Accelerating EM Clustering to Find High Quality Solutions

Carlos Ordonez
Teradata, NCR
San Diego, CA 92127, USA

Edward Omiecinski
Georgia Institute of Technology
Atlanta, GA 30332, USA *

Abstract

Clustering is one of the most important techniques used in Data Mining. This article focuses on the EM clustering algorithm. Two fundamental aspects are studied: achieving faster convergence and finding higher quality clustering solutions. This work introduces several improvements to the EM clustering algorithm, being periodic M steps during initial iterations, reseeding of low weight clusters and splitting of high weight clusters the most important. These improvements lead to two important parameters. The first parameter is the number of M steps per iteration and the second one, a weight threshold to reseed low-weight clusters. Experiments show how frequently the M step must be executed and what weight threshold values make EM reach higher quality solutions. In general, the improved EM clustering algorithm finds higher quality solutions than the Classical EM algorithm and converges in less iterations.

1 Introduction

Data Mining is an active research area [11]. Clustering, association rules and decision trees are some of the most popular data mining techniques. This work focuses on clustering. There is no universal definition of clustering, but most authors agree on the following. Clustering algorithms partition a set of points into disjoint groups such that points in the same group are similar to each other and points across groups are different from each other [10].

Most clustering algorithms work with numeric data [4, 15, 35, 37], but there are some algorithms that can cluster categorical data [14, 16, 19, 29]. Some important Data Mining clustering approaches include [1, 2, 4, 7, 18, 24, 37]. Several aspects make clustering a challenging problem. These aspects include data set size [37, 4], high dimensionality [1, 18], data sparsity [1, 15] and noise [2, 4, 7, 18].

1.1 The EM Algorithm

This article focuses on the well-known Expectation-Maximization (EM) algorithm. Based on previous work related to maximum likelihood estimation EM was formalized as a fundamental statistical algorithm in the seminal article [8] and there has been significant research on it thereafter [5, 27, 31, 35, 36, 28]. A good reference on EM can be found in [22]. The EM algorithm can be used for a variety of statistical purposes and in particular it can be used to perform clustering. The EM algorithm has many desirable features [35, 20]. It has a strong statistical basis, theoretical guarantees about optimality [8], easily explainable results [5], robustness to noise [31], and the ability to handle missing information [8]. Nevertheless, the classical EM algorithm also has several limitations. The quality of the final solution heavily depends on initialization and there is no best way to initialize it [35]. It may converge to a poor locally optimal solution. It needs an undetermined number of iterations to converge. Therefore, it is is not possible to have guaranteed performance. In the case of clustering computations EM gets unstable when variances approach zero. This

problem commonly arises when data sets have binary coded categorical values, many outliers or missing information. Probabilities vanish for outliers making computations undefined or unstable. It may also produce inaccurate results with high dimensional data sets, which are known to be difficult to cluster [1, 3].

There has been work on accelerating and improving EM to work with large data sets, like incremental EM [25], On-line EM [32, 35], EM for high dimensional binary data [29], EM programmed in SQL [27], a block-based EM [33] and Scalable EM [6], but no solution solves all problems listed above. Related work on EM will be discussed in more detail later.

1.2 Contributions and Article Outline

This work proposes several improvements for the EM algorithm to accelerate convergence and to find higher quality solutions. Cluster centroids are initialized with a small perturbation of the global mean. Sufficient statistics are combined with periodic M steps to accelerate convergence. Reseeding of low weight clusters and splitting of high weight clusters help finding higher quality solutions. A fraction of the global variance is added to cluster variance matrices to solve zero variance problems. Mahalanobis distance is used instead of probability to manage outliers. All the improvements are put together in an algorithm called FREM, which stands for Fast and Robust EM. A preliminary version of FREM was introduced in [28] and here we provide a revised version studying its properties in more depth.

The article is organized as follows. Section 2 provides definitions and a notion of the EM algorithm. Section 3 presents improvements to accelerate convergence and to find higher quality solutions, and it introduces the FREM algorithm. Section 4 presents experiments to justify FREM's parameter settings and compares FREM against the Classical EM algorithm and the On-line EM algorithm. Section 5 discusses related work. Section 6 concludes the article.

2 Preliminaries

2.1 Definitions

EM estimates the statistical parameters of a mixture of normal distributions. The multivariate normal (also known as Gaussian) density function for a vector y on d-dimensional space for cluster j, $j \in \{1, \ldots, k\}$, is:

$$P(y; C_j, R_j) = \frac{1}{\sqrt{(2\pi)^d |R_j|}} exp[-\frac{1}{2}(y - C_j)^t R_j^{-1}(y - C_j)], \quad (1)$$

where C_j is called the mean vector and R_j is called the variance matrix; C_j is a d-dimensional vector and R_j is a $d \times d$ diagonal matrix (zeroes off the diagonal). The probability of the mixture is computed as

$$P(y; C, R, W) = \sum_{j=1}^{k} W_j P(y; C_j, R_j). \quad (2)$$

Probability computations are based on Mahalanobis distance [10] instead of Euclidean distance. The basic difference between them is that R_j is used to scale each dimension for distance computation. This is particularly useful for dimensions having different scales and clusters having different spatial sizes (area of the ellipse they represent). The squared Mahalanobis distance from point y_i to cluster j is computed using

$$\delta(y_i, C_j, R_j) = \delta_{ij} = (y_i - C_j)^t R_j^{-1}(y_i - C_j). \quad (3)$$

The input to EM is a data set Y having n d-dimensional points, $Y = \{y_1, y_2, \ldots, y_n\}$, and k, the desired number of clusters. The output are the matrices C, R, W (containing the k means, k variances and k weights respectively), a measure of model quality $L(\Theta)$ (explained below) and a soft partition of Y into k subsets. Matrices C and R are $d \times k$, whereas matrix W is $k \times 1$. To manipulate matrices we use the

following convention for subscripts. For point number we use i; $i \in \{1, 2, \ldots, n\}$. For cluster number we use j; $j \in \{1, 2, \ldots, k\}$ and to refer to one dimension we use l; $l \in \{1, 2, \ldots, d\}$. To refer to one column of C or R we use the j subscript (i.e. C_j, R_j). So C_j refers to the jth cluster centroid and R_j has its corresponding variances. Each R_j is a diagonal covariance matrix that can be manipulated as a vector. To refer to the probability of y_i of belonging to cluster j we use the notation $P(y_i|j) = P(y_i; C_i, R_j)$. In general in the statistical literature all parameters are used as a single set called Θ, i.e. $\Theta = \{C, R, W\}$; Θ is also known as the mixture model. The quality of a clustering solution is measured by a quantity called average log-likelihood (the closer to zero the better), that is computed as

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^{n} log(P(y_i; \Theta)). \tag{4}$$

This is a brief overview of the EM clustering algorithm. EM starts by initializing Θ. Initialization will be described in more detail later. It has two major steps: the E step and the M step. EM iteratively executes the E step and the M step as long as the change in $L(\Theta)$ is greater than an accuracy threshold ϵ or as long as a maximum number of iterations has not been reached. The E step computes $P(y_i; C_j, R_j)$ and $P(y_i; \Theta)$. The M step updates Θ based on the probabilities computed in the E step. EM is theoretically guaranteed to monotonically increase $L(\Theta)$ at each iteration and to converge to a locally optimal and stable solution [8, 35]. Due to lack of space we do not include a more detailed exposition of the EM algorithm, but pseudo-code for the Classical EM algorithm and a more detailed description can be found in [28] and [27].

3 Improving the EM Algorithm

This section contains this article contributions. First, improvements to the EM clustering algorithm are introduced in two major groups: improvements to increase speed and improvements to find higher quality solutions. Second, the FREM clustering algorithm is presented assembling all improvements together.

3.1 Improvements

EM improvements are presented in two groups: improvements to increase speed and improvements to increase the quality of solutions. These groups are orthogonal in the sense that they can be used independently if needed.

Improvements to increase speed.

Sufficient statistics are an essential ingredient to accelerate EM. By definition sufficient statistics are summaries of groups of points [23] (in this case clusters), represented by $Y_1, \ldots, Y_k$. The idea of using sufficient statistics is not new [4, 25, 37], but the way we use them is. The sufficient statistics we are about to present are based on two important assumptions: dimensions are independent (to simplify computations) and they induce a hard partition on Y. Sufficient statistics are stored in matrices M, Q and N (M and Q are $d \times k$ and N is $k \times 1$). Matrix M stores k sums of points (k vectors with d dimensions), matrix Q has k sums of squared points (k diagonal matrices with d variances on the diagonal), and vector N counts the number of points per cluster:

$$M_j = \sum_{\forall y_i \in Y_j} y_i, \tag{5}$$

$$Q_j = \sum_{\forall y_i \in Y_j} y_i y_i^t, \tag{6}$$

$$N_j = |Y_j|. \tag{7}$$

An important related problem is that EM cannot work when variances approach zero because several computations become undefined (e.g. $|R_j|, \delta_{ij}$). This basically means that points belonging to one or more clusters have the same value on some dimensions. We solved this problem by adding to each dimension a constant $\lambda \geq 0$ multiplied by the global variance matrix (Σ) when variances are updated. This constant is chosen small enough to avoid altering the real variance values. This improvement will be used when the M step updates R from sufficient statistics making FREM numerically stable. Then the update formulas based on sufficient statistics used by FREM are:

$$C_j = \frac{1}{N_j} M_j, \tag{8}$$

$$R_j = \frac{1}{N_j} Q_j - \frac{1}{N_j^2} M_j M_j^t + \lambda \Sigma, \tag{9}$$

$$W_j = \frac{N_j}{\sum_{J=1}^{k} N_J}. \tag{10}$$

Based on these equations periodic M steps are used to accelerate convergence in an incremental fashion based on the assumption that n is large. The E step must be run for every point, i.e. n times per scan. The E step updates M, Q, N in a cumulative way after each point is read assigning it to the cluster with highest probability. Sufficient statistics reduce I/O time by decreasing the number of iterations and by allowing parameter estimation periodically as points are being read. Cluster membership is determined in the E step and C, R, W are updated in the M step. By using sufficient statistics FREM can run the M step at different times while scanning Y. At one extreme we could have an On-line EM version [35] that runs the M step after every point. That would be a bad choice. It would be sensitive to the order of points and more importantly, it would not reach the global optimal solution [4, 35] (although in some cases it can get close); this will be shown in the experimental section. On-line EM [32, 35] makes only one iteration and executes the M step after every E step for each point. It represents the fastest version that can be derived from the EM algorithm, but without guarantees about optimality or stability of the solution. At the other extreme, we could have a version that runs the M step after all n points are read. This would reduce it to a standard version of EM with a hard partition and little performance improvement would be achieved. Nevertheless, this latter version would not have any sensitivity to the order of points, would compute the correct value for $L(\Theta)$ and would have the potential of reaching the global optimal solution. Therefore, it is desirable to choose a point somewhere in the middle, but closer to the last scenario. That is, running the M step as few times as possible. When a good approximation to the solution has been reached then normal EM iterations can be executed to make the algorithm converge. It is important to observe that each new M step combines past estimations from previous M steps with the estimation based on the latest subset of points. In other words, FREM does not forget estimations from previous M steps and it uses sufficient statistics M, Q, N as a memory of previous estimations. In general it would be incorrect to reinitialize M, Q, N at each M step because FREM would tend to base its estimation on the last points read.

To control the frequency of M steps we introduce ψ, which will be an important parameter for FREM. The M step is periodically executed ψ times on the first two iterations and only once in subsequent iterations. To some extent ψ resembles the total number of iterations when each subset of points is similar to a previous subset in the sense that the clusters in one subset are similar to those in other subset. We believe that for large n this is a reasonable assumption. Based on ψ the new sample (subset of Y) considered per M step will have size equal to n/ψ. Observe that as $\psi \rightarrow n$ FREM approximates On-line EM and as $\psi \rightarrow 1$ FREM approximates a hard partition EM. Higher n implies more redundancy in the dataset, larger sample sizes and therefore more opportunity to converge to a good solution with less work. Lower n implies less redundancy, fewer M steps on the first two iterations and subsequent more iterations to converge. FREM makes at least three iterations. The first iteration gets a decent solution, the second one tunes it and further iterations make

FREM converge. The default value for ψ will be $\psi = \sqrt{n}$ which implies samples of size $\sqrt{n}$; this setting will be justified in Section 4.

Improvements to increase quality of solutions

Like many other numerical optimization methods EM depends on a good initialization. Initialization is based on the global statistics and dimensionality of Y. The global mean μ and the global covariance matrix Σ can be computed in one scan over the data via sufficient statistics, whose equations are as follows:

$$\mu = \frac{1}{n}\sum_i y_i, \tag{11}$$

$$\Sigma = (\frac{1}{n}\sum_i y_i y_i^t) - \mu\mu^t. \tag{12}$$

Initialization is then done as follows, $W_j = 1/k$, $R_j = \Sigma$, $C_j = \mu \pm \alpha\sigma r$, where σ represents standard deviations per dimension, α controls how far seeds are from the global mean and r is a random number in $[0, 1]$; $\sigma_{ll} = \sqrt{\Sigma_{ll}}, \alpha = 1/(dk)$. It must be noted that as $d \to \infty$, $C_j \to \mu$ (seeds get closer to μ, the global centroid of Y). It is easy to prove that μ is the closest point to all points in Y. Therefore centroid seeds are small perturbations of μ. On the other hand, at the beginning of each iteration sufficient statistics M, Q, N are initialized to zero. Notice that FREM could also be initialized using a sample of k different points, but ours was better.

As discussed in Section 1 EM is often criticized for finding a sub-optimal solution [5, 6, 31], a common problem with clustering algorithms [9, 10] and optimization methods. Most of the times sub-optimality involves several clusters incorrectly grouped together as one while other clusters have almost no points or are empty. So we propose splitting heavy clusters to reach higher quality results. We introduce a minimum weight threshold ω to control splitting and reseed low-weight clusters. This will be an important parameter for FREM. Let a be the index of the weight of some cluster s.t.

$$W_a \leq \frac{\omega}{k} \tag{13}$$

Let b be the index of the weight of the cluster with highest weight. Then $C_b - \sqrt{vect[R_j]}$ and $C_b + \sqrt{vect[R_j]}$ are the new centroids, where the right terms represent one standard deviation. The old value of C_a disappears and its points change membership to heavier clusters. Then index a is incremented and b is decremented to analyze the next potential low-weight cluster to reseed. Splitting gets repeated until there are no more clusters below ω/k or $a \geq b$. Cluster splitting will be done in the M step after C, R, W are updated only on the first iteration. After the first iteration FREM will not split clusters to avoid incorrectly splitting clusters. This is based on the fact that FREM will execute frequent M steps on the second iteration and will iterate giving points the chance to change membership to the best cluster. If some heavy cluster is incorrectly split in the first iteration the second and further iterations will regroup its points. This improvement will lead to higher quality solutions in most cases, but it is not guaranteed that the global optimal will be reached.

Outliers are points that cannot be fitted adequately by the model. These points are far from any cluster centroid C_j and then $P(y_i; \Theta) \approx 0$; that is, y_i has almost zero probability of belonging to any cluster. The solution we propose is simple; we use the closest cluster based on Mahalanobis distance to update sufficient statistics. FREM does not multiply the probability approximation by cluster weights. Also, almost null probabilities make EM numerically unstable; in this case due to numerical precision. So FREM has a lower threshold ϕ for probabilities. If $P(y_i; \Theta) < \phi$ then M, Q, N are updated using the nearest neighbor of y_i based on Eq. 3.

When working with data sets with dimensions of very different scales or having high dimensionality, or datasets where the set of points considered at each M step is significantly different from previous subsets

Input: $Y = \{y_1, y_2, \ldots, y_n\}$ and k.
Output: $\Theta = \{C, R, W\}$ and $L(\Theta)$
$\psi = \lfloor \sqrt{n} \rfloor$, $\omega \leftarrow 0.2$, $\alpha \leftarrow (dk)^{-1}$, $I \leftarrow 1$, $\lambda \leftarrow 0.01$ /* Parameters */
FOR $j = 1$ TO k DO
  $C_j \leftarrow \mu \pm \alpha r$ diag$[\sigma]$, $R_j \leftarrow \Sigma$, $W_j \leftarrow 1/k$
END
WHILE $|L(\Theta)^I - L(\Theta)^{I+1}| > \epsilon$ or $I \leq 3$ DO /* Iterate until convergence */
  $L(\Theta) \leftarrow 0$
  $M_j \leftarrow \bar{0}$, $Q_j = diag[\bar{0}]$, $N_j \leftarrow 0$
  FOR $i = 1$ TO n DO
    Estep()
    IF((i mod ψ) $= 0$ and $I \leq 2$) THEN
      Mstep()
      IF I=1 THEN splitClusters() ENDIF
    END
  END
  Mstep()
  IF $I <= 2$ THEN
    FOR $j = 1$ TO k DO $R_j \leftarrow \sum_{J=1}^{k} W_J R_J$ END
  END
  $I \leftarrow I + 1$
END
Estep()
  FOR $j = 1$ TO k DO
    $\delta_{ij} \leftarrow \delta(y_i, C_j, R_j)$,
    $p_{ij} \leftarrow ((2\pi)^d |R_j|)^{-1/2} exp(-\delta_{ij}/2)$
    $x_{ij} \leftarrow p_{ij} / (\sum_{j'} p_{ij'})$
  END
  IF $\sum_j p_{ij} > 0$ THEN /* cluster with highest probability */
    Let m be s.t.$p_{im} \geq p_{ij}, \forall j \in 1 \ldots k$
  ELSE /* closest cluster */
    Let m be s.t.$\delta_{im} \leq \delta_{ij}, \forall j \in 1 \ldots k$
  END
  $M_m \leftarrow M_m + y_i$, $Q_m \leftarrow Q_m + y_i y_i^t$, $N_m \leftarrow N_m + 1$
  $L(\Theta) \leftarrow L(\Theta) + log(\sum_j W_j p_{ij})/n$
Mstep()
  FOR $j = 1$ TO k DO
    IF $N_j > 1$ THEN
      $C_j \leftarrow M_j / N_j$
      $R_j \leftarrow Q_j/N_j - M_j M_j^t / N_j^2 + \lambda\Sigma$
      $W_j \leftarrow N_j / \sum_{J=1}^{k} N_J$
      $|R_j| = \Pi_{l=1}^{d} R_{lj}$
    END
  END

Figure 1: The FREM clustering algorithm

it is necessary to adjust covariance matrices so that the spatial size of some cluster is not so large so as to absorb points that may belong to other cluster and not so small so as to reject most points. At the end of the first and second iteration all covariance matrices R_j will be adjusted (tied) to be the same as follows; $R_j = \sum_{J=1}^{k} W_J R_J$ for $j = 1 \ldots k$. This adjustment assures among other things that covariance matrices of heavier clusters have more influence, all clusters have the same spatial size at the beginning of the second and third iteration and that each variance preserves the scale of each dimension. With this adjustment FREM proceeds from a numerically stable approximation to Θ. On the other hand centroids C_j do not need to be adjusted since they will be close to the final solution and then can be considered initial points at the beginning of the 3rd iteration. When FREM iterates making one M step after doing n partial E steps for all points (a full E step) each R_j will quickly get more accurate.

3.2 The FREM Algorithm

In this section all improvements are put together in one place. The FREM algorithm pseudo-code is shown in Figure 1. The input is a set of points $Y = \{y_1, y_2 \ldots y_n\}$, each having d dimensions and k, the desired number of clusters. The output will be the mixture model $\Theta = \{C, R, W\}$ (centroids, variances and weights) and $L(\Theta)$ (a measure of model quality) as defined in Section 2. Variable x_{ij} represents the weighted proba-

bility that point y_i belongs to cluster j in a soft partition; even though FREM produces a hard partition these numbers can be used to individually judge cluster membership in a soft assignment. FREM's iterations can be summarized as follows. The first iteration accelerates convergence and increases quality of solutions by splitting clusters, The second iteration accelerates convergence and regroups points when clusters were incorrectly split. The third and further iterations eliminate sensitivity to the order of points (if any), compute an accurate value for $L(\Theta)$ and make FREM converge to a stable solution. The mixture parameters of the model Θ are updated when there are at least two points in the cluster to avoid biased estimations using one point.

The first two iterations have the goal of reaching a close approximation to a good clustering solution. The first iteration splits those clusters that have too many points and re-seeds those that are almost empty. The first two iterations are somewhat sensitive to the order of points and their $L(\Theta)$ computation may not be accurate since Θ is updated at every M step and $L(\Theta)$ is not recomputed for old points. Therefore, further iterations are needed to have consistency between Θ and $L(\Theta)$. The third, fourth and subsequent iterations represent an execution of EM based on sufficient statistics and a hard partition. The third and further iterations, with the E step and the M step looking at the entire data set Y, are essential to compute an accurate value for $L(\Theta)$ like Classical EM, to reduce sensitivity to the order of points and to deal with clusters having high overlap. Therefore, after the second iteration FREM monotonically increases $L(\Theta)$ and converges to a locally optimal solution like Classical EM. The FREM algorithm has time complexity $O(dkn)$ per iteration. The E step is executed n times per iteration computing cluster membership probabilities (Eq. 1), probability of the mixture (Eq. 2) and updating sufficient statistics based on Eq. 5, 6 and 7 in time $O(dk)$. The mixture parameters C, R, W are updated in the M step using Eq. 8, 9 and 10 taking $O(dk)$ each. For a more detailed analysis of FREM's time complexity consult [28].

4 Experimental Evaluation

To evaluate quality of results and performance we tested FREM with real and synthetic data sets. FREM is compared against the Classical EM clustering algorithm [8, 31, 35] and against the On-line EM algorithm [35, 6]. Since μ, Σ are computed only once per data set that time is not included in the total time. All algorithms were implemented in the C^{++} language. Data sets were stored as plain text files and were never maintained or cached in memory. All experiments were run on a Sun server running at 800 MHz, having several GB of disk space and 128MB on main memory. The implementation used text files as input. Some time improvement could be expected if binary files were used since no parsing would have to be done; we include some experiments about this aspect. Another observation is that the FREM, EM and On-line EM implementations use dynamically sized matrices, and these require heavy pointer manipulation. Using fixed sized matrices without pointers would also improve times since memory allocation would be easier and accessing arrays by subscripts would be faster. In short, these performance experiments were performed under pessimistic conditions without making optimizations that would affect usability.

In general real data sets are more challenging to cluster and so we use them to study FREM's parameters. The first section presents extensive experiments with real data sets to justify parameters settings and to compare FREM against Classical EM and On-line EM. The second section compares FREM, EM and On-line EM with synthetic data sets assessing correctness, sensitivity to changing characteristics in the input data and performance.

4.1 Experiments with Real Data Sets

Description of Data sets

In this section we evaluate quality of results and convergence speed with real data sets from different domains. These data sets cover a wide spectrum including small, medium and large data sets, low, medium

and high dimensionality, different scales for each dimension and dimensions with missing information. *Astronomy* is a large data set containing information about stars from an observatory. The dimensions of stars include their position in the sky as (x,y) coordinates, magnitude and brightness; for this data set $n = 368,891$ and $d = 4$. This data set presented highly skewed data, significant cluster overlap and big differences in dimension scale. The *basket* data set contained a sample of aggregated information of transactions from a large data warehouse of a grocery retailer. The basket dimensions are total sales amount, total cost, number of items bought, number of distinct articles and number of distinct departments visited. Measurements about the basket were clean and there were little zero-variance problems, no missing information and no highly skewed data. That is, this was a well-behaved data set. For this data set $n = 100,000$ and $d = 5$. The *medical* data set contains information for a set of patients being treated for heart disease. The numeric dimensions include 4 artery disease (blockage) percentages in the [0-100] range and 9 numbers describing perfusion measurements in the [-1,1] range corresponding to 9 specific regions of the heart; for this data set $n = 655$ and $d = 13$. This data set presented serious problems with zero variances in most clusters, significant cluster overlap and a significant portion of outliers reflecting the variety of medical conditions of each person. The *coil* data set contains several measurements about European rivers. Categorical values were coded as binary dimensions. Dimensions had very different scales and some columns had the same value in 90% of the rows causing variances to be zero. This data set was small with $n = 200$ and $d = 18$. The *UScensus* data set contains demographic data from the 1990 US census. Record identifiers were removed and all remaining numbers were treated as clustering dimensions. This data set was high dimensional and sparse having zero values on most dimensions making it hard to cluster. In this case $n = 8000$ and $d = 68$.

Experiments varying parameters individually

As explained above FREM has defaults for its parameters. This section justifies those settings. These defaults will be used to compare FREM, Classical EM and On-line EM with real and synthetic data sets. The most important parameters are ψ, the number of M steps on the first and second iterations, and ω, the threshold to reseed low-weight clusters. Parameters λ and ϕ are used just to make FREM numerically stable and they have settings that introduce a minimum change to the clustering problem. So we did not conduct experiments varying them. We did not include experiments varying ϵ or the maximum number of iterations because FREM and Classical EM are always stopped at a locally optimal and stable solution. The maximum number of iterations was set at 500 so that FREM always stops at a stable solution but there was no case reaching this limit. FREM was run 10 times for each different parameter setting to cluster each real data set. We decided to set $k = 20$ to make the clustering problem more challenging. We will later show experiments with lower k values to compare FREM against Classical EM and On-line EM. In general the default settings for FREM's parameters were $\psi = \sqrt{n}$, $\omega = 0.2$, $\epsilon = 1.0e-5$, $\lambda = 0.01$ and $\phi = 1.0e-300$.

Table 1 compares the initialization proposed in Section 3 against the most widely used initialization for centroid based iterative algorithms (EM and K-means). Our initialization uses small changes to μ to initialize C. The standard deviation factor for seeding was set $\alpha = 1/(dk)$; in general this number worked well. We found a wide range for α that went from $1/k$ to $1/(10dk)$ that produced solutions of comparable quality. We decided to set α to $1/(dk)$ by default. Due to lack of space we do not include experiments varying α. The alternative initialization uses k different points randomly selected from Y. This initialization, known as the Forgy method, is proven to produce good results [17, 4]. For this set of tests FREM is run without acceleration or reseeding. From the table we can see that in general the mean-based initialization (μ) produces slightly higher quality solutions for both the best and the average case. The drawback is that the number of iterations is sometimes increased. These results indicate that a mean-based initialization is better for higher dimensional data which is the case of the *medical* and the *UScensus* data sets; in those two cases $L(\Theta)$ is comparatively higher than in the other 3 cases.

Table 2 compares the elapsed time per iteration of Classical EM and FREM with text and binary files. These experiments attempt to quantify how much performance is sacrificed by using text files. In each case

Table 1: Comparing different initializations for FREM at $k = 20$

Data set	Init.	Best run μ	Best run k points	Average run μ	Average run k points
Astronomy	$L(\Theta)$	-21.85	-21.85	-21.87	-21.87
	iters	31	23	22	18
Basket	$L(\Theta)$	-9.12	-9.11	-9.15	-9.16
	iters	31	18	26	12
Medical	$L(\Theta)$	-12.13	-12.53	-12.48	-12.86
	iters	12	18	10	15
Coil	$L(\Theta)$	-49.33	-49.42	-49.60	-49.72
	iters	10	12	8	9
USCensus	$L(\Theta)$	-7.91	-9.29	-8.85	-10.49
	iters	30	27	19	18

Table 2: Time per iteration in seconds: average run

Dataset	k	FREM text	FREM binary	EM text	EM binary
Astronomy	10	43.91	26.45	157.72	98.98
Basket	10	13.21	8.20	44.54	29.71
Medical	10	0.22	0.11	0.55	0.39
Coil	10	0.06	0.03	0.22	0.17
UScensus	10	9.50	4.95	23.62	16.43
Astronomy	20	72.02	44.53	321.37	186.24
Basket	20	23.62	14.91	94.58	55.58
Medical	20	0.33	0.20	1.05	0.71
Coil	20	0.17	0.05	0.38	0.28
UScensus	20	16.31	8.79	44.38	31.31

the file is opened at the beginning of the iteration and closed at the end. For binary files there is no parsing. In all cases FREM is faster than EM. For text files FREM takes about one third of time compared to EM. For binary files FREM takes about one fourth of the time EM needs. Clearly both algorithms are affected by the parsing that needs to be done with text files. However, FREM is more impacted than EM with text files because times with binary files decrease by almost 50% while those for EM decrease by 30%. A close code inspection reveals that Classical EM performs many more computations than FREM per iteration because it produces a soft partition. The computations that mainly make EM slower are those required to estimate soft-partitioned sufficient statistics for C and R which require dk computations to be updated. For more details this can be verified in E step in the EM pseudo-code in [28]. These experiments show that using binary files would benefit FREM more than EM.

Table 3 is one of the most interesting experiment summaries showing how FREM is accelerated. This table shows what happens when ψ is varied going from one extreme doing only one M step per iteration to the other extreme doing n M steps per iteration. The table shows averages; results for the best run are similar, but due to lack of space they are omitted. FREM used ψ in the first two iterations as proposed in Section 3 and did not reseed low-weight clusters. Then FREM iterates until there is a minimal change in $L(\Theta)$. Since there are several ψ values in the middle we selected $\sqrt{n}$ and $log(n)$ as good representatives because they are smaller than n and they are proportional to n. We conjecture that ψ should be directly proportional to n. The pattern is not perfect but it can be observed that, in general, the lowest number

Table 3: Varying ψ (number of M steps) at $k = 20$: average run

Data set	ψ	1	2	3	$log(n)$	$\sqrt{n}$	$n/2$	n
Astronomy	$L(\Theta)$	-21.87	-21.86	-21.87	-21.85	-21.85	-21.88	-21.87
	iters	24	22	21	19	21	20	19
Basket	$L(\Theta)$	-9.15	-9.15	-9.15	-9.13	-9.12	-9.19	-10.00
	iters	29	22	26	19	16	29	22
Medical	$L(\Theta)$	-12.60	-12.78	-12.68	-12.73	-12.46	-12.81	-12.79
	iters	11	13	12	9	9	12	13
Coil	$L(\Theta)$	-49.81	-48.38	-48.56	-48.39	-48.53	-49.07	-48.87
	iters	8	8	10	7	7	10	10
USCensus	$L(\Theta)$	-8.99	-8.31	-8.65	-6.01	-6.14	-7.88	-7.37
	iters	22	23	21	15	15	18	15

Table 4: Varying ω to split clusters at $k = 20$: best run

Data set	ω	-1	0	0.2	0.5	1.0
Astronomy	$L(\Theta)$	-21.85	-21.85	-21.84	-21.85	-21.84
	iters	21	20	16	20	32
Basket	$L(\Theta)$	-9.15	-9.14	-9.08	-9.10	-9.14
	iters	21	24	10	12	32
Medical	$L(\Theta)$	-12.45	-12.37	-12.30	-12.04	-12.20
	iters	16	13	13	18	14
Coil	$L(\Theta)$	-48.58	-48.43	-47.77	-48.30	-47.96
	iters	8	9	8	11	11
USCensus	$L(\Theta)$	-7.72	-7.82	-6.36	-5.48	-6.11
	iters	17	19	17	19	21

of iterations shows up in $log(n)$ or $\sqrt{n}$. A higher number of iterations tends to appear more on the left side of the table. Another outstanding feature is that in most cases the highest quality of solutions appears in either $log(n)$ or in $\sqrt{n}$. Surprisingly $\psi = n$ (on-line version) produced better results than $\psi = 1$ (slow version) in two cases. Therefore this table shows that ψ helps achieving faster convergence but also reaching higher quality solutions. FREM escapes local maxima by making periodic M steps. It is noteworthy that $\psi = log(n)$ produced better results in two cases, but we chose $\psi = \sqrt{n}$ as the default value because of the symmetry between number of M steps and sample size.

Now we proceed to study FREM with acceleration, reseeding of low weight clusters and splitting of high weight clusters. Tables 4 and 5 show the best and average run varying ω, the threshold to reseed low-weight clusters and split high-weight clusters. Recall that clusters s.t. $W_j \leq \omega/k$ are reseeded on the first iteration after every M step. At one extreme $\omega = -1$ represents a conservative no-reseeding strategy and at the other extreme $\omega = 1$ represents an aggresive reseeding strategy because clusters are always reseeded. When $\omega = 0$ clusters are reseeded only when there are empty clusters and thus it represents a conservative strategy. Finally $\omega = 0.2$ and $\omega = 0.5$ represent a moderate strategy. From both tables it can be seen that higher quality solutions appear on the right side where reseeding/splitting is more aggressive. In general $\omega = -1$ produced the lowest quality solutions. In some cases solutions were slightly better when $\omega = 0$. In all cases the best solutions were found at $\omega = 0.2$ or $\omega = 0.5$. To our surprise the most aggresive reseeding found better solutions than $\omega = 0.2$ in some cases. The second important fact is that the number of iterations required for $\omega = 0.2$ or $\omega = 0.5$ were less or equal than those required for FREM without reseeding. In other words, reseeding accelerated convergence in some cases. For the experiments to be presented later we

Table 5: Varying ω to split clusters at $k = 20$: average run

Data set	ω	-1	0	0.2	0.5	1.0
Astronomy	$L(\Theta)$	-21.86	-21.87	-21.85	-21.86	-21.85
	iters	18	17	15	18	25
Basket	$L(\Theta)$	-9.17	-9.15	-9.10	-9.12	-9.15
	iters	19	24	8	9	21
Medical	$L(\Theta)$	-12.77	-12.66	-12.52	-12.40	-12.47
	iters	11	12	11	15	11
Coil	$L(\Theta)$	-48.66	-49.03	-48.11	-48.44	-48.23
	iters	8	8	8	8	9
USCensus	$L(\Theta)$	-8.34	-8.21	-6.94	-6.36	-6.46
	iters	14	17	16	16	16

chose $\omega = 0.2$ as the default value, but $\omega = 0.5$ could have worked as well. We preferred $\omega = 0.2$ because it is more conservative. Summarizing, reseeding low weight clusters and splitting high weight clusters is needed to reach higher quality solutions. The number of iterations may increase, but in general it will be less or equal than the number of iterations required without reseeding/splitting.

Comparing FREM, Classical EM and On-line EM

We now proceed to compare FREM against Classical EM which is known to produce high quality solutions and On-line EM which is very fast. The same type of initialization was used to make a fair comparison. All algorithms are initialized using the global mean as explained in Section 3. This initialization produced slightly higher quality solutions than initializing with a sample of k points as shown before. FREM and EM are stopped until they converge with the same tolerance $\epsilon = 10^{-5}$ for $L(\Theta)$. This setting assured both algorithms converged to a stable and locally optimal solution. FREM's most important parameters are ψ, the number of periodic M steps and ω, the threshold to reseed low-weight clusters. Based on the experiments presented before their default settings were $\psi = \sqrt{n}$ and $\omega = 0.2$. In order to cluster real data sets, specially high dimensional ones, the regularization constant λ was required because of zero variances, which was $\lambda = 1.0e - 2$ to introduce a negligible change in the real variance values. The default value for ϕ was 1.0e-300 which is close to the smallest number in double precision. The standard deviation factor for seeding was set to $\alpha = 1/(dk)$. In general k, the desired number of clusters, was the only input parameter.

Results for the best run out of 10 runs are summarized in Table 6 and results for their average are summarized in Table 7. The three algorithms were run 10 times with $k = 5$ and 10 times with $k = 10$. Times are given in seconds. Since the globally optimal solution cannot be known we can only use $L(\Theta)$ to compare quality of results. The closer $L(\Theta)$ is to zero the better. For FREM we include measurements at the end of the 3rd scan (FREM-3scans) and after it has converged (FREM-converge). EM is always measured after it converges. A second scan is needed to compute an accurate value for $L(\Theta)$ for On-line EM, but that time is not included in the table.

This is a summary of Table 6. For the best run FREM bound better solutions than Classical EM in eight cases. In one case it found a worse solution (*Astronomy* $k = 10$) and in one case they were tied (*Astronomy* $k = 5$). In four cases FREM-3scans found better solutions than EM but in general it came behind. On-line EM was always in the last place. Performance-wise On-line EM was the fastest followed by FREM-3scans. FREM-converge came in third place and Classical EM came in fourth place. Now we summarize Table 7. For the average run Classical EM fares better than in the best run. In two cases (*Astronomy* $k = 10$ and *Medical* $k = 10$) it finds higher quality solutions than FREM-converge and in four cases it is better than FREM-3scans. The winner is FREM-converge which finds the best solution in 8 cases. On-line EM is again

Table 6: Quality of results and performance with real data sets: best run

Data set	FREM 3 scans	FREM converge	OnLine EM	EM
	$L(\Theta)$ and time in seconds			
Astronomy $k = 5$	-22.25	-22.06	-23.41	-22.06
	91	452	48	1598
Astronomy $k = 10$	-22.00	-21.97	-23.33	-21.93
	191	610	66	3779
Basket $k = 5$	-9.95	-9.73	-10.56	-10.06
	30	67	14	888
Basket $k = 10$	-8.36	-8.27	- 9.60	-8.42
	45	254	20	3942
Medical $k = 5$	-24.23	-24.08	-29.41	-24.15
	1	2	1	6
Medical $k = 10$	-21.42	-20.85	-29.79	-20.57
	1	2	1	13
Coil $k = 5$	-53.35	-52.77	-58.13	-54.28
	1	1	1	4
Coil $k = 10$	-49.74	-49.39	-56.43	-50.98
	1	1	1	8
USCensus $k = 5$	-21.95	-17.35	-64.63	-21.11
	25	62	7	415
USCensus $k = 10$	-13.39	-10.23	-83.86	-12.26
	39	158	11	741

Table 7: Quality of results and performance with real data sets: average run

Data set	FREM 3 scans	FREM converge	OnLine EM	EM
	$L(\Theta)$ and time in seconds			
Astronomy $k = 5$	-22.31	-22.11	-23.67	-22.16
	104	392	47	2166
Astronomy $k = 10$	-22.07	-22.01	-23.54	-21.99
	193	414	65	3288
Basket $k = 5$	-9.99	-9.83	-10.73	-10.17
	30	42	14	310
Basket $k = 10$	-8.56	-8.36	-9.98	-8.50
	48	94	20	911
Medical $k = 5$	-24.52	-24.29	-31.03	-24.45
	1	1	1	6
Medical $k = 10$	-21.62	-20.99	-30.64	-20.81
	1	2	1	10
Coil $k = 5$	-53.74	-52.94	-58.30	-54.51
	1	1	1	5
Coil $k = 10$	-50.03	-49.61	-56.60	-51.64
	1	1	1	6
USCensus $k = 5$	-23.27	-20.17	-66.60	-28.11
	25	82	7	273
USCensus $k = 10$	-15.04	-12.14	-83.86	-12.91
	38	157	11	605

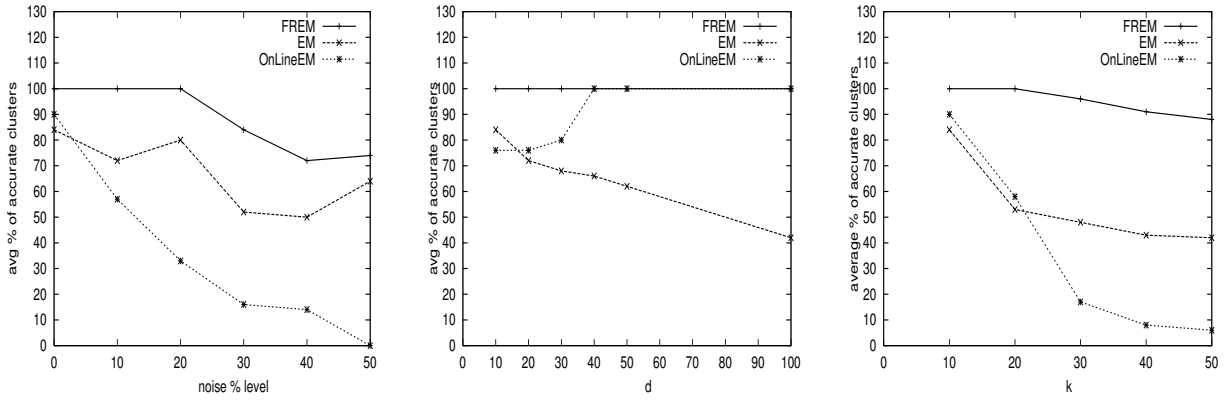

Figure 2: Quality of results: average % of accurate clusters in synthetic data sets

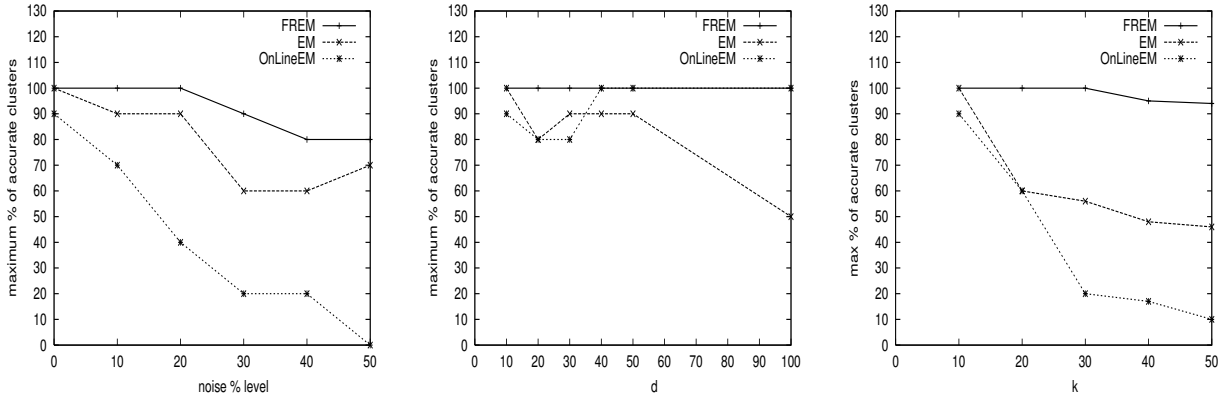

Figure 3: Quality of results: maximum % of accurate clusters in synthetic data sets

the worst. Performance-wise results are similar to the best case with FREM being faster than Classical EM and On-line EM being the fastest. These results support the idea that in some cases it may be reasonable to stop FREM after the third iteration.

4.2 Experiments with Synthetic Data Sets

This section presents experiments with synthetic data. Each experiment was run 10 times unless otherwise indicated. The times and quality of results are averaged and such average is reported. We used a synthetic data generator that created mixtures of normal distributions. In general real data sets have missing information, skewed distributions, varied scales, high dimensionality and noise. With that in mind we created synthetic data sets varying n (size), d (dimensionality), k (embedded number of clusters), η (a noise level) and β (a scale for each dimension). For a more detailed discussion on how we generated synthetic data consult [28]. We now to proceed to analyze how FREM, Classical EM and On-line EM behave as these parameters vary.

Quality of results

Noise, dimensionality and a high number of clusters have been shown to be troublesome aspects for clustering algorithms. In the following set of experiments we analyze the algorithm behavior when we vary them. The set of experiments is not complete as we do not show what happens when we vary combinations of these parameters, but we tried to choose values that are common in a data mining environment. These

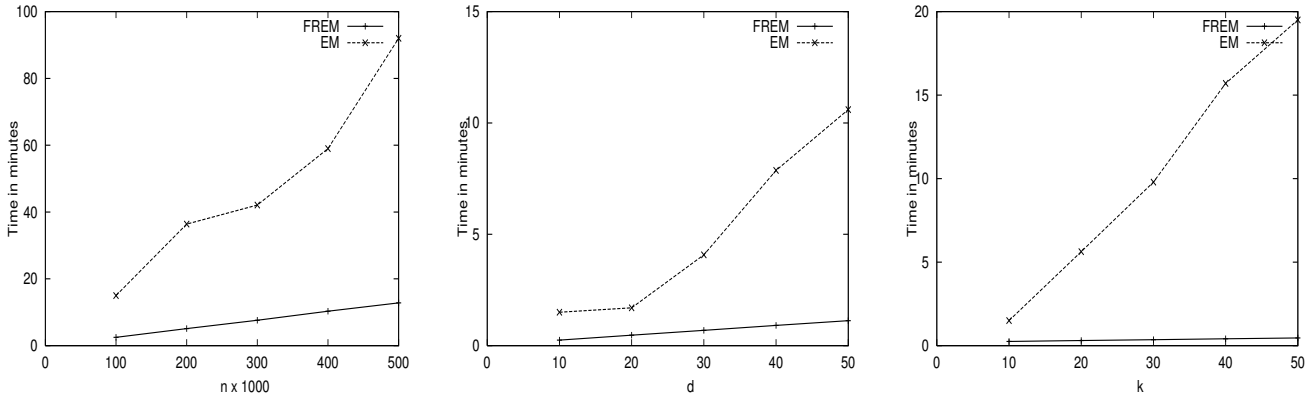

Figure 4: Time varying n, d and k

experiments were essential to test algorithm correctness.

Accuracy is the main quality concern for clustering. Since we are clustering synthetic data we already know the "true" clusters. In our case one cluster is considered accurate if there is no more than ϵ error in its centroid *and* weight. This is computed as follows. Let c_j, w_j be the correct mean and weight respectively (as given by the data generator) of cluster j having estimated C_j, W_j values by the algorithm, and let e be the relative estimation error. Then cluster j is considered accurate if $(1/d)\sum_{l=1}^{d}|c_{lj} - C_{lj}|/|c_{jl}| \leq e$ and $|w_j - W_j|/w_j \leq e$.

This is a high requirement since noise and data skew can easily distort estimations. For the experiments below we set $e = 0.1$. That is, we will consider a cluster to be correct if it differs by no more than 10% of its "true" mean and weight. The process to check clusters for accuracy is as follows. Clusters are generated in a random order and clustering results may also appear in any order given initialization on random seeds. So there are many permutations in which clusters may correspond to each other ($k!$). To simplify checking we build a list of pairs (estimated,generated) clusters. Estimated clusters are sorted in descending order by weight. Then each estimated cluster is matched with the closest synthetic cluster and both are eliminated from further consideration. For each pair we compute its accuracy error. If it is below e the cluster will be considered accurate. Any unmatched estimated cluster is considered inaccurate.

Figures 2 and 3 compare FREM vs. EM and On-line EM. Algorithms are initialized in the same manner and are run with $k+1$ (k is the actual number of embedded clusters and the extra cluster accounts for noise). The three graphs in Figure 2 show the average percentage of correctly found clusters increasing noise level, dimensionality and number of clusters. In all cases FREM clearly performs best. Noise affects both, but it is noteworthy EM performs almost equally better at the highest level of noise. Observe that FREM and On-line EM are minimally affected by dimensionality. Figure 3 compares algorithms in the best run out of 10 runs. In this case results for EM are better than the average EM case above, but FREM finds better solutions anyway. On-line EM finds lower quality models than FREM and EM for increasing noise and number of clusters It is interesting to observe that On-line EM is as good as FREM for higher dimensionality. The difference in quality becomes more significant as the number of clusters and dimensionality increase. The difference at higher levels of noise is not as big. Due to lack of space we do not show $L(\Theta)$, but it is always higher in the mixture models computed by FREM, consistent with the shown graphs.

Time

Time varying size, dimensionality and desired number of clusters is shown in Figure 4. In general On-line EM (with one scan) takes between one third and one half the time FREM (with three scans) takes to cluster a synthetic data set; those times are not shown. The synthetic data sets defaults were $n = 10k$, $d = 10$ and $k = 10$. All times exhibit linear behavior and FREM outperforms EM by one order of magnitude. Since

these data sets are synthetic and clusters are well separated it is easier for FREM to find a good solution and it converges in just 3 iterations. However, in order for EM to have equivalent performance it would need to converge to an acceptable solution in only 3 iterations. In general this is unlikely.

4.3 Limitations

FREM solves several problems related to clustering with the EM algorithm, but it is not the final answer. Its most important limitations are the following. Like Classical EM, it still needs to read all the n points at each iteration. Even though we showed experimental evidence about the default settings for ψ and ω there may be clustering problems that require careful tuning. The regularization constant λ is helpful to deal with common zero-variance problems, but it introduces a small variation in actual variance values. Computing probabilities in high dimensionality is troublesome. Given our outlier handling approach points may be assigned to the closest (and not most probable) cluster like K-means.

5 Related Work

There exist many scalable clustering algorithms. Some well-known examples include BIRCH [37], SKM [4], OPTI-GRID [18], CLIQUE [2], CURE [15], PROCLUS [1] and DBSCAN [26]. There is work on accelerating K-means using trees [30]. We did not compare FREM with any of these clustering algorithms because FREM and EM compute a statistical model and optimize log-likelihood (equation 4), whereas these algorithms compute clusters based on distance or density. That is, they have different optimization goals.

The idea of improving EM to work with large data sets is not new. An important previous work comes from the Machine Learning community. Neal and Hinton study the idea of using sufficient statistics in [25]. In this work the authors analyze several variants of EM viewing the log-likelihood optimization as an optimization of a Physics energy function. One of the variants they present is incremental and the key idea is to use sufficient statistics. They describe a simple mixture problem with $d = 1$ and $k = 2$ and show some experimental evidence of the superiority of the incremental approach. Their study is not complete as it does not address the problem of clustering large data sets with high dimensionality. The problem of minimizing disk access is not analyzed. Problems related to numerical stability, proper initialization, and outlier handling are not addressed in their work either.

The second important work that analyzes how to make EM work with large data sets was introduced in [4, 5, 6]. A scalable K-means is presented in [4] and this algorithm is used as a framework to build a very similar scalable EM algorithm in [5, 6]. In [5, 6] the authors present a heuristic scalable EM algorithm (SEM) that works in memory and also summarizes points through their sufficient statistics. This summarization is done by making compression in two phases. To avoid locally optimal solutions they re-seed empty clusters and they estimate several models concurrently sharing information across models. They require the user to specify what fraction the working buffer should be from the entire database. The buffer size is a critical parameter and its typical value is 1%. The authors cannot explain why this value gives best results. This work is different from ours in several aspects. Our initialization is based on global statistics whereas theirs is based on sampling; although FREM and SEM would work with either one. They use K-means to cluster points in main memory whereas we use EM itself. Our re-seeding strategy is different as we split high-weight clusters. They do not address the problem of handling outliers or zero variances. FREM does not estimate several models concurrently as that is expensive. FREM requires several passes over the data whereas theirs requires only one. However, EM is a CPU bound algorithm and SEM makes it even more CPU intensive rendering a slow algorithm as explained in [12]. Their log-likelihood computation is inaccurate because it is based on an approximation with sufficient statistics; an accurate computation would require a 2nd scan. Finally, they do not show SEM converges to a stable solution. They assume SEM converges based on K-means convergence (which is known to be an inferior algorithm) on compressed point sets but it is never shown that the solutions found by SEM are stable and locally optimal when making a full iteration with the

E and the M steps on all the points. In [12] it is shown that their scalable clustering algorithm is not faster than K-means and in some cases it is even slower. Also, they show that the quality of solutions is about the same. In short, their version is not consistently better than K-means. For the reasons described above we believe a direct comparison with Classical EM and On-line EM was more appropriate.

The idea of helping EM clustering deal with zero variances has been studied before in [34]. The authors use Bayesian regularization techniques to deal with zero covariances. Their approach is similar to ours, but it is different in two aspects: we use regularization together combined with sufficient statistics and we scale the regularization constant by the global covariance matrix so that Mahalanobis distance is properly computed for dimensions with different scales. Moving cluster centroids to better locations in K-means clustering is presented in [13]. This approach is different from ours in several aspects. It is based on Euclidean distance, only one bad cluster is discarded at the end of an iteration, while we may split several clusters in one M step and the splitting criterion is more complex. Other convergence acceleration methods are described in [22].

6 Conclusions

This work proposed several improvements to the EM clustering algorithm to accelerate convergence and to find higher quality solutions. These improvements were incorporated into a new algorithm called FREM. The most important improvements include mean-based initialization, periodic M steps based on sufficient statistics and low weight cluster reseeding combined with cluster splitting. FREM has two main parameters, the number of periodic M steps per iteration and the threshold to reseed low-weight clusters. These and other additional parameters have defaults. In general the desired number of clusters is the only input parameter. In the experimental section the proposed improvements were studied in isolation and the default parameter settings were justified. Experiments show the quality of solutions and speed of the algorithm with both real and synthetic data sets. FREM can produce a reasonable solution in just three iterations or it can be run until it converges to a locally optimal solution like EM. In general FREM finds solutions of higher quality than EM in fewer iterations. Compared to On-line EM running FREM with three iterations is only three times slower, but it always finds better solutions.

The proposed improvements can be extended to a general EM approach for other statistical problems. This will work as long as there is a way to estimate sufficient statistics and there is a way to move bad solutions to new locations. Other data mining problems solved with EM include factor analysis or time series, where an incremental learning approach may be feasible. We want to provide theoretical evidence about FREM's speed of convergence and the quality of solutions it finds. Clustering data with categorical attributes [16], finding outliers [21], or finding clusters in projections of high dimensional data [1] are also interesting problems for further research.

Acknowledgements

We thank Emory University for providing the *medical* data set and the California Institute of Technology for providing the *astronomy* data set. The *coil* data set and the *UScensus* data set were obtained from the University of California at Irvine Machine Learning Repository. We thank the anonymous reviewers for their comments. The first author would like to thank the reviewer who suggested justifying parameter settings and the reviewer who suggested showing both best and average runs.

References

[1] C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD Conference*, pages 70–81, 2000.

[2] R. Agrawal, J. Gehrke, D. Gunopolos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD Conference*, pages 94–105, 1998.

[3] K. Beyer, J. Goldstein, and R. Ramakrishnan. When is nearest neighbor meaningful? In *ICDT Conference*, pages 217–235, 1999.

[4] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. ACM KDD Conference*, pages 9–15, 1998.

[5] P. Bradley, U. Fayyad, and C. Reina. Scaling EM clustering to large databases. Technical report, Microsoft Research, 1999.

[6] P. Bradley, C. Reina, and U. Fayyad. Clustering very large databases using EM mixture models. In *Proc. IEEE ICPR Conference*, pages 2076–2080, 2000.

[7] M. Breunig, H.P. Kriegel, P. Kroger, and J. Sander. Data bubbles: Quality preserving performance boosting for hierarchical clustering. In *ACM SIGMOD Conference*, pages 102–113, 2001.

[8] A.P. Dempster, N.M. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of The Royal Statistical Society*, 39(1):1–38, 1977.

[9] R. Dubes and A.K. Jain. *Clustering Methodologies in Exploratory Data Analysis*. Academic Press, New York, 1980.

[10] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. J. Wiley and Sons, New York, 1973.

[11] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison/Wesley, Redwood City, California, 3rd edition, 2000.

[12] F. Fanstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *SIGKDD Explorations*, 2(1):51–57, June 2000.

[13] B. Fritzke. The LBG-U method for vector quantization – an improvement over LBG inspired from neural networks. *Neural Processing Letters*, 5(1):35–45, 1997.

[14] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus-clustering categorical data using summaries. In *ACM KDD Conference*, pages 73–83, 1999.

[15] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *ACM SIGMOD Conference*, pages 73–84, 1998.

[16] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *ICDE Conference*, pages 512–521, 1999.

[17] G. Hammerly and C. Elkan. Alternatives to k-means clustering that find better solutions. In *ACM CIKM Conference*, pages 600–607, 2002.

[18] A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality. In *VLDB Conference*, pages 506–517, 1999.

[19] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[20] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[21] E. Knorr and R. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB Conference*, pages 211–222, 1999.

[22] G.J. MacLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, 1997.

[23] A. Mood, F. Graybill, and D. Boes. *Introduction to the Theory of Statistics*. McGraw Hill, NY, 1974.

[24] A. Nanopoulos, Y. Theodoridis, and Y. Manolopoulos. C2p: Clustering based on closest pairs. In *VLDB Conference*, pages 331–340, 2001.

[25] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. Technical report, Dept. of Statistics, University of Toronto, 1993.

[26] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *VLDB Conference*, pages 144–155, 1994.

[27] C. Ordonez and P. Cereghini. SQLEM: Fast clustering in SQL using the EM algorithm. In *Proc. ACM SIGMOD Conference*, pages 559–570, 2000.

[28] C. Ordonez and E. Omiecinski. FREM: Fast and robust EM clustering for large data sets. In *ACM CIKM Conference*, pages 590–599, 2002.

[29] C. Ordonez, E. Omiecinski, Norberto Ezquerra, J. Taboada, and D. Cooke. A fast algorithm to cluster high dimensional basket data. In *IEEE ICDM Conference*, pages 633–636, 2001.

[30] D. Pelleg and A. Moore. Accelerating exact K-means algorithms with geometric reasoning. In *ACM KDD Conference*, pages 277–281, 1999.

[31] S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11:305–345, 1999.

[32] M. Sato and S. Ishii. On-line EM algorithm for the normalized Gaussian network. *Neural Computation*, 12(2):407–432, 2000.

[33] B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, 4:279–299, 2001.

[34] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[35] L. Xu and M. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.

[36] A.L. Yuille, P. Stolorz, and J. Utans. Statistical physics, mixtures of distributions and the EM algorithm. *Neural Computation*, 6(1):334–340, 1994.

[37] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD Conference*, pages 103–114, 1996.

Author Biographies

Carlos Ordonez received his B.Sc. degree in Applied Mathematics and his MS degree in Computer Science both from the UNAM University, Mexico, in 1992 and 1996 respectively. He got a PhD degree in Computer Science from the Georgia Institute of Technology, USA, in 2000. Dr. Ordonez currently works for Teradata,

NCR, conducting research and providing consulting on data mining technology. He has published more than 10 research articles and holds 3 patents.

Edward Omiecinski received his PhD degree from Northwestern University in 1984. He is currently an Associate Professor at Georgia Tech in the College of Computing. He has published more than 60 papers in international journals and conferences dealing with database systems and data mining. His research has been funded by NSF, DARPA and NLM. He is a member of the ACM.