MaxMin Linear Initialization for Fuzzy C-Means

Aybükë Oztürk, Stéphane Lallich, Jérôme Darmont, Sylvie Yona Waksman

MaxMin Linear Initialization for Fuzzy C-Means

Aybüke Öztürk*,**, Stéphane Lallich*
Jérôme Darmont*, and Sylvie Yona Waksman**

Université de Lyon, Lyon 2, France
*ERIC EA 3083 (5 avenue Pierre Mendès France, F69676 Bron Cedex)
**ArAr UMR 5138 (7 rue Raulin, 69365 Lyon Cedex 7)
{aybuke.ozturk,stephane.lallich,jerome.darmont}@univ-lyon2.fr
yona.waksman@mom.fr

Abstract. Clustering is an extensive research area in data science. The aim of clustering is to discover groups and to identify interesting patterns in datasets. Crisp (hard) clustering considers that each data point belongs to one and only one cluster. However, it is inadequate as some data points may belong to several clusters, as is the case in text categorization. Thus, we need more flexible clustering. Fuzzy clustering methods, where each data point can belong to several clusters, are an interesting alternative. Yet, seeding iterative fuzzy algorithms to achieve high quality clustering is an issue. In this paper, we propose a new linear and efficient initialization algorithm *MaxMin Linear* to deal with this problem. Then, we validate our theoretical results through extensive experiments on a variety of numerical real-world and artificial datasets. We also test several validity indices, including a new validity index that we propose, *Transformed Standardized Fuzzy Difference* (TSFD).

Keywords: *Clustering, Fuzzy C-Means, Seeding, Initialization, Maxmin Linear Method, Validity Indices*

1 Introduction

Clustering is a useful technique for grouping a set of unlabelled data points (instances) described by attributes (variables), such that points belonging to the same cluster (group) have similar characteristics, while points in different clusters have dissimilar characteristics. There are several types of clustering schemes, such as crisp, overlapping or fuzzy partitions, and hierarchies. Crisp clustering considers that each data point belongs to one and only one cluster. Contrary to crisp clustering, fuzzy clustering [1] considers that a data point can belong to more than one cluster. There are some situations where fuzzy clustering is very useful. For instance, let us consider three clusters achieved when categorizing textual documents: an economy cluster (topic), an energy cluster, and a politics cluster. Then a document containing the keyword "petrol" could belong to all

three clusters. Moreover, fuzzy clustering helps opening a discussion with domain experts regarding clustering results.

The primary objective of our paper is to avoid using highly complex clustering methods. One solution is to use iterative fuzzy methods such as Fuzzy C-Means (FCM) and Fuzzy K-Medoids. Both methods adapt the principle of the K-Means algorithm [2]. FCM, proposed by [3] and extended by [4], applies on numerical data, while Fuzzy K-Medoids [5] applies on categorical data. Since numerical data are the most common case, we choose to experiment our proposals with FCM.

The aim of the FCM algorithm is to minimize the fuzzy within-inertia FW (see Equation 1). Fuzzy inertia FI (see Equation 2) composes of the FW and the fuzzy between-inertia FB (see Equation 3). FW, FI, and FB are computed from a membership matrix U, which stores the membership coefficients u_{ik} of data point i to cluster k. Note that $FI = FW + FB$. Moreover, FI is not constant because it depends on u_{ik} value. When FW changes, the values of FI and FB also change.

$$FW = \sum_{i=1}^{n} \sum_{k=1}^{K} u_{ik}^{m} d^2(x_i, c_k) \tag{1}$$

$$FI = \sum_{i=1}^{n} \sum_{k=1}^{K} u_{ik}^{m} d^2(x_i, \overline{x}) \tag{2}$$

$$FB = \sum_{i=1}^{n} \sum_{k=1}^{K} u_{ik}^{m} d^2(c_k, \overline{x}) \tag{3}$$

where n is the number of instances, K is the number of clusters, m is the fuzziness coefficient (by default, $m = 2$. If $m = 1$, clustering is crisp. If $m > 1$, clustering becomes fuzzy), c_k is the center of the k^{th} cluster $\forall$ k, $1 \leq k \leq K$, $\overline{x}$ is the grand mean (the arithmetic mean of all data, see Equation 4), and function $d^2()$ computes the squared Euclidean distance.

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{4}$$

FCM starts by choosing K data points as initial centroids of the clusters. Then, membership matrix values u_{ik} (see Equation 5) are assigned to each data point in the dataset. Centroids of clusters c_k are updated based on Equation 6 until a termination criterion is reached successfully. In FCM, this criterion can be a fixed number of iterations t, e.g., $t = 100$. Alternatively, a threshold ϵ can be used, e.g., $\epsilon = 0.0001$. Then, the algorithm stops when the relative difference of objective function $< \epsilon$.

$$u_{ik} = \frac{1}{\sum_{j=1}^{K} \left(\frac{\|x_i - c_k\|^2}{\|x_i - c_j\|^2} \right)^{\frac{1}{m-1}}} \tag{5}$$

$$c_k = \frac{\sum_{i=1}^{n}(u_{ik}^m)x_i}{\sum_{i=1}^{n}(u_{ik}^m)} \tag{6}$$

When using FCM, an important point is the way of choosing K data points as initial centroids (seeds). An efficient initialization method should be linear, so that the FCM algorithm stays linear, too. Then, the initialization method must be evaluated using validity indices that are well suited to the fuzzy case.

To obtain a good validated clustering result, one has to minimize intra-cluster distance (compactness) and at the same times, one has to maximize inter-cluster distance (separability). The more often, proposed clustering validity indices associate a compactness index with a separability index.

Thence, we propose in this paper (1) a linear and efficient initialization method for FCM clustering called *MaxMin Linear*. Moreover, to compare our proposal with several initialization methods from the literature, we also propose (2) a new clustering validity index called *Transformed Standardized Fuzzy Difference* (TSFD), which is tailored to the fuzzy case. We perform validation experiments on several numerical real-world and artificial datasets.

The remainder of this paper is organized as follows. Section 2 presents initialization methods for iterative clustering and several clustering validity methods proposed in the literature. Sections 3 and 4 detail our contributions, i.e., the *MaxMin Linear* initialization method and the TSFD validity index, respectively. Section 5 deals with the experimental evaluation of the *MaxMin Linear* initialization method on several datasets, using several validity indices, including TSFD. Finally, we conclude this paper and provide some perspectives in Section 6.

2 Related Works

Most initialization methods are studied through K-Means clustering [2] concepts. We have reviewed various works from the literature, including much-cited papers [6,7,8]. In our study, we make use of commonly mentioned linear methods from these three papers.

The first initialization method by [2] uses the first K data points as centroids. This method is sensitive to the order of data. It is used by default in SPSS [9]. The second method by MacQueen (*MacQueen2*) takes K random data points as centroids. Moreover, [10] proposes to perform multiple relaunches of *MacQueen2*. Among the different relaunches, the one that optimizes FW (Equation 1) is considered the best candidate. This method is the standard way for initializing clusters. Its main disadvantage is that already selected points are not considered when a new seed is chosen. The second disadvantage is that outliers can be chosen. On the other hand, multiple runs ensure to improve the quality of the chosen sample.

Hand et al. [11], propose an extension of Faber's method that starts with a random set of seeds. It suggests iteratively modifying the partition by randomly moving some points to other clusters. The partition minimizing FW is chosen

as the best candidate. To move each data point to another random cluster, a probability α, e.g., $\alpha = 0.3$, must be set. The method is only interesting if parameter α is fixed for different datasets.

Bradley and Fayyad's method [12] starts by randomly partitioning the dataset into J subsets. Then, each subset is clustered with the K-Means algorithm using *MacQueen2* initialization. *MacQueen2* produces J sets of centers, each containing K points. The centers of clusters are combined into a superset. Then, the superset is clustered by K-Means J times. Each time, K-Means is initialized with a different center set, and members of the center set that give the smallest FW are selected as final centers.

The PCA-Part method [13] uses a divisive hierarchical approach based on Principal Component Analysis (PCA) [14]. The method starts with a single cluster containing the whole dataset. Then, it iteratively divides clusters with respect to FW. Clusters are divided into two sub-clusters by using a hyperplane that is orthogonal to the principal eigenvector of the cluster covariance matrix. The division process ends after K clusters are obtained.

The K-Means++ method [15] selects centroids based on a distance probability to the nearest center. First, it randomly selects an initial center $c_1 = x$ from the data point set X. Then, $d(x)$ is denoted as the shortest euclidean distance from x to its closest center. The next center c_i is randomly selected as $c_i = x' \in X$ with probability $d(x')^2 / \sum d(x)^2$.

Finally, in the literature, there are other methods having quadratic complexity [16,17]. Among quadratic methods, *MaxMin* (also called *Maximin*) [18] is particularly interesting. *MaxMin* first calculates all the paired distances between data points. Then, it chooses two centroids from the data points, which have the greatest distance to each other. Finally, the next centroid is the data point that is the farthest from its centroid. This approach helps decrease FW, which improves the homogeneity of clusters.

To summarize, Hand and Krzanowski [11] rely on user-defined parameters that may not be easy to set. *MacQueen2*, though easy to understand and implement, uses only one random sample. Faber improves the *MacQueen2*'s random sample through relaunches. In K-Means++, the random choice is replaced by a probabilistic choice and cluster homogeneity is taken into account. However, since the probabilistic selection does not always select sufficiently the large enough distance, several probabilistic samples are required and the best centers are selected from all relaunches.

In contrast, *MaxMin* constructs only one sample by decreasing FW and is thus deterministic. Thus, we can be sure that a chosen center is the best. Yet, it can be less effective than K-Means++ in the presence of outliers.

To evaluate initialization methods, we need to use fuzzy validity indices. According to [19], there are two groups of validity indices. The first group is only based on membership values and includes the partition coefficient index V_{PC} [20] (see Equation 7; $\frac{1}{K} \leq V_{PC} \leq 1$; to be maximized) and the Chen and Linkens index V_{CL} [21] (see Equation 8; $0 \leq V_{CL} \leq 1$; to be maximized).

$$V_{PC} = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} u_{ik}^2 \tag{7}$$

$$V_{CL} = \frac{1}{n}\sum_{i=1}^{n} max_k(u_{ik}) - \frac{1}{c}\sum_{k=1}^{K-1}\sum_{j=k+1}^{K}\left[\frac{1}{n}\sum_{i=1}^{n} min(u_{ik}, u_{ij})\right], \tag{8}$$

where $c = \sum_{k=1}^{K-1} k$.

V_{CL} takes in consideration both compactness (first term of V_{CL}) and separability (second term of V_{CL}). The second group of fuzzy validity indices is based on associating membership values to cluster centers and data. It includes the adaptation of the Ratio index V_{FRatio} to fuzzy clustering [22] (see Equation 9; $0 \leq V_{FRatio} \leq +\infty$; to be maximized), the penalized version of V_{FRatio} index which is the Calinski and Harabasz index V_{FCH} [22] (see Equation 10; $0 \leq V_{FCH} \leq +\infty$; to be maximized), the Fukuyama and Sugeno index V_{FS} [23] (see Equation 11; $-FI \leq V_{FS} \leq FI$; to be minimized), and the Xie and Beni index V_{XB} [24,25] (see Equation 12; $0 \leq V_{XB} \leq FI/n * min\|x_j - v_k\|^2$; to be minimized).

$$V_{FRatio} = FB/FW \tag{9}$$

$$V_{FCH} = \frac{FB/(K-1)}{FW/(n-K)} = \frac{n-K}{K-1}\frac{FB}{FW} \tag{10}$$

$$V_{FS} = FW - FB \tag{11}$$

$$V_{XB} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{n} u_{ik}^m \|x_i - v_k\|^2}{n * min_{j,k}\|v_j - v_k\|^2} \tag{12}$$

Among all the above stated validity indices, there is no single validity index that gives the best result for any dataset. Thus, there is room for a new validity index that is specifically tailored for fuzzy validation. This is why we propose the *Transformed Standardized Fuzzy Difference* index.

3 MaxMin Linear Fuzzy Clustering Initialization Method

MaxMin's simplicity and ability to build homogeneous clusters sounds very interesting. Yet, considering all paired distance between data points makes the method quadratic with respect to the number of data points. Thus, we present in this section an enhancement of *MaxMin* that makes it linear. Before introducing our changes, we first detail how *MaxMin* works in Algorithm 1 (see Section 2 for *MaxMin*'s principle).

In *MaxMin Linear*, we first calculate grand mean $\overline{x}$ (see Equation 4). Then, we choose as first centroid the data point that is nearest to $\overline{x}$. The second

Algorithm 1 *MaxMin*

Require: Set of data points $X = \{x_1, ..., x_n\}$
Require: Number of clusters K
  {Select the first two centroids c_1 and c_2}
  $c_1, c_2 \leftarrow \text{argmax}(d^2(x_i, x_j))\ i, j = 1, ..., n$
  $K^* \leftarrow 2$ {Number of seeds}
  {Find the remaining seeds}
  while $K^* < K$ **do**
    for all $x_i \neq c_{k^*}\ i = 1, ..., n,\ k^* = 1, ..., K^*$ **do**
      $d_m^2(x_i) \leftarrow \min(d^2(x_i, c_{k^*}))$
    end for
    $K^* \leftarrow K^* + 1$
    $c_{K^*} \leftarrow \text{argmax}(d_m^2(x_i))\ i = 1, ..., n$
  end while
  return $\{c_{k^*}\}\ k^* = 1, ..., K^*$

centroid is the data point that has the largest distance to the first centroid. Thus, complexity remains linear with respect to the number of data points. Afterwards, the choice of the remaining centroids remains the same as in *MaxMin*. *MaxMin Linear* is formalized in Algorithm 2.

As a final note, the use of *MaxMin Linear* is not limited to use with FCM on numerical data, but also with Fuzzy K-Medoids [26] for categorical data clustering. Thus, *MaxMin Linear* can also be applied with heterogeneous data to construct fuzzy clustering ensemble. This makes of *MaxMin Linear* a simple but noteworthy contribution, in our opinion.

4 Transformed Standardized Fuzzy Difference Validity Index

Several problems must be cleaned up to obtain a good clustering, including evaluation of the validity of the clusters and choosing the number of clusters. However, it is not an easy process. Compactness and separation level might raise problems. Firstly, if the chosen number of clusters is larger than optimal one, some clusters are broken while they could be more compact. Secondly, if the chosen number of clusters is smaller than optimal one, some clusters are merged and while they could be more separated. When it comes to addressing resolve those problems, many cluster validity indices are proposed for fuzzy clustering algorithms. The objective is to find the optimal number of clusters that can validate the best description of the data structure.

The optimal number of the cluster can be determined by considering the variation of clustering validity index. It is distinguished into two cases: The first case, if the index is not monotonic with the number of clusters, we choose the value of the number of clusters which optimizes the index. The second case, if the index is monotonic, one can prefer to use a penalized version of the index.

Algorithm 2 *MaxMin Linear*

Require: Set of data points $X = \{x_1, ..., x_n\}$
Require: Number of clusters K
{Select the first two centroids c_1 and c_2}
$\overline{x} \leftarrow \frac{1}{n} \sum_{i=1}^{n} x_i$
for $i \leftarrow 1$ **to** n **do**
$d_m^2(x_i) \leftarrow \min(d^2(\overline{x}, x_i))$
end for
$c_1 \leftarrow \text{argmin}(d_m^2(x_i))\ i = 1, ..., n$
for $i \leftarrow 1$ **to** n **do**
$d_m^2(x_i) \leftarrow \max(d^2(c_1, x_i))$
end for
$c_2 \leftarrow \text{argmax}(d_m^2(x_i))\ i = 1, ..., n$
$K^* \leftarrow 2$ {Number of seeds}
{Find the remaining seeds}
while $K^* < K$ **do**
for all $x_i \neq c_{k^*}\ i = 1, ..., n, k^* = 1, ..., K^*$ **do**
$d_m^2(x_i) \leftarrow \min(d^2(x_i, c_{k^*}))$
end for
$K^* \leftarrow K^* + 1$
$c_{K^*} \leftarrow \text{argmax}(d_m^2(x_i))\ i = 1, ..., n$
end while
return $\{c_{k^*}\}\ k^* = 1, ..., K^*$

In building TSFD, we first consider the difference $FB - FW$, which is similar to FS except for the sign). Unfortunately, $FI = FB + FW$ is not constant and $FB - FW \in [-FI, +FI]$. To take this particularity of fuzzy clustering into account, we propose to standardize $FB - FW$ by considering *Standardized Fuzzy Difference* $SFD = (FB - FW) \div FI$ instead. $SFD \in [-1, +1]$.

Finally, to obtain an index belonging to the $[0, 1]$ interval, we linearly transform SFD as $TSFD$ (see Equation 13; equal to FB/FI; $\in [0, 1]$; to be maximized)

$$TSFD = \frac{1 + SFD}{2} = \frac{FB}{FI} \tag{13}$$

5 Experimental Validation

In this section, we aim to compare *MaxMin Linear* to state of the art initialization methods for FCM-like clustering algorithms, i.e., *MacQueen2*, Faber's, K-Means++, and repeated K-Means++ (retaining the best result). These methods are indeed the most common linear methods and are good representatives for random, probability, and distance-based methods. Moreover, they do not require any parameterization. To achieve our comparison of initialization methods, we use the indices mentioned in Section 2.

5.1 Datasets

Initialization methods are compared on 15 commonly used real-life datasets from the UCI Machine Learning Repository[1] and seven artificial datasets. Their characteristics are featured in Table 1.

Table 1: Dataset features

ID	Datasets	# of data points	# of variables	# of clusters	Sources
1	Wine	178	13	3	UCI
2	Iris	150	4	3	UCI
3	Seeds	210	7	3	UCI
4	Original Wisconsin Breast Cancer (WBCD)	683	9	2	UCI
5	Wisconsin Diagnostic Breast Cancer (WDBC)	569	30	2	UCI
6	BUPA Liver Disorder (BUPA)	345	6	2	UCI
7	Pima	768	8	2	UCI
8	Glass	214	9	6	UCI
9	Vehicle	846	18	4	UCI
10	Segmentation	2310	19	7	UCI
11	Parkinson	150	22	2	UCI
12	Movement Libras	360	90	15	UCI
13	Ecoli	336	7	8	UCI
14	Yeast	1484	8	10	UCI
15	WineQuality-Red	1599	11	6	UCI
16	Bensaid	49	2	3	[27]
17	E1071-3	150	3	3	[28]
18	Ruspini_original	75	2	4	[1]
19	E1071-3-overlapped	150	3	3	[28]
20	Ruspini_noised	95	2	4	[1]
21	E1071-5	250	3	5	[28]
22	E1071-5-overlapped	250	3	5	[28]

In the case of real-life datasets, the true number of clusters in each dataset is assimilated to the number of labels. Although using the number of labels as the number of clusters is debatable, it is acceptable if the set of descriptive variables explain the labels well. In artificial datasets, the number of clusters is known by construction.

In addition, we created new artificial datasets by introducing overlapping and noise to some of the existing artificial datasets such as E1071-3, Ruspini_original, and E1071-5 datasets (see Table 1, ID 17, 18, and 21).

To create the dataset, new data points are introduced and each must be labeled. To obtain a dataset with overlapping, we modified the construction of

[1] http://archive.ics.uci.edu/ml/

the E1071 artificial datasets [28]. In the original datasets, there are three or five clusters of equal size (50). Cluster i is generated according to a Gaussian distribution $N(i; 0.3)$. To increase overlapping while retaining the same cluster size, we only change the standard deviation from 0.3 to 0.4. Then, there is no labeling problem.

Noise is introduced in each cluster by adding noisy points generated by a Gaussian variable around each label gravity center. First, for each label, we calculate the coordinates of centers, and the mean and standard deviation of each variable. With Gaussian variables, points mainly lie between "center +/- two standard deviations".

Noisy data are often generated by distributions with positive skewness. For example, in a two-dimensional dataset, for each label, we add points that are far away from the corresponding gravity center, especially on the right hand side, which generally contains the most points. Then, we draw a random number r between 0 and 1. If $r \leq 0.25$, the point is attributed to the left hand side. Otherwise, the point is attributed to the right hand side. This method helps obtain noisy data that are $^1/_4$ times smaller and $^3/_4$ times greater, respectively, than the expected value for the considered label. We apply this process to the Ruspini dataset [1].

5.2 Experimental Settings

In our experiments, we parameterize the FCM algorithm as follows: default termination criterion $\epsilon = 0.0001$ and default fuzziness coefficient value $m = 2$. We used these default settings as we are only interested in improving the initialization of FCM algorithm. All initialization methods and clustering validity indices are written in Python version 2.7.4. Repeated K-Means++ runs are performed ten times.

5.3 Experimental Results

In our experiments, we compare our method *MaxMin Linear* to all initialization methods from Section 2, on all datasets. We account for the following comparison criteria: number of iterations, V_{PC}, V_{CL}, FB, FW, FI, V_{FRatio}, V_{TSFD}, V_{FS}, and V_{XB}. We also rank the initialization methods with respect to all criteria.

Since presenting all results would take too much space, we only present three real-life datasets i.e., WineQuality-Red (Tables 2, 3, and 4), Glass (Tables 5, 6, and 7), and Segmentation (Tables 8, 9, and 10), as well as two of the artificial datasets we modified to introduce noise and overlapping, i.e., Ruspini_noised (Tables 11, 12, and 13), and E1071-5-overlapped (Tables 14, 15, and 16), respectively. Finally, the average ranking of initialization methods on all datasets is presented in Table 17.

From these experimental results, several observations can be drawn. In regard to the number of iterations, recall that Faber's and K-Means++ $\times 10$ methods are relaunches of two stochastic initialization methods: *MacQueen2* and K-Means++, respectively. With an average ranking of 1.68 (Table 17), *MaxMin*

Table 2: Experiment results on WineQuality-Red (1/2)

Initialization Method	# of iteration	V_{PC}	V_{CL}	FB	FW
MacQueen2	45	0.664	0.7455	110972.7	1224079.7
Faber	430	0.664	0.7455	**101440.4**	1224079.7
K-Means++	37	0.616	0.7029	101440.5	1089058.1
K-Means++ ×10	393	0.664	0.7455	**101440.4**	1224073.7
MaxMin Linear	**34**	**0.665**	**0.7458**	110972.7	**1224384.8**

Table 3: Experiment results on WineQuality-Red (2/2)

Initialization Method	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	1335052.363	11.0305	**0.9169**	-1113107.01	0.1621
Faber	1335052.363	11.0305	0.9148	-1113107.01	0.1621
K-Means++	1190498.537	10.7359	0.9148	-987617.57	0.2388
K-Means++ ×10	1335046.425	11.0304	0.9148	-1113101.04	0.1621
MaxMin Linear	**1335357.554**	**11.0332**	**0.9169**	**-1113412.13**	**0.1611**

Table 4: Ranking of initialization methods on WineQuality-Red

Initialization Method	# of iteration	V_{PC}	V_{CL}	FB	FW	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	3	2	2	4	2	2	2	2	2	2
Faber	5	2	2	2	2	2	2	5	2	2
K-Means++	2	5	5	3	5	5	5	3	5	5
K-Means++ ×10	4	4	4	**1**	4	4	4	4	4	4
MaxMin Linear	**1**	**1**	**1**	5	**1**	**1**	**1**	**1**	**1**	**1**

Table 5: Experiment results on Glass (1/2)

Initialization Method	# of iteration	V_{PC}	V_{CL}	FB	FW
MacQueen2	**44**	0.493	0.570	452.6	**154.1**
Faber	456	0.493	0.570	452.6	**154.1**
Kmeans++	56	0.493	0.570	452.6	**154.1**
K-Means++ ×10	366	0.493	0.570	452.6	**154.1**
MaxMin Linear	68	**0.555**	**0.645**	**508.3**	162.9

Table 6: Experiment results on Glass (2/2)

Initialization Method	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	606.8	2.94	0.74596	-298.5	2.358
Faber	606.8	2.94	0.74597	-298.5	2.358
Kmeans++	606.7	2.94	0.74593	-298.4	2.358
K-Means++ ×10	606.7	2.94	0.74604	-298.4	2.358
MaxMin Linear	**671.2**	**3.12**	**0.75725**	**-345.4**	**0.453**

Table 7: Ranking of initialization methods on Glass

Initialization method	# of iteration	V_{PC}	V_{CL}	FB	FW	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	**1**	2	2	2	**1**	2	2	4	2	5
Faber	5	3	3	3	2	3	3	3	3	2
Kmeans++	2	5	5	5	4	5	5	5	5	4
K-Means++ ×10	4	4	4	4	3	4	4	2	4	3
MaxMin Linear	3	**1**	**1**	**1**	5	**1**	**1**	**1**	**1**	**1**

Table 8: Experiment results on Segmentation (1/2)

Initialization Method	# of iteration	V_{PC}	V_{CL}	FB	FW
MacQueen2	103	0.381	0.476	12384361.4	5781042.6
Faber	731	0.398	0.488	14157566.6	5680259.6
Kmeans++	146	0.381	0.476	12388277.9	5781061.6
K-Means++ ×10	930	0.399	0.490	14254025.9	**5666840.5**
MaxMin Linear	**54**	**0.430**	**0.526**	**19234921.0**	6344612.7

Table 9: Experiment results on Segmentation (2/2)

Initialization Method	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	18165404.0	2.14	0.6818	-6603318.7	0.363
Faber	19837826.2	2.49	0.7137	-8477307.1	0.464
Kmeans++	18169339.6	2.14	0.6818	-6607216.3	0.361
K-Means++ ×10	19920866.4	2.52	0.7136	-8587185.5	**0.341**
MaxMin Linear	**25579533.7**	**3.03**	**0.7520**	**-12890308.3**	0.656

Table 10: Ranking of initialization methods on Segmentation

Initialization method	# of iteration	V_{PC}	V_{CL}	FB	FW	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	2	4	5	5	3	5	5	5	5	3
Faber	4	3	3	3	2	3	3	2	3	4
Kmeans++	3	5	4	4	4	4	4	3	4	2
K-Means++ ×10	5	2	2	2	**1**	2	2	4	2	**1**
MaxMin Linear	**1**	**1**	**1**	**1**	5	**1**	**1**	**1**	**1**	5

Table 11: Experiment results on Ruspini_noised (1/2)

Initialization Method	# of iteration	V_{PC}	V_{CL}	FB	FW
MacQueen2	9	0.775121	0.806518	219099.6	23421.0260
Faber	130	0.775125	0.806517	219100.8	23421.0258
Kmeans++	13	0.775122	0.806521	219101.1	23421.0258
K-Means++ ×10	105	**0.775128**	0.806518	219102.3	**23421.0256**
MaxMin Linear	**7**	0.775128	**0.806523**	**219105.4**	23421.0268

Table 12: Experiment results on Ruspini_noised (2/2)

Initialization Method	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	242520.7	9.3548	0.903427	-195678.6	0.063680
Faber	242521.9	9.3549	0.903427	-195679.8	0.063681
Kmeans++	242522.1	9.3549	0.903427	-195680.0	0.063676
K-Means++ ×10	242523.3	9.3549	0.903426	-195681.3	0.063681
MaxMin Linear	**242526.4**	**9.3551**	**0.903429**	**-195684.4**	**0.063672**

Table 13: Ranking of initialization methods on Ruspini_noised

Initialization Method	# of iteration	V_{PC}	V_{CL}	FB	FW	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	2	5	3	5	4	5	5	4	5	3
Faber	5	3	5	4	2	4	4	3	4	5
Kmeans++	3	4	2	3	3	3	3	2	3	2
K-Means++ ×10	4	**1**	4	2	**1**	2	2	5	2	4
MaxMin Linear	**1**	2	**1**	**1**	5	**1**	**1**	**1**	**1**	**1**

Table 14: Experiment results on E1071-5-overlapped (1/2)

Initialization Method	# of iteration	V_{PC}	V_{CL}	FB	FW
MacQueen2	8	0.735646	0.762681	219.7337	48.715631
Faber	103	0.735645	0.762683	219.7358	48.715630
Kmeans++	12	0.735651	0.762685	219.7408	48.715632
K-Means++ ×10	113	0.735645	0.762683	219.7363	**48.715629**
MaxMin Linear	**7**	**0.735652**	**0.762688**	**219.7445**	**48.715629**

Table 15: Experiment results on E1071-5-overlapped (2/2)

Initialization Method	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	268.4494	4.5105	0.818530	-171.0181	0.11574
Faber	268.4514	4.5106	0.818535	-171.0202	**0.11569**
Kmeans++	268.4565	4.5107	0.818534	-171.0252	0.11575
K-Means++ ×10	268.4519	4.5106	0.818530	-171.0207	**0.11569**
MaxMin Linear	**268.4601**	**4.5108**	**0.818537**	**-171.0288**	0.11572

Table 16: Ranking of initialization methods on E1071-5-overlapped

Initialization method	# of iteration	V_{PC}	V_{CL}	FB	FW	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	2	3	5	5	4	5	5	5	5	4
Faber	4	5	4	4	3	4	4	2	4	**1**
Kmeans++	3	2	2	2	5	2	2	3	2	5
K-Means++ ×10	5	4	3	3	**1**	3	3	4	3	2
MaxMin Linear	**1**	**1**	**1**	**1**	2	**1**	**1**	**1**	**1**	3

Table 17: Average ranking of initialization methods on all datasets

Initialization method	# of iteration	V_{PC}	V_{CL}	FB	FW	FI	V_{FRatio}	V_{TSFD}	V_{FS}	V_{XB}
MacQueen2	1.95	3.36	3.55	3.86	3.41	3.41	3.41	3.04	3.41	3.55
Faber	4.45	2.73	2.82	1.73	2.73	2.73	2.73	3.27	2.73	2.91
K-Means++	1.95	3.86	3.68	3.86	3.86	3.86	3.86	3.54	3.86	3.36
K-Means++ ×10	4.41	2.68	2.55	**1.64**	2.86	2.86	2.86	3.22	2.86	2.82
MaxMin Linear	**1.68**	**2.27**	**2.32**	3.82	**2.05**	**2.05**	**2.05**	**1.86**	**2.05**	**2.27**

Linear outperforms all other methods, including single-run methods *MacQueen2* (average ranking: 1.95) and K-Means++ (average ranking: 1.95).

Regarding clustering result quality, *MaxMin Linear* obtains the best average ranking for eight of the nine experimented quality indices (Table 17). Only the FB index yields a better result for the two multiple-runs methods, while the result of *Maxmin Linear* is similar to those of *MacQueen2* and K-Means++. However, *Maxmin Linear* achieves the best trade-off between FB and FW, and thus maximizes the indices that take both FB and FW into account (V_{FRatio}, V_{TSFD}, V_{FS} and V_{XB}). The best result for *MaxMin Linear* is obtained with V_{TSFD} (average ranking of 1.86; Table 17), the new index specially tailored for fuzzy clustering that we propose.

In conclusion, the results obtained with *MaxMin Linear* are a little better than those obtained with multiple-runs methods, but they require ten times fewer iterations. Moreover, *MaxMin Linear* is deterministic, whereas multiple-runs methods are stochastic.

6 Conclusion and Perspectives

In this paper, we propose a new, fast, and easy to implement initialization method for FCM called *MaxMin Linear*. *MaxMin Linear* is compared to several initialization methods from the literature. It is experimentally shown that *MaxMin Linear* outperforms existing methods on 22 datasets. Moreover, we also propose an appropriate fuzzy validity index, TSFD, to evaluate initialization methods.

In addition, *MaxMin Linear* can be applied to algorithms other than FCM, such as Fuzzy K-Modes and Fuzzy K-Medoids, which apply on categorical. In particular, *MaxMin Linear* allows decreasing the complexity of Park's Fuzzy K-Medoids implementation.

In consequence, an immediate perspective to our work is to propose a new clustering ensemble method for heterogeneous datasets composed of both numerical and categorical data.

Acknowledgments

This project is supported by the Rhône Alpes Region's ARC 5: "Cultures, Sciences, Sociétés et Médiations" through A. Öztürk's Ph.D. grant.

References

1. Ruspini, E.H.: Numerical methods for fuzzy clustering. Information Sciences **2**(3) (1970) 319–350
2. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Volume 1., Oakland, CA, USA. (1967) 281–297
3. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. (1973)
4. Bezdek, J.C., Ehrlich, R., Full, W.: Fcm: The fuzzy c-means clustering algorithm. Computers & Geosciences **10**(2-3) (1984) 191–203
5. Kaufman, L., Rousseeuw, P.J.: Partitioning around medoids (program pam). Finding groups in data: an introduction to cluster analysis (1990) 68–125
6. Steinley, D., Brusco, M.J.: Initializing k-means batch clustering: A critical evaluation of several techniques. Journal of Classification **24**(1) (2007) 99–121
7. Maitra, R., Peterson, A.D., Ghosh, A.P.: A systematic evaluation of different methods for initializing the k-means clustering algorithm. IEEE Transactions on Knowkedge and Data Engineering, (2011) 41
8. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert systems with applications **40**(1) (2013) 200–210
9. Norušis, M.J.: IBM SPSS statistics 19 statistical procedures companion. Prentice Hall (2012)
10. Faber, V.: Clustering and the continuous k-means algorithm. Los Alamos Science **22**(138144.21) (1994)
11. Hand, D.J., Krzanowski, W.J.: Optimising k-means clustering results with standard software packages. Computational Statistics & Data Analysis **49**(4) (2005) 969–973
12. Bradley, P.S., Fayyad, U.M.: Refining initial points for k-means clustering. In: ICML. Volume 98. (1998) 91–99
13. Su, T., Dy, J.G.: In search of deterministic methods for initializing k-means and gaussian mixture clustering. Intelligent Data Analysis **11**(4) (2007) 319–338
14. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometrics and intelligent laboratory systems **2**(1-3) (1987) 37–52
15. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics (2007) 1027–1035
16. Lance, G.N., Williams, W.T.: A general theory of classificatory sorting strategies: Ii. clustering systems. The computer journal **10**(3) (1967) 271–277
17. Astrahan, M.: Speech analysis by clustering, or the hyperphoneme method. Technical report, Stanford Univ. CA Dept of Computer Science (1970)
18. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38** (1985) 293–306

19. Wang, W., Zhang, Y.: On fuzzy cluster validity indices. Fuzzy sets and systems **158**(19) (2007) 2095–2117
20. Bezdek, J.C.: Cluster validity with fuzzy sets. (1973)
21. Chen, M.Y., Linkens, D.A.: Rule-base self-generation and simplification for data-driven fuzzy models. In: Fuzzy Systems, 2001. The 10th IEEE International Conference on. Volume 1., IEEE (2001) 424–427
22. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics-theory and Methods **3**(1) (1974) 1–27
23. Fukuyama, Y.: A new method of choosing the number of clusters for the fuzzy c-mean method. In: Proc. 5th Fuzzy Syst. Symp., 1989. (1989) 247–250
24. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Transactions on pattern analysis and machine intelligence **13**(8) (1991) 841–847
25. Pal, N.R., Bezdek, J.C.: On cluster validity for the fuzzy c-means model. IEEE Transactions on Fuzzy systems **3**(3) (1995) 370–379
26. Park, H.S., Jun, C.H.: A simple and fast algorithm for k-medoids clustering. Expert systems with applications **36**(2) (2009) 3336–3341
27. Bensaid, A.M., Hall, L.O., Bezdek, J.C., Clarke, L.P., Silbiger, M.L., Arrington, J.A., Murtagh, R.F.: Validity-guided (re) clustering with applications to image segmentation. IEEE Transactions on Fuzzy Systems **4**(2) (1996) 112–123
28. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.C., Lin, C.C., Meyer, M.D.: Package e1071. Version 1.6-8 (2017)