Fuzzy C-means++: Fuzzy C-means with effective seeding initialization

Adrian Stetco *, Xiao-Jun Zeng, John Keane

School of Computer Science, University of Manchester, United Kingdom

Article info
Article history:
Available online 22 May 2015

Keywords:
Cluster analysis
Fuzzy C-means clustering
Initialization

Abstract
Fuzzy C-means has been utilized successfully in a wide range of applications, extending the clustering capability of the K-means to datasets that are uncertain, vague and otherwise hard to cluster. This paper introduces the Fuzzy C-means++ algorithm which, by utilizing the seeding mechanism of the K-means++, improves the effectiveness and speed of Fuzzy C-means. By careful seeding that disperses the initial cluster centers through the data space, the resulting Fuzzy C-means++ approach samples starting cluster representatives during the initialization phase. The cluster representatives are well spread in the input space, resulting in both faster convergence times and higher quality solutions. Implementations in R of standard Fuzzy C-means and Fuzzy C-means++ are evaluated on various data sets. We investigate the cluster quality and iteration count as we vary the spreading factor on a series of synthetic data sets. We run the algorithm on real world data sets and to account for the non-determinism inherent in these algorithms we record multiple runs while choosing different k parameter values. The results show that the proposed method gives significant improvement in convergence times (the number of iterations) of up to 40 (2.1 on average) times the standard on synthetic datasets and, in general, an associated lower cost function value and Xie–Beni value. A proof sketch of the logarithmically bounded expected cost function value is given.
© 2015 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Partitional cluster analysis is defined as the problem of partitioning a group of objects into clusters that share similar characteristics. The most well-known and widely used partitional clustering algorithms are K-means and Fuzzy C-means (Peizhuang, 1983). When compared across clusters, members of a cluster will be different from members of all other clusters. In order to quantify the similarity/dissimilarity relationship between objects, metric functions, defined on both numeric (Euclidean, Manhattan, Cosine, etc.) or non-numeric (Hamming, Jaro-Winkler, Levenshtein, etc.) data have been used.

K-means is one of the oldest clustering algorithms (MacQueen, 1967) and refers both to the clustering task and a specific algorithm to solve it. Given a set X of input data and a parameter k, the task is to choose k representatives of X such that the distances between any points in X and their representative is minimized. The set of representatives discovered after running the K-means algorithm is enough to define a clustering of the points in the data space (the ith cluster being the set of all points in X that are closer to r, than any other representative).

In contrast to the of K-means where each point belongs to one cluster, in Fuzzy C-means each point xi in the space belongs to r, \( i \notin R \) with \( u_i \in [0, 1] \) defined in the membership matrix (of size \( n \times k \) where n is the number of points in the data space and k is the number of representatives). The use of a membership matrix increases the expressiveness of the clustering analysis, arguably presenting a more comprehensive view of relationships present in the data. Further, the hard assignment of the data points by K-means is inadequate when the points are equally distant between representatives, in which case they will be randomly assigned to one cluster or another (Doring, Lesot, & Kruse, 2006).

Fuzzy C-means mitigates this problem by assigning equal degrees of belonging through the use of the membership matrix. This method computes membership degrees at each iteration, a costly operation that gives a membership degree to a point proportional to its proximity to the cluster representatives. Moreover, the size of this matrix grows as a product of the number of points and clusters, making the algorithm computationally expensive for high values. To reduce the computational burden of the algorithm and at the same time increase its accuracy, an integration of the K-means careful seeding algorithm (Arthur, Arthur, Vassilvitskii, & Vassilvitskii, 2007) into the standard version of Fuzzy C-means is proposed, analyzed and verified in this paper.

The reminder of the paper is structured as follows: Section 2 presents work improving the performance of Fuzzy C-means;
the standard and the proposed algorithm are introduced in Section 3, together with a proof that shows the theoretical bounds of the expected cost function; Section 4 presents the datasets and the evaluation procedure used, and compares the proposed scheme with the standard algorithm; Section 5 summarizes findings and considers future work.

2. Background

Although noted both for its simplicity of implementation and its output validity, Fuzzy C-means suffers from high computational cost. For each iteration the computational complexity of the algorithm is quadratic in the number of clusters $O(NC^2P)$ where $N$ is the number of data points, $C$ is the number of clusters and $P$ is the dimension of the data points. A linear complexity approach $O(NCP)$ that removes the need to store a large matrix during the iterations was proposed in Kolen and Hutcheson (2002). In Wang, Wang, and Wang (2004) a method to obtain qualitatively better clusters (as measured using a series of validity indexes) is proposed. This approach uses a weighted Euclidean distance which incorporates feature weights. While this method showed promising results on several UCI databases, it requires a feature weight learning step of complexity $O(NE_{2CP})$.

Work by Zou, Wang, and Hu (2008) addresses the problem of initializing the cluster representatives by partitioning the space into grid blocks (finite disjoint rectangle-like units) and performing a search for condensation points. A grid block is considered dense if the number of data points present in it are bigger than a given input threshold parameter. Condensation points are geometric centers of dense grid blocks and serve as good initialization points to be chosen as cluster center before commencing the Fuzzy C-means algorithm. Although this method works well on two-dimensional datasets, the question remains how well it would work for non-spherical cluster types, and what should the block sizes and density threshold values be.

Yang, Zhang, and Tian (2010) propose a methodology for picking centres based on subtractive clustering. The potential of each point to become an initial centre is a function of its neighboring points: the more neighbors the higher the chance of being picked. Although promising, being able to select the number of $k$ parameters as well as initializing the algorithm, this method lacks enough empirical tests on real world datasets. Moreover it has four additional parameters that need tuning.

Celebi, Kingravi, and Vela (2013) conducted a comparative study on eight linear-time initialization techniques for K-means algorithm on a large variety of data sets. The study has looked at the quality (taking into consideration cost function values, external validity index) and speed number of iterations and CPU time) of the approaches. While most of these methods were non-deterministic (generating different initial points), two of them where deterministic, picking the same starting points every time when executed on the same data set. On real data sets, the non-deterministic methods (such as K-means++ (Arthur et al., 2007)) performed better than deterministic ones (with respect to minimum statistic), a fact that can be attributed to multiple local minima of the datasets and the fact that they were executed multiple times. However, the authors’ argue that deterministic methods need only one run, thus total computational complexity could be lower in their case.

The K-means++ method (Arthur et al., 2007), the basis of this work, initializes the cluster centers of the K-means algorithm by selecting points in the dataset that are further away from each other in a probabilistic manner. This method both avoids the problems of the standard method and improves speed of convergence, being theoretically guaranteed to be $O(\log k)$, and hence competitive with the optimal solution. While Celebi et al. (2013) used the standard K-means++ initialization method in their study, we focus on the more general case and apply it to Fuzzy C-means, using a parameter to control the spreading. This method improves the way in which Fuzzy C-means initializes its clusters and has several advantages over the methods discussed. The method achieves superior clustering (in terms of validity indexes) compared to using a random initialization as in the standard and fewer iterations. The proposed method is also easier to understand and implement and compared to other methods it needs just one parameter that controls the spreading factor.

The R programming language (R Development Core Team, 2013) is used here with the e1071 package (Meyer, Dimitriadou, Hornik, Weingessel, & Leisch, 2014) which, as well as containing standard clustering algorithms, contains useful cluster validity functions to test the quality of the discovered structures.

3. The algorithms

3.1. Fuzzy C-means algorithm

The standard version of the Fuzzy C-means algorithm (Peizhuang, 1983) – Algorithm 1 – minimizes the function:

$$J_m(U, R) = \sum_{i=1}^{n} \sum_{j=1}^{k} \mu_{i}^{m} [x_i - r_j]^2$$

subject to

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

Algorithm 1: Fuzzy C-means (FCM)

Given $X = \{x_i\}_{i=1}^{N}$ and $k$, return $U$ and $R$

1: procedure FCM (Data set $X$, Clusters $k$)
2: $U^0$ is randomly initialized
3: repeat
4: $r_j = \frac{1}{\sum_{i=1}^{n} \mu_{ij}^{m} x_i^{m}} \sum_{i=1}^{n} \mu_{ij}^{m} x_i^m \cdot \quad \forall j = 1 \ldots k$
5: $U_{ij} = \frac{1}{\sum_{j=1}^{k} \gamma_{ij}^{m}} \gamma_{ij}^{m}$
6: until $|U^{k+1} - U^{k}| < e$
7: end procedure

where $X = \{x_i\}_{i=1}^{N}$ the set of data points, $U = \{\mu_{ij}\}_{i,j=1}^{N,k}$ the matrix of membership degrees, $k \in N$ the number of clusters and $R = \{r_j\}_{j=1}^{k}$ the set of representatives, $m$ is the fuzzifier parameter which

![Fig. 1. Equal sized clusters with no overlap. Real cluster centers are marked in magenta, Fuzzy C-means initial clusters are marked in red, while in the yellow and green triangles we have Fuzzy C-means++ (with $p = 0.5$ and $p = 1.8$ respectively). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)](image-url)
determines the fuzziness of the cluster. At the limit \( m \to 1 \), the method becomes standard K-means, here we use a default value of \( e = 10^{-5} \) and the \( | | \) operation denotes the Frobenius norm of the difference between the matrices. The difference is default and just one of the many ways in which we can assess convergence. We use the optimized implementation of Fuzzy C-means which uses the difference between subsequent cost function values to establish convergence (Meyer et al., 2014).

### 3.2. Fuzzy C-means++ algorithm

#### 3.2.1. Algorithm

Initializing representatives by selecting random points from the input dataset results in a sub-optimal starting strategy for the standard version of the algorithm: consider Figs. 1, 3, 5 and 7. The idea behind the proposed Fuzzy C-means++ scheme (Algorithm 2) is to choose points that are spread out in the data set as representatives and update the membership matrix accordingly before commencing Steps 2–4. The representative sampled with our proposed method (plotted with yellow and (especially) green triangles in Figs. 1, 3, 5 and 7) start in a much better position, closer to the real center of the clusters (marked in magenta) and thus requiring a much lower number of steps to converge. Note that the more well behaved (non-overlapping) the data set is, the more predictable (less volatile) is the initialization outcome of the Fuzzy C-means++ scheme.

#### 3.2.2. Proof

On the assumption that a good clustering is well spread out in the data space, Arthur et al. (2007) have shown theoretically that the method for K-means and \( p = 2 \), not only achieves \( O(\log k) \) approximation of the optimal cost but, by placing the representatives further away from each other at the beginning, the algorithm converges in fewer iterations. After initialization with either of the two investigated algorithms, the cost function can only decrease monotonically as Steps 4–5 are repeatedly recomputed until convergence, thus the proof focuses on showing that the performance after initialization is \( O(\log k) \) competitive:

---

![Fig. 2. Mean number of iterations on equal sized clusters with no overlap, function of \( p \).](image-url)

![Fig. 3. Equal sized clusters with some overlap. Real clusters are marked in magenta, Fuzzy C-means initial clusters are marked in red, while in the yellow and green triangles we have Fuzzy C-means++ (with \( p = 0.5 \) and \( p = 1.8 \) respectively).](image-url)
Here we present a brief sketch of the their proof which shows that the expected cost when using this scheme is logarithmically bounded (for more details see Arthur et al. (2007)). Firstly, picking point $a_0$ at random (Step 9) from the first cluster $A_1 \in C_{OPT}$ (where $C_{OPT}$ denotes the optimal clustering) incurs an expected cost of:

$$E[\theta(A)] = \frac{1}{|A|} \sum_{a \in A} ||a - a_0||^2 = 2 \sum_{a \in A} ||a - \text{mean}(A)||^2 = 2 \theta_{OPT}(A)$$

where $|A|$ is the cardinality of set $A$, $\text{mean}(A)$ is the mean of the data points in $A$, $\theta$ is the cost incurred by picking a random point $a$ as the center of $A$. Moving to other clusters and selecting $b_0$ from $B$ using the proposed scheme, we have:

$$E[\theta(B)] = \sum_{b \in B} \frac{D^2(b_0)}{\sum_{b \in B} D^2(b)} \sum_{b \in B} \min(D(b), ||b - b_0||)^2$$

and by triangle inequality:

$$D(b_0) \leq D(b) + ||b - b_0||$$

which by using the power mean inequality for $\forall b$ becomes:

$$D^2(b_0) \leq 2D^2(b) + 2||b - b_0||^2$$

averaging $\forall b \in B$ we get:

$$D^2(b_0) \leq \frac{2}{|B|} \sum_{b \in B} D^2(b) + \frac{2}{|B|} \sum_{b \in B} ||b - b_0||^2$$

Using induction it can be shown that, in general, the expected cost of the clustering discovered by the proposed method is logarithmically bounded:

$$E[\theta(C)] \leq 8(\ln k + 2)\theta_{OPT}(C)$$

4. Evaluation

This section considers ways in which clusters can be distributed and evaluates the performance of both the proposed scheme and the standard version of Fuzzy C-means in terms of speed (the number of iterations to convergence), final cost function value and
appropriateness of data partitions using the Xie–Beni validity index on real datasets. Because of the stochastic nature of Fuzzy C-means (which randomly initializes the membership partition and centers) and the Fuzzy C-means++ method (which picks the first point at random from the data set and the others further away in a probabilistic manner), each test has been run ten times and the averages computed for the number of iterations, Xie–Beni function value and final cost function value. Although several validity indexes have been proposed in the literature (such as Gath and Geva (1989), Fukuyama and Sugeno (1989), etc.) the Xie–Beni (Xie & Beni, 1991) is used here as it has been shown to be superior and reliable (Pal & Bezdek, 1995) being defined as:

\[
XB = \frac{\sum_{i=1}^{N} \sum_{j=1}^{K} \mu_{ij}^{p} ||x_i - r_j||^2}{N \min_{i \neq j} ||r_i - r_j||^2}
\]

In (4) the numerator is the Fuzzy C-means cost function \(J_m\) and represents the tightness of the discovered clusters (the higher the value, the more cohesive are the clusters). The denominator captures the spread in the clusters by computing the minimum distance between two neighboring clusters (the higher the value, the better).

The performance of Fuzzy C-means++ has been investigated on four artificially generated and three real world datasets. The artificial datasets consist of five globular clusters (in two dimensions) that fall into four possible categories: clusters of equal size with no overlap, clusters of different sizes with no overlap, clusters of equal size and overlap and clusters of different sizes with overlap (Figs. 1, 3, 5 and 7) below.

In Fig. 1 we observe the outcomes of initializing the centers using the standard method. This method is allowed to choose freely \(k\) centers at random from the dataset supplied as input. These centers, marked in red, are often picked from the same clusters and hence have to travel many steps to their final position (magenta). For this type of data, a power \(p = 0.5\) for Fuzzy C-means++ generates initial cluster centers (yellow triangles) similar to the default method. In fact, looking at Fig. 2 above we notice
that in the range $p \in [0.1, 1.3]$, the Fuzzy C-means++ method produces very volatile results in terms of iterations with little improvement over the standard.

Progressing toward higher powers ($p \geq 1.3$) more importance is placed on picking further away centers and the algorithm achieves convergence in 3 steps as opposed to 9 required by the standard approach. For equal clusters with some overlap (Fig. 3) we observe that the standard method needs 41.7 iterations to converge (Fig 4), fewer (with some exceptions) than Fuzzy C-means++ with powers in $p \in [0.1, 1]$. Fuzzy C-means++ with powers $p \geq 1$ achieve less volatile results and perform much better than the standard approach (19.6 iterations on average).

As clusters become different in size, the performance of Fuzzy C-means over the whole range begins to degrade (Figs. 5 and 6), yet still performs better than the standard approach (it needs on average 44.8 iteration as compared to the standard which needs 65.8).

Fuzzy C-means++, on average, performs better than the standard approach on unequal and overlapping clusters, needing on average 86.8 iterations on the whole range of powers although

### Table 1
Iris dataset results.

<table>
<thead>
<tr>
<th>IRIS Dataset</th>
<th>Number of iterations FCM</th>
<th>Number of iterations FCM++</th>
<th>Within Error FCM</th>
<th>Within Error FCM++</th>
<th>Xie-Beni FCM</th>
<th>Xie-Beni FCM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>K=2</td>
<td>7.9</td>
<td>7.5</td>
<td>0.8592993</td>
<td>0.8592993</td>
<td>0.000361163</td>
<td>0.000361163</td>
</tr>
<tr>
<td>K=3</td>
<td>23.6</td>
<td>16.3</td>
<td>0.4033714</td>
<td>0.4033714</td>
<td>0.0009127244</td>
<td>0.0009127211</td>
</tr>
<tr>
<td>K=4</td>
<td>28.5</td>
<td>20.9</td>
<td>0.2827292</td>
<td>0.2933312</td>
<td>0.00159312</td>
<td>0.002174577</td>
</tr>
<tr>
<td>K=5</td>
<td>32.1</td>
<td>23.3</td>
<td>0.2350914</td>
<td>0.2186208</td>
<td>0.003126444</td>
<td>0.00188627</td>
</tr>
<tr>
<td>K=6</td>
<td>54.5</td>
<td>43.9</td>
<td>0.1873708</td>
<td>0.1788701</td>
<td>0.003396069</td>
<td>0.002843442</td>
</tr>
<tr>
<td>K=7</td>
<td>48.6</td>
<td>37.3</td>
<td>0.1604981</td>
<td>0.1496806</td>
<td>0.005690668</td>
<td>0.002834393</td>
</tr>
<tr>
<td>K=8</td>
<td>35.7</td>
<td>55.8</td>
<td>0.1423903</td>
<td>0.1232845</td>
<td>0.006304031</td>
<td>0.002155966</td>
</tr>
<tr>
<td>K=9</td>
<td>50.4</td>
<td>47.3</td>
<td>0.1121423</td>
<td>0.1080724</td>
<td>0.004296329</td>
<td>0.003102919</td>
</tr>
<tr>
<td>K=10</td>
<td>37</td>
<td>59.2</td>
<td>0.09999715</td>
<td>0.09892117</td>
<td>0.005141721</td>
<td>0.004666739</td>
</tr>
</tbody>
</table>

### Table 2
Spam dataset results.

<table>
<thead>
<tr>
<th>Spam Dataset</th>
<th>Number of iterations FCM</th>
<th>Number of iterations FCM++</th>
<th>Within Error FCM</th>
<th>Within Error FCM++</th>
<th>Xie-Beni FCM</th>
<th>Xie-Beni FCM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>K=2</td>
<td>26.9</td>
<td>23.9</td>
<td>137448.1</td>
<td>137448.1</td>
<td>1.085155e-05</td>
<td>1.084566e-05</td>
</tr>
<tr>
<td>K=3</td>
<td>34.8</td>
<td>23.9</td>
<td>73642.85</td>
<td>81690.52</td>
<td>2.116524e-05</td>
<td>1.292544e-05</td>
</tr>
<tr>
<td>K=4</td>
<td>43.7</td>
<td>31.8</td>
<td>49749.26</td>
<td>49509.64</td>
<td>4.771807e-05</td>
<td>1.864137e-05</td>
</tr>
<tr>
<td>K=5</td>
<td>71.8</td>
<td>37.4</td>
<td>38253.01</td>
<td>31621.59</td>
<td>6.594652e-05</td>
<td>3.75236e-05</td>
</tr>
<tr>
<td>K=6</td>
<td>145.1</td>
<td>48.4</td>
<td>30797.2</td>
<td>25292.62</td>
<td>6.148523e-05</td>
<td>4.307133e-05</td>
</tr>
<tr>
<td>K=7</td>
<td>76.9</td>
<td>57.7</td>
<td>25475.89</td>
<td>19654.03</td>
<td>0.0001564695</td>
<td>4.099956e-05</td>
</tr>
<tr>
<td>K=8</td>
<td>145.3</td>
<td>61.3</td>
<td>21962.28</td>
<td>15058.67</td>
<td>0.0001583545</td>
<td>5.90694e-05</td>
</tr>
<tr>
<td>K=9</td>
<td>99.5</td>
<td>66.7</td>
<td>18817.46</td>
<td>11659</td>
<td>0.0002875267</td>
<td>7.696938e-05</td>
</tr>
<tr>
<td>K=10</td>
<td>217.9</td>
<td>84.5</td>
<td>17062.23</td>
<td>8919.816</td>
<td>0.0002774631</td>
<td>7.642981e-05</td>
</tr>
</tbody>
</table>
the volatility is present over the whole range with some powers $p$
performing worse than the standard which needs on average 111
iterations.

Three real world datasets from the UCI Machine Learning
Repository (Asuncion & Newman, 2007) are considered: IRIS,
SPAM and WINE. For the Fuzzy C-means++ algorithm we pick a
power $p = 1.8$, a value proved to achieve best performance on all
the artificial cases considered above: it does not spread the centers
too far (which could result in outliers being selected) and it does
not chose them too close (which would result in performance sim-
lar to the standard approach) (see Fig. 8).

The Iris dataset consists of 150 instances belonging to one of
two classes each referring to a particular type of Iris plant: setosa,
versicolor or virginica. The data, being multidimensional, cannot be
viewed directly. By projecting it to the first two principal compo-
nents (Fig. 9) we notice the existence of two overlapped clouds of
points. These two types of clouds belonging to iris versicolor
and virginica are harder to distinguish. More information can be
obtained by looking at the variable correlations in Fig. 10.

With the exceptions of $k = 8$ and $k = 10$, Fuzzy C-means++ outper-
forms the standard. Moreover, in general, Fuzzy C-means++ achieves
both a lower cost function value and a lower Xie Beni index, indicat-
ing better clusters (Table 1). For the real case $k = 3$, FCM++ needs on
average 7.3 iterations less than the standard approach. For this case,
the two algorithms converge at the same points, having the same
cost function value and Xie Beni (see Table 1).

The SPAM dataset (4601 instances) contains attributes describ-
ing the characteristics of two categories of emails (solicited and
unsolicited) where the features (56 continuous real-valued)
describe frequencies of certain words, capital letters run length,
etc.

The PCA projection of the SPAM dataset (Fig. 11) on the first two
principal components reveals an overlapping cloud of points. Analysis reveals that most of the variables are either positively cor-
related with the first or the second component (Fig. 12). Across the
range of $k$s, the Fuzzy C-means++ algorithm with a power $p = 1.8$
outperforms the standard approach in some cases by a large factor

<table>
<thead>
<tr>
<th>Wine Dataset</th>
<th>Number of iterations FCM</th>
<th>Number of iterations FCM++</th>
<th>Within Error FCM</th>
<th>Within Error FCM++</th>
<th>Xie Beni FCM</th>
<th>Xie Beni FCM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>K=2</td>
<td>7.9</td>
<td>7.5</td>
<td>0.8592993</td>
<td>0.8592993</td>
<td>0.000361163</td>
<td>0.000361163</td>
</tr>
<tr>
<td>K=3</td>
<td>23.6</td>
<td>16.3</td>
<td>0.4033714</td>
<td>0.4033714</td>
<td>0.0009127244</td>
<td>0.0009127211</td>
</tr>
<tr>
<td>K=4</td>
<td>28.5</td>
<td>20.9</td>
<td>0.2827292</td>
<td>0.2933132</td>
<td>0.00159312</td>
<td>0.002174577</td>
</tr>
<tr>
<td>K=5</td>
<td>32.1</td>
<td>23.3</td>
<td>0.2350914</td>
<td>0.2186208</td>
<td>0.003126444</td>
<td>0.00188627</td>
</tr>
<tr>
<td>K=6</td>
<td>54.5</td>
<td>43.9</td>
<td>0.1873708</td>
<td>0.1788701</td>
<td>0.003396069</td>
<td>0.002843442</td>
</tr>
<tr>
<td>K=7</td>
<td>48.6</td>
<td>37.3</td>
<td>0.1604981</td>
<td>0.1496806</td>
<td>0.005690668</td>
<td>0.002834393</td>
</tr>
<tr>
<td>K=8</td>
<td>35.7</td>
<td>55.8</td>
<td>0.1423903</td>
<td>0.1232845</td>
<td>0.006304031</td>
<td>0.002155966</td>
</tr>
<tr>
<td>K=9</td>
<td>50.4</td>
<td>47.3</td>
<td>0.1121423</td>
<td>0.1080724</td>
<td>0.004296329</td>
<td>0.003102919</td>
</tr>
<tr>
<td>K=10</td>
<td>37</td>
<td>59.2</td>
<td>0.09999715</td>
<td>0.09892117</td>
<td>0.005141721</td>
<td>0.004666739</td>
</tr>
</tbody>
</table>
The most impressive results were obtained on the artificially generated medium overlap and equal sized clusters, where Fuzzy C-means++ was on average 2.1 times faster than the standard (40 times the standard for some particular \( p \) parameter values). On real world datasets we have seen improvements over the standard ranging from dataset 1.12 on the SPAM dataset up to 1.44 on IRIS, when the \( k \) parameter equals true number of clusters in the dataset. We also observe improvements in terms of iterations count on the whole range of \( k \) values.

A limitation of the proposed method is the choice of \( p \) parameter representing the spreading factor. A choice of \( p = 1.8 \) was a good candidate based on the algorithm results on a range of overlapping/non-overlapping, equal/non-equal synthetic datasets. Automatic parameter tuning based on data set properties may allow for even better clustering results and will be the subject of future work. Other future work will parallelize the initialization phase proposed here in combination with the standard part of the algorithm. The parallelized method should enable faster clustering of volatile high dimensional datasets with thousands of instances. We also plan to investigate the problem of finding specialized similarity measures for fuzzy clustering of data in high dimensional spaces.

**Acknowledgment**

Stetco acknowledges the support of an EPSRC doctoral training grant; Keane acknowledges an IBM Faculty Award.

**References**


