7548. [CrossRef]
30. Hashemzadeh, M.; Golzari Oskouei, A.; Farajzadeh, N. New Fuzzy C-Means clustering method based on feature-weight and cluster-weight learning. *Appl. Soft Comput.* **2019**, *78*, 324–345. [CrossRef]
31. Xing, H.-J.; Hu, B.-G. An adaptive Fuzzy C-Means clustering-based mixtures of experts model for unlabeled data classification. *Neurocomputing* **2008**, *71*, 1008–1021. [CrossRef]
32. Gamino, F.; Hernández, I.V.; Rosales, A.J.; Gallegos, F.J.; Mújica, D.; Ramos, E.; Carvajal, B.E.; Kinani, J.M.V. Block-matching Fuzzy C-Means clustering algorithm for segmentation of color images degraded with Gaussian noise. *Eng. Appl. Artif. Intell.* **2018**, *73*, 31–49. [CrossRef]
33. Cebeci, Z.; Yıldız, F. Comparison of K-Means and Fuzzy C-Means algorithms on different cluster structures. *J. Agricultural Inform.* **2015**, *6*, 13–23. [CrossRef]
34. Kaur, P. Intuitionistic fuzzy sets based credibilistic Fuzzy C-Means clustering for medical image segmentation. *Inter. J. Infor. Technol.* **2017**, *9*, 345–351. [CrossRef]
35. Tilson, L.V.; Excell, P.S.; Green, R.J. A Generalisation of the Fuzzy C-Means Clustering Algorithm. In Proceedings of the International Geoscience and Remote Sensing Symposium, Remote Sensing: Moving Toward the 21st Century, Edinburgh, UK, 12–16 September 1988.
36. Wang, X.; Wang, Y.; Wang, L. Improving Fuzzy C-Means clustering based on feature-weight learning. *Pattern Recognit. Lett.* **2004**, *25*, 1123–1132. [CrossRef]

37. Xue, Z.A.; Cen, F.; Wei, L.P. A Weighting Fuzzy Clustering Algorithm Based on Euclidean Distance. In Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Jinan, China, 18–20 October 2008.
38. Wan, R.; Yan, X.; Su, X. A Weighted Fuzzy Clustering Algorithm for Data Stream. In Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management, Guangzhou, China, 3–4 August 2008.
39. Pimentel, B.A.; de Souza, R.M.C.R. Multivariate Fuzzy C-Means algorithms with weighting. *Neurocomputing* **2016**, *174*, 946–965. [CrossRef]
40. Du, X. A robust and high-dimensional clustering algorithm based on feature weight and entropy. *Entropy* **2023**, *25*, 510. [CrossRef] [PubMed]
41. UCI Machine Learning Repository, University of California. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 22 October 2023).
42. Mukhtaruddin, R.; Rahman, H.A.; Hassan, M.Y.; Jamian, J.J. Optimal hybrid renewable energy design in autonomous system using Iterative-Pareto-Fuzzy technique. *Elect. Power Energy Syst.* **2015**, *64*, 242–249. [CrossRef]
43. Zhang, R.; Golovin, D. Random Hypervolume Scalarizations for Provable Multi-Objective Black Box Optimization. In Proceedings of the ICML'20: 37th International Conference on Machine Learning, Virtual, 13 July 2020.
44. Liu, X.; Tong, X.; Liu, Q. Profiling Pareto Front with Multi-Objective Stein Variational Gradient Descent. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online, 6–14 December 2021.
45. Kalimuthu, M.; Hayat, A.A.; Pathmakumar, T.; Rajesh Elara, M.; Wood, K.L. A deep reinforcement learning approach to optimal morphologies generation in reconfigurable tiling robots. *Mathematics* **2023**, *11*, 3893. [CrossRef]
46. Pérez, J.; Almanza, N.N.; Romero, D. Balancing effort and benefit of K-Means clustering algorithms in big data realms. *PLoS ONE* **2018**, *13*, e0201874. [CrossRef]
47. Bejarano, L.A.; Espitia, H.E.; Montenegro, C.E. Clustering analysis for the Pareto optimal front in multi-objective optimization. *Computation* **2022**, *10*, 37. [CrossRef]
48. Vimala, S.V.; Vivekanandan, K. A Kullback–Leibler divergence-based Fuzzy C-Means clustering for enhancing the potential of an movie recommendation system. *SN Appl. Sci.* **2019**, *1*, 698. [CrossRef]