Communications of the Korean Statistical Society
2012, Vol. 19, No. 1, 157–168

DOI: http://dx.doi.org/10.5351/CKSS.2012.19.1.157

A Fast EM Algorithm for Gaussian Mixtures

Hyekyung Jung[a], Byungtae Seo[1,b]

[a]Samsungcard CO., LTD.; [b]Department of Statistics, Sungkyunkwan University

Abstract

The EM algorithm is the most important tool to obtain the maximum likelihood estimator in finite mixture models due to its stability and simplicity. However, its convergence rate is often slow because the conventional EM algorithm is based on a large missing data space. Several techniques have been proposed in the literature to reduce the missing data space. In this paper, we review existing methods and propose a new EM algorithm for Gaussian mixtures, which reduces the missing data space while preserving the stability of the conventional EM algorithm. The performance of the proposed method is evaluated with other existing methods via simulation studies.

Keywords: EM algorithm, ECM algorithm, constrained Newton method.

1. Introduction

For finite mixture models, the maximum likelihood estimator(MLE) does not have a closed form and requires some numerical strategy to find the MLE. Newton-type optimization algorithms can be used as a general purpose; however, they are unstable and hard to program because mixture models involve a large number of parameters and the mixture likelihood has multiple modes. The expectation-maximization(EM) (Dempster *et al.*, 1977) algorithm would be an easy and stable alternative in this case.

The EM algorithm is notoriously slow in many cases. The convergence rate of the EM algorithm depends mainly on the amount of missing information as illustrated in Dempster *et al.* (1977). The conventional EM algorithm for mixtures is constructed based on a large missing data space that involves all component membership variables. This results in a large missing data space and leads to a slow convergence. If we could characterize a given problem with a smaller missing data space we would expect a faster convergence. This model reduction technique is well studied in Meng and Rubin (1993) and Liu and Rubin (1994), who deem this type of EM algorithms the expectation-conditional-maximization(ECM) algorithm.

Some variants utilize the ECM algorithm in the mixture literature. Celeux *et al.* (2001) proposed a component-wise EM algorithm for mixtures(CEMM) that updates each set of component parameters and the corresponding mixing weight at each iteration as a tool to reduce missing data space. The CEMM is also considered a Space-Alternating Generalized EM(SAGE) algorithm (Fessler and Hero, 1994) that updates each component parameter based on the reduced missing data space and the mixing proportions with the complete data space. The SAGE algorithm for the Gaussian mixture models is derived in Celeux *et al.* (1999). Pilla and Lindsay (1996) merged several components to

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2011-0014607).

[1] Corresponding author: Assistant Professor, Department of Statistics, Sungkyunkwan University, 53 Myeogryun-dong 3-ga, Chongno-gu, Seoul 110-745, Korea. E-mail: seobt@skku.edu

reduce missing data space in each CM step, but this can only be used for the estimation of component proportions with known component densities. Liu and Sun (1997) used a different strategy to reduce missing data space called the ECME algorithm. They used the conventional EM algorithm for the parameters in the component densities; however, the mixing weights are updated directly from the observed likelihood instead of the complete likelihood. Since the ECME algorithm does not use any missing data space when it estimates mixing proportions, it is expected to be quite fast. However, the estimation of mixing proportions in the observed likelihood is often unstable, so it loses one of the great advantages of using the conventional EM algorithm in mixtures.

In this paper, we propose a stable ECME algorithm using the constrained Newton method for mixing proportions suggested by Wang (2007). The proposed method can also be accelerated by combining CEMM or SAGE algorithms without significant computing effort. This paper is organized as follows: In Section 2 and Section 3, we give a brief review of the existing algorithms including the conventional EM algorithm and the constrained Newton method. In Section 4, we illustrate the proposed algorithm by combining the SAGE with the constrained Newton method. Some simulation studies and concluding remarks are then given in Section 5 and Section 6.

2. Review of Some Existing Methods

In this section, we briefly review the conventional EM algorithm and its variants for finite mixture models. Let us consider a finite d-variate normal mixture model,

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{j=1}^{m} p_j f\left(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j\right),$$

where $\boldsymbol{\theta} = (\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_1,\dots,\boldsymbol{\Sigma}_m,p_1,\dots,p_m)$ with constraints $p_j > 0$ and $\sum p_j = 1$. Throughout this paper, we use bold faced lower and upper case letters to represent a vector and a matrix, respectively. Since typical Newton type algorithms are known to be very unstable and difficult to program, the EM algorithm will be used to obtain the MLE of $\boldsymbol{\theta}$. The EM algorithm has many good properties such as simplicity, stability, and monotone convergence. To construct the conventional EM algorithm, one needs to interpret the mixture model as a component missing problem. That is, we assume that each observation $\boldsymbol{x}_i$ comes from one of the component densities while the component membership is missing. In this case, if we define the component membership indicator z_{ij} as

$$z_{ij} = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \text{ comes from } j^{th} \text{ component,} \\ 0, & \text{elsewhere} \end{cases}$$

the joint density of $(\boldsymbol{x}_i, z_{i1},\dots,z_{im})$ can then be expressed as

$$\prod_{j=1}^{m}\left(p_j f\left(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j\right)\right)^{z_{ij}}. \tag{2.1}$$

Now, based on the observed $\boldsymbol{x}_i$ and the unobserved $z_{i1},\dots,z_{im}$, $i = 1,\dots,n$, the conventional EM algorithm to obtain the MLE of $\boldsymbol{\theta}$ is constructed as follows:

E-step: For the current estimate $\boldsymbol{\theta}^{(t)}$

$$Q\left(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}\right) = \sum_{i=1}^{n} E\left[\log\left\{\prod_{j=1}^{m}\left(p_j f\left(\boldsymbol{x}_i;\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j\right)\right)^{z_{ij}}\right\}\middle|\boldsymbol{x}_i,\boldsymbol{p}^{(t)},\boldsymbol{\mu}_j^{(t)},\boldsymbol{\Sigma}_j^{(t)}\right]$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} \hat{z}_{ij} \log p_j + \sum_{i=1}^{n} \sum_{j=1}^{m} \hat{z}_{ij} \log f\left(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right),$$

where

$$\hat{z}_{ij} = E\left(z_{ij} | \boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right) = \frac{p_j^{(t)} f\left(x_i; \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right)}{\sum_{j=1}^{m} p_j^{(t)} f\left(\boldsymbol{x}_i; \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right)}.$$

M-step:

$$\begin{aligned} p_j^{(t+1)} &= \frac{1}{n} \sum_{i=1}^{n} \hat{z}_{ij}, \\ \boldsymbol{\mu}_j^{(t+1)} &= \frac{\sum_{i=1}^{n} \hat{z}_{ij} \boldsymbol{x}_i}{n p_j^{(t+1)}}, \\ \Sigma_j^{(t+1)} &= \frac{\sum_{i=1}^{n} \hat{z}_{ij} \boldsymbol{x}_i^T \boldsymbol{x}_i}{n p_j^{(t+1)}} - \left(\boldsymbol{\mu}_j^{(t+1)}\right)^T \boldsymbol{\mu}_j^{(t+1)}. \end{aligned}$$

Note that the missing data space for this conventional EM algorithm is $(z_{i1}, \ldots, z_{im})$ for each i. The CEMM algorithm decomposes the parameter space into each set of component parameters so that we can reduce missing data space. At each iteration t, the EM algorithm is used only to update $(\boldsymbol{\mu}_{j^*}, \boldsymbol{\Sigma}_{j^*}, p_{j^*})$, where $j^* = \mathrm{mod}(t-1, m) + 1$ and $\mathrm{mod}(a, b)$ is the remainder when dividing a by b. So, in each EM step, the missing data space is reduced to (z_{ij^*}). Although it requires to visit every $j = 1, \ldots, m$ like the conventional EM, one can expect a faster convergence than the conventional EM because the CEMM requires a smaller missing data space. This also appeals to our intuition as the CEMM uses the new information as soon as it is available. One potential drawback of the CEMM is that the sum of $p_j^{(t)}$'s may not add up to one. But Celeux *et al.* (2001) showed that the sum of $p_j^{(t)}$'s should eventually be one upon its convergence.

Similar to the CEMM, the SAGE algorithm only updates $(\boldsymbol{\mu}_{j^*}, \boldsymbol{\Sigma}_{j^*})$ but with the fixed current mixing proportion estimator $\boldsymbol{p}^{(t)}$ at each iteration. After all $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$'s are updated, $\boldsymbol{p} = (p_1, \ldots, p_m)^T$ is updated simultaneously as in the conventional EM. The SAGE keeps the monotonicity while the CEMM does not guarantee the monotonicity, because the sum of the updated p_j's is not necessarily one. The main difference between the CEMM and SAGE is that the SAGE algorithm requires the complete data space when it updates $(p_1, \ldots, p_m)$ while the CEMM only requires a reduced missing data space for each update.

The ECME algorithm updates $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$'s using the conventional EM algorithm, but it updates $(p_1, \ldots, p_m)$ from the observed likelihood rather than the complete likelihood using Newton-type algorithms. For example, the new update $\boldsymbol{p}^{(t+1)}$ can be updated by iterating $\boldsymbol{p}^{(t+1)} = \boldsymbol{p}^{(t)} - H^{-1}G$ (Liu and Sun, 1997), where

$$\begin{aligned} (\boldsymbol{G})_j &= \sum_{i=1}^{n} \frac{\hat{z}_{ij} - \hat{z}_{im}}{q_i}, \\ (\boldsymbol{H})_{jk} &= -\sum_{i=1}^{n} \left(\frac{\hat{z}_{ij} - \hat{z}_{im}}{q_i}\right)\left(\frac{\hat{z}_{ik} - \hat{z}_{im}}{q_i}\right), \end{aligned}$$

and $q_i = \sum p_j^{(t)}\hat{z}_{ij}$. Since only one iteration of the Newton-Raphson method is typically required, the ECME dramatically improves the conventional EM algorithm without an extra computing cost. Celeux *et al.* (2001) showed simple simulation studies to measure the performance of ECME, CEMM, SAGE, and the conventional EM algorithm. They reported that the ECME algorithm is unstable and requires more computing time than others. We could not exactly follow their programming; however, based on our MATLAB code, the ECME requires slightly more computing time than the conventional EM for each iteration.

Another advantage of the ECME is that it requires a new coding only for mixing proportions, thus it can be transferable to the mixtures of any family of component densities and multi-dimensional mixtures. However, as Celeux *et al.* (2001) reported, the ECME often breaks down as it inherits the instability of the Newton method. In the next section, we introduce the constrained Newton method suggested by Wang (2007) that will play a core role in the stabilization of the ECME algorithm.

3. Constrained Newton Method for Multiple Support Points

To remove the instability for the estimation of the mixing proportions in the ECME algorithm, we propose to use the method suggested by Wang (2007) who develops a method to update mixing proportions for nonparametric mixture models that is called the constrained Newton method for multiple supports(CNM). The basic idea is to transform the maximization problem with known parameters of component densities into a least square estimation with constraints using a quadratic approximation of the log likelihood.

For a fixed $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$'s, the log likelihood of $\boldsymbol{p} = (p_1, \ldots, p_m)^T$ is given by

$$\ell(\boldsymbol{p}) \equiv \sum_{i=1}^{n} \log\left(\boldsymbol{p}^T\boldsymbol{f}_i\right) = \sum_{i=1}^{n} \log\left(\sum_{j=1}^{m} p_j f\left(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)\right),$$

where $f_{ij} = f\left(x_i; \mu_j, \Sigma_j\right)$ and $\boldsymbol{f}_i = (f_{i1}, \ldots, f_{im})^T$. The first and second derivatives of $\ell(\boldsymbol{p})$ is then calculated as

$$\nabla_{\boldsymbol{p}}\ell(\boldsymbol{p}) \equiv \sum_{i=1}^{n} \frac{\boldsymbol{f}_i}{\boldsymbol{p}^T\boldsymbol{f}_i} = \boldsymbol{S}^T\mathbf{1}$$

and

$$\nabla^2_{\boldsymbol{p}}\ell(\boldsymbol{p}) \equiv -\sum_{i=1}^{n} \frac{\boldsymbol{f}_i\boldsymbol{f}_i^T}{(\boldsymbol{p}^T\boldsymbol{f}_i)^2} = -\boldsymbol{S}^T\boldsymbol{S},$$

where $\mathbf{1}$ is the column vector having all elements one with length n, and $\boldsymbol{S}^T = (\boldsymbol{f}_1/(\boldsymbol{p}^T\boldsymbol{f}_1), \ldots, \boldsymbol{f}_n/(\boldsymbol{p}^T\boldsymbol{f}_n))$.

Now, using the Taylor expansion, $\ell(\boldsymbol{p})$ around the current estimate $\boldsymbol{p}^{(t)}$ can be approximated by

$$\begin{aligned}
\ell(\boldsymbol{p}) - \ell\left(\boldsymbol{p}^{(t)}\right) &\approx \left(\boldsymbol{p} - \boldsymbol{p}^{(t)}\right)\boldsymbol{S}_{(t)}^T\mathbf{1} - \frac{1}{2}\left(\boldsymbol{p} - \boldsymbol{p}^{(t)}\right)^T \boldsymbol{S}_{(t)}^T\boldsymbol{S}_{(t)}\left(\boldsymbol{p} - \boldsymbol{p}^{(t)}\right) \qquad (3.1)\\
&= -\frac{1}{2}\left(\left\|\boldsymbol{S}_{(t)}\left(\boldsymbol{p} - \boldsymbol{p}^{(t)}\right)\right\|^2 - 2\left(\boldsymbol{p} - \boldsymbol{p}^{(t)}\right)\boldsymbol{S}_{(t)}^T\mathbf{1} + \|\mathbf{1}\|^2\right) + \frac{\|\mathbf{1}\|^2}{2}\\
&= -\frac{1}{2}\left\|\boldsymbol{S}_{(t)}\left(\boldsymbol{p} - \boldsymbol{p}^{(t)}\right) - \mathbf{1}\right\|^2 + \frac{n}{2}\\
&= -\frac{1}{2}\left\|\boldsymbol{S}_{(t)}\boldsymbol{p} - \mathbf{2}\right\|^2 + \frac{n}{2},
\end{aligned}$$

where $\boldsymbol{S}_{(t)} = \boldsymbol{S}|_{\boldsymbol{p}=\boldsymbol{p}^{(t)}}$ and $\mathbf{2}$ is the column vector having all elements two with length n. Then maximizing $\ell(\boldsymbol{p})$ is approximately equivalent to solving the least square problem $||\boldsymbol{S}\boldsymbol{p} - \mathbf{2}||^2$ with constraints $\sum_j p_j = 1$ and $p_j > 0$ for all $j = 1, \ldots, m$. There are some built-in functions in R or MATLAB to solve this problem. For a detailed description, see Wang (2007)

For a given $\boldsymbol{S}$, solving this least square problem does not require much computing cost; in addition, as shown in the next section, S can be obtained as a byproduct if we apply this to the ECME or SAGE algorithm.

4. SAGE with CNM

Now we illustrate the proposed algorithm that combines the SAGE with the CNM. The SAGE algorithm only updates the parameters of each component density in the reduced missing data space. The mixing proportions are then updated under the complete data space. One drawback of the SAGE is that it requires the complete data space, hence it is not significantly faster than the conventional EM or CEMM. To improve this, we use the CNM algorithm to update mixing proportions. The AECME algorithm (Celeux *et al.*, 2001) is basically constructed with the same idea but it is very unstable because it uses the Newton algorithm; however, the CNM algorithm is stable without extra computing effort. We summarize this combined algorithm as follows:

1. For fixed $\boldsymbol{\mu}_j^{(t)}$ and $\boldsymbol{\Sigma}_j^{(t)}$, compute $n \times m$ matrix $\boldsymbol{F} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_m)$ where $\boldsymbol{f}_j = (f(x_i; \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}), \ldots, f(x_n; \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}))^T$, and set $\boldsymbol{G} = \boldsymbol{F}D(\boldsymbol{p}^{(t)})$, where $D(\boldsymbol{a})$ is the diagonal matrix with the diagonal entry $\boldsymbol{a}$.

2. Compute a column vector $\boldsymbol{s}_j = (D((\boldsymbol{G}_{\bullet+}))^{-1}\boldsymbol{f}_j$ and $z_j = p_j^{(t)}\boldsymbol{s}_j$ for each $j = 1, \ldots m$, where $\boldsymbol{G}_{\bullet+} = (\sum_j(\boldsymbol{G})_{1j}, \ldots, \sum_j(\boldsymbol{G})_{nj})$ and $(\boldsymbol{G})_{ij}$ is the $(i, j)^{th}$ element of $\boldsymbol{G}$.

3. Repeat (a) and (b) from $j = 1$ to m.

   (a) Update $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ as

   $$\boldsymbol{\mu}_j^{(t+1)} = z_j^T \boldsymbol{X},$$

   and

   $$\Sigma_j^{(t+1)} = \frac{\boldsymbol{X}^T D\left(z_j\right) \boldsymbol{X}}{1^T z_j} - \left(\boldsymbol{\mu}_j^{(t+1)}\right)^T \left(\boldsymbol{\mu}_j^{(t+1)}\right),$$

   where $\boldsymbol{X} = (\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_n^T)^T$.

   (b) Based on $\boldsymbol{\mu}_j^{(t+1)}$ and $\boldsymbol{\Sigma}_j^{(t+1)}$, update $\boldsymbol{f}_j$, the j^{th} column of $\boldsymbol{G}$, $\boldsymbol{s}_j$, and z_j in turn.

4. Construct $n \times m$ matrix $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m)$ and apply the one-step CNM method to update $\boldsymbol{p}^{(t+1)}$ and go to Step 3 until it meets some stopping criteria.

At a glance, this algorithm seems different from the conventional EM algorithm. However, Step 1 and 2 are just a reformulation of the E-step in Section 2 so that we can handle each component separately. This also allows us to immediately use the updated information obtained at j^{th} component for the update of $(j+1)^{th}$ component without further computation. Another advantage of this is that the matrix $\boldsymbol{S}$ can be automatically obtained at the end of Step 3 as a byproduct. Note that the conventional

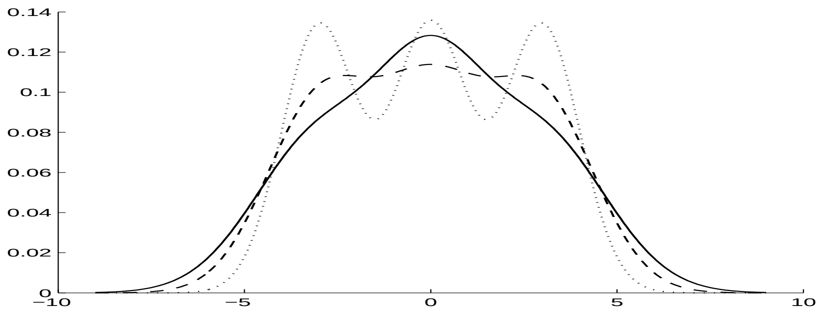


Figure 1: *Density plots for (a)· · · , (b)- -, (c) —*

EM for mixtures requires the recomputation of $f(\boldsymbol{x}_i; \boldsymbol{\mu}_j^{(t+1)}, \boldsymbol{\Sigma}_j^{(t+1)})$ for each i and j at each iteration and the overall computing time is greatly dominated by this computation. In the proposed algorithm, one can see that the number of computations for $f(\boldsymbol{x}_i; \boldsymbol{\mu}_j^{(t+1)}, \boldsymbol{\Sigma}_j^{(t+1)})$ remains the same as the conventional EM, so we can expect that this does not significantly increase the computing time for a single iteration.

5. Simulation Studies

In this section, we compare the proposed method with other existing methods. For simplicity, we consider the following 3-component univariate normal mixture models:

$$\text{(a)}\quad \frac{1}{3}N(-3,1) + \frac{1}{3}N(0,1) + \frac{1}{3}N(3,1),$$

$$\text{(b)}\quad \frac{1}{3}N(-3,2) + \frac{1}{3}N(0,2) + \frac{1}{3}N(3,2),$$

$$\text{(c)}\quad \frac{1}{3}N(-3,3) + \frac{1}{3}N(0,2) + \frac{1}{3}N(3,3).$$

Model (a), (b), and (c) represent well, intermediately, and poorly separated mixture densities, respectively. The shapes of these densities for (a), (b), and (c) are given in Figure 1. For our simulation experiment, we generate $n = 500$ random samples from each normal mixture model.

When we summarize the results from any type of the EM algorithms for mixture models, we should consider several irregular features of the mixture likelihood. First, since the mixture likelihood has multiple modes, we may need multiple initial values for parameters to ensure the set of all convergent parameter estimates contains the global maximizer. However, there is no guarantee whether or not we have the global maximizer within the set of solutions we have. To avoid this issue in our simulation, we choose the true parameter value as an initial value. We refer to this as a 'good' initial value. Since choosing such a good initial value is unrealistic in practice, we also choose a 'bad' initial value as

$$(\mu_1, \mu_2, \mu_3) = (0, 0.5, 1), \left(\sigma_1^2, \sigma_2^2, \sigma_3^2\right) = (1, 1, 1), (p_1, p_2, p_3) = (0.1, 0.8, 0.1)$$

which is far from the true parameter values. By doing so, we only measure the speed of each algorithm without considering if they converge to the global maximum.

Second, although SAGE, the conventional EM, and the proposed algorithms always increase the likelihood, they might converge to different local modes even with the same initial value. In this case, we also need to consider whether they are singular or spurious likelihood solutions. It is well

Table 1: Average ratios of required time to converge for each algorithm to the conventional EM algorithm. The number in the parenthesis represents the cases where the used algorithm shows the best performance in 100 replications.

Quality of Initial values	Algorithm	Model (a)	(b)	(c)
good	CEMM	0.7796(14)	0.7864(5)	0.8544(1)
	SAGE	0.7903(1)	0.7882(1)	0.8534(3)
	NEW	0.7105(85)	0.5719(94)	0.5906(86)
bad	CEMM	1.0636(21)	1.2471(38)	1.1898(37)
	SAGE	0.8541(2)	0.9040(12)	0.8907(6)
	NEW	0.7050(74)	0.6510(46)	0.6232(55)

known that the normal mixture likelihood is unbounded and the global maximizer always occurs on the boundary of the parameter space. These global maximizers are often called a singular solution; in addition, there are also cases where the parameter estimate is located near the boundary of the parameter space but not exactly on the boundary.

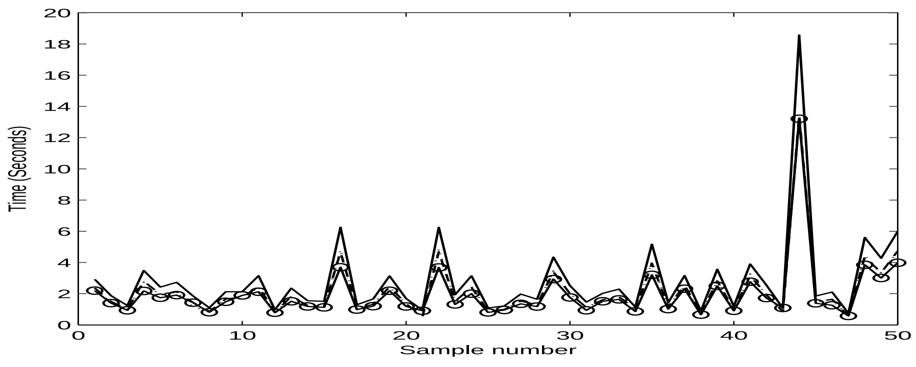
In computational point of view, when EM-type algorithms move toward such singular or spurious solutions, it converges to those likelihood solutions very quickly. Hence, if one of the considered algorithms converges to such a (nearly) singular solution, its computing time would be very small compared to other considered methods. In this case, the summary for the time or number of iterations for each algorithm may give misleading interpretation for comparison purposes. For these reasons, we summarize the cases where all tested algorithms produce the same answer. That is, for a given simulated sample, if any of the considered algorithms produces a different answer, we ignore the sample and redraw a new sample until all algorithms give the same answer. With this simulation scheme, we measure the time and number of iterations until the difference of two consecutive log-likelihood values is less than 10^{-7} for each algorithm.

Table 1 shows the average ratio of convergent times for CEMM, SAGE, and the proposed algorithm compared to the conventional EM algorithm based on 100 replications. Note that the number in the parenthesis stands for the cases in which the corresponding algorithm gives the shortest computing time among all tested algorithms. Figure 2 and Figure 3 show the times to converge with the good and bad initial values for the first 50 samples, respectively. For the well separated mixture, CEMM, SAGE, and the proposed method, all reduce the computing time by 10–30%, compared to the conventional EM. Under the poorly separated mixture, the proposed algorithm shows great time reduction though CEMM and SAGE also show some improvement compared to the conventional EM. There is also a significant improvement for the proposed method when the true initial value is used for the proposed method. This is because the likelihood at a good initial value is approximated reasonably well by the quadratic function used in (3.1).

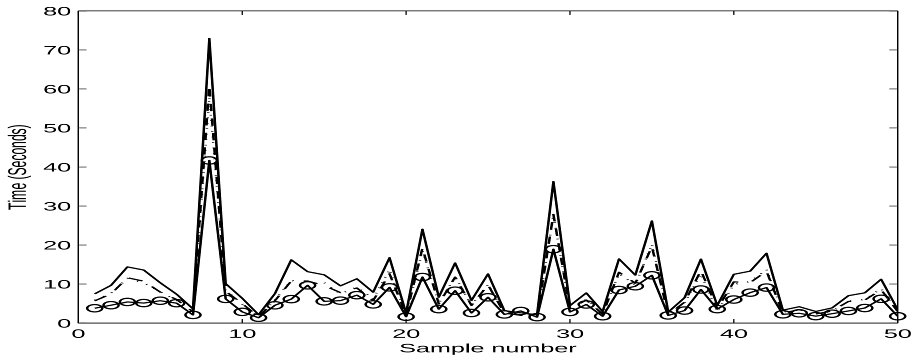
For all scenarios, all algorithms show some improvement compared to the conventional EM but the proposed method shows the best performance in terms of the time and the number of best performance cases. This improvement is more evident when we use a good initial value or the underlying model is not well-separated. We also measured the required time for a single EM iteration for each algorithm and found that the time difference for all algorithms considered here does not exceed by more than 2%.

6. Concluding Remarks

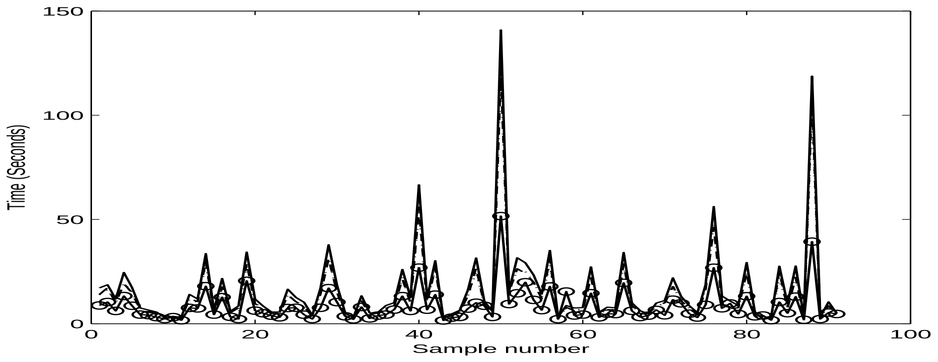
To speed up the conventional EM algorithm for mixtures, we proposed a new algorithm which combined the SAGE algorithm with the constrained Newton method. In some sense, the proposed al-



(a) $\frac{1}{3}N(-3,1)+\frac{1}{3}N(0,1)+\frac{1}{3}N(3,1)$



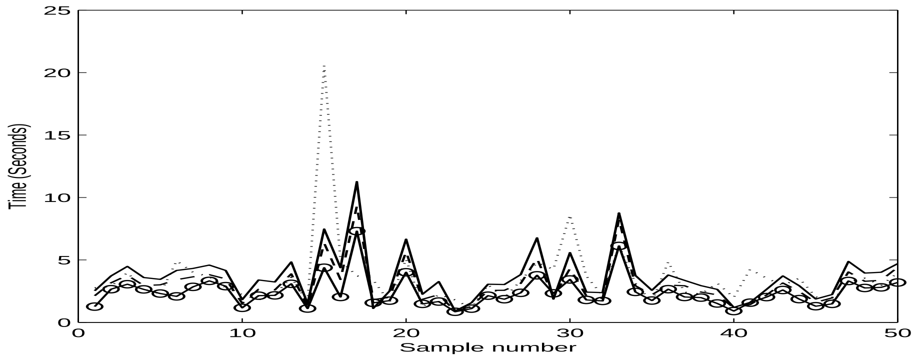
(b) $\frac{1}{3}N(-3,2)+\frac{1}{3}N(0,2)+\frac{1}{3}N(3,2)$



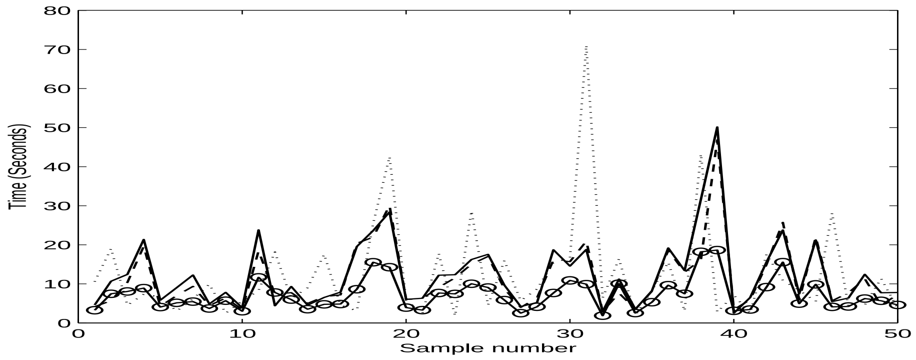
(c) $\frac{1}{3}N(-3,3)+\frac{1}{3}N(0,2)+\frac{1}{3}N(3,3)$

Figure 2: *Times to converge with the good initial value (1) —: Conventional EM, (2) $\cdots$: CEMM, (3) - -: SAGE, (4) —○: New algorithm*

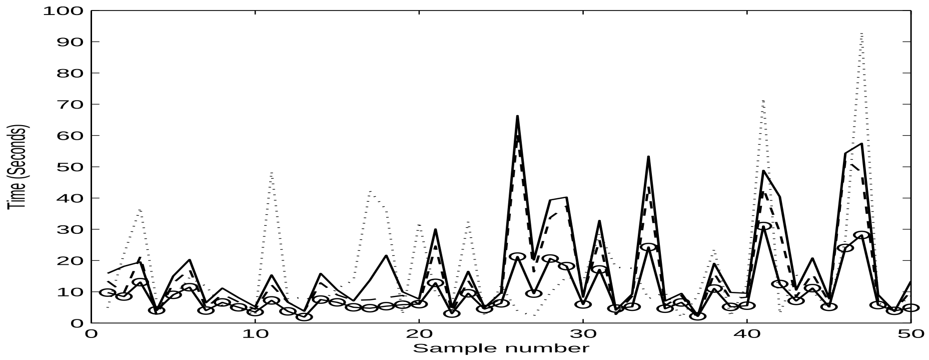
gorithm can be considered a component-wise EM algorithm, because we sequentially update each component parameters; however, it is not exactly a component-wise algorithm as the component mix-



(a) $\frac{1}{3}N(-3,1) + \frac{1}{3}N(0,1) + \frac{1}{3}N(3,1)$



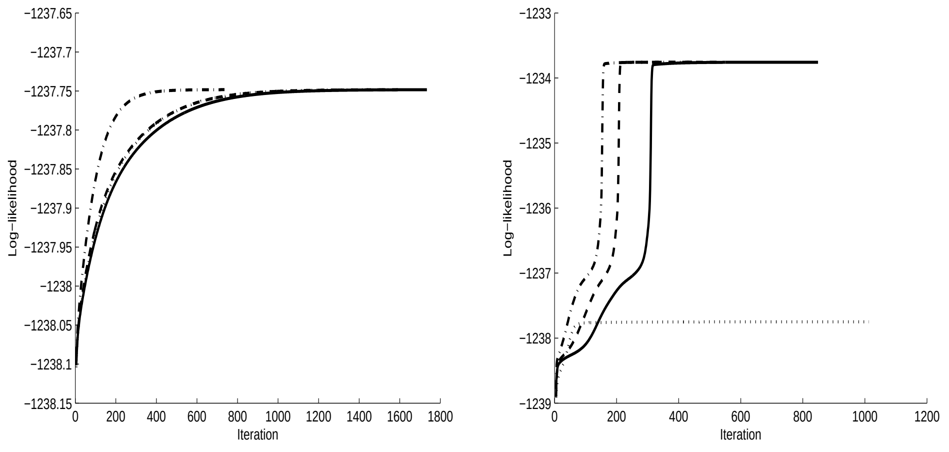
(b) $\frac{1}{3}N(-3,2) + \frac{1}{3}N(0,2) + \frac{1}{3}N(3,2)$



(c) $\frac{1}{3}N(-3,3) + \frac{1}{3}N(0,2) + \frac{1}{3}N(3,3)$

Figure 3: *Times to converge with the bad initial value (1) —: Conventional EM, (2) · · · : CEMM, (3) - - : SAGE, (4) —○: New algorithm*

ing proportions are updated simultaneously. Since it is obvious that the proposed algorithm requires a smaller missing data space than the CEMM, SAGE, and conventional EM algorithm, it is natural to



(a) good initial value (b) bad initial value

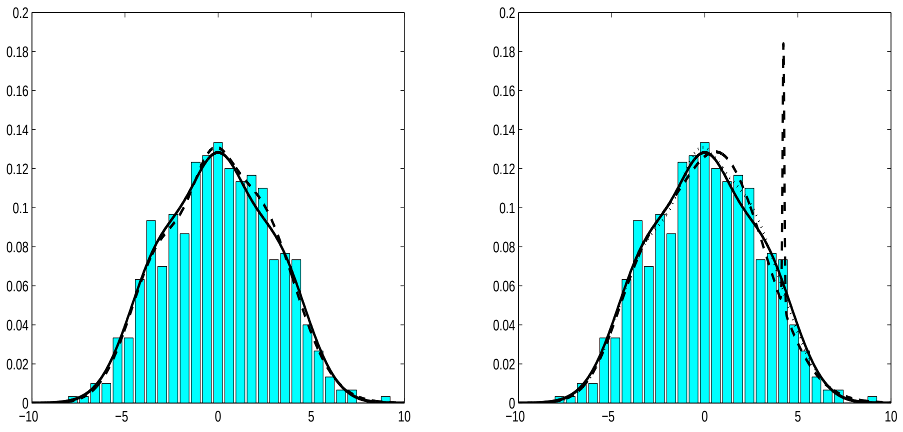
Figure 4: *Number of iterations versus log-likelihood values for Conventional EM (—), CEMM (· · ·), SAGE (- - -), and New algorithm (− · −) when the good initial value (a) and the bad initial value (b) are used*

expect a faster convergence.

Since the ECME or AECME uses a similar reduction technique, it would also be interesting to compare the proposed method with them. Indeed, we did some simulation experiments with the ECME and AECME though it is not reported here. From those simulations, we found that their performance is almost the same as the proposed method when they converge without failure. However, they are quite unstable, especially when we use a bad initial value or poorly separated mixture model. This instability would increase for mixtures with a large number of components.

It might be of great interest whether the proposed algorithm has a greater ability to find the global maximizer or a local solution with a high likelihood value. Although we ignored the cases where all algorithms do not give the same answer in our simulation, we also investigated such cases. The frequency where such suboptimal results occur is rare when a good initial value is used or the sample is drawn from a well separated mixture model. It is not obvious which algorithm tends to find a solution with a higher likelihood; however, finding the global maximum is not always preferable as we know the theoretical global maximizer is meaningless. Even after we exclude such situations, a greater ability to find a local solution with a high likelihood could imply a higher chance to find unwanted solutions due to nearly singular solutions. Hence, the ability to find a local mode with a higher likelihood is not always a desirable property for the given algorithm in the mixture likelihood.

To explain this further, we choose one of replications where one of algorithms converges to a different likelihood mode when the data are generated from the poorly separated mixture density in Section 5. Figure 4 shows the log likelihood values over iterations for each algorithm when we use the good and bad initial values used in Section 5. When the good initial value is used, all algorithms converge to the same parameter value and the corresponding fitted density given in Figure 5 (a) is close to the true density; however, when the bad initial value is used, CEMM converges to a suboptimal mode and other algorithms converges to a mode with a higher likelihood value. Note that CEMM converges to the same parameter value that we obtained from the good initial value. In this case, one may want to choose the likelihood solution obtained from EM, SAGE, or the new algorithm instead of that from CEMM, because it has a higher likelihood value than that of CEMM (See Figure 4 (b)). However,



(a) Fitted densities using the good initial value (b) Fitted densities using the bad initial value

Figure 5: *(a) Fitted densities using the good initial value ("—" and " - - -" represent the true density and fitted density from all algorithms, respectively), (b) Fitted densities using the bad initial value ("—", " $\cdots$ " and " - - -" represent the true density, fitted density from CEMM, and fitted density from other algorithms, respectively)*

as we can see in Figure 5 (b), the likelihood solution obtained from EM/SAGE/new algorithm does not seem reasonable. This nearly singular solution is called a *spurious solution* (McLachlan and Peel, 2000) and EM-type algorithms quickly converge when they climb the likelihood toward singular or spurious solutions (Biernacki and Chrétien, 2003). This spurious solution can occur with any optimization algorithm including EM-type algorithms we considered here, and it is difficult to judge which solution is optimal in general.

References

Biernacki, C. and Chrétien, S. (2003). Degeneracy in the maximum likelihood estimation of univariate gaussian mixtures with EM, *Statistics and Probability Letters*, **61**, 373–382.

Celeux, G., Chretien, S., Forbes, F. and Mkhadri, A. (1999). A component-wise EM algorithm for mixtures, Technical Report, Inria 3746, (http://www.inria.fr/RRRT/publications-fra.html).

Celeux, G., Chretien, S., Forbes, F. and Mkhadri, A. (2001). A component-wise EM algorithm for mixtures, *Journal of Computational and Graphical Statistics*, **10**, 697–712.

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

Fessler, J. A. and Hero, A. O. (1994). Space-alternating generalized expectation-maximization algorithm, *IEEE Transactions on Signal Processing*, **42**, 2664–2677.

Liu, C. and Rubin, D. B. (1994). The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence, *Biometrika*, **81**, 633–648.

Liu, C. and Sun, D. X. (1997). Acceleration of EM algorithm for mixture models using ECME, *Proceedings of the Statistical Computing Section*, Alexandria, VA: ASA,, 109–114.

McLachlan, G. J. and Peel, D. (2000). *Finite Mixture Models*, Hohn Wiley and Sons Ltd., New York.

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

Pilla, R. S. and Lindsay, B. G. (1996). Alternative EM methods for nonparametric finite mixture models, *Biometrika*, **88**, 535–550.

Wang, Y. (2007). On fast computation of the non-parametric maximum likelihood estimate of a mixing distribution, *Journal of the Royal Statistical Society, Series B*, **69**, 185–198.

Received October 2011; Revised November 2011; Accepted November 2011