Statistics and Computing **13**: 45–55, 2003
© 2003 Kluwer Academic Publishers. *Manufactured in The Netherlands.*

On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures

S. K. NG and G. J. McLACHLAN*

Department of Mathematics, University of Queensland, St. Lucia, Queensland 4072 Australia

Received October 2000 and accepted January 2002

The EM algorithm is a popular method for parameter estimation in situations where the data can be viewed as being incomplete. As each E-step visits each data point on a given iteration, the EM algorithm requires considerable computation time in its application to large data sets. Two versions, the incremental EM (IEM) algorithm and a sparse version of the EM algorithm, were proposed recently by Neal R.M. and Hinton G.E. in Jordan M.I. (Ed.), Learning in Graphical Models, Kluwer, Dordrecht, 1998, pp. 355–368 to reduce the computational cost of applying the EM algorithm. With the IEM algorithm, the available n observations are divided into B ($B \leq n$) blocks and the E-step is implemented for only a block of observations at a time before the next M-step is performed. With the sparse version of the EM algorithm for the fitting of mixture models, only those posterior probabilities of component membership of the mixture that are above a specified threshold are updated; the remaining component-posterior probabilities are held fixed. In this paper, simulations are performed to assess the relative performances of the IEM algorithm with various number of blocks and the standard EM algorithm. In particular, we propose a simple rule for choosing the number of blocks with the IEM algorithm. For the IEM algorithm in the extreme case of one observation per block, we provide efficient updating formulas, which avoid the direct calculation of the inverses and determinants of the component-covariance matrices. Moreover, a sparse version of the IEM algorithm (SPIEM) is formulated by combining the sparse E-step of the EM algorithm and the partial E-step of the IEM algorithm. This SPIEM algorithm can further reduce the computation time of the IEM algorithm.

Keywords: incremental EM algorithm, sparse IEM algorithm, partial E-step, efficient updating formulas

1. Introduction

The EM algorithm (Dempster, Laird and Rubin 1977) is a popular tool for parameter estimation in a variety of problems involving missing data or incomplete information. As set out in some detail in McLachlan and Krishnan (1997, Section 1.7), the EM algorithm has a number of desirable properties, including its simplicity of implementation and reliable global convergence. However, a common criticism is that the convergence with the EM algorithm is only at a linear rate. In the context of mixture models, various attempts have been proposed to accelerate the EM iteration. In considering methods for speeding up the convergence of the EM algorithm, it is highly desirable if the simplicity and stability of the EM algorithm can be preserved. In situations where the M-step is computationally complicated, conditional M-steps can be used to avoid the requirement of iterative M-steps. The so-called Expectation/Conditional Maximization (ECM) algorithm (Meng and Rubin 1993, Meng and van Dyk 1997) shares all the appealing convergence properties of the EM algorithm (Meng 1994). Among other approaches that have been considered in the literature, Meilijson (1989) and Jamshidian and Jennrich (1997) replace the M-step with a (quasi-) Newton-type step, while Jamshidian and Jennrich (1993) and Thiesson (1995) replace the M-step with a (conjugate) gradient step.

In applications where the M-step is computationally simple, for example, in fitting mutivariate normal mixtures, the rate of convergence of the EM algorithm depends mainly on the computation time of an E-step. Because each E-step visits each data point, the EM algorithm requires much computation time in its application to large data sets. Neal and Hinton (1998) proposed the incremental EM (IEM) algorithm to improve the rate of convergence of the EM algorithm. More specifically, suppose the available n observations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ are divided into B ($B \leq n$) blocks. The IEM algorithm proceeds by implementing the E-step for only a block of observations at a time before performing a M-step. A "pass" or scan of the IEM algorithm thus consists of B partial E-steps and B M-steps. The argument for improved rate of convergence is that the IEM algorithm exploits new information more quickly rather than waiting for a complete scan of the data before parameters are updated by an M-step. The time to convergence for the IEM algorithm against the standard EM algorithm is a tradeoff between the additional computation time per scan and the fewer number of scans required because of the more frequent updating after each partial E-step. When the data are partitioned into more and more blocks, the additional computation time per scan increases and hence the time to convergence will eventually start to rise. Thus, an appropriate choice of B is important for the performance of the IEM algorithm. In this paper, we propose a simple guide for choosing the number of blocks.

Another approach suggested by Neal and Hinton (1998) for speeding up the EM algorithm is the so-called sparse EM (SPEM) algorithm. In fitting a mixture model to a data set by maximum likelihood via the EM algorithm, the current estimates of the posterior probabilities for some components of the mixture for a given data point $\boldsymbol{y}_j$ are often close to zero. With the SPEM algorithm, these posterior probabilities are held fixed, while only the posterior probabilities for the remaining components in the mixture are updated.

In this paper, a sparse version of the IEM algorithm (SPIEM) is formulated to further reduce the computation time of the IEM algorithm, by combining the sparse E-step of the SPEM algorithm and the partial E-step of the IEM algorithm. In Section 2, we formulate the IEM algorithm and give efficient updating formulas for the extreme case of $B = n$ blocks and in Section 3, we present simulations to compare the relative performances of the IEM with various numbers of blocks B and the standard EM algorithm. In Section 4, a simple rule for choosing an appropriate number of blocks with the IEM algorithm is derived and a simulation study is performed to demonstrate the performance of the rule and compare its performance with a search method proposed by Thiesson, Meek and Heckerman (2001). The SPIEM algorithm is formulated in Section 5, and in Section 6, we present simulations to compare the SPIEM, the IEM, and the standard EM algorithms. Practical values are suggested for the tuning constants in the formulation of the SPIEM algorithm.

2. The incremental EM algorithm

Let $\boldsymbol{y} = (\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_n^T)^T$ denote an observed random sample of size n, where $\boldsymbol{y}_j (j = 1, \ldots, n)$ is a vector of p dimensions and the superscript T denotes vector transpose. It is proposed to fit a g-component normal mixture model. The log likelihood for the vector Ψ of unknown parameters is given by

$$\log L(\Psi) = \sum_{j=1}^{n} \log \left\{ \sum_{i=1}^{g} \pi_i \phi(\boldsymbol{y}_j; \mu_i, \Sigma_i) \right\} \tag{1}$$

where $\phi(\boldsymbol{y}; \mu_i; \Sigma_i)$ denotes the multivariate normal density function with mean μ_i and covariance matrix Σ_i, and the π_i denote the mixing proportions that are nonnegative and sum to one. Solutions of the likelihood equation corresponding to local maxima can be found iteratively by application of the EM algorithm.

With the IEM algorithm, the data are divided into B blocks of equal or near equal size. We let $\Psi^{(k)}$ denote the estimate of Ψ after the kth scan, and $\Psi^{(k+b/B)}$ the estimate of Ψ after the bth iteration on the $(k + 1)$th scan ($b = 1, \ldots, B$). It is computational advantageous to work in terms of the sufficient statistics.

E-step: For the first scan ($k = 1$), a full E-step is performed (see Section 3.1), which requires the calculation of

$$T_{i1}^{(0)} = \sum_{j=1}^{n} \tau_i(\boldsymbol{y}_j; \Psi^{(0)}), \tag{2}$$

$$T_{i2}^{(0)} = \sum_{j=1}^{n} \tau_i(\boldsymbol{y}_j; \Psi^{(0)})\boldsymbol{y}_j, \tag{3}$$

$$T_{i3}^{(0)} = \sum_{j=1}^{n} \tau_i(\boldsymbol{y}_j; \Psi^{(0)})\boldsymbol{y}_j\boldsymbol{y}_j^T, \tag{4}$$

where

$$\tau_i(\boldsymbol{y}_j; \Psi^{(0)}) = \frac{\pi_i^{(0)}\phi(\boldsymbol{y}_j; \mu_i^{(0)}, \Sigma_i^{(0)})}{\sum_{h=1}^{g} \pi_h^{(0)}\phi(\boldsymbol{y}_j; \mu_h^{(0)}, \Sigma_h^{(0)})} \tag{5}$$

is the current estimate of the posterior probability that $\boldsymbol{y}_j$ belongs to the ith component. The equations (2) to (4) are the conditional expectations of the sufficient statistics, using the initial value $\Psi^{(0)}$ for Ψ. On subsequent scans, these quantities are updated for only a block of observations at a time on the E-step. For example, on the $(b+1)$th iteration of the $(k+1)$th scan ($b = 0, \ldots, B-1$), the current conditional expectations of the sufficient statistics $T_{i1}^{(k+b/B)}$, $T_{i2}^{(k+b/B)}$, and $T_{i3}^{(k+b/B)}$ are obtained for $i = 1, \ldots, g$, using the relationship

$$T_{iq}^{(k+b/B)} = T_{iq}^{(k+(b-1)/B)} - T_{iq,b+1}^{(k-1+b/B)} + T_{iq,b+1}^{(k+b/B)}$$
$$(q = 1, 2, 3; b = 0, \ldots, B - 1), \tag{6}$$

where only

$$T_{i1,b+1}^{(k+b/B)} = \sum_{j \in S_{b+1}} \tau_i(\boldsymbol{y}_j; \Psi^{(k+b/B)}), \tag{7}$$

$$T_{i2,b+1}^{(k+b/B)} = \sum_{j \in S_{b+1}} \tau_i(\boldsymbol{y}_j; \Psi^{(k+b/B)})\boldsymbol{y}_j, \tag{8}$$

$$T_{i3,b+1}^{(k+b/B)} = \sum_{j \in S_{b+1}} \tau_i(\boldsymbol{y}_j; \Psi^{(k+b/B)})\boldsymbol{y}_j\boldsymbol{y}_j^T, \tag{9}$$

have to be calculated. This is because the first term on the right-hand side of (6) is available from the previous iteration, while the second term is available from the previous scan. In (7) to (9), S_{b+1} denotes the subset of $\{1, \ldots, n\}$ containing the subscripts of those $\boldsymbol{y}_j$ that belong to the $(b + 1)$th block.

M-step: For $i = 1, \ldots, g$, the estimates of π_i, μ_i and Σ_i are updated, based on the conditional expectations of the sufficient statistics, as follows:

$$\pi_i^{(k+(b+1)/B)} = T_{i1}^{(k+b/B)}/n, \tag{10}$$

$$\mu_i^{(k+(b+1)/B)} = T_{i2}^{(k+b/B)}/T_{i1}^{(k+b/B)}, \tag{11}$$

$$\Sigma_i^{(k+(b+1)/B)} = \left\{T_{i3}^{(k+b/B)} - T_{i1}^{(k+b/B)^{-1}} \times T_{i2}^{(k+b/B)} T_{i2}^{(k+b/B)^T}\right\}/T_{i1}^{(k+b/B)}. \tag{12}$$

The argument for improved rate of convergence is that the IEM algorithm exploits new information more quickly rather than waiting for a complete scan of the data before parameters are updated by an M-step.

The theoretical justification for the IEM algorithm has been provided by Neal and Hinton (1998). Let z denote the vector containing the unobservable data and let P be any distribution defined over the support of $\boldsymbol{Z}$. They considered the function

$$F(P, \Psi) = \log L(\Psi) - KL[P, f(z \mid y; \Psi)],$$

where $f(z \mid y; \Psi)$ is the conditional distribution of $\boldsymbol{Z}$ given the observed data and $KL[P, f(z \mid y; \Psi)]$ is the Kullback-Leibler information that measures the divergence of P relative to $f(z \mid y; \Psi)$. They showed that both the E- and M-steps of the IEM algorithm monotonically increase $F(P, \Psi)$ and if a local maximum (or saddle point) of $F(P, \Psi)$ occurs at P^* and Ψ^*, then a local maximum (or saddle point) for the log likelihood (1) occurs at Ψ^* as well. However, as the second argument Ψ of $\tau_i(\boldsymbol{y}_j; \Psi)$ in (5) is changing at each iteration within each scan, the same argument for proving that EM always increases the log likelihood cannot be adopted. That is, the current theoretical results for the IEM algorithm do not promise monotonic behaviour of the log likelihood as the EM algorithm does. However, it is noted that $F(P, \Psi)$ can be considered as a lower bound on the log likelihood since the Kullback-Leibler information is non-negative. For given P, as obtained in the E-step, the M-step increases $F(P, \Psi)$ with respect to Ψ. It follows that

$$F\left(P, \Psi^{(k+(b+1)/B)}\right) \geq F\left(P, \Psi^{(k+b/B)}\right) \quad (b = 0, \ldots, B-1).$$

That is, the lower bound of the log likelihood is monotonic increasing after each iteration.

In order to reduce the computational time, we approximate the value of the log likelihood after each E-step, based on the current estimate of Ψ. More precisely, the log likelihood based on $\Psi^{(k+b/B)}$ is calculated after the E-step on the $(b+1)$th iteration of the $(k+1)$th scan using the approximation.

$$\log L\left(\Psi^{(k+b/B)}\right) \approx \log L\left(\Psi^{(k+(b-1)/B)}\right) + \sum_{j \in S_{b+1}} \left\{\log f\left(\boldsymbol{y}_j; \Psi^{(k+b/B)}\right) - \log f\left(\boldsymbol{y}_j; \Psi^{(k-1+b/B)}\right)\right\}, \tag{13}$$

for $b = 0, \ldots, B-1$. Convergence tests can be based on B succesive log likelihood values or, alternatively, we can determine the convergence based on the incremental log likelihood after a complete scan of all the data (Thiesson, Meek and Heckerman 2001). The latter convergence criterion is adopted in our analysis. In our simulation experiments, the values of the incremental log likelihood after a complete scan do show monotonic increasing behaviour.

2.1. *IEM algorithm for singleton blocks*

In some applications, we may have situations where data actually appear one at a time. In this case, we may need to adopt the IEM algorithm with $B = n$ and implement the E-step for only a single observation at a time before performing a M-step. It will be seen in Section 3 that the additional CPU time for each scan starts to increase as B becomes sufficiently large. Fortunately, the time to convergence can be considerably reduced by the use of updating formulas that avoid the direct calculation of the inverses and determinants of the component-covariance matrices in the updating of the component-posterior probabilities for a single observation in (5), where the probability density function of the multivariate normal distribution is evaluated. These formulas are obtained by modifying similar formulas in Friedman (1989) for the case where an observation is deleted from the sample.

Without loss of generality, we assume that the jth block consists of the jth observation $\boldsymbol{y}_j (j = 1, \ldots, n)$. For brevity of notation, we let τ_{ij}^* and τ_{ij} denote the posterior probability $\tau_i(\boldsymbol{y}_j; \Psi)$ evaluated for Ψ equal to $\Psi^{(k-1+(j-1)/n)}$ and $\Psi^{(k+(j-1)/n)}$, respectively $(i = 1, \ldots, g)$. Also, we write $\pi_i^{(k+(j-1)/n)}$, $\mu_i^{(k+(j-1)/n)}$, and $\Sigma_i^{(k+(j-1)/n)}$ as π_i, μ_i, and Σ_i, respectively. The corresponding quantities after the M-step on the next iteration (the jth) are denoted by π_i^+, μ_i^+, and Σ_i^+, respectively. McLachlan and Ng (2000) showed that the latter can be computed in terms of the existing estimates as follows:

$$\pi_i^+ = (n\pi_i - \tau_{ij}^* + \tau_{ij})/n, \tag{14}$$

$$\mu_i^+ = \mu_i - (\tau_{ij}^* - \tau_{ij})(\boldsymbol{y}_j - \mu_i)/(n\pi_i^+), \tag{15}$$

$$\Sigma_i^{+^{-1}} = \frac{\pi_i^+}{\pi_i}\left[\Sigma_i^{-1} + \frac{(\tau_{ij}^* - \tau_{ij})\Sigma_i^{-1}(\boldsymbol{y}_j - \mu_i)(\boldsymbol{y}_j - \mu_i)^T\Sigma_i^{-1}}{n\pi_i^+ - (\tau_{ij}^* - \tau_{ij})(\boldsymbol{y}_j - \mu_i)^T\Sigma_i^{-1}(\boldsymbol{y}_j - \mu_i)}\right], \tag{16}$$

$$|\Sigma_i^+| = \left(\frac{\pi_i}{\pi_i^+}\right)^p |\Sigma_i| \left[1 - \frac{\tau_{ij}^* - \tau_{ij}}{n\pi_i^+}(\boldsymbol{y}_j - \mu_i)^T\Sigma_i^{-1}(\boldsymbol{y}_j - \mu_i)\right] \tag{17}$$

for $i = 1, \ldots, g$. The use of (14) to (17) considerably reduces the amount of computation time in the updating of the g component-posterior probabilities for $\boldsymbol{y}_j$.

3. Simulation results for the IEM algorithm

In this section, we discuss the results of two simulations performed to compare the performance of the IEM algorithm with different number of blocks B relative to the standard EM algorithm. For the standard EM algorithm, we used the stopping criterion that the change in the log likelihood from the current scan and that from ten scans previously differs by less than 0.000001 of the current log likelihood value (McLachlan *et al.* 1999). For the IEM algorithm, the log likelihood after a complete scan of all the data is approximated by (13) and the convergence criterion is the same as that of the standard EM algorithm, based on the change in the log likelihood after a complete scan of the data. All simulations are run on a Sun unix workstation with enough memory that there is no paging activity.

3.1. *Simulation 1*

A sample of size $n = 256 \times 256$ was generated from a seven-component trivariate normal mixture. The estimates obtained in Liang, MacFall and Harrington (1994) were used as the values of our population parameters. They are displayed in Table 1, where μ_{ip} and σ_{ip}^2 $(p = 1, 2, 3)$ are, respectively, the means and variances of the trivariate normal distribution associated with the ith group. The correlation coefficients between the variates are denoted by ρ_{i12}, ρ_{i13}, and ρ_{i23}. These seven components correspond to seven tissue types in the segmentation of a two-dimensional magnetic resonance (MR) image of the human brain.

The standard EM algorithm and the IEM version for various number of blocks B were fitted to this simulated set, using the same random start and assuming unrestricted component-covariance matrices. In order to avoid the problem of premature component starvation with the IEM algorithm (Thiesson, Meek and Heckerman 2001), a full E-step ($B = 1$) was performed before running the initial M-step.

With this implementation, the EM and IEM algorithms converged for a given start to the same local maximum of the log likelihood. The overall CPU time (in seconds), the number of scans and the overall CPU time per scan are displayed in Table 2. In addition, the average CPU times of the E-step (T_E) and the M-step (T_M) for each scan are displayed in parentheses. It can be seen that when the data are partitioned into blocks with the IEM algorithm, the average times T_E and T_M increase, but the number of scans to convergence decreases, relative to the standard EM algorithm. The time to convergence for the IEM version against the standard EM algorithm is a tradeoff between the additional computation time per scan and the fewer number of scans required because of the more frequent updating due to B partial E-steps instead of one full E-step. As the data are partitioned into more and more blocks, the time to convergence eventually starts to rise. From Table 2, it can be seen that the fastest time to convergence with the IEM algorithm was obtained with $B = 64$. For this value of B, the time to convergence is reduced by a factor of 33% compared to the standard EM algorithm.

For the extreme case of $B = n$ blocks, the convergence time of the IEM algorithm for this data set was reduced by a factor of 51% with the use of the updating formulas. They are useful in situations where data actually appear one at a time and hence $B = n$ blocks are adopted.

Table 1. *The proportions, intensity means, variances, and correlation coefficients of seven groups* ($i = 1, \ldots, 7$)

i	π_i	μ_{i1}	μ_{i2}	μ_{i3}	σ_{i1}^2	σ_{i2}^2	σ_{i3}^2	ρ_{i12}	ρ_{i13}	ρ_{i23}
1	0.06	1.50	1.00	2.48	1.09	0.48	2.37	0.55	0.38	0.74
2	0.05	4.96	8.06	10.17	6.91	10.46	17.62	0.22	0.27	0.95
3	0.11	5.30	3.25	8.01	3.19	1.90	4.74	0.43	0.42	0.79
4	0.08	6.53	12.92	15.00	2.55	6.39	0.92	−0.41	0.09	0.17
5	0.37	8.23	9.57	14.53	0.65	1.89	1.52	−0.52	−0.29	0.73
6	0.11	9.39	3.42	7.70	12.24	2.95	14.17	0.80	0.81	0.95
7	0.22	9.43	7.93	12.58	0.16	0.48	0.44	−0.12	0.26	0.49

Table 2. *CPU times (in seconds) and the number of scans for the standard EM algorithm and the IEM version for B Blocks (Simulation 1)*

Algorithm	CPU times overall (T_E, T_M)	No. of scans	Overall CPU (Time per scan)
Standard EM	601.0 (4.76, 1.07)	101	5.950
Incremental EM			
$B = 4$	458.4 (4.89, 1.26)	72	6.367
$B = 8$	426.8 (4.89, 1.26)	67	6.370
$B = 16$	414.3 (4.89, 1.26)	65	6.374
$B = 32$	408.0 (4.90, 1.25)	64	6.375
$B = 64$	404.6 (4.90, 1.28)	63	6.422
$B = 128$	406.6 (4.92, 1.28)	63	6.454
$B = 256$	410.9 (4.94, 1.32)	63	6.522
$B = 256^2 = n$	2352 (28.40, 6.87)	63	37.33
$B = 256^2$ with updating formulas	1143 (10.34, 6.80)	63	18.14

3.2. *Simulation 2*

A sample of size $n = 2000$ was generated from a four-component eight-dimensional normal mixture. The population parameters are those described in Fukunaga (1990, pp. 46), given as follows. The mixing proportions are $\pi_1 = \pi_2 = \pi_3 = 0.2$ and $\pi_4 = 0.4$. The component mean μ_1 is the null vector, while μ_2 and μ_3 have all elements zero, apart from their first, which is equal to 2.56 and 1, respectively; μ_4 is equal to

$$\mu_4 = (3.86, 3.1, 0.84, 0.84, 1.64, 1.08, 0.26, 0.01)^T.$$

The component-covariance matrices are all diagonal with Σ_1 and Σ_2 being equal to $\boldsymbol{I}_p$ and Σ_3 equal to $4\boldsymbol{I}_p$, where $\boldsymbol{I}_p$ is the $p \times p$ identity matrix; Σ_4 is given by

$$\Sigma_4 = \text{diag}(8.41, 12.06, 0.12, 0.22, 1.49, 1.77, 0.35, 2.73).$$

Table 3. *CPU times (in seconds) and the number of scans for the standard EM algorithm and the IEM version for B blocks (Simulation 2)*

Algorithm	CPU times Overall (T_E, T_M)	No. of scans	Overall CPU (Time per scan)
Standard EM	185.8 (0.34, 0.07)	446	0.417
Incremental EM			
$B = 4$	136.8 (0.37, 0.10)	287	0.477
$B = 8$	119.1 (0.37, 0.10)	249	0.478
$B = 16$	111.9 (0.38, 0.10)	230	0.487
$B = 20$	109.0 (0.39, 0.10)	218	0.500
$B = 40$	113.1 (0.40, 0.11)	215	0.526
$B = 80$	117.9 (0.42, 0.12)	213	0.554
$B = 200$	144.0 (0.52, 0.14)	210	0.686
$B = 400$	195.9 (0.68, 0.20)	210	0.933
$B = 2000 = n$	510 (1.82, 0.47)	210	2.429
$B = 2000$ with updating formulas	249 (0.63, 0.49)	210	1.186

The standard EM algorithm and the IEM version for various number of blocks B were fitted to this simulated set, using the same random start and assuming unrestricted component-covariance matrices. In order to avoid the problem of premature component starvation with the IEM algorithm, a full E-step ($B = 1$) was performed before running the initial M-step.

With this implementation, the EM and IEM algorithms converged for a given start to the same local maximum of the log likelihood. The results are presented in Table 3. It can be seen that when the data are partitioned into blocks with the IEM algorithm, the overall CPU time decreases, but eventually starts to rise as the data are partitioned into more and more blocks. From Table 3, it can be seen that the fastest time to convergence with the IEM algorithm was obtained with $B = 20$. For this value of B, the time to convergence is reduced by a factor of 41% compared to the standard EM algorithm. For the extreme case of $B = n$ blocks, the use of the updating formulas again considerably reduces the amount of computation time. For this data set, the convergence time of the IEM algorithm was reduced by a factor of 51%.

4. Choosing the number of blocks

Concerning the time taken to perform the IEM algorithm for one scan, the B partial E-steps take more time to implement than the one full E-step of the standard EM algorithm, the additional time involving the subtraction of the second term on the right-hand side of (6) and the inversion of the component-covariances matrices, which have to be performed at each of the B partial steps. Also, one scan of the IEM algorithm requires $(B - 1)$ additional M-steps in updating the estimates using (10) to (12). The time to convergence for the IEM algorithm against the standard EM algorithm is a tradeoff between this additional computation time of the IEM algorithm and the fewer number of scans required because of the more frequent updating after each partial E-step. An initial work on investigating the times to convergence for the IEM algorithm with various numbers of blocks B can be found in McLachlan and Ng (2000).

We let T be the computation time per scan of the (standard) EM, which can be decomposed as

$$\begin{aligned} T &= T_E + T_M \\ &= T_{E1} + T_{E2} + T_M, \end{aligned} \tag{18}$$

where T_E and T_M denote the time spent on the E- and M-steps, respectively, and where T_{E2} denotes that time of the E-step spent on inverting the component-covariance matrices in updating the posterior probabilities of component membership, and T_{E1} denotes the remainder of the time spent on the E-step. The latter time is devoted essentially to the updating of the conditional expectations of the sufficient statistics.

Then the time per scan of the IEM algorithm is given approximately by

$$T_{\text{IEM}} \approx T_A + T_{E1} + BT_{E2} + BT_M, \tag{19}$$

where T_A denotes an additional time due to the different programming code for the IEM algorithm. The time spent on updating the conditional expectations of the sufficient statistic is T_{E1} rather than BT_{E1}, as the updating can be confined effectively to a block of observations per scan as explained above; see (6). It follows from (19) that T_{IEM} can be expressed approximately as

$$T_{\text{IEM}} \approx (u + Bv)T, \tag{20}$$

where

$$u = (T_A + T_{E1})/T \tag{21}$$

and

$$v = (1 - T_{E1}/T). \tag{22}$$

We now let

$$s_B = s_{\text{IEM}}/s_{\text{EM}} \tag{23}$$

denote the ratio of the number of scans to convergence of the IEM and EM algorithms. It follows then that the proportionate reduction R in time to convergence with the IEM compared to the EM algorithm is given approximately by

$$\begin{aligned} R &\approx (s_{\text{EM}}T - s_{\text{IEM}}T_{\text{IEM}})/(s_{\text{EM}}T) \\ &= 1 - s_B(u + Bv). \end{aligned} \tag{24}$$

As B increases, s_B decreases and reaches a minimum at the maximum value n for B. Thus in order to maximize R, the aim is to decrease B in the right-hand side of (24) as much as possible without increasing s_B too much above its minimum value. In this way, we investigated empirically the optimal value B_{opt} for B. It appears in our simulations that log B_{opt} is proportional to log n approximately; that is,

$$B_{opt} \approx n^c, \tag{25}$$

Table 4. *Simulation results for choosing the number of blocks*

Set	n	g	p	Σ_i	B_{opt}	T_{opt}	R_{opt}	R^*	R_{TMH}	$R^*_{3/8}$
1	16,384	7	3	Unrestricted	32	146.6	0.34	0.33	0.31	0.34
2	32,768	7	3	Unrestricted	64	173.7	0.35	0.35	0.34	0.32
3	65,536	7	3	Unrestricted	64	404.6	0.33	0.33	0.33	0.33
4	30,000	4	8	Unrestricted	48	475.9	0.28	0.27	0.25	0.28
				Equal	50	644.3	0.35	0.33	0.35	0.33
				Diagonal	48	350.2	0.39	0.38	0.38	0.39
5	50,000	4	8	Unrestricted	80	1349.8	0.28	0.28	0.28	0.28
				Equal	80	830.3	0.38	0.37	0.37	0.37
				Diagonal	40	784.7	0.19	0.19	0.16	0.18
6	100,000	4	8	Unrestricted	80	1340.6	0.29	0.28	0.27	0.29
				Equal	100	2217.2	0.34	0.34	0.31	0.34
				Diagonal	40	1262.8	0.20	0.18	0.17	0.18
7	30,000	3	20	Unrestricted	60	758.8	0.27	0.27	0.27	0.26
				Equal	50	1390.4	0.33	0.33	0.33	0.33
				Diagonal	40	824.3	0.19	0.18	0.18	0.18
8	50,000	3	20	Unrestricted	80	1158.0	0.22	0.22	0.20	0.20
				Equal	50	2171.0	0.34	0.34	0.34	0.34
				Diagonal	40	1219.4	0.30	0.30	0.30	0.30
9	100,000	3	20	Unrestricted	80	2342.7	0.24	0.23	0.21	0.24
				Equal	125	4056.1	0.32	0.31	0.31	0.31
				Diagonal	50	2473.6	0.20	0.20	0.18	0.20
10	30,000	25	2	Unrestricted	60	1882.9	0.43	0.43	0.43	0.43
				Equal	50	1225.5	0.63	0.62	0.63	0.62
				Diagonal	48	2601.9	0.34	0.32	0.34	0.34
11	50,000	25	2	Unrestricted	100	3701.1	0.34	0.32	0.32	0.31
				Equal	80	2416.2	0.72	0.71	0.71	0.71
				Diagonal	40	4100.9	0.65	0.65	0.61	0.61
12	100,000	25	2	Unrestricted	125	10401.3	0.32	0.31	0.31	0.32
				Equal	80	3736.0	0.45	0.45	0.43	0.45
				Diagonal	50	6665.7	0.55	0.55	0.54	0.54

where c depends on the model adopted for the component-covariance matrices. For example, from the simulation result of Set 1 in Table 2, we approximated u by $(2 * 6.367 - 6.370)/5.950 = 1.07$ and took $s_B = 63/101 = 0.62$, $R = (601.0 - 404.6)/601.0 = 0.33$, and $v = (37.33/5.950 - u)/n = 0.00008$ in (24) and (25) to give $c = 0.44$.

We considered the optimal choice of c empirically by performing some simulations for various combinations of the parameters n, g, and p. The results are presented in Table 4, where B_{opt} (a factor of n) is the optimal number of blocks obtained from the simulation, T_{opt} is the corresponding overall CPU times, and R_{opt} is the corresponding reduction in time to convergence as a proportion of the convergence time for the standard EM algorithm.

The first three sets in Table 4 used the population parameters of Set 1 in Section 3 ($g = 7$, $p = 3$) with different values of n. Sets 4 to 6 used the population parameters of Set 2 in Section 3 ($g = 4$, $p = 8$) with different values of n. We considered a higher dimensional data $p = 20$ in Sets 7 to 9 and a larger number of groups $g = 25$ in Sets 10 to 12. For Set 4 to Set 12, we fitted the simulated sets by assuming unrestricted, equal, and diagonal component-covariance matrices respectively. It was observed in the simulations that v increased when the component-covariance matrices were restricted to being equal or diagonal. It therefore follows that, to achieve the same proportionate reduction R in convergence time, the values of c need to be smaller than in the case of unrestricted covariance matrices. From our simulations, we propose $B^* = \text{round}(n^{2/5})$, $B^* = \text{round}(n^{3/8})$, and $B^* = \text{round}(n^{1/3})$ for unrestricted, equal, and diagonal component-covariance matrices, respectively, where round (r) rounds r to the nearest integer. However, the value of the integer B^* obtained usually is not a factor of the sample size n. Thus, in our analysis, we choose the number of blocks B which is a factor of n and is the closest to B^*. The corresponding reduction in time to convergence is denoted as R^* in Table 4.

Thiesson, Meek and Heckerman (2001) suggest a search method to choose the number of blocks. For a given number of blocks B, they propose to run the IEM algorithm for two scans (the first scan involves a full E-step) and calculate the ratio

$$r = (L_2 - L_1)/t,$$

where L_1 and L_2 are the log likelihood values after the first and the second complete scan of the data respectively, and t is the time required for the second scan which involves the partial E-step. The procedure is repeated for various number of blocks B and the choice of B is that maximizes r. The corresponding reduction in time to convergence is denoted as R_{TMH} in Table 4. It can be seen from Table 4 that our proposed rule is not only simple, but also works very well in the simulations. With our formulation of the rule for choosing B_{opt}, even though it may be possible to obtain a better estimate of B_{opt} if v can be related to the values of p and g for the data set, our focus here is on a simple rule to guide in the choice of B. As Thiesson, Meek and Heckerman (2001) pointed out, the reduction in time to convergence versus the number of blocks exhibited a single broad peak around the optimal number of blocks. It follows that there is no requirment to identify the exact optimal number of blocks in practice, and hence, our simple rule may also be good enough as a guide for applications where a different type of models is considered. For example, it is interested to note that if we choose $B^* = \text{round}(n^{3/8})$ for unrestricted or diagonal component-covariance matrices, the reductions in time to convergence, denoted as $R^*_{3/8}$ in Table 4, are still very close to R_{opt}.

5. Sparse IEM algorithm

In fitting a mixture model by maximum likelihood via the EM algorithm, it is often observed that the posterior probabilities for some components of the mixture for a given data point $\boldsymbol{y}_j$ are close to zero (for example, $\tau_i(\boldsymbol{y}_j; \Psi^{(k)}) < C = 0.01$, where $\Psi^{(k)}$ is the value of Ψ after the kth iteration of the EM algorithm). With the Sparse EM (SPEM) algorithm proposed by Neal and Hinton (1998), their posterior probabilities of component membership are held fixed, while only those posterior probabilities for the remaining components in the mixture are updated. To examine this more closely, suppose that the SPEM version is to be implemented on the $(k+1)$th scan, where the current estimates of the posterior probabilities of component membership of $\boldsymbol{y}_j$ are close to zero and held fixed for components i with $i \in A_j$, where A_j is a subset of $\{1, \dots, g\}$. Then on the E-step on the $(k+1)$th iteration of the SPEM algorithm, if A_j is the null set for observation $\boldsymbol{y}_j$, update the posterior probabilities of component membership to $\tau_i(\boldsymbol{y}_j; \Psi^{(k)})(i = 1, \dots, g)$. If A_j is not the null set, then update the posterior probabilities of component membership to

$$\sum_{h \notin A_j} \tau_h\big(\boldsymbol{y}_j; \Psi^{(k-1)}\big) \frac{\tau_i\big(\boldsymbol{y}_j; \Psi^{(k)}\big)}{\sum_{h \notin A_j} \tau_h\big(\boldsymbol{y}_j;\ \Psi^{(k)}\big)} \tag{26}$$

for those components i which do not belong to A_j; otherwise do not update the posterior probabilities $\tau_i(\boldsymbol{y}_j; \Psi^{(k-1)})$. This sparse E-step will take time proportional only to the number of components $i \notin A_j$ $(j = 1, \dots, n)$. The calculation of the conditional expectations of the sufficient statistics can also be done efficiently by updating only the contribution to the sufficient statistics for those components $i \notin A_j$. For example,

$$T_{i1}^{(k)} = \sum_{j=1}^{n} I_{A_j}(i)\tau_i\big(\boldsymbol{y}_j; \Psi^{(k-1)}\big) + \sum_{j=1}^{n} I_{A_j^c}(i)\tau_i\big(\boldsymbol{y}_j; \Psi^{(k)}\big), \tag{27}$$

where A_j^c is the complement of A_j and $I_{A_j}(i)$ is the indicator function for the set A_j. The first term on the right-hand side of (27) can be saved for use in the subsequent SPEM iterations. Similar arguments apply to $T_{i2}^{(k)}$ and $T_{i3}^{(k)}$. After running the sparse version a number of iterations k_1, a standard EM iteration is then performed, and a new set A_j $(j = 1, \dots, n)$ is selected.

A sparse version of the IEM algorithm (SPIEM) can be formulated by combining the sparse E-step of the SPEM algorithm and the partial E-step of the IEM algorithm. Suppose that the SPEM is to be implemented on the subsequent B iterations of the $(k+1)$th scan. Then on the E-step of the $(b+1)$th iteration for $b = 0, \dots, B-1$, the posterior probabilities of component membership for all $j \in S_{(b+1)}$ are updated to

$$\sum_{h \notin A_j} \tau_h\big(\boldsymbol{y}_j; \Psi^{(k-1+b/B)}\big) \frac{\tau_i\big(\boldsymbol{y}_j; \Psi^{(k+b/B)}\big)}{\sum_{h \notin A_j} \tau_h\big(\boldsymbol{y}_j; \Psi^{(k+b/B)}\big)}$$

for those components i which do not belong to A_j; otherwise leave the posterior probabilities unchange as $\tau_i(\boldsymbol{y}_j; \Psi^{(k-1+b/B)})$.

To avoid the problem of premature component starvation, we suggest running a standard EM iteration for the first scan and then performing the IEM for five scans before running the sparse version SPIEM. After running the spare version for a number of scans k_1, the IEM is performed for a scan, and a new set A_j $(j = 1, \dots, n)$ is selected. The efficient step (27) described above is also applicable for the SPIEM algorithm.

6. Simulation results for the SPIEM algorithm

In this section, we report some simulations performed to compare the relative performances of the SPIEM, IEM, and the standard EM algorithms. Moreover, we assess the quality of the estimates obtained by the SPIEM algorithm, relative to the standard EM algorithm, by comparing the final log likelihood values. The stopping criteria for the standard EM and the IEM algorithms are described in Section 4. For the SPIEM algorithm, the convergence is based on the change in the log likelihood after a complete scan of the data with full EM steps (B iterations of IEM without SPIEM steps).

6.1. *Simulation 1*

The first simulated data used in Section 3 is considered again here. The number of blocks B is set to be 64 and we consider the combinations of $k_1 = 2, 3, 4, 5, 6$ and $C = 0.05, 0.01, 0.005$. For this simulated data, all the algorithms converged to the same local maximum of the log likelihood. The overall CPU time, the average times for the E-step (T_E) and M-step (T_M), and the number of scans are displayed in Table 5.

Table 5. *CPU times (in seconds) and the number of scans for the SPIEM algorithm for $B = 64$ blocks (Simulation 1)*

k_1	C	CPU times Overall (T_E, T_M)	No. of scans
2	0.05	323 (2.88, 0.75)	83
2	0.01	319 (3.31, 0.88)	71
2	0.005	325 (3.39, 0.89)	71
3	0.05	316 (2.62, 0.69)	89
3	0.01	300 (3.03, 0.80)	73
3	0.005	308 (3.12, 0.82)	73
4	0.05	343 (2.42, 0.64)	105
4	0.01	318 (2.84, 0.74)	83
4	0.005	313 (2.97, 0.78)	78
5	0.05	325 (2.25, 0.59)	107
5	0.01	304 (2.82, 0.71)	80
5	0.005	291 (2.84, 0.72)	77
6	0.05	369 (2.22, 0.57)	124
6	0.01	312 (2.73, 0.69)	85
6	0.005	315 (2.82, 0.71)	83

On comparing Table 5 with Table 2 for $B = 64$, it can be seen that the number of scans to convergence for each combination of C and k_1 with the SPIEM algorithm is larger than that for the IEM algorithm. However, as to be anticipated, T_E and T_M for the SPIEM algorithm are smaller than that for the IEM algorithm, which allows the SPIEM algorithm for the present combinations of C and k_1 to converge faster than the IEM algorithm. Concerning the optimal choice of C and k_1, the overall CPU time is a tradeoff between the computation time per scan and the number of scans required. It can be seen from Table 5 that the SPIEM algorithm converges fastest for $k_1 = 5$ and $C = 0.005$. With these values of the tuning constants, the time to convergence is reduced by a factor of 52% and 28%, compared with the standard EM algorithm and the IEM algorithm for $B = 64$, respectively.

In Fig. 1, the log likelihood is plotted against the number of scans for the various algorithms. It shows how the log likelihood changes in the initial fifty scans. It is noted that for the implementation of the SPIEM algorithm, the IEM algorithm is performed for the first five scans before running the first SPIEM step. Thus in Fig. 1, the curves corresponding to the IEM and the SPIEM algorithms coincide up to the fifth scan. From Fig. 1, it is observed that the SPIEM algorithm takes more scans to converge compared with the IEM algorithm. But, both the SPIEM and the IEM algorithms take less scans when compared to the standard EM algorithm. In Fig. 2, the log likelihood is plotted against the elapsed time for the various algorithms. It shows how the log likelihood changes during the initial two hundred and fifty seconds.

6.2. *Simulation 2*

The second simulated data used in Section 3 is considered again here. The number of blocks B is set to be 20 and the

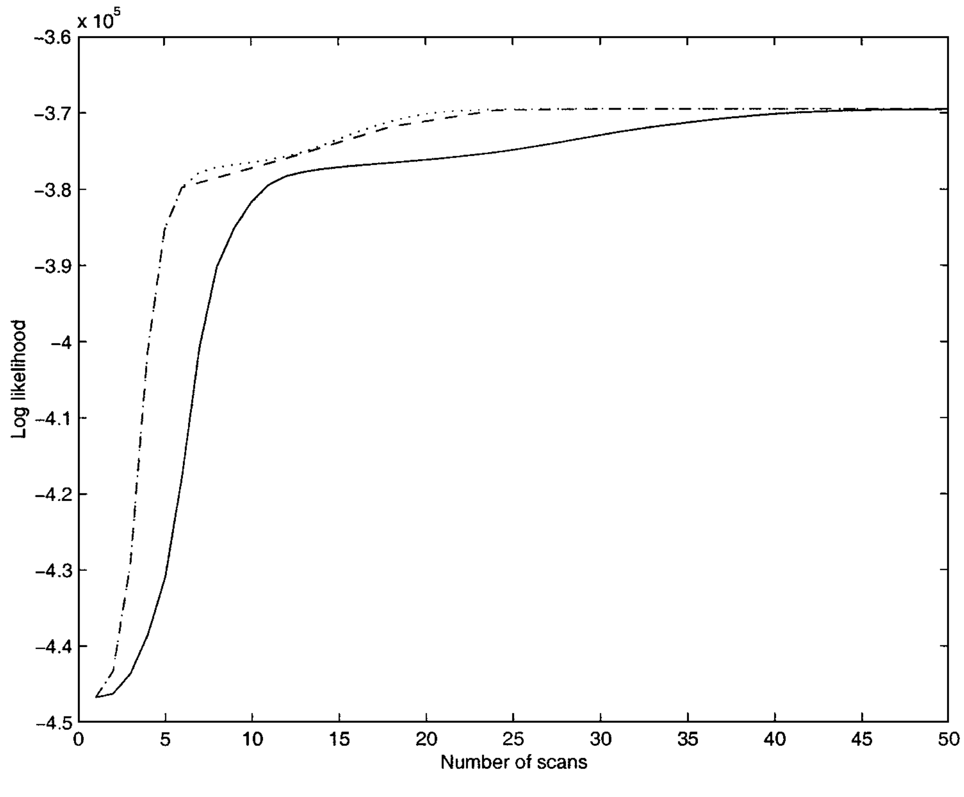

Fig. 1. *Log likelihood versus number of scans. Standard EM (solid curve); Incremental EM with $B = 64$ (dotted curve); SPIEM with $B = 64$, $k_l = 5$, and $C = 0.005$ (dashed curve)*

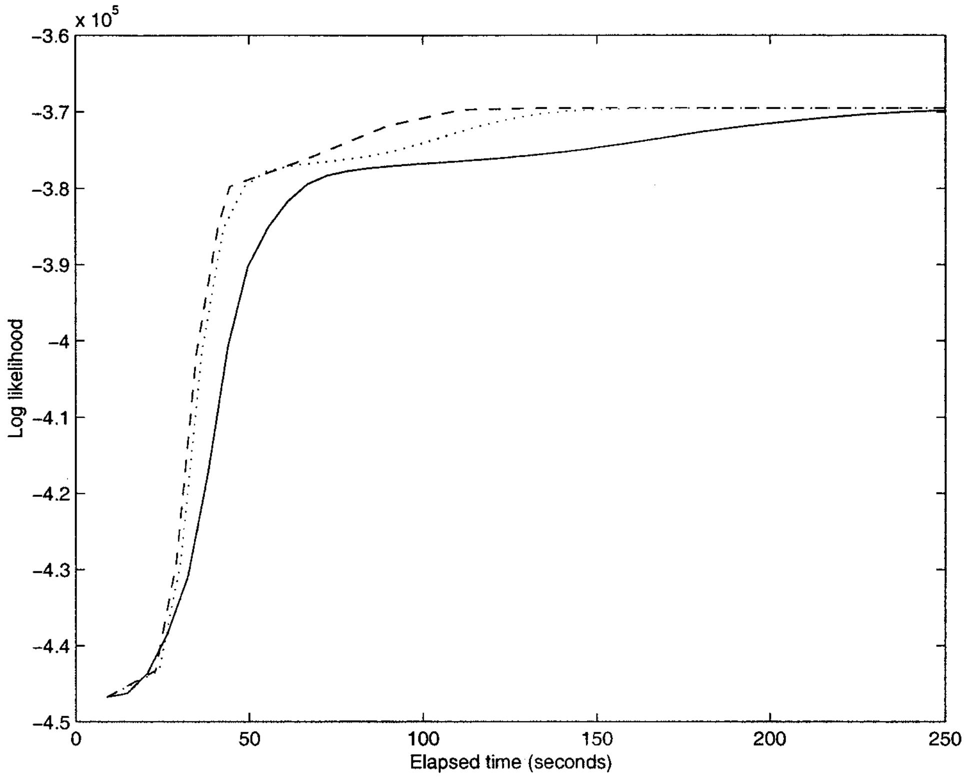

Fig. 2. *Log likelihood versus elapsed time. Standard EM (solid curve); Incremental EM with $B = 64$ (dotted curve); SPIEM with $B = 64$, $k_1 = 5$, and $C = 0.005$ (dashed curve)*

combinations of $k_1 = 2, 3, 4, 5, 6$ and $C = 0.05, 0.01, 0.005$ are considered. The results are displayed in Table 6. For this simulated data, some runs of the SPIEM algorithm with $C = 0.05$ and $C = 0.01$ converge to a slightly smaller local maximum of the log likelihood value. It implies that an appropriate choice of the tuning constants of the SPIEM algorithm is required.

Table 6. *CPU times (in seconds) and the number of scans for the SPIEM algorithm for $B = 20$ blocks (Simulation 2)*

k_1	C	CPU times Overall (T_E, T_M)	No. of scans	Log likelihood
2	0.05	96 (0.30, 0.07)	254	−30384.7
2	0.01	86 (0.34, 0.08)	203	−30381.6
2	0.005	91 (0.35, 0.09)	203	−30381.6
3	0.05	95 (0.27, 0.07)	273	−30384.7
3	0.01	84 (0.31, 0.08)	209	−30381.6
3	0.005	86 (0.34, 0.08)	201	−30381.6
4	0.05	100 (0.26, 0.07)	300	−30384.7
4	0.01	84 (0.30, 0.07)	215	−30381.6
4	0.005	85 (0.32, 0.08)	205	−30381.6
5	0.05	100 (0.25, 0.06)	317	−30384.7
5	0.01	83 (0.30, 0.07)	221	−30381.6
5	0.005	81 (0.31, 0.08)	203	−30381.6
6	0.05	104 (0.24, 0.06)	341	−30384.7
6	0.01	90 (0.29, 0.07)	243	−30384.7
6	0.005	82 (0.31, 0.07)	208	−30381.6

On comparing Table 6 with Table 3 for $B = 20$, it can be seen that T_E and T_M for the SPIEM algorithm are smaller for each combination of C and k_1 considered than that for the IEM algorithm, which is to be expected. For some of these combinations of C and k_1, the SPIEM algorithm required a fewer number of scans to convergence, while for all combinations its time to convergence is smaller than that of the IEM algorithm. As in the first simulation, the convergence time of the SPIEM algorithm is smallest for $k_1 = 5$ and $C = 0.005$. With these values of the tuning constants, the time to convergence is reduced by a factor of 56% and 26%, compared with the standard EM algorithm and the IEM algorithm for $B = 20$, respectively. In particular, the sparse E-steps does not introduce bias to the estimates of the parameters and the final log likelihood obtained is the same as that from the standard EM algorithm.

7. Conclusions

We have compared the relative performances of the IEM algorithm with various numbers of blocks B and the standard EM algorithm. It can be seen from Sections 3 and 4 that the IEM algorithm with an appropriate choice of B can significantly reduce the computation times. The time to convergence is found to be reduced by a factor of 19% to 71%, compared with the standard EM algorithm (Table 4). A simple guide for choosing the optimal number of blocks B is proposed: $B \approx n^{2/5}$, $B \approx n^{3/8}$, and $B \approx n^{1/3}$ for unrestricted, equal, and diagonal component-covariances matrices respectively. Alternatively, we may choose $B \approx n^{3/8}$ for the above different assumptions of the component-covariances matrices. As presented in Section 4, this simple rule works very well in the simulations. For the extreme case of $B = n$ blocks, the convergence time of the IEM algorithm is found to be reduced by a factor of 51% with the use of the updating formulas. These formulas are useful in applications where data actually appear one at a time and $B = n$ blocks are adopted.

Further reduction in the time to convergence can be achieved by using a sparse version of the IEM algorithm. For the implementation of this SPIEM algorithm, we suggest choosing the number of blocks B obtained from the proposed simple rule and taking $C = 0.005$ and $k_1 = 5$. For these values of C and k_1, the time to convergence is reduced by a factor of 52% for the first simulated data set, compared with the standard EM algorithm, and a factor of 28%, compared with the IEM algorithm for $B = 64$. For the second simulation, the time to convergence is reduced by a factor of 56%, compared with the standard EM algorithm, and a factor of 26%, compared with the IEM algorithm for $B = 20$.

Other approaches for speeding up the EM algorithm for mixtures have been considered in Bradley, Fayyad and Reina (1999) and Moore (1999). The former developed a scalable version of the EM algorithm to handle very large databases with a limited memory buffer. It is based on identifying regions of the data that are compressible and regions that must be maintained in memory; see for example McLachlan and Peel (2000, ch. 12). Moore (1999) has made use of multiresolution *k*d-trees (*mrk*d-trees) to speed up the fitting process of the EM algorithm. Here *k*d stands for *k*-dimensional where, in our notation, $k = p$, the dimension of a feature vector $\boldsymbol{y}_j$. His approach builds a multiresolution data structure (*mrk*d-tree) to summarize the database at all resolutions of interest simultaneously. The *mrk*d-tree is a binary tree that recursively splits the whole set of data points into partitions. Each non-leaf-node has two children nodes, which divide their parent's data points between them. The leaf-nodes are the smallest possible partitions this *mrk*d-tree offers. It can be shown that the relative time to convergence, compared to the standard EM algorithm, is roughly proportional to the ratio of the number of leaf-nodes to the number of data points n. Further reduction in the computation time can be achieved by pruning the tree. At a given node, if the minimum and maximum values that any point $\boldsymbol{y}_j$ in the node can have for its current posterior probabilities $\tau_i(\boldsymbol{y}_j; \Psi^{(k)})$ are close and satisfy some pruning criterion for all $i = 1, \ldots, g$ (see for example Moore 1999), the node is treated as if it is a leaf node and its descendents need not be searched at this iteration.

Neither the scalable EM algorithm of Bradley, Fayyad and Reina (1999) nor the *mrk*d-tree proposed by Moore (1999) maintain the desirable convergence guarantees of the standard EM algorithm. Moreover, McCallum, Nigam and Ungar (2000) point out that the scalable EM algorithm becomes less efficient when the number of groups g is large, and the *mrk*d-trees-based approach slows down as the dimension p increases; see also Thiesson, Meek and Heckerman (2001).

References

Bradley P.S., Fayyad U.M., and Reina C.A. 1998. Scaling EM (expectation-maximization) clustering to large databases. Technical Report No. MSR-TR-98-35 (revised February, 1999), Microsoft Research, Seattle.

Dempster A.P., Laird N.M., and 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.

Friedman J.H. 1989. Regularized discriminant analysis. Journal of the American Statistical Association 84: 165–175.

Fukunaga K. 1990. Statistical Pattern Recognition, 2nd edn. Academic Press, New York.

Jamshidian M. and Jennrich R.I. 1993. Conjugate gradient acceleration of the EM algorithm. Journal of the American Statistical Association 88: 221–228.

Jamshidian M. and Jennrich R.I. 1997. Acceleration of the EM algorithm by using quasi-Newton methods. Journal of the Royal Statistical Society Series B 59: 569–587.

Liang Z., MacFall J.R., and Harrington D.P. 1994. Parameter estimation and tissue segmentation from multispectral MR images. IEEE Transactions on Medical Imaging 13: 441–449.

McCallum A., Nigam K., and Ungar L.H. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In: Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, pp. 169–178.

McLachlan G.J. and Krishnan T. 1997. The EM Algorithm and Extensions. Wiley, New York.

McLachlan G.J. and Ng S.K. 2000. A sparse version of the incremental EM algorithm for large databases. Technical Report, Centre of Statistics, University of Queensland.

McLachlan G.J. and Peel D. 2000. Finite Mixture Models. Wiley, New York.

McLachlan G.J., Peel D., Basford K.E., and Adams P. 1999. The EMMIX software for the fitting of mixtures of normal and t-components. Journal of Statistical Software 4(2).

Meilijson I. 1989. A fast improvement to the ECM algorithm on its own terms. Journal of the Royal Statistical Society Series B 51: 127–138.

Meng X.L. 1994. On the rate of convergence of the ECM algorithm. Annals of Statistics 22: 326–339.

Meng X.L. and Rubin D.B. 1993. Maximum likelihood estimation via the ECM algorithm: A general framework. Biometrika 80: 267–278.

Meng X.L. and van Dyk D. (1997). The EM algorithm—An old folk-song sung to a fast new tune (with discussion). Journal of the Royal Statistical Society Series B 59: 511–567.

Moore A.W. 1999. Very fast EM-based mixture model clustering using multiresolution kd-trees. In: Kearns M.S., Solla S.A., and

Cohn D.A. (Eds.), Advances in Neural Information Processing Systems 11. MIT Press, Cambridge, MA, pp. 543–549.

Neal R.M. and Hinton G.E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan M.I. (Ed.), Learning in Graphical Models. Kluwer, Dordrecht, pp. 355–368.

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

Thiesson B., Meek C., and Heckerman D. 2001. Accelerating EM for large databases. Machine Learning 45: 279–299.