AUSTRIAN JOURNAL OF STATISTICS
Volume 32 (2003), Number 1&2, 143–161

On some Variants of the EM Algorithm for the Fitting of Finite Mixture Models

Shu-Kay Ng and Geoffrey J. McLachlan
The University of Queensland, Australia

Abstract: Finite mixture models are being increasingly used in statistical inference and to provide a model-based approach to cluster analysis. Mixture models can be fitted to independent data in a straightforward manner via the expectation-maximization (EM) algorithm. In this paper, we look at ways of speeding up the fitting of normal mixture models by using variants of the EM, including the so-called sparse and incremental versions. We also consider an incremental version based on imposing a multiresolution kd-tree structure on the data. Examples are given in which the EM algorithm can be speeded up by a factor of more than fifty for large data sets of small dimension.

Keywords: EM Algorithm, Finite Mixture Models, Multiresolution kd-tree, Sparse Incremental EM Algorithm.

1 Introduction

Finite mixture models have continued to receive increasing attention over the years, from both a practical and theoretical point of view. Indeed, in the past decade, the extent and the potential of the applications of finite mixture models in statistical inference and cluster analysis have widened considerably. Fields in which mixture models have been successfully applied include astronomy, biology, genetics, medicine, psychiatry, economics, engineering, and marketing, among many other fields in the biological, physical, and social sciences; see McLachlan and Peel (2000), Chapter 1 and the references therein.

The Expectation-Maximization (EM) algorithm of Dempster et al. (1977) is a popular tool for iterative maximum likelihood (ML) estimation of finite mixture distributions (McLachlan and Basford, 1998). 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. For most commonly used parametric formulations of finite mixture models, the use of the EM algorithm to find a local maximizer of the likelihood function is straightforward. However, a common criticism is that the convergence with the EM algorithm is only at a linear rate (Moore, 1999; Neal and Hinton, 1998). 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) shares all the appealing convergence properties of the EM algorithm (Meng, 1994).

In applications where the M-step is computationally simple, for example, in fitting multivariate normal mixtures, the rate of convergence of the EM algorithm depends mainly

on the computation time of the E-step. Because each E-step visits each data point, the EM algorithm requires much computation time in its application to large data sets. In this paper, we look at ways of speeding up the fitting of normal mixture models by using variants of the EM algorithm, including the so-called sparse and incremental versions proposed by Neal and Hinton (1998) and the multiresolution kd-tree approach proposed by Moore (1999). We also consider an incremental version based on imposing a multiresolution kd-tree structure on the data.

The rest of the paper is organized as follows: Section 2 describes the finite mixture models and the EM algorithm. In Section 3, we review the incremental version (IEM) and the sparse version (SPEM) of the EM algorithm proposed by Neal and Hinton (1998) We also formulate a combined version of these two variants (SPIEM algorithm) and introduce an inexact IEM algorithm proposed by Nowlan (1991). In Section 4, the multiresolution kd-tree approach is reviewed and in Section 5, we present how an IEM algorithm can be performed based on imposing a multiresolution kd-tree structure. Section 6 reports a comparative performance analysis of the variants of the EM algorithm, using some simulated and real data on medical magnetic resonance (MR) images. In Section 7, we end the paper by presenting some concluding remarks.

2 Finite Mixture Models and the EM Algorithm

With the mixture approach, the observed p-dimensional vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are assumed to have come from a mixture of a finite number, say g, of groups in some unknown proportions $\pi_1, \ldots, \pi_g$. The mixture density of $\boldsymbol{x}_j$ is expressed as

$$f(\boldsymbol{x}_j; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i f_i(\boldsymbol{x}_j; \boldsymbol{\theta}_i) \qquad (j = 1, \ldots, n), \tag{1}$$

where π_i $(i = 1, \ldots, g)$ sum to one and the group-conditional densities $f_i(\boldsymbol{x}_j; \boldsymbol{\theta}_i)$ are specified up to a vector $\boldsymbol{\theta}_i$ of unknown parameters $(i = 1, \ldots, g)$. Usually, the group-conditional densities are taken to belong to the same parametric family, for example, the normal. In this case,

$$f_i(\boldsymbol{x}; \boldsymbol{\theta}_i) = \phi(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

where $\phi(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the p-dimensional multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The vector of all the unknown parameters is given by $\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_g^T)^T$, where the superscript T denotes vector transpose, and the log likelihood for $\boldsymbol{\Psi}$ is then given by

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{n} \log\{\sum_{i=1}^{g} \pi_i \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}.$$

Solutions of the likelihood equation corresponding to local maxima can be found iteratively by application of the EM algorithm. Within the EM framework, each $\boldsymbol{x}_j$ is conceptualized to have arisen from one of the g groups. We let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ denote the unobservable group-indicator vectors, where the ith element z_{ij} of $\boldsymbol{z}_j$ is taken to be one or zero according as the jth feature vector $\boldsymbol{x}_j$ does or does not come from the ith group. We denote

the complete data by $\boldsymbol{y} = (\boldsymbol{x}^T, \boldsymbol{z}^T)^T$, where $\boldsymbol{x} = (\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_n^T)^T$ and $\boldsymbol{z} = (\boldsymbol{z}_1^T, \ldots, \boldsymbol{z}_n^T)^T$. The complete-data log likelihood for $\boldsymbol{\Psi}$ is given by

$$\log L_c(\boldsymbol{\Psi}) = \sum_{i=1}^{g} \sum_{j=1}^{n} z_{ij} \{\log \pi_i + \log \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}. \tag{2}$$

The EM algorithm proceeds iteratively in two steps, the expectation (E) step and the maximization (M) step. On the $(k+1)$th scan of the EM algorithm, the E- and M-steps are

- **E-step:** Compute the conditional expectation of the complete-data log likelihood, given $\boldsymbol{x}$ and the current estimates $\boldsymbol{\Psi}^{(k)}$. As the complete-data log likelihood (2) is linear in $\boldsymbol{z}$, we simply have to replace the unknown zero-one component-label variables z_{ij} by their current conditional expectations, $\tau_{ij}^{(k)}$, where

$$\tau_{ij}^{(k)} = \tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k)})$$

and

$$\begin{aligned} \tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}) &= E(Z_{ij} \mid \boldsymbol{x}; \boldsymbol{\Psi}) \\ &= \pi_i \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) / \sum_{l=1}^{g} \pi_l \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l) \end{aligned}$$

for $i = 1, \ldots, g$ and $j = 1, \ldots, n$. For mixtures with normal component densities, it is computationally advantageous to work in terms of the sufficient statistics. That is, the E-step calculates the current conditional expectations of the sufficient statistics

$$T_{i1}^{(k)} = \sum_{j=1}^{n} \tau_{ij}^{(k)}, \tag{3}$$

$$\boldsymbol{T}_{i2}^{(k)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} \boldsymbol{x}_j, \tag{4}$$

$$\boldsymbol{T}_{i3}^{(k)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} \boldsymbol{x}_j \boldsymbol{x}_j^T. \tag{5}$$

- **M-step:** Update the estimates to $\boldsymbol{\Psi}^{(k+1)}$, based on the current conditional expectations of the sufficient statistics, $T_{i1}^{(k)}$,$\boldsymbol{T}_{i2}^{(k)}$, and $\boldsymbol{T}_{i3}^{(k)}$, as follows:

$$\pi_i^{(k+1)} = T_{i1}^{(k)}/n, \tag{6}$$

$$\boldsymbol{\mu}_i^{(k+1)} = \boldsymbol{T}_{i2}^{(k)}/T_{i1}^{(k)}, \tag{7}$$

$$\boldsymbol{\Sigma}_i^{(k+1)} = \left\{ \boldsymbol{T}_{i3}^{(k)} - T_{i1}^{(k)^{-1}} \boldsymbol{T}_{i2}^{(k)} \boldsymbol{T}_{i2}^{(k)^T} \right\} / T_{i1}^{(k)}. \tag{8}$$

The E- and M-steps are alternated repeatedly until convergence; see for example McLachlan and Krishnan (1997), Chapter 1. Let $\hat{\boldsymbol{\Psi}}$ be the ML estimates for $\boldsymbol{\Psi}$. We can give an

outright or hard clustering of the data by assigning each $\boldsymbol{x}_j$ to the component of the mixture to which it has the highest estimated posterior probability of belonging. That is, z_{ij} is estimated by $\hat{z}_{ij}$, where

$$\begin{aligned}\hat{z}_{ij} &= 1, \text{ if } i = \arg\max_h \hat{\tau}_{hj}, \\ &= 0, \text{ otherwise},\end{aligned}$$

for $i = 1, \ldots, g; j = 1, \ldots, n$, and where

$$\hat{\tau}_{hj} = \hat{\pi}_h \phi(\boldsymbol{x}_j; \hat{\boldsymbol{\mu}}_h, \hat{\boldsymbol{\Sigma}}_h) / \sum_{l=1}^{g} \hat{\pi}_l \phi(\boldsymbol{x}_j; \hat{\boldsymbol{\mu}}_l, \hat{\boldsymbol{\Sigma}}_l).$$

It can be seen from (3) to (5) that the E-step is implemented for each feature vector $\boldsymbol{x}_j$. Hence the computational time spent in the E-step depends linearly on the number of observations and the number of groups in the finite mixture models. On the other hand, the time spent on the M-step ((6) to (8)) depends linearly only on the number of groups. For a huge data set, the computational cost in the E-step is therefore enormous, compared to that in the M-step.

3 Incremental and Sparse Versions of EM

3.1 The IEM Algorithm

With the IEM algorithm proposed by Neal and Hinton (1998), the available n observations are divided into B $(B \leq n)$ blocks and the E-step is implemented for only a block of data at a time before the next M-step is performed. A scan of the IEM algorithm thus consists of B partial E-steps and B M-steps. We let $\boldsymbol{\Psi}^{(k+b/B)}$ denote the estimate of $\boldsymbol{\Psi}$ after the bth iteration on the $(k+1)$th scan $(b = 1, \ldots, B)$.

- **(Partial) E-step:** For the first scan $(k = 0)$, a full E-step is performed to avoid premature component starvation (Ng and McLachlan, 2002; Thiesson et al., 2001); that is, we have $\boldsymbol{T}_{iq}^{(0)}$ $(q = 1, 2, 3)$ evaluated as in (3) to (5), with initial value $\boldsymbol{\Psi}^{(0)}$ for $\boldsymbol{\Psi}$. On subsequent scans, the conditional expectations of the sufficient statistics are calculated for only a block of observations at a time. 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 $T_{i1}^{(k+b/B)}$, $\boldsymbol{T}_{i2}^{(k+b/B)}$, and $\boldsymbol{T}_{i3}^{(k+b/B)}$ are obtained for $i = 1, \ldots, g$, using the relationship

$$\boldsymbol{T}_{iq}^{(k+b/B)} = \boldsymbol{T}_{iq}^{(k+(b-1)/B)} - \boldsymbol{T}_{iq,b+1}^{(k-1+b/B)} + \boldsymbol{T}_{iq,b+1}^{(k+b/B)} \qquad (q = 1, 2, 3), \tag{9}$$

for $b = 0, \ldots, B$-1, where only

$$T_{i1,b+1}^{(k+b/B)} = \sum_{j \in S_{b+1}} \tau_{ij}^{(k+b/B)}, \tag{10}$$

$$\boldsymbol{T}_{i2,b+1}^{(k+b/B)} = \sum_{j \in S_{b+1}} \tau_{ij}^{(k+b/B)} \boldsymbol{x}_j, \tag{11}$$

$$\boldsymbol{T}_{i3,b+1}^{(k+b/B)} = \sum_{j \in S_{b+1}} \tau_{ij}^{(k+b/B)} \boldsymbol{x}_j \boldsymbol{x}_j^T, \tag{12}$$

have to be calculated. This is because the first term on the right-hand side of (9) is available from the previous iteration, while the second term is available from the previous scan. In (10) to (12), S_{b+1} denotes the subset of $\{1, \ldots, n\}$ containing the subscripts of those $\boldsymbol{x}_j$ that belong to the $(b+1)$th block.

- **M-step:** For $i = 1, \ldots, g$, the estimates of π_i, $\boldsymbol{\mu}_i$, and $\boldsymbol{\Sigma}_i$ are updated, based on the conditional expectations of the sufficient statistics, as follows:

$$\begin{aligned}
\pi_i^{(k+(b+1)/B)} &= T_{i1}^{(k+b/B)}/n, \\
\boldsymbol{\mu}_i^{(k+(b+1)/B)} &= \boldsymbol{T}_{i2}^{(k+b/B)}/T_{i1}^{(k+b/B)}, \\
\boldsymbol{\Sigma}_i^{(k+(b+1)/B)} &= \left\{\boldsymbol{T}_{i3}^{(k+b/B)} - T_{i1}^{(k+b/B)^{-1}}\boldsymbol{T}_{i2}^{(k+b/B)}\boldsymbol{T}_{i2}^{(k+b/B)^T}\right\}/T_{i1}^{(k+b/B)}.
\end{aligned}$$

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 relative performances of the IEM algorithm with various number of blocks B have been studied by Ng and McLachlan (2002). Adopting here their simple rule to determine the optimal value of B, we choose the number of blocks B to be that factor of n that is the closest to $B^* = \text{round}(n^{2/5})$, where round$(r)$ rounds r to the nearest integer.

3.2 The SPEM 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 $\boldsymbol{x}_j$ are close to zero (for example, $\tau_{ij}^{(k)} < 0.005$). With the SPEM algorithm proposed by Neal and Hinton (1998), 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. To examine this more closely, let A_j $(j = 1, \ldots, n)$ be a subset of $\{1, \ldots, g\}$ which component-posterior probability of $\boldsymbol{x}_j$ is close to zero, say less than 0.005, and hence is held fixed. Suppose that a set of A_j is selected on the kth scan for $j = 1, \ldots, n$, if $\tau_{ij}^{(k)} < 0.005$, then A_j contains the ith component; otherwise A_j^c (the complement of A_j) contains i. Now suppose that the sparse EM step is to be implemented on the subsequent $(k+1)$th scan. Then on the E-step on the $(k+1)$th scan of the SPEM algorithm, consider for all $j = 1, \ldots, n$

- for all $i \in A_j$, set $\tau_{ij}^{(k)} = \tau_{ij}^{(k-1)}$,
- for all $i \in A_j^c$, calculate the posterior probabilities of component membership based on the current estimates $\boldsymbol{\Psi}^{(k)}$, $\tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k)})$, and form $\tau_{ij}^{(k)}$ by rescaling $\tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k)})$ as

$$\tau_{ij}^{(k)} = \sum_{h \in A_j^c} \tau_{hj}^{(k-1)} \, \frac{\tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k)})}{\sum_{h \in A_j^c} \tau_h(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k)})}. \tag{13}$$

This sparse E-step thus will take time proportional only to the number of components $i \in A_j^c$ $(j = 1, \ldots, n)$. The calculation of $T_{i1}^{(k)}$, $\boldsymbol{T}_{i2}^{(k)}$, and $\boldsymbol{T}_{i3}^{(k)}$ are the same as (3) to

(5), but it can now be done efficiently by updating only the contribution to the sufficient statistics for those components $i \in A_j^c$. For example,

$$T_{i1}^{(k)} = \sum_{j=1}^{n} I_{A_j}(i)\tau_{ij}^{(k-1)} + \sum_{j=1}^{n} I_{A_j^c}(i)\tau_{ij}^{(k)}, \tag{14}$$

where $I_{A_j}(i)$ is the indicator function for the set A_j. The first term on the right-hand side of (14) is calculated at the kth scan and can be saved for use in the subsequent SPEM scan. Similar arguments apply to $\boldsymbol{T}_{i2}^{(k)}$ and $\boldsymbol{T}_{i3}^{(k)}$. After running the sparse version a number of scans, a standard EM scan is then performed, and a new set A_j $(j = 1, \ldots, n)$ is selected.

3.3 The SPIEM Algorithm

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 a set of A_j is selected on the kth scan for $j = 1, \ldots, n$. That is, on the $(b+1)$th iteration of the kth scan ($b = 0, \ldots, B$-1), if $\tau_{ij}^{(k-1+b/B)} < 0.005$ for $j \in S_{b+1}$, then A_j contains the ith component; otherwise A_j^c (the complement of A_j) contains i. Now suppose that the sparse IEM step is to be implemented on the subsequent B iterations of the $(k+1)$th scan. Then on the $(b+1)$th iteration ($b = 0, \ldots, B$-1), consider for all $j \in S_{b+1}$,

- for all $i \in A_j$, set $\tau_{ij}^{(k+b/B)} = \tau_{ij}^{(k-1+b/B)}$,
- for all $i \in A_j^c$, calculate the posterior probabilities of component membership based on the current estimates $\boldsymbol{\Psi}^{(k+b/B)}$, $\tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})$, and form $\tau_{ij}^{(k+b/B)}$ by rescaling $\tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})$ as

$$\tau_{ij}^{(k+b/B)} = \sum_{h \in A_j^c} \tau_{hj}^{(k-1+b/B)} \frac{\tau_i(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})}{\sum_{h \in A_j^c} \tau_h(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})}. \tag{15}$$

Similar to the SPEM algorithm, this sparse version of the partial E-step will take time proportional only to the number of components $i \in A_j^c$ $(j = 1, \ldots, n)$. The calculation of $T_{i1,b+1}^{(k+b/B)}$, $\boldsymbol{T}_{i2,b+1}^{(k+b/B)}$, and $\boldsymbol{T}_{i3,b+1}^{(k+b/B)}$ in (10) to (12) can also be done efficiently by updating only the contribution to the sufficient statistics for those components $i \in A_j^c$, as described in last subsection.

The performance of the SPIEM algorithm has been studied by Ng and McLachlan (2002). To avoid the problem of premature component starvation, we follow their procedure that the standard EM step is performed in the first scan, followed by five scans of IEM step before running the sparse version SPIEM step. After running the SPIEM step for five scans, the IEM step is performed for a scan, and a new set A_j $(j = 1, \ldots, n)$ is selected.

3.4 An Inexact IEM Algorithm

An incremental variant of the EM algorithm somewhat similar to that of (9) was investigated by Nowlan (1991) and described in detail in Neal and Hinton (1998). His variant calculates the current conditional expectations of the sufficient statistics as an exponentially decaying average of recently-visited data. That is, (9) is replaced by

$$\boldsymbol{T}_{iq}^{(k+b/B)} = \alpha \boldsymbol{T}_{iq}^{(k+(b-1)/B)} + \boldsymbol{T}_{iq,b+1}^{(k+b/B)} \qquad (q = 1, 2, 3; b = 0, \ldots, B-1),$$

where $0 < \alpha < 1$ is a decay constant. As described in Neal and Hinton (1998), this algorithm will not converge to the exact answer if α is kept at some fixed value. However, this inexact IEM algorithm could be faster than the IEM algorithm as it can forget out-of-date sufficient statistics more rapidly.

4 Multiresolution kd-tree Algorithm

The use of multiresolution kd-tree has been proposed by Moore (1999) to speed up the EM algorithm. Here kd stands for k-dimensional where, in our notation, $k = p$, the dimension of a feature vector $\boldsymbol{x}_j$. The kd-tree is a binary tree that recursively splits the whole set of observations into regions. Each node in the kd-tree includes a bounding box that specifies a subset of the observations and the root node owns all the observations. The children of a node are smaller bounding boxes, generated by splitting along the parent node's widest dimension. The multiresolution kd-tree is constructed top-down, starting from the root node and the splitting procedure continues until the range of observations in the widest dimension of a descendant node is smaller than some threshold γ. This node is then declared to be a leaf-node and is left unsplit. Hence if $\gamma = 0$, then all leaf nodes denote singleton or coincident data points. In this case, the multiresolution kd-tree algorithm cannot speed up the EM algorithm.

Let n_L be the total number of leaf nodes. With the help of the multiresolution data structure built up by the kd-tree, the E-step becomes

- **E-step:** Compute the conditional expectations of the sufficient statistics, given $\boldsymbol{x}$ and the current estimates $\boldsymbol{\Psi}^{(k)}$. For the mth leaf node LN_m $(m = 1, \ldots, n_L)$, the conditional expectations of the sufficient statistics are simplified by treating all the observations in it to have the same posterior probabilities $\tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)})$ calculated at the mean of its observations, where

$$\tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)}) = \pi_i^{(k)} \phi(\bar{\boldsymbol{x}}_m; \boldsymbol{\mu}_i^{(k)}, \boldsymbol{\Sigma}_i^{(k)}) / \sum_{l=1}^{g} \pi_l^{(k)} \phi(\bar{\boldsymbol{x}}_m; \boldsymbol{\mu}_l^{(k)}, \boldsymbol{\Sigma}_l^{(k)}),$$

  for $i = 1, \ldots, g$, and where $\bar{\boldsymbol{x}}_m$ is the mean of observations belonging to the leaf node LN_m. Thus, the contribution of the mth leaf node LN_m $(m = 1, \ldots, n_L)$ to the conditional expectations of the sufficient statistics can be approximated as

$$T_{i1,m}^{(k)} = \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)}) n_m, \tag{16}$$

$$\boldsymbol{T}_{i2,m}^{(k)} = \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)}) n_m \bar{\boldsymbol{x}}_m, \tag{17}$$

$$\boldsymbol{T}_{i3,m}^{(k)} = \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)}) \sum_{j \in LN_m} \boldsymbol{x}_j \boldsymbol{x}_j^T \tag{18}$$

for $i = 1, \ldots, g$, where n_m is the number of observations in the leaf node LN_m. The conditional expectations of the sufficient statistics are therefore obtained as

$$\boldsymbol{T}_{iq}^{(k)} = \sum_{m=1}^{n_L} \boldsymbol{T}_{iq,m}^{(k)}, \qquad (i = 1, \ldots, g; q = 1, 2, 3).$$

The M-step is the same as that of the standard EM algorithm ((6) to (8)). From (16) to (18), it can be seen that the multiresolution kd-tree (without pruning) algorithm speeds up the EM algorithm roughly a factor of n on n_L. Moreover, as the calculation of the sufficient statistics is approximated by treating all the data points in an node as at their mean, the multiresolution kd-tree algorithm is inexact.

4.1 Pruning of Multiresolution kd-tree

In Moore (1999), a further (pruning) step was introduced to reduce the computational time. For each group i at a given node $(i = 1, \ldots, g)$, compute the minimum and maximum values that any observation in the node can have for its current posterior probabilities. Denote these values $\tau_{i,min}$ and $\tau_{i,max}$, respectively. If the differences between $\tau_{i,min}$ and $\tau_{i,max}$ for all $i = 1, \ldots, g$ are small and satisfy some pruning criteria (see below), then the node is treated as if it is a (pseudo) leaf node. Hence its descendants need not be searched at this scan.

Let n_s be the number of observations in the sth node, and $\tau_{i,total}$ the sum of the posterior probabilities of ith group membership for all the observations. With reference to the source code of Moore (1999), we prune if

1. $n_s(\tau_{i,max} - \tau_{i,min}) < \beta\tau_{i,total}$ with $\beta = 0.01$ for all $i = 1, \ldots, g$, and
2. $\log\{\sum_{i=1}^{g} \pi_i \phi_{i,max}\} - \log\{\sum_{i=1}^{g} \pi_i \phi_{i,min}\} < 0.1 \mid \log\{\sum_{i=1}^{g} \pi_i \phi(\bar{\boldsymbol{x}}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\} \mid$,

where $\bar{\boldsymbol{x}}$ is the mean of the data points in the node. For the first criterion, if a larger value of β is adopted, the number of pseudo-leaf nodes will decrease. Hence the time to convergence decreases, but the contributions to the conditional expectations of the sufficient statistics may not be well approximated by (16) to (18). The second criterion ensures that the pruning step will not reduce significantly the value of log likelihood. However, this additional criterion increases the number of pseudo-leaf nodes and hence the time to convergence.

In practice, the time to convergence for this algorithm against that without pruning is a tradeoff between the additional computational time for $\tau_{i,min}$ and $\tau_{i,max}$ $(i = 1, \ldots, g)$ and the fewer number of leaf nodes in each scan. There are some possibilities, such as those described in the source code of Moore (1999), to reduce the amount of computation of $\tau_{i,min}$ and $\tau_{i,max}$ and hence favour the adoption of the pruning step. For example, if $\tau_{i,max}$ is found to be close to zero at a given node, for instance, $\tau_{i,max} < 0.5\tau_{h,min}$ for some other group h, then there is no need to compute $\tau_{i,min}$ and $\tau_{i,max}$ in descendants of this node. It means that, near the tree's leaves, the limiting values of the posterior probabilities need to be computed only for a small fraction of g. In addition, it is easy to determine whether the hyper-rectangle of the current node is very far away from the mean $\boldsymbol{\mu}_i$. If it

is the case, $\tau_{i,max}$ and $\tau_{i,min}$ may be set to zero and hence the ith group is not considered in descendants of the current node. These two procedures will considerably reduce the computational time in cases where there are large number of groups and the overlapping of the groups is small. As described in Moore (1999), the computation of $\tau_{i,min}$ and $\tau_{i,max}$ is much easier to formulate in terms of bounds on the density at the feature vector belonging to the node. It means that the minimum and maximum Mahalanobis squared distances between the mean $\boldsymbol{\mu}_i$ $(i = 1, \ldots, g)$ and any feature vector within the hyper-rectangle are required, the Mahalanobis squared distance between vector $\boldsymbol{x}_j$ and $\boldsymbol{\mu}_i$ is defined as

$$\Delta^2 = (\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}_j - \boldsymbol{\mu}_i).$$

Let $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$ be the minimum and maximum Mahalanobis squared distances, respectively. Then a lower bound on the ith group-conditional density at the feature vector $\boldsymbol{x}_j$ in the node is given by

$$\phi_{i,min} = (2\pi)^{-p/2} \mid \boldsymbol{\Sigma}_i \mid^{-1/2} \exp(-\tfrac{1}{2}\Delta^2_{i,max}),$$

and, similarly, an upper bound $\phi_{i,max}$ is obtained for this density. If follows that a lower bound of the posterior probability is given by

$$\tau_{i,min} = \pi_i \phi_{i,min} / (\pi_i \phi_{i,min} + \sum_{l \neq i} \pi_l \phi_{l,max}).$$

Similarly, an upper bound $\tau_{i,max}$ can be obtained. Moore (1999) used quadratic programming with the hyper-rectangular constraints to find $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$ for $i = 1, \ldots, g$. For data with dimension p less than or equal to three, we propose an analytic geometry approach to obtain the lower and upper bounds of τ_i. The idea is to transform the feature vectors by a matrix of normalized eigenvectors so that the covariance matrix becomes an identity matrix. By doing this, the Mahalanobis squared distance becomes the Euclidean squared distance. Analytic tools within the context of vector geometry can then be applied to find the minimum and maximum values. This analytic geometry approach is found to be faster than the quadratic programming subroutine E04NFF of the FORTRAN NAG library for computing $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$.

In the implementation of the algorithm, we do not perform the pruning step on every scan because it will slow down the algorithm. Our proposed procedure is to use the kd-tree (without pruning) on the first scan, followed by five scans with pruning of the kd-tree. Then we fix the pseudo-leaf nodes obtained from the fifth scan and run five more scans by searching only on the pseudo-leaf nodes of the kd-tree. That is, no pruning step is required in these five scans and hence computational time is saved. Then we perform a scan with pruning of the kd-tree and determine a new set of pseudo-leaf nodes. However, due to the implementation of the pruning step, the number of pseudo-leaf nodes at each scan is different, and hence the approximate log likelihood calculated using the mean of each pseudo-leaf node is not monotonic increasing after each scan. This algorithm can be terminated by considering the convergence of the estimates at each scan; see Section 6.1.

5 An Incremental EM with Multiresolution kd-tree Algorithm

With the multiresolution kd-tree structure, it can be seen from Section 4 that the number of leaf nodes is unchanged once the kd-tree is constructed. In other words, with the multiresolution kd-tree (without pruning) algorithm, the number of leaf nodes is a constant at each scan. Now, we perform the IEM algorithm based on this kd-tree structure without pruning. That is, the leaf nodes are divided into B blocks (Figure 1(a)). At each scan, the partial E-step is implemented for only a block of leaf nodes at a time before the next M-step is performed. With this IEM–kd-tree algorithm, the equations (10) to (12) in Section 3.1 are now replaced by

$$
\begin{aligned}
T_{i1,b+1}^{(k+b/B)} &= \sum_{m \in S_{b+1}} \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k+b/B)}) n_m, \\
\boldsymbol{T}_{i2,b+1}^{(k+b/B)} &= \sum_{m \in S_{b+1}} \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k+b/B)}) n_m \bar{\boldsymbol{x}}_m, \\
\boldsymbol{T}_{i3,b+1}^{(k+b/B)} &= \sum_{m \in S_{b+1}} \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k+b/B)}) \sum_{j \in LN_m} \boldsymbol{x}_j \boldsymbol{x}_j^T,
\end{aligned}
$$

for those LN_m $(m = 1, \ldots, n_L)$ in the $(b+1)$th block, where S_{b+1} now denote a subset of $\{1, \ldots, n_L\}$ containing the subscripts of those leaf nodes LN_m that belong to the $(b+1)$th block. The sparse version of the IEM algorithm (SPIEM) described in Section 3.3 may also be adopted to further reduce the computational time in applying the EM algorithm. As in Section 3.1, we choose the number of blocks B based on the simple rule proposed by Ng and McLachlan (2002) and implement the algorithm according to the procedure, as described for the IEM algorithm in Section 3.1. The flowchart of the algorithm is depicted in Figure 1(b).

6 Comparison of Variants of the EM Algorithm

6.1 Simulation I

A random sample of size $n = 256 \times 256$ observations was generated from a seven-component trivariate normal mixture $(g = 7,\ p = 3)$. The estimates obtained in Liang et al. (1994) were used as the values of our population parameters (Table 1). These seven components correspond to seven tissue types (outer table of the skull and skin; inner table of the skull; temporalis muscle and internal occipital protuberance; Cerebral spinal flow space; gray matter; subcutaneous fat and diploic space; white matter) in the segmentation of a 2D MR image of the human brain.

In this paper, all the algorithms used the same initial estimates as starting values. The algorithms were terminated when the absolute values of the relative changes in the estimates of the means all fell below 0.0001. This stopping criterion was adopted because the approximate log likelihood was not monotonic increasing for multiresolution kd-tree (with pruning) algorithm; see Section 4.1. In the simulation study, we considered the threshold γ to be 1%, 0.5%, and 0.3% of the range in the splitting dimension of the

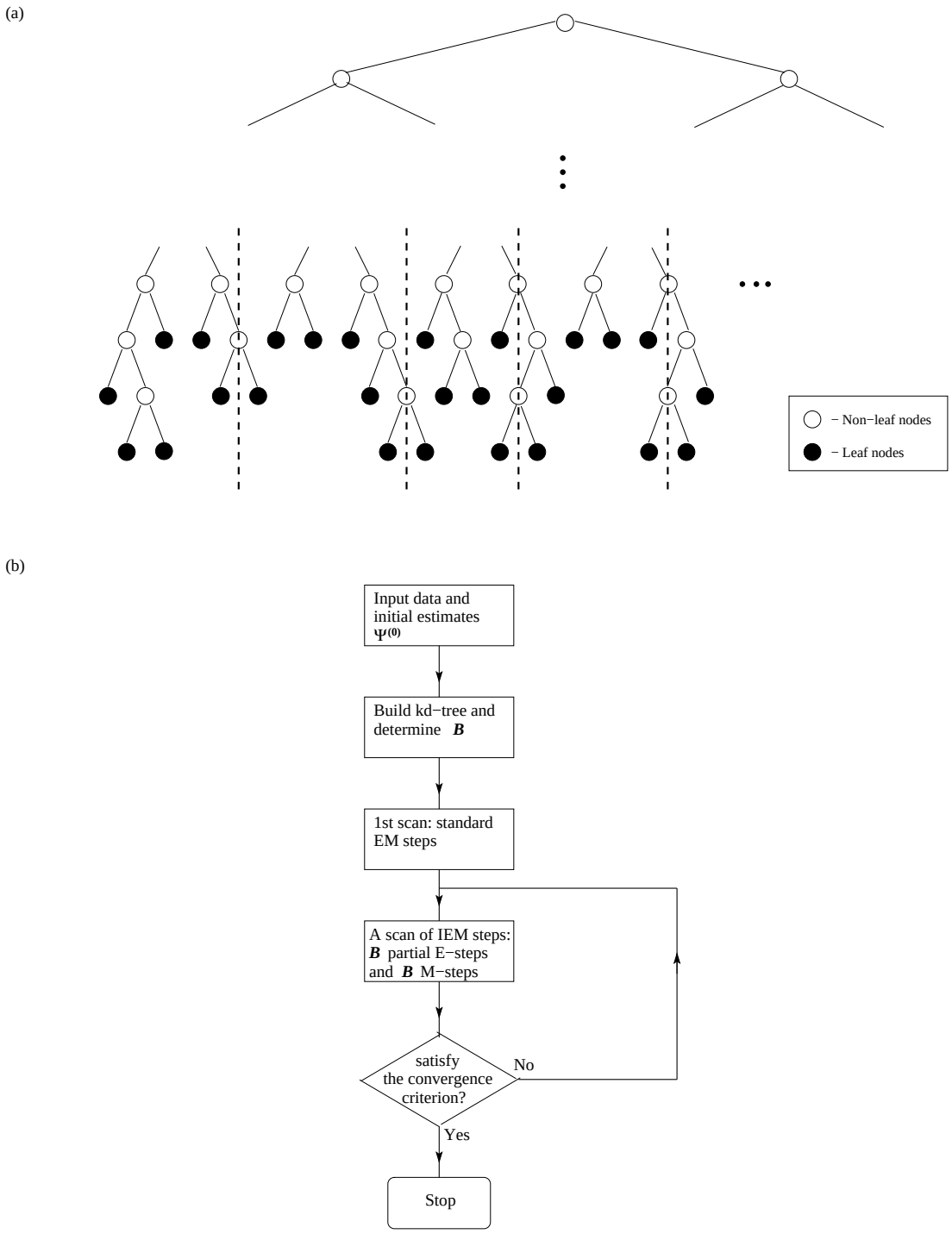

Figure 1: The IEM–kd-tree algorithm: (a) the partition of leaf nodes (say blocks of 6 leaf nodes); (b) the flowchart of the algorithm.

whole data set. The results are summarized in Table 2. All the algorithms are written in FORTRAN and the simulations are run on a Sun unix workstation. With $\gamma = 0.01$, the number of leaf nodes n_L is 18973. For $\gamma = 0.005$ and 0.003, $n_L = 35519$ and 47824, respectively.

Table 1: The Proportions, Intensity Means, Variances, and Correlation Coefficients of Seven Groups: μ_{id} and σ_{id}^2 $(d = 1, 2, 3)$ 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}.

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

In Table 2, **CPU** represents the CPU time in seconds for various algorithms, **nscan** indicates the number of scans to convergence, **log likelihood** is the log likelihood of the simulated data calculated using the final estimates obtained, **error** is the percentage of incorrect segmentation calculated from the final estimates, and **speedup** is the ratio of CPU times compared to that of the standard EM algorithm. The CPU time consists of the computations of the E- and M-steps for the standard EM algorithm, and the partial E- and M-steps for the IEM algorithm. For the kd-tree-based algorithms, it consists of the computation of the E-step (or partial E-step), the M-step, and the construction of the kd-tree.

It can be seen from Table 2 that the standard EM, the IEM, the SPIEM, the multiresolution kd-tree (without pruning, $\gamma = 0.003$), and the IEM–kd-tree ($\gamma = 0.003$) algorithms converged to essentially the same log likelihood value. For the multiresolution kd-tree-based algorithms, it is observed that the smaller value of γ decreases the error rate but increases the CPU time to convergence. It can also be seen that the pruning step with $\gamma \geq 0.005$ does not reduce the computational time, compared with the kd-tree (without pruning) algorithm. The error rates and the values of log likelihood obtained by these two algorithms, however, are similar.

For the inexact IEM algorithm, it can be seen from Table 2 that a larger value of α decreases the error rate but increases the CPU time to convergence. As commented by Neal and Hinton (1998), though the inexact IEM algorithm could be faster than the IEM algorithm with an appropriate value for α, it did not converge to the same log likelihood value as that of the IEM algorithm. From Table 2, it can also be seen that the inexact IEM algorithm has comparable error rates and speedup factors with the multiresolution kd-tree algorithms. But the log likelihood values obtained by the inexact IEM algorithm are in general smaller.

For the IEM–kd-tree algorithm, it leads to a larger speedup factor and a slightly larger value of the log likelihood for $\gamma \leq 0.005$, compared with the kd-tree (without pruning) algorithm. The error rate is also found to be decreased for $\gamma \leq 0.005$. Comparing with the IEM or SPIEM algorithms, it can be seen from Table 2 that the IEM–kd-tree algorithm can lead to a larger speedup factor (with $\gamma = 0.01$), but the error rate is slightly larger. When

Table 2: Summary of Simulation I ($n = 256 \times 256$)

Algorithm	CPU	nscan	log likelihood	error	speedup
standard EM	303.0	90	-367223.2	11.73	1.0
IEM (B=64)	217.0	52	-367223.2	11.72	1.4
SPIEM (B=64)	121.2	56	-367223.2	11.72	2.5
kd-tree (no pruning)					
$\gamma = 0.01$	119.5	89	-367228.5	11.75	2.5
$\gamma = 0.005$	225.1	90	-367223.5	11.74	1.4
$\gamma = 0.003$	312.9	90	-367223.2	11.74	1.0
kd-tree (pruning)					
$\gamma = 0.01$	181.0	88	-367228.7	11.77	1.7
$\gamma = 0.005$	257.0	88	-367223.9	11.74	1.2
$\gamma = 0.003$	285.6	88	-367223.8	11.74	1.1
Inexact IEM (B=64)					
$\alpha = 0.95$	155.4	36	-367231.0	11.74	2.0
$\alpha = 0.96$	184.6	43	-367228.2	11.74	1.6
$\alpha = 0.97$	230.9	54	-367226.0	11.73	1.3
IEM–kd-tree					
$\gamma = 0.01$	81.6	55	-367228.5	11.75	3.7
$\gamma = 0.005$	150.4	55	-367223.5	11.73	2.0
$\gamma = 0.003$	230.3	55	-367223.2	11.73	1.3

a smaller value of γ is adopted, the IEM–kd-tree algorithm provides accurate estimates as that of the IEM or SPIEM algorithms. The speedup factor is also similar to that using the IEM or SPIEM algorithms. However, when the data set is large and redundant, the IEM–kd-tree algorithm can converge to accurate estimates faster than the IEM or SPIEM algorithms, as presented in the next section.

6.2 Simulations II and III

We adopted the same parameter values as in Simulation I, but changed n to $128\times128\times128$ and $256 \times 256 \times 256$ in Simulations II and III, respectively. The results are presented in Tables 3 and 4. For $n = 128^3$, the numbers of leaf nodes n_L are 91321, 281128, and 513235 for $\gamma = 0.01$, 0.005, and 0.003, respectively. For $n = 256^3$, $n_L = 128268$, 560274, and 1216506, respectively.

From Tables 3 and 4, it can be seen that the speedup factors of the IEM and SPIEM algorithms are similar when n is increased. On the other hand, for the kd-tree-based algorithms, the speedup factors increase with n. It can also be seen that the pruning step with $\gamma \leq 0.005$ does reduce the computational time compared to the kd-tree without pruning, but the log likelihood values are slightly smaller.

For the inexact IEM algorithm, the appropriate choice of α seems to depend on n. From Tables 3 and 4, it can be seen that the algorithm requires less number of scans

Table 3: Summary of Simulation II ($n = 128 \times 128 \times 128$)

Algorithm	CPU	nscan	log likelihood	error	speedup
standard EM	10143	96	-11750666	12.00	1.0
IEM (B=256)	6166	55	-11750665	12.00	1.6
SPIEM (B=256)	4114	61	-11750665	12.00	2.5
kd-tree (no pruning)					
$\gamma = 0.01$	773	95	-11750898	12.02	13.1
$\gamma = 0.005$	2179	96	-11750682	12.00	4.7
$\gamma = 0.003$	3802	96	-11750668	12.00	2.7
kd-tree (pruning)					
$\gamma = 0.01$	923	95	-11750904	12.02	11.0
$\gamma = 0.005$	1603	95	-11750689	12.00	6.3
$\gamma = 0.003$	1963	95	-11750680	12.00	5.2
Inexact IEM (B=256)					
$\alpha = 0.95$	1728	12	-11750812	12.01	5.9
$\alpha = 0.97$	2504	18	-11750734	12.00	4.0
$\alpha = 0.99$	6074	45	-11750674	12.00	1.7
IEM–kd-tree					
$\gamma = 0.01$	506	57	-11750899	12.02	20.1
$\gamma = 0.005$	1314	56	-11750682	12.00	7.7
$\gamma = 0.003$	2713	56	-11750667	12.00	3.7

to convergence when n increases. Compared to the multiresolution kd-tree-based algorithms, the values of the log likelihood obtained by the inexact IEM algorithm are smaller.

As in the simulation I, the IEM–kd-tree algorithm leads to a larger speedup factor, a slightly larger value of the log likelihood, and a smaller value of error rate for $\gamma \leq 0.005$, compared with the kd-tree (without pruning) algorithm. From Tables 3 and 4, it can be seen that the IEM–kd-tree algorithm converges to accurate estimates faster than the IEM and SPIEM algorithms. For a less accurate estimates, this algorithm can speed up the EM algorithm by a factor of 22, compared to the SPIEM algorithm and a factor of 56, compared to the standard EM algorithm.

6.3 Real Data

We applied the variants of the EM algorithm described in Sections 3, 4, and 5 to a real MR image data set concerning the human brain. The image was a 2D MR image acquired by a 2 Tesla Bruker Medspac whole body scanner. The acquisition matrix was 256×256. The image intensities were scaled to the range of (0,20) for the parameter estimation. We assumed $g = 7$ and adopted the same initial estimates for all the algorithms. The relative performances of the variants of the EM algorithm are summarized in Table 5. The number of leaf nodes are given by $n_L = 15419$, 24871, and 33024 for $\gamma = 0.01$, 0.005, and 0.003, respectively.

From Table 5, it can be seen that the relative performances of the various versions of

Table 4: Summary of Simulation III ($n = 256 \times 256 \times 256$)

Algorithm	CPU	nscan	log likelihood	error	speedup
standard EM	88950	95	-94015922	11.99	1.0
IEM (B=1024)	59425	54	-94015918	11.99	1.5
SPIEM (B=1024)	35845	62	-94015918	11.99	2.5
kd-tree (no pruning)					
$\gamma = 0.01$	1860	94	-94018940	12.02	47.8
$\gamma = 0.005$	5079	95	-94016148	11.99	17.5
$\gamma = 0.003$	10138	95	-94015939	11.99	8.8
kd-tree (pruning)					
$\gamma = 0.01$	2150	94	-94018863	12.02	41.4
$\gamma = 0.005$	3647	94	-94016164	11.99	24.4
$\gamma = 0.003$	4727	94	-94016001	11.99	18.8
Inexact IEM (B=1024)					
$\alpha = 0.98$	13274	11	-94028005	12.01	6.7
$\alpha = 0.99$	16309	14	-94020334	11.99	5.5
$\alpha = 0.995$	27992	25	-94017080	11.99	3.2
IEM–kd-tree					
$\gamma = 0.01$	1589	56	-94018948	12.02	56.0
$\gamma = 0.005$	3962	56	-94016148	11.99	22.5
$\gamma = 0.003$	8516	56	-94015937	11.99	10.4

the EM algorithm on real MR data are comparable to that presented in Simulation I. As demonstrated in the Simulation I, with the sample size $n = 256^2$, the SPIEM algorithm speeds up the EM algorithm and provides an exact estimates as that of the EM algorithm. The IEM–kd-tree algorithm can converge faster, but only to a smaller log likelihood value.

7 Discussion

We have considered several ways of speeding up the fitting of normal mixture models by using variants of the EM algorithm, including the incremental and sparse versions of EM and the multiresolution kd-tree-based algorithms. The relative performances of these algorithms has been studied, using simulated and real data on MR images. Both the IEM and SPIEM algorithms improve the time to convergence of the EM algorithm by reducing the number of scans required, their performances (in terms of the speedup factor) remain similar when the sample size of the data increases; see Tables 2, 3, and 4. These two algorithms, however, maintain strictly the accurate sufficient statistics in each scan. Thus they both converge to the same likelihood value as that of the standard EM algorithm. The inexact IEM algorithm, on the other hand, requires less number of scans to convergence when the sample size increases. Because it forgets out-of-date sufficient statistics more rapidly, it has larger speedup factor, compared to the IEM algorithm. However, it cannot converge to the exact likelihood value as that of the IEM or the SPIEM algorithms.

Table 5: Analysis of Real 2D MR Image Data ($n = 256 \times 256$)

Algorithm	CPU	nscan	log likelihood	speedup
standard EM	265.1	79	-178810.9	1.0
IEM (B=64)	189.6	46	-178809.4	1.5
SPIEM (B=64)	107.0	51	-178809.3	2.5
kd-tree (no pruning)				
$\gamma = 0.01$	90.2	81	-179077.6	2.9
$\gamma = 0.005$	148.1	82	-178848.8	1.8
$\gamma = 0.003$	203.9	81	-178821.7	1.3
kd-tree (pruning)				
$\gamma = 0.01$	137.3	88	-179067.6	1.9
$\gamma = 0.005$	202.3	92	-178843.6	1.3
$\gamma = 0.003$	248.8	90	-178821.0	1.1
Inexact IEM (B=64)				
$\alpha = 0.92$	105.2	24	-178850.0	2.5
$\alpha = 0.94$	125.4	29	-178835.2	2.1
$\alpha = 0.96$	187.8	40	-178823.5	1.4
IEM–kd-tree				
$\gamma = 0.01$	64.3	47	-179082.9	4.1
$\gamma = 0.005$	107.4	49	-178848.7	2.5
$\gamma = 0.003$	148.7	48	-178820.5	1.8

Moreover, it is found that the inexact IEM algorithm needs an adequate randomization of the data such that each block of the data has entities from each of the components. Otherwise, the algorithm will converge to points that would be very poor estimates. An appropriate choice of the decay constant α is also crucial to the ultimate qualities of the final estimates.

With the multiresolution kd-tree structure, the number of leaf nodes n_L does not linearly increase with n. For example, with $\gamma = 0.01$, $n_L = 18973$, 91321, and 128268 for $n = 256^2$, 128^3, and 256^3, respectively. Hence, when the sample size increases, kd-tree-based algorithms have better relative performances for speeding up the EM algorithm. As approximation is made in the calculation of the sufficient statistics (Equations (16) to (18)), the multiresolution kd-tree method is inexact. However, it can be seen from Tables 2 to 4 that for sufficient small value of γ (smaller sized leaf nodes), say $\gamma \leq 0.003$, the multiresolution kd-tree (without pruning) and the IEM–kd-tree algorithms converge to essentially the same maximum likelihood value as the standard EM algorithm.

With the simulated data sets II and III, we found that the pruning step (with $\gamma \leq 0.005$) can reduce the computational time, compared with the kd-tree (without pruning) algorithm. It implies that the fewer number of leaf nodes in each scan offsets the additional computational time in finding the minimum and maximum values of the posterior probabilities. However, due to the implementation of the pruning step, the number of pseudo-leaf nodes at each scan will be different. Hence, with the pruning version, there is no

guarantee that the log likelihood value is monotonic increasing after each scan. In fact, we have seen that the log likelihood value for some simulated data starts to decrease just before the convergence of the estimates.

When an IEM algorithm is adopted in conjunction with a multiresolution kd-tree, it can be seen that the speedup factor increases and the error rate decreases. The algorithm is also stable because the partition is performed on the leaf nodes and their number is constant in each scan; see Section 5. This implies that the approximate log-likelihood values are monotonic increasing after each scan.

For applications to high dimensional data sets, it is noted that the number of leaf nodes will increase dramatically when the dimension of the feature vector p increases. This implies that multiresolution kd-tree-based algorithms will not be able to speed up the EM algorithm when the dimension of the feature vector p increases (McCallum et al., 2000; Sand and Moore, 2001; Thiesson et al., 2001). Moore (1999) conducted a simulation experiment to study the impact of the dimension p on the number of leaf nodes. He generated $n = 160000$ data points from a mixture of p-variate normal. In that study, $g = 20$ with equal mixing proportions were considered. The ratio of n/n_L was found to be 4.4, 2.6, and 1.7 for $p = 4$, $p = 5$, and $p = 6$, respectively. Beyond $p = 6$, the ratio of n/n_L is not much greater than one and hence there is little gain for using multiresolution kd-tree approaches, even though the pruning step can reduce the number of leaf nodes. Recent work on dimensionality reduction, however, may enable the multiresolution kd-tree-based algorithms to be used on high dimensional data. These methods such as Broomhead and Kirby (2000), Dasgupta (2000), and Jimenez and Landgrebe (1999) reduce dimensionality of the data via a linear projection that optimizes a function known as the projection index. The effectiveness in applying these dimensionality reduction techniques to huge data sets needs further exploration.

Acknowledgements

The authors wish to thank Dr Andrew Moore and Mr Peter Sand for helpful discussion on the pruning criteria for the multiresolution kd-tree algorithm.

References

D.S. Broomhead and M. Kirby. A new approach to dimensionality reduction: theory and algorithms. *SIAM J. Appl. Math.*, 60:2114–2142, 2000.

S. Dasgupta. Experiments with random projection. In C. Boutillier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 143–151. Morgan Kaufmann, San Francisco, 2000.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *J. Royal Statist. Soc. B*, 39(1):1–38, 1977.

L.O. Jimenez and D.A. Landgrebe. Hyperspectral data analysis and supervised feature reduction via projection pursuit. *IEEE Trans. Geoscience and Remote Sensing*, 37(6): 2653–2667, 1999.

Z. Liang, J.R. MacFall, and D.P. Harrington. Parameter estimation and tissue segmentation from multispectral MR images. *IEEE Trans. Medical Imaging*, 13(3):441–449, 1994.

A. McCallum, K. Nigam, and L.H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In R. Bayardo R. Ramakrishnan, S. Stolfo and I. Parsa, editors, *Proc. of the Sixth ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining*, pages 169–178. ACM Press, New York, 2000.

G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1998.

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

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

X.L. Meng. On the rate of convergence of the ECM algorithm. *Ann. Statist.*, 22(1): 326–339, 1994.

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

A.W. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In S.A. Solla M.S. Kearns and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 543–549. MIT Press, Cambridge, Massachusetts, 1999.

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

S.K. Ng and G.J. McLachlan. On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. *Statistics and Computing, in Press*, 2002.

S.J. Nowlan. *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures*. PhD Thesis, Carnegie Mellon University, Pittsburgh, 1991.

P. Sand and A.W. Moore. Repairing faulty mixture models using density estimation. In C.E. Brodley and A.P. Danyluk, editors, *Proc. 18th Intern. Conf. on Machine Learning*, pages 457–464. Morgan Kaufmann, San Francisco, 2001.

B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, 45(3):279–299, 2001.

Authors' addresses:

Professor Geoffrey J. McLachlan
Department of Mathematics
The University of Queensland
Brisbane Q4072
Australia

Tel. +61 7 33652150
Fax +61 7 33651477
E-mail: gjm@maths.uq.edu.au

Dr. Shu-Kay Ng
Department of Mathematics
The University of Queensland
Brisbane Q4072
Australia

Tel. +61 7 33656139
Fax +61 7 33651477
E-mail: skn@maths.uq.edu.au