Scaling EM (Expectation-Maximization) Clustering to Large Databases

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

Microsoft Research

November 1998

Revised October 1999

Technical Report

MSR-TR-98-35

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Efficient Probabilistic Data Clustering: Scaling 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 clustering algorithms typically center upon an iterative refinement optimization procedure to compute a locally optimal clustering solution that maximizes the fit to data. These algorithms typically require many database scans to converge, and within each scan they require the access to every record in the data table. For large databases, the scans become prohibitively expensive. We present a scalable implementation of the Expectation-Maximization (EM) algorithm. The database community has focused on distance-based clustering schemes and methods have been developed to cluster either numerical or categorical data. Unlike distance-based algorithms (such as K-Means), EM constructs proper statistical models of the underlying data source and naturally generalizes to cluster databases containing both discrete-valued and continuous-valued data. The scalable method is based on a decomposition of the basic statistics the algorithm needs: 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 main memory buffer and requires at most a single database scan. Data resolution is preserved to the extent possible based upon the size of the main memory buffer and the fit of the current clustering model to the data. We extend the method to efficiently update multiple models simultaneously. Computational tests indicate that this scalable scheme outperforms sampling-based approaches – the straightforward alternatives to "scaling" traditional in-memory implementations to large databases.

1 Preliminaries and Motivation

Data clustering is important in many fields, including data mining [FPSU96], statistical data analysis [KR89,BR93], compression [ZRL97], and vector quantization [DH73]. Applications include data analysis and modeling [FDW97,FHS96], image segmentation, marketing, fraud detection, predictive modeling, data summarization, general data reporting tasks, data cleaning and exploratory data analysis [B*96]. Clustering is a crucial data mining step and performing this task over large databases is essential.

A general view of clustering places it in the framework of density estimation [S86, S92, A73]. Clustering can be viewed as identifying the dense regions of the data source. An efficient representation of the probability density function is the *mixture model*, which asserts that the data is a combination of *k* individual component densities, corresponding to the *k* clusters. Basically, the problem is this: given data records (observations), identify a set of *k* populations in the data, and provide a model (density distribution) of each of the populations. Since the model assumes a mixture of populations, it is often referred to as a *mixture model*.

The Expectation-Maximization (EM) algorithm [DLR77, CS96] is an effective and popular technique for estimating the mixture model parameters or fitting the model to the database. The EM algorithm iteratively refines an initial cluster model to better fit the data and terminates at a solution which is locally optimal or a saddle point of the underlying clustering criterion [DLR77, B95]. The objective function is log-likelihood of the data given the model measuring how well the probabilistic model fits the data.

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 compute correct statistical models of the data. The K-Mean algorithm minimizes the sum of squared Euclidean distances of between data records in a cluster and the cluster's mean vector. This assignment criterion implicitly assumes that clusters are represented by spherical Gaussian distributions located at the *k* cluster means [BB95, B95]. Since the K-Mean algorithm utilizes the Euclidean metric, it does not generalize to the problem of clustering discrete or categorical data. The K-Mean algorithm also uses a membership function which assigns each data record to exactly one cluster. This harsh criteria does not allow for uncertainty in the membership of a data record in a cluster. The mixture model framework relaxes these assumptions.

Due to the probabilistic nature of the mixture model, arbitrary shaped clusters (i.e. non-spherical, etc.) can be effectively represented by the choice of suitable component density functions (e.g. Poission, non-spherical Gaussians, etc.). Categorical or discrete data is similarly handled by associating discrete data distribution over these attributes (e.g. Mutinomial, Binomial, etc.).

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 *average age* over the data records is 41 years and an *average income* is $26K/year (with associated variances). However, this may be rather deceptive and uninformative. The data may be a mixture of working people, retired people, and

children. A more informative summary might identify these subsets or clusters, and report the cluster parameters. Such results are shown in Table 1.1:

Table 1.1: Sample data summary by segment

"name" (not given)	Size	Average Age	Average Income
"working"	45%	38	$45K
"retired"	30%	72	$20K
"children"	20%	12	$0
?	5%	NULL	NULL
TOTAL	100%	41	$26K

Note that the "name" column is not given (only the age and income columns are given). Note also that the row labeled total gives less insight of the data than does the summary into 4 rows (or segments). Assuming no apriori definition of the various sub-populations, how can such segments be identified? How can this identification be made when the data has many more dimensions (attributes) and the various patterns may not be obvious? This is where clustering plays an important role and identifies these dense regions in multidimensional data.

We specifically address the problem of computing mixture models over large databases. Over such databases, hundreds of iterations or more may be required by iterative refinement clustering procedures such as EM. Although guaranteed to terminate finitely, a general bound on the number of iterations required by EM is not available. We make the assumption that a single scan over a large database is expensive, and computing a mixture model over large databases via the standard EM implementation is unacceptable. We present a scalable version of the EM algorithm supporting the following requirements.

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

The single forward only data scan addresses 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 rescanning the data on the server may be an expensive operation in the background.

We present a scalable decomposition of the EM algorithm satisfying the goals stated above. The problem of computing a mixture model is decomposed resulting in the ability to cluster arbitrarily large databases while utilizing a small amount 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 can be compressed, and records that must be retained in memory. We demonstrate that the scalable EM algorithm (SEM) indeed preserves clustering fidelity and that it outperforms traditional methods addressing large databases via sampling. Localized data access properties of SEM have additional benefits relating to improved utilization of fast, on-chip memory caches in modern CPU's. This results in faster execution times even on machines having enough resident memory to hold the entire database in main memory.

The framework presented is general and can accommodate many iterative refinement clustering algorithms including K-Mean [BFR98]. This work generalizes previous work on K-Mean and addresses probabilistic clustering in which every data point belongs to *all* clusters, but with different probability. This generalizes to include realistic situations in which, say a customer of a web site, really belongs to two or more segments (e.g. sports enthusiast, high tech enthusiast, and coffee addict). 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 (data records belonging to different clusters with different probabilities of membership). A scalable approach to density estimation is proposed in [ZRL99] utilizing the *CF-Tree* data structure from [ZRL97], which is limited to continuous-valued data in which the Euclidean metric is appropriate.

We specifically address the problem of clustering a large database into segments consisting of records which are more similar to other members the same segment than those of other segments. The probabilistic nature of the models we use admits overlap naturally when appropriate. The mixture model does not require the specification of distance metrics, readily admitting categorical *and* continuous attributes (simultaneously). The EM algorithm [DLR77, CS96] has been shown to be superior to other alternatives for statistical modeling purposes [GMPS97, PE96, B95, CS96, NH99]. Utility of the statistical model computed via EM has been demonstrated in

approximating OLAP aggregate queries on continuous data [SFB99] and approximating nearest-neighbor queries [BFG99]. These applications require the statistical semantics and theory, which is a substantial advantage over clustering methods that do not derive statistically proper models.

We next discuss the standard EM approach to mixture model estimation.

1.1 Mixture Model Estimation via the EM Algorithm

The mixture model approximates the data distribution by fitting k component density functions f_h, $h=1,\ldots,k$ to a database D having m records and d attributes. Let $x \in D$ be a record from D, the mixture model probability density function evaluated at x is:

$$p(x) = \sum_{h=1}^{k} w_h \cdot f_h(x \mid \phi_h). \qquad \textbf{(1)}$$

The weights w_h represent the fraction of database records belonging to cluster h and sum to one: $\sum_{h=1}^{k} w_h = 1, w_h \geq 0$. The functions $f_h(\ x \mid \phi_h\ )$ $h=1,\ldots,k$ are the cluster or component density functions modeling the records of the h-th cluster, and ϕ_h represents the specific parameters used to compute the value of f_h (e.g. for a Gaussian component density function, ϕ_h is the mean and covariance matrix). We note that the mixture model can approximate any continuous density function given enough components and properly chosen mixture model parameters [B95].

The mixture model also allows for "overlapping" clusters in the sense that data records may belong to all k clusters, but with different probabilities of membership. The assignment of data points to clusters in the EM algorithm generalizes the "hard" assignment employed by K-Means-type algorithms [BR93, BMS97, DH73, F90] where data records are members of one and only one cluster. The probability of membership or "weight" of data record x in cluster h is:

$$w_h(x) = \frac{w_h \cdot f_h(x \mid \phi_h)}{\sum_i w_i \cdot f_i(x \mid \phi_i)}. \qquad \textbf{(2)}$$

By making the assumption that the attributes of the database are independent over records within a given cluster, the component density functions can be decomposed as a product of density functions over each attribute $j = 1,\ldots,d$:

$$f_h(x \mid \phi_h) = \prod_{j=1}^{d} f_{h,j}(x_j \mid \phi_h). \qquad \textbf{(3)}$$

Suppose the database of objects has $d = 5$ attributes and attributes A_1 and A_2 are "color" and "shape"; and attributes A_3, A_4, and A_5 corresponds to continuous coordinates indicating location in

3-dimensional space. The categorical attributes A_1 and A_2 can be modeled by choosing functions $f_{h,1}$ and $f_{h,2}$ to be Multinomial distributions [B95]. The continuous attributes A_3, A_4, and A_5 can be modeled by a multivariate Gaussian distribution in 3 dimensions [DH73]. The probability of a data record x in cluster h is then the product of the probability of the given "color" value given by $f_{h,1}$, the probability of the given "shape" value given by $f_{h,2}$, and the probability returned by the multivariate Gaussian for cluster h over the values of A_3, A_4, and A_5.

Although the framework we present is general enough to address mixture model estimation over large databases having both continuous and discrete attributes, we focus on its application to continuous-valued data, modeled by multivariate Gaussians. This choice of Gaussians for continuous-valued data 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 multivariate Gaussian distribution for cluster $h = 1,\ldots k$, is parameterized by the d-dimensional mean vector μ_h and $d \times d$ covariance matrix Σ_h :

$$f_h(x \mid \mu_h, \Sigma_h) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_h|}} exp\left(-\frac{1}{2}(x-\mu_h)^{\mathrm{T}}(\Sigma_h)^{-1}(x-\mu_h)\right), \qquad \textbf{(4)}$$

where x and μ_h are column vectors, the superscript ^{T} indicates transpose to a row vector, $|\Sigma_h|$ is the determinant of Σ_h and $(\Sigma_h)^{-1}$ is its matrix inverse. The Gaussian mixture model parameters consist of the means and covariance matrices for each cluster $h = 1,\ldots,k$ along with the weights w_h associated with each cluster. Let $\Phi = \{ (w_h, \mu_h, \Sigma_h), h = 1,\ldots,k \}$ be the collection of mixture model parameters. The quality of a given set of parameters Φ is a measure of how well the corresponding mixture model "fits" the data. This is quantified by the log-likelihood of the data given the mixture model:

$$L(\Phi) = \sum_{x \in D} \log\left(\sum_{h=1}^{k} w_h \cdot f_h(x \mid \mu_h, \Sigma_h)\right). \qquad \textbf{(5)}$$

The EM algorithm begins with an initial estimation of Φ and iteratively updates it. The sequence of Φ-values is such that $L(\Phi)$ is non-decreasing at each iteration [DLR77, B95]. We next outline the standard EM algorithm.

Algorithm 1 Standard Expectation Maximization (EM): Given a database D with m records with d continuous-valued attributes, a stopping tolerance $\varepsilon > 0$ and mixture model parameters Φ^j at iteration j, compute Φ^{j+1} at iteration $j+1$ as follows:

1. For each database record $x \in D$:

Compute the membership probability of x in each cluster $h = 1,\ldots,k$:

$$w_h^j(x) = \frac{w_h^j \cdot f_h(x \mid \mu_h^j, \Sigma_h^j)}{\sum_i w_i^j \cdot f_i(x \mid \mu_i^j, \Sigma_i^j)}.$$

2. Update mixture model parameters: $w_h^{j+1} = \sum_{x \in D} w_h^j(x)$,

$$\mu_h^{j+1} = \frac{\sum_{x \in D} w_h^j(x) \cdot x}{\sum_{x \in D} w_h^j(x)} \quad \Sigma_h^{j+1} = \frac{\sum_{x \in D} w_h^j(x)\left(x - \mu_h^{j+1}\right)\left(x - \mu_h^{j+1}\right)^{\mathrm{T}}}{\sum_{x \in D} w_h^j(x)} \quad h = 1,\ldots,k.$$

Stopping Criteria: If $\left|L(\Phi^j) - L(\Phi^{j+1})\right| \le \varepsilon$, stop. Else set $j \leftarrow j+1$ and go to 1. $L(\Phi)$ is given in (5) above.

Note that a full data scan is required at each iteration of the standard EM algorithm to compute the membership probability for each data record in each cluster in step 1. The number of iterations required before the stopping criteria is satisfied is dependent upon the initial parameter values and the actual data distribution. In general the number of iterations is arbitrary but the procedure is guaranteed to converge. We next discuss a scalable variant of EM requiring at most a single database scan.

2 Model Estimation over Large Databases

Motivation for this work centers on the observation that EM may be applied to a batch data sample which may be *triaged* based on identifying information from the sample that needs to be preserved in detail, summarized, or parts which do not need to be retained. Effective parameter estimation can be obtained by selectively storing "important" portions of the database and summarizing data in other regions. The approach assumes an interface to the database allowing the algorithm to load sequential random data samples. The allocated main memory buffer is initially filled with a random sample from the database. The given initial mixture model is updated over the contents of the buffer via the standard EM algorithm. Then each data point in the buffer is classified, depending upon the degree to which the data point is summarized by the current mixture model, as belonging to one of three sets.

1. The retained set R consisting of records that will remain in the buffer and are not summarized. The data in the set R consist of records yielding the greatest information with respect to the cluster model. Elements of the set R are typically records with uncertainty in cluster membership.

2. The discard set B consists of records which can be safely discarded and modeled effectively by the sufficient statistics associated with one of the clusters. Elements of the set B are typically records whose cluster membership is nearly "certain". Since they will not change cluster membership, it suffices to summarize them and update the corresponding cluster only through sufficient statistics. Note that admitting data into the set B and subsequently only storing the data's sufficient statistics frees space in the buffer allowing the method to load more data. See Section 2.3.1 on primary data summarization.
3. The compressed set C consists of sets of records that update each component as a unit. These records are removed from the memory buffer and updates to the mixture model occur only over their sufficient statistics. Similar to 2 above, representing sets of records in C by their sufficient statistics frees main memory allowing the method to load more data. See Section 2.3.2 on secondary data summarization.

Prior to specifying the algorithm, we define a few useful terms. For a set S, $|S|$ denotes its cardinality. The average log-likelihood for a mixture model with parameters Φ over a database with m records is given by:

$$\bar{L}(\Phi, m) = \frac{1}{m} L(\Phi). \qquad \textbf{(6)}$$

Here $L(\Phi)$ is defined in (5).

We formalize the algorithm now noting that specific components are described in detail in following sections.

Algorithm 2 Scalable EM (SEM): Initialize the summary sets $R^0 = \varnothing$, $C^0 = \varnothing$ and $B^0 = \varnothing$. Initialize the number of data points processed $m^0 = 0$. Given a database D with m records, initial mixture model parameters Φ^0, an interface to database D that supplies sample subsets $S^j \subset D$, a specified memory buffer size and a stopping tolerance ε, set iteration counter $j = 0$ and update Φ^j to Φ^{j+1} as follows:

1. Set $R^{j+1} = R^j \cup S^j$ so that memory for $R^{j+1} \cup C^j \cup D^j$ equals the buffer size. Set $m^{j+1} = m^j + |S^j|$.

2. Apply extended EM (ExEM, Section 2.2) to $R^{j+1} \cup C^j \cup D^j$, updating Φ^j to Φ^{j+1}.

3. Partition singleton data points in R^{j+1}:

3.1 Identify subsets updating single Gaussian components (see Section 2.3.1, Primary Data Summarization). Augment D^{j+1} incorporating sufficient statistics from these subsets, remove subset from R^{j+1}.

3.2 Identify subsets updating Gaussian components as a group (see Section 2.3.2, Secondary Data Summarization). Augment C^{j+1} incorporating sufficient statistics from these subsets, remove subsets from R^{j+1}.

Stopping Criteria: If $j > 1$ and $\left|\overline{L}(\Phi^j, m^j) - \overline{L}(\Phi^{j+1}, m^{j+1})\right| \le \varepsilon$, terminate. If $m^{j+1} = m$, terminate (database D has been fully scanned). Otherwise set $j \leftarrow j+1$, go to 1. $\overline{L}$ is given in (6) above.

We elaborate on the SEM algorithm by first by describing the sufficient statistics for the Gaussian mixture model. Note that sufficient statistics are specific to the choice of cluster or component distributions.

2.1 Data Summarization and Storage

Step 3 of the algorithm determines data records to be added to the summary sets B or C. A data record entering either of the sets B or C is summarized with other elements of the set via sufficient statistics, which are then used to update the Gaussian mixture model. After the data records are summarized, their sufficient statistics remain in the main memory buffer and the individual data records used to generate the statistics are purged. We next describe the specific sufficient statistics used to update the Gaussian mixture model.

Let $S = \{x^1, x^2, \ldots, x^N\} \subset D$, be a subset of data to be summarized by the sufficient statistics triple (θ,Γ,N): $\theta = \sum_{i=1}^{N} x^i$ is a vector and $\Gamma = \sum_{i=1}^{N} \left(x^i\right) \cdot \left(x^i\right)^{\mathrm{T}}$ is an $n \times n$ matrix. Computing the mean and covariance of the set S, μ^S and Σ^S, from triple (θ,Γ,N), is as follows:

$$\mu^S = \frac{1}{N}\theta, \quad \Sigma^S = \frac{1}{N}\left(\Gamma - \frac{1}{N}\theta \cdot \theta^{\mathrm{T}}\right).$$

The sufficient statistics resulting from merging two sets S^1 and S^2 represented by (θ^1,Γ^1,N^1) and (θ^2,Γ^2,N^2), respectively, are obtained for $S^1 \cup S^2$ by: $(\theta^1 + \theta^2, \Gamma^1 + \Gamma^2, N^1 + N^2)$.

2.2 Model Update over Sample + Sufficient Statistics

Step 2 of Algorithm 2 requires updating the mixture model parameters over the contents of the buffer: $R^{j+1} \cup C^j \cup B^j$ consisting of singleton data records $x \in R^{j+1}$ and sets of sufficient statistics $(\theta,\Gamma,N) \in C^j \cup B^j$. The *Extended EM (ExEM) Algorithm* performs this operation. ExEM updates the model parameters exactly as the standard EM algorithm (Algorithm 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.

Prior to specifying the ExEM algorithm, we define the log-likelihood for a given set of mixture model parameters $\Phi = \{ (w_h, \mu_h, \Sigma_h), h = 1,\ldots,k \}$ over the contents of the main memory buffer.

$$\hat{L}(\Phi, R, C, B) = \sum_{x \in R} \log\left(\sum_{h=1}^{k} w_h \cdot f_h(x \mid \mu_h, \Sigma_h) \right) + \sum_{(\theta,\Gamma,N) \in C \cup B} N \cdot \log\left(\sum_{h=1}^{k} w_h \cdot f_h\left(\frac{1}{N}\theta \,\middle|\, \mu_h, \Sigma_h \right) \right). \quad \textbf{(7)}$$

Algorithm 3: Extended EM (ExEM): Given the set R^{j+1} of singleton data records, a set $C^j \cup B^j$ of sufficient statistics and initial mixture model parameters Φ^0, set iteration counter $t = 0$, update Φ^t to Φ^{t+1} as follows:

1. **(Compute membership for singleton records in R^{j+1}):** For singleton records $x \in R^{j+1}$ compute the membership probability of x in each cluster $h = 1,\ldots,k$:

$$w_h^t(x) = \frac{w_h^t \cdot f_h(x \mid \mu_h^t, \Sigma_h^t)}{\sum_i w_i^t \cdot f_i(x \mid \mu_i^t, \Sigma_i^t)}.$$

2. **(Compute membership for sufficient statistics in $C^j \cup B^j$):** Compute the membership probability of each subset summarized by (θ, Γ, N) in each cluster $h = 1,\ldots,k$:

$$w_h^t((\theta,\Gamma,N)) = \frac{w_h^t \cdot f_h\left(\frac{1}{N}\theta \,\middle|\, \mu_h^t, \Sigma_h^t \right)}{\sum_i w_i^t \cdot f_i\left(\frac{1}{N}\theta \,\middle|\, \mu_i^t, \Sigma_i^t \right)}$$

3. **(Update Cluster Parameters):** Set:

$$N(h) = \sum_{x \in S} w_h^t(x) + \sum_{(\theta,\Gamma,n) \in T} N \cdot w_h^t((\theta,\Gamma,N)),\ h = 1,\ldots,k,\ \ N = \sum_{h=1}^{k} N(h).$$

The value of $N(h)$ is the total portion of the database processed so far having membership in population h. The Gaussian parameters are updated as follows: $w_h^{t+1} = \frac{N(h)}{N}$,

$$\mu_h^{t+1} = \frac{1}{N(h)} \left[\sum_{x \in R^{j+1}} w_h^t(x) \cdot x + \sum_{(\theta,\Gamma,N) \in C^j \cup B^j} N \cdot w_h^t((\theta,\Gamma,N)) \cdot \theta \right],$$

$$\Sigma_h^{t+1} = \frac{1}{N(h)}\left[\left(\sum_{x\in R^{j+1}} w_h^t(x)\cdot\left(x\cdot x^{\mathrm{T}}\right) + \sum_{(\theta,\Gamma,N)\in C^j\cup B^j} N\cdot w_h^t\left((\theta,\Gamma,N)\right)\cdot\Gamma\right) - \right.$$

$$\left.\frac{1}{N(h)}\left[\left(\sum_{x\in R^{j+1}} w_h^t(x)\cdot x\right)\left(\sum_{x\in R^{j+1}} w_h^t(x)\cdot x\right)^{\mathrm{T}} + \sum_{(\theta,\Gamma,N)\in C^j\cup B^j} N\cdot w_h^t\left((\theta,\Gamma,N)\right)\left(\theta\cdot\theta^{\mathrm{T}}\right)\right]\right].$$

$h = 1,\ldots,k.$

Stopping Criteria: If $\left|\hat{L}\left(\Phi^t, R^{j+1}, C^j, D^j\right) - \hat{L}\left(\Phi^{t+1}, R^{j+1}, C^j, D^j\right)\right| \le \varepsilon$, terminate (i.e. if the log-likelihood over the contents of the buffer $R^{j+1} \cup C^j \cup B^j$ is sufficiently close, terminate). Otherwise set $t \leftarrow t+1$, go to 1. $\hat{L}$ is given in (7) above.

In addition to the generalization to work over sufficient statistics, ExEM also has the following post processing step.

Model Reset for Empty Clusters: Upon termination, the ExEM algorithm employs a step that performs a Gaussian distribution reset if one or more of the clusters has a sum total 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-type 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. The Gaussian distribution for each "inactive" cluster h_{empty} is reset to the mean of a summarized subset which is least likely under the current mixture model. Cluster h_{empty} also inherits the corresponding covariance matrix of the summarized subset chosen above. With the reseeded cluster, re-run ExEM.

2.3 Data Summarization

Data summarization involves two phases. *Primary summarization* occurs near the modes of the k component Gaussians and produces the set B. *Secondary data summarization* occurs in "dense" regions of data not near the k component Gaussian modes and produces the set C. After summarization, the mixture model update is no longer *exact* (vs. EM updating over the full database). However, by accurately determining regions to summarize in the primary and secondary phases, we aim to minimize future update error when constrained to the single data scan and limited memory buffer. A subset of data is summarized only if the resulting sufficient statistics require less storage than the subset.

2.3.1 Primary Data Summarization

If the current mixture model is accurate, areas near the component Gaussian means μ_h will have the greatest density. The full mixture model can be sufficiently updated by locally modeling these regions with the sufficient statistics triples (θ,Γ,N). The basic intuition is to discard items that are not likely to change degree of membership in clusters. One approach performs sensitivity analysis computations over bounding regions where model parameters are expected to be confined as future data is encountered. Another approach is a simple approximation – which in practice works well. Data modeled well under a cluster (probability close to 1.0 w.r.t. this cluster) is unlikely to change membership or influence other model components. We identify these regions by thresholding the Mahalanobis radius [DH73], $MDist(x,\mu,\Sigma)=\sqrt{(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)}$ near the k component Gaussian means and summarizing data points within this radius. For each component $h=1,\ldots,k$, a Mahalanobis radius r^h is determined so that p% of the data in the retained set R^{j+1} is within this radius. This subset of records then enters $B^{j+1}(h)=\left\{x\in R^{j+1}\,\middle|\,MDist(x,\mu_h^{j+1},\Sigma_h^{j+1})\le r^h\right\}$. The statistics summarizing $B^{j+1}(h)$ are merged with the statistics summarizing points near μ_h in previous iterations. Note that p% of the data is summarized in the primary phase by this thresholding method, and the entire set B^{j+1} contains the sufficient statistics for each set $B^{j+1}(h)$, $h = 1,\ldots,k$. The data records that are summarized are removed from the set R^{j+1}. Figure 2.1 illustrates an example with 3 Gaussians in 2 dimensions. The shaded area indicates regions where data is summarized in the primary phase near the 3 Gaussian means.

2.3.2 Secondary Data Summarization

Secondary data summarization identifies sub-clusters of points among data not summarized in the primary phase that contribute to clusters as a group versus individually. Secondary data summarization has two parts: 1) locate candidate subsets over the singleton records in the memory buffer R^{j+1} not summarized in the primary phase, and 2) determine if these candidate subsets pass a criterion for being treated as a unit. We employ the approximation that a candidate subset with a sufficiently small covariance updates the clusters as if all points where at the subset mean. This can be shown to be valid when a data set is in the tail region of a Gaussian. Candidate regions are determined by applying the standard K-Mean algorithm [F90, DH73] to the current set of singleton data elements, with data summarized in the primary phase removed. To increase the likelihood of finding "dense" candidates, K-Mean searches for a large number of candidate clusters $k' > k$ and is initialized by randomly selecting k' singleton data elements from those

remaining in the buffer after the primary summarization phase. K-Mean (or "harsh" EM [NH99]) is employed since hard subset memberships are required to summarize elements. Once *k'* sub-clusters are determined, the criterion requiring all the sub-cluster covariances be bounded by a threshold β is applied: For a subset locally modeled by a Gaussian with (θ,Γ,N), if:

$$\max_{t=1,\ldots,d}\left\{\sqrt{\frac{1}{d}\cdot\left[(\Gamma)_{tt}-\frac{1}{n}(\theta)_t^2\right]}\right\}\le\beta, \qquad \textbf{(8)}$$

then the subset is deemed satisfactory. Suppose $k'' \le k'$ sub-clusters satisfy this criteria. The sufficient statistics for these *k''* sub-clusters are appended to from the updated set C^{j+1}: $C^{j+1}\leftarrow C^{j}\cup\{(\theta^1,\Gamma^1,N^1),\ldots,(\theta^{k''},\Gamma^{k''},N^{k''})\}$. The elements of C^{j+1} are then merged via hierarchical agglomerative clustering [DH73]. The nearest two sub-clusters are determined, and merged if their merge results in a new sub-cluster that does not violate the criterion above, the merged sub-cluster is kept and the smaller ones removed. The shaded area of Figure 2.2 indicates the dense regions to be summarized in the secondary summarization phase after the data summarized in the primary phase (see Figure 2.1) has been removed.

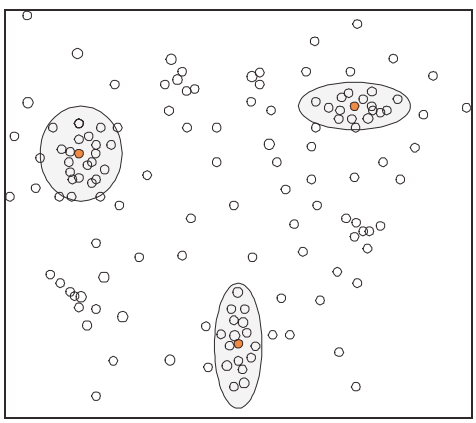

Figure 2.1: Primary data summarization.

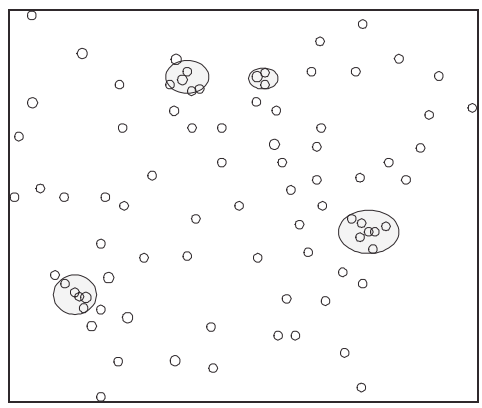

Figure 2.2: Secondary data summarization.

3 Generalization: Exploring Multiple Models in a One Scan

The scheme presented involves updating a single model over a database. However, the summarization machinery also admits the possibility of updating multiple models simultaneously, within a single data scan. The key insights for this generalization are: 1) retained singleton points in R and the sufficient statistics in the set C (representing local dense structures) are shared among all models; 2) each model, M_i, $i = 1,\ldots,\nu$ will have its own discarded data set B_i (ν models = $\nu\times k$ discard sets); 3) the sufficient statistics for discarded data sets B_i for one of the models M_i are simply viewed as members of the global set C by all models other than M_i.

The overall algorithm remains the same, except that model updating and data summarization steps are now performed over multiple models. In addtion to the 3 observations above, there is one data summarization item worthy of further discussion. The algorithm decides on an individual data point basis which discard set fits it best. A data point that qualifies as a discard item (satisfies the criteria to be summarized in the primary phase, see Section 2.3.1) for two models simply goes to the discard set of the nearest model compoent (measured by Mahalanobis metric). A data point cannot be allowed to enter more than one discard set as it would be multiply accounted for in the update loop. Let x qualify as a discard item w.r.t. a cluster of model M_1 and a cluster of M_2. If it were admitted to both, then model M_1 will "feel" the effect of this point twice: once in its own discard set B_1 and another time when it updates over B_2 which will be treated as part of C from the "viewpoint" of M_1. Similarly for M_2. By entering in exactly one discard set, the point x is correctly accounted for and correctly influences both models.

It is worthy of note that since the multiple models may "communicate" with each other via the implicit coupling through the sets B and C. Empirically, it seems that this interaction helps models recover from poor local minima. If a particular cluster falls towards a poor local minima, a nearby component from another model may pull it from this location during the update over C. We empirically observe this desirable behavior when running in multiple model mode and results, typically, in better overall mixture models.

A weakness of our scheme is that the data summarization, once performed, cannot be "undone" (in the single scan framework). However, it turns out that the error resulting from summarization of data to sufficient statistics introduces over future updates to the mixture model is more than made up for by the fact that the algorithm gets to see much more data than the sampling alternatives. We next demonstrate this next.

4 Experimental 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] for the problem of initial models) 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] are based). 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.) Hence it is not possible to compare with such algorithms experimentally (see Section 5). Here we compare with alternative algorithms on both synthetic data (where we know what the real solution should be) and on a few real data sets.

4.1 Synthetic Data

Evaluations over synthetically generated data target scalability and mixture model quality of the proposed approach. For all experiments *except* those studying sensitivity to parameters, scalable EM parameters were fixed as: primary summarization threshold p = 0.5, secondary summarization standard deviation threshold β = 0.5 and the number of secondary candidates k' = $2k$ where k is the number of mixture model components. Hence the parameters where not tuned to any of the data sets. Parameter sensitivity results are given in Section 4.2.

4.1.1 Scalability

The first experiment is intended to simply illustrate scalability behavior. Synthetic data was generated by a mixture of 10 Gaussian distributions in 25 dimensions. Scalable EM (Algorithm 2) was applied to data sets where the numbers of rows (points) were varied from 10,000 to 1 million. Scalable EM (SEM) was applied with the buffer size equivalent to storage needed for 5% of the total dataset. Ten mixture models were simultaneously updated. Initial mixture model means were chosen by randomly selecting 10 data points and initial covariance matrices were the identity matrix. Running times for SEM and standard (vanilla) EM (Algorithm 1) applied to the given data sets are summarized in Figure 4.1. SEM scales linearly while standard EM begins to utilize virtual memory and page to disk when the number of points reaches 600,000. This experiment was executed on a PII-400 Pentium workstation with 128 MB RAM.

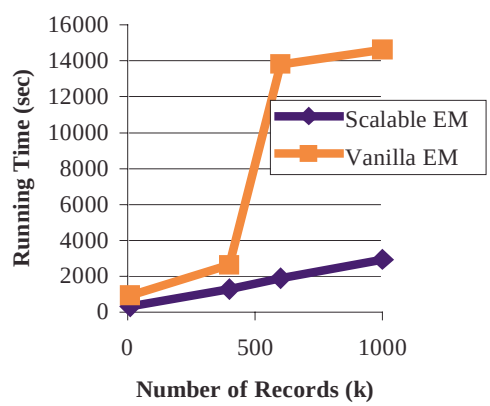

Figure 4.1: Running time versus number of records

Note that despite the summarization and data triage overhead, SEM runs faster than standard EM, even over datasets that fit into main memory. This seemingly puzzling fact is explained by an

unintended desirable behavior: SEM has a much tighter loop over data (local to the buffer), hence it implicitly makes much better use of the on-CPU memory cache. Updating over the entire data is much more likely to cause a cache fault. Updating over cache memory is significantly faster as it avoids memory bus accesses off the CPU. Because of the large jump in run time once 'vanilla EM' starts paging to disk, the graph underemphasize the speed up factor for small data set sizes. The scalable EM is actually running at least 3 times faster even when standard (vanilla) EM can fit all the data in main memory. A very desirable property. In the next sections, we will demonstrate that the speedups are obtained with no sacrifice in quality of obtained model.

4.1.2 Mixture Model Quality

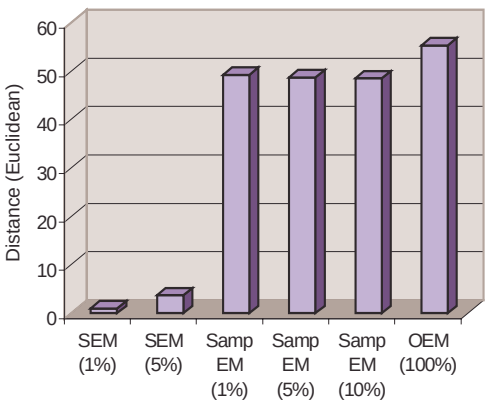

Figure 4.2: Quality of solutions (20k records, 20 dim.)

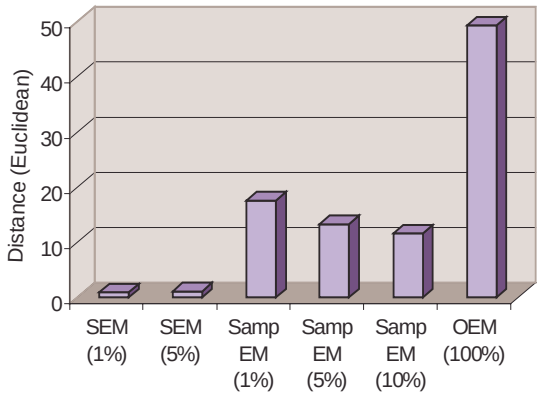

Figure 4.3: Quality of solutions (50k records, 50 dim.)

The first investigation of cluster quality focused on recovery of the means of a true mixture model used to generate synthetic data. The data was created for a number of attributes d = 20, 50 and 100 and sampled from a Gaussian mixture model with k = 5, 10 and 100 Gaussian clusters, respectively. Cluster means m were sampled from a uniform distribution on [-5,5]. Elements of the diagonal covariance matrix S were sampled from a uniform distribution on [0.7,1.5]. Once the means and covariance matrices were determined, 20k records were sampled from the model with d = 20 and k = 5. The model with d = 50 and k = 10 was used to generate 50k records and the model with d = 100 and k = 100 was used to generate 100k records. The quality of the clustering solution is quantified by measuring distance from the estimated population means to the true Gaussian means generating the synthetic data (after performing an optimal "matching").

We compare solutions obtained by SEM (Algorithm 3) and the following methods:

- SampEM: Standard EM (Algorithm 1) is executed over a random database sample.
- 1EM: "Online" EM algorithm. This is the simplest variant is an incremental version [NH98] of EM in which one data record is scanned, the mixture model is updated,

another data record is scanned, and so forth updating the model over a single sample at a time. 1EM is limited to a single data scan in this case in order to say within the framework of interest.

Figures 4.2 and 4.3 summarize ratios of 2-norm (Euclidean) distance from the estimated population means to the true Gaussian means (percentage in parentheses following the method indicates the memory buffer relative full dataset storage). The distance ratios are with respect to the best population mixture model – that with minimum distance (scalable EM with 1% sample size both synthetic datasets). Results represent averages over 10 mixture models initialized randomly. The sampling solution (Samp EM) is given a chance to sample 10 different random samples from the population. Hence random sampler results are over 100 total trials. Scalable EM (SEM) is given a buffer size given as a percentage of size of data set. Results on the 100-*d* data sets with *k*=100 are similar to 20-*d* and 50-*d*. In terms of distance to true Gaussian means SEM(1%) was best with average distance twice as bad for the sampling-based EM. 1EM did much worse on this set.

Mixture model quality was also quantified by the average log-likelihood of the data given the mixture model. SEM is compared with SampEM and 1EM (described above). SEM was also compared with a batch variant of an incremental EM (BEM) implementation in which the memory buffer allocated to the clustering process is filled with a random sample from the data set, the mixture model is updated over the contents of the buffer via the standard EM algorithm, then the buffer is purged and filled with another sample. This process is repeated until the entire data set has been scanned one time. BEM is hence similar to our method except that it does not keep summary statistics beyond the model.

Results are summarized in Table 4.1 and Figure

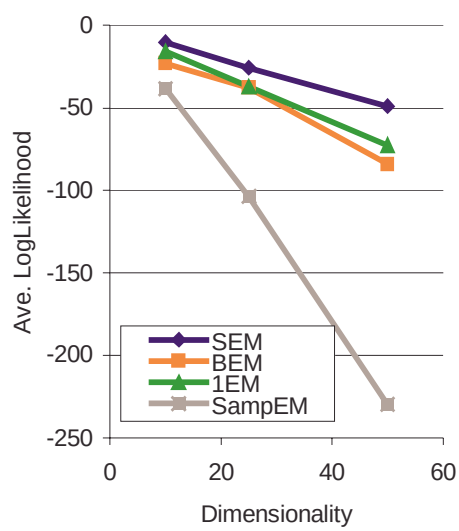

Figure 4.4: Cluster quality vs dim.

Table 4.1: Cluster quality vs. dimensionality *n* (best in bold).

Dimensionality *d*	SEM	BEM	1EM	SampEM
10	**-10.3**± 0.3	-23.0± 3.2	-16.0± 0.0	-38.4± 5.0
25	**-25.6**± 2.2	-37.6± 2.0	-37.3± 0.0	-103.7± 13
50	**-49.0**± 4.8	-84.1± 8.2	-72.7± 0.0	-229.7± 19

4.4 for data sets with 50k points and dimensionality d = 10, 25 and 50, synthetically generated by a mixture model with 10 Gaussian clusters. Average log-likelihood of the data given the mixture model is determined for the 10 mixture models computed via SEM and the 10 best mixture models computed via SampEM, BEM and 1EM. Entries in Table 1 are averages over 10 mixture models plus/minus one standard deviation. As dimensionality increases, quality of mixture models for all methods decreases, but SEM mixture model quality degrades more gracefully than SampEM, BEM or 1EM mixture model quality.

For fixed data set dimensionality d = 25, we varied the number of data points m = 10k, 50k and 100k generated by a Gaussian mixture model with 10 clusters. As the number of data points increases, each approach improves as expected, but SEM computes models superior to SampEM (average log-likelihood improvement as much as 32% for m = 50k) and 1EM (average log-likelihood improvement as much as 36% for m = 100k).

4.2 Parameter Sensitivity

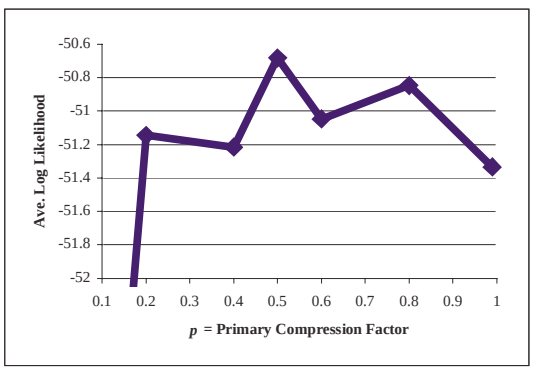

Figure 4.5: Quality versus primary summarization factor.

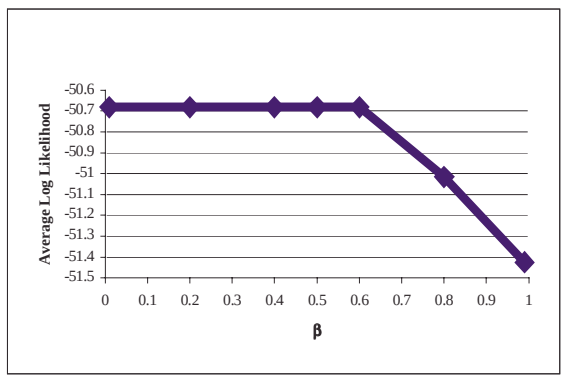

Figure 4.6: Quality versus standard tolerance

4.2.1 Varying Primary Summarization Factor

(See Section 2.3.1)**:** Evaluations of cluster quality for various values of the primary summarization fraction p were conducted over a synthetic database with 50k records and 50 dimensions. See Figure 4.5. Secondary summarization was not used by SEM in these tests. For small values of p (near 0.0), there is little or no summarization, the memory buffer fills and SEM terminates 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 summarized 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.

4.2.2 Varying Standard Tolerance

(See Section 2.3.2): Evaluations of cluster quality for various values of β were conducted. The primary summarization factor p was set to 0.5 (default), the maximum buffer size was 100kB. The results are shown in Figure 4.6. We allowed SEM to construct large candidate secondary sub-clusters to better study the effects of β. Figure 4.3 shows that cluster quality remains high for values of β to 0.8. For small values of β (near 0.0), secondary summarization is minimal and hence there is no loss of data resolution. As β is increased, secondary summarization plays a larger role. For values of β in [0.2,0.6], when summarization 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.

4.3 Real World Data

We compared SEM against the other variants: 1EM, BEM, and SampEM (see Section 4.1.2). SEM has three major parameters: primary summarization factor p (Section 2.3.1), standard tolerance β (Section 3.3.2), and number of secondary clusters k' (Section 2.3.2). Throughout all experiments we set these parameters to constant values: p = 0.5, β = 0.5(maximum global data variance), k' = 2k. We showed sensitivity to these parameters in Section 4.2 where we explicitly varied them.

SEM, BEM, 1EM and SampEM were applied to a subset of the **US Census "Adult"** data set (www.sgi.com/Technology/mlc/db/) consisting of 299,285 data points with 11 continuous dimensions. SEM, BEM and SampEM were allocated buffer sizes equivalent to storage of 0.5% and 0.1% of the data set. The results showing average log likelihood plus/minus one standard deviation for the 10 Gaussian mixture models estimated by SEM and the 10 best mixture model computed via BEM, 1EM and SampEM are as follows:

SEM 0.5%	SEM 0.1%	BEM 0.5%	BEM 0.1%	1EM	SampEM 0.5%	SampEM 0.1%
3.4 ± 3.3	2.3 ± 2.4	1.0 ± 0.9	-7.0 ± 6.3	-28.4 ± 0.0	-13483	-26055

Standard deviation is not reported for SampEM as results are orders of magnitude worse than other approaches. The best mixture model was computed via SEM with the 0.5% buffer

The algorithms were applied to a subset of the **Reuters text classification database** (www.research.att.com/~lewis/reuters21578/) consisting of word counts for the 302 most frequently occurring words for 12,902 documents. SEM, BEM and SampEM were allocated

buffer sizes equivalent to 10% and 5% total data set storage. The table below shows averages over 10 mixture models for SEM, 1EM and SampEM. BEM results are averaged over the 4 best mixture models (all others had average log-likelihood at least an order of magnitude smaller). SEM with the larger buffer size computed the best Gaussian mixture models with 10 components. Simple sampling (SampEM) produces extremely poor results in this case.

SEM 10%	SEM 5%	BEM 10%	BEM 5%	1EM	SampEM 10%	SampEM 5%
-39 ± 40	-216 ± 40	-425 ± 883	-375 ± 167	-396 ± 1	-590188	-910364

The **REV Digits Recognition** dataset consists of 13,711 data items with 64 continuous dimensions. Each record represents the gray-scale level of an 8x8 image of a handwritten digit. The algorithms were allocated buffer sizes of 10% and 5% total data set storage size. The results here were actually statistically indistinguishable amongst the various methods, except for 1EM which produced significantly worse models whose log likelihood was 1.5 times worse on average. The fact that SEM did as well here is explained by the fact that number of dimensions to number of rows ratio here is very high, hence all algorithms are really observing a small sample anyway.

Finally, results are summarized below for the **Astronomy** database of measurements of sky objects from the Second Palomar Sky Survey at Caltech. The database consists of 648,291 records in ***R***[29]. Results are averaged over the best 10 models computed by the respective method. In this case, surprisingly, 1EM produces the best mixture models. Our hypothesis to explain this is that this data set has so many records that multiple iterations of 1EM are 'simulated' within its single pass. However, we are surprised at this result. Generally 1EM exhibits high variance in its performance over various databases.

SEM 0.1%	BEM 0.1%	1EM	SampEM 0.1%
-77 ± 79	-433 ± 153	**-6** ± 1	-367 ± 63

Our experience with 1EM (as evidenced by synthetic and other empirical results) is that it does much worse than alternatives. However, the result on this data set suggests that, since 1EM is so cheap to compute, that in the scalable implementation it be computed on the side for an additional isolated model. At the end of the clustering, the algorithm can compare results and select the best model.

5 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 probabilistic cluster assignment step (Algorithm 1, step 1) 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 no sufficient statistics are kept other than in the clusters themselves (as in regular EM). Our method introduces the notion of secondary summarization and the set *C*. 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 results in this paper comparing with OEM confirm this.

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 summarize 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 summary once the first pass terminates. The fundamental difference between BIRCH and the scalable EM algorithm lies in the approach to data summarization. BIRCH summarizes data as a step independent of the clustering algorithm whereas the scalable EM algorithm's summarization 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 [ZRL99]. However, the data being clustered must be continuous with a distance metric defined. Because we operate on probability distributions, our method readily admits discrete attributes (modeled, say, as multinomials), and a mixture of discrete and continuous attributes. In addition, density estimates obtained using kernel density estimation are much more difficult to interpret. Clustering with a mixture model to identify K populations in the data is an established and trusted tool for data reduction , summarization, and descriptive modeling in the statistics literature [B95].

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. The problem of interest in this paper is how to best model a large data set with a mixture distribution. Once the distribution is obtained, standard statistics can then be leveraged to do all sorts of probabilistic inference, including applications in indexing [BFG99] and compressing data cubes for OLAP [SFB99].

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 bounding 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 summarization 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.

6 Conclusion

The algorithm presented easily scales to very large databases. The memory requirement is simply to hold a small sub-sample in RAM. All updates occur 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 models concurrently typically results in better solutions and improved summarization since the *synergy between the models* explored adds more opportunity for summarization. In turn more space is available in the buffer and the algorithm maximizes its exposure to new data during model updates. The advantages come

primarily from the ability to explore multiple models simultaneously (with inter-model interactions) and from the ability to avoid bad local minima with empty clusters. Detailed results on these effects cannot be shown due to space constraints.

The results on both synthetic data and 4 real data sets indicate that the proposed scalable scheme produces good solutions and outperforms the well-known and current practice methods for scaling this method: sampling data or on-line algorithms. It also outperforms variants on sampling and on-line EM (1EM) where the process is batched over consecutive samples (BEM). Hence even though both algorithms see all the data, our added data triage and summarization stages preserve important information. The retained data vectors/summarized clusters do indeed lead to much better solutions when compared with a memoryless (other than clusters) version of on-line 1 case-at-a-time EM implementation.

Acknowledgements

We thank Daphne Koller for suggesting the batch variant of incremental EM experiments.

7 References

[A73] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, New York, 1973

[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*, 49: 803-821, 1993.

[BFG99] K. P. Bennett, U. M. Fayyad, and D. Geiger. Density-Based Indexing for Approximate Nearest-Neighbor Queries. In C. Chaudhuri and D. Madigan (eds.), *Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining (KDD99)*, pp. 233-243, ACM Press, New York, 1999.

[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*. John Wiley & Sons, New York, 1973.

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

[FPSU96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (eds.). *Advances in Knowledge Discovery and Data Mining*. MIT Press, Menlo Park, CA, 1996.

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

[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. *Finding Groups in Data*. John Wiley & Sons, New York, 1989.

[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*. John Wiley & Sons, New York, 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.

[SFB99] J. Shanmugasundaram, U. M. Fayyad and P. S. Bradley. Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. In S. Chaudhuri and D. Madigan (eds.), *Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining (KDD99)*, pp. 223-232, ACM Press, New York, 1999.

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

[S96] P. Smyth. Clustering using Monte Carlo Cross-Validation. In *Proc. 2nd Intl. 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):141-182, 1997.

[ZRL99] T. Zhang, R. Ramakrishnan, and M. Livny. Fast Density Estimation Using CF-kernel for Very Large Databases. In S. Chaudhuri and D. Madigan (eds.), *Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining (KDD99)*, pp. 312-316, ACM Press, New York, 1999.