Machine Learning, 45, 279–299, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

Accelerating EM for Large Databases

BO THIESSON thiesson@microsoft.com
CHRISTOPHER MEEK meek@microsoft.com
DAVID HECKERMAN heckerma@microsoft.com
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

Editor: Lawrence Saul

Abstract. The EM algorithm is a popular method for parameter estimation in a variety of problems involving missing data. However, the EM algorithm often requires significant computational resources and has been dismissed as impractical for large databases. We present two approaches that significantly reduce the computational cost of applying the EM algorithm to databases with a large number of cases, including databases with large dimensionality. Both approaches are based on partial E-steps for which we can use the results of Neal and Hinton (In Jordan, M. (Ed.), *Learning in Graphical Models*, pp. 355–371. The Netherlands: Kluwer Academic Publishers) to obtain the standard convergence guarantees of EM. The first approach is a version of the incremental EM algorithm, described in Neal and Hinton (1998), which cycles through data cases in blocks. The number of cases in each block dramatically effects the efficiency of the algorithm. We provide a method for selecting a near optimal block size. The second approach, which we call lazy EM, will, at scheduled iterations, evaluate the significance of each data case and then proceed for several iterations actively using only the significant cases. We demonstrate that both methods can significantly reduce computational costs through their application to high-dimensional real-world and synthetic mixture modeling problems for large databases.

Keywords: Expectation Maximization algorithm, incremental EM, lazy EM, online EM, data blocking, mixture models, clustering

1. Introduction

The EM algorithm has been the object of considerable interest since its presentation in general form by Dempster, Laird, and Rubin (1977). It is a widely applicable, and remarkably simple algorithm for computing maximum likelihood estimates for parameters in incomplete data models. By alternating between an expectation step (E-step), which finds expected completions of data given the current parameterization, and a maximization step (M-step), which re-estimates parameters on the basis of completed data, the EM algorithm gradually improves the likelihood for the observed data until convergence at a local maximum (or saddle point).

Unlike various other numerical techniques for optimization, the EM algorithm guarantees convergence. A common criticism, however, is that convergence can be slow—see comments by several discussants of Dempster, Laird, and Rubin (1977). For this reason, various ways of accelerating the EM algorithm have been suggested.

A partial M-step may accelerate the algorithm in situations where the M-step is computationally inefficient. There are a variety of partial M-step methods that have the same convergence guarantees as the standard EM algorithm. These methods can usually be cast as variants of the generalized EM (GEM) algorithm, described in Dempster, Laird, and Rubin (1977), which in the M-step only improves rather than maximize the expected complete data log-likelihood. In particular, the Expectation Conditional Maximization (ECM) algorithms (Meng & Rubin, 1993) with further generalizations in Meng and van Dyk (1997) are well-known examples of such accelerations. A variety of other approaches to accelerating EM, including the replacement of the M-step with a faster (conjugate) gradient or (quasi) Newton type step, have also been considered in the literature. See (e.g.) Jamshidian and Jennrich (1993) and Thiesson (1995) for examples of the former, and Louis (1992) and Meilijson (1989) for examples of the latter.

Because the time spent in the E-step depends linearly on the number of cases, the above described acceleration methods do not address critics of the EM algorithm who suggest that it is impractical for application to large databases. See (e.g.) Zhang, Ramakrishnan, and Livny (1996). In this paper, we instead provide strong empirical evidence that the use of partial E-steps, as suggested in Neal and Hinton (1998), can significantly reduce the cost of applying the EM algorithm to large real-world databases without losing the guaranteed convergence to a local maximum of the likelihood function.

Other approaches for reducing the cost of applying EM by reducing the time spent in the E-step have been considered in Moore (1999), McCallum, Nigam, and Ungar (2000), and Bradley, Fayyad, and Reina (1998) for finite mixture model settings. None of the methods (as described by the authors) maintain the desirable convergence guarantees of EM, but will instead approximate a solution. Furthermore, whereas the methods in Moore (1999) show great promise for domains with a small number of variables, the benefits diminish significantly as the number of variables increase. The method in McCallum, Nigam, and Ungar (2000) is closely related to Moore's method, but can potentially provide significant speedup for high-dimensional data sets. The method requires an inexpensive way of creating an initial crude partition of the data into overlapping subsets of similar cases called canopies. The speedup obtained by this method depends on the quality of the initial partition into canopies. When the initial partition yields well-separated canopies the speedups can be significant. In this paper, we examine data sets with dimensionality larger than those of Moore (1999) and without requiring the existence of a method to provide an initial crude clustering of the data set.

The two methods that we investigate are remarkably easy to implement and provide significant computational benefit. The first method is a version of the incremental EM algorithm from Neal and Hinton (1998). The method partitions data into blocks which it traverses in a cyclic manner while incrementally updating the expected complete data log-likelihood and parameters after visiting each of the blocks. We can expect an improved rate of convergence for this algorithm over standard EM due to the fact that information contained in a block of data is exploited more quickly. In particular, the incremental EM algorithm immediately incorporates the information obtained from a block of data into the parameterization used to update the expected complete data log-likelihood associated with the following block. In contrast, the standard EM algorithm updates the parameterization only after the expected complete data log-likelihood for the entire data set is updated.

We discuss several implementation issues for the incremental EM, which are not discussed by Neal and Hinton (1998), regarding partitioning of data into blocks of cases, convergence, and the initial pass through the data. In particular, the partitioning of data into blocks of proper size is important for the acceleration. Based on experimental results, we provide a method to select a block size that results in near optimal acceleration. One interesting fact is that it is far from optimal to traverse data cases one-by-one; the optimal block size is much larger. This suggests that the similar online EM algorithms (see, for instance, Nowlan, 1991 or Sato, 1999) may also benefit from blocking data before processing them.

The second method that we investigate, the lazy EM algorithm, also accelerates the EM algorithm on the basis of partial E-steps. As opposed to incremental EM, it does not traverse statically defined blocks of data. At scheduled iterations, the algorithm selects a subset of data cases upon which to perform several successive partial E- and M-steps. For this method, we address issues such as how the significant data cases are to be selected and how long they should be used before re-selection takes place.

We provide empirical results for both of the above methods on two real-world mixture modeling problems and on mixture modeling problems for synthetic data. We show that both methods can significantly reduce the computational cost of the EM algorithm without degrading the resulting solution. It should be noted that for one of the real-world problems, *MSNBC* (see Section 4), the data is extremely sparse. Chickering and Heckerman (1999) consider a method for speeding up the E-step for finite mixture models with sparse data. They report a significant speedup while maintaining the convergence guarantees of EM. We exploit their method for the *MSNBC* experiments and gain additional computational efficiency by applying the partial E-step methods described in this paper.

We emphasize that although experiments are restricted to finite mixture models, the suggested acceleration methods are generally applicable for any setting where the standard EM algorithm is applicable.

2. ML estimation methods for incomplete data

In this paper, we consider statistical models for variables X, that is, families of distributions $p(X \mid \theta)$ parameterized by $\theta \in \Theta$. We define $X = (X_1, \ldots, X_N)$ where N is the number of cases in the data that we are considering and X_i is a set of variables describing a case. We use uppercase roman letters to denote variables (e.g., X), and lowercase roman letters to denote values of these variables (e.g., x). Suppose we wish to find the ML parameterization for this model given incomplete data $y \in Y \subseteq X$. Let $z \in Z = X \setminus Y$ denote an arbitrary value for the unobserved variables, then $x = (y, z)$ will be referred to as completed data for y, and $x_i = (y_i, z_i)$ referred to as a completion for the observed incomplete case y_i. We assume that the observed data is incomplete in an uninformative way—that is, missing completely at random. Hence, the log-likelihood for the observed data satisfies

$$l(\theta \mid y) = \log p(y \mid \theta) = \log \int p(y, z \mid \theta)\, dz \tag{1}$$

When it is hard to maximize the log-likelihood in (1) directly, but easy to work with the complete data log-likelihood $l(\theta \mid x) = \log p(x \mid \theta)$, the EM algorithm can be use to solve the maximization problem iteratively. In what follows, we describe the EM algorithm and the partial E-step variants that we consider in our experiments.

2.1. *The EM algorithm*

Roughly speaking, the EM algorithm converts the ML estimation problem into a sequence of "pseudo-estimations" with respect to the statistical model for complete data. Let θ^n denote the current value of θ after n iterations. Each iteration of the EM algorithm involves two steps. The nth iteration of the algorithm is as follows; in the E-step, obtain the distribution for X given current parameterization and observed data, $\tilde{p}^n = p(X \mid y, \theta^n)$ and use it to construct the conditional expectation for the complete data log-likelihood $Q(\theta \mid \theta^n, y) = E_{\tilde{p}^n}[l(\theta \mid X)]$. In the M-step, choose θ^{n+1} as the value $\theta \in \Theta$ that maximizes $Q(\theta \mid \theta^n, y)$. In the case where the statistical model is a subfamily of an exponential family, the EM algorithm becomes an alternation between an E-step that computes expected sufficients for the statistical model and an M-step that re-estimates the parameters of the model by treating the expected sufficient statistics as if they were actual sufficient statistics (Dempster, Laird, & Rubin, 1977).

2.2. *The incremental EM algorithm*

The incremental EM algorithm attempts to reduce the computational cost by performing partial E-steps. Let $y = \{y_1, \ldots, y_K\}$ denote a particular partition of the data into mutually disjoint blocks of data cases. Observe that, in this subsection, we abuse the notation in that a subscript refers to a block of data cases rather than an individual case. The incremental EM algorithm iterates through the blocks in a cyclic way. Each iteration performs a partial E-step by updating only a part of the conditional expectation for the complete data log-likelihood (the Q-function) before performing a M-step. The nth iteration of the algorithm is sketched below.

```
E-step: Select block of data $y_i$, where $i = n$ modulus $K$
        Obtain $\tilde{p}_i^n = p(X_i \mid y_i, \theta^n)$
        Set $Q_j(\theta \mid \theta^n, y_j) = Q_j(\theta \mid \theta^{n-1}, y_j)$ for $j \neq i$
        Compute $Q_j(\theta \mid \theta^n, y_j) = E_{\tilde{p}_j^n}[l(\theta \mid X_j)]$ for $j = i$
        Construct $Q(\theta \mid \theta^n, y) = \sum_j Q_j(\theta \mid \theta^n, y_j)$
M-step: Choose $\theta^{n+1}$ as the value $\theta \in \Theta$ that maximizes $Q(\theta \mid \theta^n, y)$.
```

Notice the way in which the E-step incrementally constructs the Q-function to be maximized. In each iteration, the algorithm only computes a fraction of the Q-function under consideration, namely the Q_i associated with the block of data y_i. For all other data blocks, the algorithm reuses previously computed contributions to the Q-function. In an efficient implementation, we incrementally update the Q-function by adding the difference between the new and old Q_i components:

$$Q(\theta \mid \theta^n, y) = Q(\theta \mid \theta^{n-1}, y) + Q_i(\theta \mid \theta^n, y_i) - Q_i(\theta \mid \theta^{n-1}, y_i)$$

Note that this algorithm has an additional cost beyond EM, which is the storage of Q_i for all blocks $i = 1, \ldots, K$.

As in the EM algorithm, if the statistical model is a subfamily of an exponential family, then the E-step can be cast as constructing expected sufficient statistics for the statistical model.

The theoretical justification for the incremental EM algorithm is due to Neal and Hinton (1998). They cast the algorithm into a method for maximizing the function

$$F(\tilde{p}, \theta) = \sum_i F_i(\tilde{p}_i, \theta) = \sum_i \big(E_{\tilde{p}_i}[\log p(X_i \mid \theta)] + H(\tilde{p}_i)\big) \tag{2}$$

where i indexes blocks, $H(\tilde{p}_i) = -E_{\tilde{p}_i}[\log \tilde{p}_i]$ is the entropy of $\tilde{p}_i$, $\tilde{p} = \prod_i \tilde{p}_i$, and $\tilde{p}_i = p(X_i, y_i)$ is a distribution over possible values of X_i. This distribution has no support for those values of X_i not compatible with the observed block of data y_i. That is, $\tilde{p}_i = 0$ for $\{x_i \in X_i : x_i \backslash z_i \neq y_i\}$. Neal and Hinton (1998) show that both the E- and M-step of the incremental EM algorithm monotonically increase $F(\tilde{p}, \theta)$ until convergence at $(\tilde{p}^*, \theta^*)$, where θ^* is a local maximum (or saddle point) for the log-likelihood in Eq. (1). A key step in the proof is to show that given $\theta = \theta^n$, the E-step maximizes $F_i(\tilde{p}_i, \theta)$ with respect to $\tilde{p}_i$ by setting $\tilde{p}_i = p(X_i \mid y_i, \theta^n)$ (for more details, see Lemma 1 of Neal & Hinton, 1998). For the remaining blocks, the term $F_j(\tilde{p}_j, \theta)$ $(j \neq i)$ is unchanged because the E-step only affects the term associated with the current block. Thus, an E-step increases F by improving $\tilde{p}_i$. Showing that the M-step increases F is straightforward. Given fixed $\tilde{p}_i$'s, as obtained in the E-step, the M-step improves θ by maximizing the Q-function, defined above. Because the sum of expected log-likelihood terms in Eq. (2) is actually the Q-function, and the entropy terms do not depend on θ, the M-step improves F.

Notice from this proof sketch that we may create a generalized incremental EM algorithm in the spirit of generalized EM by improving rather than maximizing the Q-function in the M-step for incremental EM.

The partitioning of the data into blocks in the incremental EM algorithm leads to three implementation issues that do not arise in the standard EM algorithm. The first issue concerns the number of cases to place in each block of data. We will restrict attention to blocks of (approximately) equal size, where cases are blocked together in the order they appear in the data set. Some block sizes will be better than others. We will return to this issue in Section 4, where we provide a near optimal way of selecting the block size.

The second issue is how one should handle convergence. In practice, we do not run an algorithm infinitely to convergence. Instead we run the algorithm until some convergence criterion is satisfied. In the design of convergence tests for incremental EM, we have strived to mimic tests for standard EM—tests based on relative differences in successive log-likelihood values. Recall that K denotes the number of blocks in the partition of data. Note that, in standard EM, $K = 1$. For domains with discrete variables only, we base our convergence criterion on the following:

$$c_K^n = -\frac{l^n - l^{n-K}}{l^n} \tag{3}$$

For domains involving continuous variables, we base our convergence criterion on the following:

$$c_K^n = \frac{l^n - l^{n-K}}{l^n - l^0} \tag{4}$$

where l^0 is the log-likelihood value for the parameterization obtained after initialization (see below). Convergence is said to have taken place at iteration n if c_K^n falls below some *convergence threshold*. We use l^0 in the convergence test for domains involving continuous variables because densities are not restricted to the interval [0, 1]. If we do not subtract l^0 in the denominator, we would experience an unexpected sign change if successive likelihoods straddle the value one. Furthermore, without this subtraction, convergence would be affected by the scales of the continuous variables.

The convergence criteria for standard EM are based on the fact that the log-likelihood values on successive iterations increase monotonically. For incremental EM, current theoretical results guarantee that the F function in Eq. (2) increases monotonically, and at convergence yields a local maximum for the log-likelihood. However, monotonic behavior is not guaranteed for log-likelihood values evaluated at successive iterations. Furthermore, the evaluation of the log-likelihood values would add additional computational cost to the incremental EM algorithm for the following reason. In standard EM the log-likelihood is basically constructed as part of the evaluation of all the data cases in the E-step. This is not the case for incremental EM, because the E-step for this algorithm evaluates only one block of data. Consequently, we use the convergence criteria in Eqs. (3) and (4) with an approximation of the log-likelihood for all data by the incremental update

$$l(\theta^n, y) \approx \tilde{l}(\theta^n, y) = \tilde{l}(\theta^{n-1}, y) - l(\theta^{n-1}, y_i) + l(\theta^n, y_i)$$

where $\tilde{l}$ denotes an approximated log-likelihood. Theoretical results showing monotonic behavior of this approximation are also lacking. Nonetheless, experimental results suggest that sub-sequences of the approximate log-likelihoods, obtained one full pass of the data apart, increase monotonically. In other words, we have found that $l^n > l^{n-K}$ for all $n > K$. Therefore, similar to standard EM, we deem incremental EM to have converged when c_K^n falls below a convergence threshold.

As an alternative to likelihood-based convergence tests, one may use a distance measure between parameter values of successive steps or between parameter values obtained one full pass of the data apart. Another possibility is to base convergence tests on values for the F function in Eq. (2) at successive iterations or successive full passes through the data. Both alternatives add a computational overhead compared to the likelihood-based convergence tests we use. This overhead will, in most practical situations, be insignificant compared to the overall runtime for the algorithm. It will, however, become significant if one chooses to test convergence after each step of the algorithm and the block size is small.

The last implementation issue that we consider is related to the initial pass through the data. This pass can be problematic for small block sizes. As an illustration, consider a two component univariate Gaussian mixture model with the sequence of observations 1, 2, 10, 1, 0, 11. If we apply the incremental EM algorithm with blocks of size two with initial component means near 1 and 10, then the Gaussian with mean near 1 will be supported by the inferred completion $\tilde{p}_i^n = p(X_i \mid y_i, \theta^n)$ for both cases in the first block, whereas the Gaussian with mean near 10 will have practically no support. In fact, if we are using a maximum likelihood approach, the weight of the component with mean near 10 will after the first M-step be equal to 0 (to within the precision of the computer). It is therefore impossible to infer any support for this component in following iterations. This problem of "premature starvation" and other small sample issues can be avoided by waiting several iterations before performing the first M-step. A conservative choice, the one used in our experiments in Section 4, is to perform the first M-step only after completing one full pass through the data.

2.3. *The lazy EM algorithm*

The lazy EM algorithm is based on the assumption that not all data is of equal significance for each iteration. Given data cases $y_1, y_2, \ldots, y_N$, the lazy EM algorithm attempts to periodically identify significant cases and focuses attention on this subset of data for several iterations. Roughly speaking, a case is significant if the change in the inferred completion, $\tilde{p}_i$, between two successive iterations is large. Let $y_{l\bar{a}zy}$ denote the set of data cases assessed as significant and denote the remaining data cases by y_{lazy}. According to a predetermined schedule, each iteration involves either a full or lazy E-step followed by a standard M-step, as illustrated below. A full E-step updates the expected complete data log-likelihood (the Q-function) for all cases in the data set. In addition, it identifies the set of significant data cases to be used in lazy iterations. A lazy E-step only updates the part of the Q-function associated with significant cases. Naturally, the E-step in the first iteration must be scheduled as full.

```
If full E-step is scheduled for iteration $n$:
        Obtain $\tilde{p}^n = p(X \mid y, \theta^n)$
        Identify $y_{lazy}$ the set of data cases to be ignored in lazy iterations.
        Construct $Q(\theta \mid \theta^n, y) = E_{\tilde{p}^n}[l(\theta \mid X)]$
Else lazy E-step is scheduled for iteration $n$:
        Obtain $\tilde{p}^n_{l\bar{a}zy} = p(X_{l\bar{a}zy} \mid y_{l\bar{a}zy}, \theta^n)$
        Set $Q_{lazy}(\theta \mid \theta^n, y_{lazy}) = Q_{lazy}(\theta \mid \theta^{n-1}, y_{lazy})$
        Compute $Q_{l\bar{a}zy}(\theta \mid \theta^n, y_{l\bar{a}zy}) = E_{\tilde{p}^n_{l\bar{a}zy}}[l(\theta \mid X_{l\bar{a}zy})]$
        Construct $Q(\theta \mid \theta^n, y) = Q_{l\bar{a}zy}(\theta \mid \theta^n, y_{l\bar{a}zy}) + Q_{lazy}(\theta \mid \theta^n, y_{lazy})$
M-step: Choose $\theta^{n+1}$ as the value $\theta \in \Theta$ that maximizes $Q(\theta \mid \theta^n, y)$.
```

Similar to the incremental EM algorithm, an efficient implementation updates the Q-function for the lazy E-step in the following way

$$Q(\theta \mid \theta^n, y) = Q(\theta \mid \theta^{n-1}, y) - Q_{l\bar{a}zy}(\theta \mid \theta^{n-1}, y_{l\bar{a}zy}) + Q_{l\bar{a}zy}(\theta \mid \theta^n, y_{l\bar{a}zy})$$

Hence, we obtain computational efficiency for lazy E-steps at the additional cost of storing $Q_{l\bar{a}zy}$ associated with the fraction of data for which we update. Again, when the statistical model is a subset of an exponential family, the E-steps can be cast as computing the expected sufficient statistics.

The viability of the lazy EM algorithm rests in part on the assumption that not all data is of equal importance. It also depends on the computational overhead for assessing significance of individual data cases and the cost of storing the inferred completions needed for assessing significance.

For finite mixture models, we can construct a significance criterion for which the computational overhead can be greatly reduced and the cost of storing inferred completions can be avoided. The intuition behind this criterion is based on the following empirical observation. If a data case is strongly assigned to a particular component in the mixture, it is unlikely to move to another component; and if it does, it will not suddenly jump but rather gradually shift towards the other component. Consequently, we expect that cases that are not strongly assigned to a component contribute the most to parameter changes. The case y_i is therefore assessed as significant if the maximum inferred support for any component in the mixture

$$Supp(y_i) = \max_{x_i \in X_i}[p(x_i \mid y_i, \theta^n)] \tag{5}$$

is less than some predefined significance threshold, denoted ST.

The lazy EM algorithm is theoretically justified by noticing that the formulation of Neal and Hinton (1998) is applicable for any arbitrary partitioning of data, as long as all data is visited regularly. Hence, the lazy EM algorithm is guaranteed to converge to a local maximum (or saddle point).

As with the incremental EM algorithm, monotonic behavior of log-likelihood values is not guaranteed. Nonetheless, convergence tests can be based on log-likelihood values obtained at full E-steps at little additional computational cost.

3. MAP estimation methods

As a alternative to traditional ML estimation we may consider a Bayesian interpretation of the estimation problem. Rather than maximize the likelihood, we incorporate prior information about parameters and find the largest posterior mode, the MAP. Dempster, Laird, and Rubin (1977) briefly describe how to modify the EM algorithm to produce the MAP. This has been further discussed in Green (1990).

Suppose we have information about parameters θ in the form of a prior distribution $p(\theta)$, then

$$p(\theta \mid y) \propto p(y \mid \theta)\, p(\theta)$$

We can then use the EM algorithm for MAP estimation by simply constructing the conditional expectation for the log-posterior mode

$$Q^*(\theta \mid \theta^n, y) = Q(\theta \mid \theta^n, y) + \log p(\theta)$$

In the M-step, Q^* is maximized instead of Q.

It is a trivial exercise to apply the results of Neal and Hinton (1998) to cover convergence guarantees for MAP estimation based on partial E-steps.

4. Experiments

Our experiments were performed on two real-world databases. The first data set, *MSNBC*, is a sparse, discrete data set that encodes the stories that people read on the MSNBC web site on December 21, 1998. The variables correspond to the most popular 303 stories. The data cases correspond to 597,971 visitors. Each data case stores, for each story, whether or not a particular visitor read that story.

The second real-world data set, *Speech*, holds data cases for a single sub-phonetic event observed for continuous speech. 33 continuous variables represent 12 mel-scale frequency cepstrum coefficients (MFCCs), log-energy and their first and some second order dynamics (Huang et al., 1995). The data set contains 21,888 cases for which the particular sub-phonetic event was observed for 10 ms time frames of the speech.

For both domains, we investigate finite mixture models in which each component model encodes the mutual independence of the variables $X = (X^1, X^2, \ldots, X^m)$. These models can be viewed as a naive-Bayes model with a hidden class node, also known as an AutoClass model (Cheeseman & Stutz, 1995). For the *MSNBC* domain, we consider mixtures with 25, 50, and 100 mixture components, where each variable in each component has a binomial distribution.[1] For the *Speech* domain, we consider mixtures of 2, 4, and 8 components, where each variable in each component is a univariate Gaussian—that is, mixtures of multivariate Gaussians with diagonal covariance matrices.

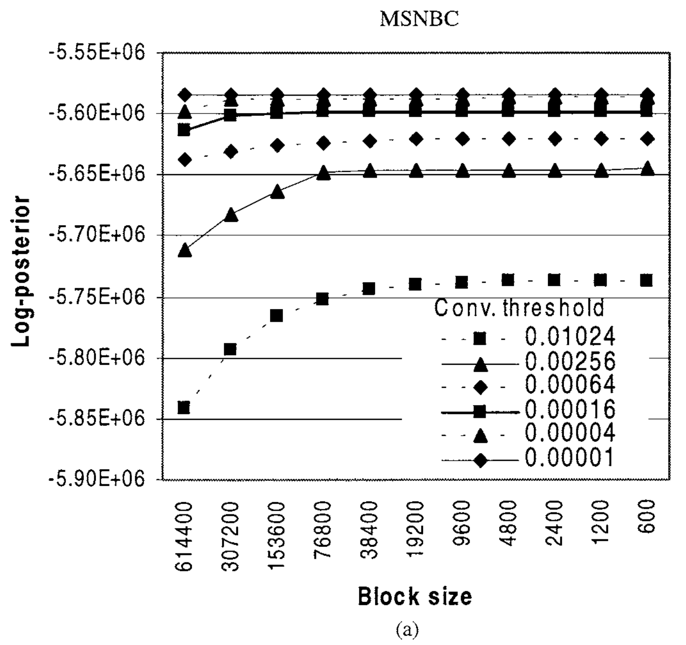
For the data sets, we evaluate the effectiveness of our methods for different sizes of training data by creating various subsets of the original data to be used as training data.

All experiments are performed on a PII 333 mHz dual processor machine with enough random access memory (RAM) to avoid paging.

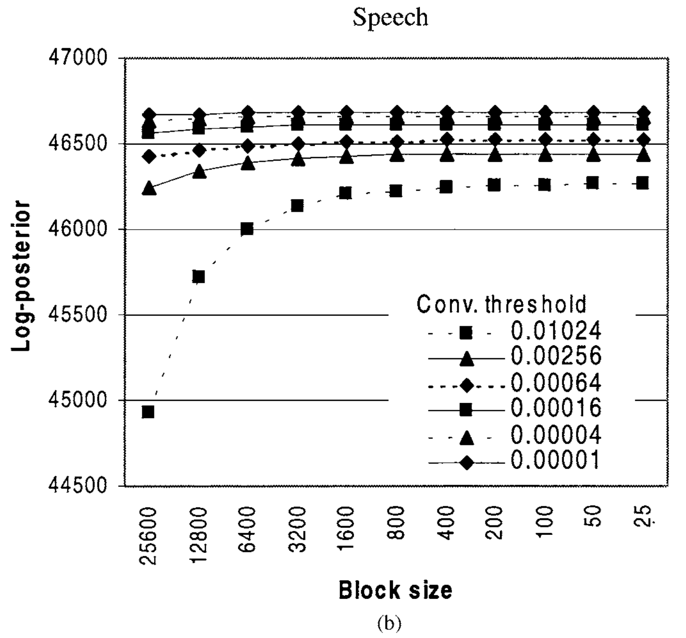
In addition to the two real-world data sets, we created numerous synthetic data sets, *Synthetic*, by drawing 40,000 random cases from mixtures of M-variate Gaussians with N mixture components for $M = 1, 2, 4, 8, 16$ and $N = 5, 10, 20, 40, 80$. Each Gaussian has a mean positioned randomly within the unit hypercube, and the overlap between Gaussians is controlled by a diagonal covariance matrix with randomly generated elements between 0 and $4\sigma^2$ for $\sigma = 0.05, 0.1, 0.2, 0.4$. The results for the synthetic data sets are similar to those for the real data sets, and will not be reported in detail.

4.1. Evaluation method

For our experiments we chose to perform MAP estimation using diffuse priors similar to those described in Chickering and Heckerman (1997) and in Thiesson, Meek, Chickering,



(a)



(b)

Figure 1. Incremental EM compared to standard EM for (a) *MSNBC* and (b) *Speech*. Each curve correspond to a particular convergence threshold and shows the log-posterior of the test set obtained for different block size partitionings. Standard EM correspond to block size 614400 for *MSNBC* and block size 25600 for *Speech*.

and Heckerman (1999) for discrete and continuous domains, respectively. Hence, we evaluate an algorithm according to how fast it approaches the MAP estimate compared to standard EM when starting from the same parameter initialization. A *speedup* factor is computed as the elapsed time for an algorithm to reach convergence divided by the elapsed time for standard EM to reach convergence. Thus, a speedup factor greater than 1 means the algorithm improves performance.

We can measure the quality of an estimate as an algorithm progresses through decreasing convergence thresholds by monitoring the progress in the log-posterior value, $\log p(\theta \mid y) = \log p(y \mid \theta) + \log p(\theta)$ for each iteration. However, for the real-world experiments, the computational cost is too high to be manageable in practice, when we compute the log-likelihood for all data at each iteration stepping through the data in small blocks. Instead, we compute the log-likelihood for a fixed random subset of the training data. For *MSNBC*, we use 29,899 of the training cases. For *Speech*, we use 4378 cases. To obtain the log-posterior value we proceed as if the random subset is representative of all data and scale the obtained log-likelihood appropriately before adding the log-prior. In the *Synthetic* experiments we use all training cases for the log-likelihood computations.

We use 10^{-5} as the convergence threshold for all of our experiments. To help justify this threshold, figure 1 shows representative convergence results for *MSNBC* with 50 mixture components and *Speech* with 4 components. The figure shows, at different values for the convergence threshold, the log-posterior of the test set obtained by incremental EM for different block-size partitionings. Standard EM correspond to block size 614400 for *MSNBC* and block size 25600 for *Speech*.

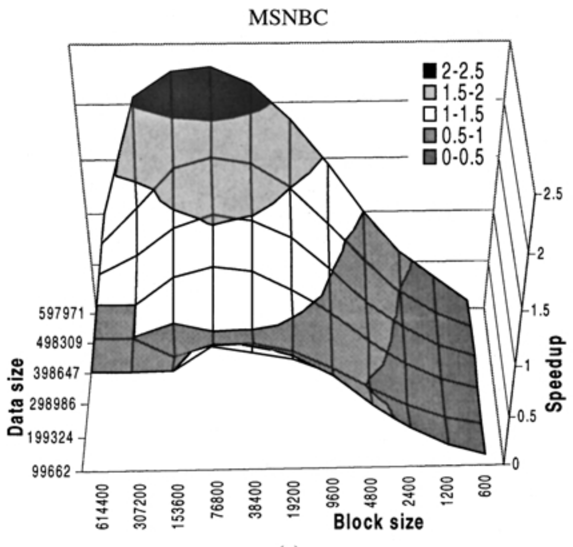
Two patterns are visible in the figure. One, as we decrease block size, the incremental EM converges to a higher log-posterior value than standard EM. Two, as we decrease the convergence threshold, the difference between incremental and standard EM diminishes. For a threshold of 10^{-5}, both algorithms converge to essentially the same log-posterior value. The second observation is important, because it allows us to fairly compare runtimes for the two algorithms. That is, we easily can measure the time it takes for both algorithms to converge to the similar log-posterior values, by using a small convergence threshold for both algorithms. A similar pattern can be observed for the lazy EM algorithm.

4.2. *Results for incremental EM*

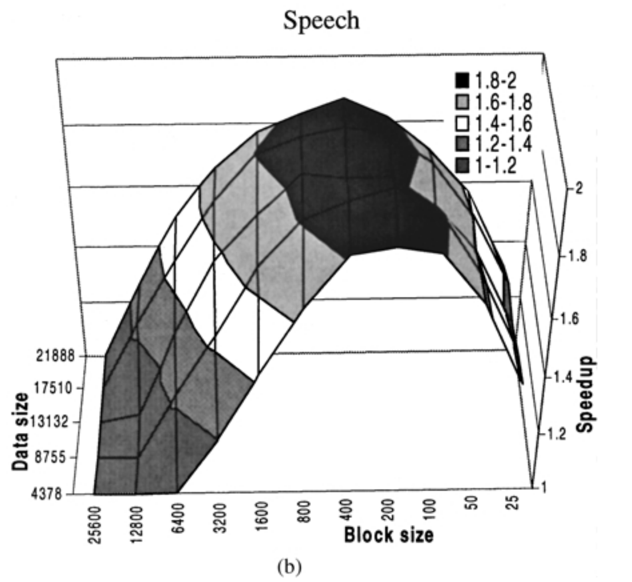
All models were trained with the incremental EM algorithm for partitions of the training data into various block sizes. For *MSNBC*, we used block sizes 600×2^n for $n = 0, 1, \ldots, 10$; for *Speech* we used blocks of size 25×2^n with $n = 0, 1, \ldots, 10$. The last block size was most often smaller than the others. As described in Section 2.2, we were careful about initialization in all experiments: we processed each data block once before performing the initial M-step.

Figure 2 shows speedup results for *MSNBC* with 50 mixture components and *Speech* with 4 components. These results are representative for the remaining experiments. The figure shows the speedup for varying sizes of training data with different block size partitionings.

For each data set, our algorithm converged to the same log-posterior value regardless of block size. With full data sets, the incremental algorithm speeds up EM by a factor of 2.3



(a)



(b)

Figure 2. Incremental EM compared to standard EM for (a) *MSNBC* and (b) *Speech*. The figures show speedup as a function of data size and block size partitioning. Entries with block sizes greater than the data size correspond to standard EM.

for a partition of the *MSNBC* data into 8 blocks of size 76,800, and a factor of 1.9 for the *Speech* data partitioned into 55 blocks of size 400. Note that the speedup on the *MSNBC* data is in addition to the improvement from data sparsity, mentioned in Section 1.

Two characteristic patterns can be observed in the figure. First, optimal block size is fairly constant as training data size is varied. Second, speedup increases with training data size. The second observation follows from the first, because increasing training data size results in more blocks, which is exploited by the incremental EM algorithm. In fact, we expect the speedup to increase indefinitely with sample size. Roughly speaking, we expect the optimal speedup to increase by a factor between one and two when doubling the size of the training data. The more similar the data, the greater the speedup factor.

Figure 2 also reveals a great variation in speedup for different block sizes. In particular, the figure shows that too large a block size is not optimal and too small a block size may even slow the algorithm down compared to standard EM. For example, speedup for full data *MSNBC* starts decreasing at block size 38,400 and at block size 2,400 it is slower than standard EM. This observation raises an important issue in order to make incremental EM operational as an acceleration method: How do we select the block size for data partitioning?

Further experimental investigations revealed the following three patterns that lead to an answer in situations where the local maximum in log-posterior is independent of block size, for instance, obtained by a small convergence threshold (see Section 4.1). One, the block size that yields the best speedup during the initial iterations of the incremental EM algorithm is roughly the block size that yields the best speedup at convergence. Furthermore, if one block size is better than another during the initial iterations, the same holds true at convergence. Here, speedup at iteration n (n small) is measured as the elapsed time for incremental EM to reach the same log-posterior value achieved by standard EM at iteration n divided by the elapsed time for standard EM to get this far. Two, the log-posterior values increase approximately linearly with time during initial iterations. Three, speedup (at convergence) versus block size exhibit a single broad peak around the optimal block size.

Given these observations, we can roughly identify the optimal block size as follows. Let s_0 and s_1 be the log-posterior values that the incremental EM algorithm achieves after, respectively, initialization and a single pass though the data. Let t be the time required for this first pass. Let r be the ratio $(s_1 - s_0)/t$. Given the first two observations, the block size that minimizes r will be roughly equal to the optimal block size. Given the third observation, we can find the block size that minimizes r using a simple search.

Our selection method applied to the *MSNBC* and *Speech* domains is illustrated in Tables 1 and 2. Table 1 shows r versus block size. Table 2 shows speedup at convergence both for the optimal block size and the block size selected using our approach. We see that, using our method, we achieve a significant speedup over standard EM that is optimal or close to optimal. In the Speech experiments, although the selected and optimal block sizes are not equal, the corresponding speedups are nearly identical.

Finally, the results in this section have been presented in terms of speedup factors obtained for a convergence level of 10^{-5}. It is also interesting, however, to look at how much time it takes standard EM and incremental EM to achieve various degrees of convergence. To investigate this dependence, we plot the log-posterior value obtained on successive iterations

Table 1. Initial speedup ratios ($r = (s_1 - s_0/t)$) used to select the block size for incremental EM.

	Block size										
	614400	307200	153600	76800	38400	19200	9600	4800	2400	1200	600
MSNBC 25	40.7	46.2	48.4	47.9	45.1	38.0	30.2	20.7	12.7	7.2	3.8
MSNBC 50	15.6	17.7	18.8	18.9	17.4	15.0	11.6	7.9	4.8	2.7	0.7
MSNBC 100	8.1	9.3	10.0	10.1	9.5	8.1	6.3	4.3	2.6	0.9	0.4
	Block size										
	25600	12800	6400	3200	1600	800	400	200	100	50	25
Speech 2	0.37	0.95	1.60	1.68	1.84	1.90	1.86	1.83	1.72	1.55	1.28
Speech 4	0.14	0.44	0.85	0.93	1.06	1.12	1.11	1.10	1.03	0.96	0.79
Speech 8	0.46	0.70	0.92	0.96	1.03	1.07	1.04	1.04	0.98	0.90	0.76

Block size 614400 for *MSNBC* and block size 25600 for *Speech* correspond to standard EM. The number after the domain name is the number of components in the considered mixture model.

versus the accumulated time to complete the iteration for the standard EM algorithm and compare this with a similar plot for the incremental EM algorithm.

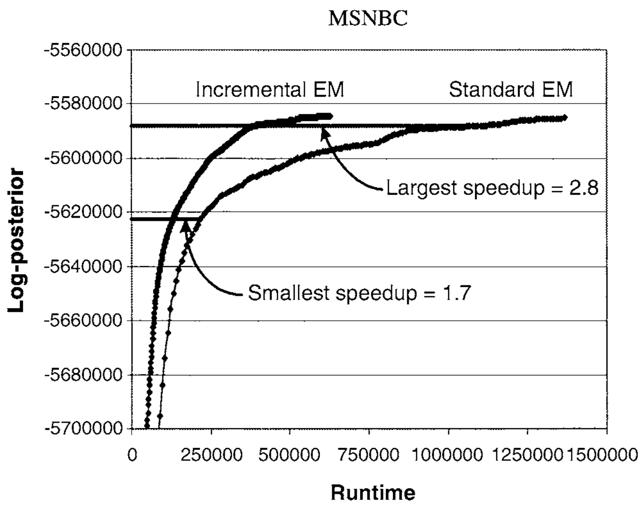
Figure 3(a) contains comparative plots for the incremental EM algorithm with block size 76,800 and the standard EM algorithm for the *MSNBC* data set. Similarly, figure 3(b) contains comparative plots for the incremental EM algorithm with block size 400 and the standard EM algorithm for the *Speech* data set. Both figures are truncated at the log-posterior value obtained for standard EM at iteration 10.

First note that, for both datasets, incremental EM dominates standard EM in that incremental EM achieves any particular log-posterior value more quickly than does standard EM. Furthermore, note that speedup varies (non-monotonically) with log-posterior. For both comparative plots we have marked the largest and smallest speedups with horizontal lines. The largest speedup for *MSNBC* is obtained at iteration 129 with a value of 2.8 and the smallest at iteration 26 with a value of 1.7. At convergence determined by a threshold of 10^{-5}, the speedup is 2.3. Similarly, for *Speech*, the largest speedup is obtained at iteration 12 with a value of 2.2, the smallest speedup is 1.8 obtained at iteration 1 (not in the figure), and the speedup at convergence is 1.9.

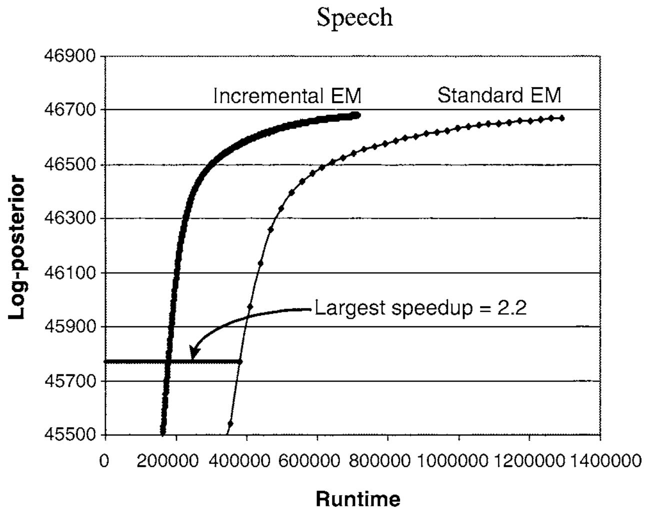
Table 2. Block sizes and speedup at convergence for block sizes determined by our approach (auto) and by search at convergence (optimal).

	Auto block size	Optimal block size	Auto speedup	Optimal speedup
MSNBC 25	153600	153600	2.0	2.0
MSNBC 50	76800	76800	2.3	2.3
MSNBC 100	76800	76800	1.4	1.4
Speech 2	800	800	1.9	1.9
Speech 4	800	400	1.8	1.9
Speech 8	800	200	1.8	2.0

The number after the domain name is the number of components in the considered mixture model.



(a)



(b)

Figure 3. The curves show log-posterior value versus accumulated time at each iteration for standard EM and incremental EM for (a) *MSNBC* and (b) *Speech*. Both figures are truncated at the log-posterior value obtained for standard EM at iteration 10. Horizontal lines show largest and smallest speedup points, except for the *Speech* minimum speedup point, which is before the first iteration depicted in the graph.

4.3. Results for lazy EM

Speech models were trained with the lazy EM algorithm for schedules with 1, 2, 4, 8 lazy steps between full EM steps. The subset of cases used in lazy steps were determined by significance thresholds, ST set to 50, 70, 80, 85, 90, 95, 99 percent.

Representative of most experiments (we will comment on the remaining experiments later), figure 4 shows the speedup results for full data with four mixture components.

As indicated in the figure, there is an optimal value for ST. With a low value for ST, only a few cases will be considered in the lazy E-steps and time to convergence is long. On the other hand, choosing too large a value for ST leads to most cases being significant and hence used for the lazy E-steps. Lazy EM will in this case approximate standard EM with the additional computational overhead for assessing significant data. In general, the best results were obtained for ST equal to 90% or 95%.

There is also an optimal value for the laziness schedule. This pattern is explained by the pattern in figure 5, which shows a trace of log-posterior values versus accumulated runtime at each iteration for the full data *Speech* experiment with eight lazy E-steps and ST set to 95%. For reference, the figure also shows a similar trace for standard EM. We see that the effect of successive lazy steps decreases in a monotonic way until a new set of lazy cases are selected by a full E-step. After a certain number of successive lazy steps, the increase in log-posterior value is too small and we will do better by re-setting the set of lazy cases. In general, the best results were obtained for 2 or 4 successive lazy steps with little difference between the two choices.

For the *Synthetic* experiments, we found that with increasing overlap between mixture components, cases were more rarely assigned to a particular component with high

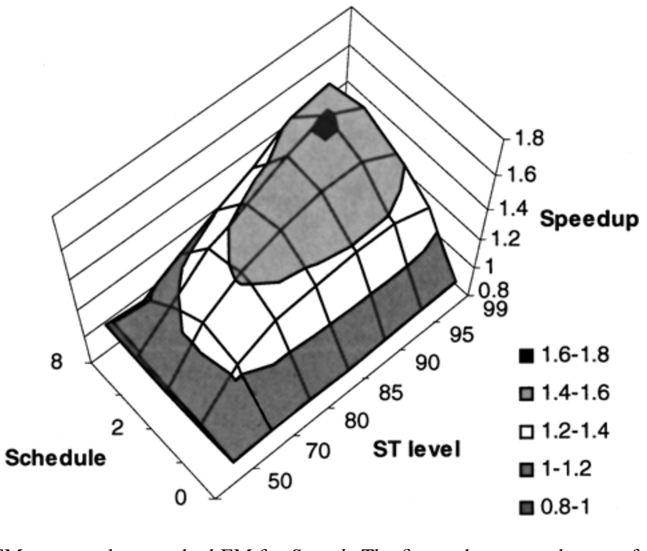


Figure 4. Lazy EM compared to standard EM for *Speech*. The figure shows speedup as a function of laziness schedule and significance thresholds. Standard EM correspond to a laziness schedule of zero.

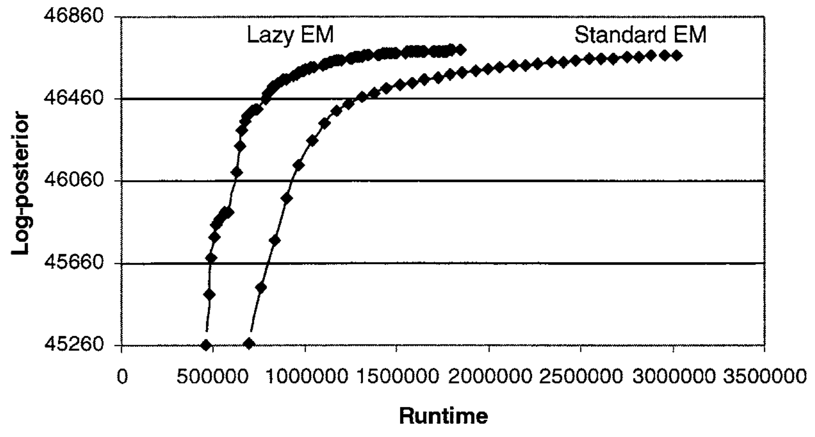


Figure 5. Traces of log-posterior value versus accumulated runtime at each iteration for standard and lazy EM *Speech* experiments. The figure is truncated at the log-posterior value obtained for lazy EM at the second full E-step (iteration 10).

probability. In some of the experiments we had to consider a lower ST value in order to prevent all cases from being assigned to the subset used for lazy steps. A natural alternative that addresses this problem of our simplistic method for choosing significant cases is to select the subset of cases used in lazy steps as some (decaying) percentage of most significant cases.

The speedup factor for the *Speech* experiments is roughly 1.6 and in general a little less for the *Synthetic* experiments.

5. Related work

There are several alternative methods for accelerating EM in situations where the size of the data makes the computational cost of the E-step expensive. We describe two types of methods: methods that partially forget earlier statistics and methods that compress or summarize multiple cases.

A *forgetful* variant of incremental EM is a method that uses expected sufficient statistics computed as an (exponential) decaying average of recently visited data blocks. Forgetful or *on-line* variants of EM have been proposed by several authors. See, for instance, Nowlan (1991) or Sato and Ishii (2000) with generalizations in Sato (1999). These variants often assume that the blocks consist of individual cases, but can be easily extended to blocks of arbitrary size.

Forgetful methods are often used to estimate time-varying parameters, however, they are also potentially useful for extremely large data sets. In such situations, it may be possible for a forgetful incremental EM algorithm to converge more quickly than a non-forgetful algorithm. This possibility arises due to the fact that the expected statistics for blocks computed during early passes through the data contain relatively large inaccuracies. By forgetting these statistics one can more rapidly improve the parameter estimates. On the other hand, in our limited comparison of forgetful and non-forgetful methods, we have found that the forgetful methods do not provide significant improvement over incremental EM in situations where multiple passes through the data are required for convergence. This observation is not surprising, given that the incremental EM is in fact a coarse forgetful method which completely forgets the statistics computed for a block after one pass through the data.

Application of forgetful methods to large data sets requires selection of a decay rate. In addition, to allow convergence in less than one pass through the data, an alternative convergence criterion is required. Finally, unlike the incremental EM algorithm, naive implementations cannot guarantee convergence to a local maximum of the likelihood for finite data sets.

The second type of method for accelerating EM relies on compression. Roughly, in such a method, one represents a set of cases by a *compressed case*. These methods accelerate EM by (approximately) updating the statistics for a set of cases associated with a compressed case by computing the expected statistics for the compressed case. Note that the choice of cases to be compressed need not be fixed in advance and different sets of compressed cases can be used on different iterations of the algorithm.

One such compression method is that of Moore (1999) who reports exceptional speedup results for low dimensional mixture model problems with large sample size. In Moore's approach, one first builds a multi-resolution kd-tree for the cases in the data set. In this representation, a compressed case is a vertex in the kd-tree that represents all of the cases stored below that vertex. Then, one runs a version of EM in which the expected sufficient statistics are computed on the basis of compressed cases. The choice of compressed cases (i.e., the choice of vertices) is determined anew on each E-step so as to balance the quality of the approximation with the computational cost of the E-step. Unfortunately, kd-tree algorithms do not scale well with dimensionality.

A closely related method, which scales with dimensionality, is that of McCallum, Nigam, and Ungar (2000). Instead of a hard partitioning of the data into a multi-resolution kd-tree, this method uses a crude similarity measure to first divide the data into so-called canopies, which are subsets of the data that may be overlapping. During an iteration of the EM algorithm, individual components of a mixture model are associated with one or more overlapping canopies, and the update of a mixture component is only influenced by the subset of data cases that belong to canopies associated with that component. In a variation of the method, the update of a component is additionally influenced by the mean of each canopy not associated with the component. The speedup and accuracy of their method is closely related to the overlap of canopies. The resulting speedup will be significant if the data cases permit a division into well-separated canopies and the method identifies such a division. When data cases are generated from overlapping clusters, however, we expect that either the speedup or the accuracy of the final estimate will decrease. Although McCallum, Nigam, and Ungar (2000) do not report results as a function of cluster overlap, Moore (1999), whose method is similar, does. In experiments with synthetic data, he finds that speedup indeed decreases with increasing clustering overlap. In contrast, in similar synthetic experiments, our algorithms do not experience a decrease in speedup.

A different type of compression method is described in Bradley, Fayyad, and Reina (1998). Their method partitions the data set into blocks of cases, and then iterates across these blocks, adding a new block to a *working set* on each iteration. During an iteration, a standard EM algorithm is run to convergence, and then cases that are close in Mahalanobis distance are compressed into a compressed case. The E-step is modified to accommodate these compressed cases. Once cases are compressed, they can not be separated. Consequently, this approach does not guarantee convergence to a local maximum of the likelihood.

6. Summary and future work

We have considered two approaches for accelerating the EM algorithm for large databases. Both approaches, the incremental EM and lazy EM, are based on partial E-steps for which we can use the theoretical results of Neal and Hinton (1998) to obtain standard convergence guarantees of EM. For both of these methods we have discussed implementation details that are important for coding operational algorithms that will in fact accelerate EM.

For incremental EM, we described a method for selecting a near-optimal block size for partitioning the training data. Our experiments showed that this selection is an important part of the incremental EM algorithm, as the speedup obtained depends dramatically on the chosen block size. We also described a convergence test for incremental EM that can be related to standard convergence tests for EM. Finally, we pointed out that small sample problems may appear during the first pass through the data, and we described how to avoid these problems.

We have demonstrated a significant speedup for incremental EM over standard EM—a speedup that increases with sample size. For extremely large data sets, forgetful variants of incremental EM may even increase the speedup further, as argued in Section 5. Perhaps our method for selecting an optimal block size can be used for these variants as well.

Additional research into choosing the optimal block size is also needed in situations where only a subset of the data fits in RAM. In these situations, the paging cost needs to be factored into the choice of block size.

The viability of lazy EM rests on the assumption that not all data is of equal importance throughout the iterations. We described a computationally efficient criterion which can be used to determine the significance of cases in finite-mixture-model settings. The speedup depends on the scheduling for lazy iterations as well as the threshold for choosing the subset of data used during these iterations. Using a simple approach with a constant schedule and a constant significance threshold, we were able to demonstrate the effectiveness of the approach.

For both the lazy EM and incremental EM algorithms, we expect that further research on alternative schedules and selection methods will improve both algorithms. Potential alternatives include non-cyclical traversal of blocks in the incremental EM algorithm, dynamic schedules for choosing the number of lazy iterations and choosing the significant data in the lazy EM algorithm, and mechanisms for ensuring disparity among cases in the considered block of data for both algorithms.

Finally, the lazy and incremental algorithms can be combined into a single algorithm which could lead to further improvements. There are many possibilities. For instance, one could imagine an incremental algorithm which, in addition to an incremental iteration for a block of data, performs a few lazy iterations based on significant cases in that block before moving on to the next block of data. Another possibility would be to perform the full E-step iteration in the lazy algorithm as an incremental pass through the data.

Acknowledgments

We thank Christopher M. Bishop for helpful discussions on convergence criteria, and we thank Steven White from MSNBC for making data available. We also thank the anonymous reviewers for their comments on a preliminary version of this paper.

Note

1. This model is also known as a log-linear model with main effects only. See, for example, Agresti (1990).

References

Agresti, A. (1990). *Categorical Data Analysis*. New York: John Wiley and Sons.

Bradley, P., Fayyad, U., & Reina, C. (1998). Scaling EM (Expectation Maximization) clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research.

Cheeseman, P. & Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In U. Fayyad, G. Piatesky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 153–180). Menlo Park, CA: AAAI Press.

Chickering, D. M. & Heckerman, D. (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning, 29*, 181–212.

Chickering, D. M. & Heckerman, D. (1999) Fast learning from sparse data. In K. B. Laskey & H. Prade (Eds.), *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 109–115). San Mateo, CA: Morgan Kaufmann Publishers.

Dempster, A. P., Laird, N., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B, 39*, 1–38.

Green, P. J. (1990). On use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society, Series B, 52*, 443–452.

Huang, X., Acero, A., Alleva, F., Hwang, M.-Y., Jiang, L., & Mahajan, M. (1995). Microsoft Windows highly intelligent speech recognizer: Whisper. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1995. ICASSP-95* (Vol. 1, pp. 93–96).

Jamshidian, M. & Jennrich, R. I. (1993). Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association, 88*(421), 221–228.

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society, Series B, 44*(2), 226–233.

McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In R. Ramakrishnan & S. Stolfo (Eds.), *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 169–178). New York: ACM.

Meilijson, I. (1989). A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society, Series B, 51*(1), 127–138.

Meng, X.-L. & Rubin, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika, 80*(2), 267–278.

Meng, X.-L. & van Dyk, D. (1997). The EM algorithm—an old folksong sung to a fast new tune (with discussion). *Journal of the Royal Statistical Society, Series B, 59*, 511–567.

Moore, A. (1999). Very fast EM-based mixture model clustering using multiresolution kd-trees. In M. S. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems. Proceedings of the 1998 Conference* (Vol. 11, pp. 543–549). Cambridge, MA: MIT Press.

Neal, R. & Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan (Ed.), *Learning in Graphical Models* (pp. 355–371). The Netherlands, Kluwer Academic Publishers.

Nowlan, S. J. (1991). Soft competitive adaptation: Neural network learning algorithms based on fitting statistical mixtures. Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Sato, M. (1999). Fast learning of on-line em algorithm. Technical Report, ATR Human Information Processing Research Laboratories 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan.

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

Thiesson, B. (1995). Accelerated quantification of Bayesian networks with incomplete data. In U. M. Fayyad, & R. Uthurusamy (Eds.), *Proceedings of First International Conference on Knowledge Discovery and Data Mining* (pp. 306–311). Menlo Park, CA: AAAI Press.

Thiesson, B., Meek, C., Chickering, D., & Heckerman, D. (1999). Computational efficient methods for selectiong among mixtures of graphical models, with discussion. In J. M. Bernardo, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.), *Bayesian Statistics: Proceedings of the Sixth Valencia International Meeting* (Vol. 6, pp. 631–656). Oxford: Oxford University Press.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). Birch: An efficient data clustering method for very large databases. In *Proceedings of the Fifteenth ACM SIGMOD International Conference on Management of Data and Symposium on Principles of Database Systems* (pp. 103–114). New York: ACM.

Received April 27, 2000
Revised April 11, 2001
Accepted April 11, 2001
Final manuscript April 23, 2001