Accepted Manuscript

Serial and parallel implementations of model-based clustering *via* parsimonious Gaussian mixture models

P.D. McNicholas, T.B. Murphy, A.F. McDaid, D. Frost

PII: S0167-9473(09)00063-2
DOI: 10.1016/j.csda.2009.02.011
Reference: COMSTA 4320

To appear in: *Computational Statistics and Data Analysis*

Please cite this article as: McNicholas, P.D., Murphy, T.B., McDaid, A.F., Frost, D., Serial and parallel implementations of model-based clustering *via* parsimonious Gaussian mixture models. *Computational Statistics and Data Analysis* (2009), doi:10.1016/j.csda.2009.02.011

Serial and Parallel Implementations of Model-Based Clustering *via* Parsimonious Gaussian Mixture Models

P.D. McNicholas [a,*], T.B. Murphy [b], A.F. McDaid [c], D. Frost [d]

[a] *Department of Mathematics and Statistics, University of Guelph, ON, Canada.*

[b] *School of Mathematical Sciences, University College Dublin, Ireland.*

[c] *School of Mathematics, Trinity College Dublin, Ireland.*

[d] *Trinity Centre for High Performance Computing, Trinity College Dublin, Ireland.*

Abstract

Model-based clustering using a family of Gaussian mixture models, with parsimonious factor analysis-like covariance structure, is described and an efficient algorithm for its implementation is presented. This algorithm uses the alternating expectation-conditional maximization (AECM) variant of the expectation-maximization (EM) algorithm. Two central issues around the implementation of this family of models, namely model selection and convergence criteria, are discussed. These central issues also have implications for other model-based clustering techniques and for the implementation of techniques like the EM algorithm, in general. The Bayesian information criterion (BIC) is used for model selection and Aitken's acceleration, which is shown to outperform the lack of progress criterion, is used to determine convergence. A brief introduction to parallel computing is then given before the implementation of this algorithm in parallel is facilitated within the master-slave paradigm. A simulation study is then carried out to confirm the effectiveness of this parallelization. The resulting software is applied to two data sets to demonstrate its effectiveness when compared to existing software.

Key words: AECM algorithm; Aitken's acceleration; EM algorithm; factor analysis; Gaussian mixture models; model-based clustering; MPI; parallel programming; parsimonious Gaussian mixture models; PGMMs.

* Corresponding author. Paul D. McNicholas, Department of Mathematics and Statistics, University of Guelph, Guelph, Ontario, Canada, N1G 2W1. Tel: +1 519 8244120, ext. 53136. Fax: +1 519 8370221.

Email addresses: pmcnicho@uoguelph.ca (P.D. McNicholas), brendan.murphy@ucd.ie (T.B. Murphy), dfrost@tchpc.tcd.ie (D. Frost).

1 Introduction

Statistical learning can be either supervised or unsupervised, depending on whether the outcome variable is present, or even known. In an unsupervised learning context, the outcome variable is either absent or non-existent. Clustering is a form of unsupervised learning where the outcome variable is categorical. Model-based clustering is a clustering methodology whereby the group memberships are learned by imposing some assumed mixture modeling structure on data and then estimating the parameters, usually using some variant of the expectation-maximization (EM) algorithm (Dempster et al. 1977). Most commonly, the Gaussian mixture model has been used for model-based clustering (Titterington et al. 1985, Ghahramani & Hinton 1997, Tipping & Bishop 1999, Fraley & Raftery 2002, McLachlan et al. 2003, McNicholas & Murphy 2008).

The density of the Gaussian mixture model is of the form

$$g(\mathbf{x}) = \sum_{g=1}^{G} \pi_g f(\mathbf{x} \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g), \tag{1}$$

where

$$f(\mathbf{x} \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g) = \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_g|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_g)'\boldsymbol{\Sigma}_g^{-1}(\mathbf{x}-\boldsymbol{\mu}_g)\right\},$$

$\boldsymbol{\mu}_g$ is the group mean, $\boldsymbol{\Sigma}_g$ is the group covariance matrix and π_g is the probability of membership of group g.

In Section 2, we review the MCLUST model-based clustering framework which is based on Gaussian mixture models with constrained eigen-decomposed covariance structure. We also review an extension of the MCLUST method that incorporates variable selection. In Section 3, we review the parsimonious Gaussian mixture model (PGMM) approach to model-based clustering, which is based on Gaussian mixture models with parsimonious factor analysis-like covariance structures; the implementation of this family of models is the primary focus of this paper.

Aspects of model fitting in the PGMM paradigm are discussed in Section 4 with particular emphasis on the alternating expectation-conditional maximization (AECM) algorithm (Section 4.2), convergence assessment (Section 4.3) and model selection (Section 4.4). In Section 5, parallel implementation of the PGMM approach, within the master-slave paradigm, is described. The resulting software is then applied to simulated data to confirm that the speed-up gained by using this parallel implementation is linear up to a point.

The PGMM approach is demonstrated on two datasets in Section 6 and the method is shown to give clustering performance that is competitive with popular model-based clustering methods. We conclude, in Section 7, with a summary of this paper.

2 MCLUST and Variable Selection

Banfield & Raftery (1993), Celeux & Govaert (1995) and Fraley & Raftery (1998, 2002) used an eigenvalue decomposition of the $\boldsymbol{\Sigma}_g$ to give rise to the MCLUST family of mixture models. The eigenvalue decomposition in question is of the form

$$\boldsymbol{\Sigma}_g = \lambda_g \mathbf{D}_g \mathbf{A}_g \mathbf{D}_g', \tag{2}$$

where λ_g is a constant, $\mathbf{D}_g$ is a matrix consisting of the eigenvectors of $\boldsymbol{\Sigma}_g$ and $\mathbf{A}_g$ is a diagonal matrix with entries proportional to the eigenvalues of $\boldsymbol{\Sigma}_g$. This model-based clustering technique is supported by the `mclust` package (Fraley & Raftery 2003) for the statistical software R (R Development Core Team 2008).

Raftery & Dean (2006) developed a variable selection technique based on the MCLUST family of models. Variables are selected in a step-wise manner and models are compared using Bayes factors (Kass & Raftery 1995). The variable selection technique is supported by the `clustvarsel` package (Dean & Raftery 2006) for R, which involves repeated application of `mclust`.

Computationally, it should be noted that while the `mclust` and `clustvarsel` software packages are very efficient implementations, the members of the MCLUST family with non-diagonal covariance structure have $\mathcal{O}(p^2)$ covariance parameters. Therefore, these model-based clustering techniques are somewhat limited, by their nature, in applications involving high-dimensional data.

3 Parsimonious Gaussian Mixture Models

Factor analysis (Spearman 1904) is a data reduction technique in which a p-dimensional real-valued data vector $\mathbf{x}$ is modelled using a q-dimensional vector of latent variables $\mathbf{u}$, where $q \ll p$. The $\mathbf{u}$ are often called factors and they can be considered unobservable. The factor analysis model can be written $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{u} + \boldsymbol{\epsilon}$, where the factor loadings are given by the $p \times q$ matrix $\boldsymbol{\Lambda}$, the latent variables $\mathbf{u} \sim N(\mathbf{0}, \mathbf{I}_q)$ and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{\Psi})$, where $\boldsymbol{\Psi} = \text{diag}(\psi_1, \psi_2, \ldots, \psi_p)$. Therefore, the marginal distribution of $\mathbf{x}$ is multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Lambda}\boldsymbol{\Lambda}' + \boldsymbol{\Psi}$.

Recall Equation 1 and impose the factor analysis covariance structure on the group covariance matrix, so that $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g' + \boldsymbol{\Psi}_g$. Now, the $\boldsymbol{\Lambda}_g$ and $\boldsymbol{\Psi}_g$ matrices could be constrained to be equal across groups and the isotropic constraint $\boldsymbol{\Psi}_g = \psi_g\mathbf{I}_p$, could also be imposed. These constraints lead to the eight PGMMs that were introduced by McNicholas & Murphy (2005, 2008). The covariance structure of these eight mixture models, their nomenclature and number of covariance parameters are given in Table 1.

Table 1
The covariance structure, nomenclature and number of covariance parameters for each PGMM.

Model ID	$\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$	$\boldsymbol{\Psi}_g = \boldsymbol{\Psi}$	$\boldsymbol{\Psi}_g = \psi_g\mathbf{I}_p$	Covariance Parameters
CCC	Constrained	Constrained	Constrained	$[pq - q(q-1)/2] + 1$
CCU	Constrained	Constrained	Unconst.	$[pq - q(q-1)/2] + p$
CUC	Constrained	Unconst.	Constrained	$[pq - q(q-1)/2] + G$
CUU	Constrained	Unconst.	Unconst.	$[pq - q(q-1)/2] + Gp$
UCC	Unconst.	Constrained	Constrained	$G[pq - q(q-1)/2] + 1$
UCU	Unconst.	Constrained	Unconst.	$G[pq - q(q-1)/2] + p$
UUC	Unconst.	Unconst.	Constrained	$G[pq - q(q-1)/2] + G$
UUU	Unconst.	Unconst.	Unconst.	$G[pq - q(q-1)/2] + Gp$

From Table 1, a key computational fact becomes apparent: the number covariance parameters for each model is $\mathcal{O}(p)$. Therefore, this family of models has greater potential for application to high-dimensional data than either the MCLUST or variable selection techniques.

Note that the UCU and UUU models are also known as the mixtures of factor analyzers model (Ghahramani & Hinton 1997, McLachlan et al. 2003) and the UUC model is known as the mixtures of probabilistic principal component analyzers model (Tipping & Bishop 1999).

4 Model Fitting

4.1 The EM Algorithm

The EM algorithm provides an iterative method of finding maximum likelihood estimates where data is incomplete or some data is missing. Importantly, data does not actually need to be incomplete and framing problems

as incomplete-data problems often leads to efficient solutions using the EM algorithm.

In the E-step, the expected value of the complete-data log-likelihood is computed based on the current estimates of the model parameters and the complete-data vector, which is the vector of observed data plus missing data. In the M-step, this expected value is maximized with respect to the model parameters.

These two steps are repeated iteratively until convergence is reached. There are a variety of ways to measure convergence: one common approach is to take convergence as the point at which the difference in successive estimates of the log-likelihood is sufficiently small. This, however, is not a convergence criterion but rather an indication of lack of progress: this is discussed further in Section 4.3 and an actual convergence criteria is presented. A comprehensive overview of the EM algorithm and variants is given by McLachlan & Krishnan (2008).

4.2 The AECM Algorithm

The maximum likelihood estimates of the parameters for each PGMM are found using an alternating expectation-conditional maximization (AECM) algorithm (Meng & van Dyk 1997). The expectation-conditional maximization (ECM) algorithm (Meng & Rubin 1993) replaces the M-step by a series of conditional maximization steps and the AECM algorithm allows a different specification of complete-data for each conditional maximization step.

In the context of the PGMMs, at the first stage of the AECM algorithm, when estimating π_g and $\boldsymbol{\mu}_g$, the unobserved group membership labels $\mathbf{z}$ are taken as the missing data. Note that $\mathbf{z}$ is defined so that $z_{ig} = 1$ if observation i belongs to group g and $z_{ig} = 0$ otherwise. At the second stage of the AECM algorithm, when estimating $\boldsymbol{\Lambda}_g$ and $\boldsymbol{\Psi}_g$, the group membership labels $\mathbf{z}$ and the latent factors $\mathbf{u}$ are taken as the missing data. McLachlan & Peel (2000*b*) give extensive details of fitting the AECM algorithm in the case where no constraints are imposed.

Using an AECM algorithm, the estimates $\hat{\mathbf{z}}$, $\hat{\boldsymbol{\mu}}_g$ and $\hat{\pi}_g$ are the same for each member of the PGMM family:

$$\hat{z}_{ig} = \frac{\hat{\pi}_g f(\mathbf{x}_i \mid \hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Lambda}}_g, \hat{\boldsymbol{\Psi}}_g)}{\sum_{g'=1}^{G} \hat{\pi}_{g'} f(\mathbf{x}_i \mid \hat{\boldsymbol{\mu}}_{g'}, \hat{\boldsymbol{\Lambda}}_{g'}, \hat{\boldsymbol{\Psi}}_{g'})}, \quad \hat{\boldsymbol{\mu}}_g = \frac{\sum_{i=1}^{n} \hat{z}_{ig} \mathbf{x}_i}{n_g} \quad \text{and} \quad \hat{\pi}_g = \frac{n_g}{n},$$

where $n_g = \sum_{i=1}^{n} \hat{z}_{ig}$. The estimates for $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ in the CCC case are given herein and details of the parameter estimates for each member of the PGMM family are given by McNicholas & Murphy (2008).

Let α be some variable, then write $\hat{\alpha}$ to denote the most recent estimate of α and $\hat{\alpha}^{\text{new}}$ to denote the new estimate of α. Using this notation, the maximum likelihood estimates for the CCC model are,

$$\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\Lambda}}'(\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Lambda}}' + \hat{\psi}\mathbf{I}_p)^{-1}, \quad \hat{\boldsymbol{\Lambda}}^{\text{new}} = \tilde{\mathbf{S}}\hat{\boldsymbol{\beta}}'\tilde{\boldsymbol{\Theta}}^{-1} \text{ and } \quad \hat{\psi}^{\text{new}} = \frac{1}{p}\operatorname{tr}\left\{\tilde{\mathbf{S}}' - \hat{\boldsymbol{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}\tilde{\mathbf{S}}\right\},$$

where $\hat{\psi}$ is a real number such that $\hat{\boldsymbol{\Psi}} = \hat{\psi}\mathbf{I}_p$, $\tilde{\boldsymbol{\Theta}} = \mathbf{I}_q - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\Lambda}} + \hat{\boldsymbol{\beta}}\tilde{\mathbf{S}}\hat{\boldsymbol{\beta}}'$ is a symmetric $q \times q$ matrix and $\tilde{\mathbf{S}} = \sum_{g=1}^{G}\hat{\pi}_g\mathbf{S}_g$, where $\mathbf{S}_g = (1/n_g)\sum_{i=1}^{n}\hat{z}_{ig}(\mathbf{x}_i - \boldsymbol{\mu}_g)(\mathbf{x}_i - \boldsymbol{\mu}_g)'$.

Now it is possible to outline the AECM algorithm in the CCC case:

initialize $\hat{\mathbf{z}}$ randomly
initialize $n_g, \hat{\pi}_g, \hat{\mu}_g, \tilde{\mathbf{S}}$ as per updates
run Householder's algorithm with QL reduction on $\tilde{\mathbf{S}}$ to get $\tilde{\mathbf{S}} = \mathbf{PDP}^{-1}$
initialize $\hat{\boldsymbol{\Lambda}} = \mathbf{dP}$, where $\mathbf{d}$ is the element-wise square root of $\operatorname{diag}\{\mathbf{D}\}$
initialize $\hat{\psi} = \frac{1}{p}\operatorname{tr}\{\tilde{\mathbf{S}} - \hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Lambda}}'\}$
while not converged:
    CM-step 1 { update $\hat{\pi}_g$; update $\hat{\mu}_g$
    E-step { **if** not first iteration: update $\hat{\mathbf{z}}$ **end if**
    update $\tilde{\mathbf{S}}, \hat{\boldsymbol{\Lambda}}, \hat{\psi}, \hat{\boldsymbol{\beta}}, \tilde{\boldsymbol{\Theta}}$
    CM-step 2 { update $\hat{\boldsymbol{\Lambda}}^{\text{new}}$; update $\hat{\psi}^{\text{new}}$
    E-step { update $\hat{\mathbf{z}}$
    compute log-likelihood
    check for convergence
    set $\hat{\boldsymbol{\Lambda}} = \hat{\boldsymbol{\Lambda}}^{\text{new}}$ and $\hat{\psi} = \hat{\psi}^{\text{new}}$
end while

Although $\hat{\mathbf{z}}$ is initialized randomly in this CCC case, the $\hat{\mathbf{z}}$ that are output here are used as the starting values for the other members of the PGMM family. Of course, other initialization tactics could be used. For example, each of the eight models could be run from a number of different starting values of $\hat{\mathbf{z}}$, for a limited number of iterations. Then the starting values of $\hat{\mathbf{z}}$ that led to the best model (see Section 4.4) from amongst these quick runs could be used for

the actual analysis.

Note that the elements of $\text{diag}\{\mathbf{D}\}$ are in order of size; that is, the first element is the largest eigenvalue, the second element is the second largest eigenvalue and so on. Details of Householder's algorithm with QL reduction are given by Press et al. (1992).

4.3 Convergence Criteria

4.3.1 Aitken's Acceleration

The Aitken's acceleration procedure was used to determine convergence of each AECM algorithm. Specifically, it was used to estimate the asymptotic maximum of the log-likelihood at each iteration and then to make a decision about whether the AECM algorithm had converged or not.

The Aitken's acceleration at iteration k is given by

$$a^{(k)} = \frac{l^{(k+1)} - l^{(k)}}{l^{(k)} - l^{(k-1)}},$$

where $l^{(k+1)}$, $l^{(k)}$ and $l^{(k-1)}$ are the log-likelihood values from iterations $k+1$, k and $k-1$, respectively. Then the asymptotic estimate of the log-likelihood at iteration $k+1$ is given by

$$l_\infty^{(k+1)} = l^{(k)} + \frac{1}{1 - a^{(k)}}(l^{(k+1)} - l^{(k)})$$

(Böhning et al. 1994). This is the estimated value of the log-likelihood, based on the last three iterations, that the algorithm will converge to asymptotically. Lindsay (1995) proposes that the algorithm can be stopped when

$$l_\infty^{(k+1)} - l^{(k+1)} < \epsilon, \tag{3}$$

where ϵ is small. We propose a very similar convergence criterion: that the algorithm can be stopped when

$$l_\infty^{(k+1)} - l^{(k)} < \epsilon, \tag{4}$$

where ϵ is small. The criterion in Equation 4 is necessarily no less strict than that in Equation 3, since $l^{(k+1)} \geq l^{(k)}$.

4.3.2 Lack of Progress

Some model-based clustering algorithms, such as that described by Fraley & Raftery (1998), use the difference in successive log-likelihoods as a convergence

criterion. That is, the algorithm is considered to have converged when

$$l^{(k+1)} - l^{(k)} < \epsilon, \tag{5}$$

where ϵ is small. The condition in Equation 5 is not a convergence criterion but rather an indication of lack of progress (Lindstrom & Bates 1988).

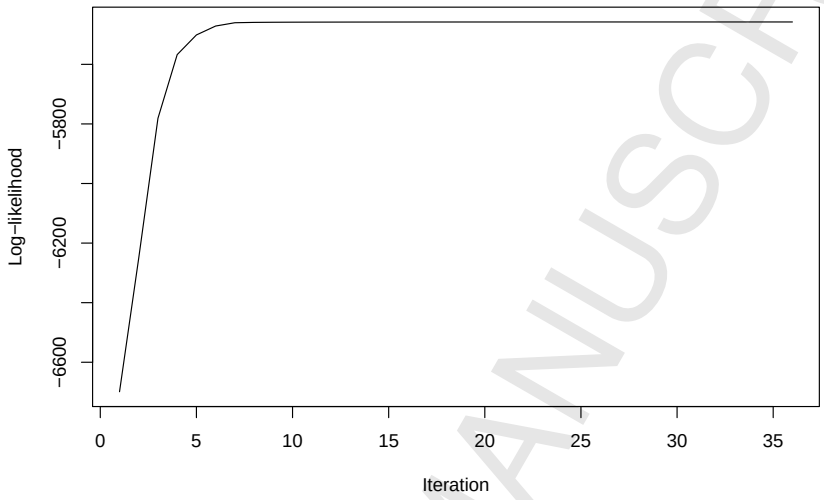

Fig. 1. A classically-shaped plot of iteration number versus log-likelihood for an AECM algorithm.

Figure 1 shows a classically-shaped plot of iteration number versus log-likelihood and in this case the criteria in equations 4 and 5 will give very similar results. However, figures 2 and 3 illustrate situations where the difference between a lack of progress criterion and a genuine convergence criterion can become very apparent. In both of these cases, a lack of progress criterion might greatly underestimate the correct value of the log-likelihood.

Note that figures 1, 2 and 3 all arise from real analyzes using the AECM algorithm detailed herein. Furthermore, the convergence criteria in Equation 4 is necessarily at least as strict as the lack of progress criteria given in Equation 5. This follows from the fact that $0 \leq a^{(k)} < 1$ in a neighborhood of the maximum value and so $1/(1 - a^{(k)}) \geq 1$. Hence, $l_{\infty}^{(k+1)} - l^{(k)} \geq l^{(k+1)} - l^{(k)}$ in a neighborhood of the maximum.

4.4 Model Selection

For the MCLUST and PGMM techniques, the Bayesian information criterion (BIC, Schwartz 1978) is used to select the 'best' member of the family for given data. The BIC can also be used to compare PGMM and MCLUST models.

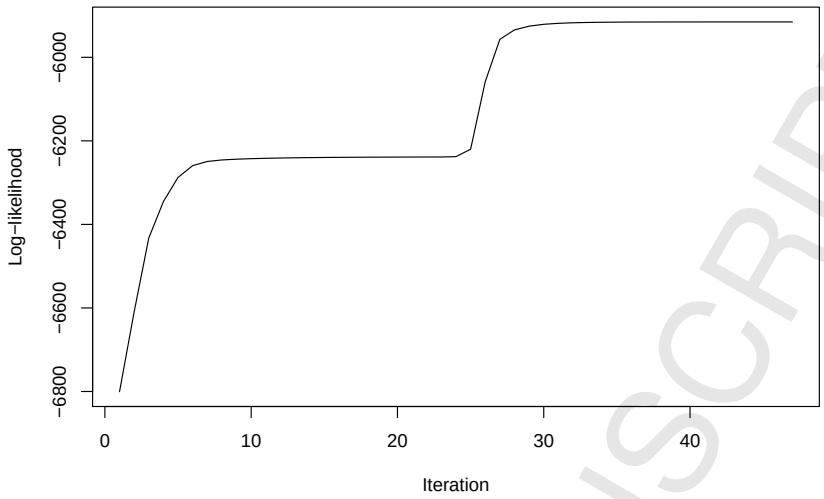

Fig. 2. A plot of iteration number versus log-likelihood for an AECM algorithm, illustrating a single step. A lack of progress criterion could stop the algorithm at the first step, depending on the value of ϵ used.

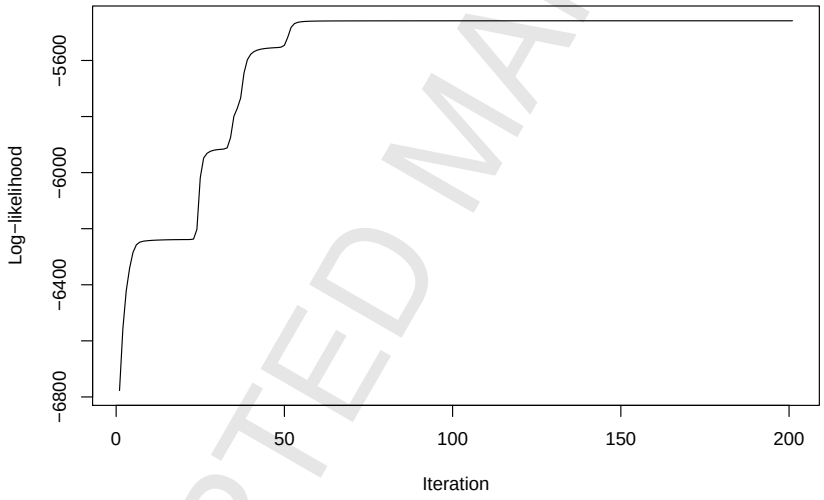

Fig. 3. A plot of iteration number versus log-likelihood for an AECM algorithm, illustrating multiple steps.

For a model with parameters $\mathbf{\Phi}$, the BIC is given by

$$\text{BIC} = 2l(\mathbf{x}, \hat{\mathbf{\Phi}}) - \nu \log n,$$

where $l(\mathbf{x}, \hat{\mathbf{\Phi}})$ is the maximized log-likelihood, $\hat{\mathbf{\Phi}}$ is the maximum likelihood estimate of $\mathbf{\Phi}$, ν is the number of free parameters in the model and n is the number of observations.

Leroux (1992) shows that the BIC does not underestimate the number of components in a mixture model and Keribin (2000) shows that BIC consistently estimates the number of mixture components under certain regularity conditions. Lopes & West (2004) provide a simulation study where they show

that the BIC has excellent performance in selecting the number of factors in a factor analysis model, even when the sample size is small.

5 Parallel Implementation

5.1 Parallel Computation

Parallel computation describes the use of multiple processors to execute a number of tasks simultaneously, or in parallel. Parallel computation is very attractive when all of these tasks, run simultaneously, take much less time to execute than if the tasks were run in serial. Broadly, there are two types of parallelization: coarse-grain parallelization and fine-grain parallelization. Fine-grain parallelization involves frequent inter-processor communication and is often dependant on the characteristics of the interconnect between the processors for good performance. Coarse-grain parallelization involves a small amount of inter-processor communication and so performance will not generally be affected by the quality of the interconnect between processors.

The message passing interface (MPI) is used to enable the parallel communication between the processes. MPI is a specification for communications between processes in a compute cluster. It is programming language independent and there are versions available for use with C, C++, Fortran, Python and many other programming languages. Optimized implementations of MPI are also available for different interconnects such as Infiniband, Myrinet and on the IBM BlueGene systems. Further details on MPI can be found at `www.mpi-forum.org` and `www.open-mpi.org`, and a comprehensive guide to MPI is given by Gropp et al. (1999).

A computation that can be split into completely independent tasks that have no data or time dependency between them is often called 'trivially parallelizable'. Ray tracing in computer graphics, signal processing, brute force attacks in cryptography and gene sequence alignment are all examples of problems that are trivially parallelizable. The most simple way to program these problems is using the master-slave model. In this system one process (the master) divides the work to be done into discrete work packages and distributes them to all of the other processes (the slaves). Once a slave has finished the required calculation it sends the results back to the master and requests another work package. The master then collates the results from all of the slaves to generate an overall result from the calculation. Projects such as SETI@home and Folding@home use the master-slave model.

Parallel computation has been used in a variety of statistical applications.

These include parallel implementations of algorithms for kernel estimation (Racine 2002), linear models (Kontoghiorghes 2000, Yanev & Kontoghiorghes 2006), partial least squares (Milidiú & Rentera 2005) and regression submodels (Gatu et al. 2007).

5.2 Approach

In order to fully exploit the computational attractiveness of the PGMMs that arises from the linear relationship between the number of covariance parameters and the dimensionality of the data, AECM algorithms like that outlined in Section 4.2 were parallelized.

The problem of finding the best member of the PGMM family can be considered as maximizing with respect to the BIC over the triple $(\mathcal{M}, G, q)$, where $\mathcal{M} \in \{\text{CCC}, \ldots, \text{UUU}\}$ is the model in question, G is the number of groups and q is the number of factors. The nature of the problem makes it trivially parallelizable: that is, each triple $(\mathcal{M}, G, q)$ can be sent to a different processor and processors can work independently of one another. Note that, apart from running the CCC models first so that we get the starting values of $\hat{\mathbf{z}}$ for the other seven models, these triples can be run in any order. Therefore, the slaves can be used as soon as they become free.

The prospect of parallelizing within-triple is not implemented here because any within-triple parallelization may actually cost time since the saving achieved by sending jobs triple-wise to processors may well be so great as to negate any possible advantage of within-triple parallelization. Therefore, a master-slave paradigm was used to facilitate parallelization. This software was originally written in the C language and then parallelized using MPI.

5.3 Implementation

5.3.1 Overview

Due to the strategy adopted for parallelization it was necessary to write two functions, one for the master and one for the slaves. As is normal in master-slave implementations, the only communication is between the master and the various slaves. There is no communication between different slave processes. Outlines of the master and slave functions are given in the following subsections.

5.3.2 Master

The master function can be summarized as follows:

read command line arguments
standardize data
randomly populate $\hat{\mathbf{z}}$
send CCC jobs to available slaves
receive results from slaves
send other jobs to available slaves
receive results from slaves
kill all slaves
print results to files

5.3.3 Slave

The slave function can be summarized as follows:

while 1:
    receive parameter G
    if G=-1:
        break while
    end if
    receive all other parameters
    receive $\hat{\mathbf{z}}$
    run AECM for model and parameters received
    send all results to master
end while

Note that sending $G = -1$ is the method by which the master 'kills' the slave.

5.4 Speed-Up

The speed-up for ρ processors is defined herein as the time taken using ρ processors divided by the time taken using 1 processor. To evaluate the speed-up that had been achieved by parallelization, the wine data that is described in Section 6.2 was analyzed and the parameters associated with the best model were noted. These parameters were then used to generate six datasets, using the R software. As in the case of the original dataset, each of these datasets consisted of 178 cases of 27 variables. These simulated datasets were then analyzed using 2, 3, 5, 9, 17 and 33 processors respectively; that is, $2^n +$

1 processors, or 2^n slaves, for $n = 0, 1, \ldots, 5$. The software was run for a maximum of 8 groups and 8 factors per group.

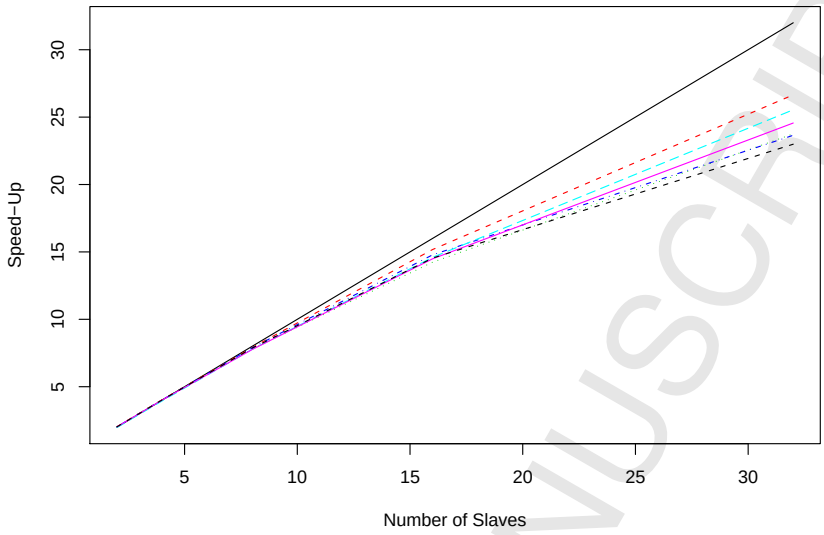

Fig. 4. Speed-up versus number of slaves for the wine data.

The speed-up for each of the six runs is shown in Figure 4. The solid black line on Figure 4 represents the linear case where 'speed-up' = 'number of slaves'. The other lines depict the time taken in the real world, according to the `real` output to `stderr` given by the `time()` function, in each case. Within a master-slave paradigm, the ideal situation occurs when the speed-up is directly proportional to the number of processors — this is known as linear speed-up. Figure 4 shows roughly linear speed-up as far as about 16 slaves.

The departure from linear speedup at larger numbers of slaves is caused by contention at the master process. Since the dataset used in this example is relatively small, the time taken by a slave to carry out the required calculations is short. With larger numbers of slaves, the master becomes overloaded with results and is not able to handle the requests in a timely manner. With a larger data set, the time taken for each work package, or triple $(\mathcal{M}, G, q)$, to be completed will be longer, reducing the percentage of time each slave spends communicating with the master, thereby reducing congestion on the master process. The congestion at the master is also greatest at the start of execution. Each slave will take close to the same length of time to complete its work package and therefore all slaves will try to communicate back to the master at approximately the same time.

The ideal number of slave processes to use depends on a combination of various factors including CPU performance, interconnect bandwidth and latency, dataset size, memory usage and filesystem performance. In all master-slave implementations, the goal is to find a balance between keeping the slaves busy and not overloading the master. The only way to determine the optimum

number of processors to use is through experimentation.

5.5 Discussion

The parallel implementation described in this section demonstrates how well-suited the PGMM family of models is to parallelization. Although MPI has been previously used for parallel clustering (see Pizzuti & Talia 2003), the implementation herein is the first to illustrate the relative simplicity of parallel implementation of model-based clustering techniques. In fact, MCLUST could be parallelized in a very similar fashion. The parallelization of the AutoClass technique (Cheeseman & Stutz 1996) that was carried out by Pizzuti & Talia (2003) was very efficient on the hardware that they used. However, this technique requires more inter-processor communication than the parallelization of the PGMM family that is outlined herein. Therefore, its performance is more dependent on the quality of interconnects in the hardware on which the software is run. Furthermore, the master-slave paradigm that was used to facilitate the PGMM parallelization ensures that slaves can be used as soon as they become free and virtually eliminates the problem of idle processors.

6 Examples

Two examples of the PGMM approach to model-based clustering are given. The first, an analysis of biological measurements taken on leptograpus crabs, is used to facilitate comparison of the PGMMs with MCLUST and variable selection. The second, an analysis of physical and chemical properties of wine, is used to illustrate the effectiveness of the parallel software when applied to a higher dimensional data set.

6.1 Leptograpus Crabs Data

6.1.1 The Data

The data consist of biological measurements on 200 leptograpus crabs collected in Fremantle, Western Australia; 50 male and 50 female, for each of two species; 100 orange and 100 blue. The data were sourced from the `MASS` library for R; which contains datasets from Venables & Ripley (2002). There are five variables in these data, corresponding to the five measurements that were taken on each crab; these measurements are given in Table 2.

Table 2
Details of the five measurements that were taken on the leptograpsus crabs.

Variable	**Measurement**
FL	Frontal lobe size in millimeters.
RW	Rear width in millimeters.
CL	Carapace length in millimeters.
CW	Carapace width in millimeters.
BD	Body depth in millimeters.

These data first appeared in Campbell & Mahon (1974) and were subsequently analyzed by Ripley (1996), McLachlan & Peel (1998, 2000*a*) and Raftery & Dean (2006). These data were selected for analysis here because the results will be directly comparable with the MCLUST and variable selection analyses of Raftery & Dean (2006).

6.1.2 The PGMM Family of Models

The PGMM family was fitted to the data for $G \in \{1, 2, \ldots, 5\}$ and $q \in \{1, 2, \ldots, 5\}$ by running the software from three random starting values, so that a total of 600 models were fitted. Figure 5 shows the BIC for the best PGMM for each (G, q) over the eight models and all three random starts. The BIC values for the best three models are given in Table 3.

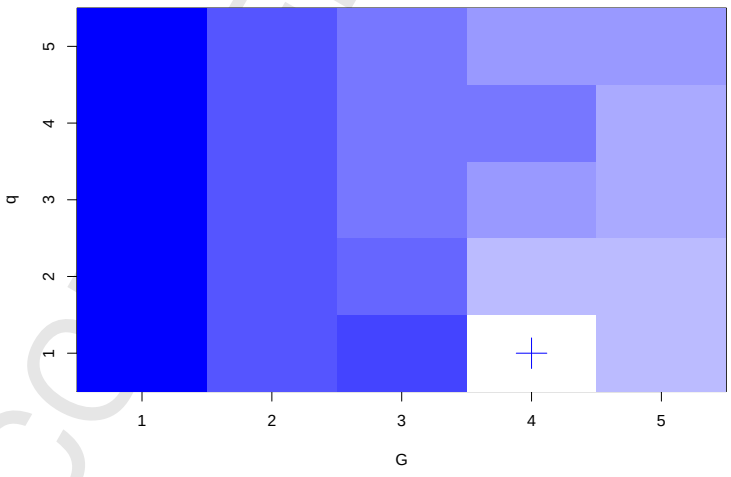

Fig. 5. A heat map giving the greatest BIC values for each PGMM at (G, q) for the crabs data.

The best model was a UCU model with $G = 4$ and $q = 1$. The BIC for this model was 197.87 and a classification table for this best PGMM is given in Table 4.

Table 3
The best three models, by BIC, for the PGMM family applied to the crabs data.

Model	Number of Groups (G)	Number of Factors (q)	BIC
UCU	4	1	197.87
CCU	5	1	110.89
CCU	5	2	110.88

Table 4
Classification table for the best PGMM for the crabs data.

		1	2	3	4
Blue	Male	39	11		
	Female		50		
Orange	Male			50	
	Female			4	46

Looking at a pairs plot of the variables in the crabs data, given in Figure 6, it becomes apparent that a linear relationship exists between each of the five variables. Therefore, it is somewhat natural that the best PGMM had just one latent variable.

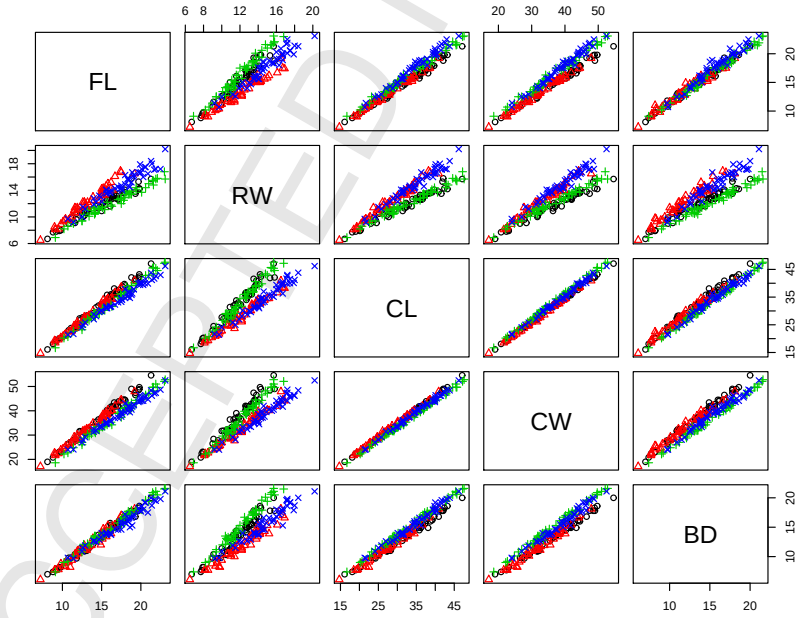

Fig. 6. A pairs plot of the crabs data, produced using R.

6.1.3 MCLUST & Variable Selection

Raftery & Dean (2006) report the results of applying MCLUST to the crabs

data. Although they do not state a BIC value, they do report a classification table, which is given herein as Table 5. Raftery & Dean (2006) also used vari-

Table 5
Classification table from the MCLUST analysis of the crabs data.

		1	2	3	4	5	6	7
Blue	Male	32					18	
	Female		31				19	
Orange	Male			28				22
	Female				24	21		5

able selection to analyze these data and the classification table that resulted from this analysis is given in Table 6.

Table 6
Classification table from the variable selection analysis of the crabs data.

		1	2	3	4
Blue	Male	40	10		
	Female		50		
Orange	Male			50	
	Female			5	45

6.1.4 Model Comparison

Table 7 gives the Rand index (Rand 1971), adjusted Rand index (Hubert & Arabie 1985) and the error rate for the three models that were applied to the crabs data. The variable selection and PGMM techniques were the best for

Table 7
Rand and adjusted Rand indices and error rates for all of the models that were applied to the crabs data.

Model	Rand Index	Adjusted Rand Index	Error Rate
PGMM	0.932	0.817	0.075
MCLUST	0.851	0.533	0.425
Variable Selection	0.931	0.815	0.075

the crabs data; with the best PGMM having very slightly greater Rand The error rates, however, were identical.

6.2 Wine Data

6.2.1 The Data

Forina et al. (1986) recorded twenty-eight chemical and physical properties of three types of wine (Barolo, Grignolino, Barbera) from Italy. We report results on the analysis of twenty-seven of the variables from the original study (Table 8); the sulphur measurements from the original data were not available.

Table 8
The twenty-seven chemical and physical properties of Italian wines used in this study.

Chemical and Physical Properties		
Alcohol	Sugar-free extract	Fixed acidity
Tartaric acid	Malic acid	Uronic acids
pH	Ash	Alcalinity of ash
Potassium	Calcium	Magnesium
Phosphate	Chloride	Total phenols
Flavonoids	Nonflavonoid phenols	Proanthocyanins
Color Intensity	Hue	OD280/OD315 of diluted wines
OD280/OD315 of flavonoids	Glycerol	2-3-butanediol
Total nitrogen	Proline	Methanol

6.2.2 The PGMM Family of Models

The eight PGMMs were fitted to the data for $G \in \{1, 2, \ldots, 6\}$ and $q \in \{1, 2, \ldots, 6\}$ and three random starts were used for each model. Hence, a total of 864 different models were fitted to the data. The model with the highest BIC was selected; this was the CUU model with $G = 3$ and $q = 4$. Figure 7 shows the maximum BIC for the eight parsimonious models for each pair (G, q) and details of the top three models are given in Table 9. The application of the PGMM family to these data was previously reported in McNicholas & Murphy (2008).

The choice of the CUU model can be explained because the loading matrix for each group in this setting has $pq - q(q-1)/2$ free parameters. Hence, the penalty for allowing the loading matrix to be different across groups is very large. A cross tabulation of the classifications from the best PGMM versus the true wine type is given in Table 10: this model correctly classifies all but one of the wines.

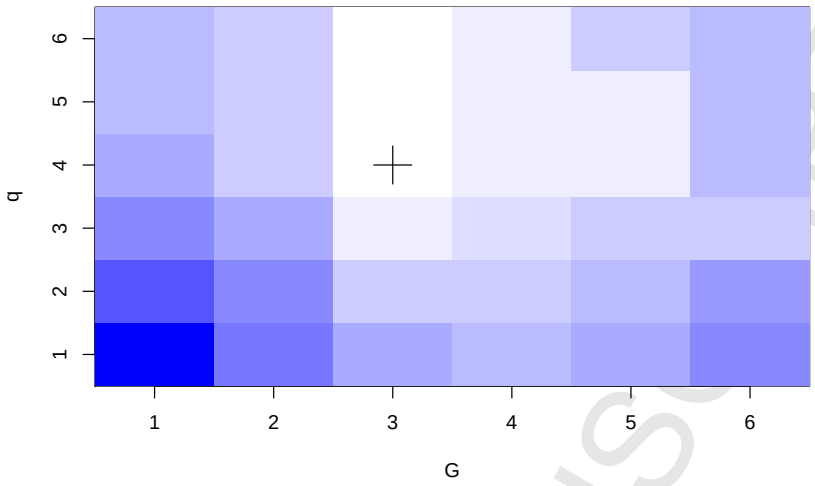

Fig. 7. A heat map representation of the maximum BIC value for each value of (G, q).

Table 9
The best three models, by BIC, for the PGMM family applied to the wine data.

Model	Number of Groups	Number of Factors	BIC
CUU	3	4	-11,454.11
CUU	3	5	-11,457.70
CUU	3	6	-11,503.95

Table 10
A classification table for the best PGMM model for the wine data.

	1	2	3
Barolo	59		
Grignolino		70	1
Barbera			48

6.2.3 MCLUST & Variable Selection

Model-based clustering was also completed on the wine data using `mclust` and `clustvarsel`. A cross tabulation of the classifications for each approach and the wine types is shown in tables 11 and 12.

Table 11
The classification table for the best MCLUST model for the wine data.

	1	2	3
Barolo	58	1	
Grignolino	4	66	1
Barbera			48

Table 12
Classifcation table for the best model found using variable selection on the wine data.

	1	2	3	4
Barolo	52	7		
Grignolino		17	54	
Barbera		1		47

6.2.4 Model Comparison

A comparison of the clustering results on the wine data indicates that all of the methods are very good at discovering the group structure in the wine data. The PGMM, MCLUST and variable selection techniques are all good at separating the wines into types. However, variable selection uses two groups to model the Grignolino wines. The fact that the variable selection technique has not performed as well as the MCLUST family of models here suggests that the subset of variables selected by variable selection is not as good at separating the wines into type as the full range of variables are. Table 13 shows how the methods differ in clustering performance.

Table 13
Rand index, adjusted Rand index and error rate for the PGMM, MCLUST and variable selection techniques on the wine data.

Model	Rand Index	Adjusted Rand	Error Rate
PGMM	0.99	0.98	0.006
MCLUST	0.95	0.90	0.034
Variable Selection	0.91	0.78	0.140

7 Summary

The PGMM approach to model-based clustering has been described and the algorithm for fitting these models has been outlined and demonstrated on real data. The use of Aitken's acceleration for convergence assessment has been demonstrated as being superior to the frequently-used lack of progress criterion.

The AECM algorithm used for parameter estimation was parallelized within the master-slave paradigm using MPI and the resulting speed-up has been shown to be linear up to a certain point. This parallelization allows the efficient application of the PGMM family of models to high-dimensional data, thus fully exploiting the computational attractiveness of the PGMMs that arises

from the linear relationship between the number of covariance parameters and data-dimensionality.

This software was applied to two datasets: one giving biological measurements taken on leptograpus crabs and the other on chemical and physical properties of Italian wines. The crabs data were used to compare the PGMM model-based clustering technique with two popular model-based clustering techniques: MCLUST and variable selection. The PGMMs performed favorably. The wine data was used to demonstrate the effectiveness of the parallel implementation in a higher-dimensional setting. The PGMM family gives superior clustering performance to MCLUST and variable selection when applied to the wine data.

8 Acknowledgements

The authors are happy to acknowledge that some of the parallel computing described herein was carried out using facilities that were funded in part by the Canada Foundation for Innovation, the Ontario Research Fund's Research Infrastructure program, and the Natural Sciences and Engineering Research Council of Canada (NSERC). The equipment was provided by Silicon Graphics Inc.

The IITAC cluster at the Trinity Centre for High Performance Computing was also used to run some of the parallel analyzes reported in this work. Thanks in this regard are due to the IITAC research project, the Higher Education Authority, the National Development Plan and the Trinity Centre for High Performance Computing.

This work was supported by a Science Foundation of Ireland Basic Research Grant (04/BR/M0057) and an NSERC Discovery Grant.

References

Banfield, J. D. & Raftery, A. E. (1993), 'Model-based Gaussian and non-Gaussian clustering', *Biometrics* **49**(3), 803–821.

Böhning, D., Dietz, E., Schaub, R., Schlattmann, P. & Lindsay, B. (1994), 'The distribution of the likelihood ratio for mixtures of densities from the one-parameter exponential family', *Annals of the Institute of Statistical Mathematics* **46**, 373–388.

Campbell, N. A. & Mahon, R. J. (1974), 'A multivariate study of variation in two species of rock crab of genus leptograpsus', *Australian Journal of Zoology* **22**, 417–425.

Celeux, G. & Govaert, G. (1995), 'Gaussian parsimonious clustering models', *Pattern Recognition* **28**, 781–793.

Cheeseman, P. & Stutz, J. (1996), *Bayesian classification (AutoClass): theory and results*, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 153–180.

Dean, N. & Raftery, A. E. (2006), *The clustvarsel Package*. R package version 0.2-4.

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

Forina, M., Armanino, C., Castino, M. & Ubigli, M. (1986), 'Multivariate data analysis as a discriminating method of the origin of wines', *Vitis* **25**, 189–201.

Fraley, C. & Raftery, A. E. (1998), 'How many clusters? Which clustering methods? Answers via model-based cluster analysis', *The Computer Journal* **41**(8), 578–588.

Fraley, C. & Raftery, A. E. (2002), 'Model-based clustering, discriminant analysis, and density estimation', *Journal of the American Statistical Association* **97**, 611–631.

Fraley, C. & Raftery, A. E. (2003), 'Enhanced software for model-based clustering, density estimation, and discriminant analysis: MCLUST', *Journal of Classification* **20**, 263–286.

Gatu, C., Yanev, P. & Kontoghiorghes, E. (2007), 'A graph approach to generate all possible regression submodels', *Computational Statistics and Data Analysis* **52**(2), 799–815.

Ghahramani, Z. & Hinton, G. E. (1997), The EM algorithm for factor analyzers, Technical Report CRG-TR-96-1, University Of Toronto, Toronto.

Gropp, W., Lusk, E. & Skjellum, A. (1999), *Using MPI: Portable Parallel Programming with the Message-passing Interface*, 2nd edn, MIT Press, Cambridge, MA.

Hubert, L. & Arabie, P. (1985), 'Comparing partitions', *Journal of Classification* **2**, 193–218.

Kass, R. E. & Raftery, A. E. (1995), 'Bayes factors', *Journal of the American Statistical Association* **90**, 773–795.

Keribin, C. (2000), 'Consistent estimation of the order of mixture models', *Sankhyā. The Indian Journal of Statistics. Series A* **62**(1), 49–66.

Kontoghiorghes, E. J. (2000), Parallel algorithms for linear models: Numerical methods and estimation problems, *in* 'Advances in Computational Economics', Vol. 15, Kluwer Academic Publishers, Boston, MA.

Leroux, B. G. (1992), 'Consistent estimation of a mixing distribution', *The Annals of Statistics* **20**, 1350–1360.

Lindsay, B. G. (1995), Mixture models: Theory, geometry and applications, *in* 'NSF-CBMS Regional Conference Series in Probability and Statistics', Vol. 5, Hayward, California: Institute of Mathematical Statistics.

Lindstrom, M. J. & Bates, D. M. (1988), 'Newton-Raphson and EM algorithms

for linear mixed-effects models for repeated-measures data', *Journal of the American Statistical Association* **83**(404), 1014–1022.

Lopes, H. F. & West, M. (2004), 'Bayesian model assessment in factor analysis', *Statistica Sinica* **14**, 41–67.

McLachlan, G. J. & Krishnan, T. (2008), *The EM Algorithm and Extensions*, 2nd edn, Wiley, New York.

McLachlan, G. J. & Peel, D. (1998), Robust cluster analysis via mixtures of multivariate t-distributions, *in* 'Lecture Notes in Computer Science', Vol. 1451, Springer-Verlag, Berlin, pp. 658–666.

McLachlan, G. J. & Peel, D. (2000*a*), *Finite Mixture Models*, John Wiley & Sons, New York.

McLachlan, G. J. & Peel, D. (2000*b*), Mixtures of factor analyzers, *in* P. Langley, ed., 'Seventh International Conference on Machine Learning', Morgan Kaufmann, San Francisco, pp. 599–606.

McLachlan, G. J., Peel, D. & Bean, R. W. (2003), 'Modelling high-dimensional data by mixtures of factor analyzers', *Computational Statistics & Data Analysis* **41**(3–4), 379–388.

McNicholas, P. D. & Murphy, T. B. (2005), Parsimonious Gaussian mixture models, Technical Report 05/11, Department of Statistics, Trinity College Dublin.

McNicholas, P. D. & Murphy, T. B. (2008), 'Parsimonious Gaussian mixture models', *Statistics and Computing* **18**(3), 285–296.

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

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

Milidiú, R. L. & Rentera, R. P. (2005), 'Dpls and ppls: two pls algorithms for large data sets', *Computational Statistics and Data Analysis* **48**(1), 125 – 138.

Pizzuti, C. & Talia, D. (2003), 'P-autoclass: Scalable parallel clustering for mining large data sets', *IEEE Transactions on Knowledge and Data Engineering* **15**(3), 629–641.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1992), *Numerical Recipies in C — The Art of Scientific Computation*, 2nd edn, Cambridge University Press.

R Development Core Team (2008), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Racine, J. (2002), 'Parallel distributed kernel estimation', *Computational Statistics and Data Analysis* **40**(2), 293–302.

Raftery, A. E. & Dean, N. (2006), 'Variable selection for model-based clustering', *Journal of the American Statistical Association* **101**(473), 168–178.

Rand, W. M. (1971), 'Objective criteria for the evaluation of clustering methods', *Journal of the American Statistical Association* **66**, 846–850.

Ripley, B. D. (1996), *Pattern Recognition and Neural Networks*, Cambridge

University Press, Cambridge, UK.

Schwartz, G. (1978), 'Estimating the dimension of a model', *Annals of Statistics* **6**, 31–38.

Spearman, C. (1904), 'The proof and measurement of association between two things', *American Journal of Psychology* **15**, 72–101.

Tipping, T. E. & Bishop, C. M. (1999), 'Mixtures of probabilistic principal component analysers', *Neural Computation* **11**(2), 443–482.

Titterington, D. M., Smith, A. F. M. & Makov, U. E. (1985), *Statistical Analysis of Finite Mixture Distributions*, John Wiley & Sons, Chichester.

Venables, W. N. & Ripley, B. D. (2002), *Modern Applied Statistics with S-PLUS*, 4th edn, Springer, New York.

Yanev, P. & Kontoghiorghes, E. J. (2006), 'Efficient algorithms for estimating the general linear model', *Parallel Computing* **32**(2), 195–204.