An exact Newton's method for ML estimation of a Gaussian mixture

Grigory Alexandrovich
Hans-Meerwein-Str., D-35032 Marburg, Germany. [1]

Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany.

We discuss the problem of computing the MLE of the parameters of a multivariate Gaussian mixture. The most widely used method for solving this problem is the EM algorithm. Although this method converges globally under some general assumptions, does not require much storage and is simple to implement, it yields only a (super-) linear convergence rate and may become extremely slow in certain situations.

We introduce an alternative - a hybrid of the EM algorithm and an exact Newton's method, which converges locally quadratic and yields an estimate of the Fisher information matrix. This alternative outperforms pure EM clearly in constellations with a high proportion of missing data. In our context these are mixtures with a high number of components and a large sample size.

We discuss a parameterization of the mixture density, which ensures the adherence of several restrictions on the parameters during the Newton iterations.

For the involved first and second derivatives of a multivariate Gaussian mixture log-likelihood expressions appropriate for implementation are presented[2]. The techniques and results may also be useful for calculations of derivatives of mixtures in other elliptical families.

Furthermore we consider a penalization of the log-likelihood function to avoid convergence towards the boundary of the parameter space, as suggested by Chen and Tan (2009).

Keywords: normal mixtures, maximum likelihood, Newton's method, Fisher information, EM algorithm.

Mathematics Subject classification: 62-07, 62F10, 62H12

1 Introduction

Finite normal (or Gaussian) mixtures are widely used in statistical data analysis due to their high flexibility and the fact that they are well studied, see e.g. Fraley and Raftery (2002), McLachlan and Bashford (1988), Titterington et al (1985). Important areas of application for normal mixtures are clustering and classification.

[1]E-mail: alexandrovich@mathematik.uni-marburg.de.
[2]See technical supplement: url

Normal mixtures are usually estimated from data via ML (maximum likelihood) or Bayesian approaches. The most widely used method to calculate the MLE is the EM algorithm, see e.g. Redner and Walker (1984). A problem here is the mere linear convergence rate of the EM algorithm and the lack of additional information about the estimate like its Fisher information.

For these reasons, we apply Newton's method (NM) for calculating the MLE. Newton's method has local quadratic convergence rate and provides as a by-product an estimate of the Fisher information matrix.

The main difficulty involved in its the development were the elaborate calculations of the first and second derivatives of a multivariate Gaussian mixture log-likelihood and presenting them in a form appropriate for implementation. The analytic expressions obtained may also be useful for the calculations of derivatives of mixtures in other elliptical families.

The properties of the EM algorithm and respectively Newton's method in its context are subject of several contributions. Everitt (1984) compares six different algorithms for calculation of the MLE of a two-component univariate Gaussian mixture (GM). In particular he compares the EM algorithm, variants of Newton's method with approximated and exact gradient and Hessian, Fletcher-Reeves algorithm and the Nedler-Mead simplex algorithm and concludes that the most satisfactory algorithms are EM and NM with exact gradient and Hessian. Aitkin and Aitkin (1994) also consider a hybrid EM/Newton's algorithm, which starts with five EM iterations and then switches to Newton's method, if Newton's method yields a descent direction, another five EM iterations are done and so on. They use this approach for estimating a MLE of a two-component univariate GM and report a superior behavior of the hybrid algorithm over the pure EM.

Peters and Walker (1978) develop an iterative procedure for calculating the MLE of a multivariate GM. Their approach is based on the so called likelihood equations, which follow from the critical equation. They construct a locally contractive operator and obtain a fixed point problem. In the proof of the contractibility they calculate the exact derivatives of the operator. Unfortunately they do not compare their method with other algorithms and it is hard to judge how well it works. No available implementation of their method is known to us.

Lange (1995), proposes a quasi-Newton acceleration of the EM algorithm, where given an estimate θ_k, the Hessian of the observed log-likelihood is approximated by a decomposition into the Hessian of the conditional expectation of the complete log-likelihood ($Q(\theta|\theta_k)$ from the E-step) minus a part that is constructed via rank-one updates. The gradient of the observed log-likelihood is approximated by the gradient of $Q(\theta|\theta_k)$. In accordance with Lange, such an approach yields a faster converging sequence. Jamshidian and Jennrich (1997) also consider several acceleration methods of the EM algorithm, in which they use a Quasi-Newton approach among others. They find examples where the accelerated versions are dramatically faster than the pure EM algorithm.

Xu and Jordan (1995) discuss the properties of the EM algorithm for calculation of the MLE for GMs. They prove a superiority statement of the EM algorithm over the constrained gradient ascent in a setting with known component weights and covariance matrices and conjecture that it holds also in a general setting. In the numerical experi-

ments they demonstrate a general superiority of EM over the constrained gradient ascent. In their remarks Xu and Jordan speak against the use of NM for estimating GMs due to computational costs and numerical considerations. However they do not compare NM with EM in the numerical experiments and they do not address the problem of high proportion of unobserved information in GM.

In contrast to the mentioned approaches, we suggest the use of the exact gradient and Hessian of the observed log-likelihood of a multivariate Gaussian mixture without restrictions on the number of components. Thanks to a convenient parameterization, the implementation of the analytic derivatives in C, and the possibility to use parallel calculations, the computation of these quantities is done relatively efficiently (see Section 6).

Since Newton's method converges only locally, one needs to provide good starting values. For this purpose we use a k-means clustering followed by a few EM iterations.

An important question is the parameterization of the covariance matrices of the components. The matrices must be symmetric and positive definite. To ensure these restrictions we set $\Sigma^{-1} = LL^T$ and work with L, where L is a lower triangular matrix. Such an approach has the advantage that no matrix inversions have to be calculated during the iterations, so many floating point operations are saved and numerical instabilities are avoided. A similar technique was introduced by Pourahmadi (1999). The representation of the mixture parameters is described in Section 2. In Section 3 Newton's method is briefly discussed. A further subject of the current work is the problem of penalizing the log-likelihood to avoid divergence of the algorithm towards boundary points, see e.g. Chen and Tan (2009), Ciuperca et al (2003). It is described in Section 4. The subject of Section 5 is the Fisher information and some aspects of its calculation.

Several numerical results are given in Section 6, particularly comparisons with EM-implementations from the R packages Mclust 3.4.11 by C. Fraley and A. E. Raftery and Rmixmod 1.1.3. by Remi Lebret et al. An implementation of our method is contained in the R package pGME[3].

The analytic derivatives of the log-likelihood function of a normal mixture as well as some additional results are given in the technical supplement.

2 Parameterization of the Gaussian mixture

A Gaussian mixture is a distribution in $\mathbb{R}^D$ with a density of the following form:

$$g(x; \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, p_1, \ldots, p_{K-1}) = \sum_{i=1}^{K} p_i \phi(x; \mu_i, \Sigma_i), \tag{1}$$

where

$$\phi(x; \mu_i, \Sigma_i) = \frac{1}{\sqrt{2\pi}^D \sqrt{|\Sigma_i|}} e^{-\frac{1}{2}(x-\mu_i)^\intercal \Sigma_i^{-1}(x-\mu_i)}, \qquad p_K = 1 - \sum_{i=1}^{K-1} p_i.$$

[3] http://www.uni-marburg.de/fb12/stoch/research/rpackage

$\phi(x; \mu_i, \Sigma_i)$ is the density of the D-dimensional multivariate Gaussian distribution with mean μ_i and covariance matrix Σ_i. The weights of the components p_i lie in $[0, 1]$ and sum to 1. Now, our aim is to calculate the MLE of the parameter vector

$$\theta = (\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, p_1, \ldots, p_{K-1}).$$

The log-likelihood function for an i.i.d. sample $x_1, \ldots, x_n$ is given by

$$l(\theta) = \log \prod_{t=1}^{n} g(x_t, \theta) = \sum_{t=1}^{n} \log g(x_t, \theta).$$

As indicated before, we apply Newton's method to maximize $l(\theta)$. First of all, we have to find an appropriate parameterization of the mixture g. To ensure that the weights $p_1, \ldots, p_{K-1}$ stay in the interval $[0, 1]$ during the iterations, we parameterize them as follows:

$$p_i = p_i(q) := \frac{q_i^2}{q_1^2 + \ldots + q_{K-1}^2 + 1}, \quad 1 \leq i \leq K-1, \tag{2}$$

where $q = (q_1, \ldots, q_{K-1}) \in \mathbb{R}^{K-1}$. With this approach we avoid optimization under the K inequality restrictions: $p_i \geq 0,\ 1 \leq i \leq K-1,\ \sum_{i=1}^{K-1} p_i \leq 1$.

Regular covariance matrices and their inverses are s.p.d (symmetric and positive definite) matrices, so they can be written as

$$\Sigma_i^{-1} = L_i L_i^{\mathsf{T}} \tag{3}$$

with lower triangular L_i via the Cholesky-decomposition. The only requirement on L_i is that it has only non-zero elements on the diagonal.
From this point on, we parameterize the family of multivariate normals by μ and L:

$$\phi(x; \mu, L) = \frac{1}{\sqrt{2\pi}^D} |L| e^{-\frac{1}{2}(x-\mu)^{\mathsf{T}} L L^{\mathsf{T}} (x-\mu)},$$

and set $\Sigma^{-1} = LL^{\mathsf{T}}$.
So the mixture becomes

$$g(x; \mu_1, \ldots, \mu_K, L_1, \ldots, L_K, q_1, \ldots, q_{K-1}) = \sum_{i=1}^{K} p_i(q) \phi(x; \mu_i, L_i). \tag{4}$$

3 Newton's method

In this section we give a brief introduction to Newton's method for maximizing a twice-differentiable function $f : U \to V$, where $U \subset \mathbb{R}^d, V \subset \mathbb{R}$ for some $d \in \mathbb{N}$. The essence of the approach is to find an appropriate root of the equation $\nabla_\theta f(\theta) = 0$. In our case the log-likelihood function will play the role of f. For a more detailed overview of Newton's

method see e.g. Kelley (1995) or Nocedal and Wright (2006).

The Basics
Newton's method is an iterative method, that constructs a sequence $(\theta_k)_{k\in\mathbb{N}}$, which converges towards the solution θ^*. The iteration is defined by

$$\theta_{k+1} = \theta_k + t_k\Delta_k, \tag{5}$$

where

$$\Delta_k := -H_k^{-1}\nabla_\theta f(\theta_k). \tag{6}$$

H_k is the Hessian of f evaluated at θ_k and t_k is a positive step size at the iteration k. We can consider Δ_k as the maximizer of a quadratic approximation of f in θ_k: $f(\theta_k + p) \approx f(\theta_k)+\nabla_\theta f(\theta_k)^\mathsf{T} p+\frac{1}{2}p^\mathsf{T} H_k p$. The quality of such an approximation depends on the length of p and the smoothness of f in the neighbourhood of θ_k.

The iteration ends as soon as some convergence criterion is fulfilled, e.g. $||\Delta_k|| \leq \epsilon$ for some small ϵ, or the maximal number of iterations is achieved. To start the iterations one has to supply an appropriate starting point θ_1. The selection of the starting point may be a hard problem, since the neighbourhood of θ^* where Newton's method converges may be very small. One possibility to find a starting point, is to prefix another algorithm such as a gradient method or as in our case the EM algorithm. In a sufficiently small neighbourhood of θ^* Newton's method has a quadratic convergence rate, meaning that $||\theta_{k+1} - \theta^*|| \leq c||\theta_k - \theta^*||^2$ for a $c > 0$.

Line Search
Given a direction Δ_k we need to decide how deep to follow it, i.e. to select an appropriate step length t_k, see (5). Since we want to maximize a function, namely the log-likelihood function, one suitable choice would be

$$t_k := \underset{t>0}{\operatorname{argmax}}\, f(\theta_k + t\Delta_k).$$

An exact solution of this problem is often difficult to obtain, so one tries to find an approximation. To achieve a sufficient increase of the objective function, the step length t_k must satisfy the so called Wolfe conditions:

$$\begin{aligned} f(\theta_k + t_k\Delta_k) &\geq f(\theta_k) + c_1 t_k \nabla_\theta f(\theta_k)^\mathsf{T}\Delta_k \\ \nabla_\theta f(\theta_k + t_k\Delta_k)^\mathsf{T}\Delta_k &\leq c_2 f(\theta_k)^\mathsf{T}\Delta_k, \end{aligned}$$

with $0 < c_1 < c_2 < 1$. The constant c_1 is often chosen quite small, near 10^{-4}, see Nocedal and Wright (2006).

The first inequality is sometimes called Armijo condition and ensures that f will make a sufficient increase along the direction Δ_k and the second condition ensures that the step size t_k will be not too small. In practice one often uses the so called backtracking approach to find an appropriate step length. So do we in our implementation.
For more detailed explanation we again refer to Nocedal and Wright (2006).

Solving for Δ_k
At every iteration we have to solve the following system of linear equations for Δ_k:

$$H_k \Delta_k = -\nabla_\theta f(\theta_k).$$

The matrix H_k is symmetric, but not necessarily positive definite. We use the so-called rational Cholesky decomposition $H_k = C_k D_k {C_k}^\mathsf{T}$ with lower triangular matrix C_k and diagonal matrix D_k. It is called rational, since no roots have to be calculated and it works even if some elements in D_k are negative.

As mentioned above, we apply Newton's method to find an appropriate root of the equation

$$\nabla_\theta l(\theta) = 0,$$

where l is the log-likelihood function of the mixture and θ is the parameter vector. For this purpose we need the gradient and the Hessian of the log-likelihood, which are given in the appendix.

4 Penalization

In this section we subscript the log-likelihood and the penalty function with n, to indicate that they depend on the sample size.

The log-likelihood function of a Gaussian mixture is unbounded. To see this, we observe that for a parameter θ with $\mu_1 = x_1$, $\Sigma_1 = \epsilon I$ and μ_2, $\Sigma_2 > 0$ arbitrary, the log-likelihood diverges to ∞ for $\epsilon \to 0$. So it may happen that the algorithm converges toward such a solution. In order to avoid such bad solutions one can penalize the log-likelihood with an additive term $s_n(\theta) = s_n(L_1, \ldots, L_K)$, where n is the sample size. Chen and Tan (2009) formulate conditions that must be satisfied by s_n in order to make the resulting estimator consistent. A function which satisfies these conditions is

$$s_n(L_1, \ldots, L_K) = -a_n \Big(\sum_{i=1}^{K} \operatorname{tr}(S_x L_i {L_i}^\mathsf{T}) + \log \frac{1}{|L_i|^2} \Big),$$

where S_x is the sample covariance matrix and $a_n \to 0$ (e.g. $a_n = \frac{1}{n}$ or $a_n = \frac{1}{\sqrt{n}}$). The penalized log-likelihood has now the form

$$pl_n(\theta) = l_n(\theta) + s_n(\theta). \tag{7}$$

If we enable penalization in our algorithm, the log-likelihood function is replaced by its penalized version. The derivatives of this function arise as the sum of the derivatives of the summands, both are given in the appendix.

5 Fisher information

5.1 Fisher information

At every iteration of Newton's method we obtain the Hessian of the log-likelihood function $\nabla^2_\theta l(\theta_k)$. A well known result from the MLE framework is that under certain conditions

(Redner and Walker, 1984)

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \xrightarrow{d} Z, \qquad Z \sim N(0, I_{\theta_0}^{-1}),$$

where $\hat{\theta}_n$ is a root of the gradient of the log-likelihood function based on n observations and $I_{\theta_0} = \mathbb{E}\nabla_{\theta_0} \log g(X) \nabla_{\theta_0} \log g(X)^\mathsf{T} = -\mathbb{E}\nabla^2_{\theta_0} \log g(X)$, the Fisher information matrix. An approximation of $\mathbb{E}\nabla^2_{\theta_0} \log g(X)$ is given by $-\hat{I}_{\hat{\theta}_n} = \frac{1}{n} \sum_{t=1}^{n} \nabla^2_{\hat{\theta}_n} \log g(X_t)$. The last term is the Hessian of the log-likelihood multiplied by $\frac{1}{n}$. The covariance matrix of Z allows us to construct confidence sets for θ_0.

5.2 Parameter transformation

The parameter of interest is $\theta_{int} = (\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, p_1, \ldots, p_{K-1})$, we however obtain the Hessian w.r.t $\theta_{new} = (\mu_1, \ldots, \mu_K, L_1, \ldots, L_K, q_1, \ldots, q_{K-1})$ as defined in Section 2. Let ψ be the map with $\psi(\theta_{new}) = \theta_{int}$ and D_ψ its derivative matrix. By the chain rule the Fisher information matrix for θ_{int} is $D_\psi^{-\mathsf{T}} I_{\theta_0} D_\psi^{-1}$. ψ is identity in $\mu_1, \ldots, \mu_k$. The partial derivatives of ψ w.r.t. q_i and L_i are given in the appendix 10.2.

6 Numerical experiments

The computation of the gradient and the Hessian of the log-likelihood is the most expensive part of the algorithm and grows linearly with the sample size. We implemented this step and the solver for linear equations in C, since a direct implementation in R was too slow.

An advantage of deriving the log-likelihood is that we have to deal with sums:

$$\nabla_\theta l(\theta) = \sum_{t=1}^{n} \frac{1}{g(x_t, \theta)} \nabla_\theta g(x_t, \theta)$$

resp.

$$\nabla^2_\theta l(\theta) = \sum_{t=1}^{n} \frac{1}{g(x_t, \theta)} \left(\nabla^2_\theta g(x_t, \theta) - \frac{1}{g(x_t, \theta)} \nabla_\theta g(x_t, \theta) \nabla^\mathsf{T}_\theta g(x_t, \theta) \right).$$

The summands can be computed in parallel. We used the OpenMP API in our C-Code for this purpose. The parallelized version is available only for Unix OS.

Algorithms
We compared our algorithm with the EM-implementations from the R packages **Mclust** 3.4.11. and **Rmixmod** 1.1.3.. In addition we considered the SEM algorithm, which is also contained in the package Rmixmod. The SEM algorithm is a stochastic version of the EM algorithm, where in each iteration the unobserved variables (the cluster labels z) are drawn from the conditioned density $g_z(z|x; \theta_k) = g_c(x, z; \theta_k)/g(x; \theta_k)$ and then the simulated complete likelihood g_c is maximized. This algorithm was designed to overcome

the drawbacks of the EM algorithm, such as convergence towards saddle points and slow convergence rate. See [5] for a more detailed explanation.

The interesting characteristics were the execution time, the accuracy of the solution, measured by the BIC values and the number of the iterations. The BIC (Bayesian information criterion) of θ is given by $2l(\theta) - k \log n$, where k is the number of free parameters. In our case k was fixed, so we essentially compared the achieved values of the log-likelihood.

The initial solution for Newton's method was found by a k-means clustering, followed by the EM algorithm, which terminated as soon as the relative log-likelihood change $\frac{l(\theta_{k+1}) - l(\theta_k)}{l(\theta_k)}$ fell below 1e-6. The succeeding Newton's method terminated as soon as one of the following criteria was fulfilled:

C1. The number of iterations achieved 10.

C2. The Hessian of the log-likelihood became singular.

C3. No positive step length was found during back-tracking.

C4. The norm of the Newton's direction Δ_k and the norm of the gradient of the log-likelihood fell below 1e-12.

Newton's method was tested with enabled penalization ($a_n > 0$) and without it ($a_n = 0$). In order to determine the effect of the parallelization, we also tested the parallel version.

The initial solution for the EM algorithm from the package Mclust was the same k-means clustering, which was used for Newton's method, and the algorithm terminated as soon as the relative log-likelihood change fell below 1e-8 (default value in the package Mclust). The method of initialization of the EM/SEM algorithms from the package Rmixmod was an internal random start, since there was no possibility to supply an initial solution. The termination rule was the same as for the EM algorithm from the package Mclust. No restrictions on the parameters were made. We use the following abbreviations for the considered algorithms:

- `NM = Newton's method without penalization`
- `NMP = Newton's method with penalization`
- `EMC = EM algorithm from the package Mclust`
- `EMIX = EM algorithm from the package Rmixmod`
- `SEM  = SEM algorithm from the package Rmixmod`

Procedure

All experiments were realized on a benchmark machine with 12 Intel Xeon X5675 3.07GHz CPUs and 24Gb RAM. We compared the algorithms for five different models, which mimicked some relevant (but of course not all) situations which may occur in the practice. The procedure was the following:

1. Generate N data points from a model.
2. Calculate the MLE with Newton's method (penalized and unpenalized).
3. Calculate the MLE with the EM and SEM algorithms (Mclust and Rmixmod).
4. Save the corresponding numbers of iterations, execution times and BICs.

We repeated this procedure 1000 times and obtained thereby samples of the execution times, iterations numbers and BICs for all algorithms.

Comparing results
We evaluated the results by pairwise comparing the samples of the execution times and the BIC values of the algorithms. In order to compare two samples we used the Wilcoxon-Mann-Whitney test.

Given two algorithms A and B, and the corresponding time samples t_A, t_B and the BIC samples BIC_A, BIC_B we considered the following four hypotheses and corresponding p-values:

1. $H^t_{A\lesssim B} : t_A \geq t_B$ and $p^t_{A\lesssim B}$,
2. $H^{BIC}_{A\lesssim B} : BIC_A \leq BIC_B$ and $p^{BIC}_{A\lesssim B}$,
3. $H^t_{A\gtrsim B} : t_A \leq t_B$ and $p^t_{A\gtrsim B}$,
4. $H^{BIC}_{A\gtrsim B} : BIC_A \geq BIC_B$ and $p^{BIC}_{A\gtrsim B}$.

There is a significant advantage of A over B in terms of $crit \in \{t, BIC\}$ if we can reject the hypothesis $H^{crit}_{A\lesssim B}$. We assume that there is no significant advantage of B over A if we cannot reject the hypothesis $H^{crit}_{A\gtrsim B}$. These thoughts lead us to the following definition of a benchmark $\text{bench}(A, B) \in \{-1, 0, 1, 2\}$.

$$\begin{aligned}\text{bench}(A,B) := &- \mathbf{1}_{\{p^t_{A\gtrsim B}\leq 0.05 \,\wedge\, p^{BIC}_{A\gtrsim B}\leq 0.05\}} \\ &+ \mathbf{1}_{\{p^t_{A\lesssim B}\leq 0.05 \,\wedge\, p^{BIC}_{A\gtrsim B}\geq 0.05\}} \\ &+ \mathbf{1}_{\{p^{BIC}_{A\lesssim B}\leq 0.05 \,\wedge\, p^t_{A\gtrsim B}\geq 0.05\}} \\ &+ 2\cdot \mathbf{1}_{\{p^t_{A\lesssim B}\leq 0.05 \,\wedge\, p^{BIC}_{A\lesssim B}\leq 0.05\}}.\end{aligned}$$

In words, we set $\text{bench}(A, B)$ to -1 if A was both significantly slower and significantly worse (in terms of BIC) than B. We set $\text{bench}(A, B)$ to 1 if *either* A was significantly faster than B and at the same time was not significantly worse *or* if A was significantly better than B and at the same time was not significantly slower. Furthermore, if *either* A was significantly faster and significantly worse than B *or* if A was significantly better and significantly slower than B we set $\text{bench}(A, B)$ to 0. Finally, we set $\text{bench}(A, B)$ to 2 if A was significantly faster and significantly better than B.

A higher $\text{bench}(A, B)$ implies an advantage of A over B in a given model/sample size constellation. However $\text{bench}(A, B)$ never achieved 2 in our simulations, since there were

no significantly differences in BIC. Some tables with benchmark values are given in the appendix and the rest in the supplementary material to this paper.

Convergence failures
Especially Newton's method fails to converge if the initial solution is not well chosen. We say the algorithm A failed to converge if either a fatal numerical error occurred (such as an attempt to invert a singular matrix) or the solution was an extreme outlier. We say a result of the algorithm A as an extreme outlier if and only if the distance between the corresponding BIC value and the median of the BIC sample is greater than 3 times the interquartile range of the sample. Such atypical BIC values correspond to saddle points or local maxima of the log-likelihood with very poor parameter estimates. We removed all such failure cases from the data before we compared the algorithms. The counts of such failures for each model and each sample size are given in the appendix in Table 5.

Models
Following models were considered:
Model 1 $K = 2,\ D = 3.$

$$\begin{aligned}
\mu_1 &= (1\ \ 1\ \ 1)^\mathsf{T}, \quad \mu_2 = (2\ \ 2\ \ 2)^\mathsf{T} \\
\Sigma_1^{i,j} = \Sigma_2^{i,j} &= \begin{cases} 0.2 & i \neq j \\ 1 & i = j \end{cases} \quad 1 \leq i, j \leq D, \\
p_1 &= p_2 = 0.5.
\end{aligned}$$

This model is interesting, since we are in $\mathbb{R}^3$ and the corresponding covarince matrices are dense, so we have a relatively complex correlation structure within the components and a moderate number of paramters to estimate.

Model 2 $K = 5,\ D = 2.$

$$\begin{aligned}
&\mu_1 = (4\ \ 5)^\mathsf{T}, \quad \mu_2 = (1.5\ \ 5)^\mathsf{T}, \mu_3 = (2\ \ 4.5)^\mathsf{T}, \quad \mu_4 = (4.1\ \ 1)^\mathsf{T}, \quad \mu_5 = (5\ \ 1)^\mathsf{T}, \\
&\Sigma_1 = \begin{pmatrix} 0.3 & 0.05 \\ 0.05 & 0.3 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.1 & 0.05 \\ 0.05 & 0.1 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix}, \Sigma_{4,5} = \begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.2 \end{pmatrix}, \\
&p_1 = p_4 = p_5 = 0.2, \quad p_2 = 0.25, \quad p_3 = 0.15.
\end{aligned}$$

The interesting characteristics of this model are the high number of components and a strong overlap between components 2 and 3, and 4 and 5, the corresponding 2-component mixtures are unimodal or weakly bimodal respectively.

Model 3 $K = 2,\ D = 5.$

$$\begin{aligned}
\mu_1 &= (1\ \ 1\ \ 1\ \ 1\ \ 1)^\mathsf{T}, \quad \mu_2 = (2\ \ 2\ \ 2\ \ 2\ \ 2)^\mathsf{T}, \\
\Sigma_1^{i,j} = \Sigma_2^{i,j} &= \begin{cases} 0.2 & i \neq j \\ 1 & i = j \end{cases} \quad 1 \leq i, j \leq D, \\
p_1 &= p_2 = 0.5.
\end{aligned}$$

We include this model in our consideration, since it is quite high dimensional and it is interesting to study the behavior of Newton's method in higher dimensions, but with small number of components.

Model 4 $K = 7,\ D = 2.$

$$\mu_1 = (4\ \ 5)^\mathsf{T}, \quad \mu_2 = (1.5\ \ 5)^\mathsf{T}, \quad \mu_3 = (2\ \ 4.5)^\mathsf{T}, \quad \mu_4 = (4.1\ \ 1)^\mathsf{T}, \quad \mu_5 = (5\ \ 1)^\mathsf{T},$$
$$\mu_6 = (3\ \ 2)^\mathsf{T}, \quad \mu_7 = (5\ \ 2)^\mathsf{T},$$
$$\Sigma_1 = \begin{pmatrix} 0.3 & 0.05 \\ 0.05 & 0.3 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.1 & 0.05 \\ 0.05 & 0.1 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix}, \Sigma_{4,5,6,7} = \begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.2 \end{pmatrix},$$
$$p_1 = 0.2, \quad p_2 = p_3 = 0.15, \quad p_4 = p_5 = 0.1, \quad p_6 = p_7 = 0.15.$$

Model 5 $K = 9,\ D = 2.$

$$\mu_1 = (4\ \ 5)^\mathsf{T}, \quad \mu_2 = (3\ \ 5)^\mathsf{T}, \quad \mu_3 = (2\ \ 4.5)^\mathsf{T}, \quad \mu_4 = (4.1\ \ 1)^\mathsf{T}, \quad \mu_5 = (5\ \ 1)^\mathsf{T},$$
$$\mu_6 = (3\ \ 2)^\mathsf{T}, \quad \mu_7 = (5\ \ 2)^\mathsf{T}, \quad \mu_8 = (-1\ \ 2)^\mathsf{T}, \quad \mu_9 = (1\ \ -2)^\mathsf{T}.$$
$$\Sigma_1 = \begin{pmatrix} 0.3 & 0.05 \\ 0.05 & 0.3 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.1 & 0.05 \\ 0.05 & 0.1 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix}, \Sigma_{4,5,6,7,8} = \begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.2 \end{pmatrix},$$
$$\Sigma_9 = \begin{pmatrix} 0.3 & -0.1 \\ -0.1 & 0.3 \end{pmatrix}, \quad p_i = 1/9, \quad 1 \le i \le 9.$$

Last two models represent settings with a high amount of the unobserved information.

Sample sizes
We chose sample sizes 250, 500 and 1000 for Models 1, 2 and 3. Model 4 and Model 5 were constructed to have a high amount of unobserved information, so we tested them only for the sample size 5000.

Choice of the penalty weight
The theory doesn't say anything concrete about the optimal choice of the penalty weights a_n in practice. We determined the best choice for each model and each sample size by a grid search over an equidistant grid of 200 values in $[0, \frac{5}{\sqrt{n}}]$. For each value of a_n on the grid, each model and each sample size ($n \in \{250, 500, 1000\}$) Newton's method was applied 1000 times to a randomly generated sample and the number of failures n_f was counted. The grid value with the lowest n_f was used in the simulation study. It came out that penalization could not effectively reduce the number of failures. In several situations the best choice for the penalization weight was 0 (no penalization). We will discuss this point below.

Simulation results
The complete set of tables and figures is contained in the supplementary material. We present here only a few of them. As we can see from the results, our algorithm was in many cases faster than the EM algorithm. The differences in BIC were in the most cases not significant, as the p-values from the corresponding Wilcoxon tests suggest. An

exception was the SEM algorithm for Models 2 and 3 - in the cases where SEM converged, it achieved a significantly better BIC values than the rest. The only problem was, that such cases were quite rare (see Table 5), so we could achieve the same effect for any other algorithm by considering e.g. only the best 30% of results.

The results differ depending on the model and the sample size. For sample sizes 250 and 500 the EM algorithm usually had a better performance, followed by Newton's method and the SEM algorithm. The comparison of both the EM implementations shows an advantage for the package Mclust in most cases. In many cases when the Mclust implementation of EM was faster, the Rmixmod's one was slower than Newton's method. The constellations where the Rmixmod EM implementation was faster than the Mclust's one, were Model 3 and $n = 1000$ and Model 4 and Model 5 and $n = 5000$.

Newton's method outperformed the rest clearly for $n = 1000$ and Model 2 and $n = 5000$ and Model 4 and Model 5 - the constellations with the highest amount of the unobserved information in the EM setting.

The plots of the ecdfs of the time samples show that the ecdf of Newton's method often lies under the corresponding ecdf of the EM algorithm for small values on the x axis, and above it for higher values. That means that the EM time distribution has more mass at small values but also more mass at high values than Newton's method.

The failure counts are presented in Table 5. We note that the numbers for Newton's method are higher than for both EM implementations. The reason for such a behaviour, is the quite small convergence radius of Newton's method. The penalization could not reduce the number of the failures of Newton's method (see Table 5). The reason for this is that most failures of Newton's method correspond to saddle points and other critical points of the log-likelihood and not to the boundary of the parameter space. One astonishing observation is that penalized Newton's method was in some cases superior over the non-penalized version, see e.g. Table 2 or supplement material. In fact the ecdf's of the time samples confirm this claim. A possible explanation of this fact may be that penalizing makes the log-likelihood more smoothly.

A notable observation is that the SEM algorithm was outperformed by other algorithms throughout almost all models and all sample sizes and failed to converge conspicuously often, especially in the case of Model 2 and sample sizes 500 and 1000 and Model 3 and all sample sizes (see Table 5). SEM was very unstable as well, as the standard deviations of the corresponding BIC samples suggest. It is quite an unexpected result, since the SEM algorithm was designed as an improvement of the EM algorithm. Gaussian mixture models seem not to be the application where the advantages of SEM justify its usage, like mixtures of distributions outside the exponential family (see e.g. [11]).

The parallel version of Newton's method was 2-6 times faster than the non-parallel one, depending on model and sample size.

7 Conclusion

The numerical experiments show that our algorithm, which is a combination of the EM algorithm and Newton's method, is in many cases faster than the pure EM algorithm.

It is well known that the EM algorithm has difficulties if the amount of missing data is high. This amount increases with K and n, and indeed we see a clear advantage of our approach for Model 2 ($K = 5$) and sample size $n = 1000$ and Models 4 ($K = 7$) and 5 ($K = 9$) and sample size $n = 5000$. These constellations correspond to the highest amount of unobserved information in the EM setting among the tested models.

Such results would be impossible without the chosen parameterization of the covariance matrices of the components. It avoids the numerically unstable and costly matrix inversions.

However, compared to the EM algorithm, Newton's method requires much more storage and much more floating point operations per iteration. The size of the Hessian is $\mathcal{O}(K^2D^4)$, so one would guess the EM algorithm should outperform Newton's method in higher dimensions. Indeed, our implementation of Newton's method became slower than the EM algorithm on average machines for dimensions $D \geq 5$ (see results of Model 3).

Better implementations and faster computers should redress this problem. Quasi-Newton methods, which do not require the computation of the Hessian matrix, would redress the dimensionality issue as well. However the local quadratic convergence rate would become lost in that case.

The advantages of Newton's method should carry more weight in other mixture models, such as mixtures of t-distributions, since no explicit update formulas for all parameters in the EM algorithm exist there. Also in other settings with a high fraction of the unobserved information, where the EM algorithm is applied for the parameter estimation, Newton's method should perform faster.

One of the most relevant drawbacks of Newton's method in practice is the necessity of providing rather accurate starting values. The preceding EM iterations can resolve this problem only partly, since it is not clear a-priori how long to iterate before starting the Newton's iterations. Our approach to iterate the EM algorithm until the relative log-likelihood change fell below 1e-6 worked well, but in some few cases it was not enough to achieve the convergence region of Newton's method and algorithm failed, see Table 5. Xu und Jordan find in [22] a representation of the EM iteration as $\theta_{k+1} = \theta_k + P_k \nabla l(\theta_k)$, where a P_k is a well-conditioned matrix, which takes the place of the inverse of the Hessian H_k^{-1} in NM iterations. Hence EM can be considered as a variant of the Quasi-Newton methods. A possible approach for improvement of the both methods should be the use of a convex combination of the both matrices $\omega_k P_k + (1-\omega_k) H_k^{-1}$ as the iteration matrix. In doing so, one should adapt $\omega_k \in [0, 1]$ during the iterations. At the beginning ω_k should be near 1 and at the end near 0. The difficulty is to find appropriate criteria for adapting ω_k, it may depend on the condition number of the resulting matrix and/or on the negative definiteness of H_k.

8 Acknowledgements

I would like to thank Prof. Dr. Hajo Holzmann for his help in the writing of this manuscript.

9 Tables and figures

9.0.1 Tables

All tables and figures are given in the supplement to this paper. We present here only some results for Models 2, 3 and 4.

Algo	Model 1			Model 2			Model 3			Model 4	Model 5
n =	250	500	1000	250	500	1000	250	500	1000	5000	5000
NM	3	2	2	7	11	30	2	9	20	48	70
NMP	2	2	2	7	11	30	6	10	23	48	70
EMC	0	0	0	11	5	2	0	0	1	0	23
EMIX	0	0	0	6	4	0	7	0	3	0	3
SEM	52	1	0	814	637	333	397	347	222	91	249

Table 1: Failure counts (out of 1000).

A \ B	NM	NMP	EMC	EMIX	SEM
n = 1000					
NM	-	0	1	1	0
NMP	0	-	1	1	0
EMC	0	0	-	0	0
EMIX	0	0	1	-	0
SEM	0	0	0	0	-

Table 2: **Model 2. Benchmarks bench**(A, B)

A \ B	NM	NMP	EMC	EMIX	SEM
n = 250					
NM	-	0	0	-1	-1
NMP	1	-	0	-1	0
EMC	1	1	-	0	0
EMIX	1	1	0	-	0
SEM	1	0	0	0	-
n = 500					
NM	-	0	0	-1	0
NMP	1	-	0	-1	0
EMC	1	1	-	1	0
EMIX	1	1	0	-	0
SEM	0	0	0	0	-
n = 1000					
NM	-	0	0	0	0
NMP	1	-	0	0	0
EMC	1	1	-	0	0
EMIX	1	1	1	-	0
SEM	0	0	0	0	-

Table 3: **Model 3. Benchmarks bench**(A, B)

A \ B	NM	NMP	EMC	EMIX	SEM
n = 5000					
NM	-	0	1	1	0
NMP	0	-	1	1	0
EMC	0	0	-	0	0
EMIX	0	0	1	-	0
SEM	0	0	0	0	-

Table 4: **Model 4. Benchmarks bench**(A, B)

A \ B	NM	NMP	EMC	EMIX	SEM
n = 5000					
NM	-	0	1	0	0
NMP	0	-	1	0	0
EMC	0	0	-	-1	-1
EMIX	0	0	1	-	-1
SEM	0	0	1	1	-

Table 5: **Model 5. Benchmarks** $bench(A, B)$

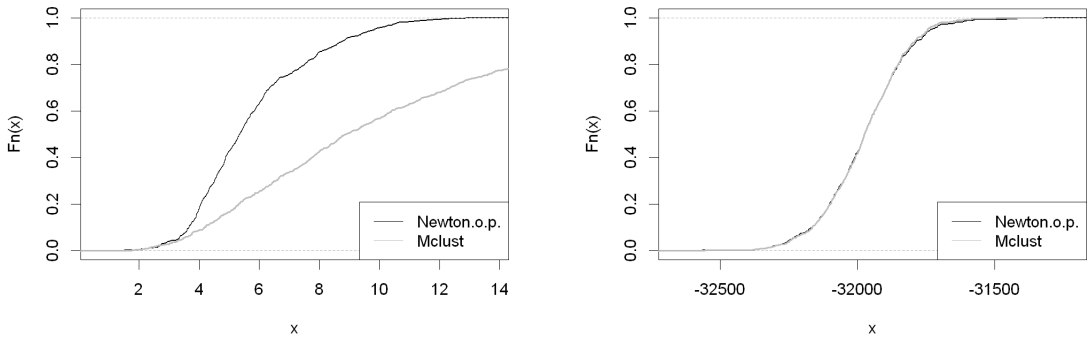

Table 7: ecdf's of the time samples (left) and the BIC samples (right) of the Newton's method (black) and the Mclust-EM algorithm (grey) for Model 5 and sample size of fitted data 5000.

9.0.2 Figures

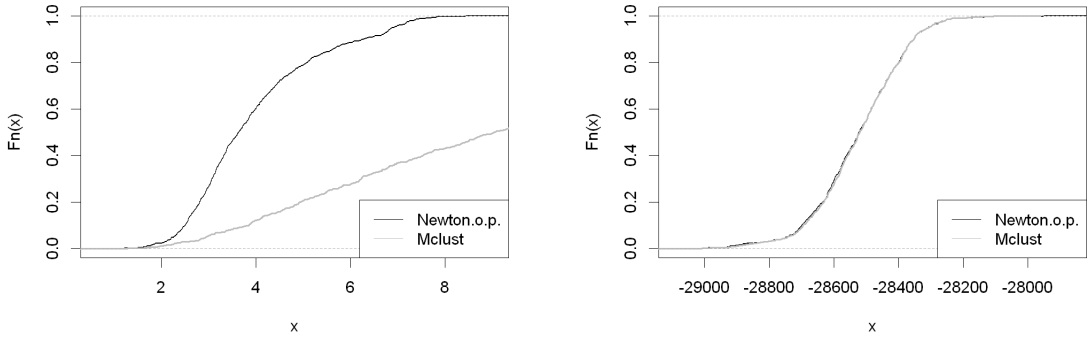

Table 6: ecdf's of the time samples (left) and BIC samples (right) of Newton's method (black) and the Mclust-EM algorithm (grey) for Model 4 and sample size of fitted data 5000.

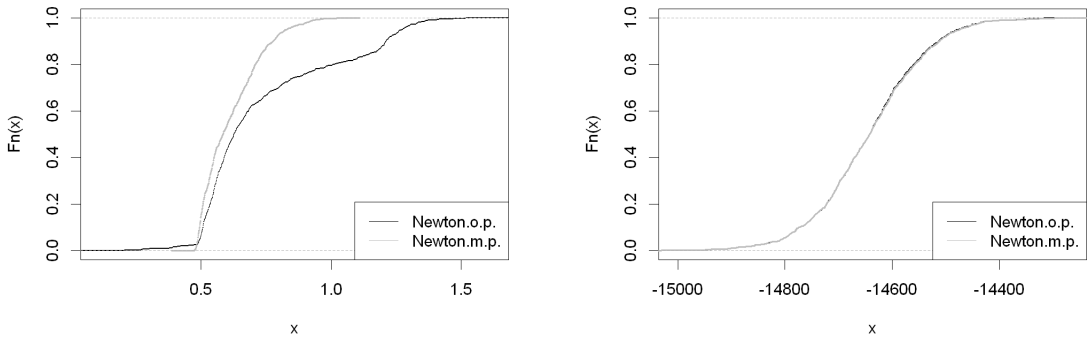

Table 8: ecdf's of the time samples (left) and BIC samples (right) of Newton's method (black) and the penalized Newton's method (grey) for Model 3 and sample size of fitted data 1000.

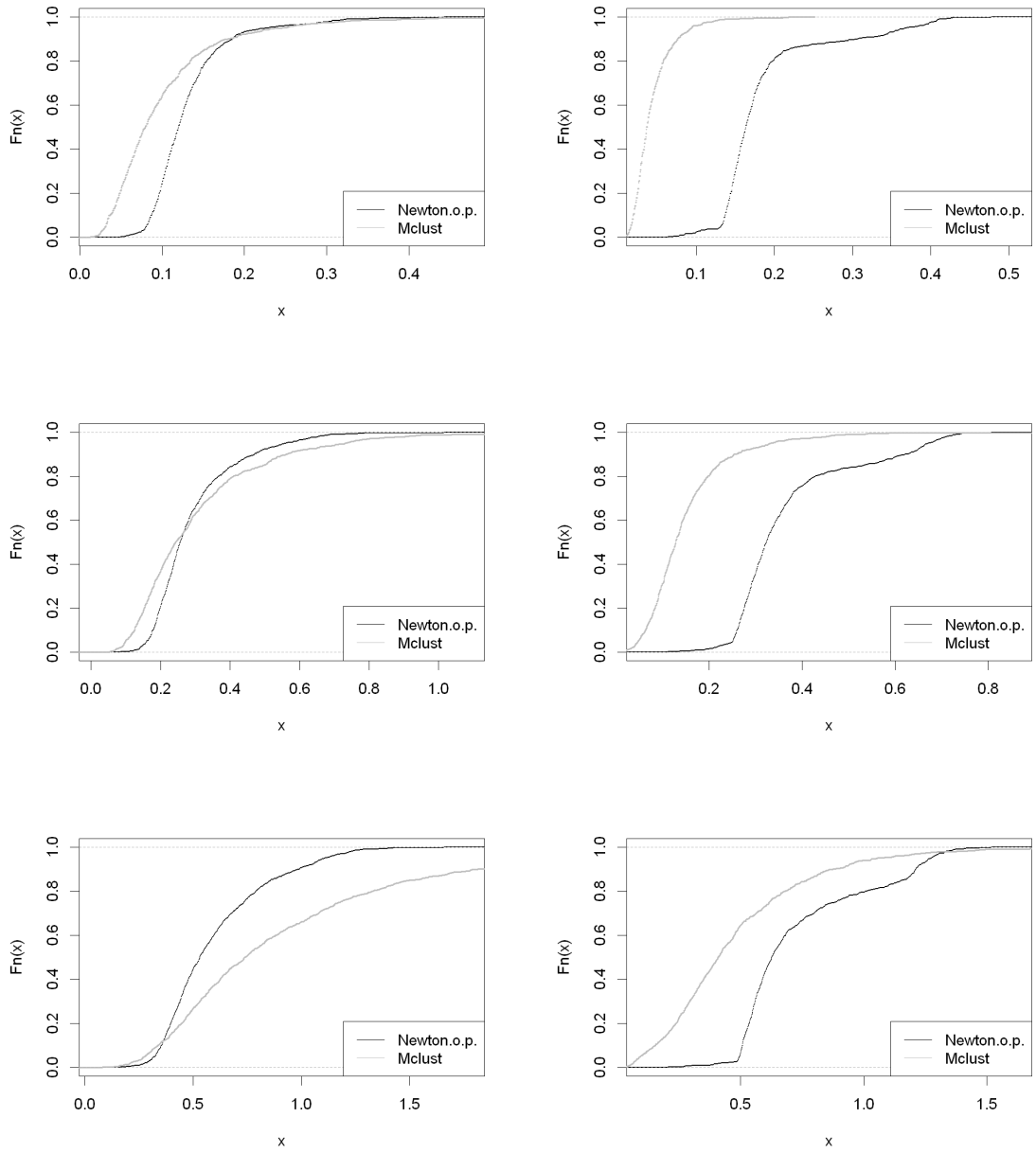

Table 9: ecdf's of the time samples of Newton's method (black) and the Mclust-EM algorithm (grey) for the Models 2 (left column) and 3 (right column) and sample sizes of fitted data 250, 500, 1000 (top-down).

References

[1] Aitkin M., Aitkin I.: *A hybrid EM/Gauss-Newton algorithm for maximum likelihood in mixture distributions*. Statistics and Computing (1996), pp. 127-130.

[2] Everitt B.S.: *Maximum Likelihood Estimation of the Parameters in a Mixture of Two Univariate Normal Distributions; A Comparison of Different Algorithms*. Journal of the Royal Statistical Society. Series D, vol. 33, no. 2 (1984), pp. 205-215.

[3] Louis T.A.: *Finding the Observed Information Matrix when Using the EM Algorithm*. Journal of the Royal Statistical Society. Series B, vol. 44, no. 2 (1982), pp. 226-233.

[4] Peters B.C. JR., Walker H.F.: *An Iterative Procedure for Obtaining Maximum-Likelihood Estimates of the Parameters for a Mixture of Normal Distributions*. SIAM Journal on Applied Mathematics, vol. 25, no. 2 (1978), pp. 362-378.

[5] Celeux G., Chauveau D., Diebolt J.: *On Stochastic Versions of the EM Algorithm*. Inria, Rapport de recherche no. 2514 - Mars 1995.

[6] Chen J., Tan X.: *Inference for multivariate normal mixtures*. Journal of Multivariate Analysis 100 (2009) pp. 1367-1383.

[7] Ciuperca G., Ridolfi A., Idier J.: *Penalized Maximum Likelihood Estimator for Normal Mixtures*. Scandinavian Journal of Statistics, vol 30 (2003) pp. 45-59.

[8] Dempster A. P., Laird N. M. and Rubin D. B.: *Maximum Likelihood from Incomplete Data via the EM Algorithm*. Journal of the Royal Statistical Society, Series B, vol. 39, no.1 (1977), pp. 1-38.

[9] Fraley C., Raftery A. E.: *Mclust Version 3 for R: Normal Mixture Modeling and Model-Based Clustering*.

[10] Jamshidian M., Jennrich I. R.: *Acceleration of the EM Algorithm by Using Quasi-Newton Methods*. Journal of the Royal Statistical Society. Series B, 59 (1997), pp. 569-587.

[11] Jank W.: *The EM Algorithm, Its Stochastic Implementation and Global Optimization: Some Challenges and Opportunities for OR*, (2006).

[12] Kelley C.T.: *Iterative methods for linear and nonlinear equations*. SIAM Publications, Philadelphia, PA, (1995).

[13] Lange K.: *A quasi-Newton acceleration of the EM algorithm*. Statistica Sinica 5 (1995) pp.1-18

[14] Lebret R., Ioveff S., Langrognet F.: *Rmixmod: An interface of MIXMOD, v.1.1.3*.

[15] McLachlan G. J., Bashford, K.E.: *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, (1988).

[16] McLachlan G. J., Krishnan, T.: *The EM Algorithm and Extensions*. Willey, (2008).

[17] Nocedal J., Wright S. J.: *Numerical Optimization*. Springer, (2006).

[18] Pourahmadi M.: *Joint mean-covariance models with applications to longitudinal data: unconstrained parameterisation*. Biometrika vol. 86 (1999) pp. 677-690.

[19] Redner R. A., Walker H.F.: *Mixture Densities, Maximum Likelihood and the Em Algorithm*. SIAM Review, Vol. 26, No. 2 (1984), pp. 195-239.

[20] R-Package pGME. http://www.uni-marburg.de/fb12/stoch/research/rpackage.

[21] Titterington D.M., Smith A.F.M., Makov U.E.: *Statistical Analysis of Finite Mixture Distributions.*. John Wiley & Sons, New York, (1985).

[22] Xu L., Jordan M.I.: *On Convergence Properties of the EM Algorithm for Gaussian Mixtures*. Neural Computation, vol. 8 (1995), pp. 129-151.

The following part is intended for online publication only.

10 Technical supplement

List of Notations

e_i	The i'th unit-vector
v_k	The k'th vector
v_k^i	The i'th element of the vector v_k
L_k	The k'th matrix
$L_k^{i,j}$	The element in the i'th row and the j'th column of the matrix L_k
$L_k^{i,\cdot}$	The i'th row of the matrix L_k
$L_k^{\cdot,i}$	The i'th column of the matrix L_k
$\vec{L}_k$	The half-vectorization of the quadratic matrix L_k (see definition 10.1)
$\vec{L}_k^i$	The i'th element of the vector $\vec{L}_k$
$\vec{z}_i$	The index of the row of the i'th element of $\vec{L}_k$ in L_k
$\vec{s}_i$	The index of the column of the i'th element of $\vec{L}_k$ in L_k
$\|L\|$	The absolute value of the determinant of the matrix L_k
$\nabla_\theta l$	The gradient of the function l w.r.t. θ
$\nabla_\theta^2 l$	The Hessian of the function l w.r.t. θ
$\|\|v\|\|$	The euclidean norm of the vector v
$\delta_i(j)$	Kronecker delta
$\text{diag}(v)$	For a vector v: a diagonal matrix with elements of v on the diagonal
$\text{diag}(L)$	For a matrix L: the diagonal elements of M as a vector
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^{D\times D}$	The set of real $D \times D$ matrices
$\mathbb{R}^D$	The set $\mathbb{R}^{D\times 1}$
$\mathbb{R}_{lt}^{D\times D}$	The set of real lower triangular $D \times D$ matrices

It is important to note, that in the following subscripts are used as labels of the objects (vectors or matrices) and superscripts are used to pick elements from the objects.

10.1 Derivatives

For the calculation of the derivatives we express the parameter θ as a vector in $\mathbb{R}^{K(D+\frac{1}{2}(D+1)D+1)-1}$. In order to vectorize a lower triangular matrix L, we need the following definition:

Definition 10.1 Let $L \in \mathbb{R}_{lt}^{D\times D}$ (a $D \times D$ lower triangular or symmetric matrix). The bijective mapping

$$\text{vech} : \mathbb{R}_{lt}^{D\times D} \to \mathbb{R}^{\frac{D(D+1)}{2}},\ L \mapsto \vec{L},$$

where $\vec{L}^i := L^{\vec{z}_i,\vec{s}_i}$ for $1 \leq i \leq D(D+1)/2$, with

$$\vec{z}_i := \lceil \frac{1}{2} + \sqrt{\frac{1}{4} + 2i} - 1 \rceil, \tag{8}$$

$$\vec{s}_i := i - \frac{\vec{z}_i(\vec{z}_i - 1)}{2}. \tag{9}$$

is the *half-vectorization* of the matrix L.

The mapping vech concatenates the elements of L row wise into a vector. In each row only elements up to the diagonal are taken: $\vec{L} = (L^{i,j} : i = 1, \ldots, D, j = 1, \ldots, i)$.
The equations (8) and (9) have the following motivation: the coordinates of the i'th element of $\vec{L}$ in L, namely $\vec{z}_i$ (row index) and $\vec{s}_i$ (column index), must satisfy

$$i = \frac{(\vec{z}_i - 1)\vec{z}_i}{2} + \vec{s}_i,$$
$$0 < \vec{s}_i \leq \vec{z}_i.$$

The only solution of this problem is given by the displayed equations. In the following we will use $\vec{L}$ as well as L in our formulas, depending on what is more suitable in a concrete situation. See also the list of notations to avoid confusion.

Proposition 10.2 *Let L be a $D \times D$ lower triangular or symmetric matrix, then for $1 \leq j \leq i \leq D$ we have $L^{i,j} = \vec{L}^{k^\Delta(i,j)}$, where $k^\Delta(i,j) = \frac{i(i-1)}{2} + j$.*

In order to select the i'th row of L in $\vec{L}$ we need to pass the first $i-1$ rows, which form the first $\frac{i(i-1)}{2}$ elements in $\vec{L}$.

In the following subsection we calculate the derivatives of the log-likelihood function with respect to the parameter vector

$$\theta = (\mu_1, \ldots, \mu_K, \vec{L}_1, \ldots, \vec{L}_K, q_1, \ldots, q_{K-1}).$$

10.1.1 First derivatives of the mixture density

Now we calculate the first partial derivatives of g w.r.t. the parameters μ_i, $\vec{L}_i$ and q_i.

For $1 \leq i \leq K$ holds

$$\frac{\partial g}{\partial \mu_i} = p_i \phi(x; \mu_i, L_i)(x - \mu_i)^\mathsf{T} \Sigma_i^{-1}.$$

For $1 \leq i \leq K-1$ and $p = (p_1, \ldots, p_{K-1}) \in \mathbb{R}^{1 \times K-1}$ holds

$$\frac{\partial g}{\partial q_i} = \frac{\partial g}{\partial p} \frac{\partial p}{\partial q_i},$$

where

$$\frac{\partial g}{\partial p} = (\phi_1 - \phi_K, \ldots, \phi_{K-1} - \phi_K) \qquad \text{and} \tag{10}$$

$$\frac{\partial p}{\partial q_i} = -\frac{2q_i}{(\sum_{j=1}^{K-1} q_j^2 + 1)^2} (q_1^2, \ldots, \overbrace{-\sum_{j=1,\, j \neq i}^{K-1} q_j^2 - 1}^{i^{th} \text{ component}}, \ldots, q_{K-1}^2)^\mathsf{T}, \tag{11}$$

where $\phi_j = \phi(x; \mu_j, L_j)$.
Before we can obtain formulas for derivatives with respect to the covariance parameters, we need the following properties of a lower triangular matrix L:

1.
$$|L| = \prod_{i=1}^{D} L^{i,i},$$

2. For $i \geq j$:
$$\frac{\partial |L|}{\partial L^{i,j}} = \begin{cases} 0 & i \neq j \\ \prod\limits_{k=1,\, k\neq i}^{D} L^{k,k} & \text{otherwise} \end{cases},$$

3.
$$\frac{\partial (x-\mu)^{\mathsf{T}} L L^{\mathsf{T}} (x-\mu)}{\partial L} = 2(x-\mu)(x-\mu)^{\mathsf{T}} L.$$

The notation $\frac{\partial f}{\partial L}$ for a function f which maps a matrix L onto a real number $f(L)$ means a matrix of derivatives $(\frac{\partial f}{\partial L_{i,j}})_{i,j}$. With above formulas we obtain: for $1 \leq i \leq K$ holds

$$\frac{\partial g}{\partial \vec{L}_i} = \text{vech}\left(\frac{p_i}{|L_i|} \phi(x; \mu_i, L_i) \left[\text{diag}(\prod_{k\neq 1}^{D} L_i^{k,k}, \ldots, \prod_{k\neq D}^{D} L_i^{k,k}) - |L_i| (x-\mu_i)(x-\mu_i)^{\mathsf{T}} L_i \right] \right).$$

Since L_i is a lower triangular matrix, we can speedup the calculations of $(q_{r,s})_{r,s} := (x-\mu_i)(x-\mu_i)^{\mathsf{T}} L_i$ by setting $M_i := (x-\mu_i)(x-\mu_i)^{\mathsf{T}}$ and considering

$$q_{r,s} = \sum_{k=1}^{D} M_i^{k,r} L_i^{k,s} = \sum_{k=s}^{D} M_i^{k,r} L_i^{k,s}.$$

Now we can calculate the gradient of the mixture-density w.r.t. θ:

$$\nabla_\theta g = \left(\begin{array}{cccccccc} \frac{\partial g}{\partial \mu_1} & \cdots & \frac{\partial g}{\partial \mu_K} & \frac{\partial g}{\partial \vec{L}_1} & \cdots & \frac{\partial g}{\partial \vec{L}_K} & \frac{\partial g}{\partial q_1} & \cdots \quad \frac{\partial g}{\partial q_{K-1}} \end{array} \right)^{\mathsf{T}}.$$

10.1.2 First derivatives of the log-likelihood

In the next step we differentiate the log-likelihood function

$$l(\theta) = \log(\prod_{t=1}^{n} g(x_t, \theta)) = \sum_{t=1}^{n} \log(g(x_t, \theta)).$$

With the formulas we obtained above, it is easily done:

$$\nabla l(\theta) = \sum_{t=1}^{n} \frac{1}{g(x_t, \theta)} \nabla_\theta g(x_t, \theta).$$

10.1.3 Second derivatives of the mixture density

For Newton's method we also need the second derivatives of the log-likelihood function, e.g. its Hessian. In the first step we calculate the Hessian of the mixture density $\nabla^2_\theta g(x,\theta)$ for a fixed x. For two natural numbers a, b the value $\delta_a(b)$ is 1 if $a = b$ and 0 otherwise.
For the following computations we use $M_i = (x-\mu_i)(x-\mu_i)^\mathsf{T}$.

For $1 \le i < k \le K$ holds

$$\frac{\partial^2 g}{\partial q_i \partial q_k} = \frac{\partial g}{\partial p}\frac{\partial^2 p}{\partial q_i \partial q_k},$$

where for $i \neq k$, $1 \le l \le K-1$:

$$\frac{\partial^2 p_l}{\partial q_i \partial q_k} = \frac{1}{(\sum_{j=1}^{K-1} q_j^2 + 1)^3} \cdot \begin{cases} 8q_i q_k q_l^2 & l \neq i,k, \\ 4q_i q_k((\sum_{j=1}^{K-1} q_j^2 + 1) - 2(\sum_{j\neq i}^{K-1} q_j^2 + 1)) & l = i, \\ 4q_i q_k(2q_k^2 - (\sum_{j=1}^{K-1} q_j^2 + 1)) & l = k. \end{cases}$$

$$\frac{\partial^2 p_l}{\partial q_i^2} = \frac{1}{(\sum_{j=1}^{K-1} q_j^2 + 1)^3} \cdot \begin{cases} 2q_l^2\left[4q_i^2 - (\sum_{j=1}^{K-1} q_j^2 + 1)\right] & l \neq i, \\ 2(\sum_{j\neq i}^{K-1} q_j^2 + 1)\left[(\sum_{j=1}^{K-1} q_j^2 + 1) - 4q_i^2\right] & \text{else.} \end{cases}$$

$$\frac{\partial^2 g}{\partial q_i \partial \vec{L}_j} = \frac{\partial^2 g}{\partial \vec{L}_j \partial p}\frac{\partial p}{\partial q_i},$$

where for $1 \le l \le K-1$

$$\frac{\partial^2 g}{\partial \vec{L}_j \partial p_l} = (\delta_j(l) + \delta_j(K))\frac{(-1)^{\delta_j(K)}}{|\vec{L}_j|}\phi(x,\mu_j,L_j)\left[\operatorname{vech}(\operatorname{diag}(\prod_{k\neq 1}^{D} L_j^{k,k}, \ldots, \prod_{k\neq D}^{D} L_j^{k,k}) - |L_j| M_j \vec{L}_j)\right]$$

and $\frac{\partial p}{\partial q_i}$ is given by (10.1.1).

For $1 \le i,j \le K$ holds

$$\frac{\partial^2 g}{\partial \mu_i \partial \mu_j^\mathsf{T}} = \delta_i(j) p_i \phi(x,\mu_i,L_i)\left[\Sigma_i^{-1}(x-\mu_i)(x-\mu_i)^\mathsf{T}\Sigma_i^{-1} - \Sigma_i^{-1}\right],$$

$$\frac{\partial^2 g}{\partial \vec{L}_i \partial \mu_j} = \delta_i(j)\left(\begin{array}{ccc} \frac{\partial^2 g}{\partial \vec{L}_i^1 \partial \mu_i} & \cdots & \frac{\partial^2 g}{\partial \vec{L}_i^{D(D+1)/2} \partial \mu_i} \end{array}\right),$$

where

$$\frac{\partial^2 g}{\partial \vec{L}_i^j \partial \mu_i} = \frac{p_i}{|L_i|}\phi(x,\mu_i,L_i)\left[(x-\mu_i)^\mathsf{T}\Sigma_i^{-1}\left(\delta_{\vec{z}_j}(\vec{s}_j)\prod_{k\neq \vec{z}_j}^{D} L_i^{k,k} - |L_i|[M_i L_i]^{\vec{z}_j,\vec{s}_j}\right) + |L_i| L_i^{\cdot,\vec{s}_j\,\mathsf{T}}(x-\mu_i)^{\vec{z}_j} + e_{\vec{z}_j} L_i^{\cdot,\vec{s}_j\,\mathsf{T}}(x-\mu_i)\right].$$

Further

$$\frac{\partial^2 g}{\partial q_i \partial \mu_j} = \frac{\partial^2 g}{\partial \mu_j \partial p} \frac{\partial p}{\partial q_i},$$

where

$$\frac{\partial^2 g}{\partial p_l \partial \mu_j} = (\delta_j(l) + \delta_j(K))(-1)^{\delta_K(j)} \phi(x, \mu_j, L_j)(x - \mu_j)^\mathsf{T} \Sigma_j^{-1}.$$

$$\frac{\partial^2 g}{\partial \vec{L}_i \partial \vec{L}_j} = \delta_i(j) \begin{pmatrix} \frac{\partial^2 g}{\partial \vec{L}_i^1 \partial \vec{L}_i} & \cdots & \frac{\partial^2 g}{\partial \vec{L}_i^{D(D+1)/2} \partial \vec{L}_i} \end{pmatrix},$$

where

$$\frac{\partial^2 g}{\partial \vec{L}_i^j \partial \vec{L}_i} = \frac{p_i \phi(x, \mu_i, L_i)}{|L_i|} \left[(\vec{M_i L_i}) \left(|L_i| (M_i L_i)^{\vec{z}_j, \vec{s}_j} - \delta_{\vec{z}_j}(\vec{s}_j) \prod_{k \neq \vec{z}_j}^{D} L_i^{k,k} \right) \right.$$
$$\left. + \delta_{\vec{z}_j}(\vec{s}_j) \text{vech}(\text{diag}(\prod_{k \neq 1, \vec{z}_j}^{D} L_i^{k,k}, \ldots, \underset{j\uparrow}{0}, \ldots, \prod_{k \neq D, \vec{z}_j}^{D} L_i^{k,k})) - \xi_j \right],$$

where ξ_j is a $\frac{D(D+1)}{2}$-vector with

$$\xi_{j,p} := \delta_{\vec{z}_p}(\vec{s}_p) \prod_{k \neq \vec{z}_p}^{D} L_i^{k,k} (M_i L_i)^{\vec{z}_j, \vec{s}_j} + \delta_{\vec{z}_j}(\vec{z}_p) |L| (x - \mu_i)^{\vec{z}_j} (x - \mu_i)^{\vec{z}_p}.$$

We are now able to calculate $\nabla_\theta^2 g(x_t, \theta)$.

10.1.4 Second derivatives of the log-likelihood

Now we have some formulas for the calculation of the Hessian w.r.t. the parameters of the mixture density. To obtain a formula for the Hessian of the log-likelihood, we need the following proposition.

Proposition 10.3 *Let $f : \mathbb{R}^D \to \mathbb{R}^D$ and $h : \mathbb{R}^D \to \mathbb{R}$ be continuously differentiable functions. Then $J_x(hf) = hJ_x(f) + f \nabla_x h^\mathsf{T}$,*

where $J_x(f)$ is the Jacobian of f w.r.t. x and $\nabla_x h$ is the gradient of h w.r.t. x.

Proof. $hf(x) = \begin{pmatrix} h(x) f_1(x) \\ \vdots \\ h(x) f_D(x) \end{pmatrix}$ and using the product rule we obtain

$$\nabla_x (h f_j) = h \nabla_x f_j + f_j \nabla_x h.$$

□

Our goal now is to calculate the Jacobian of

$$\nabla_\theta l(\theta) = \sum_{t=1}^{n} \frac{1}{g(x_t, \theta)} \nabla_\theta g(x_t, \theta).$$

We use Proposition 10.3 with $f = \nabla_\theta g$ and $h = \frac{1}{g}$ and observe that

$$\nabla_\theta \frac{1}{g(x_t,\theta)} = \frac{-1}{g(x_t,\theta)^2} \nabla_\theta g(x_t,\theta).$$

Summing over all i's we finally get

$$\nabla_\theta^2 l = \sum_{t=1}^{n} \frac{1}{g(x_t,\theta)} \left(\nabla_\theta^2 g(x_t,\theta) - \frac{1}{g(x_t,\theta)} \nabla_\theta g(x_t,\theta) \nabla_\theta^\mathsf{T} g(x_t,\theta) \right).$$

10.1.5 Derivatives of the penalty terms

The penalty function used in our algorithm is given by

$$s_n(L_1,\dots,L_K) = -a_n\big(\sum_{i=1}^{K} \operatorname{tr}(S_x L_i {L_i}^\mathsf{T}) + \log \frac{1}{|L_i|^2}\big).$$

Now we omit the index of the covariance parameter. Hence L represents any of the $L_1,\dots,L_K$. For $1 \le s \le \frac{D(D+1)}{2}$ holds

$$\frac{\partial s_n}{\partial \vec{L}^s} = -a_n\big(2\sum_{k=\vec{s}_s}^{D} (S_x)^{k,\vec{z}_s} L^{k,\vec{s}_s} - \frac{2\delta_{\vec{z}_s}(\vec{s}_s)}{\vec{L}^s}\big),$$

$$\frac{\partial^2 s_n}{\partial L_s^\Delta \partial L_t^\Delta} = -a_n\big(2\delta_{\vec{s}_s}(\vec{s}_t)(S_x)^{\delta_{\vec{z}_t}(\vec{s}_t)} + \frac{2\delta_{\vec{z}_s}(\vec{s}_s)\delta_s(t)}{(\vec{L}^s)^2}\big).$$

10.2 Derivatives of the parameter transformation ψ

We calculate the derivatives of

$$\psi(\mu_1,\dots,\mu_K,L_1,\dots,L_K,q_1,\dots,q_{K-1}) = (\mu_1,\dots,\mu_K,\Sigma_1,\dots,\Sigma_K,p_1,\dots,p_{K-1}).$$

The derivatives $\frac{\partial\psi}{\partial q}$ are given in (11). Now we calculate the partial derivatives of ψ w.r.t. L_i. We represent the matrices as vectors and calculate the derivatives of the function $\vec{L} \mapsto \operatorname{vech}(LL^{\mathsf{T}-1})$, for a lower triangular matrix L. This function can be expressed as a composition $\psi_1 \circ \psi_2$ for $\psi_1 : \vec{A} \mapsto \vec{A^{-1}}$, and $\psi_2 : \vec{L} \mapsto \operatorname{vech}(LL^\mathsf{T})$.

Proposition 10.4 *i) Let A be a regular matrix, $i \ge j$ two integers. Then*

$$\frac{\partial \psi_1}{\partial A_{ij}} = -vech(A^{-1} e_i {e_j}^\mathsf{T} A^{-1})$$

ii) Let L be a lower triangular Matrix, $i \ge j, p \ge q$ four integers. Then

$$\frac{\partial {\psi_2}_{p,q}}{\partial L_{ij}} = \delta_i(p) L_{q,j} + \delta_i(q) L_{p,j}$$

Proof. i)

$$A^{-1}A = I \Rightarrow \frac{\partial A^{-1}A}{\partial A_{ij}} = \frac{\partial A^{-1}}{\partial A_{ij}} A + A^{-1} \underbrace{\frac{\partial A}{\partial A_{ij}}}_{=e_i {e_j}^\mathsf{T}} = 0$$

$$\Leftrightarrow \frac{\partial A^{-1}}{\partial A_{ij}} = -A^{-1} \frac{\partial A}{\partial A_{ij}} A^{-1}$$

ii) It follows immediately from $(LL^\mathsf{T})_{i,j} = L_{i,\cdot}^\mathsf{T} L_{j,\cdot}$.