Published in final edited form as:

IEEE Trans Fuzzy Syst. 2014 October ; 22(5): 1229–1244. doi:10.1109/TFUZZ.2013.2286993.

Accelerating Fuzzy-C Means Using an Estimated Subsample Size

Jonathon K. Parker [Student Member, IEEE] and Lawrence O. Hall [Fellow, IEEE]

Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, USA

Abstract

Many algorithms designed to accelerate the Fuzzy c-Means (FCM) clustering algorithm randomly sample the data. Typically, no statistical method is used to estimate the subsample size, despite the impact subsample sizes have on speed and quality. This paper introduces two new accelerated algorithms, GOFM and MSERFCM, that use a statistical method to estimate the subsample size. GOFM, a variant of SPFCM, also leverages progressive sampling. MSERFCM, a variant of rseFCM, gains a speedup from improved initialization. A general, novel stopping criterion for accelerated clustering is introduced. The new algorithms are compared to FCM and four accelerated variants of FCM. GOFM's speedup was 4-47 times that of FCM and faster than SPFCM on each of the six datasets used in experiments. For five of the datasets, partitions were within 1% of those of FCM. MSERFCM's speedup was 5-26 times that of FCM and produced partitions within 3% of those of FCM on all datasets. A unique dataset, consisting of plankton images, exposed the strengths and weaknesses of many of the algorithms tested. It is shown that the new stopping criterion is effective in speeding up algorithms such as SPFCM and the final partitions are very close to those of FCM.

Index Terms

fuzzy clustering; scalable; accelerated; sampling; progressive sampling; multinomial proportion; fuzzy c-means; fcm; spfcm; ofcm; effcm; rsefcm; gofcm; mserrfcm; stopping criterion

I. Introduction

Clustering is the process of separating unlabeled objects into groups, where the members in each group have more similarity to each other than the members of other groups. Clustering is an important first step in data exploration and a primary approach for unsupervised learning. The K-means algorithm, also called the hard c-means algorithm (HCM) [1], and its variant fuzzy c-means (FCM) are widely used clustering algorithms [2].

One of the many difficulties with applying clustering algorithms is scalability. On large datasets, the length of time taken to cluster or partition the data can be prohibitive, taking so long that people will only cluster subsets. There are a number of FCM variants that

accelerate the performance of FCM with varying degrees of tradeoff between speed and quality.

This paper introduces two new accelerated fuzzy clustering algorithms, geometric progressive fuzzy c-means (GOFM) and minimum sample estimate random fuzzy c-means (MSERFCM). GOFM has elements similar to single pass fuzzy c means (SPFCM) [3] and progressive sampling [4]. MSERFCM has elements similar to rseFCM [5], [6] and psFCM [7].

Both new algorithms use a statistical method to estimate the subsample size. The subsample is the minimum size necessary to guarantee proportional representation from each different type of data (true cluster) within a specified range of error, at a specified confidence level. A subsample is used to quickly estimate cluster centers, which accelerates the base FCM algorithm by starting closer to a termination point. A novel stopping criterion was also developed that is shown to increase speed with minimal loss in quality.

Experiments showed GOFM was 4 to 47 times faster than FCM and MSERFCM was 4 to 26 times faster than FCM. GOFM consistently outperformed SPFCM with respect to speedup, but suffered a slight loss in fidelity to FCM when clustering with all the data. MSERFCM outperformed rseFCM with respect to speedup and quality in over two thirds of the experiments conducted. A total of six datasets were used; three of the datasets were also used in [5] and results are compared.

II. FUZZY C-means (FCM) based Algorithms

Fuzzy c-means (FCM) [2] and four additional FCM-based algorithms were selected and implemented in code to provide a baseline for comparison to the newly developed algorithms. Many FCM-variants have been created [3], [5], [7]–[16].

FCM and its variants employ fuzzy sets as opposed to classical (i.e. crisp or hard) sets. FCM's advantage is that an object can have a grade of membership in multiple clusters. The degree to which an object k belongs to cluster i is u_{ik} . This relationship is subject to the following constraints [2]:

$$u_{ik} \in [0, 1], 1 \leq i \leq c, 1 \leq k \leq n \quad (1)$$

$$\sum_{i=1}^c u_{ik} = 1, 1 \leq k \leq n \quad (2)$$

$$\sum_{k=1}^n u_{ik} > 0, 1 \leq i \leq c \quad (3)$$

where n is the number of data examples and c is the number of clusters.

The descriptions of FCM and its scalable variants follow in a slightly modified form from previous work [6].

A. Fuzzy c-means (FCM)

This algorithm minimizes an objective function that calculates the within-group sum of squared distances from each data example to each cluster center. FCM alternates between calculating cluster centers given the membership values of each data example and calculating the membership values given the cluster centers. If data examples are defined as feature vectors x_k in R^s , the objective function (J_m) is expressed as:

$$J_m(U, V) = \sum_{i=1}^c \sum_{k=1}^n u_{ik}^m D_{ik}(x_k, v_i) \quad (4)$$

The functions for determining membership values and cluster centers are:

$$u_{ik} = \frac{D_{ik}(x_k, v_i)^{\frac{1}{1-m}}}{\sum_{j=1}^c D_{jk}(x_k, v_j)^{\frac{1}{1-m}}} \quad (5)$$

$$v_i = \frac{\sum_{j=1}^n (u_{ij})^m x_j}{\sum_{j=1}^n (u_{ij})^m} \quad (6)$$

where

n : is the number of examples.

$m > 1$: is the 'fuzzifier'.

c : is the number of clusters.

U : is the membership matrix. u_{ik} refers to the membership value of the k^{th} data example (x_k) for the i^{th} cluster.

V : is the set of cluster centers. v_i is the i^{th} cluster center.

$D_{ik}(x_k, v_i)$: is the squared distance between the k^{th} data example and i^{th} cluster center. Any inner product induced distance metric can be used. This research used Euclidean distance.

There are multiple ways to initialize and terminate the algorithm. Any valid set of values may be used to initialize the U or V matrices. The members of U (u_{ik}) may be initialized with a set of values adhering to (1) and (3) or each member of V (v_i) may be set to some position in R^s . Typically, cluster center (V) initialization is performed by randomly selecting c examples from the dataset. The algorithm terminates when the difference between the calculated matrix norms for successive membership matrices or for cluster center matrices does not exceed a user-provided parameter ϵ .

Fuzzy c-means has been applied to many real world applications, some examples include: magnetic resonance imaging (MRI) data [17] [16], DNA microarray data [18], and power quality management [19]. FCM is a generalization of HCM, and is equivalent to HCM if $m = 1$ [20]; it is also equivalent to Expectation Maximization (EM) [21]. These algorithms also

have been successfully applied to many real world applications [22] [23] [24] [25]. Thus, one could apply FCM successfully to any domain that has had success with HCM or EM.

B. Single pass fuzzy c-means (SPFCM)

The SPFCM algorithm [3] sequentially processes the entire dataset. The data set is broken into equally sized “partial data accesses” (PDA). A user provided parameter, “fractional PDA” ($fPDA = 0.5$), is used to define the PDA size as $fPDA \times n$ where n equals the total number of data examples. Each PDA is processed by a weighted version of FCM, called Weighted FCM (WFCM). In the WFCM algorithm, each data example x_i has an associated weight w_i . The objective function and cluster center calculation are modified as follows [3], [16]:

$$J_{mw}(U, V) = \sum_{i=1}^c \sum_{k=1}^n u_{ik}^m w_k D_{ik}(x_k, v_i) \quad (7)$$

$$v_i = \frac{\sum_{j=1}^n w_j (u_{ij})^m x_j}{\sum_{j=1}^n w_j (u_{ij})^m} \quad (8)$$

Data examples are initially given a weight of 1. After cluster centers V are calculated from the first PDA, the cluster centers are assigned weights using the following equation [16]:

$$w'_i = \sum_{j=1}^n (u_{ij}) w_j, 1 \leq i \leq c \quad (9)$$

These weighted cluster centers represent the partition information of the dataset from the first PDA. The c cluster centers are added as additional data examples to the second PDA, which is then processed by WFCM. The positions of the cluster centers calculated from the first PDA are used as the initial values for V in the second PDA. This is repeated until all PDAs are processed. The set of cluster centers from the final PDA are returned by SPFCM.

The SPFCM algorithm assumes the data in the dataset is randomly ordered. Datasets with some sort of inherent order in the data, which is typical in images, can result in subsets of data that are significantly different with respect to the overall distribution. Our implementation randomizes the data prior to processing.

C. Online fuzzy c-means (OFCM)

The OFCM algorithm is similar to SPFCM, with one major difference [16]. The dataset is broken into PDAs in the same manner as SPFCM and each PDA is processed independently. Cluster centers from each PDA are created using FCM and their weights calculated using (9). Unlike SPFCM, the sets of weighted cluster centers are not added to the next PDA, but saved. After all PDAs are processed, the combined sets of weighted cluster centers are processed by WFCM as a single dataset and a final set of cluster centers returned.

A feature of OFCM is that the processing of a dataset can be separated over distance or time. This is similar in concept to parallel fuzzy clustering, reviewed in [26]. In these cases, the

cluster initialization of each PDA would be performed locally by random selection. In our experiments, we used the cluster centers from the previous PDA as an initialization. While this matches the original implementation of the algorithm [27], a poor initialization will be produced by PDAs largely consisting of just one class. Another feature of OFCM is that the dataset is not assumed to be in random order. In our implementation, we did not randomize the datasets for OFCM.

D. Extensible fast fuzzy c-means (eFFCM)

This algorithm progressively increases the size of a sample of the dataset until a statistically significant sample, compared with the distribution of the full dataset, is obtained. Statistical significance is tested for with the Chi-square (χ^2) statistic or Kullback-Leibler divergence. If the initial sample fails testing, additional data is progressively added to the sample and retested. This is repeated until a statistical test is passed [11], [14]. The size of this additional subsample is constant; the algorithm uses progression with an arithmetic schedule [4]. The final, statistically significant sample is then processed by FCM to obtain a set of cluster centers.

The use of the statistical tests implies that the distribution of the dataset is known. For most datasets the distribution must be calculated prior to running the algorithm. A successful implementation requires decisions concerning how to calculate the distribution, the statistical test to use, the rate of arithmetic progression and the termination criterion [13]. Details of the author's implementation can be found in [6].

E. Random sampling plus extension fuzzy c-means (rseFCM)

This algorithm obtains a random sample of the dataset and FCM is applied to the sample. Once cluster centers are returned from FCM, the cluster membership u_{ik} can be calculated for any example in the dataset.

This algorithm, initially developed in [5] was also reported in [6] as random fuzzy c-means (randFCM). The rseFCM algorithm provides a speedup by reducing the size of the dataset. In this work, the size of the random sample is equal to $fPDA \times n$, where $fPDA = 0.5$ is a parameter defined for SPFCM and OFCM and n equals the total number of data examples.

III. Accelerating FCM

A literal implementation of the FCM algorithm has an expected runtime complexity of $O(nisc^2)$ [3] where n is the number of data examples, c is the number of clusters, s is the dimension of the data and i is the number of iterations. It is possible to reduce the runtime to $O(nisc)$ with an optimization proposed by Kolen and Hutcheson [10]. This and other minor optimizations were implemented. In some previous comparison work these optimizations were not implemented [6].

Techniques that accelerate FCM use the base FCM algorithm or WFCM as a step in a more complex algorithm. Thus, any optimization to the FCM algorithm reduces the runtime for all variants.

Given a dataset with c natural clusters, an FCM variant can be accelerated by reducing n , s or i . The SPFCM and OFCM algorithms have the same runtime complexity as FCM [3], [6] and process 100% of the original data. The initial PDAs processed by SPFCM and OFCM provide better cluster center estimates for the next PDAs. This process will continue with the initial cluster center points having higher weights at each step. Better, weighted, cluster center estimates reduce the number of iterations for subsequent PDAs to reach the termination criterion. In SPFCM, each of the n examples is involved in a smaller number of iterations in the subsample in which it is clustered than when the whole data set is clustered at once. If the average number of iterations i_{avg} for the subsamples is less than i for the full data, an example is used in calculations $n \times i_{avg}$ times (on average) for SPFCM versus $n \times i$ with for FCM $i_{avg} < i$. The time complexity analysis can be reviewed in Section 4.5 of [28]. The eFFCM and rseFCM algorithms provide acceleration by reducing n . In both cases a subsample less than the size of the dataset will be processed. The subsequent acceleration is more or less inversely proportional to the fraction of the data processed [5], [6], [11].

These effects were directly observed in earlier research by the authors [6]. Detailed logs were recorded for each algorithm's progress. These showed that for SPFCM the initial PDAs were responsible for the majority of the total iterations. After a certain point, PDA clustering terminated after one or two iterations, with little change in the estimated cluster centers.

The OFCM algorithm, despite its similarity to SPFCM, did not achieve the same acceleration. The logs suggest a major contributing factor. OFCM sequentially processes non-randomized data. In many datasets, the PDA has a non-proportional representation of clusters, which takes more iterations to terminate. This was demonstrated empirically by comparing the performance of OFCM on non-randomized vs. randomized datasets [6].

The assumption here is that OFCM needs less iterations when the data is more uniform in its distribution. A similar observation was made in previous work when comparing the performance eFFCM and rseFCM. The eFFCM algorithm, which ensures the subsample passes a statistical test prior to processing, occasionally needed less iterations to terminate compared to rseFCM, despite processing a higher percentage of the data [6].

IV. Significant Work Related to Acceleration

The runtime for a hill-climbing algorithm, such as FCM, is reduced if the starting point is close to the final solution. This is due to a reduction in iterations before termination. Bradley and Fayyad [29] investigated the effects of an improved starting position for K-means. A better start position reduced the runtime, but their study was focused on quality, not speed.

Processing a small data sample to obtain an improved initial starting point for FCM has been partially investigated. Cheng, et al. describe an iterative process to develop a “good” starting point [8]. This method, multistage random sampling FCM (mrFCM), consists of two parts. The first part progressively samples the dataset, improving the starting clusters until a termination criterion is met. Then the starting clusters are used to initialize FCM on the full dataset.

Similarly, Altman processes a small sample of data examples with FCM to obtain a set of cluster centers. These are used to initialize the membership matrix U , prior to running FCM on the full dataset [9].

In partition simplification FCM (psFCM), Hung and Yang [7] partition the data using a k-d tree to obtain a simplified dataset, which in turn is used as a subsample to estimate the cluster centers. The resulting estimate is used to initialize FCM on the full dataset.

Provost presented an overview of the progressive sampling technique in the context of induction (aka classification) algorithms [4]. In progressive sampling, an initial subsample is used to form a classifier which is tested on labeled data. The size of the subsample is progressively increased arithmetically or geometrically, with a new classifier created each time the subsample grows in size. When the accuracy of the classifier ceases to improve significantly compared to the previous sample, the technique terminates.

Progressive sampling has been applied to clustering problems. Domingos and Hulten [30] used Hoeffding bounds in a progressive sampling technique to both estimate the initial sample size and estimate if the sample size at any point in the progression was sufficient. The technique was developed for K-means and depends on the fact that in K-means, data elements have membership in only one cluster.

Pal and Bezdek [11] and Wang et al. [15] used progressive sampling for the purpose of selecting a subsample that accurately represents the dataset. A divergence test was used to assess if the subsample matches the distribution of the dataset. If the test failed, progressively larger subsamples were taken until the test passed. Finally a clustering algorithm was run on the chosen subsample.

V. Estimating the Random Sample Size

The scalable clustering approaches discussed explicitly or implicitly sample the dataset to develop a starting point, but typically do not use a statistical method to determine the sample size. Domingos and Hulten's technique [30] estimates the initial sample size using Hoeffding bounds for K-means clustering, but does not directly generalize to FCM [14].

The SPFCM algorithm uses a parameter ($fPDA$) to define a static sample size [3]. The eFFCM algorithm uses a statistical test to validate a subsample, but not to determine the subsample size [11], [14]. The mrFCM algorithm and Altman's method also use a parameter to define the subsample size for cluster center initialization [8], [9]. The size of the subsample influences the performance of the algorithms [6], but the parameters in each case are determined empirically.

As suggested in [29], the subsample must proportionally represent each cluster to provide an improved starting point for a hill-climbing algorithm such as FCM. A known difficulty with using a subsample to generate starting points for a clustering algorithm occurs when the subsample fails to sufficiently represent all clusters [8], [29]. This results in skewed starting points. One solution to this problem is to ensure the subsample has proportional representation. This solution was suggested in [8] but not elaborated on.

Gu et al. studied the effects of an improper starting sample size using progressive sampling on supervised learning problems [31]. They implemented a divergence test on a subsample to ensure it represents the dataset distribution. Pal, Bezdek and Hathaway in [11] and [13], similarly test the subsample for the proportionality of the sample as a whole, but use the sample for calculating the clustering solution rather than estimating clusters as a starting point. Regardless, this sort of technique requires collecting information from the entire dataset to ensure proportionality. A larger subsample could be used for this purpose, but that leaves us with the same uncertainty about the validity of the size of the subsample.

Another approach is to select a probabilistically large enough subsample to sufficiently represent all clusters at a desired level of confidence. If one assumes that the clusters correspond to a set of currently unknown classes, selecting a subsample to sufficiently represent each cluster is analogous to selecting a subsample to estimate a multinomial proportion of classes. This is because if the sample provides an acceptable estimate of a proportion of classes, it will have proportional representation of the clusters in the data.

Thompson [32] came up with a technique to find the smallest sample size, λ , such that a random sample from a multinomial population will result in “class” proportions within a specified distance of the true population proportions with probability at least $1 - \alpha$. It was shown that the minimum sample size, λ , would be:

$$\lambda = \max_{\mu} z^2 (1/\mu) (1 - 1/\mu) / d^2 \quad (10)$$

where d is the maximum absolute difference from the true proportion that will be tolerated for any class. The value z is the upper $(\alpha/2)\times 100$ th percentile of the standard normal distribution.

Thompson showed that μ , an integer, is the number of classes present in the population for which the calculated value of λ is a maximum. For $\alpha = 0.10$ (which we are interested in), the maximum values for λ occur when μ is between 2 and 3. As the clustering problems we are interested in have the number of classes $c \geq 3$, accepting the maximum value for λ allows us to ignore the value of μ . For details, see [32].

Phoungphol and Zhang borrowed Thompson's definition for μ as part of a technique to estimate the sample size for HCM. They implemented a “hard” version of rseFCM where the sample size was estimated with their technique [33].

Solutions to Thompson's formula have been published in tabular form, which pairs desired significance levels (α) with a value for $d^2\lambda$ [32]. An example calculation follows. If the desired significance level is $\alpha = 0.05$, this corresponds to a value of $d^2\lambda = 1.27359$. If the desired maximum absolute difference, $d = 0.02$, the minimum sample size for is $\lambda = \frac{1.27359}{0.02^2} = 3184$. Thus a sample size of 3184 is the minimum to ensure with a 95% probability ($1 - \alpha$) that the maximum absolute difference in class representation is 0.02.

If this method is used to obtain samples for a clustering problem, the total number of classes present must be considered. Assume a full dataset, X , has 5 equally distributed classes. The

true proportion, π , of each class, c , equals 0.20. Using the example above, with $d = 0.02$, a sample size of 3184 is calculated. At the desired significance level, $\alpha = 0.05$, the method predicts with a 95% probability that the sample represents all clusters at the proportion $p = 0.20 \pm 0.02$. This is a suitable proportion for many clustering problems.

If in the example above, X instead has 100 equally distributed classes, $\pi = 0.01$. The absolute difference is still $d = 0.02$, thus the tolerated difference *exceeds* the expected proportion of each class $p = 0.01 \pm 0.02$. In this case the *average* number of examples from each class will be 32 but will *range* from 0 to 96 (with a 95% probability). Thus d must be adjusted in order to ensure that each class is represented with enough examples to be clustered.

Assuming an equal distribution, each class will have a true proportion of $\pi = \frac{1}{c}$. Thompson's formula assumes an *absolute* difference, d . At the level of significance desired, the expected proportion of each class in the sample is $\frac{1}{c} \pm d$. In the examples above, the value of d is kept fixed and the value of c is increased by a factor of 20. This causes the amount of absolute difference allowed in the sample to be greater than the true proportion of the classes: $\pi < d$.

This problem can be repaired by tying the absolute difference to c . Define a value, r , as the “relative difference”. Set $d = \frac{r}{c}$. Now the proportion of each class in the sample is $\frac{1}{c} \pm \frac{r}{c}$, though rewriting it as $\frac{1 \pm r}{c}$ makes it clear why r is defined as the “relative difference”.

Using the assumptions above, $\frac{r}{c}$ can be substituted for d . Now the formula for the desired minimum sample size can be expressed as:

$$\begin{aligned} d^2 \lambda &= v(\alpha) \\ \frac{r^2 \lambda}{c^2} &= v(\alpha) \\ \lambda &= \frac{v(\alpha) c^2}{r^2} \end{aligned} \quad (11)$$

where $v(\alpha)$ is the calculated value (or from Thompson's published table) for a specified α value and the other variables are defined as above.

An example follows. Assume that desired significance level is $\alpha = 0.05$, which corresponds to $v(\alpha) = 1.27359$, the number of clusters, $c = 5$, and the desired relative difference $r = 0.10$.

Using equation (11), the estimated sample size $\lambda = \frac{1.27359 \times 5^2}{(0.1)^2} = 3184$. Note that this is the same result from the example above.

Another example. Keeping the desired significance level at $\alpha = 0.05$ and increasing the desired relative difference to $r = 0.20$, let us find the minimum sample size for $c = 100$.

Using equation (11), the estimated sample size $\lambda = \frac{1.27359 \times 100^2}{(0.2)^2} = 318,398$.

VI. Algorithms Based on Thompson's Method

Insight on how to best leverage Thompson's method for selecting a set of examples comes from understanding how related methods function. Recall that SPFCM functions by

processing one PDA at a time. Analysis of the execution of SPFCM revealed that the number of iterations per PDA fell dramatically after the first few PDAs were processed, resulting in a lower average number of iterations per PDA compared with the total number of iterations for FCM on the full dataset. The majority of PDAs terminated after a single iteration and the changes in the cluster centers were very small. The SPFCM algorithm converged to a solution close to FCM's quickly [3], [6], [28].

The OFCM algorithm has significant similarity to SPFCM, but has poorer performance in regards to speedup and quality when run sequentially [6]. As mentioned above, one reason is that OFCM obtains PDAs by sequentially sampling the data rather than using random sampling. This makes it less likely that the PDA is proportional with respect to class. A PDA largely consisting of a single class will take longer to terminate [6]. While not required to, some implementations of OFCM pass on cluster center estimates from the previous PDA; in the case just described a poor initialization will extend the time before termination [29]. A second reason is that SPFCM stores partition data in the weighted cluster centers returned from each PDA. These in turn are added to the next PDA. The weighting of the cluster centers on average reduces the number of iterations per PDA before WFCM terminates. This was demonstrated in earlier work [6] by comparing OFCM and SPFCM with identical pre-randomized datasets and identical initializations for each PDA.

These insights and the availability of Thompson's formula led to the creation of two algorithms, geometric progressive fuzzy c-means (GOFCM) and minimum sample estimate random fuzzy c-means (MSERFCM). Both of these algorithms use Thompson's formula to estimate an initial sample size for an expected number of clusters. A base assumption, for clustering, is that a dataset processed by these algorithms has the expected number of clusters as reflected in the data attributes. If the data attributes do not provide any distinction between the clusters, the data will not have multinomial properties and Thompson's method is not valid.

A. The GOFCM Algorithm

The GOFCM algorithm is an improvement to SPFCM that leverages progressive sampling, Thompson's method (11) and a different stopping criterion. GOFCM operates like SPFCM except as follows. The initial subsample size is estimated using Thompson's method. The size of subsequent subsamples are calculated using a geometric schedule [4]. The calculated size of the subsample stops growing once it exceeds a user-provided value; once this occurs the subsample size is fixed to equal the limit. In our experimentation, the user-provided value was set to $n \times fPDA$ which is the subsample size used by SPFCM (Section II).

As in SPFCM, each subsample is processed by WFCM. The information from previous subsamples is retained and compressed by weighting the cluster centers from each step of the progressive sampling. The stopping criterion, discussed in detail below, is based on the rate of change (slope σ) of cluster center positions in successive subsamples. The algorithm terminates when the slope rises above a user-defined value.

GOFCM has the same expected runtime complexity as SPFCM, $O(nisc)$ (Section III). In practice, GOFCM will often have a shorter runtime than SPFCM due to a faster convergence

that reduces i and the new stopping criterion that reduces n . See Algorithm 1 for a detailed description of GOFM.

One of the key principles of GOFM is based on the observation that in SPFCM after a certain amount of data is processed, the result does not improve in any appreciable way. This is similar to Provost's observation concerning induction algorithms in [4]. Thus, GOFM may terminate early and without needing to process all the data.

The GOFM algorithm follows a pattern similar to those in Gu et al. [31] and Provost [4] in that multiple subsamples are selected and processed by the base algorithm. GOFM is also similar to methods that use estimation for a better set of starting cluster centers [7]–[9].

There are key differences between GOFM and these similar methods. The first is that GOFM uses Thompson's method to derive the initial subsample size. The second is that GOFM reuses the information from each subsample. This is because the cluster centers obtained from a subsample are weighted, combined with the next subsample, and used as the starting cluster centers. These differences have the benefits of generating an initial cluster center estimate using the minimum amount of sampled data, representing all the data previously processed in weighted clusters and reducing the number of iterations for each subsample until termination [3].

In GOFM, progressively larger subsamples are taken until the stopping criterion is met. The size of the subsamples is controlled by a parameter $\alpha \geq 1$, the geometric schedule factor. If $\alpha = 1$, the subsample size remains constant and the algorithm is similar to SPFCM with a different stopping criterion (see below). As noted in [4], [34], the actual type and rate of scheduling is a tradeoff between cost (as measured in speedup) and benefit (as measured in fidelity to FCM).

The GOFM algorithm is also similar to eFFCM and its variants in that it progressively samples the dataset while retaining the data already sampled. As discussed above, its method of “retaining” the data differs from eFFCM and mirrors SPFCM.

A difficult decision in the development of GOFM was the development of a stopping criterion. Provost identifies detection of convergence, in the context of induction algorithms, as an important area of future research [4]. The same is true for clustering algorithms.

Unlike Provost's method for induction algorithms with labeled data, there is no objective criteria, such as model accuracy, to compare the quality of clustering algorithms. The typical alternative means of developing a stopping criterion is to identify when some metric associated with the algorithm fails to change more than some threshold.

The GOFM stopping criterion is based on a comparison of the positions of the cluster centers (V 's) in successive subsamples. This was studied with a large dataset (MRI-017) known to cluster well with FCM and its variants (Figure 1). The mean distance between successive cluster centers was selected as the norm. While the difference between V 's initially reduced as the amount of data increased, it did not converge to a particular value.

Instead, the algorithm reached a steady state where there was significant variation of cluster center position between subsamples.

A simple thought experiment reveals why this is so. Imagine if a subsample produces the ideal set of cluster centers signifying an extrema for the objective function if all the data were present. Now another, possibly larger, subsample is drawn and the weighted cluster centers from the previous subsample are added. This is a subsample of the remaining data and is extremely likely to have a distribution that differs from the collection of the former subsamples. This will result in the cluster centers deviating between subsamples. The next subsample is drawn from the remaining data, which now has a distribution that differs slightly from the data already sampled.

The FCM algorithm seeks to minimize the objective function from the data that is present. As the distribution of the samples will always be slightly different, the cluster centers will always experience some random variation. Let us consider this random variation as noise.

Note in Figure 1 that the changes in cluster centers ($\delta(V)$) between subsamples are asymptotic when noise is removed. The shape of this curve appears to be the inverse of the learning curves noted by Provost [4] and Meek [34], but the same challenge is present. At what point in the curve should the algorithm be stopped?

If the stopping criterion is defined to be when $\delta(V)$ falls below a user-defined value, the actual value of the change is strongly overwhelmed by the noise. In test experiments using this criterion, a large degree of variability in the final partitions was noticed.

Examination of the dataset in Figure 1 and other datasets showed that this metric generally obeys the Power Law after the first few subsamples. The legend in Figure 1 shows the best fit equation is $y = 0.7324x^{-0.845}$. In Figure 2, the logarithm of the x and y coordinates are plotted. Here the best fit equation is the straight line $y = 0.845x - 0.3114$. Note that $\ln(0.7324) = -0.3114$.

Figure 2 provides a clear view of the noise generated by each subsample and suggests the stopping criterion selected for GOFM.

After each subsample is processed, the logarithm of $\delta(V)$ is saved, and simple linear regression finds the best fit equation. Then the best fit equation is converted back to the original coordinates and the slope between the last two subsamples is found. The best fit line is of the form $y = ax^{-b}$, so the slope will have a range of $(-\infty, 0)$. If this slope rises above a user-defined value (σ), GOFM terminates. Selection of σ is a tradeoff between speed and quality as measured by fidelity to FCM with the full dataset. A small value for σ provides more speed with less quality, while a large value of σ provides higher quality but less speed (more iterations). For experiments, the value of σ was determined empirically.

For a new dataset, one could estimate a suitable value for σ by using GOFM to cluster a small sample of the data using different values for σ and comparing the results to FCM on the same sample. As FCM scales linearly with n , the speedups obtained for different σ could be used to estimate runtimes for GOFM on the full dataset.

Due to noise generated by the subsampling, the algorithm would occasionally prematurely terminate. In order to prevent this from occurring, GOFM was not allowed to terminate until a minimum number of PDAs were processed. This is a similar concept to linear regression with local sampling (LRLS) used by Provost [4]. The minimum number of PDAs to process before calculating the slope is a value that could have been parameterized, but in our implementation the minimum number was set to a constant. We experimented with many values and the number 6 was the lowest minimum value that provided consistent results for the datasets tested.

The minimum number of PDAs can affect the runtime and quality of the algorithm. If set low, GOFM has an early opportunity to terminate, possibly reducing the runtime. However, if the initial data examples do not accurately reflect the dataset, the quality of the results might be poor. If set high, this might increase the runtime, but also makes it more likely the sampled data examples accurately reflect the entire dataset, thus improving quality.

To see how the stopping criterion might work in another setting, we also applied it to SPFCM. We assumed that it would save time at the cost of leaving some of the data unused. Additional experiments were run to compare SPFCM modified with this stopping criterion (MODSPFCM) to SPFCM and GOFM. The results are discussed in Section IX.

B. The MSERFCM Algorithm

Minimum Sample Estimate Random Fuzzy C-Means (MSERFCM) is designed as an improvement to rseFCM. It is also similar to methods that try to find a better set of starting cluster centers [7]–[9], but is much simpler.

The rseFCM algorithm uses c randomly selected data examples as initial cluster centers. In contrast, MSERFCM processes a subsample of the dataset to estimate initial cluster centers. This is the only major difference between rseFCM and MSERFCM unless one of the assumptions (see below) is violated.

For a dataset (X) of size n , a subsample of size (n_1) is estimated using Thompson's method. A subsample (x_1) of size n_1 is drawn without replacement from the dataset and is processed by FCM. The position of the resultant cluster centers are retained. Then a second subsample (x_2) is drawn from X . This second subsample size is specified by the user, which may indicate the amount of available RAM or some other practical concern. In our implementation, the specified (second) subsample size (n_2) of x_2 is defined as $fPDA \times n$, which is the same subsample size used by rseFCM. The retained cluster center positions are used to initialize FCM for x_2 .

The MSERFCM algorithm assumes that the estimated subsample size, n_1 , is less than the specified subsample size, n_2 , and less than the dataset size, n : $n_1 < n_2 < n$. This assumption may not be correct, which can happen when Thompson's method estimates a large value for n_1 . In these cases, MSERFCM will be less efficient than rseFCM. To correct this, the following adjustments are made.

When $n_2 < n_1 < n$, the estimated subsample size exceeds the specified subsample size and MSERFCM degenerates to rseFCM with a subsample of size n_1 . When $n_2 < n < n_1$, the estimated subsample size exceeds the available data and MSERFCM degenerates to FCM and processes the entire dataset. In both of the latter cases, the subsample could exceed memory which would provide the actual limit. In experiments below this did not happen.

MSERFCM has an expected runtime complexity of, $O(n_2 i_2 sc + n_1 i_1 sc)$ due to two successive applications of FCM (Section III). The rseFCM algorithm has an expected runtime complexity of $O(n_2 isc)$. In practice, MSERFCM will usually have a shorter runtime than rseFCM due to the improved set of starting clusters that reduces i_2 enough to more than compensate for the additional $O(n_1 i_1 sc)$ time.

VII. Experiments

GOFCM and MSERFCM were compared in terms of speedup and quality to existing algorithms. The experiments applied FCM and the six accelerated variants described above to four large real-world data sets and two artificial data sets. Cluster centers (V) predicted by the algorithms can vary based on initialization. Each experiment consisted of 30 trials, each initialized by a random selection of c data examples as initial cluster centers. The final cluster centers, runtime, number of iterations and quality metrics were recorded for each trial. The reported experimental results are the averages across these 30 trials.

Our intent was to compare our new accelerated algorithms to the algorithm on which they were based, FCM, and other accelerated variants of FCM. Demonstrating a speedup with minimal differences to the base algorithm should be useful to the clustering community.

Earlier work by Havens, et al. used the same three MRI datasets (see Section VII-B) and four of the same algorithms used in this research. An additional set of experiments, with identical parameters to those used in [5], were done to directly compare results. These experiments consisted of averages from 21 trials, to match what Havens, et al. reported [5].

The three MRI data sets were also used to compare MODSPFCM to FCM, SPFCM and GOFCM.

A. Parameters

As described in Sections II and VI above, the algorithms tested have multiple parameters. The intent of the experiments was to explore accelerating algorithms. Thus for any given dataset, only the parameter affecting the sample size ($fPDA$) was varied; the other parameters were kept fixed. Experimental parameters are summarized in Table I. The series of experiments using MODSPFCM added two additional settings for the $fPDA$ parameter. These settings, which are not listed in Table I, are $fPDA$ in $\{0.02, 0.06\}$.

The value for m , the fuzzifier, is not consistent across the datasets. Initial experiments on the MRI datasets with $m = 2.0$ provided acceptable results, but this was not the case for the other datasets. Some tuning of m was necessary; setting it to a value of 1.7 vastly improved results with respect to runtime and improved fidelity to the cluster centers of the artificial and plankton datasets.

B. Data Sets

Table II contains the details about each of the six datasets used.

Three datasets (MRI016, MRI017, MRI018) were magnetic resonance images (MRI) images of the normal human brain. The three features in these datasets are the intensities of the T1-weighted, T2 weighted and proton density weighted sequences. The values are integers ranging from 0 to 1951. These images were clustered into the three classes: cerebro-spinal fluid (CSF), gray matter (GM) and white matter (WM) [16].

The plankton dataset (PLK01) was a set of image features describing plankton collected by the Shadow Imaging Particle Profiler and Evaluation Recorder (SIPPER) imaging system [35]. This particular set of images was collected during three cruises in the Gulf of Mexico (2002, 2008 & 2010) and two in the Western Pacific (2007 & 2008) [36] [37]. The data was classified by experts at the College of Marine Science at the University of South Florida into a type of marine object (typically a class of plankton).

Initially, there were a total of 482,719 data examples with 88 attributes representing 168 classes. The number of examples in classes ranged from a single example (elongate phytoplankton: chaetoceros) to 99,414 (noise: air bubble). Attempting to cluster the entire dataset with $c = 168$ did not provide stable results. The data was stripped of examples that were noise, detritus and other small and non homogeneous classes and randomized. The attributes were normalized, and feature selection was performed with Consistency Subset Evaluation and Linear Forward Selection [38] using the WEKA data mining tool [39]. These efforts left 203,278 data examples and 21 attributes. Analysis of the cleaned dataset showed 20 predominant classes. Details on the feature selection parameters, plankton attributes and classes available at <http://www.cse.usf.edu/~jkparker/plankton/>.

Two datasets (ART01 and ART02) were artificial datasets. A simple program was written to construct these types of dataset. The program input consists of the locations of c cluster centers in R^s as well as the standard deviation and number of examples for each cluster center. Each data example deviates from its cluster center in each dimension by a random amount determined by a gaussian distribution using the provided standard deviation. The generated examples were then output in random order.

The artificial datasets were designed to moderately challenge FCM and its variants. Figure 3 provides a 3 dimensional view of 3 attributes of ART01. When attributes missing from Figure 3 are considered, the actual Euclidean distance between the true cluster centers provides more separation than is straightforward to display. Despite the crowding, FCM did a reasonably accurate job of correctly predicting the cluster centers of ART01.

C. Metrics

Three metrics were calculated to assess the speed and quality of the algorithms and to serve as a basis of comparison between them.

1) Relative Speedup (SU)—This metric calculates the ratio between the runtimes of two algorithms. If t_1 is the time for algorithm 1 and t_2 is the time for the reference algorithm, the speed up (SU_{12}) of algorithm 1 relative to algorithm 2 is:

$$SU_{12} = \frac{t_2}{t_1} \quad (12)$$

Thus if the runtime of algorithm 1 is 150ms and algorithm 2 takes 750ms, the speed up equals 5; i.e five times as fast.

2) Adjusted Rand Index (ARI)—The ARI [40] returns a value of 1 when two partitions are in complete agreement, 0 when the partitions return the value expected by chance and a negative value when the partitions are in greater disagreement than would be expected from chance.

The ARI is typically used to compare a clustering method's results to the actual class membership of the data examples. In our research, ARI is used to compare the results of an accelerated FCM-variant to the base FCM algorithm. The ARI also assumes that the clustering is discrete, i.e hard [40], [41]. Before calculating the ARI for this research, partitions were hardened by assigning the membership of an example to the cluster for which it had the highest membership value [5] and corresponding clusters from the variant and base algorithms were identified using the Hungarian method [42].

3) Cluster Membership Change (CC)—This metric compares the cluster assignments of two soft clustering algorithms on the same dataset. Small changes in cluster memberships indicate two partitions are very close. If one is the baseline, a small amount of cluster change is a high quality partition when compared to the baseline. The $CC\%$ is used as a relative quality measure for comparing an accelerated algorithm to an FCM partition which is typically an average over t (30 here) trials.

The cluster assignments from each soft algorithm are hardened by setting the highest membership value in each column of U to equal 1. All other values are set to 0. If the cluster assignment for column i is not identical for both algorithms, the indicator variable δ_i is set to 1. Otherwise this value is set to 0. The CC value is then calculated as follows [43]:

$$CC\% = \frac{\sum_{i=1}^n \delta_i}{n} * 100 \quad (13)$$

This metric uses the Hungarian method [42] to identify corresponding clusters across trials and experiments.

VIII. Results

All algorithms except for FCM and OFCM assume that the data is in random order or the implementation performs random sampling. Here, the entire dataset is randomized prior to processing each trial (excepting FCM and OFCM). In addition to recording the runtime of

the algorithm, the software implementation separately recorded the time to randomly sample the data and recorded the time to perform I/O. Unless otherwise noted, reported times and speedup comparisons below include the runtime for randomization and I/O.

It was observed that the results for all MRI datasets were similar, so results for the average of the three MRI datasets are presented in Tables III and IV and Figure 4. While no ground truth existed for the MRI images, the differences in cluster assignment by all methods studied (as measured by $CC\%$ with an FCM partition obtained from the average cluster centers over 30 trials as the baseline) never exceeded 1%. This is far more consistent than human experts, whose assignments have been observed to differ by 16% or more [44] on MRI images of the brain. FCM and its variants have successfully been used in image segmentation [45] [46]. It could be argued that FCM and its variants should be used in preference to human experts for consistency.

For these data sets MSERFCM typically had the highest speedup, rseFCM was often second and GOFM was faster than all others.

Despite the difference in dimension and number of clusters, the results for ART01 and ART02 were similar. Thus, only results from ART01 are presented (Table V and Figure 5). The fidelity to FCM as measured by $CC\%$ was very good. The speed of the algorithms was in the same order as for the MRI datasets.

The results for dataset PLK01 are shown in Table VI and Figure 6. The speedup was most for rseFCM and then GOFM. For small sample sizes the fidelity of all to FCM was poor. For larger sample sizes, the fidelity was fair with rseFCM and MSERFCM having the most fidelity, followed by GOFM and SPFCM.

The quality of GOFM and MSERFCM, as measured by $CC\%$, is very high. The $CC\%$ is less than 0.2% in all experiments, except for those using the PLK01 dataset. The cluster centers returned by FCM itself, will vary by a small % over 30 initializations, making such a small $CC\%$ nearly insignificant.

Assuming the data is pre-randomized increases the speedup considerably. Compare Tables III and IV to see the speedup difference with the MRI datasets. These two results can serve as an upper and lower bound for runtime respectively for FCM and its variants.

The GOFM algorithm provides a consistent speedup over the base FCM algorithm. Depending on the size of the PDA and dataset, the speedup ranged from roughly 4 to 47 times (Table III). Designed as an improvement over SPFCM, it also provides a consistent speedup over SPFCM. On the MRI datasets, the speedup was an average of 2 times faster than SPFCM. On ART01 and ART02, the speedup ranged from 3 to 5 times. GOFM was also consistently faster than SPFCM on PLK01.

If the time taken for randomization and I/O are not considered, GOFM provides a much greater speedup. The speedup on the MRI datasets ranged from 8 to 300 times compared to FCM and a speedup of 2 to 40 times compared to SPFCM (Table IV). Speedups on ART01,

ART02 and PLK01 were even greater, ranging from 10 to over 700 times compared to FCM. See <http://www.cse.usf.edu/~jkparker/gofcm> for additional results.

GOFCM's speedup, however, came with a loss in quality when compared to FCM. Looking at the *CC%* metric, GOFCM had a small loss in quality which ranged from 0.04% to 0.12% on the MRI datasets. SPFCM's quality loss over the same datasets ranged from 0.04% to 0.06%. On PLK01, GOFCM had a *CC%* loss in quality which ranged from 12% to 46%, compared to a 11% to 48% for SPFCM.

The *ARI* metric was specifically calculated for comparison to work by Havens (Section VIII-A), but recorded for all experiments. GOFCM had a consistent, but small loss ($< 1\%$) in quality as measured by *ARI* when compared to SPFCM. The GOFCM algorithm's quality was close to or lower than SPFCM for all datasets. However, the small difference in quality is generally going to result in partitions that are functionally the same.

MSERFCM, designed as an improvement to rseFCM, has slightly superior performance with respect to speed and quality on the MRI datasets. MSERFCM was faster than rseFCM in 80% of the experiments. With respect to quality, MSERFCM was equal to or better than rseFCM on 73%, and 67% of the experiments examining the *CC%* and *ARI* metrics respectively. For all differences of speed and quality, both algorithms were very close - on a few occasions differing by only 0.0001% or less.

On ART01 and ART02, MSERFCM was faster than rseFCM on 60% of the experiments. With respect to quality, MSERFCM was equal to or better than rseFCM on 80%, and 90% of the experiments for the *CC%* and *ARI* metrics respectively. Again, the results were extremely close in all experiments. Despite the differences between ART01 and ART02 with respect to number of clusters and dimensions, the differences between them in relative speedup and result quality across all metrics were negligible.

When randomization and I/O are not considered, MSERFCM was faster than rseFCM on 73% of the MRI experiments and 80% of the experiments with artificial data. Cases where rseFCM was faster than MSERFCM, usually occurred when the PDA was small ($fPDA = 0.001$ or 0.00333333). When this occurred the difference in speed was not trivial and there was a noticeable quality loss for rseFCM.

Results were different with the plankton dataset. Here, rseFCM consistently outperformed MSERFCM with respect to speed, while MSERFCM consistently outperformed rseFCM with respect to quality. In fact, of all six FCM variants, MSERFCM had the best quality over all metrics and all experiments with a generally consistent speedup of 4 times over FCM. The eFFCM algorithm was the only consistent competitor to MSERFCM with respect to quality, with a speedup that ranged from 2.7 to 7.3 times over FCM. An explanation of the very different performance on the plankton data is given in Section IX below.

A. Comparison to work by Havens et al

In "Fuzzy c-Means Algorithms for Very Large Data", experiments were run with rseFCM, SPFCM and OFCM using the same MRI datasets as in our research [5]. Their experiments

differed in several ways. Only 21 trials were performed, the fuzzifier was set to 1.7, and the termination criterion was changed to use the maximum change in V . Havens reports results for SU and ARI; these were rounded to nearest whole number and two decimal digits respectively.

The algorithm implementation was done in MATLAB rather than a Linux/C implementation as in our work. Havens pre-randomized the files, and did not count that step in the algorithm execution time, but did consider sampling and I/O time [47]. Our experiments included the time to randomize the files and perform I/O in the algorithm execution time. It was not possible to make the runtime results perfectly comparable, as our reported results include more overhead. As a consideration to make the experimentation as close as possible, our code was modified to pre-randomize files for OFCM. Results from Havens' experiments and from our experiments are presented in a format as identical as possible to that presented by Havens [5]. Comparison results and additional results from GOFM and MSERFCM are listed in Table VII.

For the same sets of parameters, the OFCM and rseFCM results are fairly similar for both groups. Judging from the speedup for SPFCM however, there appears to be a significant difference in the implementations. The authors suspect it is due to the additional time taken to perform randomization. Regardless, the algorithms have the same order with respect to speed: OFCM, SPFCM, rseFCM. This suggests that a MATLAB implementation of GOFM and MSERFCM similar to that in [5] would show the speedup for GOFM higher than SPFCM and the speedup for MSERFCM (on average) faster than rseFCM.

B. Comparison of MODSPFCM to SPFCM and GOFM

The speedup of MODSPFCM (Table VIII) and the quality as measured by $CC\%$ (Figure 7), when applied to the MRI data sets, falls between that of SPFCM and GOFM. In the case of speedup, MODSPFCM approaches that of GOFM as the sample rate, as measured by $fPDA$, becomes smaller. In the case of quality, GOFM maintains a slight edge over MODSPFCM at all but the lowest sample rates. In addition to the five experiments with the $fPDA$ values listed in Table I, two experiments were run with $fPDA = 0.02$ and $fPDA = 0.06$ to add evidence to the observed trends.

IX. Discussion

A. GOFM vs. Related Methods

Why is GOFM faster compared to SPFCM and other related methods? There are two main reasons: the estimated subsample size and the stopping criterion. Recall that the runtime complexity of GOFM is affected by n , i , s and c . If we are comparing the performance of two algorithms on the same dataset, assuming s and c to be constant makes the comparative runtime complexity $O(ni)$. At the beginning of the GOFM algorithm, n is the (presumably small) estimated subsample size, but initialization of the cluster centers is random. Thus the number of iterations, i , is usually large. At the second subsample there are more data examples, but the cluster center initialization is improved which requires fewer iterations. The GOFM algorithm experiences an accelerated performance because when n is small, i

is large and as n increases, i decreases. This keeps the runtime more consistent across subsamples processed, a demonstration of how subsample size and cluster center initialization impact speed.

The second reason GOFCM is faster is its stopping criterion. It stops processing data when the predicted cluster centers do not show a high degree of change. Related methods process all available data.

There is a tradeoff between speed and quality. The effect of this tradeoff is evident when GOFCM and eFFCM are compared. The eFFCM algorithm often had better quality than GOFCM, but much lower speedups (Table III, Fig. 4). The GOFCM algorithm selects a starting subsample aiming to have the number in each cluster within the specified range. This does not guarantee that the range of attribute values in each cluster is proportionally represented in the subsample. This is one difference between GOFCM and techniques that perform a divergence test on subsamples against the sample distribution [11] [15].

The process of performing a divergence test on a subsample is time consuming. The entire sample must be analyzed for ranges of values, bins must be selected, and the subsample's values must be assigned bins. This technique is also subjective, as there is no optimal way to select the bins or parameters. Analysis of an implementation of eFFCM found that 5%-42% of the dataset was sampled before a Chi-squared test was passed; these tests were performed on relatively simple data sets [11]. The author's implementation found that 0.2%-34.6% of the test datasets were sampled before the Chi-squared test was passed.

In contrast, GOFCM determines a starting subsample size via a lookup table and a simple equation. This step, while less precise, is much faster.

GOFCM's quality is controlled by the stopping criterion parameter (σ). The consistent setting for σ in the experiments provided a consistent speedup of GOFCM over SPFCM with a consistent loss in quality. While not explored, a stricter setting for σ should have resulted in a smaller speedup and higher quality. The converse would also apply.

Another limitation on GOFCM's quality is when the dataset requires a larger subsample than that allowed by the maximum subsample size ($fPDA \times n$). In these cases, GOFCM is forced to predict cluster centers with a suboptimal sample. This is clearly demonstrated in Section IX-D below.

The experiments with MODSPFCM revealed the role the stopping criterion plays in GOFCM's speedup. MODSPFCM can be considered SPFCM with GOFCM's stopping criterion or GOFCM without progressive sampling. Table VIII shows how MODSPFCM's speedup falls between that of SPFCM and GOFCM. When the $fPDA$ is a comparatively larger number (0.1, 0.06, 0.033333), the advantage of GOFCM's subsampling is clearly shown. As the $fPDA$ becomes smaller, the speedup of GOFCM and MODSPFCM approach the same value. The example below explains why this is so.

Assume an experiment with GOFCM and MODSPFCM where $fPDA = 0.001$. On the MRI datasets, using Equation 11 with the parameters from Table I, the initial subsample size for

GOFCM is about 1100 data examples. The MRI datasets have roughly 4×10^6 data examples each; when $fPDA = 0.001$ the initial subsample size for MODSPFCM is about 4000 data examples. Recall that the geometric scheduling parameter for GOFCM is set to 2.0 and in our experiments the maximum subsample size is set by $fPDA$. Thus by the third PDA, the scheduled subsample size exceeds the maximum and the PDA sizes for both algorithms are the same. In this case, GOFCM only has an advantage of using a smaller n for the first two PDAs, after which both algorithms process the same amount of data with the same stopping criterion. If the subsample size decreases below that calculated by Equation 11, GOFCM “degenerates” to MODSPFCM.

B. Artificial Datasets and OFCM

As mentioned in Section (VIII), experiments with the artificial datasets (ART01, ART02) had very similar results. Thus only results for ART01 are displayed (Table V, Fig. 5). The results for GOFCM and MSERFCM using these datasets are not radically different than those for the MRI datasets. The only surprise is the performance of OFCM.

The speedup of OFCM on ART01 (Table V) is consistently the lowest. This is also the case on the MRI datasets (Table III). The quality of OFCM, as measured by $CC\%$, is radically different on the artificial datasets (Fig. 5) compared with the MRI datasets (Fig. 4). In fact, OFCM has the highest quality of any algorithm on ART01.

Recall that OFCM processes data in order; the algorithm does not randomly sample the data. The difference in quality in this case is due to the fact that the artificial datasets were randomized prior to experimentation. Naturally occurring datasets, such as an MRI scan do not have a random ordering with respect to cluster.

Pre-randomizing the artificial data made it more likely each sample processed by OFCM was proportional. In the worst case, with $fPDA = 0.001$, the subsample size for ART01 is 1000. Using the formula from [32], a sample size of 1000 corresponds to a maximum absolute difference of 0.03 when $\alpha = 0.05$. It so happens that the 5 true clusters in ART01 each account for 20% of the total. Thus a sample size of 1000 will result in each true cluster consisting of 17-23% of the total (with a 95% probability).

The 5 cluster centers calculated from each PDA do a fairly reliable job of representing the 1000 data examples, with very little skewing due to the “uniform effect” [48]. In the last step of OFCM, the collection of weighted cluster centers are processed by WFCM. The resultant cluster centers, as measured by $CC\%$, had the best fidelity with the full FCM algorithm.

The difference in speedup of OFCM using non-randomized vs. pre-randomized data was studied in [6]. The speedup of OFCM on MRI datasets, was reported to improve from 48% to 76% of SPFCM's speedup when the dataset was pre-randomized.

This suggests that both speed [6] and quality of OFCM can be improved by an efficient method of randomly sampling the data.

C. MSERFCM

MSERFCM, designed as an improvement to rseFCM, was the fastest or second fastest algorithm in all experiments for the MRI and artificial datasets (Tables III, IV, V).

The occasions when MSERFCM's speedup was less than rseFCM was when the subsample size was small. This occurred with the MRI datasets and $fPDA = 0.001$. MSERFCM initially used a sample size of 1147 (Eq. 11 with $c=3$) to calculate the starting cluster centers for executing FCM with a subsequent subsample size of about 4000 examples. So in these cases, MSERFCM processed over 5000 examples, compared with rseFCM processing about 4000 examples. The speedup gained from a better set of starting cluster centers was not enough to compensate for the increased amount of data processed.

When ART01 was processed with the parameter $fPDA = 0.001$, the estimated sample size to predict cluster centers was 3184. As this exceeded the subsample size, the MSERFCM algorithm degenerated to rseFCM with a subsample size of 3184. This explains the faster performance of rseFCM, which used a subsample size of 1000.

With respect to quality, MSERFCM was a better choice than rseFCM the majority of the time. This is expected in situations when MSERFCM processed a larger subsample size than rseFCM. In the example above, MSERFCM processed a random subsample three times larger, making it far more likely MSERFCM will have more fidelity to FCM. This situation occurred in 6 of the 25 experiments using the MRI and artificial datasets. Of the remaining 19 experiments, one would *expect* no difference in *quality*. While MSERFCM draws a separate subsample to calculate the set of starting cluster centers, both MSERFCM and rseFCM use the exact same size subsamples to calculate the output set of cluster centers as they proceed.

This however was not the the case. Of the remaining 12 MRI experiments (of 30 trials each), ignoring ties, MSERFCM had better average quality than rseFCM in 69% and 63% of the experiments for the CC% and ARI metrics respectively. Of the remaining 7 artificial dataset experiments, ignoring ties, MSERFCM had better quality than rseFCM in 60% and 67% of the experiments for the CC% and ARI metrics respectively.

FCM is guaranteed to converge to a local (or global) minimum or saddlepoint [49]. One possibility is that the calculated starting cluster centers for MSERFCM allow the discovery of better local minima than the randomly determined starting cluster centers for rseFCM. This explanation is supported by the results of related experiments involving K-means [29].

D. Plankton Dataset Challenges

The use of the plankton dataset (PLK01) shows the limitations and capabilities of the algorithms tested. Note that the speedup compared to FCM (Table VI) for PLK01 is greater than the speedup for MRI (Table III) except in that case of MSERFCM. The MSERFCM algorithm has a consistent, but small speedup, just barely greater than that of OFCM.

This is because of the subsample size rules for MSERFCM. Thompson's method (Eq. 11) calculates a minimum subsample size of 50,944 for PLK01. Thus in all experiments,

MSERFCM degenerates to rseFCM with a subsample size of 50,944. In contrast, the largest subsample size for the other algorithms is 20,328 when $fPDA = 0.1$.

Note that the MSERFCM algorithm has the best quality (Fig. 6) and is the only one to have consistent quality across the experiments. The eFFCM algorithm has fairly consistent quality, but eFFCM has a mechanism to dynamically increase its sample size if quality measures are not met. The average subsample sizes for eFFCM on PLK01 ranged from 10.7% to 34.6%. All other algorithms suffer a degradation of quality as the subsample size is reduced. This is attributable to a suboptimal sample size. The quality, as measured by $CC\%$, make the results unusable for many applications of clustering when trying to use very small subsamples.

The quality of the MSERFCM algorithm's results demonstrates the utility of using a statistical method for determining subsample size.

The PLK01 dataset also provides a nice demonstration of GOFCM's operation. As with MSERFCM, the predicted minimum subsample size is greater than the size calculated ($fPDA \times n$). What happens in this case is that GOFCM degenerates to use the same, consistent subsample as SPFCM.

In this case, the only difference between SPFCM and GOFCM is the stopping criterion. SPFCM will always process 100% of the data, GOFCM will stop short of that if the stopping criterion is met. An examination of Table VI shows that SPFCM and GOFCM have a similar speedup when the $fPDA$ (sample rate) is 0.1, with GOFCM's speedup increasing and quality decreasing compared to SPFCM as the $fPDA$ decreases.

Observe the $CC\%$ of SPFCM, OFCM and GOFCM when the $fPDA = 0.001$ (Fig. 6). Note that the values are very similar. When $fPDA = 0.001$, the subsample size is only 204. Experimentation was kept consistent across all datasets; in reality there is no reason for such a small sample. Recall that $c = 20$ for PLK01. The cluster representation for PLK01 is not uniform and the sample size is small, making it unlikely that all clusters are even represented in each subsample. The base FCM algorithm will still try to fit 20 clusters, which will likely result in predicted cluster centers unrepresentative of the full dataset.

Finally, we note that these approaches can be applied to FCM with different distance metrics [50], [51].

X. Conclusions

In this work we introduced two significant contributions to the study of accelerated clustering algorithms: (a) use of an estimated subsample size and (b) a novel stopping criterion. These new contributions were demonstrated with two accelerated versions of FCM that leverage a statistical method to estimate the subsample size. These algorithms, GOFCM and MSERFCM, also use sound principles from other accelerated algorithms, namely progressive sampling and improved cluster center initialization. The novel stopping criterion was applied to GOFCM and shown to be the most significant factor in producing a speedup

over FCM when the subsample size was small. This led to the development of the MODSPFCM algorithm which is faster than SPFCM.

The majority of the time, the new algorithms were improvements over similar existing algorithms in terms of speed and/or quality. GOFM provided a clear advantage in processing time over SPFCM and the loss of quality due to processing less data was kept to a minimum. On average, MSERFCM edged out rseFCM on both speedup and quality. Here, quality is defined by the percentage of cluster change when compared to an average partition produced by FCM. The changes were typically quite small indicating high quality.

The plankton and artificial datasets demonstrated the value and limitations of statistical methods. Experimental parameters resulted in many algorithms using suboptimally sized subsamples with a corresponding loss in quality. GOFM was designed to conform to the user-provided parameters despite a suboptimal subsample, which resulted in speedup at the expense of quality. MSERFCM was designed to reject user-provided parameters if a suboptimal subsample would result, which resulted in quality at the expense of speedup.

Two techniques applied in this work are novel to accelerated clustering; Thompson's statistical method and the stopping criterion for GOFM. The manner in which Thompson's statistical method was used is only one possible application. Given that there are numerous accelerated FCM variants, this statistical method can potentially be applied to improve speedup and quality of results for many of them. The stopping criterion designed is a simple solution to the general problem of estimating a changing, noisy value. It enables MODSPFCM, which is a very good highly scalable clustering algorithm. This technique could be applied to other existing clustering algorithms as a means of improving speedup with a minimum loss in quality. Future work will involve discovery of additional applications for these new techniques.

Acknowledgment

This research was partially supported by by grant 1U01CA143062-01, Radiomics of NSCLC from the National Institutes of Health. The authors would like to thank Tim Havens for providing details on his research in scalable algorithms. The authors would like to thank Kurt Kramer for providing the plankton dataset and answering follow up questions.

Original implementations of FCM by Steven Eschrich [12] and SPFCM and OFCM by Prodip Hore [28] were reviewed. Some implementation techniques were adopted, but the code was entirely rewritten. Initialization parser code was obtained from an implementation by Hoyt [52]. The χ^2 calculation needed for eFFCM was performed with code from [53].

Biography

Jonathon K. Parker is a Doctoral Candidate in the Department of Computer Science and Engineering at the University of South Florida. He received an M.S. in Computer Science from Bowie State University in 1997 and a B.S. in Chemistry from the University of Wisconsin-Madison in 1988.

His interests include data mining, data clustering, machine learning, sentiment analysis and designing software applications.

Lawrence O. Hall is a Distinguished University Professor and the Chair of the Department of Computer Science and Engineering at University of South Florida. He received his Ph.D. in Computer Science from the Florida State University in 1986 and a B.S. in Applied Mathematics from the Florida Institute of Technology in 1980.

He is a fellow of the IEEE. He is a fellow of the AAAS and IAPR. He received the Norbert Wiener award in 2012 from the IEEE SMC Society. His research interests lie in distributed machine learning, extreme data mining, bioinformatics, pattern recognition and integrating AI into image processing. The exploitation of imprecision with the use of fuzzy logic in pattern recognition, AI and learning is a research theme.

He has authored or co-authored over 70 publications in journals, as well as many conference papers and book chapters. Some recent publications appear in the IEEE Transactions on Fuzzy Systems, IEEE Transactions on Pattern Analysis and Machine Intelligence, Neural Computation, Information Fusion, , IEEE Transactions on Systems, Man, and Cybernetics, Pattern Recognition, the International Conference on Pattern Recognition, the Multiple Classifier Systems Workshop, and the FUZZ-IEEE conference (<http://isl.csee.usf.edu/ailab/hall.html>).

He received the IEEE SMC Society Outstanding contribution award in 2008. He received an Outstanding Research achievement award from the Univ. of South Florida in 2004. A past president of NAFIPS. The former vice president for membership of the SMC society. He was the President of the IEEE Systems, Man and Cybernetics society for 2006-7. He was the Editor-In-Chief of the IEEE Transactions on Systems, Man and Cybernetics, Part B, 2002-05. He is the Vice President for Publications of the IEEE Biometrics Council. Also, associate editor for IEEE Transactions on Fuzzy Systems, International Journal of Intelligent Data Analysis, the International Journal of Pattern Recognition and Artificial Intelligence and International Journal of Approximate Reasoning.

REFERENCES

1. Lloyd SP. Least squares quantization of pcm. *IEEE Transactions on Information Theory*. Mar; 1982 28(2):129–137.
2. Bezdek, JC. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press; 1981.
3. Hore P, Hall LO, Goldgof DB. Single pass fuzzy c means. *IEEE International Conference on Fuzzy Systems. FUZZ-IEEE*. Jul.2007 :1–7.
4. Provost F, Jensen D, Oates T. Efficient progressive sampling. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*. 1999;1999:23–32.
5. Havens TC, Bezdek JC, Leckie C, Hall LO, Palaniswami M. Fuzzy c-means algorithms for very large data. *IEEE Trans. Fuzzy Systems*. Dec; 2012 20(6):1130–1146.
6. Parker JK, Hall LO, Bezdek JC. Comparison of scalable fuzzy clustering methods. *Fuzzy Systems (FUZZ)*, 2012 IEEE International Conference on. Jun.2012 :1–9.
7. Hung MC, Yang DL. An efficient fuzzy c-means clustering algorithm. *Proceeding of the 2001 IEEE International Conference on Data Mining (ICDM'01)*. 2001:225–232.
8. Cheng T, Goldgof D, Hall L. Fast fuzzy clustering. *Fuzzy Sets and Systems*. Jan; 1998 93(1):49–56.
9. Altman D. Efficient fuzzy clustering of multi-spectral images. *Proc. IEEE IGARSS*. 1999; 3:1594–1596.
10. Kolen J, Hutcheson T. Reducing the time complexity of the fuzzy c-means algorithm. *Fuzzy Systems, IEEE Transactions on*. Apr; 2002 10(2):263–267.
11. Pal NR, Bezdek JC. Complexity reduction for "large image" processing. *IEEE Trans. Syst. Man Cybern*. Oct; 2002 32(5):598–611.
12. Eschrich S, Ke J, Hall LO, Goldgof DB. Fast accurate fuzzy clustering through data reduction. *IEEE Trans. Fuzzy Systems*. 2003; 11:262–269.
13. Bezdek JC, Hathaway RJ. Progressive sampling schemes for approximate clustering in very large data sets. *IEEE International Conference on Fuzzy Systems*. Jul.2004 1:15–21.
14. Hathaway RJ, Bezdek JC. Extending fuzzy and probabilistic clustering to very large data sets. *Computational Statistics and Data Analysis*. Mar.2006 5:215–234.
15. Wang L, Bezdek JC, Leckie C, Kotagiri R. Selective sampling for approximate clustering of very large data sets. *International Journal of Intelligent Systems*. Mar; 2008 23(3):313–331.
16. Hore P, Hall LO, Goldgof DB, Gu Y, Maudsley AA, Darkazanli A. A scalable framework for segmenting magnetic resonance images. *J. Sign. Process Syst*. 2009; 54:183–203.
17. Ahmed MN, Yamany SM, Mohamed N, Farag AA, Moriarty T. A modified fuzzy c-means algorithm for bias field estimation and segmentation of mri data. *Medical Imaging, IEEE Transactions on*. 2002; 21(3):193–199.
18. Dembele D, Kastner P. Fuzzy c-means method for clustering microarray data. *Bioinformatics*. 2003; 19(8):973–980. [PubMed: 12761060]
19. Biswal B, Dash PK, Panigrahi BK. Power quality disturbance classification using fuzzy c-means algorithm and adaptive particle swarm optimization. *Industrial Electronics, IEEE Transactions on*. 2009; 56(1):212–220.
20. Pal NR, Bezdek JC. On cluster validity for the fuzzy c-means model. *Fuzzy Systems, IEEE Transactions on*. 1995; 3(3):370–379.
21. Alpaydm E. Soft vector quantization and the em algorithm. *Neural Networks*. 1998; 11(3):467–477. [PubMed: 12662823]
22. Jain AK. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*. Jun; 2010 31(8): 651–666.
23. de Pasquale F, Cherubini A, Péran P, Caltagirone C, Sabatini U. Influence of white matter fiber orientation on $r2^*$ revealed by mri segmentation. *Journal of Magnetic Resonance Imaging*. 2013; 37(1):85–91. [PubMed: 22987315]
24. Bailey TL, Elkan C. Fitting a mixture model by expectation maximization to discover motifs in bipolymers. 1994

25. Carson C, Belongie S, Greenspan H, Malik J. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 2002; 24(8):1026–1038.
26. Coletta LF, Vendramin L, Hruschka ER, Campello RJ, Pedrycz W. Collaborative fuzzy clustering algorithms: Some refinements and design guidelines. *Fuzzy Systems, IEEE Transactions on*. 2012; 20(3):444–462.
27. Hore P, Hall L, Goldgof D, Cheng W. Online fuzzy c means. *Annual Meeting of the North American Fuzzy Information Processing Society*. May.2008 :1–5.
28. Hore, P. Ph.D. dissertation. University of South Florida; Jun. 2007 Scalable frameworks and algorithms for cluster ensembles and clustering data streams.
29. Bradley PS, Fayyad UM. Refining initial points for k-means clustering. *Microsoft Research, Tech. Rep*. May.1998 MSR-TR-98-36.
30. Domingos P, Hulten G, Edu P, Edu C. A general method for scaling up machine learning algorithms and its application to clustering. In *Proceedings of the Eighteenth International Conference on Machine Learning*. 2001
31. Gu B, Liu B, Hu F, Liu H. Efficiently determining the starting sample size for progressive sampling. *Proceedings of the European Conference on Machine Learning*. 2001:192–202.
32. Thompson SK. Sample size for estimating multinomial proportions. *The American Statistician*. Feb; 1987 41(1):42–46.
33. Phoungphol P, Zhang Y. Sample size estimation with high confidence for large scale clustering. *Proceedings of the 3rd International Conference on Intelligent Computing and Intelligent Systems*. 2011
34. Meek C, Theisson B, Heckerman D. The learning curve sampling method applied to model based clustering. *The Journal of Machine Learning*. Mar.2002 2:397–418.
35. Samson S, Hopkins T, Remsen A, Langebrake L, Sutton T, Patten J. A system for high-resolution zooplankton imaging. *IEEE Journal of Oceanic Engineering*. Oct; 2001 26(4):671–676.
36. Remsen A, Samson S, Hopkins T. What you see is not what you catch: a comparison of concurrently collected net, optical plankton counter and shadowed image particle profiling and evaluation recorder data from the northeast gulf of mexico. *Deep Sea Research Part I: Oceanographic Research Papers*. Jan; 2004 51(1):129–151.
37. Kramer, K. Ph.D. dissertation. University of South Florida; Oct. 2010 System for identifying plankton from the sipper instrument platform.
38. Liu H, Setiono R. A probabilistic approach to feature selection-a filter solution. *Machine learning: proceedings of the Thirteenth International Conference (ICML'96)*. Jul.1996 :319–327.
39. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The weka data mining software: an update. *SIGKDD Explor. Newsl*. Nov; 2009 11(1):10–18. [Online]. Available: <http://doi.acm.org/10.1145/1656274.1656278>.
40. Hubert L, Arabie P. Comparing partitions. *Journal of Classification*. 1985; 2:193–218.
41. Rand WM. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*. Dec; 1971 66(336):846–850.
42. Kuhn HW. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*. Mar; 1955 2(1-2):83–97.
43. Gu Y, Hall LO, Goldgof DB. Evaluating scalable fuzzy clustering. *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*. Oct.2010 :873–880.
44. Clarke LP, Velthuizen RP, Clark M, Gaviria J, Hall LO, Goldgof D, Murtagh R, Phuphanich S, Brem S. Mri measurement of brain tumor response: comparison of visual metric and automatic segmentation. *Magnetic resonance imaging*. 1998; 16(3):271–279. [PubMed: 9621968]
45. Chuang K-S, Tzeng H-L, Chen S, Wu J, Chen T-J. Fuzzy c-means clustering with spatial information for image segmentation. *computerized medical imaging and graphics*. 2006; 30(1):9–15. [PubMed: 16361080]
46. Wang Z, Song Q, Soh YC, Sim K. An adaptive spatial information-theoretic fuzzy clustering algorithm for image segmentation. *Computer Vision and Image Understanding*. 2013
47. Havens TC. 2012 private communication.

48. Liang J, Bai L, Dang C, Cao F. The k-means-type algorithms versus imbalanced data distributions. *IEEE Trans. Fuzzy Systems*. Aug; 2012 20(4):728–745.
49. Bezdek JC, Hathaway RJ, Sabin MJ, Tucker WT. Convergence theory for fuzzy c-means: Counterexamples and repairs. *IEEE Transactions on Systems, Man, and Cybernetics*. Sep-Oct; 1987 SMC-17(5):873–877.
50. Wu J, Xiong H, Liu C, Chen J. A generalization of distance functions for fuzzy c-means clustering with centroids of arithmetic means. *IEEE Transactions on Fuzzy Systems*. 2012; 20(3):557–571.
51. Huang H-C, Chuang Y-Y, Chen C-S. Multiple kernel fuzzy clustering. *Fuzzy Systems, IEEE Transactions on*. 2012; 20(1):120–134.
52. Hoyt, B. inih: simple .ini parser in c. Open Source Project. [Online]. 2011. Available: <http://code.google.com/p/inih/>
53. Reiter, B.; Aquino, J. Statist 1.4.2. Open Source Project. [Online]. 2009. Available: <http://wald.intevation.org/projects/statist/>

Figure 1.
Changes in V for GOFM and dataset MRI017

Figure 2.
Log of change in V for GOFCM and dataset MRI017

Figure 3.
ART01 (3 of 6 attributes of 5 classes)

Figure 4.

Figure 5.

Figure 6.

Figure 7.
Cluster center change comparison of MODSPFCM with SPFCM and GOFCM.

TABLE I

Experiment Parameter Settings

Parameter	Value
trials	30 (21)
m	2.0 (1.7) (MRI) 1.7 (Art and PLK01)
termination criterion	max change in U (max change in V)
ε	0.001
$fPDA$	0.1, 0.0333333, 0.01, 0.00333333, 0.001
α (eFFCM)	0.200
$dPDA$ (eFFCM)	20% of the value of $fPDA$
σ (GOFCM)	-0.01
α (GOFCM)	0.05
a (GOFCM)	2.0
r (GOFCM)	0.1

Parameter values modified to match [5] are in parentheses.

MRI, Art and PLK01 refer to particular datasets.

TABLE II

Datasets

Dataset	examples (n)	attributes (s)	clusters (c)
MRI016	3,882,771	3	3
MRI017	3,898,407	3	3
MRI018	4,293,292	3	3
PLK01	203,278	21	20
ART01	1,000,000	6	5
ART02	1,000,000	10	7

TABLE III

MRI Speedup

<i>fPDA</i>	0.100	0.033	0.010	0.003	0.001
OFCM	1.39	1.55	1.32	1.04	1.05
SPFCM	3.01	3.70	4.12	4.16	4.17
eFFCM	2.08	2.12	1.93	1.55	2.51
GOFCM	4.46	6.56	8.60	9.11	9.51
MSERFCM	6.59	8.27	9.54	9.39	9.53
rseFCM	4.79	7.25	9.14	9.36	9.61

Bold type indicates fastest speedup for each *fPDA*

TABLE IV

MRI Speedup (Ignoring Randomization and I/O)

<i>fPDA</i>	0.100	0.033	0.010	0.003	0.001
OFCM	1.40	1.56	1.32	1.04	1.05
SPFCM	4.43	6.25	7.36	7.74	7.86
eFFCM	3.36	4.22	5.65	9.25	227.98
GOFCM	8.42	20.63	53.55	122.97	301.24
MSERFCM	19.15	50.94	119.70	313.54	599.68
rseFCM	9.13	28.24	85.79	241.59	575.05

Bold type indicates fastest speedup for each *fPDA*

TABLE V

Art01 Speedup

<i>fPDA</i>	0.100	0.033	0.010	0.003	0.001
OFCM	2.76	2.85	2.74	2.46	2.23
SPFCM	3.61	5.00	5.74	5.94	5.91
eFFCM	3.91	5.52	7.08	7.65	12.46
GOFCM	9.83	15.63	21.26	24.00	26.78
MSERFCM	12.50	20.21	25.02	25.32	26.16
rseFCM	6.87	14.88	23.35	26.07	27.64

Bold type indicates fastest speedup for each *fPDA*

TABLE VI

Plankton Speedup

<i>fPDA</i>	0.100	0.033	0.010	0.003	0.001
OFCM	2.84	2.75	1.75	1.47	1.11
SPFCM	6.93	13.58	18.77	21.73	22.83
eFFCM	2.74	4.08	5.45	7.29	6.28
GOFCM	8.19	18.29	33.04	41.12	47.95
MSERFCM	4.00	4.03	4.04	4.04	4.04
rseFCM	8.52	21.55	38.39	46.48	49.57

Bold type indicates fastest speedup for each *fPDA*

TABLE VII

Comparison with Havens' Results

(a) $fPDA = 0.001$														
Dataset		MRI016			MRI017			MRI018						
Researcher		Havens	Parker		Havens	Parker		Havens	Parker		Havens	Parker		
Metric	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI
OFCM	2	0.78	2	1.00	2	0.78	1	1.00	2	0.85	2	1.00		
SPFCM	16	0.96	8	1.00	14	0.97	7	1.00	12	1.00	7	1.00		
GOFCM			25	1.00			20	1.00			21	0.99		
rseFCM	29	0.97	26	0.99	25	0.97	21	0.99	22	0.97	22	0.99		
MSERFCM			26	0.99			21	0.99			23	1.00		

(b) $fPDA = 0.01$														
Dataset		MRI016			MRI017			MRI018						
Researcher		Havens	Parker		Havens	Parker		Havens	Parker		Havens	Parker		
Metric	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI
OFCM	2	0.93	2	1.00	2	0.78	2	1.00	2	1.00	2	1.00	2	1.00
SPFCM	13	0.96	7	1.00	11	0.98	6	1.00	10	1.00	6	1.00	6	1.00
GOFCM			18	1.00			15	1.00			16	0.99		
rseFCM	24	1.00	22	1.00	21	0.99	18	1.00	18	1.00	19	0.99		
MSERFCM			24	1.00			20	1.00			20	0.99		

(c) $fPDA = 0.1$														
Dataset		MRI016			MRI017			MRI018						
Researcher		Havens	Parker		Havens	Parker		Havens	Parker		Havens	Parker		
Metric	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI
OFCM	3	1.00	2	1.00	2	1.00	2	1.00	2	1.00	2	1.00	2	1.00
SPFCM	7	0.96	4	1.00	6	0.97	3	1.00	5	1.00	4	1.00	4	1.00
GOFCM			6	1.00			5	1.00			5	1.00	5	1.00

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

(c) $pDA = 0.1$													
Dataset		MRI016				MRI017				MRI018			
Researcher		Havens	Parker			Havens	Parker			Havens	Parker		
Metric	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	SU	ARI	ARI
rscFCM	8	1.00	8	1.00	8	1.00	7	1.00	6	1.00	7	0.99	
MSERFCM			10	1.00			9	1.00			9	0.99	

TABLE VIII

MODSPFCM MRI Speedup

<i>fPDA</i>	0.100	0.060	0.033	0.020	0.010	0.003	0.001
SPFCM	3.01	3.38	3.70	3.85	4.12	4.24	4.13
MODSPFCM	3.40	4.51	5.88	6.84	8.26	9.41	9.41
GOFCM	4.40	5.44	6.65	7.32	8.58	9.52	9.44

Bold type indicates fastest speedup for each *fPDA*

Algorithm 1

GOFCM Algorithm

Provide dataset X with n examples. Select values for parameters: number of clusters c , fuzzifier m , termination criterion ε , geometric schedule factor $a = 1$, maximum slope σ and maximum fraction of data to sample $fPDA$. Select the relative difference r and the significance level α for use with Thompson's method. Set $t = 1$.

Calculate the initial subsample size, n_1 of dataset X using Thompson's method.

Create the initial subsample (x_1) by randomly selecting n_1 data examples from X without replacement.

Process x_1 with WFCM

Retain weighted clusters V_1

repeat

$t = t + 1$

Calculate new subsample size $n_t = a \times n_{t-1}$

if $n_t > n \times fPDA$ **then**

$n_t = n \times fPDA$

end if

Create subsample (x_t) by randomly selecting n_t data examples from X without replacement.

Add the c weighted cluster centers V_{t-1} to x_t .

Process x_t with WFCM

Retain weighted clusters V_t

Calculate the change of cluster centers between subsamples, $\delta(V_t, V_{t-1})$.

Save $\ln(\delta(V_t, V_{t-1}))$ in a buffer.

if $t > 6$ **then**

Calculate the slope σ_t

else

$\sigma_t = \sigma$

end if

until $\sigma_t > \sigma$

return V_t