FREM: Fast and Robust EM Clustering for Large Data Sets

Carlos Ordonez
Teradata, NCR
Rancho Bernardo, CA, USA

Edward Omiecinski
Georgia Institute of Technology
Atlanta, GA, USA

ABSTRACT

Clustering is a fundamental data mining technique. This article presents an improved EM algorithm to cluster large data sets having high dimensionality, noise and zero variance problems. The algorithm incorporates improvements to increase the quality of solutions and speed. In general the algorithm can find a good clustering solution in 3 scans over the data set. Alternatively, it can be run until it converges. The algorithm has a few parameters that are easy to set and have defaults for most cases. The proposed algorithm is compared against the standard EM algorithm and the On-Line EM algorithm.

1. INTRODUCTION

Clustering is one of the most important data mining [12] techniques used nowadays. This problem has been extensively studied by the statistics [9, 29, 32], database [1, 5, 11, 16, 23, 36] and machine learning [10, 18, 30, 34] communities. Clustering algorithms partition a data set into several groups such that points in the same group are close to each other and points across groups are far from each other [10]. Most algorithms work with numeric data [3, 5, 14, 34, 36], but there is some recent work on clustering categorical data [13, 15, 17]. There has been extensive database research on clustering large data sets; some important approaches include [1, 3, 5, 7, 16, 21, 36]. The problem is challenging. High dimensionality [1, 2, 16, 27], data sparsity [1, 2, 14] and noise [3, 6, 7, 16] make clustering a harder problem. Finding optimal grid partitions for high dimensional data is introduced in [16]. Finding clusters on projections of high dimensional data has been the approach explored in [1, 2, 3]. Sampling and choosing representative points is proposed in [14].

1.1 The EM Algorithm

We present a fast and robust clustering algorithm for high dimensional and large data sets based on the well-known Expectation-Maximization (EM) algorithm [6, 8, 25, 30, 34, 35]. The EM algorithm is a general statistical method of maximum likelihood estimation [8, 34] and in particular it can be used to perform clustering.

The EM algorithm has many desirable features [34, 18]: a strong statistical basis, theoretical guarantees about optimality, easily explainable results, robustness to noise and to highly skewed data. Nevertheless, the classical EM algorithm [6, 34] also has several disadvantages. In general it is hard to initialize and the quality of the final solution depends on the quality of the initial solution. It may converge to a poor locally optimal solution. This means the solution may be acceptable but far from the optimal one. It needs an unknown number of iterations to converge to a good solution. It is hard to set a threshold on the maximum number of iterations that works in general. Therefore, it it is difficult to have guaranteed performance. For the reasons described above it usually requires many passes over the data. Two passes over the data set are required per iteration. Computations get unstable when variances approach zero. This problem commonly arises when clustering data sets with categorical attributes or when there is missing information. Probabilities vanish when points are outliers making computations undefined or unstable. It may produce inaccurate results with high dimensional data. This is related to the problem of computing distances in high dimensional space [1, 4]. There has been work on accelerating and improving EM, like incremental EM [22], On-line EM [31, 34], EM for high dimensional basket data [27] and Scalable EM [6], but no solution solves all problems listed above. A survey of other related approaches from the Machine Learning community can be found in [20].

http://doi.acm.org/10.1145/584792.584889

1.2 Contributions and article outline

This article presents an improved clustering algorithm that attempts to solve the problems described before. We introduce improvements to increase quality of solutions and speed. Initialization is based on a small perturbation of the global mean. This improves both speed and quality of solution. Multidimensional sufficient statistics and learning steps are used to accelerate convergence. This effectively reduces the number of iterations. Cluster splitting is introduced to find higher quality solutions. Regularization techniques for variance matrices are introduced to deal with zero variances. Distance is used instead of probability to manage outliers. The proposed algorithm is called FREM, which stands for Fast and Robust EM.

The contents of this article are as follows. Section 2 introduces definitions and a basic understanding of EM. Section 3 presents the FREM algorithm. Section 4 presents an experimental evaluation with real and synthetic data sets. FREM is compared against standard EM [8, 30] and On-Line EM [6, 34]. A review of related work is given in Section 5. The article concludes with Section 6, summarizing main research contributions and outlining future work.

Size	Contents
d	dimensionality
k	number of clusters
n	number of points

Table 1: Sizes

Matrix	size	contents
C	$d \times k$	means
R	$d \times k$	variances
W	$k \times 1$	weights

Table 2: Matrices

2. PRELIMINARIES

2.1 Definitions

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

$$P(y; C_j, R_j) = \frac{1}{\sqrt{(2\pi)^d |R_j|}} e^{-\frac{1}{2}(y-C_j)^t R_j^{-1} (y-C_j)}, \qquad (1)$$

where C_j is called the mean vector and R_j is called the covariance 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

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

EM uses Mahalanobis distance [10] instead of Euclidean distance. The basic difference between them is that the covariance matrix R_j is used to scale each dimension for distance computation. This is particularly useful for skewed data and dimensions having different scales. The Mahalanobis distance of point y_i to cluster j is:

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

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. C and R are $d \times k$ matrices, whereas W is a $k \times 1$ matrix. We use i, j and l as subscripts to access matrices entries. Matrix sizes, matrix names and subscripts to index them are summarized in tables 1, 2 and 3 respectively. 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

Index	range	used for
i	$1 \ldots n$	points
j	$1 \ldots k$	clusters
l	$1 \ldots d$	dimensions

Table 3: Subscripts

Input: $Y = \{y_1, y_2, \ldots, y_n\}$ and k.
Output: $\Theta = \{C, R, W\}$ and $L(\Theta)$

/* Initialization */
$\alpha \leftarrow (dk)^{-1}$, $I \leftarrow 1$, $\lambda \leftarrow 0.01$, $\omega \leftarrow 0.2/k$
FOR $j = 1$ TO k DO
  $C_j \leftarrow \mu \pm \alpha r$ diag$[\sigma]$, $R_j \leftarrow \Sigma$, $W_j \leftarrow 1/k$
ENDFOR

/* Iterate until convergence */
WHILE $|L(\Theta)^I - L(\Theta)^{I+1}| > \epsilon$ and $I \leq maxiter$ DO
  Estep()
  Mstep()
  $I \leftarrow I + 1$
ENDWHILE

Estep()
  $C' \leftarrow 0, R' \leftarrow 0, W' \leftarrow 0$
  $L(\Theta) \leftarrow 0$
  FOR $i = 1$ TO n DO
    $sump_i = 0$
    FOR $j = 1$ TO k DO
      $\delta_{ij} \leftarrow (y_i - C_j)^t R_j^{-1} (y_i - C_j)$
      $p_{ij} \leftarrow \frac{w_j}{(2\pi)^{d/2} |R_j|^{1/2}} exp[-\frac{1}{2}\delta_{ij}]$
      $sump_i \leftarrow sump_i + p_{ij}$
    ENDFOR
    $x_i \leftarrow p_i / sump_i$
    $L(\Theta) \leftarrow L(\Theta) + log(sump_i)/n$
    $C' \leftarrow C' + y_i x_i^t$, $W' \leftarrow W' + x_i$
  ENDFOR

Mstep()
  FOR $j = 1$ TO k DO
    $C_j \leftarrow C'_j / W'_j$
    FOR $i = 1$ TO n DO
      $R' \leftarrow R' + (y_i - C_j) x_{ij} (y_i - C_j)^t$
    ENDFOR
  ENDFOR
  $R = R'/n, W \leftarrow W'/n$

Figure 1: The EM clustering algorithm

the jth cluster centroid and R_j is its corresponding squared radius. Each R_j is a diagonal covariance matrix. 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 the solution is measured by a quantity called average log-likelihood, that is computed as

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

Pseudo-code for EM is shown on Figure 1. This is a brief overview. EM starts from an approximation to Θ. Intialization 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. Setting a maximum number of iterations is important to guarantee performance. 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 $L(\Theta)$ in each iteration and to converge to a locally optimal solution [8, 34].

3. IMPROVING THE EM ALGORITHM

This section describes in detail our novel clustering algorithm for large data sets. First, improvements are introduced. Second, a detailed version of the algorithm is presented. Third, time complexity is briefly analyzed. More details about FREM can be found in [24].

3.1 Improvements

We propose several improvements that we present in two groups: improvements for speed and improvements to increase the quality of solutions. These groups are orthogonal in the sense that they can be used independently if needed. Speed improvements include sufficient statistics and learning steps. The improvements for quality are tunable initialization, cluster splitting and probability computation for outliers.

Improvements for speed.

Sufficient statistics [5, 22, 36], are summaries of groups of points (in this case clusters), represented by $Y_1, \ldots, Y_k$. They were originally introduced for a one dimensional clustering problem in [22]. Here we present their multidimensional version assuming dimensions are independent and taking into account that they induce a partition on Y. Sufficient statistics are stored in matrices M, Q and N (M and Q are $d \times k$ and N is $k \times 1$). M stores sum of points, Q has sum of squared points, and N counts the number of points per cluster :

$$M_j = \sum_{\forall y_i \in Y_j} y_i, \quad (5)$$

$$Q_j = \sum_{\forall y_i \in Y_j} y_i y_i^t, \quad (6)$$

$$N_j = |D_j|. \quad (7)$$

Sufficient statistics reduce I/O time by avoiding repeated scans over the data and by allowing parameter estimation periodically as transactions are being read. EM for mixture of Gaussians cannot work when variances are zero because several computations become undefined (e.g. $|R_j|, \delta_{ij}$). This basically means that all points belonging to one or more clusters have the same value on some dimension. We solved this problem by adding to each dimension a constant $\lambda \geq 0$ multiplied by the global variance when variances are updated. This constant is chosen small enough to avoid altering the real variance values. This will be done when the M step updates R from sufficient statistics for FREM. Then the update formulas for based on sufficient statistics used by FREM are:

$$C_j = \frac{1}{N_j} M_j, \quad (8)$$

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

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

Learning steps (periodic M steps) are used to accelerate convergence. Remember that cluster membership is determined in the E step and C, R, W are updated in the M step. By using sufficient statistics the algorithm can run the M step at different times while scanning Y. At one extreme we could have an on-line version [34, 5] that runs the M step after every point. That would be a bad choice. It would be very sensitive to the order of points and more importantly, it would not reach the global optimal solution [5, 34] (although it could get close). 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 and no performance improvement would be achieved. However, it must be noted that this latter version would not be sensitive to the order of points, would compute the correct value for $L(\Theta)$ and would have the potential of reaching the global optimal solution while the on-line version would not. 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. This number of times will be related to k and d, but not to n. When a good approximation to the solution has been reached then we can run normal EM iterations to converge. On-line EM [6, 31, 34] represents the *fastest* version that can be derived from EM, but without guarantees about optimality or stability of the solution. On-line EM executes the M step after every E step for each point. So it only makes one scan over the data set. However, it is very sensitive to the order of points, rarely finds a close to optimal solution and does not provide an accurate $L(\Theta)$ computation. This will be shown in the experimental section.

The M step is periodically executed L times on the first two iterations and only once in subsequent iterations.

$$L = dk$$

if $n \geq (dk)^2$, and

$$L = k$$

otherwise. $L = dk$ is used for large data sets where FREM can exploit redundancy to converge faster. On the other hand, $L = k$ is used for small data sets where FREM may be more sensitive to the order of points. The reason behind this setting for L is that the M step should be run at least k times per scan to give the algorithm the ability to read k portions of Y, split k times if necessary and make it less sensitive to the order of points. On the other hand $L = dk$ times accelerate convergence when n is large. L resembles number of iterations. The E step must be run for every point, i.e. n times per scan. FREM makes at least three iterations. The first iteration gets a decent solution, the second one tunes it and further iterations make FREM converge.

Improvements to increase quality of solution.

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. Initialization is then done as follows.

$$W_j = \frac{1}{k},$$

$$R_j = \Sigma,$$

$$C_j = \mu \pm \alpha\sigma r,$$

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 grows C_j 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. So what we are actually doing is getting centroid seeds that are small perturbations of μ.

Besides speed of convergence EM is often criticized for finding a sub-optimal solution [6, 30], a common problem with clustering algorithms [9, 10]. Most of the times suboptimality involves several clusters grouped together as one while other clusters have almost no points. So we propose splitting clusters to reach higher quality results. We introduce a minimum weight threshold ω to control splitting. Let a be the index of the weight of a cluster s.t. $W_a < \omega$. 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 (the right terms are precisely one standard deviation). This process gets repeated until there are no more clusters below ω. The typical value ω will be $0.2/k$; that is, clusters whose weight is lower than 20% the average cluster weight are re-seeded. Cluster splitting will be done in the M step after C, R, W are updated on the first iteration.

Outliers are points that cannot be fitted adequately by the model. These points are far from any cluster and then $P(y_i; \Theta) = 0$; that is y_i has probability zero of belonging to any cluster. Basically, we use the closest cluster to update sufficient statistics. It is important to note that the algorithm 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 the algorithm has a lower threshold ϕ for probabilities. If $P(y_i; \Theta) \geq \phi$ then

$$x_{ij} = \frac{P(y_i|j)}{P(y_i; \Theta)},$$

otherwise,

$$x_{ij} = \frac{1}{\delta_{ij} \sum_{j'=1}^{k} \frac{1}{\delta_{ij'}}}.$$

3.2 The FREM algorithm

Now all improvements are put together in one place. The FREM algorithm pseudo-code is shown in Figure 2. 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 matrices $\Theta = \{C, R, W\}$ and $L(\Theta)$ as defined in section 2. x_{ij} represents the weighted probability that point y_i belongs to cluster j (soft partition). FREM's iterations can be summarized as follows: 1: Accelerate convergence and increase quality of solution splitting clusters, 2: Accelerate convergence, and 3,4,.., and further iterations: Remove sensitivity to the order of points, compute an accurate value of $L(\Theta)$ and converge.

The first two iterations have the goal of reaching a very 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. Therefore, further iterations are needed. The 3rd, 4th and subsequent iterations represent a normal execution of EM based on sufficient statistics. They are not sensitive to the order of points because the M step is executed after all n points are read. Their $L(\Theta)$ computation is accurate (Equation 4). Therefore, FREM monotonically increases $L(\Theta)$ and converges.

The FREM algorithm has complexity $O(kdn)$ 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 (Eq. 5, 6, 7). The algorithm reads n points. For each point k probabilities are computed and each of these requires d computations, one per dimension. So each point requires $O(dk)$ work and being n points the total complexity is $O(dkn)$. Note that sufficient statistics are updated on only one column per point so this operation has complexity $O(p)$ per E step. The mixture parameters C, R, W are updated taking $O(dk)$ each in the M step (Eq. 8, 9, 10). The M step is executed L times on the first two iterations and only once on the 3rd and further iterations. FREM operations are summarized in table 4. The table shows their complexity and how frequently they are executed.

4. EXPERIMENTAL EVALUATION

To evaluate quality of results and performance we tested our algorithm with real and synthetic data sets. FREM is compared against a standard version of EM and against Online EM. All algorithms are initialized in the same manner as explained in Section 3. FREM and EM are stopped until they converge with the same tolerance. Since μ, Σ are computed only once per data set that time is not included in the total time. The algorithm was implemented in C++. Data sets were stored as plain text files. All experiments were run on a Sun computer running at 800 MHz, having several Gb of disk space and 128Mb on main memory. The implementation used text files as input. Some improvement could be expected if binary files were used since no parsing would have to be done. Another observation is that

Operation	Complexity	Frequency
Compute μ, Σ	$O(dn)$	once per data set
Iteration	$O(dkn)$	three times or more per run
E step	$O(dk)$	n times per iteration
δ computation	$O(dk)$	once per E step
$p(y_i\|\Theta)$ computation	$O(k)$	once per E step
Compute x_i	$O(k)$	once per E step
Update M, Q	$O(d)$	once per E step
Update N	$O(k)$	once per E step
Compute $L(\Theta)$	$O(1)$	once per E step
M step	$O(dk)$	L times per iteration
Cluster splitting	$O(dk)$	L times; on each M step on first iteration
Update C, R, W	$O(dk)$	once per M step
Compute each $\|R_j\|$	$O(d)$	k times per M step and at initialization

Table 4: A summary of FREM operations

```
Input: Y = {y_1, y_2, ..., y_n} and k.
Output: Θ = {C, R, W} and L(Θ)

/* Initialization */
α ← (dk)^{-1}, I ← 1, λ ← 0.01, ω ← 0.2/k
IF n < (dk)^2 THEN L ← k ELSE L ← dk ENDIF
FOR j = 1 TO k DO
   C_j ← μ ± αr diag[σ],  R_j ← Σ,  W_j ← 1/k
ENDFOR

/* Iterate until convergence */
WHILE |L(Θ)^I − L(Θ)^{I+1}| > ε or I ≤ 3 DO
   L(Θ) ← 0
   M_j ← C_j,  Q_j = C_j C_j^t,  N_j ← 1
   FOR i = 1 TO n DO
      Estep()
      IF( i mod (n/L) = 0 and I ≤ 2 ) THEN
        Mstep()
        IF I=1 THEN splitClusters() ENDIF
      ENDIF
   ENDFOR
   Mstep()
   I ← I + 1
ENDWHILE

Estep()
  FOR j = 1 TO k DO
     δ_ij ← δ(y_i, C_j, R_j),
     p_ij ← ((2π)^d |R_j|)^{-1/2} exp(−δ_ij/2)
     x_ij ← p_ij / (Σ_{j'} p_ij')
  ENDFOR
  Let m be s.t. p_im ≥ p_ij, ∀j ∈ 1...k
  M_m ← M_m + y_i,  Q_m ← Q_m + y_i y_i^t,  N_m ← N_m + 1
  L(Θ) ← L(Θ) + log(Σ p_ij)/n

Mstep()
  FOR j = 1 TO k DO
     C_j ← M_j / N_j
     R_j ← Q_j/N_j − M_j M_j^t / N_j^2 + λΣ
     W_j ← N_j / Σ_{J=1}^k N_J
     |R_j| = Π_{l=1}^d R_lj
  ENDFOR
```

Figure 2: The FREM clustering algorithm

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 is easier and accessing by subscripts is faster. In short, these performance experiments were performed under pessimistic conditions without making optimizations that would affect usability.

Default Parameter Settings.

FREM has a few parameters but they have defaults to make usage easier. EM and FREM used $\epsilon = 10^{-5}$ to stop. This setting assured both algorithms converge to a stable solution. Also, in order to cluster the real data sets the regularization constant λ was required because of zero variances; $\lambda = 1.0e-2$ to introduce a negligible change in the real variance values. The minimum probability for outliers was fixed $\phi = 1.0e-200$. The standard deviation factor for seeding was set $\alpha = 1/(dk)$; this is a number that tends to work well for fairly high dimensional data. The minimum cluster weight is $\omega = 0.2/k$; this proved useful in all cases. Clusters whose weight fall below this threshold will be re-seeded in learning steps on the first iteration. In general only k is changed. In the following experiments it will clarified which values get a different value from their default.

4.1 Experiments with Real Data Sets

In this section we evaluate quality of results with real data sets from different domains. *Astronomy* is a data set containing information about stars from an observatory. The dimensions of stars include position 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 contains a sample of aggregated information of transactions from a large data warehouse of a European retailer. The dimensions we use are total sales amount, total cost, number of items bought, number of distinct articles and number of distinct departments. For this data set $n = 100,000$ and $d = 5$. This data set presented zero variance problems in a few clusters. The *medical* data set contains information for a set of $n = 655$ being treated for heart disease. The numeric dimensions include 4 artery disease (blockage) percentages and 9 numbers describing per-

Data set	FREM 3 scans	FREM converge $L(\Theta)$	OnLine EM	EM	FREM 3 scans	FREM converge Time(secs)	OnLine EM	EM
Astronomy $k=5$	-22.25	-22.06	-23.41	-22.06	91	452	48	1598
Astronomy $k=10$	-22.01	-21.99	-23.33	-22.93	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

Table 5: Quality of results and performance with real data

fusion measurements corresponding to 9 specific regions of the heart; in 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. The three algorithms were run 10 times with $k = 5$ and $k = 10$. We show measurements for the run that produced best results in each case.

Results for the best run out of 10 are summarized in table 5. Times are given in seconds and are rounded. Since the globally optimal solution cannot be known we can only use $L(\Theta)$ to compare quality of results. The left part has quality measurements ($L(\Theta)$) and the right part has time measurements in seconds. For $L(\Theta)$ the closer 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. For the basket and astronomy data sets FREM-converge produced better results than EM; for the astronomy data set they found the same solution with $k = 5$. EM produced better results with $k = 10$ for the medical set but not for $k = 5$; a closer inspection on these results on the medical data set revealed the clusters had significant overlap and several variances in R_j were almost zero. Another problem is this data set is very small and therefore FREM cannot take advantage of redundancy. In short, FREM-converge found better solutions than EM in all but one case and FREM-3scans was always close to FREM-converge. In the cases where EM found a better solution FREM-3scans was very close. FREM-3scans was always very close to FREM-converge, showing it is reasonable to stop the algorithm after the 3rd iteration before it has converged. On-line EM always found the worst quality solution. Regarding performance FREM-3scans was always one order of magnitude faster than EM. FREM-converge was slightly slower, but always exhibiting better performance than EM. In general the number of iterations required by FREM was about one third of those required by EM. These results show that it may be reasonable to stop FREM after the 3rd iteration. On-line EM was clearly the fastest; a 2nd scan was needed to compute an accurate value for $L(\Theta)$, but that time is not included in the table.

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.

A synthetic data generator

Creating data sets that mimic real data sets is difficult. One of the main complaints about synthetic data generators is that they create clean data, well behaved, easy to mine and with low dimensionality. But real data sets have missing information, hard to identify distributions, varied scales, rich data types, high dimensionality and noise. With this in mind the following variables were identified as interesting parameters to create synthetic data.

First, n, d and k should clearly be parameters to measure performance at the very least. So the generator needs to specify how many points are needed (n) and how many dimensions each point has (d). There must a number of Gaussian distributions that will be the total number of clusters k. Recall that any data set can be approximated by a mixture of Gaussians [32]. It is the parameters of the Gaussians that will make data more realistic.

The reference normal distribution it $N(0, 1)$ meaning $\mu = 0$ and $\sigma = 1$. Each dimension will be independently normally distributed varying μ and σ. The value for μ will be taken as a random value uniformly distributed in the range $[0, 1]$. This value will be multiplied by a different scale factor β in each dimension or alternatively by the same one making the entire space look like a hypercube. This will be controlled by a parameter called UC which stand for uniform C. Then $\sigma > 0$. Its value will be scaled according to how much overlap is needed among clusters. This overlap varies depending on both dimensionality and how many clusters there are in the data. A general setting that makes the problem challenging but at the same time solvable is setting $\beta = \sigma^2$ regardless of the number of clusters. Setting $\sigma^2 > \beta$ dramatically increases overlap among clusters. Clusters may have non uniform densities (# points per volume unit). Informally, it can be decided if clusters will have the same spatial size or not. If the spatial size is the same that means clusters have about the same scatter. If clusters do not have the same spatial size then some clusters may look tighter than others. In other words, the density per volume unit may be different. This will controlled by a parameter called UR which stands for uniform R. Then there must be additional points that do not really form a cluster. These points should obscure cluster boundaries. If the points had a non-uniform distribution then by definition would form certain subsets of points that could be effectively identified as clusters. So a certain percentage of points are added. These points are uniformly distributed all over the full dimensional space. This noise percentage is called η, taking a value between 0 and 100%. Then the weights will satisfy $\sum_{j=1}^{k} W_j = 1 - \eta$. At one extreme if $\eta = 0$ then all the

Parameter	possible values	meaning
d	$[1, 2, \ldots, 200]$	dimensionality
k	$[1, 2, \ldots, 200]$	number of embedded clusters
n	$n \geq k$	size of data set
η	$[0, 100)$	noise %, $1 - \eta$ is reserved for data
β	$\beta > 0$	scale to shrink or enlarge μ
σ	$\sigma > 0$	standard deviation from μ, independent from β
UW	Yes/No	Uniform weights desired? (W)?, Default=Yes
UC	Yes/No	Uniform scale for C?, Default=Yes
UR	Yes/No	Uniform spatial sizes for R? Default=Yes

Table 6: Parameters for synthetic data generation

clusters take up the entire space and if $\eta = 100\%$ then there are no clusters at all. It can be expected that $\eta > 50\%$ will blur cluster boundaries significantly. The user can specify if clusters will have uniform weights or not; if not the weights are randomly chosen and normalized. Note that clusters can have non-uniform weights and non-uniform spatial sizes (or densities). All these parameters are summarized in table 6.

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 typical data mining environment. These 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 ϵ be the estimation error. Then cluster j is considered accurate if

$$\frac{1}{d}\sum_{l=1}^{d}\frac{|c_{lj} - C_{lj}|}{|c_{jl}|} \leq e$$

and

$$\frac{|w_j - W_j|}{w_j} \leq e.$$

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 ϵ it will be considered accurate. Any unmatched estimated cluster is considered inaccurate.

Figures 3 and 4 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 3 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 4 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.

Performance

Performance varying size, dimensionality and desired number of clusters is shown in Figure 5. 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.

Limitations.

Our algorithm is not the final answer to clustering. Its main limitations are the following. Cluster splitting improves quality of solutions but it depends on an adequate ω. The setting for α slightly affects the quality of the solution; we found a wide range that went from k to $10dk$ and we decided to set it at dk by default. λ changes the cluster-

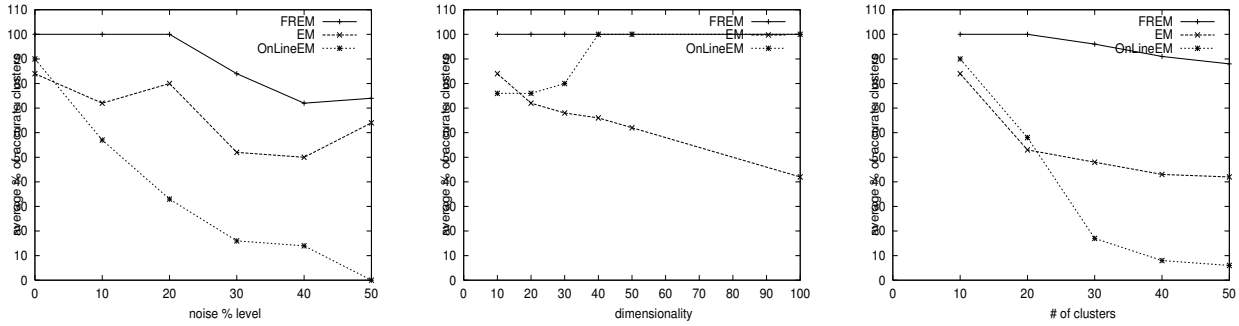

Figure 3: Quality of results: average percentage of accurate clusters (average run)

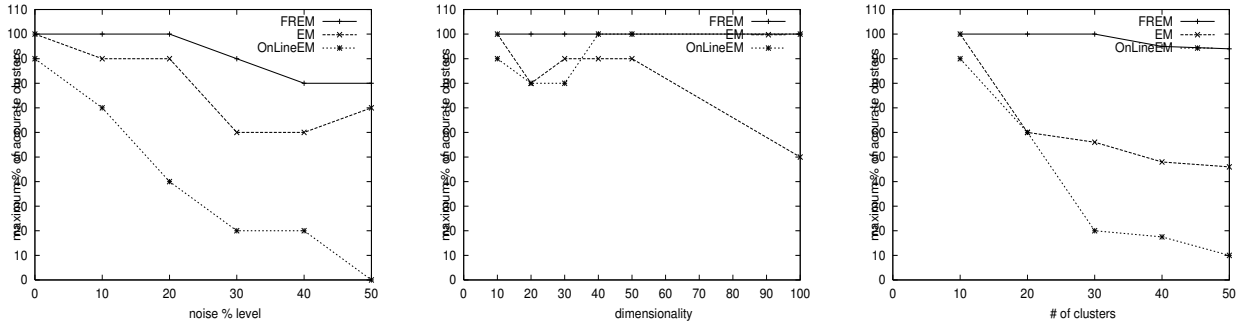

Figure 4: Quality of results: maximum percentage of accurate clusters (best run)

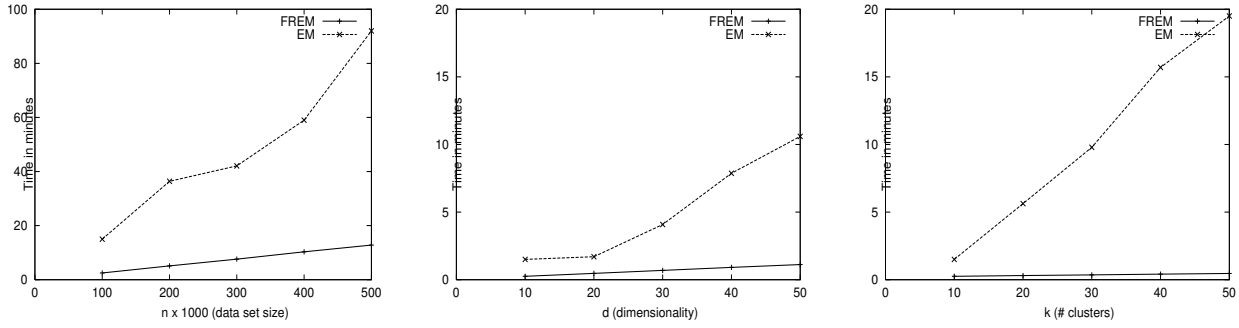

Figure 5: Performance varying size, dimensionality and number of clusters

ing results by a marginal fraction since it introduces a small variation in actual variance values; specially for overlapping clusters. But the gain is huge since zero variances are not a hinderance. Computing probabilities in high dimensionality is troublesome and given our outlier handling approach in many cases we assign points to the closest cluster like K-means. For very large data sets two iterations or even one could be enough to find an acceptable solution. This would involve doing more careful cluster splitting and increasing the number of learning steps. However, time would go down to one half (or one third), not a significant improvement given faster CPU speeds every year. Also, an additional pass would be mandatory to compute an accurate value for $L(\Theta)$. Due to lack of space we do not include experiments justifying these claims.

5. RELATED WORK

There exist many scalable clustering algorithms. Some well-known examples include OPTI-GRID [16], CLIQUE [3], CURE [14], PROCLUS [1], DBSCAN [23]. There is also work on accelerating K-means [5, 28]. 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 to improve EM to work with large data sets is not new. The most important previous work comes from the Machine Learning community. Neal and Hinton introduced the idea of using sufficient statistics in [22]. 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 [5, 6]. A scalable K-means is presented in [5] and this algorithm is used as a framework to build a scalable EM algorithm in [6]. In [6] they present a scalable EM algorithm (SEM) that iterates 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 value for this parameter is typically 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. They use K-means to cluster points in main memory whereas we use EM itself. Our re-seeding strategy is different. 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 3 or more 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 [11]. Their $L(\Theta)$ computation is not accurate 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. They assume SEM converges based on K-means convergence on compressed point sets. In [11] 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 standard EM and On-line EM was more appropriate.

The idea of helping EM clustering deal with zero variances has been studied before in [33]. 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 Σ so that distance computation is not affected for dimensions with different scales.

6. CONCLUSIONS

This article proposed an efficient and robust variant of the EM clustering algorithm. This novel algorithm is called FREM. FREM incorporates several improvements to find higher quality solutions and converge faster than EM. The improvements include mean and dimensionality based initialization, learning steps based on sufficient statistics, regularization techniques, cluster splitting and alternative probability computation for outliers. Most parameters have defaults. In general k, the desired number of clusters, is the only parameter to change. Experimental evaluation shows the effectiveness and speed of the algorithm with both synthetic and real data sets. FREM can produce a good solution in just 3 iterations or it can be run until it converges like EM. In either case FREM's performance is better than EM's and the quality of solutions is at least as good as those found by EM. Running FREM with three iterations is about three times slower than On-line EM, but always finds higher quality solutions.

Future work includes the following. We want to use FREM as a foundation to perform Data Mining based on statistical models [24]. We want to relate clustering and association rules [26]. Clustering data with categorical attributes [15], finding outliers [19], identifying clusters that exist only in subspaces but not on the entire space [3], or finding clusters in projections of high dimensional data [1] are also interesting problems for further research.

Acknowledgments

We thank Emory University for providing the medical data set and the California Institute of Technology for providing the astronomy data set. Special thanks to the anonymous referees for their helpful comments.

7. REFERENCES

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

[2] C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *ACM SIGMOD Conference*, pages 40–51, 2001.

[3] 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.

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

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

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

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

[12] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.

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

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

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

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

[17] 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.

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

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

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

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

[22] 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.

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

[24] C. Ordonez. *Mining Complex Databases Using the EM Algorithm, PhD Thesis*. Georgia Institute of Technology, Atlanta, 2000.

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

[26] C. Ordonez, E. Omiecinski, Levien de Braal, Cesar Santana, and N. Ezquerra. Mining constrained association rules to predict heart disease. In *IEEE ICDM Conference*, pages 433–440, 2001.

[27] 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.

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

[29] R.A. Redner and H.F. Walker. Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, 26:195–239, 1984.

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

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

[32] D. Scott. *Multivariate Density Estimation*. J. Wiley and Sons, New York, 1992.

[33] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton. SMEM algorithm for mixture models. In *Neural Information Processing Systems*, 1998.

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

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

[36] 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.