PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 1573 – 1589

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Speeding up the EM algorithm for mixture model-based segmentation of magnetic resonance images

Shu-Kay Ng*, Geoffrey J. McLachlan

Department of Mathematics, University of Queensland, Brisbane QLD 4072, Australia

Received 10 July 2003; accepted 19 February 2004

Abstract

Mixture models implemented via the expectation-maximization (EM) algorithm are being increasingly used in a wide range of problems in pattern recognition such as image segmentation. However, the EM algorithm requires considerable computational time in its application to huge data sets such as a three-dimensional magnetic resonance (MR) image of over 10 million voxels. Recently, it was shown that a sparse, incremental version of the EM algorithm could improve its rate of convergence. In this paper, we show how this modified EM algorithm can be speeded up further by adopting a multiresolution kd-tree structure in performing the E-step. The proposed algorithm outperforms some other variants of the EM algorithm for segmenting MR images of the human brain.
© 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: EM algorithm; Hidden Markov random field; Image segmentation; Magnetic resonance imaging; Mixture models; Multiresolution kd-trees; Sparse incremental EM algorithm; Statistical pattern recognition

1. Introduction

Finite mixture models have been widely applied in the field of unsupervised statistical pattern recognition, where a pattern is considered as a single entity and is represented by a finite dimensional vector of features of the pattern [1–3]. The aim of pattern recognition is to automate processes performed by humans. Important applications include a variety of disciplines such as medicine, computer vision, signal and image analysis, machine learning, and remote sensing. For example, automatic segmentation of different tissue cells of magnetic resonance (MR) images of the human brain facilitates an imaging-based medical diagnosis, provided an aid to surgery and treatment planning [4], as well as a means for studying the effect of the locality of abnormal tissues in neurologic disease [5,6] and the human brain activation effects to stimuli [7,8]. Such tissue segmentation of MR images is often achieved by applying statistical classification techniques to the signal intensities [9,10].

We consider here a statistical-based approach whereby the intensity on each voxel is modeled by a Gaussian mixture [11]. The expectation-maximization (EM) algorithm [12] is adopted to segment MR images and estimate the tissue parameters. An approximation to the E-step of the EM algorithm is employed based on a fractional weight version of Besag's iterated conditional modes (ICM) algorithm [13]. The prior (spatial) distribution of different tissue types is modeled by a hidden Markov random field (MRF) so as to incorporate spatial continuity constraints on the tissue segmentation. Alternative segmentation approaches using the mean field theory can be found in Refs. [14,15].

As set out in some detail in Ref. [16, 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 [17,18]. With the computer revolution, huge data sets of 10 millions of multidimensional images are now commonplace. The size of these image data sets means that there is an ever increasing demand on speeding up the EM algorithm; yet it is highly

* Corresponding author. Fax: +61-7-33651477.
E-mail address: skn@maths.uq.edu.au (S.-K. Ng).

desirable if the simplicity and stability of the EM algorithm can be preserved [19, Chapter 12].

In the context of mixture models, various attempts have been proposed recently to speed up the EM algorithm [17,20,21]. In particular, an incremental version (IEM) and a sparse, incremental version (SPIEM) of the EM algorithm both improve the rate of convergence and preserve the desirable convergence guarantees of the EM algorithm [18,22]. These two algorithms proceed by dividing the data into blocks and implementing the E-step for only a block of data at a time before performing an M-step. Detailed formulation and empirical studies on these two algorithms can be found in Ng and McLachlan [22]. A recursive, nonparametric mixture approach [23] can be regarded as a heuristic version of the IEM algorithm.

In this paper, we propose to speed up the SPIEM algorithm further by imposing a multiresolution *k*d-tree structure in performing the E-step. We also consider a second version that involves "pruning" the tree-nodes. These two new SPIEM multiresolution *k*d-tree-based algorithms provide a fast EM-based mixture model approach for segmenting three-dimensional (3D) MR images. Although the multiresolution *k*d-tree approach has been applied to speed up the EM algorithm with promising results [17,24,25], there are some important issues to be resolved with its integration to the SPIEM framework. Firstly, the multiresolution *k*d-tree approach is inexact in the sense that the contribution of all the data points in a tree node to the sufficient statistics is simplified by calculating at the mean of these data points; see Section 2. This approximation will affect the "quality" of the final segmentation which is quantified in terms of the final log likelihood value and the number of misclassified voxels. Secondly, the efficiency of the multiresolution *k*d-tree approach to speed up the EM algorithm relies on the pruning process (see Section 2.1) which involves calculations of the upper and lower bounds on the probability density function at each tree node [17]. Thirdly, as a consequence of the pruning process, the "blocking" of nodes becomes nontrivial because the number of pruned nodes is different at each iteration; see Section 3.2. Fourthly, the multiresolution *k*d-tree approach will not be able to speed up the EM algorithm for applications to high-dimensional data sets [21,26]. In this paper, we shall address these issues carefully. In particular, we show how tuning parameters can be adjusted so that the proposed SPIEM *k*d-tree-based algorithms provide an accurate solution and preserve the reliable convergence as that for the EM algorithm. We also propose a novel analytical geometry approach to speed up the pruning process. In addition, to evaluate the applicabilities of the two versions of SPIEM multiresolution *k*d-tree-based algorithms, a study is performed to compare with some other existing algorithms. Our focus is on providing a guide to the possible gains in CPU time to convergence (speedup factor) and the convergence properties of each algorithm. This comparative study thus also helps to advance our understanding of existing algorithms for speeding up the EM algorithm.

The rest of the paper is organized as follows: Section 2 reviews the multiresolution *k*d-tree approach to speed up the EM algorithm for the fitting of Gaussian mixtures. An analytical geometry approach is proposed to speed up the pruning process. In Section 3, we introduce briefly the SPIEM algorithm and develop the two new algorithms by adopting a multiresolution *k*d-tree structure and the blocking of tree-nodes. We describe in Section 4 the Gaussian mixture model with hidden Markov random field approach to the segmentation of 3D MR image data sets concerning the human brain. We show how the proposed SPIEM multiresolution *k*d-tree-based algorithms can be adopted to speed up the segmentation process. In Section 5, we report two simulation results on illustrating possible gains of using the proposed algorithms to speed up the EM algorithm with huge image data sets. A study is presented that compares the proposed algorithms with existing algorithms, and Section 6 ends the paper by presenting some concluding remarks and discussion.

2. Multiresolution *k*d-tree approach for the fitting of Gaussian mixtures

With a Gaussian mixture model, the observed p-dimensional data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are assumed to have come from a mixture of an initially specified number g of multivariate Gaussian densities in some unknown proportions $\pi_1, \ldots, \pi_g$, which sum to one. That is, each data point is taken to be a realization of the mixture probability density function,

$$f(\boldsymbol{x}; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i \phi(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{1}$$

where $\phi(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ denotes the p-dimensional multivariate Gaussian distribution with mean $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. Here the vector $\boldsymbol{\Psi}$ of unknown parameters consists of the mixing proportions $\pi_1, \ldots, \pi_{g-1}$, the elements of the component means $\boldsymbol{\mu}_i$, and the distinct elements of the component-covariance matrices $\boldsymbol{\Sigma}_i$.

Within the EM framework, each $\boldsymbol{x}_j$ is conceptualized to have arisen from one of the g components. We let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ denote the unobservable component-indicator vectors, where the ith element z_{ij} of $\boldsymbol{z}_j$ is taken to be one or zero according as the jth data point $\boldsymbol{x}_j$ does or does not come from the ith component. We put $\boldsymbol{z} = (\boldsymbol{z}_1^{\mathrm{T}}, \ldots, \boldsymbol{z}_n^{\mathrm{T}})^{\mathrm{T}}$ where the superscript T denotes vector transpose.

The use of a multiresolution *k*d-tree has been proposed by Moore [17] to speed up the EM algorithm. Here *k*d stands for *k*-dimensional where, in our notation, $k = p$, the dimension of a feature vector $\boldsymbol{x}_j$. The idea of storing the data points in a multiresolution *k*d-tree is also adopted in other clustering algorithms, such as the *k*-means [24,25]. The *k*d-tree is a binary tree that recursively splits the whole set of data into regions. Each node in the *k*d-tree includes a bounding box that specifies a subset of the data points and the root node owns all the data. The children of a node are smaller bounding boxes, generated by splitting along 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 data points 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. In Ref. [17], Moore took γ to be 1% of the range in the splitting dimension of the whole data set.

For Gaussian mixtures, it is computationally advantageous to work in terms of the sufficient statistics [22]. With the help of the multiresolution data structure built up by the kd-tree, the computation of the current conditional expectations of the sufficient statistics in the E-step can be restructured as follows on the $(k+1)$th iteration of the EM algorithm. Let n_L be the total number of leaf nodes. 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 data points in it to have the same posterior probabilities $\tau_i(\bar{\boldsymbol{x}}_m;\boldsymbol{\Psi}^{(k)})$ calculated at the mean, where

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

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

$$T_{i1,m}^{(k)}=\tau_i(\bar{\boldsymbol{x}}_m;\boldsymbol{\Psi}^{(k)})n_m,\qquad \boldsymbol{T}_{i2,m}^{(k)}=\tau_i(\bar{\boldsymbol{x}}_m;\boldsymbol{\Psi}^{(k)})n_m\bar{\boldsymbol{x}}_m,$$

$$\boldsymbol{T}_{i3,m}^{(k)}=\tau_i(\bar{\boldsymbol{x}}_m;\boldsymbol{\Psi}^{(k)})\sum_{j\in LN_m}\boldsymbol{x}_j\boldsymbol{x}_j^{\mathrm{T}} \tag{2}$$

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

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

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

$$\boldsymbol{T}_{i3}^{(k)}=\sum_{j=1}^{n}\tau_{ij}^{(k)}\boldsymbol{x}_j\boldsymbol{x}_j^{\mathrm{T}}\approx\sum_{m=1}^{n_L}T_{i3,m}^{(k)} \tag{5}$$

for $i=1,\ldots,g$, where

$$\tau_{ij}^{(k)}=E(Z_{ij}\,|\,\boldsymbol{x};\boldsymbol{\Psi}^{(k)})\quad(i=1,\ldots,g;\ j=1,\ldots,n)$$

is the current estimate of the posterior probability that $\boldsymbol{x}_j$ comes from the ith component.

The M-step updates the estimates as follows:

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

$$\boldsymbol{\Sigma}_i^{(k+1)}=\{\boldsymbol{T}_{i3}^{(k)}-T_{i1}^{(k)^{-1}}\boldsymbol{T}_{i2}^{(k)}\boldsymbol{T}_{i2}^{(k)^{\mathrm{T}}}\}/T_{i1}^{(k)}. \tag{6}$$

It is noted that the calculation of the sufficient statistics is Eqs. (3)–(5) is approximate, the multiresolution kd-tree algorithm is therefore inexact. In practice, the leaf nodes should be very small (or γ small) in order that the simplified equations (3)–(5) be applicable. However, in this situation, n_L will be close to the number of data points n, and hence there is very little computational gain over the standard EM algorithm.[1] Thus, Moore [17] introduced a further (pruning) step to reduce the computational time.

2.1. Pruning of multiresolution kd-tree

For each component i at a given node $(i=1,\ldots,g)$, compute the minimum and maximum values that any data point in the node can have for its current posterior probabilities. Denote these limiting 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 a pruning criterion (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_d be the number of data points in the dth node, and $\tau_{i,total}$ the sum of the posterior probabilities of ith component membership for all the data points. Based on the source code of Ref. [17], we prune if

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

where $\bar{\boldsymbol{x}}_d$ is the mean of the data points in the node, $\phi_{i,max}$ and $\phi_{i,min}$ are the upper and lower bound on the ith component-conditional density, respectively. For the first condition, if a larger value of β is adopted, the number of pseudo-leaf nodes will decrease. Hence the computational time at each iteration decreases, but the contributions to the conditional expectations of the sufficient statistics may not be well approximated by Eqs. (3)–(5). The second condition ensures that the pruning step will not reduce significantly the value of log likelihood. However, this additional condition increases the number of pseudo-leaf nodes and hence the computational time at each iteration.

In practice, the time to convergence for this algorithm against that without pruning is a tradeoff between the additional computational time needed to compute $\tau_{i,min}$ and $\tau_{i,max}$ $(i=1,\ldots,g)$ and the fewer number of leaf nodes in each scan. There are some possibilities 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 [17]. In addition, it is easy to determine whether the hyper-rectangle of the current node is very far away from the mean μ_i. If it is the case, $\tau_{i,min}$ and $\tau_{i,max}$ may be set to zero and the ith group is not considered in descendants of the current node [17].

[1] Noted from Eq. (2) that the multiresolution kd-tree algorithm speeds up the EM algorithm roughly a factor of n on n_L.

With these two procedures, 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. This process is known as "blacklisting" in Ref. [25] and will considerably reduce the computational time in cases where there are large number of groups and the overlapping of the groups is small.

Due to the implementation of the pruning step, the number of pseudo-leaf nodes at each scan may be different, and hence the approximate log likelihood calculated using the mean of each pseudo-leaf node is not monotonically increasing after each scan. This algorithm can be terminated by considering the convergence of the estimates at each scan.

2.2. *Analytical tools for finding the limiting values of* τ_i

As described in Ref. [17], the computation of $\tau_{i,min}$ and $\tau_{i,max}$ is much easier to formulate in terms of bounds on the density at the data point belonging to the node. Let the Mahalanobis squared distance between vector $\boldsymbol{x}_j$ and $\boldsymbol{\mu}_i$ be

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

The minimum and maximum values of Δ^2 between the mean $\boldsymbol{\mu}_i$ $(i=1,\ldots,g)$ and any data point within the hyper-rectangle are denoted by $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$, respectively. A lower bound on the ith component-conditional density at the data point $\boldsymbol{x}_j$ in the node is then given by

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

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

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

Similarly, an upper bound $\tau_{i,max}$ can be obtained.

In Ref. [17], a quadratic programming with hyper-rectangular constraints is adopted to obtain the limiting values of τ_i $(i = 1,\ldots,g)$. In this paper, we propose a novel analytical geometry approach to obtain $\tau_{i,min}$ and $\tau_{i,max}$. The idea is to transform the data points 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. For data with dimension p less than or equal to three, analytic tools within the context of vector geometry can then be applied to find the minimum and maximum values easily. An example is given in the appendix as an illustration. We found that this analytical geometry approach is faster than the quadratic programming subroutine E04NFF of the FORTRAN NAG library for computing $\tau_{i,min}$ and $\tau_{i,max}$.

3. Sparse and incremental algorithm with a multiresolution *k*d-tree structure

A sparse and incremental version of the EM algorithm (SPIEM) has been considered in Refs. [18,22] to improve the rate of convergence of the EM algorithm. The SPIEM algorithm is formulated by combining the partial E-step of the incremental EM (IEM) algorithm and the sparse E-step of the sparse EM (SPEM) algorithm [18]. With the IEM algorithm, the available n data points are divided into B $(B \leqslant n)$ 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. 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. With the SPEM algorithm, component-posterior probabilities that are below a specified threshold are held fixed while those for the remaining components in the mixture are updated. That is, instead of considering all g components, it is possible to "freeze" those $\tau_{ij}^{(k)}$ that are close to zero and save time. Therefore, this sparse E-step will take time proportional only to the number of components needed to be updated.

To examine the combined sparse and incremental version (SPIEM) 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 [22]. Let $\boldsymbol{\Psi}^{(k+b/B)}$ denote the estimate of $\boldsymbol{\Psi}$ after the bth iteration on the $(k+1)$th scan $(b=1,\ldots,B)$ and S_{b+1} denote the subset of $\{1,\ldots,n\}$ containing the subscripts of those $\boldsymbol{x}_j$ that belong to the $(b+1)$th block $(b=0,\ldots,B-1)$. 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 "nonproper" posterior probabilities of component membership, denoted as $\tau_i^*(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})$, based on the current estimates $\boldsymbol{\Psi}^{(k+b/B)}$ and then form the updated posterior probabilities $\tau_{ij}^{(k+b/B)}$ by rescaling $\tau_i^*(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})$ as

$$\tau_{ij}^{(k+b/B)} = \left[\sum_{h \in A_j^c} \tau_{hj}^{(k-1+b/B)} \right] \frac{\tau_i^*(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})}{\sum_{h \in A_j^c} \tau_h^*(\boldsymbol{x}_j; \boldsymbol{\Psi}^{(k+b/B)})}. \quad (7)$$

This sparse version of the partial E-step thus will take time proportional only to the number of components $i \in A_j^c$ $(j = 1,\ldots,n)$. 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

$$\boldsymbol{T}_{iq}^{(k+b/B)} = \boldsymbol{T}_{iq}^{(k+(b-1)/B)} - \boldsymbol{T}_{iq,b+1}^{(k-1+b/B)} + \boldsymbol{T}_{iq,b+1}^{(k+b/B)} \tag{8}$$

for $b = 0, \ldots, B-1$ and $q = 1, 2, 3$, where the first and the second terms of Eq. (8) are available from the previous iteration and previous scan, respectively. Only the third term on the right-hand side of Eq. (8) have to be calculated by updating only the contribution to the sufficient statistics for those components $i \in A_j^c$. For example,

$$T_{i1,b+1}^{(k+b/B)} = \sum_{j \in S_{b+1}} I_{A_j}(i)\tau_{ij}^{(k-1+b/B)} + \sum_{j \in S_{b+1}} I_{A_j^c}(i)\tau_{ij}^{(k+b/B)}, \tag{9}$$

where $I_{A_j}(i)$ is the indicator function for the set A_j. The first term on the right-hand side of Eq. (9) is calculated at the $(b+1)$th iteration of the kth scan and can be saved for use in the subsequent iteration on the $(k+1)$th scan. Similar arguments apply to $\boldsymbol{T}_{i2}^{(k+b/B)}$ and $\boldsymbol{T}_{i3}^{(k+b/B)}$. In the following, we consider how the SPIEM algorithm performs with a multiresolution kd-tree structure imposed on the data.

3.1. SPIEM with the multiresolution kd-tree (no pruning) algorithm

With the multiresolution kd-tree structure, it can be seen from Section 2 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 SPIEM algorithm based on this kd-tree structure without pruning. That is, the leaf nodes are divided into B blocks (Fig. 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 SPIEM-kd-tree (without pruning) algorithm, Eq. (9) is now replaced by

$$\begin{aligned} T_{i1,b+1}^{(k+b/B)} = & \sum_{m \in S_{b+1}} I_{A_m}(i)\tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k-1+b/B)})n_m \\ & + \sum_{m \in S_{b+1}} I_{A_m^c}(i)\tau_i(\bar{\boldsymbol{x}}_m; \boldsymbol{\Psi}^{(k+b/B)})n_m \end{aligned} \tag{10}$$

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. In Eq. (10), $I_{A_m}(i)$ is the indicator function for the set A_m, which contains the components that are held fixed for the leaf node LN_m.

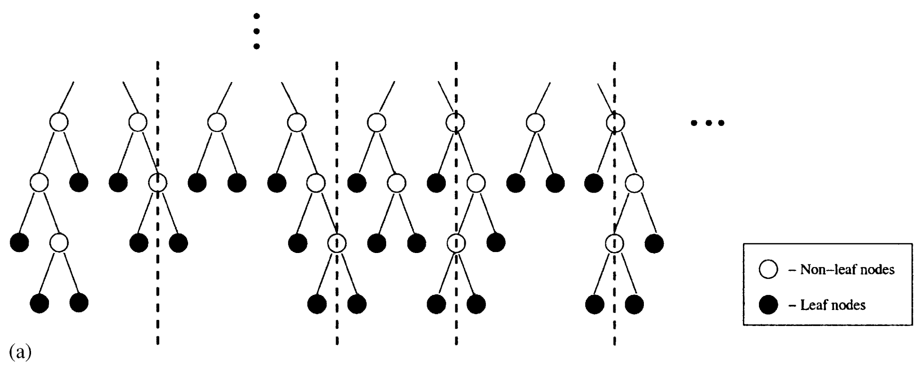
The algorithm is implemented as follows. We choose the number of blocks B based on the simple rule proposed in Ref. [22]. To avoid the problem of premature component starvation, the standard EM step is performed on multiresolution kd-tree leaf nodes in the first scan, followed by five scans with the IEM step. We fix the set A_m $(m = 1, \ldots, n_L)$ obtained from the last IEM scan and run five scans with the spares version SPIEM step. An IEM step is then performed to determine a new set of A_m. The flowchart of the algorithm is depicted in Fig. 1(b).

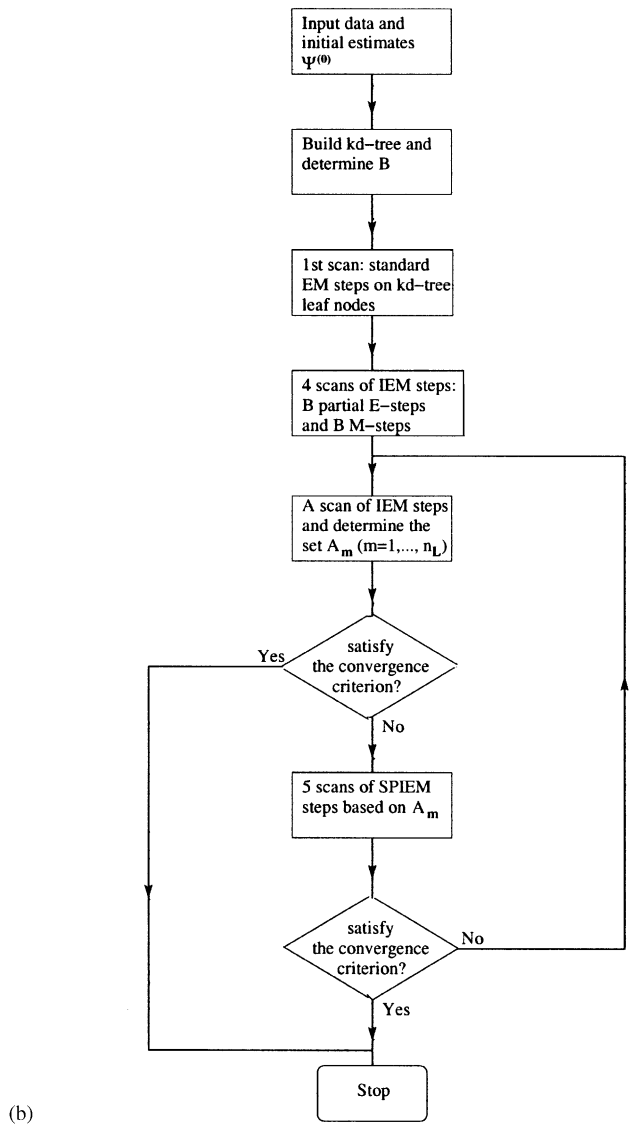
3.2. SPIEM with multiresolution kd-tree (with pruning) algorithm

With the multiresolution kd-tree (with pruning) algorithm, the number of pseudo-leaf nodes is, however, different at each scan. Thus, the division of pseudo-leaf nodes into blocks is not so straightforward as that for leaf nodes in the SPIEM-kd-tree (without pruning) algorithm. This difficulty can be solved by using an alternative procedure to divide the tree nodes. Instead of dividing the nodes at the leaves of the kd-tree (leaf-nodes) as in the SPIEM-kd-tree (without pruning) algorithm, we divide at some level of the kd-tree, say L, that the number of nodes at that level is much larger than B (Fig. 2(a)), where the number of blocks B is again so chosen based on the rule of Ng and McLachlan [22]. On the first scan of the algorithm, we perform the standard EM step on kd-tree leaf nodes without pruning. During this scan, we search down from each node at level L and determine the number of leaf nodes under each node at that level. We then divide the nodes at level L into roughly B blocks such that numbers of leaf nodes within each block are similar (Fig. 2(a)). The division of nodes into block of similar sizes ensures better convergence of the algorithm. In practice, there could be some leaf nodes present above the level L. These leaf nodes are grouped into the first block at level L.

With this SPIEM-kd-tree (with pruning) algorithm, the partial E-step is implemented by searching down from only a block of nodes at level L at a time before the next M-step is performed. It is noted that the number of pseudo-leaf nodes in each block is different, which is in contrast to the SPIEM-kd-tree (without pruning) algorithm where the number of leaf nodes in each block can be set to be exactly the same except the last block.

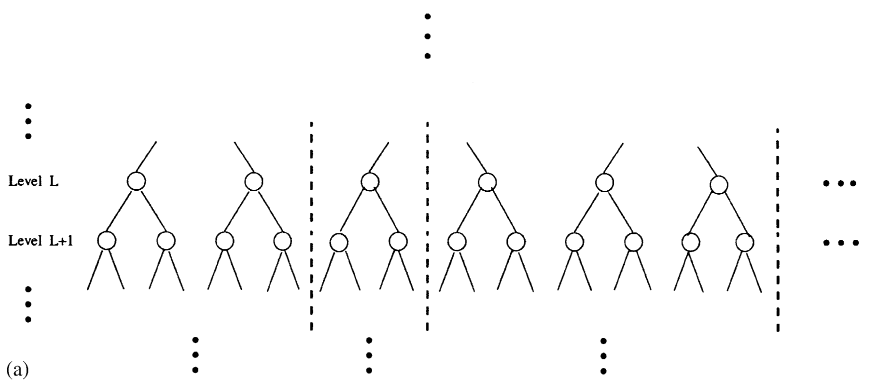
To further speed up the algorithm, we do not prune on every scan. Our procedure is to perform the standard EM step on kd-tree leaf nodes on the first scan, followed by five scans of IEM step with pruning process. We then fix the set A_m and pseudo-leaf nodes obtained from the last IEM scan. Five scans of the SPIEM step is then performed by searching only on those fixed pseudo-leaf nodes. That is, no pruning step is required in these five scans and hence computational time is saved. An IEM scan with pruning process is then performed to determine a new set of pseudo-leaf nodes and a new set of A_m $(m = 1, \ldots, n_{PL})$, where n_{PL} is the number of pseudo-leaf nodes at this scan. Both of them are fixed in the next five scans of SPIEM steps. The flowchart of the algorithm is depicted in Fig. 2(b). This new algorithm improves dramatically the rate of convergence of the multiresolution kd-tree (with pruning) algorithm.





Input data and initial estimates $\Psi^{(0)}$
Build kd–tree and determine B
1st scan: standard EM steps on kd–tree leaf nodes
4 scans of IEM steps: B partial E–steps and B M–steps
A scan of IEM steps and determine the set A_m (m=1,..., n_L)
satisfy the convergence criterion?
Yes
No
5 scans of SPIEM steps based on A_m
satisfy the convergence criterion?
No
Yes
Stop
(b)

Fig. 1. THE SPIEM-*k*d-tree (without pruning) algorithm: (a) the partition of leaf nodes (say blocks of 6 leaf nodes) and (b) the flowchart of the algorithm.



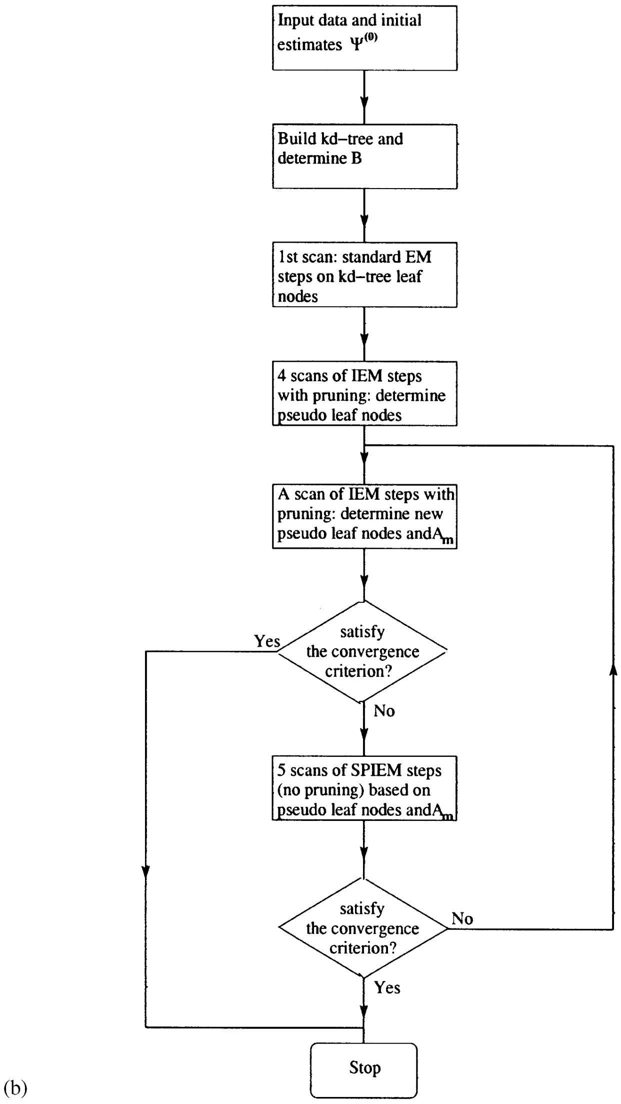

Input data and initial estimates $\Psi^{(0)}$
Build kd–tree and determine B
1st scan: standard EM steps on kd–tree leaf nodes
4 scans of IEM steps with pruning: determine pseudo leaf nodes
A scan of IEM steps with pruning: determine new pseudo leaf nodes and A_m
satisfy the convergence criterion?
Yes
No
5 scans of SPIEM steps (no pruning) based on pseudo leaf nodes and A_m
satisfy the convergence criterion?
No
Yes
Stop
(b)

Fig. 2. The SPIEM-*k*d-tree (with pruning) algorithm: (a) the partition of nonleaf nodes at Level L into blocks (number of leaf nodes under each block of nodes at Level L should be similar) and (b) the flowchart of the algorithm.

4. Mixture model-based approach to segment MR images

We consider here the Gaussian mixture model with hidden Markov random field approach of Ref. [11] to segment a real three-dimensional (3D) MR image data set concerning the human brain. Suppose that a continuous MR image is partitioned into a set of disjoint voxels labeled 1 to n, and that each voxel is assumed to belong to one of g distinct tissue types. This assumption is tenable because MR images have a spatial resolution at the range of the voxel size [27]. We let $\boldsymbol{x}_j$ denote the 3D features vector containing the values of the variables T_1, T_2, and ρ_D for the jth voxel ($j = 1, \ldots, n$). Within the context of image segmentation, the problem is to infer the unknown vector of component indicator $\boldsymbol{z}$ from the observed data $\boldsymbol{x} = (\boldsymbol{x}_1^{\mathrm{T}}, \ldots, \boldsymbol{x}_n^{\mathrm{T}})^{\mathrm{T}}$.

As detailed in Ref. [11], the segmentation of MR images is firstly implemented via a noncontextual approach where the spatial characteristics of each voxel is ignored. This noncontextual process provides fuzzy classification of tissue type τ_{ij} for each voxel and estimates of unknown parameter $\boldsymbol{\Psi}$. By assuming $\boldsymbol{x}_j$ be independent and identically distributed, the estimation of $\boldsymbol{\Psi}$ corresponds to the maximum likelihood estimation from incomplete data via the EM algorithm [11,27]. Thus, the process can be speeded up by the variants of the EM algorithm described in the previous section.

The segmentation is finalized by the iterative computation of a "contextual" process, where the spatial characteristics of each voxel is involved in the estimation. Briefly, the spatial correlation in image intensity between voxels and their neighbours is captured by the Markov random field prior in which

$$\log \pi_{ij}^{(k+1)} \propto \xi \left(\sum_m \tau_{im}^{(k)} + 1/\sqrt{2} \sum_m \tau_{im}^{(k)} + 1/\sqrt{3} \sum_m \tau_{im}^{(k)} \right), \tag{11}$$

where

$\pi_{ij}^{(k+1)} = \mathrm{pr}\{z_{ij} = 1 | z_{\delta j} = \tau_{\delta j}^{(k)}\}$

is the probability that the jth voxel belongs to the ith tissue type given the component membership of its specified neighbours δj as implied by $\tau_{\delta j}^{(k)}$. The summations on the right-hand side of Eq. (11) are over the prescribed first-, second-, and third-order neighbours, respectively, of the jth voxel. The values $1/\sqrt{2}$ and $1/\sqrt{3}$ reflect the spatial relatedness between a central voxel and its second- and third-order neighbouring voxels, respectively [9,11]. With this contextual segmentation, the E-step is approximated by the conditional expectation of z_{ij} given $\boldsymbol{x}$ and the current component-membership of the neighbours of the jth voxel:

$$\tau_{ij}^{(k)} \approx \pi_{ij}^{(k)} \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_i^{(k)}, \boldsymbol{\Sigma}_i^{(k)}) \Big/ \sum_{l=1}^{g} \pi_{lj}^{(k)} \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_l^{(k)}, \boldsymbol{\Sigma}_l^{(k)}). \tag{12}$$

With the specification of ξ a priori, the M-step is in closed form and can be implemented as in the noncontextual case (6).

In this real example, the data set was five slices of a 3D MR image acquired by a two-Tesla Bruker Medspac whole body scanner. The acquisition matrix was $256 \times 256 \times 256$. Fig. 3 displays the T_1-, T_2-, and ρ_D-weighted images of one of the five sliced images. The number of voxels was $n = 256 \times 256 \times 5 = 327,680$. In the analysis, the image intensities were scaled to the range of (0,20) for the parameter estimation. We assumed $g = 7$ and adopted the proposed SPIEM multiresolution kd-tree-based algorithms to the noncontextual segmentation process. We considered the threshold γ to be 0.7%, 0.5%, and 0.3% of the range in the splitting dimension of the whole data set. The algorithms were terminated when the absolute values of the relative changes in the estimates of the means all fell below 0.001.

For comparison, we also consider some existing variants of the EM algorithm to speed up the noncentextual process. These include an inexact IEM algorithm and a subsampling approach where a randomly selected subset of data points of size n_s is used in the estimation. The inexact IEM algorithm was proposed by Nowlan [28] where the sufficient statistics was calculated as an exponentially decaying average of recently visited data. Instead of Eq. (8), the current conditional expectations of the sufficient statistics are obtained by

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

where $0 < \alpha < 1$ is a decay constant. The subsampling method in general speeds up the EM algorithm roughly a factor of n on n_s. Here, 10 randomly selected subsets of the simulated data were used and the average was presented. The relative performances of these algorithms are summarized in Table 1.

In the contextual process, we adopted $\xi = 0.6$ and considered the first- and second-order neighbours (first and second term, respectively) in Eq. (11). Three scans of contextual process were performed and the CPU time required was about one minute. Fig. 4 depicts the final segmented image of the three main tissue types, cerebral spinal flow (CSF), white matter, and gray matter from some chosen algorithms. It can be seen that the details of the three main tissue types are all very well classified by the algorithms.

From Table 1, it can be seen that all the variants of the EM algorithms, except the subsampling approach, have convergence as reliable as the standard EM algorithm. The log likelihood values are found to be monotonically increasing after each scan. These algorithms are marked with an asterisk in Table 1. Moreover, it can be seen from Table 1 that the standard EM, the IEM, the SPIEM, the inexact IEM (with $\alpha = 0.97$), and multiresolution kd-tree-based algorithms (without pruning, $\gamma = 0.003$) converged to

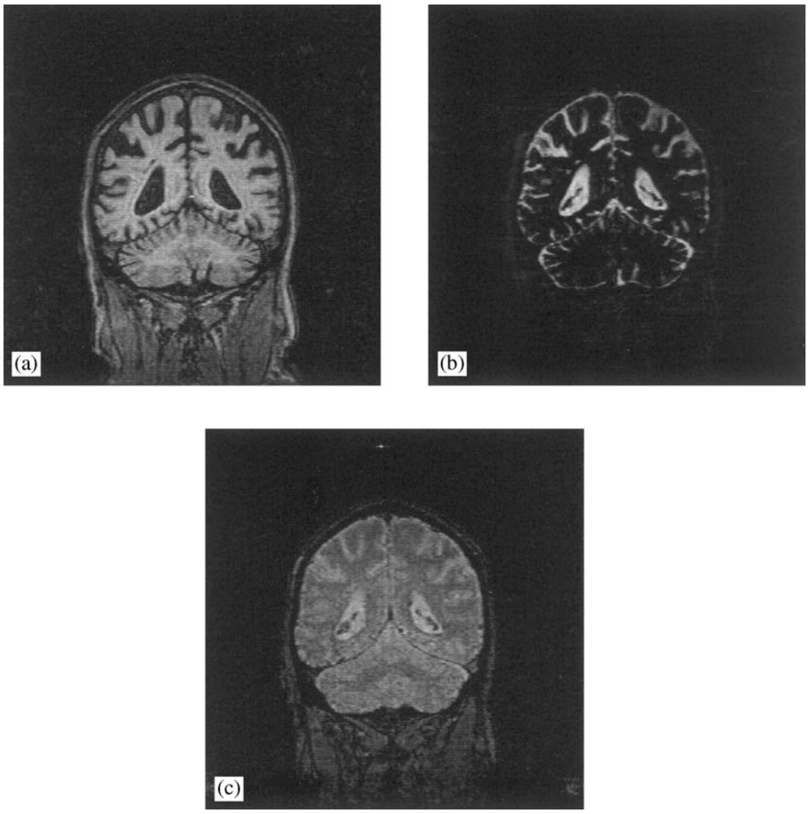


Fig. 3. Real MR data set: (a) T_1-weighted image (top left); (b) T_2-weighted image (top right); (c) ρ_D-weighted image (bottom).

essentially the same log likelihood value. Among these six algorithms, the SPIEM-kd-tree (without pruning) algorithm has the largest speedup ratio. To evaluate the relative performance of variants of the EM algorithm for speeding up the noncontextual segmentation of huge image data more closely, we perform a comparative study using simulated 3D MR data.

5. Simulation experiments

A random sample of size n observations was generated from a seven-component trivariate normal mixture ($g = 7$, $p = 3$). The estimates obtained in Ref. [27, Table II] were used as the values of our population parameters. 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 the simulation study, we consider two different sample sizes of $n = 128^3$ and 256^3, respectively, which correspond to typical number of voxels of a 3D MR image. All the algorithms used the same initial estimates as starting values and were terminated when the absolute values of the relative changes in the estimates of the means all few below 0.0001. The algorithms are written in FORTRAN and the simulations are all run on a Sun unix work-station. The results of the simulation study are summarized in Tables 2 and 3, respectively. From Tables 1–3, the characteristics of variants of the EM algorithm are summarized as follows.

(i) *IEM/SPIEM algorithms*: The IEM algorithm is exact as accurate sufficient statistics are strictly maintained in each scan. The SPIEM algorithm is also exact provided that occasional full partial E-steps are performed to obtain a new set of A_j ($j = 1, \ldots, n$). Thus, both the IEM and SPIEM algorithms have reliable convergence as the standard EM algorithm. However, as both 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. In Ref. [22], empirical studies on both algorithms using various settings of n, g, and p were reported, the speedup factors (relative to the EM algorithm) ranged from 1.3 to 1.8 to 2.1 to 2.3 for the IEM and SPIEM algorithms, respectively. These values agree with those presented in Tables 1–3. In applications to huge data sets, these two algorithms can be

Table 1
Results for real 3D MR image data ($n = 256 \times 256 \times 5$)

Algorithm[a]	CPU[b]	nscan[c]	Log likelihood[d]	Speedup[e]
Standard EM*	1251.2	70	$-1,028,022$	1.0
IEM* ($B = 160$)	916.7	41	$-1,028,019$	1.4
SPIEM* ($B = 160$)	470.0	44	$-1,028,019$	2.7
kd-tree (no pruning)				
$\gamma = 0.007$*	320.1	67	$-1,028,343$	3.9
$\gamma = 0.005$*	430.2	68	$-1,028,194$	2.9
$\gamma = 0.003$*	618.6	69	$-1,028,045$	2.0
kd-tree (pruning)				
$\gamma = 0.007$*	649.9	70	$-1,028,270$	1.9
$\gamma = 0.005$*	762.6	70	$-1,028,202$	1.6
$\gamma = 0.003$*	1053.0	71	$-1,028,085$	1.2
Inexact IEM ($B = 160$)				
$\alpha = 0.95$*	334.9	14	$-1,028,076$	3.7
$\alpha = 0.96$*	377.1	16	$-1,028,060$	3.3
$\alpha = 0.97$*	486.4	21	$-1,028,045$	2.6
Subsampling approach				
$n_s = n/8$	169.2	74	$-1,028,388$	7.4
$n_s = n/5$	258.3	72	$-1,028,194$	4.8
SPIEM-kd-tree (no pruning)				
$\gamma = 0.007$*	124.3	46	$-1,028,330$	10.1
$\gamma = 0.005$*	173.2	47	$-1,028,182$	7.2
$\gamma = 0.003$*	226.6	44	$-1,028,038$	5.5
SPIEM-kd-tree (pruning)				
$\gamma = 0.007$*	174.1	54	$-1,028,266$	7.1
$\gamma = 0.005$*	206.6	52	$-1,028,183$	6.1
$\gamma = 0.003$*	285.2	54	$-1,028,076$	4.3

[a] Algorithm is marked with an asterisk if the log likelihood value calculated using the estimates at each scan is monotonically increasing.
[b] CPU represents the CPU time in seconds for various algorithms, it includes the time to construct the kd-tree for kd-tree-based algorithms.
[c] nscan indicates the number of scans to convergence.
[d] Log likelihood is the value of the log likelihood calculated at the final estimates obtained.
[e] Speedup is the ratio of the CPU time compared to that of the standard EM algorithm.

useful at the final stage of an iterative-computational segmentation process, where an inexact method is performed initially and then an exact method with reliable convergence property is adopted to obtain the final estimates.

(ii) *Inexact IEM algorithm*: The inexact IEM algorithm requires fewer scans to convergence when n increase. As it forgets out-of-date sufficient statistics more rapidly, it can have larger speedup factor compared to the IEM algorithm provided an appropriate value of the decay constant α were adopted. In general, a larger value of α provides a larger final log likelihood value. In particular, when α is close to one, the log likelihood value obtained will be close to that obtained by the IEM or the SPIEM algorithms. However, the algorithm takes longer time to converge. Moreover, it is found from the simulation studies that the inexact IEM algorithm needs an adequate randomization of the data such that each block of the data has data points from each of the components. Otherwise, the algorithm will converge to points that would be very poor estimates. An appropriate choice of α seems to depend on n and its value is also crucial to the ultimate qualities of the final estimates.

(iii) *Subsampling approach*: The algorithm in general provides smaller log likelihood value and larger error rate compared to other variants of the EM algorithm though its performance improves when the sample size increases. The quality of final estimates however depends heavily on the randomly selected subset of the data. This effect is more pronounced in the application to the real data set (Table 1). Moreover, even a reasonably large proportion of the original data was randomly selected to include in the

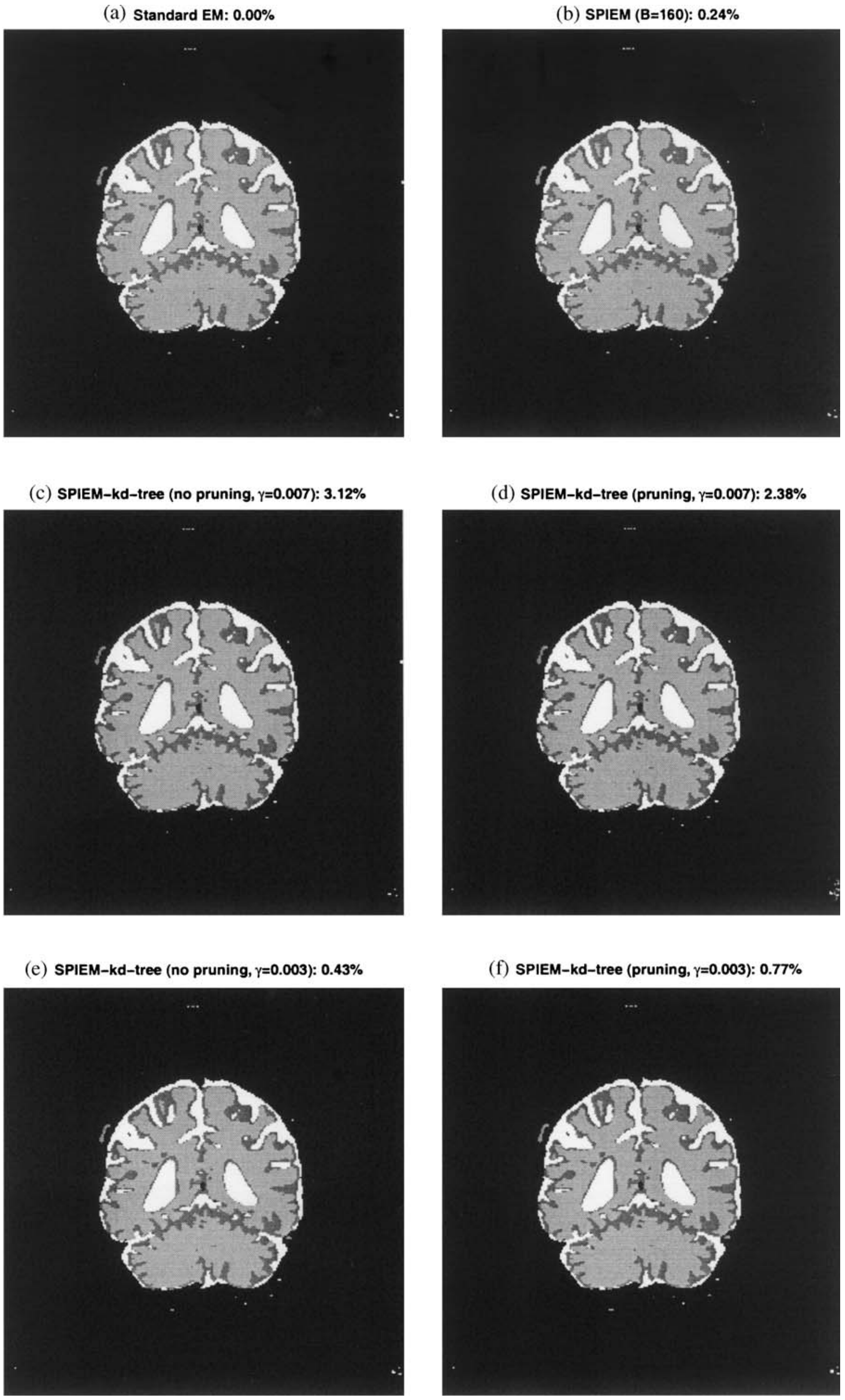

(a) Standard EM: 0.00%
(b) SPIEM (B=160): 0.24%
(c) SPIEM-kd-tree (no pruning, γ=0.007): 3.12%
(d) SPIEM-kd-tree (pruning, γ=0.007): 2.38%
(e) SPIEM-kd-tree (no pruning, γ=0.003): 0.43%
(f) SPIEM-kd-tree (pruning, γ=0.003): 0.77%

Fig. 4. Final segmented images from some algorithms, along with the percentages of voxels allocated to different tissue types compared to the segmentation obtained by the standard EM algorithm.

Table 2
Results for Simulation 1 ($n = 128 \times 128 \times 128$)

Algorithm	CPU	nscan	Log likelihood	Error (%)[a]	Speedup
Standard EM*	10,723	95	$-11,749,600$	11.979	1.0
IEM* ($B = 256$)	7132	54	$-11,749,599$	11.979	1.5
SPIEM* ($B = 256$)	4528	60	$-11,749,599$	11.978	2.4
*k*d-tree (no pruning)					
$\gamma = 0.007$*	1137	94	$-11,749,649$	11.982	9.4
$\gamma = 0.005$*	2069	95	$-11,749,618$	11.979	5.2
$\gamma = 0.003$*	3927	95	$-11,749,601$	11.977	2.7
*k*d-tree (pruning)					
$\gamma = 0.007$	3403	94	$-11,749,646$	11.984	3.2
$\gamma = 0.005$	4589	94	$-11,749,622$	11.981	2.3
$\gamma = 0.003$	5988	94	$-11,749,610$	11.980	1.8
Inexact IEM ($B = 256$)					
$\alpha = 0.95$	1693	12	$-11,749,707$	11.984	6.3
$\alpha = 0.97$*	2497	18	$-11,749,661$	11.980	4.3
$\alpha = 0.99$*	5862	44	$-11,749,610$	11.978	1.8
Subsampling approach					
$n_s \approx n/20$	522	95	$-11,750,312$	11.992	20.5
$n_s \approx n/10$	1041	95	$-11,749,920$	11.986	10.3
SPIEM-*k*d-tree (no pruning)					
$\gamma = 0.007$*	457	61	$-11,749,649$	11.982	23.5
$\gamma = 0.005$*	773	61	$-11,749,618$	11.979	13.9
$\gamma = 0.003$*	1431	61	$-11,749,600$	11.978	7.5
SPIEM-*k*d-tree (pruning)					
$\gamma = 0.007$	735	61	$-11,749,648$	11.985	14.6
$\gamma = 0.005$	1013	61	$-11,749,624$	11.982	10.6
$\gamma = 0.003$	1327	61	$-11,749,611$	11.979	8.1

[a]Error is the proportion of misclassified data points.

estimation, this subsampling approach cannot provide accurate estimates and cannot guarantee than the log likelihood value is monotonically increasing after each scan. For example, with $n = 128^3 = 2,097,152$ (Table 2), the averaged log likelihood value obtained by using $n_s = 209,715 (\approx n/10)$ randomly selected data is still far from the maximum log likelihood obtained by the standard EM algorithm. Some trails also do not possess reliable convergence. The same situation occurs with $n = 256^3 = 16,777,216$ (Table 3), where $n_s = 838,861 (\approx n/20)$ is used. More effective sampling techniques [29,30] may be adopted to improve the quality of final estimates. The subsampling approach also has a limitation that the mixing proportions π_i $(i = 1, \ldots, g)$ must be large enough so that a random sample of the data will involve a nonnegligible amount of data points from each component.

(iv) *SPIEM-k*d-*tree* (*no pruning*) *algorithm*: The algorithm reduces the time to convergence by imposing a multiresolution *k*d-tree structure on the data. It can be seen from Tables 1–3 that the number of leaf nodes n_L does not linearly increase with n. For example, with $\gamma = 0.007$, $n_L = 62,753$, 148,876 and 257,968 for $n = 256^2 \times 5$, 128^3, and 256^3, respectively. Hence the speedup factor increases when the sample size increases. As an approximation is made in the calculation of the sufficient statistics, the algorithm is inexact. However, it can be seen from Tables 1–3 that for a sufficiently small value of γ (smaller sized leaf nodes), the algorithm possesses reliable monotonic convergence (with $\gamma \leqslant 0.007$) and converges to essentially the same maximum log likelihood value as the standard EM algorithm (with $\gamma \leqslant 0.003$). One can compromise between the speed and the accuracy by choosing an appropriate value for γ.

(v) *SPIEM-k*d-*tree* (*with pruning*) *algorithm*: Similar to the above algorithm (iv), the speedup factor for the SPIEM-*k*d-tree (with pruning) algorithm increases when the sample size increases. However, due to the implementation of the pruning step, the number of pseudo-leaf nodes at each scan will be different. It is found in the simulation studies that the log likelihood value is not monotonically

Table 3
Results for Simulation 2 ($n = 256 \times 256 \times 256$)

Algorithm	CPU	nscan	Log likelihood	Error (%)	Speedup
Standard EM*	88,950	95	$-94,015,922$	11.989	1.0
IEM* ($B = 1024$)	59,425	54	$-94,015,918$	11.989	1.5
SPIEM* ($B = 1024$)	35,845	62	$-94,015,918$	11.989	2.5
*k*d-tree (no pruning)					
$\gamma = 0.007$*	2949	95	$-94,016,387$	11.993	30.2
$\gamma = 0.005$*	5079	95	$-94,016,148$	11.992	17.5
$\gamma = 0.003$*	10,138	95	$-94,015,939$	11.989	8.8
*k*d-tree (pruning)					
$\gamma = 0.007$	6682	94	$-94,016,329$	11.994	13.3
$\gamma = 0.005$	8916	94	$-94,016,138$	11.992	10.0
$\gamma = 0.003$	11,921	94	$-94,016,001$	11.990	7.5
Inexact IEM ($B = 1024$)					
$\alpha = 0.98$	13,274	11	$-94,028,005$	12.005	6.7
$\alpha = 0.99$*	16,309	14	$-94,020,334$	11.994	5.5
$\alpha = 0.997$*	42,495	38	$-94,016,277$	11.990	2.1
Subsampling approach					
$n_s \approx n/40$	2217	96	$-94,018,005$	11.995	40.1
$n_s \approx n/20$	4443	97	$-94,017,136$	11.993	20.0
SPIEM-*k*d-tree (no pruning)					
$\gamma = 0.007$*	1698	62	$-94,016,387$	11.993	52.4
$\gamma = 0.005$*	2585	61	$-94,016,147$	11.992	34.4
$\gamma = 0.003$*	4389	61	$-94,015,937$	11.988	20.3
SPIEM-*k*d-tree (pruning)					
$\gamma = 0.007$	2132	61	$-94,016,330$	11.994	41.7
$\gamma = 0.005$	2738	60	$-94,016,133$	11.992	32.5
$\gamma = 0.003$	3465	61	$-94,016,002$	11.990	25.7

increasing after each scan. However, the pruning step can be an advantage in some cases. From Tables 2 and 3, it can be seen that, when $\gamma \leqslant 0.003$, the SPIEM-*k*d-tree (with pruning) algorithm converges faster than the version without pruning. Moreover, the algorithm with pruning step does increase monotonically the log likelihood to a maximum value, similar to the final log likelihood of the standard EM algorithm, before the log likelihood value starts to decrease just before the convergence of the estimates. Therefore, the SPIEM-*k*d-tree (with pruning) algorithm is useful at the early stage to speed up the segmentation process. Then it is switched to the version without pruning for convergence to the final estimates, as the latter possesses reliable convergence. Based on the empirical studies performed, the algorithm with pruning step can be run until the absolute values of the relative changes in the estimates of the means fell below 0.001 (Table 1) before it is switched. Sand and Moore [31], on the other hand, proposed to run a fixed number of scans, say 10, and then modify the model using density estimation with *k*d-trees.

To conclude, it can be seen from Tables 2 and 3 that the standard EM, the IEM, the SPIEM, and multiresolution *k*d-tree-based algorithms ($\gamma = 0.003$) converged to essentially the same log likelihood value. Among them, the proposed SPIEM-*k*d-tree algorithms (with or without pruning) are the two fastest. The version without pruning also possesses reliable convergence as the log likelihood values are monotonically increasing after each scan.

6. Discussion

We have studied the use of a multiresolution *k*d-tree structure for speeding up the SPIEM algorithm for mixture model-based image segmentation. An analytical geometry approach has been considered to speed up the pruning process of *k*d-trees. In Section 3, we have presented how the SPIEM multiresolution *k*d-tree-based algorithms are implemented via the blocking of tree-nodes. The applicabilities and relative performances of these new algorithms,

Table 4
Number of leaf nodes (n_L) versus the dimension of data

Dimension	$n = 1,000,000$		$n = 5,000,000$	
	n_L	n/n_L	n_L	n/n_L
$p = 4$	138,502	7.22	263,682	18.96
$p = 5$	340,815	2.93	905,241	5.52
$p = 6$	582,230	1.72	1,798,116	2.78
$p = 7$	858,157	1.17	3,582,999	1.40
$p = 8$	940,475	1.06	4,328,097	1.16
$p = 9$	972,260	1.03	4,679,888	1.07
$p = 10$	982,294	1.02	4,786,099	1.05

compared with other variants of the EM algorithm, for speeding up the segmentation process has been illustrated using the real 3D MR image data (Section 4) and simulated 3D image data (Section 5).

The comparative study presented in Section 5 provides a guide to the possible gains in CPU time to convergence, the quality of the solution, and the convergence properties for each variant of the EM algorithm. In particular, it was found that the IEM, the SPIEM, and the kd-tree-based (without pruning, $\gamma \leqslant 0.007$) algorithms, along with the standard EM algorithm, possess reliable convergence and the log likelihood values are monotonically increasing after each scan.

The multiresolution kd-tree structure has some limitations of which we have not yet addressed. Firstly, in some cases, data may actually appear in block in a time and/or there may be confines of a limited memory buffer. In such situations, a scalable version of the EM algorithm proposed recently by Bradley et al. [20] may be adopted to handle huge data sets. The algorithm is based on identifying regions of the data that are compressible and regions that must be maintained in memory. It works within the confines of a limited memory buffer and is "resumable" such that incremental progress can be saved for continued computation later, possibly on new data. Our SPIEM multiresolution kd-tree-based algorithms may also be applicable in these problems. Instead of building the multiresolution kd-tree until all the data are available, multiresolution "kd-sub-tree" can be built based on the block of data available and the sufficient statistics calculated is stored for later processing.

Secondly, the number of leaf nodes will increase dramatically when the dimension of the data points p increases. This implies that multiresolution kd-tree-based algorithms will not be able to speed up the EM algorithm for applications to high dimensional data sets [21,26]. To see how the number of leaf nodes is affected by the dimension p, we generate data points from a mixture of p-variate normal and report the number of leaf nodes in each case. We consider $g = 5$ with equal mixing proportions and set each covariance matrix to be diagonal matrix with its diagonal elements generated randomly between (0,0.1). The elements of each mean $\boldsymbol{\mu}_i$ are generated randomly between (0,10). The result is displayed in Table 4. It can be seen that with $n = 1,000,000$, the upper bound on the dimension is $p = 6$, beyond which 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 further reduce the number of leaf nodes. With $n = 5,000,000$, the upper bound is $p = 7$. Recently, a number of techniques have been developed to reduce dimensionality without losing significant information and separability among components. For example, Jimenez and Landgrebe [32] adopted a parametric-projection pursuit method to reduce the dimensionality of the data by seeking a set of linear projections that minimizes the Bhattacharyya distance among the components. Their method, however, requires a training set with labeled samples to estimate the Bhattacharyya distance under a parametric assumption. Dasgupta [33] considered a projection of the original data to a randomly chosen subspace via a linear map. This random projection method can map data from a mixture of g Gaussians into $O(\log g)$ dimensions. Further exploration of these dimensionality reduction methods is required to study the distortion induced in the mapping between the original and the projected spaces and study its effectiveness in applications to huge data sets.

7. Summary

Mixture models implemented via the expectation-maximization (EM) algorithm are being increasingly used in a wide range of problems in statistical pattern recognition such as image segmentation. In this paper, we consider a fast EM-based mixture model approach to segment three-dimensional (3D) magnetic resonance (MR) images. It can segment automatically a 3D MR image of 256^3 voxels in less than 1/50th of the time taken using the standard EM algorithm. The method thus provides an useful aid in surgical planning and the diagnosis of patients, as well as a means for monitoring changes in brain haemodynamics and metabolism resulting from neuronal activity. Major contributions of our work includes

(1) A sparse, incremental EM (SPIEM) algorithm with a multiresolution *k*d-tree structure is proposed for speeding up the EM algorithm via the "blocking" of tree nodes. We divide the leaf nodes (the smallest possible partitions the *k*d-tree offers) into blocks and a "partial" E-step is implemented for only a block of nodes at a time before the next M-step is performed. The proposed algorithm improves the rate of convergence as its exploits new information more quickly rather than waiting for a complete scan of the data before parameters are updated by an M-step.
(2) We also consider a second version that involves "pruning" the tree-nodes. A novel analytical geometry approach is proposed to speed up this pruning process.
(3) With pruning step, a node which satisfies some pruning criterion is treated as a "pseudo"-leaf node. The blocking of pseudo-leaf nodes is not so straightforward as that for leaf nodes because the number of pseudo-leaf nodes is different at each iteration. Here, we propose an alternative procedure for blocking pseudo-leaf nodes.
(4) In the implementation of the SPIEM multiresolution *k*d-tree (with pruning) algorithm, we propose to "freeze" the pseudo-leaf nodes at some iterations so as to obtain further reduction in the time to convergence. This new algorithm improves dramatically the rate of convergence of the multiresolution *k*d-tree (with pruning) algorithm.
(5) We compare the two new algorithms with existing algorithms, using simulated and real 3D MR data. Our focus is on providing a guide to the possible gains in CPU time to convergence and the convergence properties of each algorithm. This study helps to advance our understanding of existing algorithms for speeding up the EM algorithm. By adjusting the tuning parameters, we show that proposed algorithms can provide an accurate solution and preserve the reliable convergence as that for the EM algorithm.

Acknowledgements

The authors wish to thanks the Editor and the referee for helpful comments on the paper. The authors wish to thanks Dr. Andrew Moore and Mr. Peter Sand for helpful discussion on the pruning criteria for the multiresolution *k*d-tree algorithm. This work was supported in part by grants from the Australian Research Council.

Appendix A.

A.1. Analytical tools for finding the limiting values of τ_i: an example

As an illustration, we consider a 2D ($p=2$) case. Suppose that we need to obtain $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$ between any data points within the node and the mean $\boldsymbol{\mu}_i$ which is depicted by the ellipse in Fig. 5(a). Here, we set

$$\boldsymbol{\mu}_i = (4,5)^{\mathrm{T}}, \quad \text{and} \quad \boldsymbol{\Sigma}_i = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

The minimizer and maximizer of Δ^2 are found to be $\Delta^2_{min} = 0.5$ at (3.5,4) and $\Delta^2_{max} = 14$ at (5,1), respectively. Now, for $\boldsymbol{\Sigma}_i$ as above, the matrix of the normalized eigenvectors, $\boldsymbol{M}$,

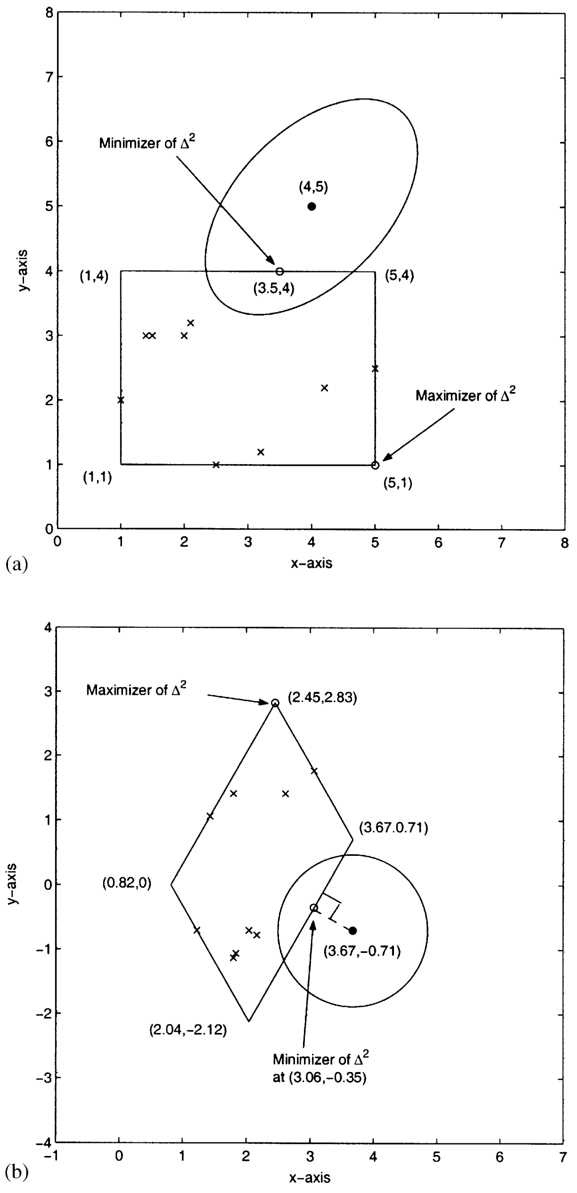


Fig. 5. The minimizer and maximizer of Δ^2: (a) original data with rectangular node and elliptical covariance matrix and (b) transformed data with parallelogram-shaped node and circular covariance matrix. The crosses denote data points belonging to the node.

is given by

$$\boldsymbol{M} = \begin{pmatrix} 1/(\sqrt{3}\sqrt{2}) & 1/(\sqrt{3}\sqrt{2}) \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}.$$

Under the transform

$$\boldsymbol{x} \mapsto \boldsymbol{Mx},$$

the covariance matrix $\boldsymbol{\Sigma}_i$ becomes an identify matrix, as depicted by the circle in Fig. 5(b), which implies that $\Delta^2_{i,min}$ and $\Delta^2_{i,max}$ can be obtained simply by computing the limiting values of Euclidean squared distance between the new transformed mean and any data point within the transformed node. However, the rectangular boundary of the node will be transformed into a parallelogram boundary. For p less than or equal to three, we can still easily compute the minimum and maximum Euclidean squared distance using analytical tools within the context of vector geometry. For example, the minimum Euclidean squared distance is the square of the distance between the new mean $(3.67, -0.71)$ and the line formed by points $(2.04, -2.12)$ and $(3.67, 0.71)$ and is found to be 0.5. The maximum Euclidean squared distance is the square of the distance between the new mean and the furthest vertex of the parallelogram and is found to be 14.

References

[1] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (1) (2000) 4–38.

[2] S.J. McKenna, S. Gong, Y. Raja, Modelling facial colour and identity with Gaussian mixtures, Pattern Recognition 31 (12) (1998) 1883–1892.

[3] G.J. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, Wiley, New York, 1992 (Chapter 13).

[4] M. Freund, S. Hahnel, M. Thomsen, K. Sartor, Treatment planning in severe scoliosis: the role of MRI, Neuroradiology 43 (6) (2001) 481–484.

[5] G.D. Cascino, C.R. Jack, J.E. Parisi, F.W. Sharbrough, K.A. Hirschorn, F.B. Meyer, W.R. Marsh, P.C. O'Brien, Magnetic resonance imaging-based volume studies in temporal lobe epilepsy: pathological correlations, Ann. Neurol. 30 (1) (1991) 31–36.

[6] R.L. Wolf, D.C. Alsop, I. Levy-Reis, P.T. Meyer, J.A. Maldjian, J. Gonzalez-Atavales, J.A. French, A. Alavi, J.A. Detre, Detection of mesial temporal lobe hypoperfusion in patients with temporal spin labeled lobe epilepsy by use of arterial perfusion MR imaging, Am. J. Neurolradiol. 22 (7) (2001) 1334–1341.

[7] L. Jancke, T.W. Buchanan, K. Lutz, N.J. Shah, Focused and nonfocused attention in verbal and emotional dichotic listening: an FMRI study, Brain Lang. 78 (3) (2001) 349–363.

[8] D. Stiller, B. Gaschler-Markefski, F. Baumgart, F. Schindler, C. Tempelmann, H.J. Heinze, H. Scheich, Lateralized processing of speech prosodies in the temporal cortex: a 3-T functional magnetic resonance imaging study, Magn. Reson. Mater. Phys. 5 (4) (1997) 275–284.

[9] H.S. Choi, D.R. Haynor, Y. Kim, Multivariate tissue classification of MRI images for 3D volume reconstruction—a statistical approach, Proceedings SPIE Medical Imaging III: Image Processing, Vol. 1092, 1989, pp. 183–193.

[10] M.W. Vannier, R.L. Butterfield, D. Jordan, W.A. Murphy, R.G. Levitt, M. Gado, Multispectral analysis of magnetic-resonance images, Radiology 154 (1) (1985) 221–224.

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

[12] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm (with discussion), J. Roy. Stat. Soc. Ser. B 39 (1) (1977) 1–38.

[13] J.E. Besag, On the statistical analysis of dirty pictures (with discussion), J. Roy. Stat. Soc. Ser. B 48 (3) (1986) 259–302.

[14] G. Celeux, F. Forbes, N. Peyrard, EM procedures using mean field-like approximations for Markov model-based image segmentation, Pattern Recognition 36 (1) (2003) 131–144.

[15] J. Zhang, J.W. Modestino, D.A. Langan, Maximum-likelihood parameter-estimation for unsupervised stochastic model-based image segmentation, IEEE Trans. Image Process. 3 (4) (1994) 404–420.

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

[17] A.W. Moore, Very fast EM-based mixture model clustering using multiresolution *k*d-tree, in: M.S. Kearns, S.A. Solla, D.A. Cohn (Eds.), Advances in Neural Information Processing Systems, Vol. 11, MIT Press, Cambridge, MA, 1999, pp. 543–549.

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

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

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

[21] B. Thiesson, C. Meek, D. Heckerman, Accelerating EM for large databases, Mach. Learning 45 (3) (2001) 279–299.

[22] S.K. Ng, G.J. McLachlan, On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures, Stat. Comput. 13 (1) (2003) 45–55.

[23] C.E. Priebe, D.J. Marchette, Adaptive mixture density estimation, Pattern Recognition 26 (5) (1993) 771–785.

[24] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient *k*-means clustering algorithm: analysis and implementation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 881–892.

[25] D. Pelleg, A.W. Moore, Accelerating exact *k*-means algorithms with geometric reasoning, in: S. Chaudhuri, D. Madigan (Eds.), Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 1999, pp. 277–281.

[26] A. McCallum, K. Nigam, L.H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data

Mining, Association for Computing Machinery, New York, 2000, pp. 169–178.
[27] Z. Liang, J.R. MacFall, D.P. Harrington, Parameter estimation and tissue segmentation from multispectral MR images, IEEE Trans. Med. Imaging 13 (3) (1994) 441–449.
[28] S.J. Nowlan, Soft competitive adaptation: neural network learning algorithms based on fitting statistical mixtures, Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.
[29] S.C. Kochar, R. Korwar, On random sampling without replacement from a finite population, Ann. Inst. Statist. Math. 53 (3) (2001) 631–646.
[30] R.R. Mettu, C.G. Plaxton, Optimal time bounds for approximate clustering, in: A. Darwiche, N. Friedman (Eds.), Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, 2002, pp. 344–351.
[31] P. Sand, A.W. Moore, Repairing faulty mixture models using density estimation, in: C.E. Brodley, A.P. Danyluk (Eds.), Proceedings of the 18th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 2001, pp. 457–464.
[32] L.O. Jimenez, D.A. Landgrebe, Hyperspectral data analysis and supervised feature reduction via projection pursuit, IEEE Trans. Geosci. Remote 37 (6) (1999) 2653–2667.
[33] S. Dasgupta, Experiments with random projection, in: C. Boutillier, M. Goldszmidt (Eds.), Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, 2000, pp. 143–151.

About the Author—SHU-KAY NG received the B.Sc. degree in civil engineering from the University of Hong Kong, Hong Kong, in 1986. He received the Ph.D. degree in Statistics from the Department of Mathematics, University of Queensland, Brisbane, Australia, in 1999. He is currently an Australian Research Council (ARC) Postdoctoral Research Fellow at the University of Queensland. His research interests include machine learning, neural networks, pattern recognition, statistical inference, and survival analysis.

About the Author—GEOFFREY JOHN MCLACHLAN received the B.Sc. (Hons.) and Ph.D. degrees from the University of Queensland in 1969 and 1973, respectively. In 1994, he was awarded a D.Sc. degree by the University of Queensland on the basis of his publications in the scientific literature. He is a professor in the Department of Mathematics and has a joint appointment with the Institute for Molecular Bioscience at the University of Queensland. Professor McLachlan is a fellow of the American Statistical Association, the Royal Statistical Society, and the Australian Mathematical Society. His research interests have been concentrated in the related fields of classification, cluster and discriminant analyses, image analysis, machine learning, neural networks, pattern recognition, and data mining, and in the field of statistical inference. More recently, he has become actively involved in the field of bioinformatics. In these fields, he has published over 145 research articles, including four monographs. The last three monographs, which are volumes in the Wiley Series in Probability and Statistics, are on the topics of discriminant analysis, the EM algorithm, and finite mixture models.