ON SPEEDING UP THE EM ALGORITHM IN PATTERN RECOGNITION: A COMPARISON OF INCREMENTAL AND MULTIRESOLUTION KD-TREE-BASED APPROACHES

Shu Kay Ng and Geoffrey John McLachlan
Centre of Statistics
Department of Mathematics
University of Queensland
St. Lucia, Brisbane, QLD 4072, Australia
skn@maths.uq.edu.au and gjm@maths.uq.edu.au

Abstract

Finite mixture models implemented via the EM algorithm are being increasingly used in a wide range of problems in the context of unsupervised statistical pattern recognition. As each E-step visits each feature vector on a given iteration, the EM algorithm requires considerable computation time in its application to large data sets. We consider two approaches, an incremental EM (IEM) algorithm and a multiresolution kd-tree-based approach, that can be used to reduce the computational time in applying the EM algorithm. In this paper, we investigate and compare their relative performances in speeding up the EM algorithm. Some simulated and real data on medical magnetic resonance (MR) images were used in this investigation. The results show that the IEM algorithm leads to the lowest error rate, but that the reduction in the time to convergence is very limited. The multiresolution kd-tree-based algorithms, on the other hand, provide larger reduction in computation time. In particular, it is demonstrated that by using an IEM algorithm in conjunction with a multiresolution kd-tree, the EM algorithm can be speeded up by a factor of 56.0 for very large data sets.

1. Introduction

Finite mixture models have been widely applied in the £eld of unsupervised statistical pattern recoginition, where a pattern is considered as a single entity and is represented by a £nite dimensional vector of features of the pattern. With the computer revolution, data are increasingly being collected in the form of images, as in Magnetic Resonance (MR) imaging. The aim of pattern recognition is to automate processes performed by humans. For example, automatic segmentation of different tissue cells of MR images of the human brain facilitates an imaging-based medical diagnosis, providing an aid to surgery and treatment planning, as well as a means for studying the effect of the locality of abnormal tissues in neurologic disease.

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

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

where the group-conditional densities $f_i(\boldsymbol{x}_j; \boldsymbol{\theta}_i)$ are speci£ed 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 it can be estimated by the maximum likelihood method implemented via the Expectation-Maximization (EM) algorithm of [1]. 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. Within the EM framework, the observed data $\boldsymbol{x}$ and $\boldsymbol{y} = (\boldsymbol{x}^T, \boldsymbol{z}^T)^T$ are regarded as the incomplete and the

complete data respectively, where $\boldsymbol{x} = (\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_n^T)^T$ and $\boldsymbol{z} = (\boldsymbol{z}_1^T, \ldots, \boldsymbol{z}_n^T)^T$. For mixtures with normal component densities, it is computationally advantageous to work in terms of the suf£cient statistics. On the $(k+1)$th iteration of the EM algorithm, the E-step calculates the current conditional expectations of the suf£cient statistics

$$\begin{aligned} T_{i1}^{(k)} &= \sum_{j=1}^{n} \tau_{ij}^{(k)}, \\ \boldsymbol{T}_{i2}^{(k)} &= \sum_{j=1}^{n} \tau_{ij}^{(k)} \boldsymbol{x}_j, \\ \boldsymbol{T}_{i3}^{(k)} &= \sum_{j=1}^{n} \tau_{ij}^{(k)} \boldsymbol{x}_j \boldsymbol{x}_j^T, \end{aligned}$$

where

$$\tau_{ij}^{(k)} = \pi_i^{(k)} \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_i^{(k)}, \boldsymbol{\Sigma}_i^{(k)}) / \sum_{l=1}^{g} \pi_l^{(k)} \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_l^{(k)}, \boldsymbol{\Sigma}_l^{(k)})$$

is the current estimate of the posterior probability that $\boldsymbol{x}_j$ comes from the ith group. The M-step updates the estimates as follows:

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

The E- and M-steps are alternated repeatedly until convergence; see for example [2,3]. An outright or hard clustering of the observations can be obtained by assigning the jth feature vector to the group to which it has the highest estimated posterior probability. In the context of automatic segmentation of MR images, this hard clustering corresponds to the noncontextual segementation, in which the spatial characteristics of each voxel is ignored [4].

As set out in some detail in [3, Section 1.7], the EM algorithm has a number of desirable properties, including reliable global convergence and the likelihood is not decreased after each EM iteration. However, as the E-step is implemented for each feature vector $\boldsymbol{x}_j$ before the next M-step is performed, the time spent in the E-step depends linearly on the number of observations and the number of groups in the mixture models. In considering methods for improving the speed of the EM algorithm for the maximum likelihood £tting of mixture models to large data, it is highly desirable if the simplicity and stability of the EM algorithm can be preserved [5, Chapter 12]. In this paper, we consider and compare the incremental EM (IEM) algorithm of [6] and the multiresolution kd-tree of [7] for speeding up the implementation of the EM algorithm for the £tting of normal mixture models. Our focus is on the relative CPU time to convergence and the error rate of the segmentation for various algorithms against those of the standard EM algorithm.

2. ALGORITHMS AND SIMULATION I

2.1. IEM Algorithm

With the IEM algorithm proposed by [6], 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 "pass" or iteration of the IEM algorithm thus consists of B partial E-steps and B M-steps. These B partial E-steps will take more time to implement than the one full E-step of the standard EM algorithm. The additional time involves the inversion of the group-covariances matrices, which have to be performed at each of the B partial steps. Moreover, one iteration of the IEM algorithm requires (B-1) additional 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 iteration 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 iteration and the fewer number of iteration required because of the more frequent updating after each partial E-step. The relative performances of the IEM algorithm with various number of blocks B have been studied by [5,8]. Adopting here their simple guided 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.

2.2. Multiresolution kd-tree (Without Pruning) Algorithm

The use of multiresolution kd-trees has been proposed by [7] 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 partitions. Each node in the kd-tree is associated with a certain partition (subset) of the observations and the root node owns all the observations. The kd-tree is constructed top-down, and the partition of the current node is implemented using the hyper-rectangle of the node, which is de£ned as the smallest bounding box that contains all the observations owned by the node. Based on this hyper-rectangle, the splitting dimension is identi£ed as the dimension of the hyper-rectangle for which the range of the observations is greatest. The current node is split at the midrange of this splitting dimension. The splitting procedure continues until

the range of observations in the splitting dimension of the hyper-rectangle of the node is smaller than some threshold γ. This node is then declared to be a leaf-node and is left unsplit.

With the help of the multiresolution data structure built up by the kd-tree, the computation of the (current) conditional expectations of the suf£cient statistics on the root node (E-step) can be obtained by summing those on all the leaf nodes. Let n_L be the total number of leaf nodes. For the mth leaf node LN_m $(m = 1, \ldots, n_L)$, its conditional expectations of the suf£cient statistics can be simpli£ed 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

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

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 conditional expectations of the suf£cient statistics on the mth leaf node LN_m $(m = 1, \ldots, n_L)$ can be approximated as

$$\begin{aligned} T_{i1,m}^{(k)} &= \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)}) n_m, \\ \boldsymbol{T}_{i2,m}^{(k)} &= \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)}) n_m \bar{\boldsymbol{x}}_m, \\ \boldsymbol{T}_{i3,m}^{(k)} &= \tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k)}) \sum_{j \in LN_m} \boldsymbol{x}_j \boldsymbol{x}_j^T \end{aligned} \qquad (2)$$

for $i = 1, \ldots, g$, where n_m is the number of observations in the leaf node LN_m. Hence, without pruning, the speed up of the EM algorithm is roughly proportional to the ratio of n against the number of leaf nodes n_L. In practice, the error rate depends on the size of the leaf nodes. If the leaf node is very small (γ small), the simpli£ed equations (2) are applicable and the consequent error rate is small.

2.3. Multiresolution kd-tree (With Pruning) Algorithm

In [7], a further (pruning) step was introduced to reduce the computation 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. 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. If the difference between the minimum and maximum values of the ith group posterior probability, denoted by $\tau_{i,min}$ and $\tau_{i,max}$, respectively, satis£es

$$n_s(\tau_{i,max} - \tau_{i,min}) < \beta \tau_{i,total} \qquad (3)$$

for $i = 1, \ldots, g$, where β is a small threshold (say, 0.01), then the node is treated as if it is a (pseudo) leaf node. Hence its descendants need not be searched at this iteration. If a larger value of β is adopted, more non-leaf-nodes will satisfy (3). Subsequently, the number of leaf nodes decreases, and hence, the time to convergence decreases, but the error rate increases. In the source code of [7], there is a further step to check whether the pruning step (3) will result in bad likelihood estimates. With this checking step, the number of leaf nodes and hence the time to convergence increase, but the error rate decreases. In practice, the time to convergence for this algorithm against that without pruning (Section 2.2) is a tradeoff between the additional computation time for $\tau_{i,min}$ and $\tau_{i,max}$ $(i = 1, \ldots, g)$ and the fewer number of leaf nodes in each iteration. There are some possibilities, such as those described in the source code of [7], 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 consider the ith group posterior probabilities of all observations in descendants of this node. It means that, near the tree's leaves, the bounds on 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 set to zero and hence the ith group is removed from all descendants of the current node. These two procedures will considerably reduce the computation time in cases where there are large number of groups and the overlapping of the groups is small. However, due to the implementation of the pruning step, the number of leaf nodes at each iteration is not the same, and hence the approximate log likelihood calculated using the mean of each leaf node is not monotonic increasing after each iteration. This algorithm can be terminated by considering the convergence of the estimates at each iteration; see Section 2.5.

As described in [7], 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 de£ned as

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

Let $\Delta_{i,min}^2$ and $\Delta_{i,max}^2$ 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

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

Similarly, an upper bound $\phi_{i,max}$ is obtained for this den-

Table 1. The Proportions, Intensity Means, Variances, and Correlation Coef£cients of Seven Groups $(i = 1, \ldots, 7)$

i	π_i	μ_{i1}	μ_{i2}	μ_{i3}	σ^2_{i1}	σ^2_{i2}	σ^2_{i3}	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

sity. 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. In [7], quadratic programming is used to £nd $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$ with the hyper-rectangular constraints 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 optimum values. 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 vectors and space geometry can then be applied to £nd the minimum and maximum values. This approach is found to be faster than the quadratic programming subroutine E04NFF of the FORTRAN NAG library for computing $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$.

2.4. IEM with Multiresolution kd-tree Algorithm

In this section, we propose an IEM algorithm in conjunction with a multiresolution kd-tree (without pruning). The number of leaf nodes is unchanged at each iteration using a kd-tree with no pruning. With this IEM–kd-tree algorithm, the number of leaf nodes are divided into B blocks and the E-step is implemented for only a block of leaf nodes at a time before the next M-step is performed. As in Section 2.1, we choose the number of blocks B based on the simple rule of [5,8].

2.5. Result of 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 [9] were used as the values of our population parameters. They are displayed in Table 1, where μ_{id} and σ^2_{id} $(d = 1, 2, 3)$ are, respectively, the means and variances of the trivariate normal distribution associated with the ith group. The correlation coef£cients 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 (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 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 kd-tree algorithm (with pruning); see Section 2.3. In the simulation study, we considered the threshold for the size of leaf node, γ, to be 1% and 0.5% of the range in the splitting dimension of the whole data set (root node). The results are summarized in Table 2. With $\gamma = 0.01$, the number of leaf nodes n_L is 18973. For $\gamma = 0.005$, $n_L = 35519$.

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

Algorithm	CPU	niter	ll_{true}	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
kd-tree (no pruning)					
($\gamma = 0.01$)	119.5	89	–367228.5	11.82	2.5
($\gamma = 0.005$)	225.1	90	–367223.5	11.76	1.3
kd-tree (pruning)					
($\gamma = 0.01; \beta = 0.01$)	283.7	89	–367296.4	11.96	1.1
($\gamma = 0.01; \beta = 0.03$)	180.1	86	–367420.0	12.10	1.7
($\gamma = 0.005; \beta = 0.01$)	302.7	88	–367295.8	11.94	1.0
($\gamma = 0.005; \beta = 0.03$)	191.2	86	–367419.9	12.06	1.6
IEM–kd-tree					
($\gamma = 0.01$)	81.6	55	–367228.5	11.83	3.7
($\gamma = 0.005$)	150.4	55	–367223.5	11.75	2.0

In Table 2, **CPU** represents the CPU time in seconds for various algorithms, **niter** indicates the number of iterations to convergence, $ll_{\mathbf{true}}$ is the log likelihood of the simulated data calculated at the £nal estimates obtained, **error** presents the percentage of incorrect segmentation, and **speedup** shows the ratio of CPU times compared to the standard EM algorithm. The CPU time involves 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 involves the computation of the E-step, the M-step, and the construction of the kd-tree.

It can be seen from Table 2 that the standard EM,

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

Algorithm	CPU	niter	ll_{true}	error	speedup
standard EM	10143	96	–11750666	12.00	1.0
IEM (B=256)	6166	55	–11750665	12.00	1.6
kd-tree (no pruning)					
($\gamma = 0.01$)	773	95	–11750898	12.15	13.1
($\gamma = 0.005$)	2179	96	–11750682	12.04	4.7
kd-tree (pruning)					
($\gamma = 0.01; \beta = 0.007$)	771	91	–11752705	12.27	13.2
($\gamma = 0.01; \beta = 0.01$)	707	95	–11753770	12.36	14.3
($\gamma = 0.005; \beta = 0.007$)	907	91	–11752697	12.20	11.2
($\gamma = 0.005; \beta = 0.01$)	819	95	–11753766	12.27	12.4
IEM–kd-tree					
($\gamma = 0.01$)	506	57	–11750899	12.16	20.1
($\gamma = 0.005$)	1314	56	–11750682	12.04	7.7

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

Algorithm	CPU	niter	ll_{true}	error	speedup
standard EM	88950	95	–94015922	11.99	1.0
IEM (B=1024)	59425	54	–94015918	11.99	1.5
kd-tree (no pruning)					
($\gamma = 0.01$)	1860	94	–94018940	12.20	47.8
($\gamma = 0.005$)	5079	95	–94016148	12.03	17.5
kd-tree (pruning)					
($\gamma = 0.01; \beta = 0.007$)	1751	95	–94034451	12.34	50.8
($\gamma = 0.01; \beta = 0.01$)	1680	93	–94040015	12.38	52.9
($\gamma = 0.005; \beta = 0.007$)	2219	95	–94034056	12.21	40.1
($\gamma = 0.005; \beta = 0.01$)	2128	93	–94039705	12.24	41.8
IEM–kd-tree					
($\gamma = 0.01$)	1589	56	–94018948	12.20	56.0
($\gamma = 0.005$)	3962	56	–94016148	12.03	22.5

the IEM, the kd-tree ($\gamma = 0.005$), and the IEM-kd-tree ($\gamma = 0.005$) algorithms converged to essentially the same log likelihood value. However, the error rates for these algorithms are different, due to the heavy overlapping of groups 3 and 6 (see Table 1). The latter means that there exists several feature vectors that have posterior probabilities of two or more group membership very close to each other. For the 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 $\beta = 0.01$ does not reduce the computation time, compared with the kd-tree (without pruning) algorithm. If a larger value of β (0.03) is adopted, the CPU times decrease but the error rates increase. Moreover, due to the implementation of the pruning step, there are only small improvements in the error rate and the £nal log likelihood when γ is changed from 0.01 to 0.005. For the proposed IEM algorithm with the multiresolution kd-tree, it leads to a larger speedup factor but similar values of the log likelihood, compared with the kd-tree (without pruning) algorithm. However, the error rate is found to be increased for $\gamma = 0.01$ but decreased for $\gamma = 0.005$. From Table 2, it can be seen that this IEM–kd-tree algorithm gives the largest speedup factor of 3.7.

3. SIMULATIONS II AND III

We adopted the same parameter values as in Simulation I, but changed n to $128 \times 128 \times 128$ 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 and 281128 for $\gamma = 0.01$ and 0.005, respectively. For $n = 256^3$, $n_L = 128268$ and 560274, respectively, for $\gamma = 0.01$ and 0.005.

It can be seen that the speedup factors of the IEM algorithm are small and similar when n is increased. On the other hand, for the kd-tree-based algorithms, the speedup factors increase with n. From Tables 3 and 4, it can be seen that the pruning step does reduce the computation time compared with the kd-tree with no pruning. However, the error rates are relatively larger and the log likelihood values are smaller. If a smaller value of β (0.007) is adopted, the error rates decreases but the CPU times increases. The IEM–kd-tree algorithm gives the largest speedup factors, equal to 20.1 and 56.0 for Simulations II and III, respectively.

4. REAL DATA

In this section, we applied the versions of the EM algorithm described in Section 2 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 results are summarized in Table 5. With $\gamma = 0.01$, $n_L = 15068$; for $\gamma = 0.005$, $n_L = 24576$.

From Table 5, it can be seen that the relative performances of the various versions of the EM algorithm on real MR data are comparable to those of Simulation I, except the kd-tree (with pruning) algorithm. This algorithm requires more number of iterations to convergence and has the smallest value of the log likelihood.

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

Algorithm	CPU	niter	ll_{true}	speedup
standard EM	265.1	79	–178810.9	1.0
IEM (B=64)	189.6	46	–178809.4	1.4
kd-tree (no pruning)				
($\gamma = 0.01$)	90.2	81	–179077.6	2.9
($\gamma = 0.005$)	148.1	82	–178848.8	1.8
kd-tree (pruning)				
($\gamma = 0.01; \beta = 0.01$)	255.1	84	–179506.6	1.0
($\gamma = 0.005; \beta = 0.01$)	300.9	85	–179433.5	0.9
IEM–kd-tree				
($\gamma = 0.01$)	64.3	47	–179082.9	4.1
($\gamma = 0.005$)	107.4	49	–178848.7	2.5

5. DISCUSSION

We have compared the relative performances of the IEM algorithm and various multiresolution kd-tree-based algorithms to speed up the implementation of the EM algorithm. As the IEM algorithm improves the time to convergence of the EM algorithm by reducing the number of iterations required, its performance remains similar when the sample size of the data increases. From Tables 2 to 4, it can be seen that the reduction in time to convergence is very limited, compared to the kd-tree-based algorithms, but the IEM algorithm leads to the lowest error rate. On the other hand, with the multiresolution kd-tree data 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 kd-tree-based algorithms, when γ decreases (smaller sized leaf nodes), it can be seen that the error rate descreases but the CPU time to convergence increases. In the applications of preliminary data analysis, $\gamma = 0.01$ may be so chosen that results can be available earlier. Moreover, within the context of automatic segmentation of MR images, contextual segmentation, in which the spatial characteristics of each voxel is considered, will be performed based on the noncontextual analysis to £nally segment the tissue cells. The misallocations of the voxels at the preliminary noncontextual segmentation can be corrected at this £nal stage [4].

With the simulated data on MR images, we found that the pruning step can reduce the computation time, compared with the kd-tree (without pruning) algorithm, it implies that the fewer number of leaf nodes in each iteration offsets the additional computation time for the minimum and maximum values of the posterior probabilities. However, the error rates are relatively higher, which may be lowered if a smaller value of β is adopted. But consequently, the CPU times increase. Moreover, when the kd-tree (with pruning) algorithm is applied to the real data on 2D MR image, it requires more number of iterations and has smaller value of the log likelihood. Its performance on larger real data sets requires further investigation.

The IEM algortihm with the multiresolution kd-tree structure is the fastest version of the EM algorithm considered here. Its superior performance has been demonstrated on simulated data in Sections 2 and 3 and on real data in Section 4. In particular, with this combined version, the value of the "true" log likelihood calculated at the parameter estimate obtained after each iteration is monotonic increasing. From Section 3, it can be seen that the IEM algorithm with the multiresolution kd-tree structure can speed the EM algorithm up by a factor of 56.0 for very large data sets.

REFERENCES

[1] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm (with discussion)," Journal of the Royal Statistical Society Series B, vol. 39, 1977, pp. 1–38.

[2] G.J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, New York, 1992.

[3] G.J. McLachlan and T. Krishman, *The EM Algorithm*, Wiley, New York, 1997.

[4] G.J. McLachlan, S.K. Ng, G. Galloway, and D. Wang, "Clutering of magnetic resonance images," Proceedings of the American Statistical Association (Statistical Computing Section), Alexandria, American Statistical Association, Virginia, 1996, pp. 12–17.

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

[6] R.M. Neal and G.E. Hinton, "A view of the EM algorithm that justi£es incremental, sparse, and other variants," in *Learning in Graphical Models*, M.I. Jordan (Eds.), Kluwer, Dordrecht, 1998, pp. 355–368.

[7] A.W. Moore, "Very fast EM-based mixture model clustering using multiresolution kd-trees," in *Advances in Neural Information Processing Systems 11*, M.S. Kearns, S.A. Solla, and D.A. Cohn (Eds.), MIT Press, Cambridge, Massachusetts, 1999, pp. 543–549.

[8] S.K. Ng and G.J. McLachlan, "On the choice of the number of blocks with the incremental EM algorithm for the £tting of normal mixtures," Technical Report, Centre of Statistics, University of Queensland, Brisbane, 2001.

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