Scaling EM (Expectation-Maximization) Clustering to Large Databases

Paul S. Bradley Usama M. Fayyad Cory A. Reina

Microsoft Research

November 1998
Revised February 1999

Technical Report

MSR-TR-98-35

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Scaling EM (Expectation-Maximization) Clustering to Large Databases

P. S. Bradley **Usama Fayyad** **Cory Reina**

bradley@microsoft.com fayyad@microsoft.com coryr@microsoft.com

Microsoft Research

Redmond, WA 98052, USA

Abstract:

Practical statistical data clustering algorithms require multiple data scans to converge. For large databases, these scans become prohibitively expensive. We present a scalable clustering framework requiring at most one scan of the database, and apply it to the Expectation-Maximization (EM) algorithm. Unlike distance-based or hard membership algorithms (such as *k*-Means) EM is known to be an appropriate optimization algorithm for constructing proper statistical models of the data. It also easily accommodates categorical and continuous data fields. It admits varying degrees of data membership in multiple clusters. Our scalable method is based on identifying regions of the data that are compressible and regions that must be maintained in memory. The approach operates within the confines of a limited memory buffer. Data resolution is preserved to the extent possible based upon the size of the memory buffer and the fit of the current clustering model to the data. We extend the framework to update multiple clustering models simultaneously. Computational tests indicate that this scalable scheme outperforms sampling-based and incremental approaches -- the straightforward alternatives to 'scaling' existing traditional in-memory implementations to large databases.

Keywords: Clustering, Probabilistic modeling, Density estimation, scalable mining, EM algorithm, iterative refinement, clustering large databases.

1 Introduction

Data clustering is important in many fields, including data mining [FPSU96], statistical data analysis [KR89,BR93], compression [ZRL97], and vector quantization. Applications include data analysis and modeling [FDW97,FHS96], image segmentation, marketing, fraud detection, predictive modeling, data summarization, and general data reporting tasks [B*96]. It has important applications in data cleaning and exploratory data analysis. Clustering is a crucial data mining step and performing the task over massive databases is essential. Previous scalable clustering work has focused on *k*-Means-type approaches [ZRL97,BFR98] and region growing [NH94, SEKX98, AGGR98]. These techniques, while effective, do not derive statistical models of the data (i.e. they are based on notions of distance metrics, etc.) and they do not allow for cluster overlap (i.e. a data record may belong to different clusters with different membership probabilities). In this paper, we focus on the task of scaling the most effective technique available for proper probabilistic clustering: the Expectation-Maximization (EM) algorithm [DLR77, CS96]. EM has additional desirable properties in that it does not require the specification of distance measures and readily admits categorical and continuous attributes (which is untrue of other clustering algorithms that either operate on continuous, e.g. *k*-Means-type algorithms, or categorical [GKR98] data exclusively). EM has been shown to be superior to other alternatives for statistical modeling purposes [GMPS97,PE96,B95,CS96,NH99].

The clustering problem has been formulated in various ways in the statistics [KR89,BR93,B95,S92,S86], pattern recognition [DH73,F90], optimization [BMS97,SI84], and machine learning literature [F87]. The fundamental problem is that of grouping together (clustering) data items that are similar to each other. The most general view places clustering in the framework of density estimation [S86, S92, BR93]. Data is generally not uniformly distributed. Some combinations of attribute values are more likely than others. Clustering can be viewed as identifying the dense regions of the probability density of the data source. An effective representation of the probability density function is the *mixture model*: a model consisting of several components (e.g. a model consisting of the sum of 3 Gaussians). Each component generates a set of data records (a "cluster"). The data set is then a mixture of clusters and the problem is to identify the data points constituting a cluster and inferring the properties of the distribution governing each cluster.

Consider a simple example with data consisting of 2 attributes: age and income. One may choose to model the data as a single cluster and report that the data records have an *average age* of 41 years and an *average income* of \$26K/year (with associated variances). However, this is rather deceptive and uninformative. The data may be a mixture of working people, retired people, and children. A more informative summary is to identify these subsets or clusters, and report the cluster parameters. Results may now be: 20% of the data have average age 12 and zero income, 45% have average age 38 and average income \$45K, and 30% have average age 72 and average income \$20K, while 5% of data had unknown incomes and ages. Assuming we have no apriori definition of the various sub-populations, how do we discover what they are? Furthermore, how do we do this if the data has many more dimensions and the various partitions are not obvious? This is where clustering plays an important role in identifying these dense regions in multidimensional data.

Density estimation via the mixture model framework can be seen as a generalization of the clustering problem [BB95, B95]. The EM (Expectation-Maximization) algorithm [DLR77, CS96] is an effective, popular technique for estimating mixture model parameters (cluster parameters and their mixture weights). The EM algorithm iteratively refines initial mixture model parameter estimates to better fit the data and terminates at a locally optimal solution.

Other similar iterative refinement clustering methods include the popular *k*-Means-type algorithms [M67,DH73,F90,BMS97,SI84]. While these approaches have received attention in the database and data mining literature [NH94,ZRL97,BFR98], they are limited in their ability to model data from a statistical perspective. The *k*-Means algorithm attempts to minimize the sum of squares of Euclidean distances between data records in a cluster and the cluster's mean vector. This assignment criterion implicitly assumes that all clusters are represented by identical spherical Gaussian distributions located at different means [BB95, B95]. In addition, since the *k*-Means algorithm is married to the Euclidean metric, it does not generalize to the problem of clustering discrete or categorical data. The *k*-Means algorithm also assumes each data record belongs to exactly one cluster. In reality, a data record may belong to multiple clusters with different probabilities of membership. The mixture model framework is more general and relaxes all these assumptions. In addition, one may choose a probability distribution over categorical attributes (e.g. the Multinomial distribution) and naturally cluster this data type.

Goals of this Work: We focus on scaling EM to compute mixture model parameter values over large databases. For large databases, hundreds of iterations or more may be required during iterative refinement clustering. Although guaranteed to converge finitely, a general bound on the number of iterations required for EM is not available. We assume a single database scan is expensive, thus computing a mixture model over large databases via standard EM would not be acceptable. We present a scalable version of the EM algorithm that satisfies the following *Data Mining Desiderata:*

1. **One scan:** The algorithm requires at most one database scan with early termination highly desirable.
2. **Anytime algorithm:** The algorithm is always able to provide a "best" answer at anytime during its computation (i.e. it exhibits "online, anytime" behavior).
3. **Interruptible and Incremental:** The algorithm is suspendable, stoppable and *resumable*. Incremental progress can be saved for continued computation later, possibly on new data.
4. **Limited RAM Requirement:** The algorithm works within the confines of a limited memory (RAM) buffer, allocated by the user, insuring good behavior when run as a server process.
5. **Forward-only cursor:** The algorithm has the ability to operate with a forward-only cursor over a view of the database.

A forward only-cursor respects the fact that the individual data records provided to the clustering algorithm may be the result of an expensive join query over a potentially distributed data warehouse. Hence a second scan may cause a second expensive join and should be avoided if possible.

Contributions of this Paper: A scalable EM algorithm is introduced that satisfies the goals stated above. The problem of computing a mixture model is decomposed so that it is possible to cluster arbitrarily large databases while utilizing small amounts of memory. The fundamental observation is that all data is not of equal importance when computing the mixture model. Data records can be classified into one of three categories: records that can be safely discarded, records that may be compressed, and records that need to be retained in memory. We demonstrate that the scalable EM algorithm (SEM) indeed preserves clustering fidelity and that it outperforms traditional ways for coping with large databases involving data sampling. The framework we present can accommodate many iterative refinement type algorithms including *k*-Means [BFR98], and also supports clustering over data with mixed continuous-discrete attributes. The localized data access properties of SEM have additional benefits relating to improved utilization of fast caches and on-chip memory caches in modern CPU's. This results in faster execution times even on machines that have enough resident memory to load the entire database.

2 Preliminaries

While the framework we present is general, we focus on its application to distributions that are mixtures of multivariate Gaussians. This choice is motivated by a result from density estimation theory stating that any distribution can be effectively approximated by a mixture of Gaussians [S92,S86]. Each population (cluster) is modeled by a *d*-dimensional Gaussian probability distribution. The multi-dimensional Gaussian distribution for cluster ℓ, $\ell = 1,\ldots,k$, is parameterized by the *d*-dimensional mean vector μ^ℓ and $d \times d$ covariance matrix Σ^ℓ : $\Pr(\boldsymbol{x} \mid \ell) = \dfrac{1}{\sqrt{(2\pi)^d \left|\Sigma^\ell\right|}} exp\left(\dfrac{1}{2}\left(\boldsymbol{x}-\mu^\ell\right)^T \left(\Sigma^\ell\right)^{-1} \left(\boldsymbol{x}-\mu^\ell\right)\right)$. **(1)**

$\boldsymbol{x}$ and μ^ℓ are column vectors, the superscript ^{T} indicates transpose to a row vector, $|\Sigma^\ell|$ is the determinant of Σ^ℓ and $(\Sigma^\ell)^{-1}$ is its matrix inverse. The *mixture model* probability density function is:

$$\Pr(\mathbf{x}) = \sum_{\ell=1}^{k} W_\ell \Pr(\mathbf{x} \mid \ell). \quad \textbf{(2)}$$

The coefficients W_ℓ (mixture weights) represent the fraction of the database represented by the corresponding cluster. A given data record $\boldsymbol{x}$ is a member of each of the *k* clusters with different probabilities of membership. Probability in population ℓ is given by: $\Pr(\ell \mid \boldsymbol{x}) = \dfrac{W_\ell \Pr(\boldsymbol{x} \mid \ell)}{\Pr(\boldsymbol{x})}$. **(3)**

EM computes locally optimal maximum likelihood model parameter values [DLR77, CS96].

2.1 Classic (Vanilla) EM Algorithm

The classic or "vanilla" EM algorithm requires initial mixture model parameter estimates as input. Given mixture model parameters, a single EM iteration provides new parameter estimates which are proven not to decrease the log likelihood of the model [DLR77,B95]. The process is repeated until the log likelihood of the mixture model at the previous iteration is sufficiently close to the log likelihood of the current model. The algorithm proceeds as follows for our Gaussian mixture model:

0. Given initial mixture model parameter values: initialize the mixture model parameters, set current iteration $j = 0$: W_ℓ^0, $\mu^{0,\ell}$ and $\Sigma^{0,\ell}$, $\ell = 1,\ldots,k$.

1. Having mixture model parameters at iteration j, update them as follows:
For each database record $x \in D$:
Compute the membership probability of $\boldsymbol{x}$ in each cluster:

$$\Pr(\ell \mid \boldsymbol{x}) = \frac{W_\ell^j \Pr^j(\boldsymbol{x} \mid \ell)}{\Pr^j(\boldsymbol{x})}, \ell = 1,\ldots,k.$$

2. Update mixture model parameters: $W_\ell^{j+1} = \dfrac{1}{N}\sum_{x \in D} \Pr(\ell \mid \boldsymbol{x})$

$$\mu^{j+1,\ell} = \frac{\sum_{\mathbf{x}\in D} \mathbf{x} \cdot \Pr(\ell \mid \mathbf{x})}{\sum_{\mathbf{x}\in D} \Pr(\ell \mid \mathbf{x})},\ \Sigma^{j,\ell} = \frac{\sum_{\mathbf{x}\in D} \Pr(\ell \mid \mathbf{x})\left(\mathbf{x} - \mu^{j+1,\ell}\right)\left(\mathbf{x} - \mu^{j+1,\ell}\right)^T}{\sum_{\mathbf{x}\in D} \Pr(\ell \mid \mathbf{x})},\ \ \ell = 1,\ldots,k.$$

3. If $\left|E^j - E^{j+1}\right| \le \varepsilon$, stop. Else set $j = j+1$ and go to 1. E^j is the log likelihood of the mixture model at iteration j: $E^j = \sum_{\mathbf{x}\in D} \log\left(\Pr^j(\mathbf{x})\right) = \sum_{\mathbf{x}\in D} \log\left(\sum_{\ell=1}^{k} W_\ell^j \cdot \Pr^j(\mathbf{x} \mid \ell)\right)$. **(4)**

3 Model Estimation over Large Databases

When clustering data, we implicitly assume that data records within the given database are not uniformly distributed. By definition clustering assumes that there exist data regions that are more dense than other regions. Our goal is to exploit this fact and design an algorithm that computes an approximate solution (empirically, a very good solution) while operating within a limited memory (RAM) buffer in one scan or less of the database. The basic scalable EM approach is summarized at a high level as follows:

0. Initialize the mixture model parameters to given values (set by user or at random).

1. (Fill Buffer). Obtain a sample from the database, filling the memory buffer.

2. (Update Mixture Model Parameters). Apply EM to update the mixture model parameters over the data contents in the buffer (including previously collected summary statistics (step 3)) producing new mixture model parameters (see Section 3.2).

3. (Identify Data to be Compressed). Given the updated mixture model, identify data regions that may be summarized, tag data in the buffer lying in these regions (see Sections 3.3).

4. (Compress Data to Sufficient Statistics). Compute sufficient for data tagged in 3. Purge these data records from buffer, retaining only their sufficient statistics (see Section 3.1).

5. (Stopping Criteria). If stopping criteria (see Section 3.4) are not satisfied return to step 1.

The basic intuition is to identify data subsets whose contribution to model updates can be effectively summarized by their sufficient statistics. Instead of revisiting these records, updates are performed over their sufficient statistics. Effectively, in the single pass over the database SEM gathers a collection of data records and summaries (compressed versions of records). The contents of the limited memory buffer are intended to be as representative as possible of the entire database. After each buffer refill, the mixture model parameters are updated over the new sample of data records, the set of data records retained in the buffer, and the compressed (summarized) representations of data discarded in past iterations.

Note that this scalable model estimation procedure satisfies the requirements set forth in the data mining desiderata. Clearly the algorithm works within a limited memory buffer by definition and only utilizes a forward-only cursor. The algorithm requires one scan (or less) of the database. Early stopping occurs when the log likelihood of the model between successive iterations is stable. The algorithm is always able to provide a "best" answer: the current mixture model parameters. It is suspendable, stoppable and resumable by persisting the contents of the buffer and mixture model parameters to disk. Additional data can easily be accommodated to update an existing model.

3.1 Data Summarization and Sufficient Statistics

We focus on step 4 of the algorithm in which a set of data has already been tagged to be compressed (this process is described in Section 3.3), and the task at hand is to summarize it via sufficient statistics. Let *S* be a record subset consisting of *n* vectors each having *d* attributes. The sufficient statistics for this set are (θ, Γ, *n*), where θ and Γ are a vector and matrix sum, respectively: $\boldsymbol{\theta} = \sum_{x \in S} \mathbf{x}$ and $\Gamma = \sum_{x \in S} \mathbf{x}\mathbf{x}^T$. Note Γ is symmetric, there is no need to store the entire matrix. For diagonal Gaussians, Γ storage collapses to that of a vector. Our SEM implementation assumes diagonal covariance matrices. For *d*-dimensional data, the simplified EM model has $k(2d+1)$ parameters to estimate: the *k* means, the *k* covariance matrix diagonals, and the vector of cluster weights. From (θ, Γ, *n*), the mean over *S* (denoted here μ^S) and the covariance matrix over *S* (denoted as Σ^S) are computed as follows:

$$\boldsymbol{\mu}^S = \frac{1}{n} \cdot \boldsymbol{\theta}, \quad \boldsymbol{\Sigma}^S = \frac{1}{n}\left(\boldsymbol{\Gamma} - \frac{1}{n}\boldsymbol{\theta}\boldsymbol{\theta}^T\right) \qquad \textbf{(5)}$$

Thus the sufficient statistics (θ, Γ, *n*) naturally update the mixture model parameters over *S*. Another useful operation we define is merging two sets of sufficient statistics:

$$merge\left\langle \left(\theta^1, \Gamma^1, n^1\right), \left(\theta^2, \Gamma^2, n^2\right) \right\rangle = \left(\theta^1 + \theta^2, \Gamma^1 + \Gamma^2, n^1 + n^2\right). \qquad \textbf{(6)}$$

3.2 Model Update over Singleton Data Records + Sufficient Statistics

Step 2 of the scalable EM algorithm requires updating the mixture model parameters over the contents of the buffer, consisting of singleton data records and sets of sufficient statistics described above. The *Extended EM (ExEM) Algorithm* performs this operation. ExEM updates the model parameters exactly as the classic EM algorithm (Section 2.1) over singleton data records. Updates over sufficient statistics treat the set represented by (θ, Γ, *n*) as a single data record weighted by *n*. ExEM proceeds as follows:

0. Given the set *S* of singleton data records, a set *T* of sufficient statistics and initial mixture model parameters: initialize the mixture model parameters, set iteration counter $j = 0$: W^0_ℓ, $\mu^{0,\ell}$,$\Sigma^{0,\ell}$, $\ell = 1,\ldots,k$.

1. Having mixture model parameter values at iteration *j*, update them as follows:

1.1. (E-Step, Singleton Records): For singleton records $\mathbf{x} \in S$ compute: $\Pr(\ell \mid \boldsymbol{x}) = \dfrac{W_\ell^j \Pr^j(\boldsymbol{x} \mid \ell)}{\Pr^j(\boldsymbol{x})}$.

1.2. (E-Step, Sufficient Statistics) compute probability that the subset of data points summarized by each (θ, Γ, *n*) belong to population ℓ: $\Pr(\ell \mid (\boldsymbol{\theta}, \boldsymbol{\Gamma}, n)) = \dfrac{W_\ell^j \Pr^j\left(\frac{1}{n}\boldsymbol{\theta} \middle| \ell\right)}{\Pr^j\left(\frac{1}{n}\boldsymbol{\theta}\right)}, \ell = 1,\ldots,k.$

2. **(Extended M-step):** Set: $N(\ell) = \sum_{x \in S} \Pr(\ell \mid \boldsymbol{x}) + \sum_{(\boldsymbol{\theta},\boldsymbol{\Gamma},n)\in T} \Pr(\ell \mid (\boldsymbol{\theta}, \boldsymbol{\Gamma}, n)), \ell = 1,\ldots,k,\; N = \sum_{\ell=1}^{k} N(\ell)$.

The value of $N(\ell)$ is the total portion of the database processed so far having membership in population ℓ. The Gaussian parameters are updated as follows: $W_\ell^{j+1} = \dfrac{N(\ell)}{N}$,

$$\mu^{new,\ell} = \frac{1}{N(\ell)}\left[\sum_{x\in S} \mathbf{x}\cdot\Pr(\ell \mid \mathbf{x}) + \sum_{(\boldsymbol{\theta},\boldsymbol{\Gamma},n)\in T} \boldsymbol{\theta}\cdot\Pr(\ell \mid (\boldsymbol{\theta}, \boldsymbol{\Gamma}, n))\right],$$

$$\Sigma^{new,\ell} = \frac{1}{N(\ell)}\left[\left(\sum_{\mathbf{x}\in S} \mathbf{x}\mathbf{x}^T \cdot \Pr(\ell \mid \mathbf{x}) + \sum_{(\theta,\Gamma,n)\in T} \Gamma \cdot \Pr(\ell \mid (\theta, \Gamma, n))\right) - \right.$$

$$\left.\frac{1}{N(\ell)}\left[\left(\sum_{\mathbf{x}\in S} \mathbf{x}\cdot\Pr(\ell \mid \mathbf{x})\right)\left(\sum_{\mathbf{x}\in S} \mathbf{x}\cdot\Pr(\ell \mid \mathbf{x})\right)^T + \sum_{(\theta,\Gamma,n)\in T} \theta\theta^T \cdot \Pr(\ell \mid (\theta, \Gamma, n))\right]\right], \ell = 1,\ldots k.$$

3. **(Stopping Criteria)** Stop if $\left|\overline{E}^{\,j} - \overline{E}^{\,j+1}\right| \le \varepsilon$, stop. Otherwise set $j = j+1$, go to 1. $\overline{E}^{\,j}$ is the log likelihood of the mixture model at iteration *j* over the contents of the buffer:

$$\overline{E}^j = \left[\sum_{\mathbf{x}\in S} \log\left(\Pr^j(\mathbf{x})\right) + \sum_{(\theta,\Gamma,n)\in T} n \log\left(\Pr^j\left(\frac{1}{n}\theta\right)\right)\right]. \qquad \mathbf{(7)}$$

In addition to the generalization to work over sufficient statistics, ExEM also has the following modification:

4. **(Model Reset for Empty Clusters):** Upon convergence (step 3), the ExEM algorithm employs a step that performs a Gaussian distribution reset if one or more of the clusters has a sum total probability of membership below a given threshold or if two Gaussians converge on the same parameters. This is a problem that plagues both EM and *k*-Means algorithms in high dimensions. In this case, rather than letting such clusters go "inactive", The ExEM algorithm is re-run, with "inactive" clusters reseeded at new points. Specifically, the Gaussian distribution for each "inactive" cluster C^l is reset to the mean of a compressed subset which is least likely under the current mixture model: $\mu^\ell = \arg\min_{\tilde{\mu}} \left\{ \log\left(\Pr^{j+1}(\tilde{\mu})\right) \middle| \tilde{\mu} = \frac{1}{n}\cdot\theta, (\theta,\Gamma,n)\in T \right\}$. The "argmin" notation means to choose the mean of compressed subset which is least likely under the updated mixture model. It also inherits the corresponding covariance matrix of the compressed subset chosen above. Return to step 1.

3.3 Data Compression and Summarization

Data summarization involves two phases. *Primary data compression* purges records that are unlikely to change membership probabilities during future SEM iterations. These are the records near the modes of the *k* cluster distributions. Data records compressed in the primary phase constitute the set of sufficient statistics *DS*. *Secondary data compression* compresses dense regions of data which are not near the Gaussian means. Data records compressed in the secondary phase form the set of sufficient statistics *CS*. *RS* refers to the set of data records which remain in the buffer *uncompressed* (R is for "retained"). Records obtained from the database are initially read into the *RS* structure.

3.3.1 Primary Data Compression

If the current mixture model is correct, then areas near the means, μ^ℓ, $\ell = 1,\ldots,k$ will have greatest density. The full mixture model can be sufficiently updated over future samples via the sufficient statistics of the set of data points near the modes. Essentially the current models are accurate representatives of these points. Primary compression removes a proportion *p* of the data that is best represented by the current model. This is achieved by purging records from *RS* that have greatest log likelihoods. This is equivalent to finding a Mahalanobis radius [DH73] r^ℓ from the center of each cluster ℓ and purging the data items within that radius as illustrated in Figure 3.1 (shaded regions go to DS^ℓ). The subset of data records that is summarized near the mean of Gaussian ℓ is precisely:

$DS^{\ell} = \left\{ \mathbf{x} \;\middle|\; \mathbf{x} \in RS, \left(\mathbf{x} - \mu^{\ell}\right)\left(\Sigma^{\ell}\right)^{-1}\left(\mathbf{x} - \mu^{\ell}\right)^{T} \le r^{\ell} \right\}, \ell = 1,...,k$. The parameter *p* defines the amount of data compressed and determines r^{ℓ}. In Section 5.2 we show that the method is not overly sensitive to this parameter. We set it to 0.5 throughout all experiments. This method guarantees a compression factor of *p*. The set of sufficient statistics of all points compressed in the primary phase is $DS=\{DS^{1},\ldots,DS^{k}\}$.

3.3.2 Secondary Data Compression

The motivation for secondary data compression is based on the observation that for a *dense region of data not near* the means of the *k* population Gaussians (modes), the values of $\Pr(\ell \mid \mathbf{x})$ (Equation 3) will be approximately the same for all data records **x** constituting the dense region. This can be shown as a property of the Gaussians in tail regions of low probability. It follows then that the value of $\Pr(\ell \mid \mu')$ will be a good approximation of this value, where μ' is the mean of the dense subset of data records. Hence the updates to the mixture parameters in the ExEM algorithm basically consist of representing such a dense subset of data by its mean vector and covariance matrix. Compressing dense subsets of data not near the *k* modes allows accurate update of the mixture parameters while freeing up buffer space for future data. Figure 3.2 illustrates this basic idea. The task is to determine these dense subsets.

This compression phase has three parts. The first consists of locating candidate dense regions (sub-clusters) over the data remaining in the buffer. The second consists of applying a criterion to each of the candidate sub-clusters ensuring that they are sufficiently dense. The third part is a merging pass that combines dense sub-clusters as long as they remain dense (as defined by previous criterion). The search for candidate sub-clusters takes place over *RS*, post primary compression purging.

Candidate sub-clusters are determined by applying either the standard *k*-Means clustering algorithm [F90,DH73] or "harsh" EM [NH99] to *RS*. The choice of these "hard" assignment clustering algorithms is justified by the fact to fully compress data records into sufficient statistics, records must have full membership in their representative sub-clusters. To increase the likelihood of identifying dense sub-

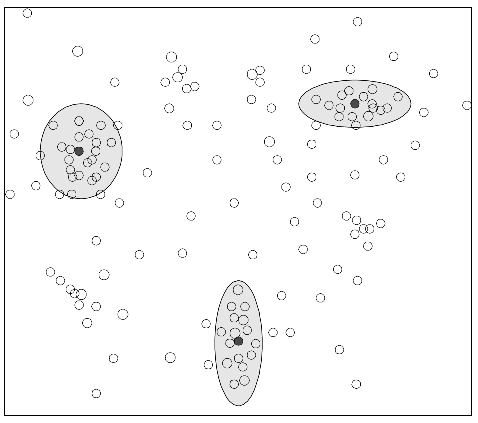
Figure 3.1: Primary data compression.

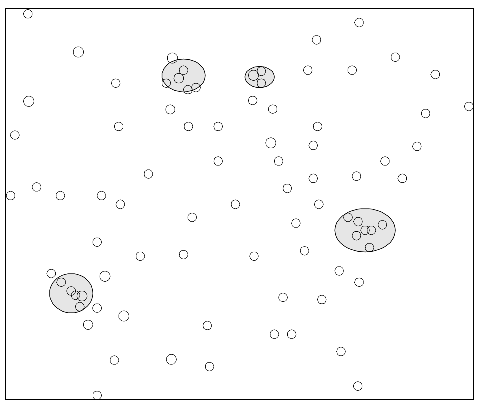
Figure 3.2: Secondary data compression.

clusters, the number of candidate sub-clusters is set to $k' > k$. In our experiments we simply set $k' \approx 2k$. The sub-clustering algorithm is initialized with k' randomly selected data records from *RS*. Once the secondary clustering algorithm has converged, we apply the "density criterion". Subclusters having elements of the covariance diagonal exceeding a parameter $\beta > 0$ are discarded. Hence the sufficient statistics for the data in a given sub-cluster (θ, Γ, n) satisfies: $\max\limits_{j=1,\dots,d}\left\{\frac{1}{n}\cdot\left[\Gamma_{jj}-\frac{1}{n}\theta_j^2\right]\right\} \le \beta$ **(8)**

Suppose $k_2 < k'$ sub-clusters satisfy the *density criterion*. The sufficient statistics of these k_2 sub-clusters are appended to the list of the "dense" sub-clusters found in previous SEM iterations. The list of "dense" sub-clusters is: $CS = \left\{CS^1, CS^2, \dots, CS^r\right\}$, where r is the previous length of the *CS* list plus k_2. The elements of *CS* are the sufficient statistics summarizing the corresponding dense sub-clusters.

The third portion of this compression phase consists of merging the elements of *CS* via hierarchical agglomerative clustering [DH73]. A given CS^i is merged (see Equation 6) with CS^j if their means are nearest (measured by Euclidean distance) and the resulting merged cluster satisfies the density criterion. Note that all operations here are over clustered representation of the buffer, hence the number of items being manipulated is relatively small.

3.4 Scalable EM Algorithm

Bringing all defined notions together, the SEM algorithm proceeds as follows:

0. Given initial mixture model parameter values and a fixed maximum memory buffer size, set SEM major iteration counter $h = 0$, initialize Gaussian mixture model parameter values W^0_ℓ, $\mu^{0,\ell}$,$\Sigma^{0,\ell}$, $\ell = 1,\dots,k$. Initialize sets DS^0, CS^0 and RS^0 to be empty. Set $N^0 = 0$ (total number of records processed).

1. Fill the buffer by loading data rocords into the set *RS*. Set $N^{h+1} = N^h$ + (# of records loaded).

2. Apply ExEM to *RS* initialized with W^h_ℓ, $\mu^{h,\ell}$,$\Sigma^{h,\ell}$, $\ell = 1,\dots,k$. producing W^{h+1}_ℓ, $\mu^{h+1,\ell}$,$\Sigma^{h+1,\ell}$, $\ell = 1,\dots,k$.

3. Perform primary compression:

For $\ell = 1,\dots k$:

(i) Denote set of data to be compressed near mean of Gaussian ℓ as $\overline{DS}^{h+1,\ell}$.

(ii) Merge with previously compressed data: $DS^{h+1,\ell} = merge\left(\overline{DS}^{h+1,\ell}, DS^{h,\ell}\right)$.

Set $DS^{h+1} = \{DS^{h+1,1}, DS^{h+1,2}, \dots, DS^{h+1,k}\}$.

4. Perform secondary compression:

(i) Determine k_1 candidate sub-clusters.

(ii) Remove sub-clusters not satisfying (Equation 8).

(iii) Merge remaining sub-clusters with current CS^h elements while (Equation 8) satisfied.

(iv) Set CS^{h+1} to resulting list of sufficient statistics.

5. Apply stopping criteria. If one full scan of the database is complete, terminate. Else if $\left|\hat{E}^{h} - \hat{E}^{h+1}\right| \le \varepsilon.$, stop. Otherwise, set $h = h+1$, go to 1. $\hat{E}^{h}$ is the average log likelihood of the current mixture model over data processed so far:

$$\hat{E}^{h} = \frac{1}{N^{h}}\left[\sum_{\mathbf{x}\in RS} log\left(\Pr^{h}(\mathbf{x})\right) + \sum_{(\theta,\Gamma,n)\in CS\cup DS} log\left(\Pr^{h}\left(\frac{1}{n}\theta\right)\right)\right].$$

The scheme presented so far involves updating a single mixture model over a database. However the data compression/summarization machinery also admits the possibility of updating multiple mixture models simultaneously, during the one database scan. We describe this generalization next.

We comment here on experimental parameter settings. In our experiments, parameters were set to *constant values and were not adjusted*: ExEM stopping tolerance $\varepsilon = 0.001$ (same as classic EM); parameter p (fraction of data compressed during primary phase) = 0.5; and "density criterion" β set to was set to 0.5. Parameters p and β were varied in experiments evaluating mixture model quality for various values of these quantities.

4 Generalization to Multiple Models

EM, like many other iterative clustering algorithms, is known to be sensitive to initial parameter values [FRB98, BF98, MH98]. EM computes a *local* solution to the problem of maximizing the likelihood of the database given the model. Since this is a local optimization procedure, the quality of the local solution is dependent upon the initial parameter values. Standard practice usually calls for running EM from many different (possibly randomly) initial parameter values and choosing the mixture model solution with best quality (quality is usually dependent upon the particular task at hand, but generally measured by log likelihood – Equation 4). For databases of modest size, this test/evaluate procedure is acceptable (though expensive). To support standard practice, SEM can naturally be generalized to explore multiple mixture models simultaneously. However, how are we to treat the DS, CS, and models themselves when other models are present? The key insights for this generalization are:

1. Singleton data records in the set *RS* and sufficient statistics in *CS* may be shared between models. This is straightforward for the data records in *RS* as singleton data would be used to update the models independently. The sufficient statistics in *CS* may be shared because they are constructed independently of any particular model. Each *CS* element represents a set of dense data records. However, we need to insure that elements of *CS* are "far" from all modes of all models. The updates over *CS* remain the same for any model (see ExEM, Section 3.2, step 1.2).
2. Each model will have its own set summarizing primary compression near its cluster modes. Denote this set of sufficient statistics for model M_i as DS_{Mi}.

3. When updating the mixture model parameters for model M_i , the sufficient statistics in the sets DS_{Mg} for $g \neq i$ are viewed as elements of the set *CS* (i.e. to model M_i, the sufficient statistics of data compressed near the means of other models can be viewed simply as dense regions not necessarily near the means of model M_i: the exact semantics of the set *CS*).

The architecture of SEM (Section 3.4) basically remains the same. Assume now that SEM is initialized with ν initial sets of model parameters. Step 2 must be modified slightly. First, we add an outer loop to update model $i = 1,\dots,\nu$. Second, prior to calling ExEM to update the mixture model parameters for model M_i, the updates over the shared set *CS* must be augmented with the *DS* from all models other than M_i. Hence, the *CS* update for M_i is over the compressed sets:

$$CS \cup DS_{M1} \cup \dots \cup DS_{M(i-1)} \cup DS_{M(i+1)} \cup \dots \cup DS_{M\nu}.$$

There is one more generalization worthy of discussion: the primary data compression stage. When multiple models are present, it is possible that a given data record **x** may satisfy the criteria for primary data compression near means of separate clusters in different mixture models. Suppose **x** satisfies the primary compression criteria for cluster ℓ in mixture model M_i, and also satisfies the criteria for cluster $\tilde{\ell}$ in mixture model $\tilde{M}_g$. Data point **x** will get compressed with population ℓ in mixture model M_i if: $\Pr_{Mi}(\ell \mid \mathbf{x}) > \Pr_{\tilde{M}g}(\tilde{\ell} \mid \mathbf{x})$ and will get compressed with population $\tilde{\ell}$ in mixture model $\tilde{M}_g$ otherwise. Here $\Pr_{Mi}(\ell \mid x)$ is computed as in Equation 3 using the parameters of model M_i . Hence **x** gets summarized with points nearest the means to which it has highest probability of membership.

By naturally exploiting the summary/compression scheme of the scalable EM algorithm, this approach is able to update multiple mixture models simultaneously over a large database in one scan. This fact allows us to explore multiple solutions simultaneously. Since the multiple models "interact" via the shared *CS* set and the updates over the *DS* sets of other models, we actually observe a *synergy* between the multiple simultaneous solutions. We have observed that the scalable architecture run with multiple models typically converges to better local minima than if run independently and sequentially. This is explained by the fact that the models are acting as guards against local minima for each other. Say one model diverges from a good solution and one of its clusters lands on a bad point (happens frequently in EM), then other models, acting as *CS* components from the first model's perspective, will tend to pull it away from that bad point. Overall, the search becomes less sensitive to local shallow minima. Of course deep local minima are much harder to recover from, but that problem will never go away as it is a fundamental problem in optimization theory and practice.

5 Empirical Evaluation

This paper targets scaling the EM algorithm and comparing performance with available alternatives. We do not address the issue of choosing initial models parameters (see [BF98,FRB98,MH98]) nor do we address the issue of setting the number of clusters *k* (an open research problem, e.g. [CS96,S96]). The goal is to study scalability properties and performance for a given *k* and set of initial conditions. Comparing against alternatives is based on quality of obtained solutions. It is an established fact in the statistical literature [PE96,GMPS97,CS96] that EM modeling results in better quality models than other simpler alternatives like *k*-Means (upon which algorithms like BIRCH [ZRL97] and CLARANS [NH94]). There is no prior work on scaling EM so we compare against *de facto* standard practices for dealing with large databases: sampling-based and on-line algorithms. Other scalable clustering algorithms exist, but do not produce mixture model representations of the database probability density function (e.g. BIRCH, DBSCAN [SEKX98], CLARANS, CURE [GRS98], etc.) Three techniques estimating mixture model parameters over large databases are evaluated: scalable EM (SEM), standard or “vanilla” EM run over random samples of the database (VEM), and an online EM implementation (OEM) [NH99, DM92]. The online EM algorithm is a stochastic gradient descent approach that operates by updating the initial mixture model one record at a time [DM92]. A single record is read and its membership probabilities in each of the *k* clusters is computed. The cluster parameters are then updated and the record is purged from memory.

SEM has three major parameters: primary compression factor *p* (Section 3.3.1), standard tolerance β (Section 3.3.2), and number of secondary clusters *k’* (Section 3.3.2). **Throughout all experiments we set these parameters to constant values**: $p = 0.5$, $\beta = 0.5$(maximum global data variance), $k' = 2k$. We study the sensitivity to these parameters in Section 5.2 where we explicitly vary them.

5.1 Synthetic Data

In clustering, it is difficult to judge the quality of a produced solution as the true solution is unknown. Hence studying the algorithm on synthetically generated data is important to verify correctness and to understand the effect of parameters, etc. Synthetic databases were generated by sampling from $k = 10$ multivariate Gaussian distributions. Gaussian means were chosen uniformly on [0.0, 10.0] and diagonal covariance matrices were chosen uniformly on [0.8, 1.2]. It is expected for a large number of attributes, the clusters are fairly separated. For all tests initial mixture model parameters were chosen randomly. Each attribute is normalized to have global mean zero and global standard deviation equal to one to simplify analysis.

5.1.1 Scalability

The running time of SEM is compared with VEM without sampling. All databases in this evaluation had *d* = 25 attributes and had 10k, 400k, 600k and 1M records. SEM was run with maximum buffer size = 5% total database size. SEM and VEM each updated 10 mixture models simultaneously. VEM was not constrained to a single database scan, nor to a limited RAM requirement. Tests were run on a P400 Pentium workstation with 128 MB RAM. Results are summarized in Figure 5.1.

Application of VEM to the database with 1M (1000k) records could not be completed on the system described above and was run on a P300 PentiumII workstation with 256 MB RAM. We note that for databases with 25 continuous attributes, the system RAM limit of 128 MB is reached at 640k records, ignoring any other RAM requirements. Paging was observed for VEM over the databases with 600k and 1000k records. As expected, SEM scales linearly.

However, **note that SEM is running faster than the full in-memory VEM algorithm for databases that fit in memory:** 10k and 400k records. We expect the scalable implementation to run slightly slower in this range. The explanation has two parts. First, we have observed empirically that EM typically tends to iterate fewer times with less data. Second, our approach has a localized memory access pattern, and hence allows the operating system to utilize faster CPU caches. Pentium II processors have on-chip memory caches, and we suspect we are optimizing their use. The traditional VEM implementation does an in-memory scan of the full data at each iteration, hence the likelihood of a cache fault is high.

5.1.2 Cluster Quality

Cluster quality is quantified by the log likelihood of the data under the computed mixture model (Equation 4). This measure quantifies the likelihood that the database records were generated by the computed mixture model [B95]. It was empirically determined that SEM running time was 5 times

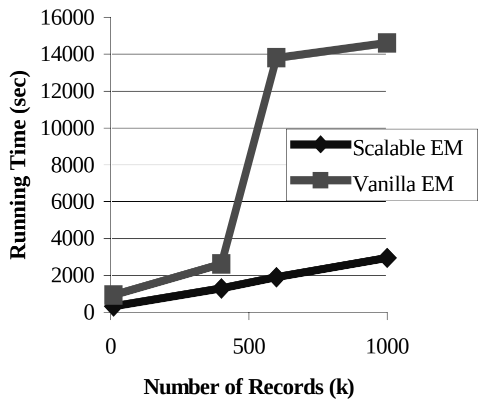

Figure 5.1: Running time versus number of records

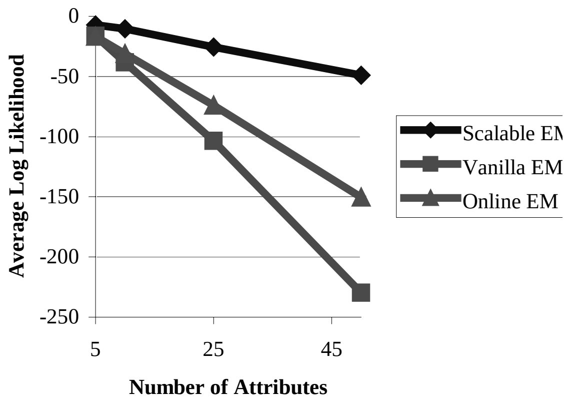

Figure 5.2: Quality versus number of attributes.

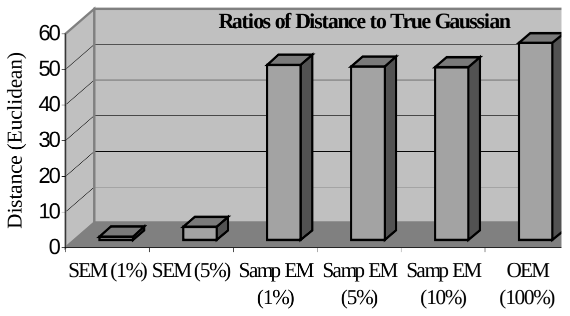

Figure 5.3: Quality of Solutions (20K records, d=20).

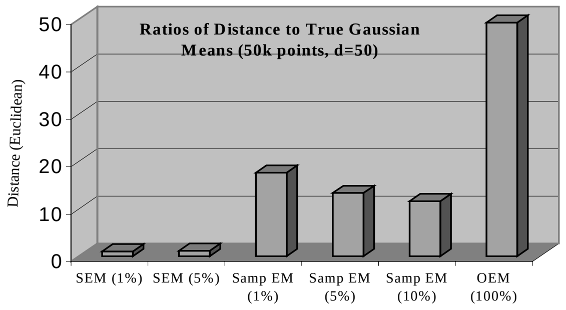

Figure 5.4: Quality of Solutions (50K records, d=50).

VEM running time over buffer sizes/sample percentages of 5% and 10%. Hence, VEM is allowed to explore 5 times the number of SEM models for these percentages. SEM running time was 10 times VEM and OEM time over buffer/samples of 1%, 0.5% and 0.1% and these algorithms explore 10 times the number of SEM models for these values.

Number of Database Attributes and Clusters

Figure 5.2 summarizes results comparing quality of mixture models computed by SEM, VEM and OEM for databases with 50k records and varying number attributes. Cluster quality degrades linearly for mixture models computed by SEM (with buffer size equivalent to 1% of the database), VEM (running over 1% random samples) and OEM. The degradation for SEM is most graceful. Detailed results are shown in Figures 5.3 and 5.4 where we compare the distance between true Gaussian means and computed solutions. Note the graphs show ratios of distances relative to best solution, so *differences are very significant*. Number of clusters for these figures changed from 5 clusters at 20k records, to 10 clusters at 50k records. Similar results are observed for 100 clusters at 100k records, in 100 dimensions.

5.2 Parameter Sensitivity

Varying Primary Compression Factor (see Section 3.3.1)**:** Evaluations of cluster quality for various values of the primary compression factor p were conducted over a synthetic database with 50k records and 50 dimensions. See Figure 5.5. Secondary compression was not used by SEM in these tests. For small values of p (near 0.0), there is little or no compression, the memory buffer fills and SEM terminates

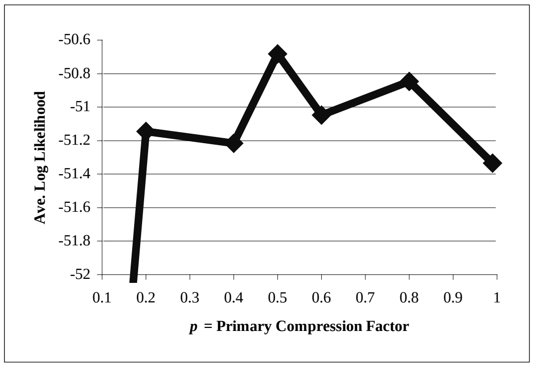

Figure 5.5: Quality vs. primary compression factor.

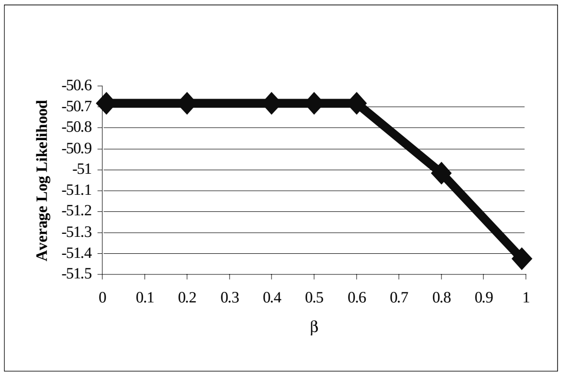

Figure 5.6: Quality versus standard tolerance.

prior to processing the entire database, hence poor mixture models are obtained. For large values of *p* (near 1.0), almost all of the data is compressed in the primary phase and data resolution is lost; poor mixture models are also obtained. Optimal values for *p* occur between these two extremes. The curve exhibits reasonable robustness away from extrema.

Varying Standard Tolerance β (see Section 3.3.2): Evaluations of cluster quality for various values of β were conducted. The primary compression factor *p* was set to 0.5 (default), the maximum buffer size was 100kB. We allowed SEM to construct large candidate secondary sub-clusters to better study the effects of β. Figure 5.6 shows that cluster quality remains high for values of β to 0.8. For small values of β (near 0.0), secondary compression is minimal and hence there is no loss of data resolution. As β is increased, secondary compression plays a larger role. For values of β in [0.2,0.6], when compression does occur, the sub-clusters are "tight" and data resolution is preserved, hence quality solutions are obtained. For large values of β, data resolution is lost and the mixture model quality suffers, as expected.

5.3 Real World Data

SEM parameter settings were set to their defaults. Initial mixture model values were chosen randomly. The sampling rate varied from 1% to 10% for smaller data sets, and were varied from 0.1% to 1% for larger data sets (to keep buffer sizes small as we move to large databases).

Reuters Information Retrieval Database: The Reuters text classification database is derived from the original Reuters-21578 corpus (see www.research.att.com/~lewis/reuters21578/README.txt). It consists of 12,902 news articles. Our data subset consists of word count vectors for the most popular 302 keywords for each of the 12,902 documents. The database was globally normalized and mixture models with *k* = 10 Gaussians were computed. Results are summarized in the following table showing the difference in log likelihood from the best solution. The results are averaged over 10 models for SEM and 50 models for VEM (10% and 5%) and 100 models for OEM and VEM (1%) are shown in Table 5.1.

The models computed via SEM for all buffer sizes/sample percentages are superior.

Figure 5.7 summarizes results of applying SEM and VEM with 10% buffer size/sample percentage for various values of *k*. Notice that SEM exploits the added freedom of more clusters and computes improved models as *k* increases. VEM has great

Table 5.1: Delta Log Likelihoods for Reuters

	Buffer size as percent of DB size		
Method	10%	5%	1%
SEM	**0**	-51.8	-228.8
VEM	$-63.9\text{x}10^4$	$-114.8\ \text{x}10^4$	$-1568.9\ \text{x}10^4$
OEM	-408.4		

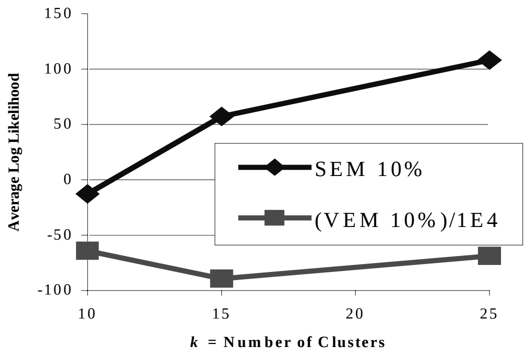

Figure 5.7: Quality versus *k* [Reuters]

difficulty improving solutions as *k* grows.

Census Database: The U.S. Bureau Census "adult" is publicly available from U.C. Irvine repository, but a larger version was obtained from SGI's public archive on data mining. The subset used here consists of 299,285 records each having 11 numeric attributes. Results are summarized in the following table, again these are differences in log likelihood to best model:

Scalable EM (SEM)				Sampling EM (VEM)				OEM
5%	**1%**	0.5%	0.1%	5%	1%	0.5%	0.1%	100%
-0.0178	**0**	-1.8706	-3.0047	-8.119	-14204.2	-1486.35	-85241.8	-33.6861

For all tests SEM produced better mixture models than either VEM or OEM. Given enough memory, VEM will generate the best model by definition. But if the memory budget for is severely constrained, SEM computes superior models over either VEM or OEM on the Census database.

Astronomy Database: The Astronomy database consists of measurements of sky objects from the Second Palomar Sky Survey at Caltech. Number of records is 648,291, each representing objects on a photographic plate having 29 numeric attributes. Results are summarized in the following table.

Scalable EM (SEM)			Sampling EM (VEM)			**OEM**
1%	0.5%	**0.1%**	1%	0.5%	0.1%	**100%**
-197.826	-86.737	**0**	-216.901	-225.36	-892.29	**43.490**

Note that in this case OEM performed the best. This is a bit surprising. In fact it does better than running VEM with even larger samples (e.g. 10% or more). We have no good explanation at this time. However, note that SEM significantly outperformed VEM, consistent with our expectation.

REV Digits Recognition Database: The REV Digits database consists of 13,711 data items with 64 attributes. Due to small number of records, we used larger buffer sizes. Each record represents the gray-scale level of an 8 by 8 image of a handwritten digit. Each attribute has been globally normalized to have mean zero and standard deviation one. Mixture models with k = 10 Gaussian components were computed. On this data set, we observe comparable performance between SEM and VEM at the 5% and 10% sample levels. OEM did significantly worse (by a factor of 1.3). VEM at 1% level deteriorated to OEM's performance while SEM remained at same original level. In this data set we believe all clustering methods are computing poor models (hence similar quality). It was reassuring that SEM remained stable as buffer size/sample percentage was decreased.

6 Related Work

The closest approach to this work is proposed in [NH99] in which incremental versions of the general EM algorithm are analyzed. In this case the E-Step consists not only of updating the distribution over the unobserved variables, but also updating the sufficient statistics given this distribution. Its operation is similar to OEM except that sufficient statistics are kept with each cluster. Our method introduces the notion of secondary compression and the *CS* set. Furthermore, our framework is specifically designed to operate over a single scan of the database. In contrast, the results of [NH99] indicate that multiple scans of the database are required.

The BIRCH algorithm [ZRL97] first summarizes the database in a main memory, balanced tree structure, the CF-tree. The resolution of the data summarization is determined by the amount of main memory allocated to this structure. The nodes of the CF-tree compress the data as spherical Gaussian distributions with a single scalar variance. In contrast, SEM admits covariance matrices. BIRCH also takes a second pass to cluster the compressed representation once the first pass terminates. The fundamental difference between BIRCH and the scalable EM algorithm lies in the approach to data compression. BIRCH compresses data as a step independent of the clustering algorithm whereas the scalable EM algorithm's compression is closely related to the current fit of the mixture model to the data. In addition, SEM can accommodate probabilistic cluster membership weights. BIRCH is designed to support the k-Means algorithm and not the probabilistic EM algorithm. However, the CF-tree structure can be used to do non-parametric (kernel-based) density estimation with a large number of kernels [LRZ96].

A set of related clustering algorithms, which for the same reasons cannot be compared with our density estimation approach, are DBSCAN, GDBSCAN [SEKX98], and CLARANS [NH94] which are designed primarily for spatial data clustering. DBSCAN and GDBSCAN are region growing algorithms and do not aim to probabilistically model the density. The same applies to the CLIQUE algorithm [AGGR98] which grows dense data regions by attempting to find all clustered subspaces of the original data space and present the result to the user in a minimal DNF expression. CLIQUE requires many data scans to identifying cluster subspaces in the bottom-up fashion. It does not naturally extend itself to a probabilistic interpretation. In fact the authors state that a data mining algorithm "… should not presume some canonical form for data distribution."

The CURE algorithm [GRS98] is a scalable clustering technique based upon the hierarchical agglomerative clustering (HAC) approach. Initially each data record is considered a cluster and the two nearest clusters are merged. The difference between CURE and standard HAC is that clusters are represented by a given number of "well-scattered" records within each cluster. This set of points, determined in the merging procedure are "shrunk" toward the cluster mean by a multiplier α in [0,1]. The authors state that the shrinking process reduces the effect of outliers. It is also stated that representing the cluster by a set of "well-scattered" points allows CURE to recognize non-spherical clusters. The CURE algorithm is scaled via random sampling and the authors present a theoretical result stating the sufficient sample size so that a given number of points from each cluster appear in the random sample. The algorithm is also scaled by employing a pre-clustering phase and applying CURE to the result. This is conceptually similar to the technique employed by BIRCH [ZRL97].

In addition to differences between hierarchical and mixture model approaches [DH73], the fundamental difference between CURE and scalable EM are threefold: (i) data compression to achieve scalability is done independent of the clustering algorithm, there is no notion of fit of the clustering solution to the database, (ii) the approach is not naturally extended to multiple model updates, and (iii) the representation of a cluster as a group of "well-separated" records does not readily give rise to a probabilistic model or notion of density estimation.

7 Conclusion

The algorithm presented effectively scales to very large databases. Required memory consists of holding a small sub-sample in RAM. All clustering (primary and secondary) occurs over the contents of the buffer. The approach can be run with a small RAM buffer and can effectively be applied to large-scale databases. We have observed that running multiple solutions typically results in improved performance of the compression schemes since the *synergy between the models* being explored allow for added opportunity for compression. In turn this frees up more space in the buffer and allows the algorithm to maximize its exposure to new data during model updates. The results indicate that the proposed scalable scheme produces good solutions and outperforms simple sampling approaches. Simultaneous multiple model updating decreases the likelihood of converging in a bad local minima with empty clusters. The retained data vectors/compressed clusters do indeed lead to much better solutions when compared with a memoryless OEM. This demonstrates memory in addition to the model itself is essential.

Results on real-world data support the utility of SEM in real clustering/data modeling applications. SEM is particularly suited to modeling databases with many attributes when memory requirements are severe – results on the Reuters corpus (302 attributes) and the REV Digits database (64 attributes) show SEM to be superior on this situation. When applied to databases with a large number of records with a severely constrained memory budget, SEM produces superior models to VEM (Census and Astronomy).

From a DBMS perspective the scheme presented possesses many desirable properties. It satisfies the goals of one scan or less, forward-only cursor to data for efficient operation over join views, incremental behavior, and anytime algorithm properties. The scheme can be viewed as a generalized version of a sampling-based scheme. Terminating after one buffer fill, SEM is VEM over random samples. By allowing the buffer hold all data in memory, we simulate in-memory VEM. However, our experiments indicate that even if huge main memory was available on the server, it would not be desirable to increase the buffer. The smaller data buffers cause the algorithm to exhibit an improved cache behavior due to localized data access in the inner loop. This leads to faster execution time since the number of accesses to main memory are minimized. Most modern PC CPU's have on-chip memory caches.

Fortunately, the scalable framework decouples data and model. In order to update the model, degree of membership of each data point in the buffer to each cluster is the only requirement. Hence this scheme accommodates clustering over discrete data by choosing an appropriate data distribution (e.g. the Multinomial distribution). We are currently investigating using this framework on mixed attribute types.

8 References

[AGGR98] R. Agrawal, J. Gehrke, D Gunopulos and P Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *", in Proc. ACM SIGMOD Int. Conf. On Management of Data (SIGMOD98).*

[BR93] J. Banfield and A. Raftery, "Model-based gaussian and non-Gaussian Clustering", Biometrics, vol. 49: 803-821, pp. 15-34, 1993.

[B95] C. Bishop, 1995. Neural Networks for Pattern Recognition. Oxford University Press.

[BB95] L. Bottou and Y. Bengio. "Convergence Properties of the K-Means Algorithm", in Advances in Neural Processing Systems 7, G. Tesauro, D. S. Touretsky, and T. K. Leen (Eds.), MIT Press, 1995.

[B*96] R. Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, and E. Simoudis, "Industrial Applications of Data Mining and Knowledge Discovery." Communications of ACM 39(11). 1996.

[BMS97] P. S. Bradley, O. L. Mangasarian, and W. N. Street. 1997. "Clustering via Concave Minimization", in Advances in Neural Information Processing Systems 9, M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.) pp 368-374, MIT Press, 1997.

[BF98] P. Bradley and U. Fayyad, "Refining Initial Points for K-Means Clustering", Proc. 15th International Conf on Machine Learning, Morgan Kaufmann, 1998.

[BFR98] P. Bradley, U. Fayyad, and C. Reina, "Scaling Clustering Algorithms to Large Databases", *Proc. 4th International Conf. on Knowledge Discovery and Data Mining (KDD98)*, AAAI Press, 1998.

[CS96] P. Cheeseman and J. Stutz, "Bayesian Classification (AutoClass): Theory and Results", in in Advances in Knowledge Discovery and Data Mining, Fayyad, U., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy(Eds.), pp. 153-180. MIT Press, 1996.

[DM92] C. Darken and J. Moody. "Towards Faster Stochastic Gradient Search". In *Advances in Neural Information Processing Systems 4,* Moody, Hanson, and Lippmann, (Eds.), Morgan Kaufmann, Palo Alto, 1992.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via theEM algorithm". Journal of the Royal statistical Society, Series B, 39(1): 1-38, 1977.

[DH73] R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis. New York: John Wiley and Sons. 1973

[FHS96] U. Fayyad, D. Haussler, and P. Stolorz. "Mining Science Data." Communications of the ACM 39(11), 1996.

[FPSU96] Fayyad, U., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.) Advances in Knowledge Discovery and Data Mining. MIT Press, 1996.

[FDW97] U. Fayyad, S.G. Djorgovski and N. Weir, "Application of Classification and Clustering to Sky Survey Cataloging and Analysis", Computing Science and Statistics, vol. 29(2), E. Wegman and S. Azen (Eds.), pp. 178-186, Fairfax, VA: Interface Foundation of North America, 1997.

[FRB98] U. Fayyad, Cory Reina, and Paul Bradley, "Refining Initialization of Clustering Algorithms", *Proc. 4th International Conf. On Knowledge Discovery and Data Mining (KDD98),* AAAI Press, 1998.

[F87] D. Fisher. "Knowledge Acquisition via Incremental Conceptual Clustering". Machine Learning, 2:139-172, 1987.

[F90] K. Fukunaga, Introduction to Statistical Pattern Recognition, San Diego, CA: Academic Press, 1990.

[GKR98] D. Gibson, J. Kleinberg, P. Raghavan. "Clustering Categorical Data: An Approach Based on Dynamical Systems". *Proc. Of VLDB-98*. 1998.

[GMPS97] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. "Statistical Themes and Lessons for Data Mining", Data Mining and Knowledge Discovery, vol. 1, no. 1. 1997.

[GRS98] S. Guha, R. Rastogi and K. Shim. "CURE: An Efficient Clustering Algorithm for Large Databases*", in Proc. ACM SIGMOD Int. Conf. On Management of Data (SIGMOD98).* ACM Press, 1998.

[KR89] L. Kaufman and P. Rousseeuw, 1989. Finding Groups in Data, New York: John Wiley and Sons.

[LRZ96] M. Livny, R. Ramakrishnan and T. Zhang. "Fast Density and Probability Estimation Using CF-Kernel Method for Very Large Databases". Computer Sciences Technical Report, Computer Sciences Department, University of Wisconsin-Madison, Madison, WI. July, 1996.

[M67] J. MacQueen, "Some methods for classification and analysis of multivariate observations", in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume I, Statistics, L. M. Le Cam and J. Neyman (Eds.). University of California Press, 1967.

[MH98] M. Meila and D. Heckerman, 1998. "An experimental comparison of several clustering methods", Microsoft Research Technical Report MSR-TR-98-06, Redmond, WA.

[NH94] R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining", *Proc of VLDB-94*, 1994.

[NH99] R.M. Neal and G.E. Hinton. "A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants", *Learning in Graphical Models*, M. I. Jordan (ed.), MIT Press, 1999.

[PE96] D. Pregibon and J. Elder, "A statistical perspective on knowledge discovery in databases", in Advances in Knowledge Discovery and Data Mining, Fayyad, U., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy(Eds.), pp. 83-116. MIT Press, 1996.

[R92] E. Rasmussen, "Clustering Algorithms", in Information Retrieval Data Structures and Algorithms, W. Frakes and R. Baeza-Yates (Eds.), pp. 419-442, Upper Saddle River, NJ: Prentice Hall, 1992.

[SEKX98] J. Sander, M. Ester, H. Kriegel, X. Xu, "Density-based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications", *Data Mining and Knowledge Discovery*, 2:2, pp. 169-194, 1998.

[S92] D. W. Scott, Multivariate Density Estimation, New York: Wiley. 1992

[SI84] S. Z. Selim and M. A. Ismail, "K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality." IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 1, 1984.

[S86] B.W. Silverman, Density Estimation for Statistics and Data Analysis, London: Chapman & Hall, 1986.

[S96] P. Smyth, "Clustering using Monte Carlo Cross-Validation*", Proc. of Second International Conf. on Knowledge Discovery and Data Mining (KDD96),* AAAI Press, 1996.

[ZRL97] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: A new data clustering algorithm and its applications." Data Mining and Knowledge Discovery 1(2). 1997.