

Provided for non-commercial research and educational use only.  
Not for reproduction or distribution or commercial use.

This article was originally published in a journal published by Elsevier, and the attached copy is provided by Elsevier for the author's benefit and for the benefit of the author's institution, for non-commercial research and educational use including without limitation use in instruction at your institution, sending it to specific colleagues that you know, and providing a copy to your institution's administrator.

All other uses, reproduction and distribution, including without limitation commercial reprints, selling or licensing copies or access, or posting on open internet sites, your personal or institution's website or repository, are prohibited. For exceptions, permission may be sought for such use through Elsevier's permissions site at:

<http://www.elsevier.com/locate/permissionusematerial>



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

ScienceDirect

Computational Statistics & Data Analysis 51 (2006) 215–234

COMPUTATIONAL  
STATISTICS  
& DATA ANALYSIS

[www.elsevier.com/locate/csda](http://www.elsevier.com/locate/csda)

# Extending fuzzy and probabilistic clustering to very large data sets

Richard J. Hathaway<sup>a,\*</sup>, James C. Bezdek<sup>b</sup>

<sup>a</sup>Department of Mathematical Sciences, Georgia Southern University, Statesboro, GA 30460, USA

<sup>b</sup>Department of Computer Sciences, University of West Florida, Pensacola, FL 32514, USA

Available online 2 March 2006

## Abstract

Approximating clusters in very large (VL = unloadable) data sets has been considered from many angles. The proposed approach has three basic steps: (i) progressive sampling of the VL data, terminated when a sample passes a statistical goodness of fit test; (ii) clustering the sample with a literal (or exact) algorithm; and (iii) non-iterative extension of the literal clusters to the remainder of the data set. Extension accelerates clustering on all (loadable) data sets. More importantly, extension provides feasibility—a way to find (approximate) clusters—for data sets that are too large to be loaded into the primary memory of a single computer. A good generalized sampling and extension scheme should be effective for acceleration and feasibility using any extensible clustering algorithm. A general method for progressive sampling in VL sets of feature vectors is developed, and examples are given that show how to extend the literal fuzzy (*c*-means) and probabilistic (expectation-maximization) clustering algorithms onto VL data. The fuzzy extension is called the generalized extensible fast fuzzy *c*-means (geFFCM) algorithm and is illustrated using several experiments with mixtures of five-dimensional normal distributions.

© 2006 Published by Elsevier B.V.

**Keywords:** Clustering; Data mining; Extensibility; Fuzzy *c*-means; Goodness-of-fit; Progressive sampling; Very large data sets

## 1. Introduction

Huber (1996) classifies data set size as in Table 1. Huber states that “Some simple standard database management tasks with computational complexity  $O(n)$  or  $O(n \log n)$  remain feasible beyond terabyte monster sets, while others (e.g., clustering) blow up already near large data sets.” We have added one column to Huber’s table, called very large (VL) data.

Today data of more than 10 gigabytes is probably beyond the primary memory capacity of most workstations. Different computers can handle different maximally sized data sets, and their capacity will continue to increase, but so will data size. There will always be data sets that are simply too large for any computer, so methods that are extensible to VL data sets are of continued importance (Cutting et al., 1992; Baeza-Yates and Ribeiro-Neto, 1999).

The data set to be clustered is either  $X_L$  or  $X_{VL}$ . These two clustering situations are shown in Fig. 1, where  $X_\infty$  denotes the population from which the data is drawn;  $X_{VL}$  represents a very large data set that cannot be loaded into primary memory;  $X_L$  represents a large data set that can be loaded into primary memory; and  $X_{SS}$  represents a subset (or subsample) of either  $X_{VL}$  or  $X_L$ . In a nutshell, the proposed fuzzy (geFFCM: generalized extended fast

\* Corresponding author. Tel.: +1 912 681 5619; fax: +1 912 681 0654.

E-mail addresses: [rhathaway@georgiasouthern.edu](mailto:rhathaway@georgiasouthern.edu), [r.hathaway@ieee.org](mailto:r.hathaway@ieee.org) (R.J. Hathaway).

Table 1  
Size of data sets, after Huber (1996)

Bytes “size”	$10^2$ tiny	$10^4$ small	$10^6$ medium	$10^8$ large
	$10^{10}$ huge	$10^{12}$ monster	$10^{n>12}$ VL	$\infty$ Infinite

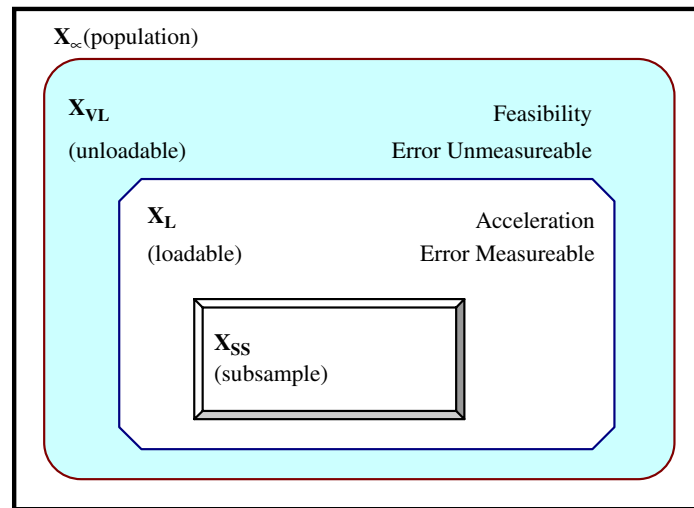


Fig. 1. Population  $X_\infty$  and samples  $X_{VL}$ ,  $X_L$ ,  $X_{SS}$ .

fuzzy  $c$ - means) and probabilistic (geFEM: generalized extended fast expectation maximization) algorithms choose  $X_{SS}$ , cluster it, and then extend the result to  $X_L$  or  $X_{VL}$ .

There are some fundamental differences between the two cases. Our test for judging whether  $X_{SS}$  is representative of the source sample depends on being able to process the full sample. We can load  $X_L$  into primary memory and do the required processing (to test the subsample). We cannot load  $X_{VL}$ , but it is often feasible to page through  $X_{VL}$  once to gather simple statistics (e.g., bin counts for a histogram) needed to assess the quality of candidate subsamples.

Another fundamental difference between the two cases is the calculation of approximation error. We call the application of any algorithm to an entire data set a literal implementation. (The machine learning community terminology is (unsupervised) learning with all the examples.) In this paper FCM refers to the popular fuzzy  $c$ -means clustering algorithm, which we will also sometimes denote as LFCM (literal FCM) to distinguish the exact (literal) implementation of FCM from approximations to it. If the available data set is  $X_L$ , we can assess the quality of an extended clustering by comparing it to the literal clustering obtained using the same parameters on the whole data set. But if the set is  $X_{VL}$ , then a quality comparison is not possible because the literal clustering (using all the examples) cannot be obtained. Our confidence in the accuracy of geFFCM (generalized extended fast expectation maximization) in the unverifiable case ( $X_{VL}$ ) is based on its verified good behavior for various  $X_L$  experiments.

Scalability is often cited as a qualification for clustering algorithms for VL data. The usual definition of scalability: an algorithm is scalable if its runtime complexity increases linearly with the number of records in the input data (Ganti, 1999a). Scalability is often confused with a related issue—viz., acceleration of existing algorithms. Indeed, the  $c$ -means algorithms (Bezdek et al., 1999) are all scalable in the just defined sense, but are famously slow when processing lots of samples, so scalability alone is not enough. And while we are always on the lookout for ways to make clustering algorithms faster, no amount of acceleration solves the VL data problem, wherein the data cannot be processed in aggregate at all.

Progressive sampling is used in various ways for both  $c$ -means and other clustering approaches. This interesting field has seen a lot of growth in recent years. Provost et al. (1999) provide a very readable analysis and summary of progressive sampling schemes. They assert that the central component of any progressive sampling scheme is the

sampling schedule  $S = \{n_1, n_2, \dots, n_k\}$ ,  $n_i$  denotes sample size at step  $i$ , and  $i < j \Rightarrow n_i < n_j \leq N$ , where  $N$  is the cardinality of the whole data set ( $X_L$  or  $X_{VL}$ ). These authors have done a careful study of several issues, including efficient schedules, termination detection, and adaptive scheduling. Their study shows that progressive sampling is more efficient than literal implementations over many different circumstances. Another recent paper that is closely related to our work was written by Meek et al. (2002), and that approach is specifically targeted to using samples for expectation-maximization (EM) clustering. These authors advocate the use of a learning curve associated with the expected gain to be realized by moving to the next sample size in the schedule, against the expected loss of the additional runtime it takes to get the next point on the learning curve.

An important difference between the approaches discussed in Provost et al. (1999) and Meek et al. (2002) and ours is that in our approach clustering is never exercised until an adequate sample is selected. Then we make but one run (a single run, as opposed to single pass methods discussed in the next paragraph) of the algorithm on the selected sample, and rely on extensibility to approximate a solution on the remaining examples. By contrast, the authors of Provost et al. (1999) and Meek et al. (2002) assume the algorithm in use will be run on each sample in the schedule until progressive sampling terminates, and the algorithmic output on the terminal sample is the final result (the unprocessed samples remain that way). Nonetheless, we will use some results from these two important papers to partially justify the geFFCM sampling schedule that is defined in Section 4.

Single pass algorithms use all of the VL data sequentially. Single pass methods begin by randomly drawing a largest subset  $X_1$  from  $X$  that can be loaded into a buffer/memory.  $X_1$  is clustered, summarized (typically by the cluster centers at (4) or the current value of  $J_1$ ) and then utilized by one of many possible strategies. Removal of  $X_1$  frees up space in the buffer/memory, more points from  $X$  are loaded, and the procedure is repeated until  $X$  is exhausted. Conceptually, single pass methods divide  $X$  into  $q$  subsets,  $X = X_1 \cup X_2 \cup \dots \cup X_q$ ,  $X_k = |n_k|$  for all  $k$ , where ( $q$ ) depends on the depletion method. The hard  $c$ -means objective function guides the construction of clusters  $U_k$  for  $X_k$ ,  $k = 1, \dots, q$ , and the final partition of  $X$  is made by concatenation of the  $q$   $c \times n_k$  matrices  $\{U_k\}$ , so  $U = [U_1|U_2|\dots|U_q]$  (one problem with this approach is that  $U$  may or may not itself be a valid partition of  $X$ ). Some methods in this group depend on the cluster centers to retain “historical structure” from subset to subset, while others just start afresh with each update of the buffer.

Depletion can be partial, as in the method due to Bradley et al. (1998a, b) and BIRCH (Zhang et al., 1996; Ganti et al., 1999b); or it can be complete, as in the recent work of Farnstrom et al. (2000). The results of Farnstrom et al. show that single pass methods can be as reliable as the literal version of hard  $c$ -means on data sets small enough to be processed in a single pass. The principle drawback to this group of methods is that the final clusters are built up from “sub-clusters” found by processing each of the  $X_j$ 's in turn. No attempt is made to construct the  $X_j$  so that each  $X_i$  is representative of the overall structure of  $X$ . While random selection of  $X_i$  from  $X$  may overcome this defect, it is easy to see that misrepresentative sampling leads to poor clusters. Nonetheless, single pass methods hold great promise.

Ng and Han (1994) populate the nodes of a graph with random samples that are crisp partitions of  $X_{VL}$ , and then search through the graph for the optimal crisp clusters in  $X_{VL}$ . This method does not alter the number of clusters ( $c$ ), the dimensionality of the data ( $s$ ) or the sample size ( $N$ ); its efficiency for VL data lies in cutting down the time it takes to search the set of crisp  $c \times N$  partitions. Another sample-based scheme is the hierarchical algorithm CURE (Guha et al., 1998), which can discover clusters of arbitrary shapes. The method of Fayyad and Smyth (1996) is in some ways similar to the method we propose. These authors draw a random subsample from  $X_{VL}$ , find probabilistic clusters in it, and finally, extend the clustering to all of  $X_{VL}$  by finding “residual” points in the rest of  $X_{VL}$  that do not fit the current clusters very well. If the clusters are not good enough (too many residual points), the process is repeated with a different random subsample.

The progressive subsampling method that evolved into extensible fast fuzzy  $c$ -means (eFFCM) was introduced by Uma Shankar and Pal (1994), and was called fast fuzzy  $c$ -means (FFCM). FFCM generates a sequence of extended partitions of the full data set by applying LFCM to a nested sequence of larger and larger subsamples (which do not satisfy any criterion of goodness). FFCM terminates as soon as successive extended partitions differ by less than a given threshold. The multistage random sampling fuzzy  $c$ -means (mrFCM) procedure of Cheng et al. (1995) is similar to eFFCM. In the first stage, successively larger subsamples of  $X_L$  are processed by LFCM to find terminal memberships and prototypes, which are then used to initialize a final run. Stage 2 of mrFCM runs LFCM on  $X_L$  using the improved initial guess, which hopefully reduces the iteration time needed by LFCM on all of  $X$ . Because mrFCM runs LFCM on the whole data set, eFFCM is faster than mrFCM. In the experiments discussed (Pal and Bezdek, 2002), the average

speedup factor (over LFCM) for mrFCM is about 1.6, as compared to about 4.2 for eFFCM. Moreover, mrFCM does not offer a means for approximating partitions in  $X_{VL}$ .

Other approaches to accelerate FCM include the bit reduction fuzzy  $c$ -means (brFCM) procedure of Eschrich et al. (2003), which quantizes and aggregates the  $X_L$  into a smaller data set before LFCM is applied. The brFCM approach attained excellent average speedup factors (over LFCM) of 59–290 on discrete data corresponding to magnetic resonance and infrared images. This method is distantly related to the approximate fuzzy  $c$ -means (AFCM) algorithm (Cannon et al., 1986), which replaces certain arithmetic computations by table lookups. Kolen and Hutcheson (2002) reduce the storage and time requirements of LFCM with an implementation that eliminates most of the variables involved in the LFCM iteration.

The last  $c$ -means method mentioned here is actually a modification of crisp  $c$ -means, but it is related to eFFCM in that it employs a progressive sampling scheme that attempts to assess the adequacy of a given subsample. Domingos and Hulten (2001) derive an approximation to the difference in the cluster means obtained by the current subsample and those that could be obtained with perfect (population) information. This approximation is then used to determine when a sufficiently good subsample is identified. Analysis for the crisp case is not easily generalized to the fuzzy case; this ambitious undertaking has not as yet been done.

Section 2 discusses  $c$ -means clustering and extensibility. Section 3 discusses progressive sampling in general (non-image) data. Each of the changes to the eFFCM scheme is exemplified and analyzed by clustering in various subsamples of a mixture of two bivariate normal distributions. Section 3 culminates with the generalized eFFCM (geFFCM) algorithm. Section 4 contains several examples of geFFCM applied to samples drawn from mixtures of five-dimensional normal mixtures. Also, we demonstrate that the progressive subsampling method is applicable beyond FCM by coupling it with the EM algorithm for normal mixtures to obtain a probabilistic clustering scheme demonstrated in the final numerical example. Section 5 contains our conclusions and additional comments.

## 2. Clustering and extensibility

Pal and Bezdek (2002) developed the extensible fast fuzzy  $c$ -means clustering (eFFCM) algorithm for segmentation of VL digital images. eFFCM begins with progressive sampling which terminates with a subsample of pixels that is representative of all pixels in the image. Next, LFCM is applied to the set of feature vectors corresponding to the subsample. Finally, the extensibility property of FCM is used to extend the clusters to the rest of the image. The extended partition is obtained at a fraction of the computational cost required to obtain the LFCM partition. Examples in Pal and Bezdek (2002) show that an extended partition of image data can be a very accurate approximation to the literal partition. Our objective is to generalize eFFCM so that it can be used for any VL data set. Specifically, we aim to: (1) accelerate LFCM for  $X = X_L$ ; and (2) provide feasibility: for clustering when  $X = X_{VL}$ .

Object data clustering is the partitioning of a data set  $X = \{x_1, \dots, x_N\} \subset R^s$ . The  $j$ th coordinate of  $x_i$  is called the  $j$ th feature of  $x_i$ . We acknowledge the importance of selecting a good value for  $c$ —the number of clusters in  $X$ —before looking for them (tendency assessment); and for verifying the utility of the clusters after finding them (cluster validation). But we do not consider these two problems here. Interested readers can see Jain and Dubes (1988) or Bezdek et al. (1999) for a variety of analytical approaches for determining  $c$ . Bezdek and Hathaway (2002) discuss visual approaches to tendency assessment based on displays of reordered distances between pairs of data points in  $X$ .

The fuzzy  $c$ -means model represents clusters in  $X$  by a set of cluster centers  $V = \{v_1, \dots, v_c\} \subset R^s$  and a non-degenerate  $c \times N$  fuzzy  $c$ -partition matrix  $U$  that satisfies three conditions:

$$u_{ik} \in [0, 1] \quad \forall i, k, \quad (1a)$$

$$\sum_{i=1}^c u_{ik} = 1 \quad \forall k, \quad (1b)$$

$$\sum_{k=1}^N u_{ik} > 0 \quad \forall i. \quad (1c)$$

Here  $u_{ik}$  is the degree of membership of  $x_k$  in cluster  $i$ . The set of cluster centers  $V = \{v_1, \dots, v_c\}$  gives the approximate locations of the  $c$  cluster centers in  $R^s$ . Optimal  $(U, V)$  pairs are found as extreme points of the FCM objective function

$$J_m(U, V) = \sum_{i=1}^c \sum_{k=1}^N u_{ik}^m \|x_k - v_i\|^2, \quad (2)$$

where  $m > 1$  is the fuzzification constant and  $\|\cdot\|$  is any inner product induced norm. Users of the FCM model assume that  $(U, V)$  pairs that produce small values of  $J_m$  in (2) usefully identify and represent clusters in  $X$ . The LFCM algorithm is an instance of alternating optimization (Bezdek and Hathaway, 2003) that attempts to minimize (2) by initializing either the  $U$  or  $V$  variables, and then iterating between update equations for the unknowns, which are based on the first order necessary conditions for extrema of  $J_m$ ,  $m > 1$ :

$$v_i = \frac{\sum_{k=1}^N u_{ik}^m x_k}{\sum_{k=1}^N u_{ik}^m} \quad \forall i, \quad (3)$$

$$u_{ik} = \left( \sum_{j=1}^c \left( \frac{\|x_k - v_i\|^2}{\|x_k - v_j\|^2} \right)^{1/(m-1)} \right)^{-1} \quad \forall i, k. \quad (4)$$

Iteration through (3) and (4) terminates if a maximum number of iterations ( $T$ ) has been reached; or when the distance between successive iterates, as measured by  $\|U_{\text{current}} - U_{\text{previous}}\|$  or  $\|V_{\text{current}} - V_{\text{previous}}\|$  is less than or equal to some stopping tolerance  $\varepsilon$ . Bezdek et al. (1999) give a fairly complete discussion of LFCM and many of its relatives. It is our experience that LFCM iteration always produces an approximate minimizer of (2) for real data sets.

A useful notion for processing very large data sets called extensibility is introduced in Pal and Bezdek (2002). Roughly speaking, an extensible algorithm is one that can produce an approximate result (e.g., a cluster analysis) for a full sample by first solving one subproblem using a subsample, and then non-iteratively extending the subsample result to obtain (approximate) results for the full sample.

Not all algorithms are extensible, and among those that are, extension is not necessarily beneficial. An extensible algorithm is efficiently extensible if it runs faster than its literal parent. Since the time complexity of the approximator and its literal parent are the same, the efficiency of an extensible algorithm as defined here is a computationally established property, not a theoretical one. Our numerical experiments suggest that LFCM is efficiently extensible.

### 3. The geFFCM algorithm

For image data each pixel  $p_k$  is associated with two values,  $(g_k, x_k)$ , where  $g_k$  is the intensity of the  $k$ th pixel. The intensity data  $\{g_1, \dots, g_N\}$  guides the subsample selection in eFFCM, but in the case of non-image data, preservation of the intensity distribution cannot be used to guide sampling; there is only a multivariate datum  $x_k$  for each object  $o_k$ .  $X = \{x_1, \dots, x_N\}$  is the direct source for the representative subsample  $X_{SS}$ . The essential modification to eFFCM replaces intensity-based subsample selection with a more general scheme.

Generalized eFFCM differs from eFFCM in four ways. geFFCM uses: (1) subsample selection and enhancement using sampling without replacement; (2) subsample selection based on various subsets of all data features; (3) histogram bins of unequal width derived from the initial subsample; and (4) only one statistical test (the divergence test). We first state the geFFCM algorithm. Then we discuss differences between geFFCM and eFFCM, as well as other issues relevant to our general approach.

#### 3.1. Generalized eFFCM (geFFCM)

*Step 1:* For LFCM choose: the number of clusters  $c$ , fuzzification constant  $m > 1$ , inner product norm  $\|\cdot\|$  on  $R^s$ , stopping tolerance  $\varepsilon$ , and maximum number of iterations  $T$ . geFFCM uses these parameters, and also, choices of:  $a$ , the required number of features for which  $X_{SS}$  and  $X$  must agree, the initial set of features (i.e., feature indices) available for testing  $I_0 \subseteq \{1, 2, \dots, s\}$ , simultaneous or cumulative acceptance (SA or CA), the  $s$  numbers of bins  $r_1, \dots, r_s$ , the

$s$  significance levels  $\alpha_1, \dots, \alpha_s$ , the initial sample percentage  $p$ , and the incremental sample percentage  $\Delta p$ . Initialize the current set of available features  $I$ , by  $I = I_0$ .

*Step 2:* Get  $X_{SS}$  by randomly drawing  $(pN)/100$  times from  $X$  without replacement. For each currently available feature  $x_j$  (i.e.,  $j \in I$ ): use the appropriate order statistics based on  $X_{SS}$  to define histogram bins of approximately equal size  $n_{jb} \sim N/r_j$  for all  $b = 1, \dots, r_j$ , where  $r_j$  is the number of bins used for feature  $j$ , and use the bins to calculate the corresponding histogram bin counts  $N_{j1}, N_{j2}, \dots, N_{jr_j}$  for  $X$ .

*Step 3:* Compute  $D_j$ , the symmetric Kullback–Leibler divergence in (5), for each index  $j \in I$  using the bin counts  $n_{j1}, \dots, n_{jr_j}$  for the current subsample  $X_{SS}$ , where  $p_{jb} = N_{jb}/N$  and  $q_{pb} = n_{jb}/n$ :

$$D_j = n \left( \sum_{b=1}^{r_j} p_{jb} \ln \left( \frac{p_{jb}}{q_{jb}} \right) + \sum_{b=1}^{r_j} q_{jb} \ln \left( \frac{q_{jb}}{p_{jb}} \right) \right) = n \sum_{b=1}^{r_j} (p_{jb} - q_{jb}) \ln \left( \frac{p_{jb}}{q_{jb}} \right). \quad (5)$$

*Step 4:* For each  $j \in I$ , compare the corresponding  $D_j$  value with  $F^{-1}(1 - \alpha_j)$ , where  $F$  is the cdf for the chi-square distribution with  $(r_j - 1)$  degrees of freedom, and if  $D_j \leq F^{-1}(1 - \alpha_j)$ , then delete  $j$  from  $I$  (i.e., delete  $j$  if  $D_j \leq F^{-1}(1 - \alpha_j)$ ). (Feature  $j$  is deleted from  $I$  if there is good agreement between  $X_{SS}$  and  $X$  in this feature.)

*Step 5:* If the number of features for which  $X_{SS}$  and  $X$  agree ( $=|I_0| - |I|$ ) is at least as large as the required number  $a$ , then go to Step 6. Otherwise: add to  $X_{SS}$  an extra  $\min\{(\Delta pN)/100, |X| - |X_{SS}|\}$  random draws without replacement from  $X - X_{SS}$ , set  $I = I_0$  in case of simultaneous acceptance (SA), and go to Step 3. (Simultaneous acceptance requires the most current subsample itself to agree with  $X$  in the required number  $a$  or more components.)

*Step 6:* Apply FCM to  $X_{SS}$ , obtaining the pair  $(U_{X_{SS}}, V_{X_{SS}})$ .

*Step 7:* Use  $V_{X_{SS}} = \{v_{iss} : 1 \leq i \leq c\}$  with (4) to compute  $U_{\text{ext}} = U_{X_{SS}} \cup U_{X - X_{SS}}$  on  $X$ .

$$u_{ik} = \left( \sum_{j=1}^c \left( \frac{\|x_k - v_{iss}\|^2}{\|x_k - v_{jss}\|^2} \right)^{1/(m-1)} \right)^{-1} \quad \forall i, k. \quad (4')$$

*Step 8:* (Optional, for testing). When  $X = X_L$ , run LFCM on  $X_L$ , get the partition  $U_{\text{lit}}$ , harden  $U_{\text{ext}}$ , and  $U_{\text{lit}}$ , and compute the approximation and training errors.

### 3.1.1. Discussion: the sampling schedule

The first issue is the sampling schedule  $S = \{n_1, n_2, \dots, n_t\}$ . Our schedule,  $S_a = \{p + (\tau \cdot \Delta p) : \tau = 0, 1, \dots, n_t\}$ , is arithmetic, whereas  $S_g = \{\lambda^\tau \cdot p : \tau = 0, 1, \dots, n_t\}$  with  $\lambda \in \{2, 3, \dots\}$  is a geometric schedule. For example, choosing  $\lambda = 2$  results in  $S_g = \{p, 2p, 4p, \dots\}$ . The number  $n_t$  of subsamples that are drawn in either case is a function of the input data and the method of termination.

Provost et al. (1999) make an interesting analysis of some general differences between arithmetic and geometric sampling schedules. In particular, they show that geometric sampling is more efficient than arithmetic sampling unless the algorithm involved is linear in  $N$  and more than about  $N/6 \sim 16\%$  of the samples are needed before  $n_k$  is attained. FCM is linear in  $N$ , viz.,  $\mathbf{O}(c^2 pN)$ , and, as we shall see, the only sampling strategy we use that terminates using less than 16% of  $X$  is the least stringent strategy (called “first accept”)—a strategy that led us to poor results. From this standpoint, we feel that an arithmetic schedule is well justified.

In addition to the difference in the type of sampling schedule (arithmetic vs. geometric), there are three other important differences between geFFCM and the algorithms discussed by Provost et al. (1999). Their work is done in the context of classifier design (induction algorithms), whereas FCM is a clustering method. And much more importantly, their analysis assumes that the algorithm of choice will (i) be run on each sample in the schedule, whereas FCM is not invoked until progressive sampling terminates, and then run only once and (ii) in contrast to geFFCM, once the induction algorithm is run on the terminal sample, no extension to the rest of the data is made. So, there are some striking differences between the underlying assumptions in Provost et al. (1999) and our approach. Nonetheless, Provost et al. (1999) is an excellent account of the important issues in progressive sampling, and we recommend it highly to readers interested in this topic.

### 3.1.2. Discussion: the termination criterion

eFFCM used the chi-squared and divergence (Eq. (5)) tests. Are they both needed? Fig. 3 in the next section shows what typically happens, which is that the two tests yield qualitatively identical results. Because of this redundancy, we chose to use only the divergence test, which has the slight practical advantage of not requiring bin combination when expected counts in histograms are too small. While equal-content bins insure that the observed counts in each bin are not too small, it is still possible (but not very probable) that the expected count in one or more bins is less than 5. Moreover, although we used a statistical function to terminate progressive sampling, we view it merely as a measure of goodness of fit. Indeed, we believe that comparing the vectors  $\mathbf{p} = (p_{j1}, \dots, p_{jr_j})$  and  $\mathbf{q} = (q_{j1}, \dots, q_{jr_j})$  of values used in (5) with any norm to a termination threshold  $\varepsilon$ ,  $\|\mathbf{p} - \mathbf{q}\| \leq \varepsilon$ , could be adapted to serve well.

### 3.1.3. Discussion: sampling with/without replacement

Pal and Bezdek (2002) sampled with replacement. Our intuition (rightly) tells us that we should expect more (on average) from a larger subsample than a smaller one, but the selection procedure used by Pal and Bezdek is not biased towards easier acceptance of larger subsamples. This appears to make the subsample selection procedure potentially sensitive to the initial subsample size. One possible adjustment is to make the significance level  $\alpha$  a decreasing function of subsample size  $n$ , which will make it relatively easier for a larger subsample to be accepted. (This is called an adaptive schedule by Provost et al. (1999).) This strategy, however, creates the additional problem of determining how quickly to decrease  $\alpha$ .

A second, easier-to-implement adjustment is to build  $X_{SS}$  by sampling without replacement from  $X$ . A consequent of this choice is that (5) is no longer approximately chi-squared for large subsamples  $X_{SS}$ . On the other hand, the divergence is used only to help identify a highly representative subsample, and not as a conventional hypothesis test, where it is typically important to have a firm bound on the error of false rejection. Computing and comparing the divergence in our situation (without replacement) with the appropriate chi-square percentage point still gives us a useful, normalized measure to assess the quality with which  $X_{SS}$  represents  $X$ .

### 3.1.4. Discussion: feature selection

We assume that all features of  $X$  are equally important for subsample selection. Here the objective (and meaning) of feature selection is to find features that signal a good subsample of  $X$ . This is different from the usual meaning of feature selection in pattern recognition (i.e., the selection of feature subsets that contain “enough” information for cluster analysis or classifier design).

The simplest method is to test a subsample of each feature, and accept it as soon as any one of the  $s$  features passes the test (i.e., when the divergence lies in the extreme left part of the appropriate chi-square distribution). At the other extreme, we may choose to wait until all of the features are accepted in the same subsample before terminating the subsample enhancement process. Or we may terminate using a selection of features in between these two extremes. geFFCM allows precise control of the selection strategy. We will study subsample quality as a function of the selection strategy in numerical examples in the next section.

### 3.1.5. Discussion: bin selection

For eFFCM, Pal and Bezdek (2002) subdivided the range of univariate (intensity) data  $\{g_1, \dots, g_N\}$  into  $r$  bins of equal width, and then combined bins when the expected or actual bin counts were too small. For images, the bin range is known without computation because the gray scale is known, but in general, finding the exact range of feature values in  $X$  would itself require a pass through the entire data set—extra computation we would like to avoid.

Alternatively, we can base the bin definition on the data from a subsample. We propose to use order statistics for each feature of the data in the initial subsample  $X_{SS}$  to define bins of equal content, rather than equal width. By equal content, we mean that the number of points in  $X_{SS}$  corresponding to each bin is approximately the same (or exactly the same, if the order statistics are distinct and  $|X_{SS}| \div r$  is an integer). We refer to this as an equal content or EC histogram. For each feature this strategy places many (narrow) bins in areas dense with data and fewer (wide) bins in areas with little data. Statistical tests recommend against having too few actual or expected counts in each bin. The EC approach circumvents this problem without the need to find and combine low count bins.

Fig. 2 illustrates EC bin selection using one feature (say feature  $j$ ) for  $|X_{SS}| = 500$  and  $r = 5$ . The bins in Fig. 2, from left to right, are:  $(-\infty, x_{(101)})$ ,  $[x_{(101)}, x_{(201)})$ ,  $[x_{(201)}, x_{(301)})$ ,  $[x_{(301)}, x_{(401)})$ ,  $[x_{(401)}, +\infty)$ . The bin endpoints

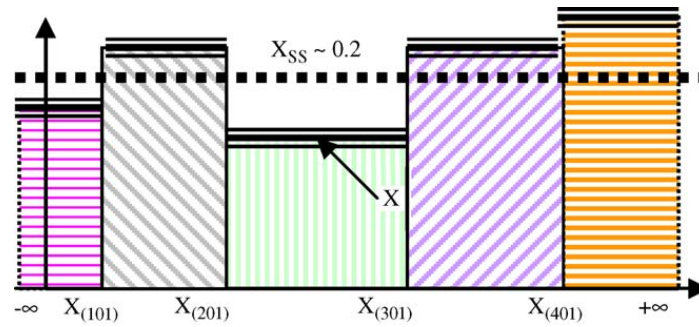


Fig. 2. EC histograms of  $X$  and  $X_{SS}$  using  $X_{SS}$  order statistics  $\{x(1), \dots, x(500)\}$ .

depend only on the original candidate subsample  $X_{SS}$  and are not redefined if  $X_{SS}$  is enhanced by adding more data from  $X$ . Each bin contains 20% of the original sample  $X_{SS}$ , represented by the dashed horizontal line, giving a uniform histogram height of 0.2 from left to right. The bin proportions for  $X$  are different, and are represented by the horizontal “three-lines” segments in Fig. 2.

When additional data are added to  $X_{SS}$ , then from (5), the subsample histogram heights  $q_{j1}, \dots, q_{jr_j}$  will change and generally become closer to the histogram heights  $p_{j1}, \dots, p_{jr_j}$  for  $X$ . The divergence test triggers acceptance of the subsample on the basis of the feature when a normalized difference between the histograms for enhancements of  $X_{SS}$  and  $X$  becomes sufficiently small. We compared equal-content bins based on  $X_{SS}$  with those based on  $X$ . You routinely expect something based on more data to be “better”, but we found that using  $X$  for bin determination did not produce more accuracy in the final clustering results than equal-content bins based on the initial  $X_{SS}$ . Since the computational cost of sorting  $X$  is typically much greater (assuming it can be done) than that for the initial  $X_{SS}$ , we chose to define bins using order statistics derived from the various features of the initial subsample  $X_{SS}$ .

#### 4. Numerical examples

All of our numerical examples use mixtures of normal distributions as the source of data sets. We do not assume that all data sets have an underlying form of this kind (in fact, we believe quite the opposite). Nonetheless, large data sets of this kind are easy to generate, and manipulation of their parameters provides a convenient way to study various tradeoffs that are made in algorithms such as geFFCM. Readers having a particular interest in normal mixtures may consult Titterington et al. (1985), Hathaway et al. (1987) for the general theory; Bezdek et al. (1997) for a recent study of how to detect the number of components in normal mixtures; Bezdek and Dunn (1975), Hathaway and Bezdek (1986) for relationships between FCM and the EM algorithm for estimating the parameters of normal mixtures; and Fraley and Raftery (2002) for an up to date account of EM clustering. The EM algorithm is extensible, and we think that progressive sampling and extension with EM will be equally effective for probabilistic clustering with EM as it is with FCM. The applicability of the proposed progressive sampling scheme to EM is illustrated via a single numerical example at the end of this section.

##### 4.1. Determination of control parameters

Figs. 3–6 are based on sampling from a fixed set  $X$  consisting of 100,000 (pseudo-) random observations drawn from a mixture of two bivariate normal distributions with component parameters given by:

$$p_1 = 0.5, \quad \mu_1 = (0, 0)^T, \quad \Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad (6a)$$

and

$$p_2 = 0.5, \quad \mu_2 = (10, 0)^T, \quad \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \quad (6b)$$

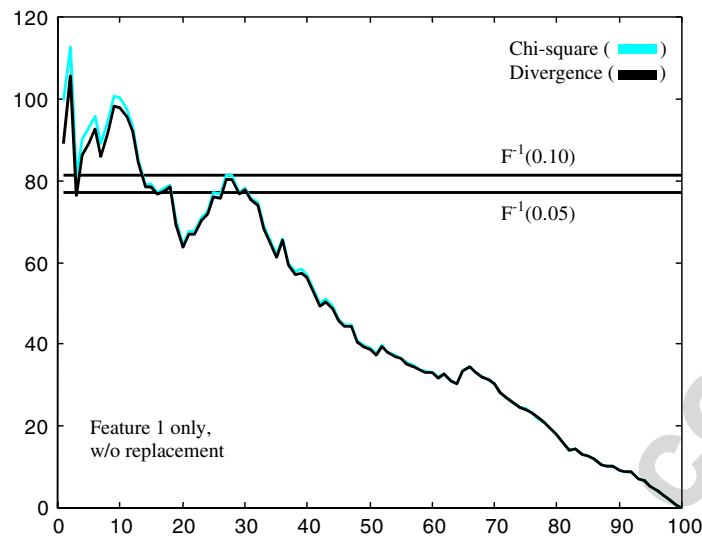


Fig. 3. Chi-square and divergence vs.  $|X_{SS}|/|X|\%$ .

The mixing proportions  $p_1 = p_2 = 0.5$  insure that approximately half of the data came from each component. All experiments used  $r = 100$  equal-content bins. The initial subsample  $X_{SS}$  was obtained by randomly selecting  $1000 = ((pN)/100)$  data from  $X$ , and it was used to define the bins. Enhancements were obtained by adding  $1000 = ((\Delta pN)/100)$  additional data to the current subsample. Depending on the experiment, sampling was either done with replacement from  $X$ , or without replacement from  $X - X_{SS}$ .

The horizontal axis in Figs. 3–6 gives the size of  $|X_{SS}|$  as a percentage of  $|X| = 100,000$ , so a value of 50 on the horizontal axis corresponds to  $|X_{SS}| = 50,000$ . Since larger subsamples are obtained by enhancing smaller ones, the subsamples used in a table are nested in the following way: the sample corresponding to a value of 50 on the horizontal axis is contained in the sample corresponding to a value of 51. A value of 100 on the horizontal axis, when sampling without replacement, corresponds to  $X_{SS} = X$ . The samples used to generate each figure are the same in that, for example, the 50% random sampling results on two different figures used the same random sample. In all the following figures, the subsample  $X_{SS}$  corresponding to a certain percentage is simply a randomly chosen subsample (of that size) and not one that necessarily signals acceptance using a particular acceptance strategy.

#### 4.1.1. The divergence test

Fig. 3 plots the chi-square goodness-of-fit (red line) and divergence (black line) test statistics for feature 1. Sampling was done without replacement and  $r = 100$  EC bins. The lines labeled  $F^{-1}(0.05)$  and  $F^{-1}(0.10)$  give the critical values from the chi-square distribution (with  $r - 1 = 99$  degrees of freedom) corresponding to the left most 5% and 10%. For example, a value of a test statistic for a given feature under the line labeled  $F^{-1}(0.05)$  would signal acceptance of the current subsample according to that feature at the level  $\alpha = 0.95$ .

Fig. 3 shows close agreement between the chi-squared and divergence tests. For the experiment producing Fig. 3, the divergence test statistic first signaled acceptance at 3% (of the samples in  $X$ ), and then again at 16%, while the chi-squared test did not first signal until 19%. The graphical similarity was typical of many figures we examined while studying these two test statistics. Based on similarity of the statistics over a range of examples we elected to use only the divergence test in our later experiments.

#### 4.1.2. Sampling without replacement

Let  $V_{X_{SS}}$  and  $V_X$  denote the terminal prototypes produced by LFCM on  $X_{SS}$  and  $X$ , respectively. These prototypes are the computed vectors in  $R^s$  that represent the (approximate) locations of the fuzzy cluster centers of the data. Fig. 4 compares the quality of the prototypes obtained by sampling with (red line) and without (black line) replacement by plotting the (Euclidean) norm  $\|V_{X_{SS}} - V_X\|$  against subsample size as a percentage of  $|X| = 100,000$ .

Over the range of sample sizes in Fig. 4, sampling without replacement usually produced a slightly more accurate result. The sampling without replacement curve necessarily includes the point (100,0). Fig. 4 shows good agreement

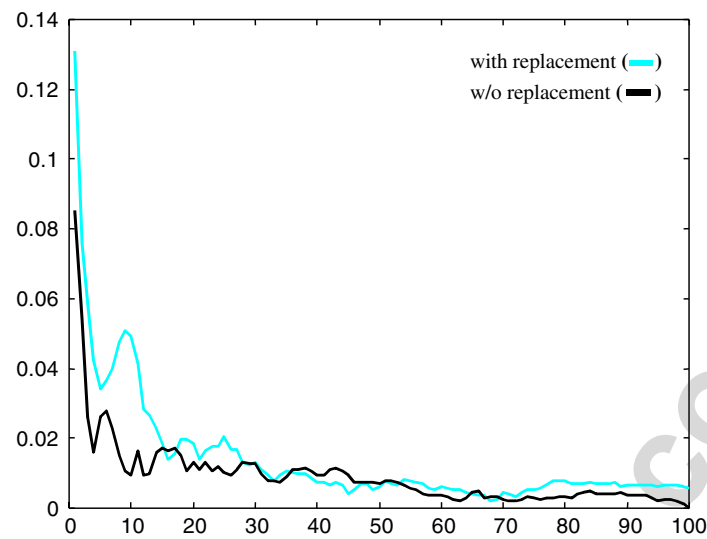


Fig. 4.  $\|V_{X_{SS}} - V_X\|_2$  vs.  $|X_S|/|X|\%$ .

between the literal and subsample prototypes for non-statistically tested samples drawn either with or without replacement for all sample sizes  $|X_{SS}| \geq \approx 0.3|X|$ .

#### 4.1.3. Efficient termination

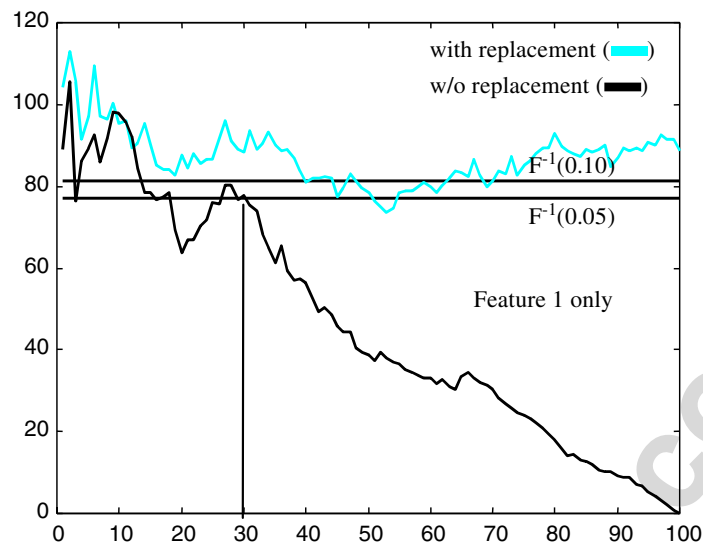
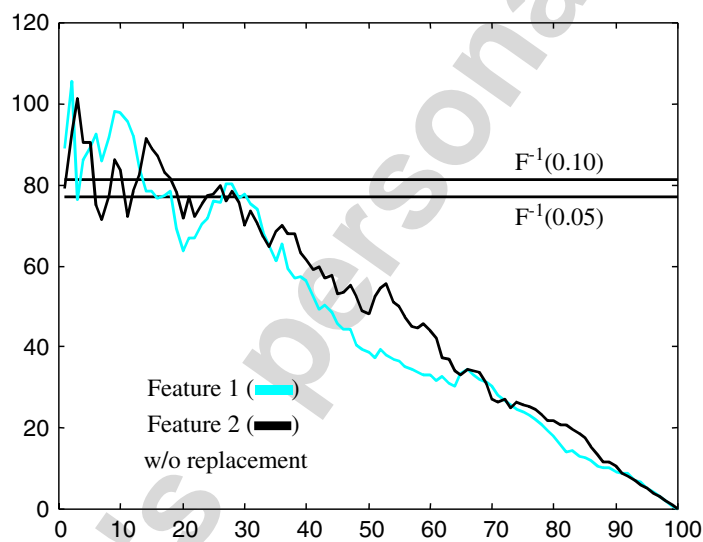
Provost et al. (1999) describe progressive sampling in terms of learning curves that plot subsample size against accuracy (recall that their work is on classifier design). Typically, a learning curve has three phases: (i) steep early slope, (ii) gradual middle ascent, followed by (iii) a plateau or horizontal asymptote that approaches the accuracy of learning with all the examples. They advocate the termination of progressive sampling at the end of gradual ascent in region (ii)—i.e., when adding more samples yields only marginal performance improvement. The curves in Fig. 4 can be viewed as learning curves (with descending error rather than ascending accuracy) exhibiting the three phases, with the error eventually becoming zero for sampling without replacement. Notice that the end of gradual ascent in the region (ii) part occurs around 30% for the example of Fig. 4

Of course curves of the type in Fig. 4 cannot be used to select a subsample since  $V_X$  is typically unavailable. But we can easily monitor the accuracy of our subsample histogram in approximating the histogram for all of  $X$ , as measured by the divergence test statistic. Does the divergence test statistic exhibit behavior that is consistent with, or indicative of, the error trend depicted in Fig. 4? Fig. 5 plots divergence test values for samples from feature 1. The answer is NO for sampling with replacement and YES for sampling without replacement.

Recall that the proposed termination of sampling occurs for values under the lines labeled  $F^{-1}$ . In Fig. 5, sampling with replacement is below the  $F^{-1}(0.05)$  critical value only briefly, between 50% and 60%, and even fails to accept the subsample  $|X_{SS}| = 100\%$  of  $|X| = 100,000$ . This indicates that when sampling with replacement, adding more samples does not lead to continuing improvement in the divergence test statistic, and therefore the divergence test statistic values will have little chance of correlating with a typical learning curve such as that in Fig. 4. (This behavior for sampling with replacement holds in general and is not a peculiarity of the example reported on here.)

On the other hand, the curve in Fig. 5 corresponding to sampling without replacement trends downward with increasing subsample size, as we would expect any reasonable measure of clustering error to do. (The learning curve for clustering shown as Fig. 1 in Meek et al. (2002) roughly agrees with the behavior seen in our Fig. 5.) Note also that the curve for sampling without replacement in Fig. 5 signals and holds acceptance starting at 31%, which is the approximate percentage in Fig. 4 at which prototype error starts to level off.

In summary, any reasonable measure of approximation error for the clustering problem should exhibit a trend of decreased error as subsample size increases, and any reasonable subsample acceptance criterion should be consistent with this trend. The probability that  $X_{SS}$ , sampled with replacement, is accepted depends on  $\alpha$  but not on the subsample

Fig. 5. Replacement:divergence vs.  $|X_{SS}|/|X|\%$ .Fig. 6. Features: divergence vs.  $|X_{SS}|/|X|$ .

size  $n$ . We avoid this inconsistency between sample acceptance and approximation error by choosing to sample without replacement.

#### 4.1.4. Feature selection strategy

Should we use all of the features, or just some subset of them, to guide subsample acceptance? Fig. 6 plots the divergence for features 1 (cyan) and 2 (black) vs. subsample size. Stopping as soon as one of the features is accepted gives a subsample size of 3% (when feature 1 first signals), and the more conservative approach of requiring both features to cumulatively accept gives 6% (when feature 2 first signals). An even more conservative strategy is to terminate sampling only when a subsample  $X_{SS}$  is found for which all (in this case both) features simultaneously indicate acceptance. For this bivariate mixture, simultaneous acceptance yields a 20% subsample. Referring back to Fig. 4, we see that 20% is where the quality of the prototypes just begins to stabilize.

We know that here feature 1 is more useful for discriminating between these two component normals than feature 2. However, Fig. 6 does not show significant differences for the two features that might be used to infer this. It may be that feature selection in the pattern recognition sense is never possible as a by-product of subsample selection.

#### 4.2. Experimentation with control parameters

In this subsection we embark on a series of numerical experiments that study the tradeoff between subsample size and various parameters of geFFCM. Our experiments were again done in MATLAB on a PC with a 2.0 GHz P-4 chip and 512 MB of RAM. Sections 4.2.1–4.2.3 use draws from a mixture of 4 five-dimensional normal distributions with these mixing proportions  $\{p_i\}$ , means  $\{\mu_i\}$  and covariance matrices  $\{\Sigma_i\}$ :

$$p_1 = p_2 = p_3 = p_4 = 0.25, \quad (7a)$$

$$\mu_1 = \begin{pmatrix} 0 \\ 1 \\ 3 \\ 0 \\ 0 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mu_3 = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mu_4 = \begin{pmatrix} 3 \\ 3 \\ 3 \\ 0 \\ 0 \end{pmatrix}, \quad (7b)$$

$$\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \sigma^2 I_5, \quad (7c)$$

where  $I_5$  denotes  $5 \times 5$  the identity matrix. The common variance  $\sigma^2$  in (7c) had one of three values: 0.1 (good separation), 0.5 (moderate separation), or 1.0 (poor separation).

The following parameter values for LFCM and geFFCM were used in all experiments. Number of clusters  $c = 4$ ; fuzzification constant  $m = 2$ ;  $\|\cdot\|$ =Euclidean norm on  $R^5$ ; stopping tolerance  $\varepsilon = 0.00001$ ; maximum number of iterations  $T = 1000$  (never attained). Additional values for geFFCM: bin numbers  $r_1 = r_2 = r_3 = r_4 = r_5 = r$  and significance levels  $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha$  are given in the tables; and  $((pN)/100) = ((\Delta pN)/100) = 1000$ . FCM was initialized at the correct hard partition acquired during data generation, and terminated when  $\max_{i,k} |u_{i,k}^{t+1} - u_{i,k}^t| < \varepsilon$ . We study subsample selection based on four strategies:

(SS1) *(Single) accept*:  $A = 1$  accepts  $X_{SS1}$  when a specified choice of any feature signals acceptance.

(SS2) *First accept*:  $FA = 1$  accepts  $X_{SS2}$  when any one of the ( $s = 5$ ) features signals acceptance.

(SS3) *Cumulative accept*:  $CA = 5$  accepts  $X_{SS3}$  when each of the ( $s = 5$ ) features has signaled acceptance for either the current or one of the previous candidate subsamples.

(SS4) *Simultaneous accept*:  $SA = 5$  accepts  $X_{SS4}$  when all ( $s = 5$ ) features simultaneously signal acceptance.

We hope the letter combinations ( $A, FA, CA, SA$ ) are mnemonically helpful; actual strategies are specified by choices of  $A, CA, SA$  and  $I_0$  in step 1 of geFFCM. Selection rule (SS1) is analogous to the acceptance strategy for image data based on univariate gray levels that was used by Pal and Bezdek (2002). Subsamples produced by (SS2)–(SS4) always satisfy:

$$|X_{SS2}| \leq |X_{SS3}| \leq |X_{SS4}|. \quad (8)$$

We may state (8) as  $FA \leq CA \leq SA$ . By definition, event “FA” is a precursor to events “CA” and “SA”. Were it not for the vagaries of sampling, we could not assert the second of the inequalities above. However, it can (and often did!) happen that a feature, which signaled at, say,  $\hat{p}\%$ , failed to signal at a larger value, say  $\hat{p} + (\tau \cdot \Delta p)\%$ . Hence, CA and SA are not always simultaneous events.

##### 4.2.1. Sample size vs. five parameters

Tables 2–4 give subsample sizes and computation times for samples from mixtures with  $\sigma^2 = 0.10, 0.50$  and 1.00. Each table is based on 25 trials; each trial uses a different sample of size 100,000.

4.2.1.1. *Subsample size vs. separation.* The average over the 64 percentages in each of the three tables is 18.4%, 18.6% and 18.4%. Since each entry in the tables is itself an average of 25 trials, these three values represent 1600 trials each. This shows that, over a pretty extensive set of trials, the degree of separation between the four components has very little effect on the size of the selected subsample. However, we will see in Section 4.3 that while the degree

Table 2

Subsample sizes (as % of  $|X| = 100,000$ ) and times (averages of 25 trials,  $\sigma^2 = 0.10 = \text{Hi Sep}$ )

	$r = 25$		$r = 50$		$r = 100$		$r = 200$	
	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$
$A = 1, x_1$ , accepted size, $ X_{SS1} $	19	21	15	19	12	14	7	10
$A = 1, x_2$ , accepted size, $ X_{SS1} $	17	27	14	19	9	14	8	12
$A = 1, x_3$ , accepted size, $ X_{SS1} $	17	26	16	21	13	15	9	13
$A = 1, x_4$ , accepted size, $ X_{SS1} $	13	22	12	17	13	19	10	12
$A = 1, x_5$ , accepted size, $ X_{SS1} $	19	27	18	23	13	18	11	13
$FA = 1$ , accepted size, $ X_{SS2} $ CPU time, s	3 0.14	6 0.15	4 0.16	6 0.17	3 0.18	5 0.19	3 0.20	4 0.21
$CA = 5$ , accepted size, $ X_{SS3} $ CPU time, s	35 0.17	47 0.19	30 0.20	35 0.21	24 0.22	29 0.25	17 0.26	22 0.29
$SA = 5$ , accepted size, $ X_{SS4} $ CPU time, s	44 0.21	54 0.23	35 0.22	40 0.24	27 0.25	33 0.26	21 0.29	26 0.31

 $r =$  Number of bins;  $\alpha =$  significance level.

Table 3

Subsample sizes (as % of  $|X| = 100,000$ ) and times (averages of 25 trials,  $\sigma^2 = 0.50 = \text{Med Sep}$ )

	$r = 25$		$r = 50$		$r = 100$		$r = 200$	
	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$
$A = 1, x_1$ , accepted size, $ X_{SS1} $	21	27	20	25	13	20	10	13
$A = 1, x_2$ , accepted size, $ X_{SS1} $	15	24	14	17	11	14	9	11
$A = 1, x_3$ , accepted size, $ X_{SS1} $	18	25	11	18	13	16	10	12
$A = 1, x_4$ , accepted size, $ X_{SS1} $	17	25	15	21	11	16	7	9
$A = 1, x_5$ , accepted size, $ X_{SS1} $	15	27	13	20	11	15	8	11
$FA = 1$ , accepted size, $ X_{SS2} $ CPU time, s	3 0.14	5 0.15	3 0.17	5 0.18	3 0.18	5 0.19	3 0.20	3 0.21
$CA = 5$ , accepted size, $ X_{SS3} $ CPU time, s	37 0.17	48 0.19	31 0.22	37 0.24	24 0.22	29 0.25	18 0.25	21 0.28
$SA = 5$ , accepted size, $ X_{SS4} $ CPU time, s	47 0.21	54 0.22	38 0.25	42 0.26	29 0.25	33 0.26	21 0.28	24 0.28

 $r =$  Number of bins;  $\alpha =$  significance level.

of separation does not seem to affect the size of accepted subsamples, it does affect the quality of geFFCM approximations to LFCM partitions of  $X$ .

**4.2.1.2. Subsample size vs.  $\alpha$ .** Tables 2–4 contain 96 pairs of percentages for ( $\alpha = 0.90 \leftrightarrow \alpha = 0.95$ ). The average increase on passing from 0.90 to 0.95 in the 96 pairs is 4.95%. 95 of the 96 pairs increased, while the pair at  $FA = 1$ , medium separation, 200 bins did not change. Thus, increasing  $\alpha$  will most likely increase the size of the accepted subsample. We presume that larger subsamples obtained by using  $\alpha = 0.95$  will yield better approximations to LFCM.

**4.2.1.3. Subsample size vs. number of bins.** Tables 2–4 compare the numbers of bins for ( $\alpha = 0.90 \leftrightarrow \alpha = 0.95$ ) to subsample size. In general, increasing  $\alpha$  increases the size of the subsample, but the relative effect decreases as the number of bins increases. If we average all subsample sizes under each bin size, the percentages for each bin size are 25.5% ( $r = 25$ ), 19.8% ( $r = 50$ ), 16.4% ( $r = 100$ ) and 12.3% ( $r = 200$ ). Thus, using 200 EC bins instead of 25 reduces the average subsample size by more than half!

Table 4

Subsample sizes (as % of  $|X| = 100,000$ ) and times (averages of 25 trials,  $\sigma^2 = 1.00 = \text{Lo Sep}$ )

	$r = 25$		$r = 50$		$r = 100$		$r = 200$	
	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.90$	$\alpha = 0.95$
$A = 1, x_1$ , accepted size, $ X_{SS1} $	22	29	14	23	14	18	11	15
$A = 1, x_2$ , accepted size, $ X_{SS1} $	14	23	11	17	11	17	8	12
$A = 1, x_3$ , accepted size, $ X_{SS1} $	16	19	12	19	11	15	7	11
$A = 1, x_4$ , accepted size, $ X_{SS1} $	23	29	13	19	12	14	8	12
$A = 1, x_5$ , accepted size, $ X_{SS1} $	17	29	9	15	10	17	10	13
FA = 1, accepted size, $ X_{SS2} $ CPU time, s	3 0.13	5 0.14	2 0.15	4 0.16	3 0.17	4 0.18	2 0.19	4 0.21
CA = 5, accepted size, $ X_{SS3} $ CPU time, s	39 0.17	48 0.19	28 0.18	35 0.21	24 0.22	30 0.24	19 0.25	23 0.28
SA = 5, accepted size, $ X_{SS4} $ CPU time, s	48 0.20	54 0.21	35 0.22	40 0.23	30 0.25	34 0.26	22 0.28	25 0.29

$r$  = Number of bins;  $\alpha$  = significance level.

4.2.1.4. *Subsample size vs. acceptance strategy.* Average percent acceptances across all columns for the four strategies in Tables 2–4 are very consistent:

Strategy	Table 2	Table 3	Table 4
A	15.7	15.7	15.5
FA	4.3	3.8	3.4
CA	29.9	30.6	30.8
SA	35.0	36.0	36.0

This again demonstrates how indifferent subsample size is to the amount of separation in the source population. Percentages of about 30% for CA and SA are very close to those reported for the eFFCM image sampling method in Pal and Bezdek (2002). As pointed out earlier, this argues in favor of an arithmetic sampling schedule, per arguments given by Provost et al. (1999).

4.2.1.5. *Subsample size vs. CPU time.* CPU times for the FA, CA and SA acceptance strategies in Tables 2–4 range from 0.13 to 0.31 s. The maximum subsample size in Tables 2–4 is 54%, which occurs in all three tables for 25 bins at SA = 5,  $\alpha = 0.95$ . The general trend is that CPU times increase (almost linearly) with increasing bin size—and not with the size of the accepted subsample. This is due to the higher computational cost of obtaining the bin counts and computing  $D$  in (5) for the larger numbers of equal content bins.

Section 4.2.1 used data sets of size  $|X| = 100,000$ . Once  $X$  is large enough to give a sufficiently accurate calculation of bin probabilities, additional increases in  $|X|$  should result in only minor marginal cost to geFFCM. Increasing  $|X|$  should result in a much greater marginal cost to LFCM. If true, a greater and greater computational advantage will accrue to geFFCM as  $|X|$  becomes larger and larger.

#### 4.2.2. Sample size as a function of $|X|$

We extended Section 4.2.1 using only moderate separation ( $\sigma^2 = 0.50$ ) by increasing the sample size of  $X$  by a factor of 16, so that the larger  $|X| = 1,600,000$ . Our notation in Table 5 is  $|X_N| = 100,000$  and  $|X_N| = 1,600,000$ , which is very near the largest size that could readily be stored and processed by MATLAB on the PC used. For convenience in comparison, we repeat the relevant values from Table 3, which are listed to the left of the corresponding results for the larger data set.

4.2.2.1. *Subsample size vs. sample size.* Comparing pairs of columns at equal numbers of bins in Table 5 shows that the percentage of  $X$  required to signal termination of sampling decreases as  $|X|$  increases in all cases. The least decrease

Table 5

Subsample sizes (as %) and times: averages of 25 trials,  $|X_N| = 100,000$ ,  $|X_N| = 1,600,000$ ,  $\alpha = 0.95$ ,  $\sigma^2 = 0.50 = \text{Med Sep}$

	$r = 25$		$r = 50$		$r = 100$		$r = 200$	
	$X = X_n$	$X = X_N$	$X = X_n$	$X = X_N$	$X = X_n$	$X = X_N$	$X = X_n$	$X = X_N$
$A = 1, x_1$ , accepted size, $ X_{SS1} $	27	15	25	12	20	8	13	9
$A = 1, x_2$ , accepted size, $ X_{SS1} $	24	12	17	10	14	9	11	5
$A = 1, x_3$ , accepted size, $ X_{SS1} $	25	17	18	11	16	9	12	6
$A = 1, x_4$ , accepted size, $ X_{SS1} $	25	15	21	13	16	8	9	8
$A = 1, x_5$ , accepted size, $ X_{SS1} $	27	13	20	14	15	11	11	9
$FA = 1$ , accepted size, $ X_{SS2} $ CPU time, s	5 0.15	1 1.9	5 0.18	1 2.2	5 0.19	1 2.4	3 0.21	1 2.7
$CA = 5$ , accepted size, $ X_{SS3} $ CPU time, s	48 0.19	35 2.4	37 0.27	31 2.8	29 0.25	21 3.1	21 0.28	17 3.5
$SA = 5$ , accepted size, $ X_{SS4} $ CPU time, s	54 0.22	52 3.3	42 0.26	40 3.5	33 0.26	29 3.6	24 0.28	23 4.1

$r = \text{Number of bins}; X = \text{data set}.$

occurs for  $\alpha = 0.95$ ,  $r = 25$ , and the SA strategy (SS4) highlighted in Table 5, the subsample size for  $X_n$  is 54% of 100,000, while the subsample size for  $X_N$  only drops to 52% of 1,600,000. Of course, this is a huge increase in the actual size of  $|X_{SS}|$ , which is roughly 54,000 points from  $X_n$ , compared to 832,000 for  $X_N$ . We were quite surprised that 52% of the larger sample was required to get an equivalent subsample approximation to the histogram of  $X$ . It may be that the large size of the simulation creates some problems in accurately tabulating the statistics, calculating the critical values or generating sufficiently random data. On the other hand, a large  $|X|$  means that sampling without replacement will be roughly equivalent to sampling with replacement for smaller sizes of  $|X_{SS}|$ ; and recall that the probability any one feature will signal acceptance is only 5% for sampling with replacement.

4.2.2.2. *Subsample size vs. acceptance strategy.* The trends discussed in Section 4.2.1.4 for the four acceptance strategies hold up well in Table 5. For the larger data set, the averages for the four strategies are:  $FA = 1\%$ ;  $A = 10.7\%$ ;  $CA = 26\%$  and  $SA = 36\%$ .

4.2.2.3. *Subsample size vs. CPU time.* The average of the 24 CPU times shown in Table 5 is 0.23 s for  $X_n$ , compared to 2.96 s for  $X_N$ . Certainly an increase is expected. The size of the sample increased 16 times, while the average CPU time needed to secure an accepted subsample increased by a factor of about 13. The CPU times given in Table 5 are for subsample selection only. Total computation time is studied in Section 4.2.3.

Now we turn to the question of whether or not  $X_{SS}$  represents  $X$  well enough so that clustering  $X_{SS}$ , followed by non-iterative extension to  $X - X_{SS}$ , produces good approximations to LFCM clusters of  $X$ . We use four measures to assess the quality of geFFCM estimates. When  $X = X_L$ , we cluster  $X_L$  with LFCM, obtaining  $(U_{lit}, V_{lit})$ . The cluster centers can be compared directly by calculating the Euclidean distance between the two sets of prototype vectors:

$$E_V = \|V_{X_{SS}} - V_{lit}\| / \|V_{lit}\|. \tag{9a}$$

To assess the LFCM and geFFCM partitions of  $X_L$ , we first harden (“defuzzify”)  $U_{lit}$  and  $U_{ext}$ . In the usual way, let  $H$  be the function that replaces each fuzzy column of any soft partition by the crisp column that has a 1 at the entry of maximum membership, and 0’s elsewhere. Then we compare the hardened version of  $U_{ext}$  to the hardened version of  $U_{lit}$  and also to the correct labels  $U_{true}$  for the points in  $X_L$  by computing the normalized Euclidean distances:

$$E_{app} = 50 \bullet \|H(U_{ext}) - H(U_{lit})\| / |X_L|. \tag{9b}$$

$$E_{tr} = 50 \bullet \|H(U_{ext}) - (U_{true})\| / |X_L|. \tag{9c}$$

$E_{app}$  and  $E_{tr}$  are normalized so that they express percentages (relative to  $X_L$ ). Eq. (9b) counts the number of mismatches between columns in  $H(U_{lit})$  and  $H(U_{ext})$ , so  $E_{app}$  measures the approximation error as shown in Fig. 1.  $E_{app}$  is not

Table 6

Average (of averages) of acceleration and approximation errors,  $|X| = 100,000$ .

		→→→ Decreasing separation of clusters →→→					
		$\sigma^2 = 0.10$		$\sigma^2 = 0.50$		$\sigma^2 = 1.00$	
		geFFCM	LFCM	geFFCM	LFCM	geFFCM	LFCM
SS2 FA = 1	Acceleration $T_{acc}$	5.21	1	12.44	1	28.41	1
	% Training error $E_{tr}$	0.03	0.03	8.90	8.85	22.40	22.18
	% Approx. error $E_{app}$	0.00	0	0.95	0	3.15	0
	Prototype error $E_V$	0.00	0	0.01	0	0.03	0
SS3 CA = 5	Acceleration $T_{acc}$	2.61	1	3.59	1	4.34	1
	% Training error $E_{tr}$	0.03	0.03	8.86	8.85	22.19	22.18
	% Approx. error $E_{app}$	0.00	0	0.23	0	0.78	0
	Prototype error $E_V$	0.00	0	0.00	0	0.00	0
SS4 SA = 5	Acceleration $T_{acc}$	2.30	1	2.97	1	3.41	1
	% Training error $E_{tr}$	0.03	0.03	8.85	8.85	22.19	22.18
	% Approx. error $E_{app}$	0.00	0	0.20	0	0.69	0
	Prototype error $E_V$	0.00	0	0.00	0	0.00	0

an “error rate” in the usual sense, but  $E_{tr}$  is the classification (training) error rate of geFFCM when its fuzzy partition is hardened and taken as an estimate of the true labels of points in the sample.

The last measure of utility we compute provides an estimate of the time saved by using geFFCM instead of LFCM. We define the dimensionless acceleration factor  $T_{acc}$  as

$$T_{acc} = \left( \frac{\text{LFCM time}}{\text{geFFCM time}} \right). \quad (9d)$$

geFFCM time is the total time to find and cluster  $X_{SS}$ , and then extend the clustering to  $X - X_{SS}$ .

LFCM cannot process  $X_{VL}$ , so  $E_{app}$  and  $E_{tr}$  cannot be obtained when  $X = X_{VL}$ . However, small values of  $E_{app}$ ,  $E_V$  and  $E_{tr}$  for the  $X_L$  case will buoy our confidence that geFFCM builds accurate estimates of non-computable partitions and cluster centers in the  $X_{VL}$  case. Now we are ready to study the quality of geFFCM approximations to LFCM.

#### 4.2.3. Quality of approximations

We did eight sets of experiments three times, corresponding to choices for  $(\alpha, r, \sigma^2)$  as in Tables 2–4:  $\alpha = 0.90, 0.95$  for each bin size  $r = 25, 50, 75, 100$  and each  $\sigma^2 = 0.10, 0.50$  and 1.00. Simulations reported for each of the 24 triples  $(\alpha, r, \sigma^2)$  are again averages of 25 samples of size 100,000 from the mixture distribution at (7). Variations in the averages over  $(\alpha, r)$  for fixed  $\sigma^2$  were surprisingly stable, so Table 6 reports averages of the acceleration factor and the quality of approximation to LFCM by geFFCM taken over the 8 sets of averages corresponding to variables  $(\alpha, r)$  for fixed  $\sigma^2$ .

**4.2.3.1. Separation vs. acceleration.** Scanning the  $T_{acc}$  rows in Table 6 shows that decreasing the separation (increasing  $\sigma^2$ ) has the very predictable effect of increasing the amount of computing time needed for LFCM and geFFCM. This effect is implicit in the values shown for the acceleration factor.  $T_{acc}$  increases with  $\sigma^2$ , most dramatically for selection strategy SS2, which shows an increase of 545% from  $\sigma^2 = 0.10$  to 1.00. The smallest increase for these two cases is for SS4, where  $T_{acc}$  increases by 48%.

**4.2.3.2. Selection strategy vs. acceleration.** Examining the columns in Table 6 against values of  $T_{acc}$  shows that the acceleration factor decreases as we pass from SS2 = FA → SS3 = CA → SS4 = SA (remember that FA ≤ CA ≤ SA). Again we see the most dramatic change at  $\sigma^2 = 1.00$ , where  $T_{acc}$  decreases by 833% (28.41/3.41).

**4.2.3.3. Separation and selection strategies vs. training error.** As expected,  $E_{tr}$  increases with decreasing separation for both LFCM and geFFCM, and is essentially unchanged with increasing stringency of the subsample selection strategy. More importantly, training errors of the literal and approximation schemes are almost identical in all cases

Table 7

Average (of averages) of acceleration and approximation errors,  $|X| = 100,000$ ,  $\sigma^2 = 0.10$ 

		FCM		EM	
		geFFCM	LFCM	geFEM	LEM
SS2 FA = 1	Acceleration $T_{acc}$	5.09	1	8.71	1
	% Training error $E_{tr}$	0.03	0.03	0.03	0.03
	% Approx. error $E_{app}$	0.00	0	0.01	0
	Prototype error $E_V$	0.01	0	0.01	0
SS3 CA = 5	Acceleration $T_{acc}$	2.56	1	2.98	1
	% Training error $E_{tr}$	0.03	0.03	0.03	0.03
	% Approx. error $E_{app}$	0.00	0	0.00	0
	Prototype error $E_V$	0.00	0	0.00	0
SS4 SA = 5	Acceleration $T_{acc}$	2.29	1	2.61	1
	% Training error $E_{tr}$	0.03	0.03	0.03	0.03
	% Approx. error $E_{app}$	0.00	0	0.00	0
	Prototype error $E_V$	0.00	0	0.00	0

in Table 6. The biggest difference occurs at  $\sigma^2 = 1.00$  and SS2, where geFFCM is 0.22% higher than LFCM. Thus, subsampling plus extension does produce reliable estimates of LFCM clusters, and hence, increases our confidence in using geFFCM for the (non-computable) VL data case.

**4.2.3.4. Separation and selection strategies vs. approximation error.** The approximation error for LFCM is necessarily zero. For geFFCM,  $E_{app}$  increases with decreasing separation, and decreases with increasing stringency of the subsample selection strategy. Both of these trends agree with intuition.

**4.2.3.5. Separation and selection strategies vs. prototype error.** Prototype error for LFCM must be zero. There is not enough difference between the prototypes produced by LFCM and geFFCM to see (at two decimal places) in seven of the nine cases in Table 6. The differences seen for  $\sigma^2 = 0.50$  and 1.00 at SS2 are completely unremarkable.

### 4.3. Extensible EM example

This final example demonstrates that the extension approach can be applied to other extensible algorithms, and we have chosen to do an extension of the popular EM algorithm for probabilistic clustering based on normal mixture decomposition (McLachlan and Peel, 2000). The purpose of this example is not to rigorously compare EM and FCM as tools for clustering and parameter estimation (Bezdek et al., 1985) or to determine which of them works best as an extensible partner to our progressive sampling scheme. More modestly, we simply want to show that divergence-based sampling can be used with other extensible algorithms, in this case a probabilistic one, to gain advantage.

This example is most succinctly described by saying that it replicates the simulation used to obtain the first two columns of Table 6 (those corresponding to  $\sigma^2 = 0.10$ ), except that: (1) results are given for both FCM and EM, and (2) 10 rather than 25 trials were done for each choice of acceptance strategy, number of bins  $r$ , and significance level  $\alpha$ . In Table 7 we denote literal EM by LEM and the subsample version of EM by geFEM. The geFEM algorithm is obtained from geFFCM by making two simple modifications. First, the mean updates for FCM in (3) are replaced by the EM parameter updates given ( $\forall i$ ) by:

$$p_i^{\text{new}} = \frac{1}{n} \sum_{j=1}^n u_{ij}^{\text{old}}, \quad (10a)$$

$$v_i^{\text{new}} = \frac{\sum_{j=1}^n u_{ij}^{\text{old}} x_j}{\sum_{j=1}^n u_{ij}^{\text{old}}} \quad (10b)$$

and

$$\Sigma_i^{\text{new}} = \sum_{j=1}^n u_{ij}^{\text{old}} (x_j - v_i^{\text{new}}) (x_j - v_i^{\text{new}})^T \bigg/ \sum_{j=1}^n u_{ij}^{\text{old}}. \quad (10c)$$

Second, the equations for the FCM update and extension of  $U$  in (4) are replaced by the EM equations given ( $\forall i, k$ ) by

$$u_{ik}^{\text{new}} = p_i^{\text{new}} f_{ik}^{\text{new}} \bigg/ \sum_{j=1}^c p_j^{\text{new}} f_{jk}^{\text{new}}, \quad (11a)$$

where

$$f_{ik}^{\text{new}} = \frac{\exp\left(-0.5(x_k - v_i^{\text{new}})^T (\Sigma_i^{\text{new}})^{-1} (x_k - v_i^{\text{new}})\right)}{\sqrt{(2\pi)^s |\Sigma_i^{\text{new}}|}}. \quad (11b)$$

As pointed out on p. 82 of [McLachlan and Peel \(2000\)](#), the covariance matrix updates can be calculated in a more efficient way than (10c), but as stated earlier, we are not attempting to compare the most efficient versions of EM and FCM to each other. The purpose is to show that a typical extensible algorithm other than FCM can be beneficially combined with the proposed sampling scheme. The results are given in [Table 7](#).

The accuracy results for geFFCM and geFEM were quite similar (and good!), comparing each to their respective literal counterparts. The geFEM algorithm produced good acceleration factors of 8.71, 2.98 and 2.61, and these factors are slightly higher than those obtained in the FCM portion of the simulation. This experiment illustrates an important property of the proposed sampling scheme; it is very general and provides an easily adaptable framework for any extensible algorithm.

## 5. Conclusions and future research

We have extended the eFFCM approach of [Pal and Bezdek \(2002\)](#) to geFFCM for general, non-image data. geFFCM is well suited for continuous or discrete data having a large number of distinct feature values. And, with a suitable restriction on the number of bins, it can also be applied to coarse data exhibiting only a small number of distinct feature values. Termination of progressive sampling is controlled by choices made by the user. Four acceptance strategies were examined: accept when a specified feature signals (accept  $A = 1$ , SS1); accept when one of the features signals (first accept  $FA = 1$ , SS2); accept when all  $s$  features sequentially signal (cumulative accept  $CA = s$ , SS3); and accept when all  $s$  features simultaneously signal (simultaneous accept  $SA = s$ , SS4). We believe that having a small number of required acceptances (such as  $CA = 5$ ) is sufficient even if the number of features  $s \gg 5$ .

One of the most interesting results of the numerical tests was that even the apparently minimal FA strategy provided sample misclassification error rates comparable to LFCM. Another slight surprise was that large subsamples were sometimes required to satisfy the SA strategy. While better accuracy is obtained with CA and SA, they should probably be used with a reduced value of the significance level  $\alpha$ .

For some applications, using subsamples of size 20–50% of  $|X|$  generated by the cumulative accept (SS3) or simultaneous accept (SS4) strategies is overkill. There are several obvious ways that the criterion for subsample acceptance can be relaxed, First, the first accept strategy (SS2) can be used, or one that requires acceptance of some specified percentage of all the features. Second, we can use either geometric or adaptive sampling. For example, the significance level  $\alpha$  can be lowered and/or the number of bins can be increased to make the schedule adaptive. Finally, a geometric sampling schedule becomes attractive when  $|X_{SS}|$  falls below about 16% at termination ([Provost et al., 1999](#)).

We observed no trends in the numerical results that support the conjecture that geFFCM is useful for feature selection (in the pattern recognition sense), or to determine the degree of overlap between clusters. Indications about either of these characteristics of the input data would provide helpful feedback to the selection of a useful subsample, so this search will remain open in future extensions of the work presented here.

While none of the data sets used in this paper are of the “VL” (unloadable) size, our examples show that geFFCM achieves the two objectives set for it: (i) it definitely accelerates LFCM in all cases where  $X_L$  is loadable, and (ii),

acceleration is accomplished with very little apparent sacrifice in the quality of extended approximations to LFCM partitions and prototypes. These numerical results encourage us to assert that geFFCM would, if called upon to do so, yield very acceptable and reasonably accurate results in the VL data case. An important follow-up to this research is to apply geFFCM to a variety of other well-studied data sets to show that our method is generally applicable to other types of data (and not just to normal mixtures).

Finally, we want to emphasize that the methods given in this article for subsampling and extension should be equally effective for acceleration, approximation and extension to VL data of any extensible clustering algorithm (not just FCM). The key requirement is algorithmic extensibility, discussed in greater detail in Pal and Bezdek (2002). The final example of Section 4 demonstrates that the EM algorithm, well known in mixture decomposition, is one of the extensible algorithms that can be coupled with the progressive sampling schemes presented here.

Finally, we believe that there are computationally cheaper and/or better methods for terminating progressive sampling. Is the divergence test (or some elastic modification of it) the best predictor of the cluster-specific usefulness of  $X_{SS}$ ? The answer is probably no. Answering this question will be a next step in extending clustering to VL data.

## References

- Baeza-Yates, R., Ribeiro-Neto, B., 1999. *Modern Information Retrieval*. Addison Wesley Longman, Reading, MA.
- Bezdek, J.C., Dunn, J.C., 1975. Optimal fuzzy partitions: a heuristic for estimating the parameters in a mixture of normal distributions. *IEEE Trans. Comput.* 24 (8), 835–838.
- Bezdek, J.C., Hathaway, R.J., 2002. VAT: a tool for visual assessment of (cluster) tendency. In: *Proceedings of the IJCNN 2002*, IEEE Press, Piscataway, NJ, pp. 2225–2230.
- Bezdek, J.C., Hathaway, R.J., 2003. Convergence of alternating optimization. *Neural, Parallel Scient. Comput.* 11, 351–368.
- Bezdek, J.C., Hathaway, R.J., Huggins, V.J., 1985. Parametric estimation for normal mixtures. *Pattern Recognition Lett.* 3, 79–84.
- Bezdek, J.C., Li, W.Q., Attikiouzel, Y.A., Windham, M.P., 1997. A geometric approach to cluster validity for normal mixtures. *Soft Comput.* 1, 166–179.
- Bezdek, J.C., Keller, J.M., Krishnapuram, R., Pal, N.R., 1999. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer, Norwell.
- Bradley, P., Fayyad, U., Reina, C., 1998a. Scaling clustering algorithms to large databases, In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, pp. 9–15.
- Bradley, P., Fayyad, U., Reina, C., 1998b. Scaling EM (expectation-maximization) clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA.
- Cannon, R.L., Dave, J.V., Bezdek, J.C., 1986. Efficient implementation of the fuzzy  $c$ -means algorithm. *IEEE Trans. PAMI* 8, 248–255.
- Cheng, T.W., Goldgof, D.B., Hall, L.O., 1995. Fast clustering with application to fuzzy rule generation. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, Tokyo, Japan, pp. 2289–2295.
- Cutting, D.R., Karger, D.R., Pederson, J.O., Tukey, J.W., 1992. Scatter/gather: a cluster-based approach to browsing large document collections, In: *Proceedings of the ACM SIGIR'92*, pp. 318–329.
- Domingos, P., Hulten, G., 2001. A general method for scaling up machine learning algorithms and its application to clustering. In: *Proceedings of the 18th International Conference on Machine Learning*, pp. 106–113.
- Eschrich, S., Ke, J., Hall, L.O., Goldgof, D.B., 2003. Fast accurate fuzzy clustering through data reduction. *IEEE Trans. Fuzzy Systems* 11, 262–269.
- Farnstrom, F., Lewis, J., Elkan, C., 2000. Scalability for clustering algorithms revisited. *SIGKDD Explorations*, vol. 2(1). ACM press, New York, pp. 1–7.
- Fayyad, U., Smyth, P., 1996. From massive data sets to science catalogs: applications and challenges. In: Kettenring, J., Pregibon, D. (Eds.), *Proceedings of the Workshop on Massive Data Sets*, National Research Council.
- Fraleigh, C., Raftery, A.E., 2002. Model-based clustering, discriminant analysis, and density estimation. *J. Amer. Statist. Assoc.* 97 (458), 611–631.
- Ganti, V., Gehrke, J., Ramakrishnan, R., 1999a. Mining very large databases, *Computer*, August, pp. 38–45.
- Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A.L., French, J.C., 1999b. Clustering large datasets in arbitrary metric spaces, *Proceedings of the 15th International Conference on Data Engineering*, IEEE CS Press, Los Alamitos, CA, pp. 502–511.
- Guha, S., Rastogi, R., Shim, K., 1998. CURE: an efficient clustering algorithm for large databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 73–84.
- Hathaway, R.J., Bezdek, J.C., 1986. On the asymptotic properties of fuzzy  $c$ -means cluster prototypes as estimators of mixture subpopulation centers. *Comm. Statist. (A)* 15 (2), 505–513.
- Hathaway, R.J., Redner, R., Bezdek, J.C., 1987. Estimating the parameters of mixture models with modal estimators. *Comm. Statist. (A)* 16 (9), 2639–2660.
- Huber, P., 1996. *Massive Data Sets Workshop: The Morning After*, Massive Data Sets. National Academy Press, pp. 169–184.
- Jain, A.K., Dubes, R.C., 1988. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ.
- Kolen, J.F., Hutcheson, T., 2002. Reducing the time complexity of the fuzzy  $c$ -means algorithm. *IEEE Trans. Fuzzy Systems* 10, 263–267.
- McLachlan, G., Peel, D., 2000. *Finite Mixture Models*. Wiley, New York.
- Meek, C., Thiesson, B., Heckerman, D., 2002. The learning curve sampling method applied to model based clustering. *J. Mach. Learning Res.* 2, 397–418.

- Ng, R.T., Han, J., 1994. Efficient and effective clustering methods for spatial data mining. In: Proceedings of the 20th International Conference on Very Large Databases, Morgan Kauffman, San Francisco, pp. 144–155.
- Pal, N.R., Bezdek, J.C., 2002. Complexity reduction for large image processing. *IEEE Trans. Systems, Man, Cybernet. Part B: Cybernet.* 32, 598–611.
- Provost, F., Jensen, D., Oates, T., 1999. Efficient progressive sampling, In: Proceedings of the Fifth KDDM, ACM Press, New York, pp. 23–32.
- Titterton, D., Smith, A., Makov, U., 1985. *Statistical Analysis of Finite Mixture Distributions*. Wiley, New York.
- Uma Shankar, B., Pal, N.R., 1994. FFCM: an effective approach for large data sets. In: Proceedings of the Third International Conference on Fuzzy Logic, Neural nets, and Soft Computing, IIZUKA, Fukuoka, Japan, pp. 332–332.
- Zhang, T., Ramakrishnan, R., Livny, M., 1996. BIRCH: an efficient data clustering method for very large databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, New York, pp. 103–114.

Author's personal copy