Notes on methods based on maximum-likelihood estimation for learning the parameters of the mixture of Gaussians model

Luis E. Ortiz* Leslie Pack Kaelbling†

Computer Science Department, Box 1910
Brown University, Providence, RI 02912 USA
{leo,lpk}@cs.brown.edu

February 28, 1999

Abstract

In these notes, we present and review different methods based on maximum-likelihood estimation for learning the parameters of the mixture-of-Gaussians model. We describe a method based on the *likelihood equations*, traditional *gradient-based methods* (among them *steepest ascent* and *gradient ascent*), *expectation-maximization (EM)*, *conjugate gradient* methods, and proposed *accelerations of EM* (among them *Aitken acceleration* or *parameterized EM*, and *conjugate-gradient acceleration of EM*). We describe all the methods in the context of the mixture-of-Gaussians model. We relate the methods presented through the generalized gradient-based formulation. None of the methods presented are theoretically dominant. We empirically analyze their performance on synthetic datasets and suggest cases in which particular methods perform better than others.

1 Introduction

The problem of classification in unsupervised learning is of great interest to the Artificial Intelligence (AI) community. In the classification problem in unsupervised learning, we would like to learn a classification into M known classes from unclassified data. Assuming a probabilistic model for classification, Maximum Likelihood (ML) estimation provides a statistical foundation for learning the parameters of that model. Methods for finding ML estimates (MLE) for the parameters of a model have become standard in the AI community.

* This material is based upon work supported in part under a National Science Foundation Graduate Fellowship and by DARPA/Rome Labs Planning Initiative grant F30602-95-1-0020.

† This work was supported in part by DARPA/Rome Labs Planning Initiative grant F30602-95-1-0020.

The utility of the ML methods for classification results from assuming a probabilistic model and therefore expressing the problem as one of ML estimation.

ML methods have long being used in the Applied Math community in many different domains. Although the utility of the ML methods to the AI community have been established, there has not been a careful comparison of these methods in the context of the AI community. In this paper we attempt to compare these methods, concentrating primarily on comparing their speed of convergence in log likelihood space for the problem of learning the parameters of the *mixture of Gaussian model* from data. In our case, this model results from looking at the data as vectors in $\Re^d$, and assuming that the true process generating the data is a mixture of M Gaussians distributions. Although theoretical results on the speed of convergence for the different methods exist, the amount of computation per iteration on which they are based differs and this makes those results uninformative in practice. Therefore, we try to compare the methods taking into account the amount of computation they perform per iteration.

Particularly, the problem of learning the parameters of the mixture-of-Gaussians model has been widely studied and different methods have been proposed [RW84]. As motivated above, the methods we present in this paper are all based on *maximum-likelihood estimates* (MLE). The organization of the paper is as follows. We first describe the mixture-of-Gaussians model and explains what MLE means with respect to learning the parameters of this model. We then present a system of equations called the *likelihood equations* whose solution satisfies a necessary condition for a MLE for the parameters of this model. We present some theoretical results about the solution to the likelihood equations and briefly present general intuitions about those results. We then present an iterative method proposed to solve the system of equations that follows naturally from a re-expression of the likelihood equations. We then present gradient methods used in nonlinear optimization as alternative methods for finding MLE for the parameters of a model. We present some general convergence characteristics of the gradient methods. We then present what has become the most typical way to find the MLE solution for this problem: *Expectation-Maximization (EM)* [DLR77]. We relate this method to the gradient methods, present some convergence properties, and compared those properties to that of gradient methods. We then present conjugate gradient methods and briefly discuss some of their properties. We present some acceleration techniques proposed to speedup the convergence of EM. Finally, we present empirical results testing some of the proposed accelerations proposed for EM. We discuss our results, present future work and conclude.

2 Mixture of Gaussians model

In the mixture-of-Gaussians model we assume that each of the independently randomly sampled d-dimensional points in our data set comes from one of M d-dimensional Gaussians with some probability. Figure 1 shows an example of a plot of sampled data from five different two-dimensional Gaussians with equal probability. The parameters $\boldsymbol{\theta}$ of the model are, for each Gaussian j, the probability of generating a point from it, α_j, its mean, $\boldsymbol{\mu}_j$, and its covariance matrix, $\boldsymbol{\Sigma}_j$. More formally, let $\boldsymbol{\alpha}$ be a vector from the set $\boldsymbol{\mathcal{A}}$ of all possible

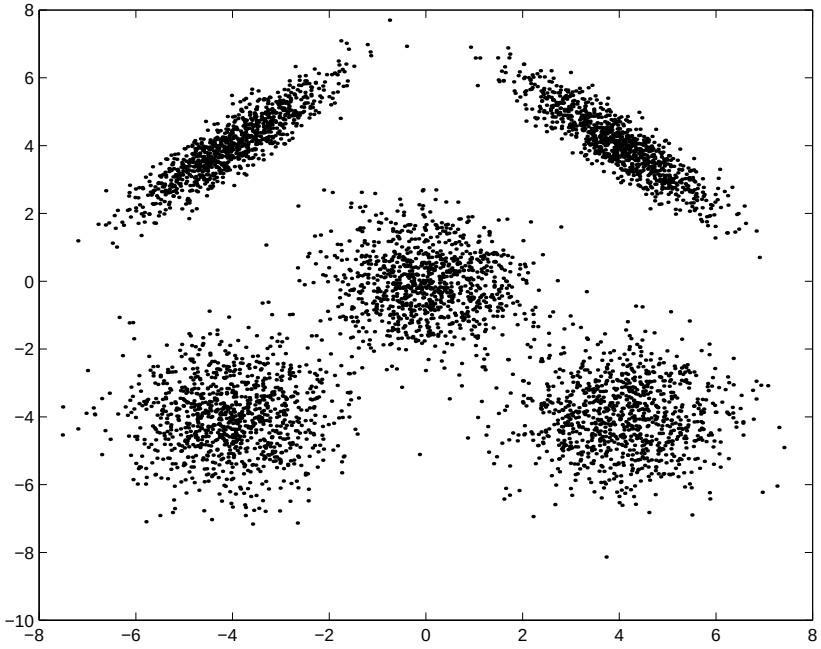


Figure 1: Plot of 5000 sampled data points from five two-dimensional Gaussians. All of them have the same probability (i.e., $\alpha_j = 0.2$ for $1 \leq j \leq M$).

probability distributions over M disjoint events and represented as vectors,

$$\boldsymbol{\mathcal{A}} = \left\{ \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_M \end{pmatrix} \middle| 1 \leq j \leq M, \alpha_j \geq 0 \text{ and } \sum_{j=1}^{M} \alpha_j = 1 \right\}.$$

Let α_j denote the j^{th} element of vector $\boldsymbol{\alpha}$. Let the vector $\boldsymbol{\mu}_j$ be a d-dimensional vector, and the matrix $\boldsymbol{\Sigma}_j$ be a $(d \times d)$ real, symmetric and positive definite matrix for $1 \leq j \leq M$. Then we can define the parameters $\boldsymbol{\theta}$ of the model as

$$\boldsymbol{\theta} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_M \\ \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_M \\ \boldsymbol{\Sigma}_1 \\ \vdots \\ \boldsymbol{\Sigma}_M \end{pmatrix}. \tag{1}$$

Although we do not generally refer to the parameters as a vector, note that if we expand the vectors and the matrices by their respective elements, the parameters do form a vector with

elements equal to particular scalar parameters that form the more general vector and matrix parameters. Therefore, those times when we use the parameters as a vector will become clear from the context.

The probability associated with each d-dimensional data point $\boldsymbol{x}_i$ given the parameters $\boldsymbol{\theta}$ of the model is

$$p(\boldsymbol{x}_i \mid \boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j). \tag{2}$$

The density $g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ is the value of the Gaussian density function with mean $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$ evaluated at $\boldsymbol{x}_i$. In other words, if we let $\mid \boldsymbol{\Sigma}_j \mid$ denote the determinant of matrix $\boldsymbol{\Sigma}_j$, then this density function is

$$g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{(2\pi)^{d/2}\sqrt{\mid \boldsymbol{\Sigma}_j \mid}} e^{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)}. \tag{3}$$

We sometimes refer the elements of the vector $\boldsymbol{\alpha}$ as the *mixture proportions* or *mixture parameters* and the densities $g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ for all j the *mixture components* or the *Gaussian components*. The mixture of Gaussians model is a mixture density model where the component densities are multivariate Gaussians [DH73].

The likelihood of the parameters $\boldsymbol{\theta}$ with respect to a set $\boldsymbol{D}$ of N independent and randomly sampled data points is

$$p(\boldsymbol{D} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{x}_i \mid \boldsymbol{\theta}). \tag{4}$$

The last equation defines a function on $\boldsymbol{\theta}$ which we call the *likelihood function*.

Next, we will present another form of the likelihood function that is going to be important later. Note that if, along with our data points we also obtain data with information about which of the M Gaussians generated each point in our dataset then we can express the likelihood accordingly. That is, if we also obtain a dataset $\boldsymbol{Z}$ of N M-dimensional indicator vectors $\boldsymbol{z}_i$ with elements $z_{i,j}$ for $1 \leq j \leq M$, then

$$p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left(\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right)^{z_{i,j}} \tag{5}$$

where we have constrained α_j further such that $\alpha_j > 0$ for all j. We can view the elements $z_{i,j}$ as *indicators* stating whether Gaussian j generated the data point $\boldsymbol{x}_i$. The reader can verify that $p(\boldsymbol{D} \mid \boldsymbol{\theta}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta})$ as expected, where the sum is over *all possible datasets* $\boldsymbol{Z}$.

Traditionally, the problem of finding MLE for the parameters of a probabilistic model is to find the value of the parameters that maximize the likelihood function. However, for this model, this problem is impossible to solve since the likelihood function has no global maximum [PW78, DH73, Vap95]. We can see an example of this in one dimension. By assigning

one of the data points to be the mean of one of the Gaussians, call it the k^{th} Gaussian, fixing the value of all the other parameters in the model except the variance associated with that Gaussian, as we let the value of the variance associated with that Gaussian approach zero, the value of the likelihood function increases arbitrarily. We call solutions of this kind *singular solutions* or *singularities*. Intuitively, those solutions correspond to collapsing one of the Gaussians in the mixture. Trying to avoid those solutions, what we really want is to find the value at which the likelihood function attains its largest local maximum. This is, however, hard in general, since the likelihood function associated with this problem has many local maxima. Although there are ways to handle the problem of many local maxima, in practice what we do is to find a local maximum of the likelihood function and hope this is the largest of them all.

Another problem with the idea of maximum likelihood estimation is the fact that the likelihood can have more than one largest local maximum. One reason is that we can exchange the parameters associated with one Gaussian, including the mixture proportion associated with it, with the parameters of another Gaussian and obtain another model with the same probability density function. Another reason is that the model that generated the data have two or more Gaussian components with the same mean and covariance matrix. For instance, if the parameters are such that model has two Gaussian components with the same mean and covariance matrix, then any assignment to the mixture proportions that respect the constraints yields a model with the same probability density function. This problem is not really a concern to us since all we are really interested is in finding a model that best fits the data and for this purpose any solution in the set of largest local maxima leads to equivalent models.

Finally, it is important to note that we do not maximize the likelihood function directly, but the logarithm of the likelihood function or the *log likelihood function*. The log likelihood function is

$$L(\boldsymbol{\theta}) = \ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \ln p(\boldsymbol{x}_i \mid \boldsymbol{\theta}), \tag{6}$$

where the last equality follows from the property of the logarithm function. This is primarily done for simplicity. The reader can verify that, because of the monotonicity of the logarithm function, the likelihood and the log likelihood function attain maxima at the same points in the domain.

3 Likelihood equations

One of the first MLE methods for this problem resulted from the notion of *likelihood equations* [PW78, RW84]. The likelihood equations come from a necessary condition for a local maximum of the log likelihood function $L(\boldsymbol{\theta}) = \ln p(\boldsymbol{D} \mid \boldsymbol{\theta})$; that is (see Appendix A),

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \mathbf{0}.$$

Next, we will compute the gradient of the log-likelihood function with respect to each of the parameters and set them to zero in order to obtain more specific necessary conditions on

the parameters for a maximum. Doing this for each α_j (taking care of the constraints on $\boldsymbol{\alpha}$ using a *Lagrange multiplier* λ), we get

$$\begin{aligned}\frac{\partial L(\boldsymbol{\theta})}{\partial \alpha_j} &= \left(\sum_{i=1}^{N} \frac{\partial \ln p(\boldsymbol{x}_i \mid \boldsymbol{\theta})}{\partial \alpha_j}\right) - \lambda \frac{\partial}{\partial \alpha_j}\left(\sum_{j=1}^{M} \alpha_j - 1\right) \\ &= \sum_{i=1}^{N} \frac{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} - \lambda \\ &= 0.\end{aligned}$$

In order to solve for λ we multiply both sides of the last equation by α_j and sum both sides for all M Gaussians,

$$\sum_{i=1}^{N}\sum_{j=1}^{M} \alpha_j \frac{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} - \lambda \sum_{j=1}^{M} \alpha_j = 0.$$

Solving that last equation for λ we obtain

$$\lambda = N.$$

Performing the same operations as above for all M Gaussians and substituting back our value for λ we get the first set of likelihood equations,

$$N = \sum_{i=1}^{N} \frac{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})}, \text{ for } 1 \leq j \leq M,$$

which we will write in a more convenient form

$$\alpha_j = \frac{1}{N} \sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})}, \text{ for } 1 \leq j \leq M. \tag{7}$$

Similarly for $\boldsymbol{\mu}_j$ we have

$$\begin{aligned}\nabla_{\boldsymbol{\mu}_j} L(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \nabla_{\boldsymbol{\mu}_j} \ln p(\boldsymbol{x}_i \mid \boldsymbol{\theta}) \\ &= \sum_{i=1}^{N} \frac{\alpha_j}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} \nabla_{\boldsymbol{\mu}_j} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ &= \sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} \nabla_{\boldsymbol{\mu}_j} \left(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)\right) \\ &= \sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j) \\ &= \mathbf{0}.\end{aligned}$$

For convenience, expressing the last equation recursively in terms of $\boldsymbol{\mu}$ and performing the same operations for all M Gaussians we get the second set of likelihood equations,

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i|\boldsymbol{\theta})}\boldsymbol{x}_i}{\sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i|\boldsymbol{\theta})}}, \text{for } 1 \leq j \leq M. \tag{8}$$

Finally, computing the gradient with respect to each $\boldsymbol{\Sigma}_j$, we get

$$\begin{aligned}
\nabla_{\boldsymbol{\Sigma}_j} L(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \nabla_{\boldsymbol{\Sigma}_j} \ln p(\boldsymbol{x}_i \mid \boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} \frac{\alpha_j}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} \nabla_{\boldsymbol{\Sigma}_j} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j).
\end{aligned}$$

$$\begin{aligned}
\nabla_{\boldsymbol{\Sigma}_j} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) &= \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)} \nabla_{\boldsymbol{\Sigma}_j} \left(\frac{1}{\sqrt{\mid \boldsymbol{\Sigma}_j \mid}} \right) + \\
&\quad \frac{1}{(2\pi)^{d/2}\sqrt{\mid \boldsymbol{\Sigma}_j \mid}} \nabla_{\boldsymbol{\Sigma}_j} \left(e^{-\frac{1}{2}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)} \right) \\
&= g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \left(-\frac{1}{2 \mid \boldsymbol{\Sigma}_j \mid} \nabla_{\boldsymbol{\Sigma}_j} (\mid \boldsymbol{\Sigma}_j \mid) + \right. \\
&\quad \left. \nabla_{\boldsymbol{\Sigma}_j} \left(-\frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j) \right) \right) \\
&= g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \left(-\frac{1}{2 \mid \boldsymbol{\Sigma}_j \mid} \mid \boldsymbol{\Sigma}_j \mid \text{vec}\left(\boldsymbol{\Sigma}_j^{-1}\right) + \right. \\
&\quad \left. \frac{1}{2} \text{vec}\left(\boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} \right) \right) \\
&= -\frac{1}{2} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \text{vec}\left(\boldsymbol{\Sigma}_j^{-1} \left(\boldsymbol{\Sigma}_j - (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \right) \boldsymbol{\Sigma}_j^{-1} \right).
\end{aligned}$$

$$\begin{aligned}
\nabla_{\boldsymbol{\Sigma}_j} L(\boldsymbol{\theta}) &= -\frac{1}{2} \text{vec} \left(\boldsymbol{\Sigma}_j^{-1} \sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} \left(\boldsymbol{\Sigma}_j - (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \right) \boldsymbol{\Sigma}_j^{-1} \right) \\
&= \mathbf{0}.
\end{aligned}$$

For convenience, expressing the last equation recursively in terms of $\boldsymbol{\Sigma}_j$ and doing the same operation for all M Gaussians we obtain the last set of equations

$$\boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T}{\sum_{i=1}^{N} \frac{\alpha_j g(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{p(\boldsymbol{x}_i|\boldsymbol{\theta})}}, \text{for} 1 \leq j \leq M. \tag{9}$$

The equations above are the likelihood equations. They form a system of nonlinear equations and their solution

$$\boldsymbol{\theta}^N = \begin{pmatrix} \alpha_1^N \\ \vdots \\ \alpha_M^N \\ \boldsymbol{\mu}_1^N \\ \vdots \\ \boldsymbol{\mu}_M^N \\ \boldsymbol{\Sigma}_1^N \\ \vdots \\ \boldsymbol{\Sigma}_M^N \end{pmatrix}$$

is a critical point of the log-likelihood function and therefore we need to evaluate the *Hessian matrix* (see Appendix B) of the log likelihood at that point to further characterize the solution. Note that we use the superscript to denote that the solution depends on the data and therefore on N. One important result about a solution $\boldsymbol{\theta}^N$ to this system of equations is that if some regularity conditions on the log likelihood function hold and the *Fisher information matrix* $\boldsymbol{F}(\boldsymbol{\theta})$ (see Appendix C) is well defined and positive definite, and we are in a sufficiently small neighborhood of the true values $\boldsymbol{\theta}^*$, then for a sufficiently large amount of data N there is a unique solution $\boldsymbol{\theta}^N$ of the likelihood equations in the neighborhood of $\boldsymbol{\theta}^*$ and this solution locally maximizes the log likelihood function. In addition, the random variable $\sqrt{N}(\boldsymbol{\Theta}^N - \boldsymbol{\theta}^*)$ is asymptotically normally distributed with mean zero and covariance matrix $\boldsymbol{F}(\boldsymbol{\theta}^*)^{-1}$ [RW84, Mei89]. This tells us that the variance on the solution we obtain from the likelihood equations, for a fixed amount of data and with respect to the true values, depends on the inverse of the Fisher information matrix. [RW84] empirically show that the diagonal elements of $\boldsymbol{F}(\boldsymbol{\theta}^*)^{-1}$ are much larger when the Gaussians in the true model are not well separated than when they are well separated. Therefore, in the case that the Gaussians are not well separated in the true model, we need a significantly larger amount of data than when the Gaussians are well separated in order to reduce the variance and obtain more precise estimates for the true parameters.

The last result regarding the asymptotically normal distribution of the solution is a very general result about the (strongly consistent maximum) solution of likelihood equations [Cha54]. Next we will present a very informal argument of why the solution to the likelihood equations is asymptotically normally distributed. Assuming that N is sufficiently large and that in the neighborhood of the $\boldsymbol{\theta}^*$ where the solution to the likelihood equations, $\boldsymbol{\theta}^N$, lies, the function is (approximately) quadratic (i.e., the partial derivatives of greater than the third order are 0 or negligible if the function is only approximately quadratic), a Taylor expansion of the gradient of the log likelihood yields

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*) + \boldsymbol{H}(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*).$$

Since the gradient evaluated at the solution $\boldsymbol{\theta}^N$ is $\mathbf{0}$, we get

$$\mathbf{0} = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*) + \boldsymbol{H}(\boldsymbol{\theta}^*)(\boldsymbol{\theta}^N - \boldsymbol{\theta}^*),$$

or equivalently

$$-\boldsymbol{H}(\boldsymbol{\theta}^*)(\boldsymbol{\theta}^N - \boldsymbol{\theta}^*) = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*).$$

Then we divide each side of the last equation by N to get

$$-\frac{1}{N}\boldsymbol{H}(\boldsymbol{\theta}^*)(\boldsymbol{\theta}^N - \boldsymbol{\theta}^*) = \frac{1}{N}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*).$$

Noting that as N increases, $-\frac{1}{N}\boldsymbol{H}(\boldsymbol{\theta}^*) \rightarrow \boldsymbol{F}(\boldsymbol{\theta}^*)$, we get

$$\boldsymbol{F}(\boldsymbol{\theta}^*)(\boldsymbol{\theta}^N - \boldsymbol{\theta}^*) = \frac{1}{N}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*) \tag{10}$$

Now we note that,

$$E_{\boldsymbol{X}}\left[\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{X} \mid \boldsymbol{\theta}^*)\right] = \mathbf{0},$$

and

$$\begin{aligned} Var_{\boldsymbol{X}}\left[\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{X} \mid \boldsymbol{\theta}^*)\right] &= E_{\boldsymbol{X}}\left[\left(\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{X} \mid \boldsymbol{\theta}^*)\right)\left(\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{X} \mid \boldsymbol{\theta}^*)\right)^T\right] \\ &= \boldsymbol{F}(\boldsymbol{\theta}^*). \end{aligned}$$

Therefore, we can apply the *Central Limit Theorem* to the right hand side of equation (10) to obtain

$$\frac{1}{N}\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^*) \sim \mathcal{N}(\mathbf{0}, \frac{1}{N}\boldsymbol{F}(\boldsymbol{\theta}^*)).$$

Therefore, from equation (10) and the last result we obtain

$$\boldsymbol{F}(\boldsymbol{\theta}^*)(\boldsymbol{\Theta}^N - \boldsymbol{\theta}^*) \sim \mathcal{N}(\mathbf{0}, \frac{1}{N}\boldsymbol{F}(\boldsymbol{\theta}^*)).$$

We can also express the last result as

$$\sqrt{N}(\boldsymbol{\Theta}^N - \boldsymbol{\theta}^*) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{F}(\boldsymbol{\theta}^*)^{-1}),$$

which is the second part of the result presented in the last paragraph.

In general, the Hessian of a function tells us how the first derivatives of the function are changing (Figure 2). Assume that the we evaluate the Hessian where the function attains a local maximum and some regularity conditions on the function hold. Then the Hessian evaluated at that point is a symmetric negative definite matrix. In particular, if the first derivatives do not change much and their magnitude is small in all directions in the neighborhood of the local maximum, then the surface in that area is almost flat corresponding to a Hessian whose elements are negative numbers with relatively small magnitude. If the change in first derivatives is only in the direction of the axes, then its off-diagonal elements are very close to zero. If the first derivatives do change significantly in all directions, the surface of the function is more peaked than in the previous case, and this corresponds to a Hessian whose components have relatively large magnitude. If the changes in first derivatives occur

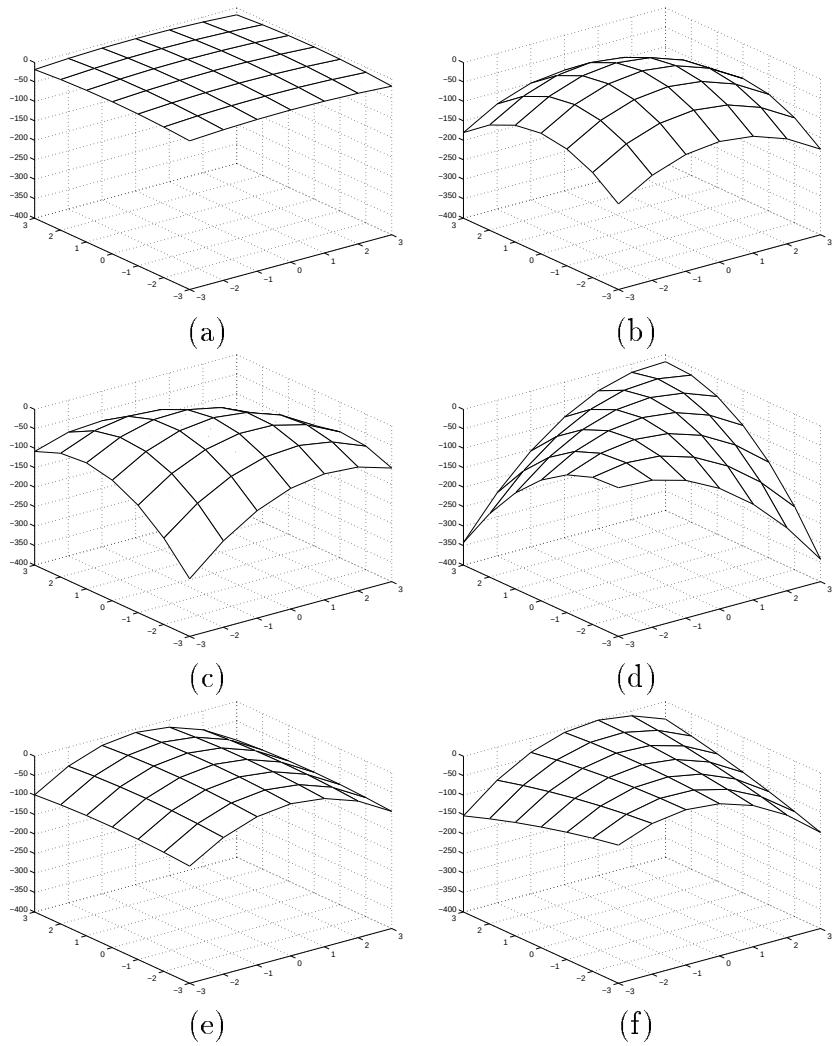


Figure 2: Plot of the function $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}$ for (a) $\boldsymbol{A} = [-1, 0; 0, -1]$, (b) $\boldsymbol{A} = [-10, 0; 0, -10]$, (c) $\boldsymbol{A} = [-10, -4; -4, -10]$, (d) $\boldsymbol{A} = [-10, 9; 9, -10]$, (e) $\boldsymbol{A} = [-10, 0; 0, -1]$, and (f) $\boldsymbol{A} = [-10, 3; 3, -1]$. The Hessian of f is $\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{A}$. Their respective condition numbers are: (a) $\kappa[\boldsymbol{A}] = 1$, (b) $\kappa[\boldsymbol{A}] = 1$, (c) $\kappa[\boldsymbol{A}] = 2.3$, (d) $\kappa[\boldsymbol{A}] = 19$, (e) $\kappa[\boldsymbol{A}] = 10$, and (f) $\kappa[\boldsymbol{A}] = 118.99$.

in directions that are not exactly those of the axes, then the surface of the function is elongated along those directions and this behavior corresponds to a Hessian whose off-diagonal elements are significantly different from zero. The magnitude of the elongation depends on the magnitude of the off-diagonal terms. The direction of the elongation depends on the sign of the off-diagonal terms. The function is also flat in the direction of the elongation and how flat it is in this direction depends on the magnitude of the elongation itself.

A problem is *ill-conditioned* if different approximate solutions give about the same fit to the data [RW84]. This behavior corresponds to regions where the function is very flat, and typically happens in the mixture-of-Gaussians model when the Gaussians are not well separated (Figure 3). In nonlinear optimization, the condition of a problem is typically

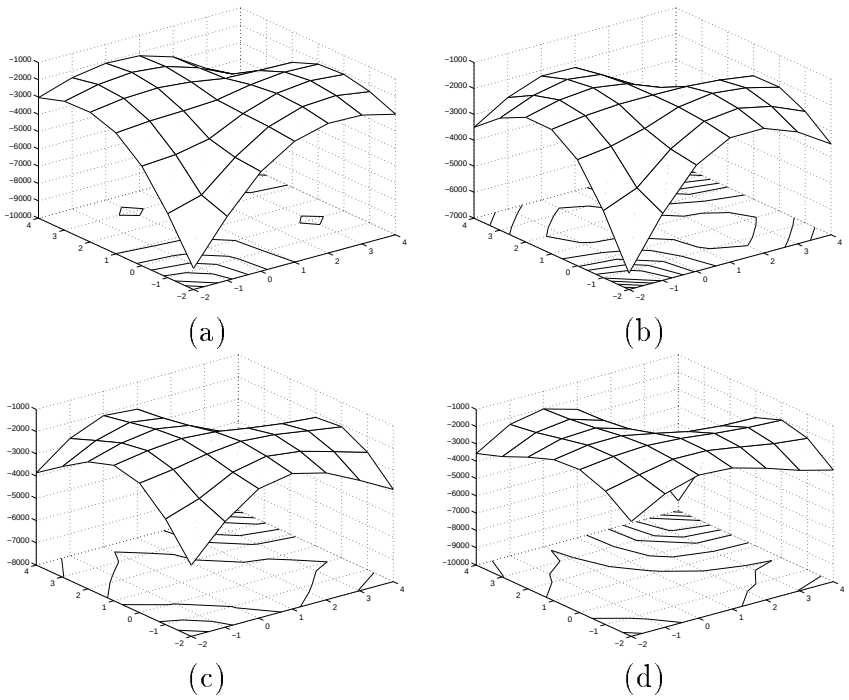


Figure 3: Plots of $L(\boldsymbol{\theta})$ for 1-dimensional data as a function of μ_1 and μ_2 with $\alpha_1 = \alpha_2 = 0.5, \sigma_1 = \sigma_2 = 1$. In this case the data is 1000 sampled points from the mixture model with parameters $\alpha_1 = \alpha_2 = 0.5, \sigma_1 = \sigma_2 = 1, \mu_1 = 0$ and (a) $\mu_2 = 3$, (b) $\mu_2 = 2$, (c) $\mu_2 = 1$, and (d) $\mu_2 = 0$. Note how the log likelihood function gets flatter around the maximum as the means in the mixture get closer to each other.

measured by the condition number of the Hessian evaluated at the solution [1]. The condition number of a matrix measures the degree to which a pair of columns of the matrix are linearly dependent. In two dimensions, the columns are close to linear dependence when the off-diagonal elements of the matrix get close in magnitude to the diagonal elements. When the matrix is the Hessian of a function in 2 dimensions evaluated at a local maximum of the function, this corresponds to the function being more elongated, and therefore flatter, along some direction. Note that we can approximate $L(\boldsymbol{\theta})$ in the neighborhood of a local maxima at $\boldsymbol{\theta}^N$ using a quadratic Taylor series expansion around $\boldsymbol{\theta}^N$:

$$\begin{aligned} L(\boldsymbol{\theta}) &\approx L(\boldsymbol{\theta}^N) + (\boldsymbol{\theta} - \boldsymbol{\theta}^N)^T \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^N) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^N)^T \nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^N)(\boldsymbol{\theta} - \boldsymbol{\theta}^N) \\ &= L(\boldsymbol{\theta}^N) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^N)^T \boldsymbol{H}(\boldsymbol{\theta}^N)(\boldsymbol{\theta} - \boldsymbol{\theta}^N) \end{aligned}$$

[1] Given some matrix norm $\|\cdot\|$, the *condition number* of a matrix $\boldsymbol{A}$ is

$$\kappa[\boldsymbol{A}] = \|\boldsymbol{A}\| \cdot \left\|\boldsymbol{A}^{-1}\right\| = \lambda_M[\boldsymbol{A}]/\lambda_m[\boldsymbol{A}],$$

where $\lambda_M[\boldsymbol{A}]$ is the eigenvalue of $\boldsymbol{A}$ with the largest magnitude and $\lambda_m[\boldsymbol{A}]$ is the eigenvalue with the smallest magnitude.

As we get more data, $\boldsymbol{\theta}^N \to \boldsymbol{\theta}^*$ and $\frac{1}{N}\boldsymbol{H}(\boldsymbol{\theta}^N) \to -\boldsymbol{F}(\boldsymbol{\theta}^*)$, yielding

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \left(-N\boldsymbol{F}(\boldsymbol{\theta}^*)\right)(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$$

Since, for sufficiently large N, the condition numbers of $\boldsymbol{H}(\boldsymbol{\theta}^N)$, $-N\boldsymbol{F}(\boldsymbol{\theta}^*)$ and $\boldsymbol{F}(\boldsymbol{\theta}^*)$ are the same, the condition number of the Fisher information matrix evaluated at the true values for the parameters determines the condition of the problem for sufficiently large N. Therefore, the condition number of $\boldsymbol{F}(\boldsymbol{\theta}^*)$ determines the limits of accuracy with which we can numerically approximate the solution in practice.

Note that we can develop an iterative procedure from the likelihood equations. The method is as follows. We first initialize the parameters of the model somehow. We evaluate the right-hand side of the likelihood equations (as presented above) using the current setting of the parameters and get new settings for the parameters by assigning the result of the computation to the appropriate parameter in the left-hand side of the likelihood equations. We continue iterating until some stopping condition holds. More specifically, for this iterative method, we initialize the parameters $\boldsymbol{\theta}^{(0)}$ to some values and iterate to get new estimates using the updates below:

$$\alpha_j^{(k+1)} = \frac{1}{N}\sum_{i=1}^{N} \alpha_j^{(k)} \frac{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})}, \text{ for } 1 \le j \le M. \tag{11}$$

$$\boldsymbol{\mu}_j^{(k+1)} = \frac{\sum_{i=1}^{N} \frac{\alpha_j^{(k)} g(\boldsymbol{x}_i | \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{p(\boldsymbol{x}_i | \boldsymbol{\theta}^{(k)})} \boldsymbol{x}_i}{\sum_{i=1}^{N} \frac{\alpha_j^{(k)} g(\boldsymbol{x}_i | \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{p(\boldsymbol{x}_i | \boldsymbol{\theta}^{(k)})}}, \text{for } 1 \le j \le M. \tag{12}$$

$$\boldsymbol{\Sigma}_j^{(k+1)} = \frac{\sum_{i=1}^{N} \frac{\alpha_j^{(k)} g(\boldsymbol{x}_i | \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{p(\boldsymbol{x}_i | \boldsymbol{\theta}^{(k)})} (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})^T}{\sum_{i=1}^{N} \frac{\alpha_j^{(k)} g(\boldsymbol{x}_i | \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{p(\boldsymbol{x}_i | \boldsymbol{\theta}^{(k)})}}, \text{for } 1 \le j \le M. \tag{13}$$

This method monotonically increases the log likelihood and is guaranteed to converge (modulo singularities). We will see why in Section (5). Before a prove of convergence existed, researchers observed that this method converged often in practice [PW78]. Acceleration methods based on Aitken acceleration (see Section (7.1)) have also been used for this method [PW78].

4 Gradient-based methods

Gradient-based methods for nonlinear optimization give us another way to find a zero of the Jacobian of the function we want to optimize. In general, if the method converges, the result will give us a critical point of the original function, and therefore we need to perform additional analysis in order to fully characterize the solution. Let $\boldsymbol{A}^{(k)}$ be a matrix that

can depend on $\boldsymbol{\theta}^{(k)}$, and $\boldsymbol{r}^{(k)} = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(k)})$ be the gradient of the function with respect to the parameters evaluated at the current setting of the parameters $\boldsymbol{\theta}^{(k)}$. In general, in these methods we get new values at each iteration using the following update rule,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{A}^{(k)} \boldsymbol{r}^{(k)}, \tag{14}$$

where

$$\gamma^{(k)} = \underset{\gamma}{\operatorname{argmax}}\, f(\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{A}^{(k)} \boldsymbol{r}^{(k)}). \tag{15}$$

This update rule says that our new setting for the parameters is our current setting for the parameters offset by a step in some transformed direction $\gamma^{(k)} \boldsymbol{A}^{(k)} \boldsymbol{r}^{(k)}$ that depends on the current setting of the parameters, where $\boldsymbol{r}^{(k)}$ is the original direction, $\boldsymbol{A}^{(k)}$ is the transforming matrix, and $\gamma^{(k)}$ is the magnitude of the offset along the transformed direction. Note that using this general form, where we optimize the step size $\gamma^{(k)}$ at each iteration, the value of the function will not decrease after each iteration.

Many variants of this general formulation exist. We refer the reader to [Ber95, Pol71, PTVF92] for more detailed information on the different variants. In this paper, we briefly describe some of the most common variants.

Note that the gradient methods presented here are methods for unconstraint optimization and therefore we need to adapt them to handle the constraints on the parameters imposed by the mixture of Gaussians model.

The most typical convergence rate analysis of gradient-based methods is the *local analysis*, where we are interested in the behavior of the method when close to a solution (see Appendix D). Let us first study the convergence characteristic of a gradient-based method that uses $\boldsymbol{A}^{(k)} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix (i.e., the step is taken strictly in the direction of the gradient). This method is the *steepest ascent* method [Ber95]. Expressed in this way, the update rule in (14) becomes

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{r}^{(k)}, \tag{16}$$

with $\gamma^{(k)}$ as in (15).

Next, we present a local convergence rate result for the steepest ascent method that ties the convergence rate to the condition number of the Hessian of the function evaluated at the solution. For clarity, assume that the function is quadratic. For non-quadratic functions, the idea behind the result is the same but requires more care. First note that by Taylor expansion around the solution $\boldsymbol{\theta}^N$,

$$\boldsymbol{r}^{(k)} = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^N) + \nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^N) \left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N\right) = \boldsymbol{H}(\boldsymbol{\theta}^N) \left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N\right).$$

Using the expression above for the value of the gradient of f at $\boldsymbol{\theta}^{(k)}$ in the update rule given in (16), we get the approximation to the update rule below,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{H}(\boldsymbol{\theta}^N) \left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N\right).$$

Subtracting the solution $\boldsymbol{\theta}^N$ at both sides, we get

$$\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^N = \left(\boldsymbol{I} + \gamma^{(k)}\boldsymbol{H}(\boldsymbol{\theta}^N)\right)\left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N\right),$$

from which we get (see Appendix (E)),

$$\left\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^N\right\| \leq \left\|\boldsymbol{I} + \gamma^{(k)}\boldsymbol{H}(\boldsymbol{\theta}^N)\right\| \left\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N\right\|.$$

We let the convergence rate factor $s^{(k)}$ be

$$s^{(k)} = \left\|\boldsymbol{I} + \gamma^{(k)}\boldsymbol{H}(\boldsymbol{\theta}^N)\right\| = \lambda_M\left[\boldsymbol{I} + \gamma^{(k)}\boldsymbol{H}(\boldsymbol{\theta}^{\boldsymbol{N}})\right], \tag{17}$$

by a property of eigenvalues and the matrix norm. Also by a property of eigenvalues,

$$\lambda_M\left[\boldsymbol{I} + \gamma^{(k)}\boldsymbol{H}(\boldsymbol{\theta}^N)\right] = \max\left\{\left|1 + \gamma^{(k)}\lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]\right|, \left|1 + \gamma^{(k)}\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]\right|\right\}. \tag{18}$$

Therefore, the value of the step size $\gamma^{(k)}$ for which the convergence rate factor $s^{(k)}$ attains its minimum value is

$$\gamma^N = \underset{\gamma}{\operatorname{argmin}} \max\left\{\left|1 + \gamma^{(k)}\lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]\right|, \left|1 + \gamma^{(k)}\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]\right|\right\},$$

for which we can find a solution by solving the following equation,

$$-\left(1 + \gamma^N\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]\right) = 1 + \gamma^N\lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right].$$

By solving the last equation we get the value for γ^N,

$$\gamma^N = \frac{-2}{\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right] + \lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]}. \tag{19}$$

Substituting back the value of γ^N given in (19) into the bound for the convergence rate factor $s^{(k)}$ given in (18) and using the result to re-express the upper bound of $s^{(k)}$ given in (17), we get

$$s^{(k)} \leq \frac{\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right] - \lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]}{\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right] + \lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]} \leq 1 - \frac{\lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]}{\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]} = 1 - \kappa^{-1}[\boldsymbol{H}(\boldsymbol{\theta}^N)].$$

Therefore, an estimate of the convergence rate on the parameter space for steepest ascent is

$$\left\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^N\right\| \leq \left(1 - \kappa^{-1}[\boldsymbol{H}(\boldsymbol{\theta}^N)]\right)\left\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N\right\|. \tag{20}$$

Using a similar but more complicated analysis we can obtain a similar estimate for the convergence rate on the function space for steepest ascent,

$$\begin{aligned}\frac{f(\boldsymbol{\theta}^{(k+1)}) - f(\boldsymbol{\theta}^N)}{f(\boldsymbol{\theta}^{(k)}) - f(\boldsymbol{\theta}^N)} &\leq \left(\frac{\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right] - \lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]}{\lambda_M\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right] + \lambda_m\left[\boldsymbol{H}(\boldsymbol{\theta}^N)\right]}\right)^2 \\ &\leq \left(1 - \kappa^{-1}[\boldsymbol{H}(\boldsymbol{\theta}^N)]\right)^2.\end{aligned}$$

We can obtain the general form of the gradient methods given in (14) if we perform steepest ascent on a transformed space. Let the transforming matrix $\boldsymbol{T}$ be [2]

$$\boldsymbol{T} = (\boldsymbol{A}^{(k)})^{\frac{1}{2}}, \tag{21}$$

and the new parameter space $\boldsymbol{\phi}$ resulting from the transformation be such that

$$\boldsymbol{\theta} = \boldsymbol{T}\boldsymbol{\phi}. \tag{22}$$

Let the function defined on the new parameter space be

$$h(\boldsymbol{\phi}) = f(\boldsymbol{T}\boldsymbol{\phi}). \tag{23}$$

Performing steepest ascent on the function $h(\boldsymbol{\phi})$ we get the update rule

$$\boldsymbol{\phi}^{(k+1)} = \boldsymbol{\phi}^{(k)} + \gamma^{(k)} \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k)}). \tag{24}$$

From the definition of $h(\boldsymbol{\phi})$ given in (23) and the chain rule for gradients we get

$$\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}) = \boldsymbol{T} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}). \tag{25}$$

Premultiplying both sides of the update rule given in (24) by the matrix $\boldsymbol{T}$ as given in (21) we get

$$\boldsymbol{T}\boldsymbol{\phi}^{(k+1)} = \boldsymbol{T}\boldsymbol{\phi}^{(k)} + \gamma^{(k)} \boldsymbol{T} \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k)}).$$

Substituting the gradient of $h(\boldsymbol{\phi})$ given in (25) into the last expression for the update rule and using the definition of the matrix $\boldsymbol{T}$ given in (21) we get

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{A}^{(k)} \boldsymbol{r}^{(k)}, \tag{26}$$

which is the general form of the update rule for gradient methods. From the above exposed relationship between steepest ascent and the general gradient methods, and the estimates on the convergence rate for steepest ascent, we can get estimates on the convergence rate of general gradient methods. That is, from the convergence rate results for steepest ascent above,

$$\begin{aligned} \left\| \boldsymbol{\phi}^{(k+1)} - \boldsymbol{\phi}^{N} \right\| &\leq \left\| \boldsymbol{I} + \gamma^{(k)} \nabla^2_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{N}) \right\| \left\| \boldsymbol{\phi}^{(k)} - \boldsymbol{\phi}^{N} \right\| \\ &= \lambda_M \left[\boldsymbol{I} + \gamma^{(k)} \nabla^2_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{N}) \right] \left\| \boldsymbol{\phi}^{(k)} - \boldsymbol{\phi}^{N} \right\| \\ &\leq \left(1 - \kappa^{-1} [\nabla^2_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{N})] \right) \left\| \boldsymbol{\phi}^{(k)} - \boldsymbol{\phi}^{N} \right\|. \end{aligned}$$

Since, from the expression for the gradient of $h(\boldsymbol{\phi})$ given in (25) and the chain rule for gradients,

$$\nabla^2_{\boldsymbol{\phi}} h(\boldsymbol{\phi}) = \boldsymbol{T} \nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \boldsymbol{T},$$

[2] The *symmetric square root* of a square symmetric nonnegative definite matrix $\boldsymbol{A}$, denoted as $\boldsymbol{A}^{\frac{1}{2}}$, is a symmetric matrix such that $\boldsymbol{A}^{\frac{1}{2}} \boldsymbol{A}^{\frac{1}{2}} = \boldsymbol{A}$.

we get

$$\left\| \left((\boldsymbol{A}^{(k)})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{N} \right) \right\| \leq \left\| \boldsymbol{I} + \gamma^{(k)} (\boldsymbol{A}^{(k)})^{\frac{1}{2}} \boldsymbol{H}(\boldsymbol{\theta}^{N}) (\boldsymbol{A}^{(k)})^{\frac{1}{2}} \right\| \left\| \left((\boldsymbol{A}^{(k)})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{N} \right) \right\|$$

and

$$\left\| \left((\boldsymbol{A}^{(k)})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{N} \right) \right\| \leq \left(1 - \kappa^{-1} [(\boldsymbol{A}^{(k)})^{\frac{1}{2}} \boldsymbol{H}(\boldsymbol{\theta}^{N}) (\boldsymbol{A}^{(k)})^{\frac{1}{2}}] \right) \left\| \left((\boldsymbol{A}^{(k)})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{N} \right) \right\|. \tag{27}$$

We can also get another form for the convergence rate by substituting the Taylor expansion of the gradient directly; that is,

$$\left\| \boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{N} \right\| \leq \left\| \boldsymbol{I} + \gamma^{(k)} \boldsymbol{A}^{(k)} \boldsymbol{H}(\boldsymbol{\theta}^{N}) \right\| \left\| \boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{N} \right\| \tag{28}$$

From the last result we see that if we let $\boldsymbol{A}^{(k)} = -\boldsymbol{H}(\boldsymbol{\theta}^{N})^{-1}$, the value of $\kappa^{-1}[(\boldsymbol{A}^{(k)})^{\frac{1}{2}} \boldsymbol{H}(\boldsymbol{\theta}^{N}) (\boldsymbol{A}^{(k)})^{\frac{1}{2}}]$ becomes 1, the optimal value for $\gamma^{(k)}$ becomes 1, and $s^{(k)}$ becomes 0 which gives us a method with super-linear convergence. Therefore, from the standpoint of speed of convergence, we should use $\boldsymbol{A}^{(k)}$ as close as possible to $-\boldsymbol{H}(\boldsymbol{\theta}^{N})^{-1}$ since then the value of $\kappa^{-1}[(\boldsymbol{A}^{(k)})^{\frac{1}{2}} \boldsymbol{H}(\boldsymbol{\theta}^{N}) (\boldsymbol{A}^{(k)})^{\frac{1}{2}}]$ will be close to 1, the optimal value of $\gamma^{(k)}$ will be close to 1, and $s^{(k)}$ will be close to 0 which gives us a method with close to super-linear convergence. Since we do not know where the function has the maximum (that is exactly what we are trying to find out), the idea behind the methods below is to find ways to approximate $-\boldsymbol{H}(\boldsymbol{\theta}^{N})^{-1}$, or, from the standpoint of speed of convergence, ways to improve the condition number of the *effective Hessian* $(\boldsymbol{A}^{(k)})^{\frac{1}{2}} \boldsymbol{H}(\boldsymbol{\theta}^{N}) (\boldsymbol{A}^{(k)})^{\frac{1}{2}}$.

4.1 Newton-Raphson method

To obtain Newton-Raphson method, we let $\boldsymbol{A}^{(k)} = -\boldsymbol{H}(\boldsymbol{\theta}^{(k)})^{-1}$, and $\gamma^{(k)} = 1$. Therefore, we obtain the update rule

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \boldsymbol{H}(\boldsymbol{\theta}^{(k)})^{-1} \boldsymbol{r}^{(k)}. \tag{29}$$

Newton-Raphson method has super-linear convergence when close to the solution where the function can be well approximated by a quadratic function and therefore, the Hessian is almost constant and $\boldsymbol{H}(\boldsymbol{\theta}^{(k)}) \approx \boldsymbol{H}(\boldsymbol{\theta}^{N})$. However, the $\boldsymbol{H}(\boldsymbol{\theta}^{(k)})$ is typically singular and/or not positive definite when not close to a solution and it may not converge to a solution. Note the extra work of computing the Hessian at every step which can be a very expensive operation. Furthermore, the size of the matrix increases as the square of the number of parameters. Therefore, the method presents problems when dealing with a large number of parameters.

4.2 Quasi-Newton or variable metric methods

This method uses approximations to the Hessian at every step that are, among other things, easier to compute and non-singular. To obtain this type of method, we let $\boldsymbol{A}^{(k)} = \boldsymbol{M}^{(k)}$, where $\boldsymbol{M}^{(k)}$ is a symmetric, positive-definite matrix that we use to approximate $-\boldsymbol{H}(\boldsymbol{\theta}^{(k)})^{-1}$. We also let $\gamma^{(k)} = 1$ as in the Newton-Raphson method. The update rule is then,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \boldsymbol{M}^{(k)}\boldsymbol{r}^{(k)} \tag{30}$$

Some ways of computing $\boldsymbol{M}^{(k)}$ is to use the diagonals of the Hessian instead of the full Hessian (as long as they are all negative) or to do a secant update of the Hessian at each step. This method features super-linear convergence when close to a solution and typically improves on the global convergence properties of the regular Newton-Raphson method because now, by imposing that $\boldsymbol{M}^{(k)}$ be symmetric, positive-definite, the matrix is never singular, even if it is far away from a solution.

4.3 Gradient ascent

We obtain the general form of the gradient ascent method if we let $\boldsymbol{A}^{(k)} = \boldsymbol{I}$, and $\gamma^{(k)}$ be some constant or sequence which does not necessarily maximize $f(\boldsymbol{\theta}^{(k)} + \gamma\boldsymbol{r}^{(k)})$. The update rule becomes,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma^{(k)}\boldsymbol{r}^{(k)} \tag{31}$$

Note that what we are doing in this method is strictly moving in the direction of the gradient with a step size determined by the value of $\gamma^{(k)}$. Note that this method looks like a quasi-Newton method where we approximate the Hessian at every step as $\gamma^{(k)}\boldsymbol{I}$. Not surprisingly, this method, because of the very rough approximation to the inverse of the Hessian it uses, typically has slower linear convergence. Also it is not always clear how to determine the most appropriate value for $\gamma^{(k)}$. If it is a constant, it should typically be small. However, too small a value typically gives slow convergence. Too large a value can improve convergence speed but also increases the chances of divergence. Several rules and heuristics exists for establishing this sequence. If we are lucky and pick the optimal parameter at each step, the method is equivalent to steepest ascent and therefore can show super-linear convergence in some instances.

5 Expectation-Maximization (EM)

EM [DLR77] is the method most commonly used to obtain the MLE parameters for the mixture-of-Gaussians model. This is a very general method for producing MLE parameters from incomplete data. Next we present a very specific derivation of this method for the mixture-of-Gaussians model. First, note that from the chain rule of probability,

$$p(\boldsymbol{Z}, \boldsymbol{D} \mid \theta) = P(\boldsymbol{Z} \mid \boldsymbol{D}, \theta)p(\boldsymbol{D} \mid \theta).$$

Now we apply the natural logarithm function to both sides and use the properties of logarithms on the right side of the equation to get

$$\ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) = \ln P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}) + \ln p(\boldsymbol{D} \mid \boldsymbol{\theta}).$$

Using algebra to rearrange the equation we get

$$\ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) = \ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) - \ln P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}).$$

Assume that we have a different setting of the parameters $\boldsymbol{\theta}'$, and therefore another distribution on the indicators $\boldsymbol{Z}$ conditioned on the data points $\boldsymbol{D}$ and $\boldsymbol{\theta}'$; denote this distribution by $P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}')$. Note that we can take an expectation of the logarithm terms with respect to $P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}')$. We can do this by multiplying each side of the equation by $P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}')$. Performing that operation and distributing over the subtraction we get

$$\begin{aligned} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) &= P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) - \\ & \quad P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln p(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}). \end{aligned}$$

Summing over all possible values of the indicators $\boldsymbol{Z}$, we get

$$\begin{aligned} \sum_{\boldsymbol{Z}} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) &= \sum_{\boldsymbol{Z}} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) - \\ & \quad \sum_{\boldsymbol{Z}} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}). \end{aligned}$$

We now simplify by using the rules of probabilities to obtain the equation

$$\begin{aligned} \ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) &= \sum_{\boldsymbol{Z}} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) - \\ & \quad \sum_{\boldsymbol{Z}} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}). \end{aligned}$$

If we let the log likelihood $L(\boldsymbol{\theta}) = \ln p(\boldsymbol{D} \mid \boldsymbol{\theta})$, the expected log likelihood $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}') = \sum_{\boldsymbol{Z}} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta})$, and the entropy $H(\boldsymbol{\theta} \mid \boldsymbol{\theta}') = \sum_{\boldsymbol{Z}} P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \ln P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta})$, then we can rewrite the last equation as

$$L(\boldsymbol{\theta}) = Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}') - H(\boldsymbol{\theta} \mid \boldsymbol{\theta}'). \tag{32}$$

Let $KL(P' || Q')$ be the *Kullback-Leibler or relative entropy distance* between the distribution P' and Q' over the same set of events [CT91]. Note then that the difference in entropies $H(\boldsymbol{\theta}' \mid \boldsymbol{\theta}') - H(\boldsymbol{\theta} \mid \boldsymbol{\theta}')$ is

$$H(\boldsymbol{\theta}' \mid \boldsymbol{\theta}') - H(\boldsymbol{\theta} \mid \boldsymbol{\theta}') = KL(P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \mid\mid P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta})) \geq 0,$$

where the resulting inequality comes from a result that is a consequence of *Jensen's inequality* and is known as the *Information inequality*. Equality in the above inequality holds if and only if $P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') = P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta})$ for all $\boldsymbol{Z}$.

Therefore, if $P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') \neq P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta})$ and $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}') > Q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}')$, then $L(\boldsymbol{\theta}) > L(\boldsymbol{\theta}')$. In addition, if $P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}') = P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta})$ and $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}') = Q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}')$, then $L(\boldsymbol{\theta}) = L(\boldsymbol{\theta}')$. In other words, if we increase the expected value of the log-likelihood for the *complete* data (i.e., the data observed and the indicators for the Gaussian used to generate each data point) with respect to the *new* parameters given the current setting of the parameters and the *incomplete* (i.e., observed) data, then the log-likelihood of the parameters for the *incomplete* data (i.e., the function that we really want to maximize) will also increase.

The EM method results from the last observation. In this method, we start with an initial estimate of the parameters $\boldsymbol{\theta}^{(0)}$. Then, at each iteration $k+1$, we perform the following two steps:

Expectation step

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) = E_{\boldsymbol{Z}}[\ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) \mid \boldsymbol{D}, \boldsymbol{\theta}^{(k)}]; \tag{33}$$

Maximization step

$$\boldsymbol{\theta}^{(k+1)} \in \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}). \tag{34}$$

If the set $\arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$ is a singleton, as it is for the case of the mixture-of-Gaussians model, we replace the symbol $\in$ by the symbol $=$ in the expression above. On the other hand, note that we do not really need to maximize $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$ with respect to $\boldsymbol{\theta}$ at each iteration in order to monotonically increase the log likelihood at each iteration. All we need to do is to find $\boldsymbol{\theta}^{(k+1)}$ such that $Q(\boldsymbol{\theta}^{(k+1)} \mid \boldsymbol{\theta}^{(k)}) \geq Q(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\theta}^{(k)})$. A version of EM that does just that is called *Generalized EM (GEM)*.

There is another way in which we can derive and interpret EM. This interpretation is the basic idea behind current work that tries to cope with the computational issues that arise when we are dealing with models for which the expectation step is complex [Gha97]. By the property of marginalization of probabilities,

$$\ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) = \ln \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}).$$

Introducing a new distribution over the indicators $W(\boldsymbol{Z})$ as above, we get

$$\ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) = \ln \sum_{\boldsymbol{Z}} W(\boldsymbol{Z}) \frac{p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta})}{W(\boldsymbol{Z})}.$$

Now, by Jensen's inequality and some manipulation,

$$\begin{aligned} \ln p(\boldsymbol{D} \mid \boldsymbol{\theta}) &\geq \sum_{\boldsymbol{Z}} W(\boldsymbol{Z}) \ln \frac{p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta})}{W(\boldsymbol{Z})} \\ &= \sum_{\boldsymbol{Z}} W(\boldsymbol{Z}) \ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) - \sum_{\boldsymbol{Z}} W(\boldsymbol{Z}) \ln W(\boldsymbol{Z}) \\ &= \mathcal{F}(W, \boldsymbol{\theta}). \end{aligned}$$

If we define $\ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta})$ to be the *energy* of a global configuration $(\boldsymbol{Z}, \boldsymbol{D})$, then the lower bound $\mathcal{F}(W, \boldsymbol{\theta})$ is the negative of a quantity known in statistical physics as the *free energy*. Viewing it this way, EM is now, starting from some initial setting of the parameters $\boldsymbol{\theta}^{(0)}$:

Expectation step

$$W^{(k+1)} \in \arg\max_W \mathcal{F}(W, \boldsymbol{\theta}^{(k)}); \tag{35}$$

Maximization step

$$\boldsymbol{\theta}^{(k+1)} \in \arg\max_{\boldsymbol{\theta}} \mathcal{F}(W^{(k+1)}, \boldsymbol{\theta}). \tag{36}$$

From the above steps, we can also see EM as improving a lower bound on the log-likelihood function with respect to the current setting of the parameters at each iteration. If we let $W^{(k+1)}(\boldsymbol{Z}) = P(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{\theta}^{(k)})$, then the inequality becomes an equality; that is, $\mathcal{F}(W^{(k+1)}, \boldsymbol{\theta}^{(k+1)}) = \ln p(\boldsymbol{D} \mid \boldsymbol{\theta}^{(k+1)})$. We can see the last statement from equation (32) in the previous derivation of EM. Therefore, that assignment for W maximizes the expectation step. Substituting back the maximizing value with respect to W in the expectation step we get the traditional version of the EM method that results in the steps given in equations (33) and (34).

Using the formulation of EM given by equations (33) and (34) above we can determine more specific update rules for the parameters in our problem. First, let us develop the expectation step. By taking the conditional expectation of the logarithm of the complete data likelihood $p(\boldsymbol{Z}, \boldsymbol{D}|\boldsymbol{\theta})$ as defined in equation (5) we obtain

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) = E_{\boldsymbol{Z}} \left[\ln \prod_{i=1}^{N} \prod_{j=1}^{M} \left(\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right)^{z_{i,j}} \Big| \boldsymbol{D}, \boldsymbol{\theta}^{(k)} \right].$$

Using the properties of logarithms and expectation, we get

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) = \sum_{i=1}^{N} \sum_{j=1}^{M} E_{z_{i,j}} \left[z_{i,j} \ln \left(\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right) \Big| \boldsymbol{x}_i, \boldsymbol{\theta}^{(k)} \right].$$

Since the factor $\ln \left(\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right)$ is constant with respect to the indicators $\boldsymbol{Z}$, we can simplify the last equation by

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) = \sum_{i=1}^{N} \sum_{j=1}^{M} E_{z_{i,j}} \left[z_{i,j} \mid \boldsymbol{x}_i, \boldsymbol{\theta}^{(k)} \right] \ln \left(\alpha_j g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right)$$

The expectation term is the expected value of the indicators with respect to the dataset and the current setting of the parameters. This value is

$$\hat{z}_{i,j}^{(k)} = E_{z_{i,j}} \left[z_{i,j} \mid \boldsymbol{x}_i, \boldsymbol{\theta}^{(k)} \right] = P(z_{i,j} = 1 \mid \boldsymbol{x}_i, \boldsymbol{\theta}^{(k)}).$$

In other words, $\hat{z}_{i,j}^{(k)}$ is the probability that the Gaussian j generated datum $\boldsymbol{x}_i$ with respect to the current setting of the parameters $\boldsymbol{\theta}^{(k)}$. Using this information, we can finally express

the expectation step of the method as computing

$$\hat{z}_{i,j}^{(k)} = \frac{\alpha_j^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{\sum_{j=1}^{M} \alpha_j^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})} = \frac{\alpha_j^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \tag{37}$$

for all Gaussians and all data points. We can also express the function Q as

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \hat{z}_{i,j}^{(k)} \left(\ln \alpha_j + \ln g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right) \tag{38}$$

To perform the maximization step we take partial derivatives of equation (38) with respect to the parameters $\boldsymbol{\theta}$ and set them to zero. First let us determine the update rule for the α's. Taking derivatives of equation (38) with respect to α_j, using the *Lagrange multiplier* λ to satisfy the constraints on the α's, and applying the properties of derivatives we get

$$\begin{aligned} \frac{\partial Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})}{\partial \alpha_j} &= \left(\sum_{i=1}^{N} \sum_{j=1}^{M} \hat{z}_{i,j}^{(k)} \frac{\partial \left(\ln \alpha_j + \ln g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right)}{\partial \alpha_j} \right) - \\ & \lambda \frac{\partial (\sum_{j=1}^{M} \alpha_j - 1)}{\partial \alpha_j}. \end{aligned}$$

Taking derivatives of the subcomponents and setting the result to zero we obtain

$$\left(\sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \frac{1}{\alpha_j} \right) - \lambda = 0. \tag{39}$$

Trying to solve for λ, we sum over all the Gaussians and obtain

$$\sum_{j=1}^{M} \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} - \lambda \sum_{j=1}^{M} \alpha_j = 0.$$

Applying the properties of the α's, substituting for $\hat{z}_{i,j}^{(k)}$ from equation (37), and solving for λ we get

$$\lambda = N.$$

Substituting the value for λ from the last equation into equation (39) we get the update rule for each α_j

$$\alpha_j^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)}. \tag{40}$$

This update says that if we knew exactly which Gaussian generated which data point we would estimate the probability of each Gaussian to be the average number of points generated by that Gaussian. Since we do not know the *exact* number of points generated by

each Gaussian, we take the average over the *expected* number of points that each Gaussian generated.

Similarly, we compute the gradient of Q from equation (38) with respect to $\boldsymbol{\mu}_j$, and set it to the zero vector to get

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}_j} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) &= \sum_{i=1}^{N}\sum_{j=1}^{M} \hat{z}_{i,j}^{(k)} \nabla_{\boldsymbol{\mu}_j} \left(\ln \alpha_j + \ln g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right) \\
&= \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \frac{1}{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \nabla_{\boldsymbol{\mu}_j} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\
&= \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \frac{1}{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \nabla_{\boldsymbol{\mu}_j} \left(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)\right) \\
&= \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \left(-\frac{1}{2}\right) (-2\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)) \\
&= \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \boldsymbol{\Sigma}_j^{-1} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j\right) \\
&= \mathbf{0}.
\end{aligned}$$

Solving for $\boldsymbol{\mu}_j$ in the last equation we get the update rule for each mean $\boldsymbol{\mu}_j$

$$\boldsymbol{\mu}_j^{(k+1)} = \frac{\sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \boldsymbol{x}_i}{\sum_{i=1}^{N} \hat{z}_{i,j}^{(k)}}. \tag{41}$$

This update says that if we knew exactly which Gaussian generated which data point we would estimate the mean of each Gaussian to be the mean of the points generated by that Gaussian. Since for each data point we do not know *exactly* which Gaussian generated it, we weight each point by the probability that the Gaussian generated it given the current setting of the parameters and divide that by the *expected* number of points that the Gaussian generated. Another way of looking at this update is that the mean of a Gaussian is the *expected* value of the coordinates of the data points, where we take the expectation with respect to a conditional distribution over the data points given that Gaussian and the current setting of the parameters.

Finally, we compute the gradient of Q from equation (38) with respect to $\boldsymbol{\Sigma}_j$, and set it

to the zero vector to get

$$\begin{aligned}
\nabla_{\boldsymbol{\Sigma}_j} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) &= \sum_{i=1}^{N}\sum_{j=1}^{M} \hat{z}_{i,j}^{(k)} \nabla_{\boldsymbol{\Sigma}_j} \left(\ln \alpha_j + \ln g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right) \\
&= \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \frac{1}{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \nabla_{\boldsymbol{\Sigma}_j} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\
&= \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \frac{1}{g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \\
&\quad \left(\frac{-1}{2} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \mathrm{vec}\left(\boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{\Sigma}_j - (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T\right)\boldsymbol{\Sigma}_j^{-1}\right)\right) \\
&= \frac{-1}{2} \sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} \mathrm{vec}\left(\boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{\Sigma}_j - (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T\right)\boldsymbol{\Sigma}_j^{-1}\right) \\
&= \mathbf{0}.
\end{aligned}$$

Solving for $\boldsymbol{\Sigma}_j$ in the last equation we get the update rule for each covariance matrix $\boldsymbol{\Sigma}_j$

$$\boldsymbol{\Sigma}_j^{(k+1)} = \frac{\sum_{i=1}^{N} \hat{z}_{i,j}^{(k)} (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})^T}{\sum_{i=1}^{N} \hat{z}_{i,j}^{(k)}}. \tag{42}$$

Note again that the update rule is just what we expect. In the case that we observe the assignment of points to Gaussians, the new variance is the sample variance of the points associated with that Gaussian. Since we do not know that assignment, we weight the variance associated with each point by the conditional probability that the point belongs to the Gaussian we are updating given data and the current setting of the parameters.

Moreover, note that both the expectation and the maximization steps for this model are simple. This is no coincidence. The *complete-data likelihood* function for this model belongs to the exponential family. When this happens, the expectation step reduces to computing the *expected sufficient statistics* given the data and the current setting of the parameters. For the mixture-of-Gaussians model, the *sufficient statistics* are, for each Gaussian, the number of data points assigned to it (i.e., $\sum_{i=1}^{N} z_{i,j}$), and the mean vector (i.e., $(1/N)\sum_{i=1}^{N} z_{i,j}\boldsymbol{x}_i$) and covariance matrix (i.e., $(1/N)\sum_{i=1}^{N} z_{i,j}\boldsymbol{x}_i\boldsymbol{x}_i^T$) of those points. Since we do not know the assignment of Gaussians to data points, we can use the data and the current setting of the parameters to compute their expected value (i.e., $\hat{z}_{i,j}$). Now, to obtain the maximization step for such models, we replace the sufficient statistics in the expressions for the MLE for the model by the expected sufficient statistics. One of the reason why EM is so popular is that models from the exponential family occur often in practice, and for these models, the expressions for the MLE are simple. Therefore all we really need to do is to determine the expression for the expected value of the sufficient statistic given the data and the current setting of the parameters.

At this point, it is important to note the similarity between the EM method for the mixture-of-Gaussians model and the iterative method presented at the end of Section (3). They only differ in the way they update the covariance matrices: EM uses the value of the

means just computed while the iterative method uses the old value for the means. We can think of the iteration method as first sequentially maximizing $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$ first with respect to $\boldsymbol{\alpha}$, then with respect to each $\boldsymbol{\Sigma}_j$, and finally with respect to each $\boldsymbol{\mu}_j$ [Xu97]. Doing this we have

$$\begin{aligned}
Q(\boldsymbol{\alpha}^{(k+1)}, \boldsymbol{\mu}_1^{(k)}, \ldots, \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_1^{(k)}, \ldots, \boldsymbol{\Sigma}_M^{(k)} \mid \boldsymbol{\theta}^{(k)}) \geq \\
Q(\boldsymbol{\alpha}^{(k)}, \boldsymbol{\mu}_1^{(k)}, \ldots, \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_1^{(k)}, \ldots, \boldsymbol{\Sigma}_M^{(k)} \mid \boldsymbol{\theta}^{(k)}), \\
Q(\boldsymbol{\alpha}^{(k+1)}, \boldsymbol{\mu}_1^{(k)}, \ldots, \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_1^{(k+1)}, \ldots, \boldsymbol{\Sigma}_M^{(k+1)} \mid \boldsymbol{\theta}^{(k)}) \geq \\
Q(\boldsymbol{\alpha}^{(k+1)}, \boldsymbol{\mu}_1^{(k)}, \ldots, \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_1^{(k)}, \ldots, \boldsymbol{\Sigma}_M^{(k)} \mid \boldsymbol{\theta}^{(k)}), \\
Q(\boldsymbol{\alpha}^{(k+1)}, \boldsymbol{\mu}_1^{(k+1)}, \ldots, \boldsymbol{\mu}_M^{(k+1)}, \boldsymbol{\Sigma}_1^{(k+1)}, \ldots, \boldsymbol{\Sigma}_M^{(k+1)} \mid \boldsymbol{\theta}^{(k)}) \geq \\
Q(\boldsymbol{\alpha}^{(k+1)}, \boldsymbol{\mu}_1^{(k)}, \ldots, \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_1^{(k+1)}, \ldots, \boldsymbol{\Sigma}_M^{(k+1)} \mid \boldsymbol{\theta}^{(k)}),
\end{aligned}$$

or, in other words,

$$Q(\boldsymbol{\theta}^{(k+1)} \mid \boldsymbol{\theta}^{(k)}) \geq Q(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\theta}^{(k)}).$$

Therefore, we can view the iterative method based on the likelihood equations as a Generalized EM method, which implies that that iterative method monotonically increases the log likelihood at each iteration.

Finally, note that the result of the update rule for $\boldsymbol{\alpha}$ in both EM and the iterative method, given that $\boldsymbol{\alpha}^{(k)}$ satisfies the constraints (i.e., $\sum_{j=1}^{M} \alpha_j = 1$ and $\alpha_j > 0$ for all j), is a value for $\boldsymbol{\alpha}^{(k+1)}$ that also satisfy the constraints. The result of the update rule for $\boldsymbol{\Sigma}_j$, for all j, in both methods, given that $\boldsymbol{\theta}^{(k)}$ satisfies the constraints (i.e., the constraints on $\boldsymbol{\alpha}$ and that the covariance matrices, $\boldsymbol{\Sigma}_j$ for all j, are all symmetric and positive definite), is a symmetric matrix and it is positive definite with probability one assuming that N is large enough such that the matrix is of full rank [XJ96]. Redner and Walker [RW84] say on page 218 that $\boldsymbol{\Sigma}_j^{(k+1)}$ is positive definite with probability 1 if $N > d$. However, they state on page 227 that each $\boldsymbol{\Sigma}_j^{(k+1)}$ is "in the convex hull of" $\left\{(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})^T\right\}_{i=1,\ldots,N}$, "a set of rank-one matrices which, of course, are not positive definite. Thus there is no guarantee that a sequence of matrices" $\left\{\boldsymbol{\Sigma}_j^{(k)}\right\}_{k=0,1,2,\ldots}$ "produced by the EM iteration will remain bounded from below." The last statement seems to be justified given the possibility of singular solutions as discussed at the end of Section (2).

5.1 Relation between EM and gradient-based methods

The results presented in this section are mainly from recent articles on this subject [JX93, XJ96, Xu97]. The idea is to think of EM as a generalized gradient method that uses the conditioning matrix $\boldsymbol{A}^{(k)}$ on each step. This is done by equating the EM update rule to the gradient-based update rule and solving for $\boldsymbol{A}^{(k)}$. We denote that matrix as $\boldsymbol{A}_{\boldsymbol{\theta}^{(k)}}^{EM} = \text{diag}\left(\boldsymbol{A}_{\boldsymbol{\alpha}^{(k)}}^{EM}, \boldsymbol{A}_{\boldsymbol{\mu}_1^{(k)}}^{EM}, \ldots, \boldsymbol{A}_{\boldsymbol{\mu}_M^{(k)}}^{EM}, \boldsymbol{A}_{\boldsymbol{\Sigma}_1^{(k)}}^{EM}, \ldots, \boldsymbol{A}_{\boldsymbol{\Sigma}_M^{(k)}}^{EM}\right)$. We will find that $\boldsymbol{A}_{\boldsymbol{\theta}^{(k)}}^{EM}$ is always positive definite with probability 1 for sufficiently large N. Therefore, EM moves in a positive projection of the gradient. EM is similar to a variable metric approach since $\boldsymbol{A}_{\boldsymbol{\theta}^{(k)}}^{EM}$ changes

at every step. Note, however, that in EM we do not need to set step sizes and still will guarantee not to decrease the log likelihood function at every iteration and also, in the mixture-of-Gaussians model, to satisfy the constraints.

First of all, note that the iterative method presented in Section (3) based on the likelihood equations differs from EM only in the update rule for the covariance matrices. The update rule for the covariance matrix in the iterative method in Section (3) uses the current value of each mean $\boldsymbol{\mu}_j^{(k)}$. Instead EM uses the value new value of each mean $\boldsymbol{\mu}_j^{(k+1)}$ just computed in order to obtain new estimates for the covariance matrices. Therefore, we will start the analysis by first finding the matrix $\boldsymbol{A}^{(k)}$ corresponding to the iterative method. We denote that matrix as $\boldsymbol{A}^{LE}_{\boldsymbol{\theta}^{(k)}} = \operatorname{diag}\left(\boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}}, \boldsymbol{A}^{LE}_{\boldsymbol{\mu}_1^{(k)}}, \ldots, \boldsymbol{A}^{LE}_{\boldsymbol{\mu}_M^{(k)}}, \boldsymbol{A}^{LE}_{\boldsymbol{\Sigma}_1^{(k)}}, \ldots, \boldsymbol{A}^{LE}_{\boldsymbol{\Sigma}_M^{(k)}}\right)$. Since the transforming matrices corresponding to $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}_j$ for all j are the same for both methods (i.e., $\boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}} = \boldsymbol{A}^{EM}_{\boldsymbol{\alpha}^{(k)}}$ and $\boldsymbol{A}^{LE}_{\boldsymbol{\mu}_j^{(k)}} = \boldsymbol{A}^{EM}_{\boldsymbol{\mu}_1^{(k)}}$ for all j), all that is left is to relate the matrix corresponding to the updates of the covariance matrices obtained for the iterative method to that of EM.

For the mixture proportions ($\boldsymbol{\alpha}$) updates, we want to find a matrix $\boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}}$ such that

$$\boldsymbol{\alpha}^{(k)} + \boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}} \boldsymbol{r}_{\boldsymbol{\alpha}^{(k)}} = \frac{1}{N} \sum_{i=1}^{N} \hat{\boldsymbol{z}}_i^{(k)}.$$

As computed before, the gradient of the log likelihood function with respect to $\boldsymbol{\alpha}$ evaluated at $\boldsymbol{\theta}^{(k)}$ is

$$\boldsymbol{r}_{\boldsymbol{\alpha}^{(k)}} = \nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\theta}^{(k)}) = \sum_{i=1}^{N} \frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \begin{pmatrix} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix}.$$

Equating the gradient-based expression and the update of the iterative method we obtain

$$\begin{aligned} \boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}} \sum_{i=1}^{N} \frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \begin{pmatrix} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix} &= \sum_{i=1}^{N} \frac{1}{N} \left(\hat{\boldsymbol{z}}_i^{(k)} - \boldsymbol{\alpha}^{(k)}\right), \\ \sum_{i=1}^{N} \frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}} \begin{pmatrix} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix} &= \sum_{i=1}^{N} \frac{1}{N} \left(\hat{\boldsymbol{z}}_i^{(k)} - \boldsymbol{\alpha}^{(k)}\right). \end{aligned}$$

Let

$$\boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}} = \frac{1}{N} \left(\boldsymbol{B}^{LE}_{\boldsymbol{\alpha}^{(k)}} - \boldsymbol{C}^{LE}_{\boldsymbol{\alpha}^{(k)}}\right).$$

Now we want to find a matrix $\boldsymbol{B}^{LE}_{\boldsymbol{\alpha}^{(k)}}$ for which

$$\frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \boldsymbol{B}^{LE}_{\boldsymbol{\alpha}^{(k)}} \begin{pmatrix} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix} = \hat{\boldsymbol{z}}_i^{(k)}.$$

Let

$$\boldsymbol{B}^{LE}_{\boldsymbol{\alpha}^{(k)}} = \begin{pmatrix} \alpha_1^{(k)} & 0 & \cdots & 0 \\ 0 & \alpha_2^{(k)} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_M^{(k)} \end{pmatrix} = \mathrm{diag}(\alpha_1^{(k)}, \ldots, \alpha_M^{(k)}).$$

Then

$$\frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \boldsymbol{B}^{LE}_{\boldsymbol{\alpha}^{(k)}} \begin{pmatrix} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix} =$$

$$\begin{aligned} &= \frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \begin{pmatrix} \alpha_1^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ \alpha_M^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix} \\ &= \begin{pmatrix} \hat{z}_{i,j}^{(k)} \\ \vdots \\ \hat{z}_{i,j}^{(k)} \end{pmatrix} \\ &= \hat{\boldsymbol{z}}_i^{(k)}. \end{aligned}$$

We also want to find a matrix $\boldsymbol{C}^{LE}_{\boldsymbol{\alpha}^{(k)}}$ such that

$$\frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \boldsymbol{C}^{LE}_{\boldsymbol{\alpha}^{(k)}} \begin{pmatrix} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix} = \boldsymbol{\alpha}^{(k)}.$$

Let

$$\boldsymbol{C}^{LE}_{\boldsymbol{\alpha}^{(k)}} = \begin{pmatrix} \alpha_1^{(k)}\alpha_1^{(k)} & \alpha_1^{(k)}\alpha_2^{(k)} & \cdots & \alpha_1^{(k)}\alpha_M^{(k)} \\ \alpha_2^{(k)}\alpha_1^{(k)} & \alpha_2^{(k)}\alpha_2^{(k)} & & \alpha_2^{(k)}\alpha_M^{(k)} \\ \vdots & & \ddots & \vdots \\ \alpha_M^{(k)}\alpha_1^{(k)} & \alpha_M^{(k)}\alpha_2^{(k)} & \cdots & \alpha_M^{(k)}\alpha_M^{(k)} \end{pmatrix} = \boldsymbol{\alpha}^{(k)} \left(\boldsymbol{\alpha}^{(k)}\right)^T.$$

Then

$$\frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \boldsymbol{C}^{LE}_{\boldsymbol{\alpha}^{(k)}} \begin{pmatrix} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_1^{(k)}, \boldsymbol{\Sigma}_1^{(k)}) \\ \vdots \\ g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_M^{(k)}, \boldsymbol{\Sigma}_M^{(k)}) \end{pmatrix} =$$

$$
\begin{aligned}
&= \frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \begin{pmatrix} \alpha_1^{(k)} \sum_{j=1}^{M} \alpha_j^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)}) \\ \vdots \\ \alpha_M^{(k)} \sum_{j=1}^{M} \alpha_j^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)}) \end{pmatrix} \\
&= \frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \begin{pmatrix} \alpha_1^{(k)} p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)}) \\ \vdots \\ \alpha_M^{(k)} p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)}) \end{pmatrix} \\
&= \begin{pmatrix} \alpha_1^{(k)} \\ \vdots \\ \alpha_M^{(k)} \end{pmatrix} \\
&= \boldsymbol{\alpha}^{(k)}.
\end{aligned}
$$

Therefore,

$$
\boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}} = \frac{1}{N} \left(\mathrm{diag}(\alpha_1^{(k)}, \ldots, \alpha_M^{(k)}) - \boldsymbol{\alpha}^{(k)} \left(\boldsymbol{\alpha}^{(k)} \right)^T \right). \tag{43}
$$

By the definition of a positive definite matrix, the matrix $\boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}}$ is, by definition, positive definite if for all M-dimensional non-zero vectors $\boldsymbol{v}$, the inequality $\boldsymbol{v}^T \boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}} \boldsymbol{v} > 0$ holds. Therefore, by Equation (43), $\boldsymbol{A}^{LE}_{\boldsymbol{\alpha}^{(k)}}$ is positive definite if

$$
\boldsymbol{v}^T \mathrm{diag}(\alpha_1^{(k)}, \ldots, \alpha_M^{(k)}) \boldsymbol{v} > 0,
$$

and

$$
\boldsymbol{v}^T \mathrm{diag}(\alpha_1^{(k)}, \ldots, \alpha_M^{(k)}) \boldsymbol{v} > \boldsymbol{v}^T \boldsymbol{\alpha}^{(k)} \left(\boldsymbol{\alpha}^{(k)} \right)^T \boldsymbol{v}.
$$

The first condition is true since $\alpha_j > 0$ for $1 \leq j \leq M$. The second condition is also true since

$$
\boldsymbol{v}^T \mathrm{diag}(\alpha_1^{(k)}, \ldots, \alpha_M^{(k)}) \boldsymbol{v} = \sum_{j=1}^{M} \alpha_j^{(k)} v_j^2,
$$

and, because $\boldsymbol{\alpha}$ is a probability distribution and the function $f(x) = x^2$ is concave, we can apply Jensen's inequality to get

$$
\begin{aligned}
\boldsymbol{v}^T \mathrm{diag}(\alpha_1^{(k)}, \ldots, \alpha_M^{(k)}) \boldsymbol{v} &> \left(\sum_{j=1}^{M} \alpha_j^{(k)} v_j \right)^2 \\
&= \left(\sum_{j=1}^{M} v_j \alpha_j^{(k)} \right) \left(\sum_{j=1}^{M} \alpha_j^{(k)} v_j \right) \\
&= \boldsymbol{v}^T \boldsymbol{\alpha}^{(k)} \left(\boldsymbol{\alpha}^{(k)} \right)^T \boldsymbol{v}.
\end{aligned}
$$

Therefore, the matrix $\boldsymbol{A}^{LE}_{\vec{\alpha}^{(k)}}$ is positive definite given the constraints on $\boldsymbol{\alpha}$.

Similarly, we want to find a matrix $\boldsymbol{A}^{LE}_{\boldsymbol{\mu}_j^{(k)}}$ for $1 \leq j \leq M$ such that

$$\boldsymbol{\mu}_j^{(k)} + \boldsymbol{A}^{LE}_{\boldsymbol{\mu}_j^{(k)}} \boldsymbol{r}_{\boldsymbol{\mu}_j^{(k)}} = \frac{\sum_{i=1}^N \hat{z}_{i,j}^{(k)} \boldsymbol{x}_i}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}}.$$

As computed before, the gradient of the log likelihood function with respect to $\boldsymbol{\mu}_j$ evaluated at $\boldsymbol{\theta}^{(k)}$ is

$$\boldsymbol{r}_{\boldsymbol{\mu}_j^{(k)}} = \nabla_{\boldsymbol{\mu}_j} L(\boldsymbol{\theta}^{(k)}) = \sum_{i=1}^N \hat{z}_{i,j}^{(k)} \left(\boldsymbol{\Sigma}_j^{(k)}\right)^{-1} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)}\right).$$

Then

$$\sum_{i=1}^N \hat{z}_{i,j}^{(k)} \boldsymbol{A}^{LE}_{\boldsymbol{\mu}_j^{(k)}} \left(\boldsymbol{\Sigma}_j^{(k)}\right)^{-1} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)}\right) = \frac{\sum_{i=1}^N \hat{z}_{i,j}^{(k)} \boldsymbol{x}_i}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} - \boldsymbol{\mu}_j^{(k)},$$

$$\sum_{i=1}^N \hat{z}_{i,j}^{(k)} \left(\sum_{i=1}^N \hat{z}_{i,j}^{(k)}\right) \boldsymbol{A}^{LE}_{\boldsymbol{\mu}_j^{(k)}} \left(\boldsymbol{\Sigma}_j^{(k)}\right)^{-1} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)}\right) = \sum_{i=1}^N \hat{z}_{i,j}^{(k)} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)}\right).$$

The last equality holds if we let

$$\boldsymbol{A}^{LE}_{\boldsymbol{\mu}_j^{(k)}} = \frac{1}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} \boldsymbol{\Sigma}_j^{(k)}. \tag{44}$$

From Equation (44), the matrix $\boldsymbol{A}^{LE}_{\boldsymbol{\mu}_j^{(k)}}$ is positive definite since the covariance matrix $\boldsymbol{\Sigma}_j^{(k)}$ is positive definite (assuming that the update did not take us out of the constraint space) and the denominator in the factor is always positive for $1 \leq j \leq M$.

Similarly, for $\boldsymbol{A}^{LE}_{\boldsymbol{\Sigma}_j^{(k)}}$ and all j, we have

$$\text{vec}(\boldsymbol{\Sigma}_j^{(k)}) + \boldsymbol{A}^{LE}_{\boldsymbol{\Sigma}_j^{(k)}} \boldsymbol{r}_{\boldsymbol{\Sigma}_j^{(k)}} = \frac{\sum_{i=1}^N \hat{z}_{i,j}^{(k)} \text{vec}\left((\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})^T\right)}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}}.$$

The gradient of the log likelihood function with respect to $\boldsymbol{\Sigma}_j$ evaluated at $\boldsymbol{\theta}^{(k)}$ is

$$\begin{aligned}
\boldsymbol{r}_{\boldsymbol{\Sigma}_j^{(k)}} &= \nabla_{\boldsymbol{\Sigma}_j} L(\boldsymbol{\theta}^{(k)}) \\
&= -\frac{1}{2}\text{vec}\left(\left(\boldsymbol{\Sigma}_j^{(k)}\right)^{-1} \sum_{i=1}^N \frac{\alpha_j^{(k)} g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(k)}, \boldsymbol{\Sigma}_j^{(k)})}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{(k)})} \left(\boldsymbol{\Sigma}_j^{(k)} - (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})^T\right) \left(\boldsymbol{\Sigma}_j^{(k)}\right)^{-1}\right) \\
&= -\frac{1}{2}\text{vec}\left(\left(\boldsymbol{\Sigma}_j^{(k)}\right)^{-1} \sum_{i=1}^N \hat{z}_{i,j}^{(k)} \left(\boldsymbol{\Sigma}_j^{(k)} - (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})^T\right) \left(\boldsymbol{\Sigma}_j^{(k)}\right)^{-1}\right).
\end{aligned}$$

From the update rule for the covariance matrices in the iterative method, we have

$$\begin{aligned}
\boldsymbol{\Sigma}_j^{(k+1)} &= \boldsymbol{\Sigma}_j^{(k)} + \frac{\sum_{i=1}^N \hat{z}_{i,j}^{(k)} (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})^T}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} - \boldsymbol{\Sigma}_j^{(k)} \\
&= \boldsymbol{\Sigma}_j^{(k)} - \frac{\sum_{i=1}^N \hat{z}_{i,j}^{(k)} \left(\boldsymbol{\Sigma}_j^{(k)} - (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k+1)})^T \right)}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} \\
&= \boldsymbol{\Sigma}_j^{(k)} + \frac{2}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} \boldsymbol{\Sigma}_j^{(k)} \mathrm{mat}(\boldsymbol{r}_{\boldsymbol{\Sigma}_j^{(k)}}) \boldsymbol{\Sigma}_j^{(k)}.
\end{aligned}$$

The *Kronecker* product of a $(m \times n)$ matrix $\boldsymbol{A}$ and a $(q \times m)$ matrix $\boldsymbol{B}$ is the $(mq \times nm)$ matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2n}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{pmatrix}$$

Using the property

$$\mathrm{vec}(\boldsymbol{ABC}) = (\boldsymbol{C}^T \otimes \boldsymbol{A})\mathrm{vec}(\boldsymbol{B}),$$

we can rewrite the last expression for $\boldsymbol{\Sigma}_j^{(k)}$ and use it in the gradient-based expression of the update rule for the covariance matrices to obtain

$$\mathrm{vec}(\boldsymbol{\Sigma}_j^{(k)}) + \boldsymbol{A}_{\boldsymbol{\Sigma}_j^{(k)}}^{LE} \boldsymbol{r}_{\boldsymbol{\Sigma}_j^{(k)}} = \mathrm{vec}(\boldsymbol{\Sigma}_j^{(k)}) + \frac{2}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} \left(\boldsymbol{\Sigma}_j^{(k)} \otimes \boldsymbol{\Sigma}_j^{(k)} \right) \boldsymbol{r}_{\boldsymbol{\Sigma}_j^{(k)}},$$

or, finally,

$$\boldsymbol{A}_{\boldsymbol{\Sigma}_j^{(k)}}^{LE} = \frac{2}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} \left(\boldsymbol{\Sigma}_j^{(k)} \otimes \boldsymbol{\Sigma}_j^{(k)} \right).$$

Proving that $\boldsymbol{A}_{\boldsymbol{\Sigma}_j^{(k)}}^{LE}$ is positive definite with probability 1 for sufficiently large N requires some additional work which we do not present here. Refer Xu and Jordan [XJ96] for the details.

Now, we can relate the EM update rule for the covariance matrices to that of the iterative method by expressing the EM update rule as

$$\boldsymbol{\Sigma}_j^{(k+1)} = \frac{\sum_{i=1}^N \hat{z}_{i,j}^{(k)} (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(k)})^T}{\sum_{i=1}^N \hat{z}_{i,j}^{(k)}} + (\boldsymbol{\mu}_j^{(k)} - \boldsymbol{\mu}_j^{(k+1)})(\boldsymbol{\mu}_j^{(k)} - \boldsymbol{\mu}_j^{(k+1)})^T.$$

From this expression, following a similar derivation as for the iterative method, we can obtain

$$\mathrm{vec}(\boldsymbol{\Sigma}_j^{(k+1)}) = \mathrm{vec}(\boldsymbol{\Sigma}_j^{(k)}) + \boldsymbol{A}_{\boldsymbol{\Sigma}_j^{(k)}}^{LE} \boldsymbol{r}_{\boldsymbol{\Sigma}_j^{(k)}} + (\boldsymbol{\mu}_j^{(k)} - \boldsymbol{\mu}_j^{(k+1)}) \otimes (\boldsymbol{\mu}_j^{(k)} - \boldsymbol{\mu}_j^{(k+1)}).$$

The argument is now that the third term in the right hand side of the equation above is negligible when we are close to the solution [Xu97]. As a result of this argument, we have $\boldsymbol{A}^{EM}_{\boldsymbol{\Sigma}_j^{(k)}} \approx \boldsymbol{A}^{LE}_{\boldsymbol{\Sigma}_j^{(k)}}$.

Therefore, we can see EM as a gradient-based method where the direction is always a positive transformation of the gradient of the log-likelihood function evaluated at the current setting of the parameters. Since the transforming matrix changes from iteration to iteration, EM has the flavor of a quasi-Newton or variable metric method, with the transforming matrix being updated in a very particular way which guarantees both convergence and the satisfaction of the constraints.

5.2 Convergence properties of EM

There exist many results about the convergence properties of EM both general [MK97] and specific [RW84]. In this section we will mention some of them. By now we know of the monotonicity property of EM (i.e., the likelihood function is guaranteed not to decrease at each iteration). Under some conditions on the log likelihood, the monotonicity is strict except at a critical point of the likelihood function. Also, under some conditions on the log likelihood function, EM is guaranteed to converge to a critical point of the log likelihood function [DLR77, Wu83]. In general, the method has linear convergence when close to a solution, although it can show super-linear convergence in some instances [RW84, XJ96]. In practice, it converges in the log-likelihood space much faster than in the parameter space. For the case of a mixture of Gaussian model, the speed of convergence directly depends on the degree of separation between the Gaussians (See Figure 4). The experience in practice is that EM is very slow to converge to a solution when the Gaussians are very close together. Theoretically, it converges faster in likelihood space than in parameter space. As with any other MLE method, the theoretical and practical limits of accuracy of the parameters depend on the Fisher information matrix as presented in Section 3.

Using the relation between EM and gradient methods, we can establish bounds on the convergence rate of EM for the mixture-of-Gaussians model. We can find those bounds by letting $\gamma^{(k)} = 1$ and $\boldsymbol{A}^{(k)} = \boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}}$, which we found in Section (5.1), in the convergence rate bound results for the general form of gradient methods; that is,

$$\left\| \left((\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^N \right) \right\| \leq \left\| \boldsymbol{E}^T \left(\boldsymbol{I} + (\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \boldsymbol{H}(\boldsymbol{\theta}^N) (\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \right) \right\| \left\| \left((\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N \right) \right\|,$$

or,

$$\left\| \boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^N \right\| \leq \left\| \boldsymbol{E}^T \left(\boldsymbol{I} + \boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}} \boldsymbol{H}(\boldsymbol{\theta}^N) \right) \right\| \left\| \boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N \right\|,$$

where $\boldsymbol{E}^T$ is a projection matrix necessary to satisfy the constraints on the parameters. We can tight the bound above by performing further simplifications [XJ96, Xu97].

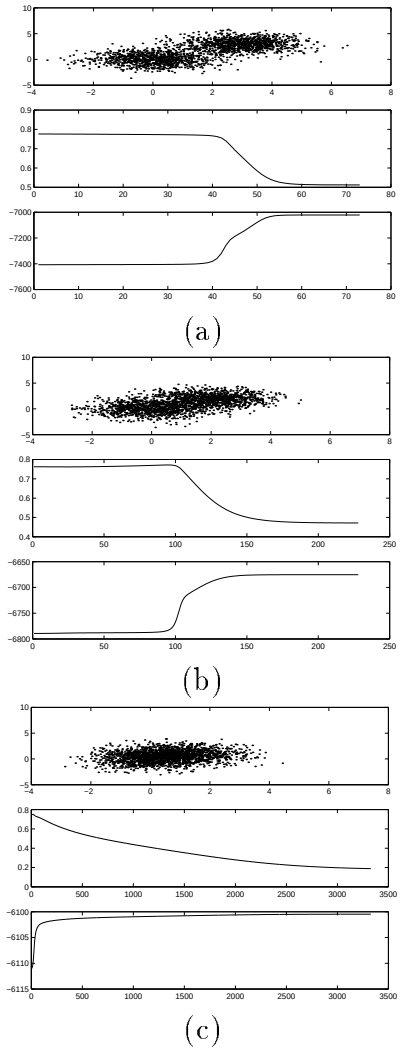


Figure 4: The plots in this figure suggest how the problem of estimation for the mixture-of-Gaussians model, in this case for $M = 2$ two-dimensional Gaussians, depends on the separation of the Gaussians for a fixed number of data points $N = 2000$. The true parameters are $\alpha_1 = \alpha_2 = 0.5, \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{I}, \boldsymbol{\mu}_1 = [0,0]^T$, and (a) $\boldsymbol{\mu}_2 = [3,3]^T$, (b) $\boldsymbol{\mu}_2 = [2,2]^T$, and (c) $\boldsymbol{\mu}_2 = [1,1]^T$. For all sub-figures, the top plot shows the data points. The middle plot shows the value of one of the mixing parameters as a function of the number of iterations. The bottom plot shows the value of the *average* log-likelihood function (i.e., $(1/N)L(\boldsymbol{\theta})$) also as a function of the number of iterations. The results are for a single run and we start each run with the same initial parameters. Other runs for different models with the same separation characteristics behave similarly.

Algorithm 1 Method of Conjugate directions

Select an initial direction $\boldsymbol{d}^{(0)}$
$k \leftarrow 0$
repeat
  $\gamma^{(k)} \leftarrow \arg_\gamma \max f(\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)})$
  $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{d}^{(k)}$
  Find new direction $\boldsymbol{d}^{(k+1)}$ that is orthogonal to all previous directions $\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(k)}$
  $k \leftarrow k + 1$
until stopping condition is met

6 Conjugate gradient

These methods are a little different from the previous gradient based methods. The reader should consult [Ber95, She94, Pol71] for more information on the subject. They were first developed for solving linear systems of equations. The idea behind them goes as follows. First, let us assume for the purpose of presenting the general idea that the function we want to maximize is convex. Assume we pick an initial point in the domain of the function and a direction or line along this point. Then we can find the point where the function has the largest value with respect to all the points along this line. Once we have selected such point we pick a different direction that is *orthogonal* (with respect to some matrix transforming the space) or *conjugate* to the direction we took in the previous step and find another point where the function attains a largest value with respect to all the points in the line along this direction. And again we find another direction along this third point that is conjugate to the first two directions taken and so on. Then after finding as many relative maximum points as there are parameters in the function (i.e., the dimensions of the domain of the function) we will be guaranteed to find the point in the domain where the function attains its largest value.

Now, the function we are maximizing is in general non-convex with respect to the parameters and therefore if we only took as many steps as there are dimensions we are not guaranteed to find a solution (i.e., the value of the function at the resulting point is not necessarily a maximum, or saddle point). If we let the number of dimensions of the domain of the function be d, then we iterate until we find a solution by restarting from a new base direction after we have taken d steps. Algorithm 1 gives a description of the *method of conjugate directions*.

We can use the *Gram-Schmidt procedure* to determine mutually $\boldsymbol{M}$-conjugate directions (i.e., conjugate directions with respect to some matrix $\boldsymbol{M}$). Given $k+1$ linearly independent vectors, $\boldsymbol{v}^{(0)}, \ldots, \boldsymbol{v}^{(k)}$, the procedure allows us to obtain $k+1$ mutually $\boldsymbol{M}$-conjugate directions, $\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(k)}$, such that $\boldsymbol{v}^{(0)}, \ldots, \boldsymbol{v}^{(k)}$ and $\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(k)}$ span the same subspace. The method goes as follows. First let $\boldsymbol{d}^{(0)} = \boldsymbol{v}^{(0)}$. Then assume we have computed $i < k$ mutually $\boldsymbol{M}$-conjugate directions $\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(i)}$ and take $\boldsymbol{d}^{(i+1)}$ to be of the form

$$\boldsymbol{d}^{(i+1)} = \boldsymbol{v}^{(i+1)} + \sum_{j=0}^{i} \beta^{(i+1,j)} \boldsymbol{d}^{(j)}.$$

By the definition of $\boldsymbol{M}$-conjugate directions, $\left(\boldsymbol{d}^{(i)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(i)} = 0$, for all $i \neq j$. Therefore, we can find values for $\beta^{(i+1,j)}$, $0 \leq j \leq i$, such that $\boldsymbol{d}^{(i+1)}$ is $\boldsymbol{M}$-conjugate to $\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(i)}$ by solving the equation, for $0 \leq l \leq i$,

$$\left(\boldsymbol{d}^{(i+1)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)} = \left(\boldsymbol{v}^{(i+1)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)} + \left(\sum_{j=0}^{i} \beta^{(i+1,j)}\boldsymbol{d}^{(j)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)} = 0.$$

Since the directions $\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(i)}$ are mutually $\boldsymbol{M}$-conjugate, we have $\left(\boldsymbol{d}^{(j)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)} = 0$ if $l \neq j$. From this we get,

$$\left(\boldsymbol{v}^{(i+1)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)} + \beta^{(i+1,l)} \left(\boldsymbol{d}^{(l)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)} = 0,$$

or

$$\beta^{(i+1,l)} = -\frac{\left(\boldsymbol{v}^{(i+1)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)}}{\left(\boldsymbol{d}^{(l)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)}}.$$

Intuitively, the procedure is computing new directions by taking a new vector and eliminating all the components resulting from projecting the vector (with respect to the matrix $\boldsymbol{M}$) into the old directions.

The conjugate gradient method results from applying the method of conjugate directions where we obtain the conjugate directions by applying the Gram-Schmidt procedure. In addition, the vectors used to generate the conjugate directions are the gradients evaluated at the estimate to the solution at each iteration. Interestingly, the Gram-Schmidt process in this case reduces to computing the coefficient with respect to the last direction, since all the remaining coefficients are zero. We can see this from the following results. Let the quadratic function be $f(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^T \boldsymbol{M}\boldsymbol{\theta} - \boldsymbol{b}^T\boldsymbol{\theta} + c$, where the matrix $\boldsymbol{M}$ is symmetric, negative definite, $\boldsymbol{b}$ is a vector constant, and c is a scalar constant. Then $\boldsymbol{r}^{(k)} = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(k)}) = \boldsymbol{M}\boldsymbol{\theta}^{(k)} - \boldsymbol{b}$. First of all, from the step of the method of conjugate directions in which we minimize along the conjugate direction $\boldsymbol{d}^{(k)}$, we have

$$\left.\frac{\partial f(\boldsymbol{\theta}^{(k+1)})}{\partial \gamma}\right|_{\gamma=\gamma^{(k)}} = \left.\frac{\partial f(\boldsymbol{\theta}^{(k)} + \gamma\boldsymbol{d}^{(k)})}{\partial \gamma}\right|_{\gamma=\gamma^{(k)}} = \left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{d}^{(k)} = 0.$$

Therefore, for $0 \leq i < k$,

$$\begin{aligned}
\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{d}^{(i)} &= \left(\boldsymbol{M}\boldsymbol{\theta}^{(k+1)} - \boldsymbol{b}\right)^T \boldsymbol{d}^{(i)} \\
&= \left(\boldsymbol{\theta}^{(i+1)} + \sum_{j=i+1}^{k} \gamma^{(j)}\boldsymbol{d}^{(j)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(i)} - \boldsymbol{b}^T\boldsymbol{d}^{(i)} \\
&= \left(\boldsymbol{\theta}^{(i+1)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(i)} - \boldsymbol{b}^T\boldsymbol{d}^{(i)} \\
&= \left(\boldsymbol{r}^{(i+1)}\right)^T \boldsymbol{d}^{(i)} \\
&= 0.
\end{aligned}$$

Hence, $\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{d}^{(i)} = 0$ for $0 \leq i < k$. Using this result, we get, for $i < l$,

$$\begin{aligned} \left(\boldsymbol{r}^{(l)}\right)^T \boldsymbol{d}^{(i)} &= \left(\boldsymbol{r}^{(l)}\right)^T \boldsymbol{v}^{(i)} - \sum_{j=0}^{i-1} \beta^{(i,j)} \left(\boldsymbol{r}^{(l)}\right)^T \boldsymbol{d}^{(j)} \\ 0 &= \left(\boldsymbol{r}^{(l)}\right)^T \boldsymbol{v}^{(i)}. \end{aligned}$$

Now,

$$\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)} = \boldsymbol{M}\left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\right) = \boldsymbol{M}\gamma^{(k)}\boldsymbol{d}^{(k)}.$$

Therefore, for $j < i+1$, we have

$$\begin{aligned} \left(\boldsymbol{r}^{(i+1)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(j)} &= \frac{1}{\gamma^{(j)}} \left(\boldsymbol{r}^{(i+1)}\right)^T \left(\boldsymbol{r}^{(j+1)} - \boldsymbol{r}^{(j)}\right) \\ &= \begin{cases} 0 & \text{if } 0 \leq j \leq i-1, \\ \frac{1}{\gamma^{(i)}} \left(\boldsymbol{r}^{(i+1)}\right)^T \boldsymbol{r}^{(i+1)} & \text{if } j = i. \end{cases} \end{aligned}.$$

Similarly, we have

$$\left(\boldsymbol{d}^{(i)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(i)} = \frac{1}{\gamma^{(i)}} \left(\boldsymbol{d}^{(i)}\right)^T \left(\boldsymbol{r}^{(i+1)} - \boldsymbol{r}^{(i)}\right).$$

Using $\boldsymbol{v}^{(i)} = \boldsymbol{r}^{(i)}$ in the Gram-Schmidt process, we obtain

$$\begin{aligned} \beta^{(i+1,l)} &= -\frac{\left(\boldsymbol{r}^{(i+1)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)}}{\left(\boldsymbol{d}^{(l)}\right)^T \boldsymbol{M}\boldsymbol{d}^{(l)}} \\ &= \begin{cases} 0 & \text{if } 0 \leq l \leq i-1, \\ -\frac{\left(\boldsymbol{r}^{(i+1)}\right)^T \boldsymbol{r}^{(i+1)}}{\left(\boldsymbol{d}^{(i)}\right)^T \left(\boldsymbol{r}^{(i+1)} - \boldsymbol{r}^{(i)}\right)} & \text{if } l = i. \end{cases} \end{aligned}$$

From this we have

$$\beta^{(k+1)} \equiv \beta^{(k+1,k)} = -\frac{\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{r}^{(k+1)}}{\left(\boldsymbol{d}^{(k)}\right)^T \left(\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)}\right)}. \tag{45}$$

But, we can simplify the denominator of the last expression as

$$\begin{aligned} \left(\boldsymbol{d}^{(k)}\right)^T \left(\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)}\right) &= \left(\boldsymbol{r}^{(k)} + \beta^{(k)}\boldsymbol{d}^{(k-1)}\right)^T \left(\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)}\right) \\ &= \left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k+1)} + \beta^{(k)} \left(\boldsymbol{d}^{(k-1)}\right)^T \boldsymbol{r}^{(k+1)} + \\ &\quad - \left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)} - \beta^{(k)} \left(\boldsymbol{d}^{(k-1)}\right)^T \boldsymbol{r}^{(k)} \\ &= -\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)}, \end{aligned}$$

Algorithm 2 Conjugate gradient method

Initialize $\boldsymbol{\theta}^{(0)}$ somehow
$\boldsymbol{r}^{(0)} \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(0)})$
$\boldsymbol{d}^{(0)} \leftarrow \boldsymbol{r}^{(0)}$
$k \leftarrow 0$
repeat
  $\gamma^{(k)} \leftarrow \arg_{\gamma} \max f(\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)})$
  $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{d}^{(k)}$
  $\boldsymbol{r}^{(k+1)} \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(k+1)})$
  $\beta^{(k+1)} \leftarrow \frac{(\boldsymbol{r}^{(k+1)})^T \boldsymbol{r}^{(k+1)}}{(\boldsymbol{r}^{(k)})^T \boldsymbol{r}^{(k)}}$ or $\beta^{(k+1)} \leftarrow \frac{(\boldsymbol{r}^{(k+1)})^T (\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)})}{(\boldsymbol{r}^{(k)})^T \boldsymbol{r}^{(k)}}$
  if $((k+1) \mod d) == 0$ **then**
    $\boldsymbol{d}^{(k+1)} \leftarrow \boldsymbol{r}^{(k+1)}$
  else
    $\boldsymbol{d}^{(k+1)} \leftarrow \boldsymbol{r}^{(k+1)} + \beta^{(k+1)} \boldsymbol{d}^{(k)}$
  end if
  $k \leftarrow k + 1$
until stopping condition is met

because all the other terms are zero (from previous results). Finally, we obtain

$$\beta^{(k+1)} = \frac{\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{r}^{(k+1)}}{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)}}.$$

Therefore, all we need to remember is the last direction in order to compute the new direction. Algorithm 2 gives a description of the conjugate gradient method. Again, by the way we compute $\beta^{(k+1)}$, each new direction $\boldsymbol{d}^{(k+1)}$ is guaranteed to be conjugate to all previous directions already taken (i.e., $\boldsymbol{d}^{(i)}$ for $0 \leq i < k+1$). One of the problems of this method is that for the case of non-linear optimization, the problem of performing the *line search* or minimization along a direction $\boldsymbol{d}^{(k)}$ is typically very hard. We can use several methods to accomplish this. Hence, even though the procedure converges after performing a fixed number of line searches if the function is convex, each line search can take a long time. Also, even if the function is convex, we sometimes need to perform more iterations than the number of parameters we want to estimate because numerical errors that can arise during the computation of the conjugate directions $\boldsymbol{d}^{(k+1)}$ and estimation errors from the line searches make the directions lose conjugacy. Because of the directions losing conjugacy, which arises in practice, we have two different methods to determine the new conjugate directions. One is *Fletcher-Reeves* [Pol71, She94, PTVF92] and the other *Polak-Ribière* [Pol71, She94, PTVF92]. We obtain Fletcher-Reeves if we use the update for $\beta^{(k+1)}$ on the left hand-side as presented in the description of the conjugate gradient method. Using the update presented at the right-hand side we get Polak-Ribière. They are mathematically equivalent with respect to finding conjugate directions and those directions are the same if the function has a quadratic form [She94, PTVF92]. However, in practice, they are different and it has been suggested that Polak-Ribière converges faster [She94, PTVF92] because, when it is not doing much progress, it tends to yield $\beta^{(k)} \approx 0$; hence, it produces a new

direction that is equal to the gradient, essentially restarting the method (i.e., $\boldsymbol{d}^{(k+1)} \leftarrow \boldsymbol{r}^{(k+1)}$) [PTVF92].

Not surprisingly, conjugate gradient methods and gradient based methods share similar characteristics in general. They have only local convergence guarantees and the rate of convergence is in general linear [Ber95]. Also, note that conjugate gradient methods as presented here are unconstrained optimization methods and therefore they need to be adapted in order to use them for our problem, where the mixture-of-Gaussians model imposes constraints on its parameters (i.e., $\boldsymbol{\alpha}$, and $\boldsymbol{\Sigma_j}$ for all j).

As in gradient methods, there is a more general form for the conjugate gradient methods which we can obtain by performing regular conjugate gradient (as presented above) in a transformed space. That is, let

$$\boldsymbol{\theta} = \boldsymbol{T}\boldsymbol{\phi},$$

where $\boldsymbol{T}$ is a invertible symmetric matrix. Now, let

$$h(\boldsymbol{\phi}) = f(\boldsymbol{T}\boldsymbol{\phi}) = \frac{1}{2}\boldsymbol{\phi}^T\boldsymbol{T}\boldsymbol{M}\boldsymbol{T}\boldsymbol{\phi} - \boldsymbol{b}^T\boldsymbol{T}\boldsymbol{\phi} + c.$$

Then, each iteration is now

$$\boldsymbol{\phi}^{(k+1)} = \boldsymbol{\phi}^{(k)} + \gamma^{(k)}\tilde{\boldsymbol{d}}^{(k)},$$

with

$$\tilde{\boldsymbol{d}}^{(0)} = \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(0)}),$$

and, for all k,

$$\tilde{\boldsymbol{d}}^{(k+1)} = \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k+1)}) + \beta^{(k+1)}\tilde{\boldsymbol{d}}^{(k)},$$

with,

$$\beta^{(k+1)} = \frac{\left(\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k+1)})\right)^T \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k+1)})}{\left(\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k)})\right)^T \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k)})}.$$

Since now we have

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k)}) &= \boldsymbol{T}\boldsymbol{M}\boldsymbol{T}\boldsymbol{\phi} - \boldsymbol{T}\boldsymbol{b} \\ &= \boldsymbol{T}\boldsymbol{r}^{(k)}, \end{aligned}$$

if we let $\boldsymbol{d}^{(k)} = \boldsymbol{T}\tilde{\boldsymbol{d}}^{(k)}$ and $\boldsymbol{H} = \boldsymbol{T}^2$, then we obtain

$$\begin{aligned} \boldsymbol{d}^{(0)} &= \boldsymbol{H}\boldsymbol{r}^{(0)}, \\ \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} + \gamma^{(k)}\boldsymbol{d}^{(k)}, \\ \beta^{(k+1)} &= \frac{\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{H}\boldsymbol{r}^{(k+1)}}{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{H}\boldsymbol{r}^{(k)}}, \\ \boldsymbol{d}^{(k+1)} &= \boldsymbol{H}\boldsymbol{r}^{(k+1)} + \beta^{(k+1)}\boldsymbol{d}^{(k)}. \end{aligned}$$

Algorithm 3 Generalized conjugate gradient method with preconditioning matrix $\boldsymbol{H}$

Initialize $\boldsymbol{\theta}^{(0)}$ somehow
$\boldsymbol{r}^{(0)} \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(0)})$
$\boldsymbol{g}^{(0)} \leftarrow \boldsymbol{H}\boldsymbol{r}^{(0)}$
$\boldsymbol{d}^{(0)} \leftarrow \boldsymbol{g}^{(0)}$
$k \leftarrow 0$
repeat
  $\gamma^{(k)} \leftarrow \arg_{\gamma} \max f(\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)})$
  $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{d}^{(k)}$
  $\boldsymbol{r}^{(k+1)} \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(k+1)})$
  $\boldsymbol{g}^{(k+1)} \leftarrow \boldsymbol{H}\boldsymbol{r}^{(k+1)}$
  $\beta^{(k+1)} \leftarrow \frac{(\boldsymbol{g}^{(k+1)})^T \boldsymbol{r}^{(k+1)}}{(\boldsymbol{g}^{(k)})^T \boldsymbol{r}^{(k)}}$ or $\beta^{(k+1)} \leftarrow \frac{(\boldsymbol{g}^{(k+1)})^T (\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)})}{(\boldsymbol{g}^{(k)})^T \boldsymbol{r}^{(k)}}$
  if $((k+1) \mod d) == 0$ **then**
    $\boldsymbol{d}^{(k+1)} \leftarrow \boldsymbol{g}^{(k+1)}$
  else
    $\boldsymbol{d}^{(k+1)} \leftarrow \boldsymbol{g}^{(k+1)} + \beta^{(k+1)} \boldsymbol{d}^{(k)}$
  end if
  $k \leftarrow k + 1$
until stopping condition is met

The resulting general form is called *generalized conjugate gradient*. The matrix $\boldsymbol{H}$ associated with the transformation of the space is called a *preconditioning matrix*. Actually, we can see regular conjugate gradient as a generalized conjugate gradient method with preconditioning matrix equal to the identity matrix. Using a *good* preconditioning matrix results in better performance than regular conjugate gradient methods. Therefore, selecting a good preconditioning matrix is an important part of using the generalized conjugate gradient algorithm [JJ93]. A convergence rate analysis of conjugate gradient similar to that performed for the general form of gradient methods suggests that the most appropriate preconditioning matrix is the negative of the inverse of the Hessian of the log likelihood evaluated at the solution. As in gradient-based methods, this is impossible since we do not know what the solution is. Several heuristics exist for the selection of a *good* preconditioning matrix. Which one is better depends on the nature of the problem. Algorithm 3 describes the generalized conjugate gradient method.

7 Accelerations of EM

As we mentioned above, there are some instances where EM shows slow convergence. Several techniques have been proposed to alleviate this problem; we present two such techniques in this section. The first one results from the Aitken acceleration method. The other borrows from the generalized conjugate gradient method.

7.1 Aitken acceleration: Line search and Parameterized EM

A typical acceleration method for EM can be derived from the Aitken acceleration method [3] as follows. Suppose we have an update rule $\boldsymbol{u}(\boldsymbol{\theta})$ and an iterative method $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{u}(\boldsymbol{\theta}^{(k)})$ associated with it that converges to $\boldsymbol{\theta}^N$. Let $\boldsymbol{I}$ be the identity matrix. Then we can rewrite the method as

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \boldsymbol{I}(\boldsymbol{u}(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)}).$$

Note that we can view the rewrite as a gradient-based method where $\gamma^{(k)} = 1$, $\boldsymbol{A}^{(k)} = \boldsymbol{I}$ and $\boldsymbol{r}^{(k)} = \boldsymbol{u}(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)}$. Therefore we are looking for the zero of the vector-valued function $\boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{u}(\boldsymbol{\theta}) - \boldsymbol{\theta}$. Since we assumed that the iterative method associated with the update rule converges, then the identity matrix must be sufficiently close to the Jacobian of the function $\boldsymbol{f}$.

Note that EM converges and therefore we can use this method. If we let $\boldsymbol{u}_{EM}(\boldsymbol{\theta}^{(k)})$ be the EM update rule, then we let

$$\boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{A}_{\boldsymbol{\theta}}^{EM} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \boldsymbol{u}_{EM}(\boldsymbol{\theta}) - \boldsymbol{\theta}$$

to obtain the acceleration method.

We can improve the speed of convergence of the accelerated method just presented by finding better approximation to the Jacobian of $\boldsymbol{f}$ than that of the identity matrix. We can theoretically use the actual Jacobian of $\boldsymbol{f}$ or some modified version of it. If we use the true Jacobian we will get Newton-Raphson method, or if we use a modified version we get any of its other variants including some of the gradient-based methods described before. One typical approximation, similar to the one we used for a gradient ascent method, is to approximate the Jacobian of $\boldsymbol{f}$ by $\gamma^{(k)}\boldsymbol{I}$ instead of just $\boldsymbol{I}$, where $\gamma^{(k)}$ has some of the properties described above in Section 4.3. If we do that and rearrange using algebra we will get the acceleration method

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma^{(k)}(\boldsymbol{u}_{EM}(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)}).$$

Intuitively, the value of the parameter $\gamma^{(k)}$ should be greater than zero. Actually, this method is also called a *line search* because, as in the general gradient method, we can try to find the optimal value of $\gamma^{(k)}$. In particular, the direction along which the value of $\gamma^{(k)}$ is optimized is $\boldsymbol{A}_{\boldsymbol{\theta}}^{EM} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \boldsymbol{u}_{EM}(\boldsymbol{\theta}) - \boldsymbol{\theta}$. Unfortunately, trying to find an optimal parameter $\gamma^{(k)}$ at each iteration is often hard. Instead, we typically set the parameter to be constant throughout or define a prestablished way of computing a sequence for it. Values for $\gamma^{(k)}$ smaller than one typically slow down convergence while values larger than one speed up convergence, with the penalty that convergence is no longer guaranteed unless $\gamma^{(k)}$ is updated appropriately. For the mixture of Gaussians model, the optimal value of $\gamma^{(k)}$ is always greater than one, lies very close to one when the Gaussian are well separated and cannot be much smaller than two when they are not [PW78, RW84]. Convergence is guaranteed if we start close to

[3] We base the following description on information given in Appendix A of [Mei89]. We use this explanation because it gives a high-level intuition for the method. [MK97] provides a more mathematically precise derivation.

a local maximum and $0 < \gamma^{(k)} < 2$. Note that the value of $\gamma^{(k)}$ needs to be such that $\boldsymbol{\theta}^{(k+1)}$ satisfy the constraint on the parameters. We can express bounds on the theoretical, local convergence rate of this acceleration method by letting $\boldsymbol{A}^{(k)} = \gamma^{(k)} \boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}}$ in the convergence rate bound results for the very general form of gradient methods; that is,

$$\left\| \left((\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{N} \right) \right\| \leq \left\| \boldsymbol{E}^T \left(\boldsymbol{I} + \gamma^{(k)} (\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \boldsymbol{H}(\boldsymbol{\theta}^{N}) (\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \right) \right\| \left\| \left((\boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}})^{\frac{1}{2}} \right)^{-1} \left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{N} \right) \right\|,$$

or,

$$\left\| \boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{N} \right\| \leq \left\| \boldsymbol{E}^T \left(\boldsymbol{I} + \gamma^{(k)} \boldsymbol{A}^{EM}_{\boldsymbol{\theta}^{(k)}} \boldsymbol{H}(\boldsymbol{\theta}^{N}) \right) \right\| \left\| \boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{N} \right\|,$$

where $\boldsymbol{E}^T$ is a projection matrix necessary to satisfy the constraints on the parameters. We can tight on the resulting bound further by more specific simplifications. In addition, we can speed up convergence if $\gamma^{(k)} > 0.5$ [Xu97].

Rewriting this rule provides us with an intuition of how it works [PW78, RW84],

$$\boldsymbol{\theta}^{(k+1)} = \gamma^{(k)} \boldsymbol{u}_{EM}(\boldsymbol{\theta}^{(k)}) + (1 - \gamma^{(k)}) \boldsymbol{\theta}^{(k)}.$$

This update rule computes the new parameters as a weighted combination of the old parameters and the EM parameters. If the parameter $\gamma^{(k)}$ is greater than one, then we extrapolate the value suggested by EM and give negative weight to our current parameter setting. If it is less than one then the new setting of the parameters is a weighted average of the setting suggested by EM and the current ones. Finally, if the parameter $\gamma^{(k)}$ is equal to one we get exactly EM.

A very similar EM update rule derived from a framework applied to the Bayesian Network model with discrete variables bears the name *parameterized EM* [BKS97]. The idea is that we have a current MLE for the parameters from some previous data and we get new data. Now we do not want to completely throw away our current estimates for the parameters, since we believe they are useful. Therefore, we learn a new setting of the parameters from the new data that is good with respect to the new data but is not too far away from the current setting. In short, we want a tradeoff between how good the estimates are with respect to the new data and how far away they are from the current estimates.

Although we can use the framework described above for batch learning, the framework is mainly intended for on-line parameter estimation where we keep receiving data continuously and need to update our parameters accordingly without having to go back and consider all the data we have received so far. More formally, assume we already have a maximum likelihood estimate for the parameters $\hat{\boldsymbol{\theta}}$ from previous data we had gathered and assume we get new data $\tilde{\boldsymbol{D}}$. Let $L_{\tilde{\boldsymbol{D}}}(\tilde{\boldsymbol{\theta}})$ be the log likelihood function of the parameters $\tilde{\boldsymbol{\theta}}$ with respect to the new data $\boldsymbol{D}'$. Let $d(\tilde{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}})$ be some measure of distance between some new setting for the parameters $\tilde{\boldsymbol{\theta}}$ and the old estimates $\hat{\boldsymbol{\theta}}$. Then, for a fixed positive number η, we want to find new parameters such that we maximize the function

$$F(\tilde{\boldsymbol{\theta}}) = \eta (1/N) L_{\tilde{\boldsymbol{D}}}(\tilde{\boldsymbol{\theta}}) - d(\tilde{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}})$$

with respect to $\tilde{\boldsymbol{\theta}}$. Note that $(1/N) L_{\tilde{\boldsymbol{D}}}(\tilde{\boldsymbol{\theta}})$ is the *normalized* or *average* value of the log likelihood function with respect to the new data. For the parameters of that model, if we

let the distance be the χ^2 distance between two distributions, then we will obtain a rule equivalent to the one presented above, but for the EM update rule associated with the Bayesian Networks model with discrete variables and with the step size parameter $\gamma^{(k)} = \eta$.

7.2 Conjugate gradient acceleration of EM

The conjugate gradient acceleration of EM proposed by Jamshidian and Jennrich [JJ93] is based on the idea that in a neighborhood close to a solution $\boldsymbol{\theta}^N$, the change in parameters after an EM iteration is approximately the generalized gradient defined by the negative of the inverse of the Hessian of the function $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^N)$ with respect to $\boldsymbol{\theta}$, evaluated at $\boldsymbol{\theta}^N$. The function Q is as defined in the expectation step of EM (i.e., $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{'}) = E_{\boldsymbol{Z}}[\ln p(\boldsymbol{Z}, \boldsymbol{D} \mid \boldsymbol{\theta}) \mid \boldsymbol{D}, \boldsymbol{\theta}^{'}]$), More precisely,

$$\boldsymbol{u}_{EM}(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)} \approx -\left(\nabla^2_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^N)\right)^{-1} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(k)}). \tag{46}$$

Note that as we mentioned in Section 6, in the generalized conjugate gradient method, we would like our preconditioning matrix to be as close as possible to the negative of the inverse of the Hessian of the function we are optimizing, evaluated at the solution. In our case, we would like our preconditioning matrix to be

$$\boldsymbol{H} = -\left(\nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^N)\right)^{-1}.$$

Instead, as an approximation to that matrix, we are going to use the preconditioning matrix

$$\boldsymbol{H} = -\left(\nabla^2_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}^N \mid \boldsymbol{\theta}^N)\right)^{-1}. \tag{47}$$

Note that the importance of this approximation stems from the fact that we do not really need to compute the preconditioning matrix or perform any matrix-vector operations since the change in parameters provided by the EM iteration provides an approximation to the generalized gradient. The conjugate gradient acceleration of EM results from applying the generalized conjugate gradient with the preconditioning matrix given in equation (47). Algorithm 4 presents a description of the conjugate gradient acceleration of EM. Note that the way in which we compute the coefficient $\beta^{(k+1)}$ differs from that of the Fletcher-Reeves or Polack-Ribière presented in the algorithms for conjugate gradient before. Note that this expression for $\beta^{(k+1)}$ is the Polack-Ribière-like, generalized version (i.e., obtained from performing regular conjugate gradient on a transformed space) of the expression given in equation (45), which is an unsimplified expression for $\beta^{(k+1)}$; that is, using some of the notation from the description of the generalized conjugate gradient method presented in

Algorithm 4 Conjugate gradient acceleration of EM

Initialize $\boldsymbol{\theta}^{(0)}$ (i.e., the result of performing a number of EM steps)
$\boldsymbol{r}^{(0)} \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(0)})$ and $\boldsymbol{\theta}_{EM}^{(1)} \leftarrow \boldsymbol{u}_{EM}(\boldsymbol{\theta}^{(0)})$
$\boldsymbol{g}^{(0)} \leftarrow \boldsymbol{\theta}_{EM}^{(1)} - \boldsymbol{\theta}^{(0)}$
$\boldsymbol{d}^{(0)} \leftarrow \boldsymbol{g}^{(0)}$
$k \leftarrow 0$
repeat
  $\gamma^{(k)} \leftarrow \arg_{\gamma} \max f(\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)})$
  $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \gamma^{(k)} \boldsymbol{d}^{(k)}$
  $\boldsymbol{r}^{(k+1)} \leftarrow \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(k+1)})$ and $\boldsymbol{\theta}_{EM}^{(k+2)} \leftarrow \boldsymbol{u}_{EM}(\boldsymbol{\theta}^{(k+1)})$
  $\boldsymbol{g}^{(k+1)} \leftarrow \boldsymbol{\theta}_{EM}^{(k+2)} - \boldsymbol{\theta}^{(k+1)}$
  $\beta^{(k+1)} \leftarrow -\frac{(\boldsymbol{g}^{(k+1)})^T (\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)})}{(\boldsymbol{d}^{(k)})^T (\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)})}$
  if $((k+1) \mod d == 0)$ **then**
    $\boldsymbol{d}^{(k+1)} \leftarrow \boldsymbol{g}^{(k+1)}$
  else
    $\boldsymbol{d}^{(k+1)} \leftarrow \boldsymbol{g}^{(k+1)} + \beta^{(k+1)} \boldsymbol{d}^{(k)}$
  end if
  $k \leftarrow k + 1$
until stopping condition is met

Section (6), we have

$$\begin{aligned}
\beta^{(k+1)} &= -\frac{\left(\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k+1)})\right)^T \left(\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k+1)}) - \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k)})\right)}{\left(\tilde{\boldsymbol{d}}^{(k)}\right)^T \left(\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k+1)}) - \nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}^{(k)})\right)} \\
&= -\frac{\left(\boldsymbol{T}\boldsymbol{r}^{(k+1)}\right)^T \left(\boldsymbol{T}\boldsymbol{r}^{(k+1)} - \boldsymbol{T}\boldsymbol{r}^{(k)}\right)}{\left(\boldsymbol{T}^{-1}\boldsymbol{d}^{(k)}\right)^T \left(\boldsymbol{T}\boldsymbol{r}^{(k+1)} - \boldsymbol{T}\boldsymbol{r}^{(k)}\right)} \\
&= -\frac{\left(\boldsymbol{r}^{(k+1)}\right)^T \boldsymbol{H} \left(\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)}\right)}{\left(\boldsymbol{d}^{(k)}\right)^T \left(\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)}\right)}.
\end{aligned}$$

Jamshidian and Jennrich [JJ93] say that computing $\beta^{(k+1)}$ in this way has the advantage that, for a quadratic function, the directions $\boldsymbol{d}^{(k+1)}$ and $\boldsymbol{d}^{(k)}$ are conjugate even if the line searches are not exact. Note that, given the similarity of the computation of the gradient of the log likelihood and the computation of the EM iteration, we can compute both the gradient and the generalized gradient at the same time without performing much more additional computation than that of computing the EM iteration alone, where the extra amount of computation is insignificant if N, the number of data points, is much larger than M, the number of Gaussians in the model, and d^3, where d is the dimension of the data points.

8 Experiments

As we have seen, there are many methods and variants of them that we can use to learn the parameters of the mixture-of-Gaussians model from data. However, in general, no one is theoretically dominant. Their speed of convergence will depend on the nature of the problem. One thing we know is that their speed of convergence is slower when the Gaussians are not well separated, because of the flatness of the surface; in other words, because the condition number of the Hessian of the log likelihood function at the solution is larger. Therefore we decided to experiment with some of those methods and test them with problems with varying separation.

For simplicity, we implemented the methods we tested in Matlab. Instead of comparing running times or CPU times, we compare the number of calls to compute the gradient of the log likelihood or to compute an EM iteration. Our basis for comparison is that, for the mixture of Gaussian model, when N, the number of data points, is much larger than M, the number of Gaussians, and d^3, where d is the dimension of the data, the computation of the gradient of the log likelihood function and the EM iteration require about the same computation. Also, given the similarity of their respective computations, we can compute both the gradient and the EM iteration with not much significant extra computation. We consider all three types of computations an *EM-equivalent iteration.*

Xu and Jordan [XJ96] already presented empirical results suggesting that EM is better than gradient ascent for finding the maximum likelihood estimate for the mixture-of-Gaussians model. In this paper, we are particularly interested in empirically comparing Aitken-based and the conjugate gradient accelerations of EM with regular EM. This is because they only require the computation of gradient and EM iterations, or, as we mention before, EM-equivalent iterations.

Other proposed accelerations of EM based on Newton and Quasi-Newton accelerations exist. However, we do not know of any study comparing the practical speed up, taking into account the extra computation of the Hessians or their approximations. The main difficulty with comparing EM with methods that use higher order derivatives (i.e., Hessians) and require the computation of matrices of size squared the number of parameters, like Newton and Quasi-Newton methods, is precisely in establishing the basis for comparison. For instance, we can compare them using running times. However, we believe this requires a very careful implementation of the various methods and a significant effort on optimizing the implementation. In addition to that, the space required for the execution of such methods significantly increases with the number of parameters in the model (i.e., the number of Gaussians and the dimensionality of the input), and therefore, their scalability becomes an issue when we are dealing with complex, real problems. Based on the statements above, we decided not to include methods of this kind in our study.

8.1 Methods tested

We present below the pseudo-code for the algorithms we tested. We adapted all the algorithms to perform only the necessary computations with respect to an EM-equivalent iteration.

8.1.1 EM

Algorithm 5 gives a description of the EM method we used. We discuss the stopping criterion we used later in the paper.

Algorithm 5 EM

Initialize $\boldsymbol{\alpha}^{(0)}$, and, for all $1 \leq j \leq M$, initialize $\boldsymbol{\mu}_j^{(0)}$ and $\boldsymbol{\Sigma}_j^{(0)}$
$k = 0$
repeat
  For all $1 \leq i \leq N$, $1 \leq j \leq M$, compute $\hat{z}_{i,j}^{(k)}$ using equation (37)
  For all $1 \leq j \leq M$, compute $\alpha_j^{(k+1)}$, $\boldsymbol{\mu}_j^{(k+1)}$, and $\boldsymbol{\Sigma}_j^{(k+1)}$ using equations (40), (41), and (42), respectively.
  $k \leftarrow k + 1$
until stopping condition is met

8.1.2 GEM (Generalized EM)

The name of this method will become apparent soon. In this method we try to eliminate the constraints on $\boldsymbol{\alpha}$ by reparameterizing them as follows: for $1 \leq j \leq M$,

$$\alpha_j(\boldsymbol{\omega}) = \frac{e^{\omega_j}}{\sum_{k=1}^{M} e^{\omega_k}}, \tag{48}$$

where now $\boldsymbol{\omega} \in \Re^M$ and therefore unconstrained. Doing this, the expression for the parameters in our model becomes

$$\boldsymbol{\theta} = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_M \\ \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_M \\ \boldsymbol{\Sigma}_1 \\ \vdots \\ \boldsymbol{\Sigma}_M \end{pmatrix}. \tag{49}$$

Note that this is not really necessary for regular EM, since it takes into account the constraints on $\boldsymbol{\alpha}$ automatically. However, this is one way of making the problem (or at least part of it) an unconstrained optimization problem, which can be useful when we use other unconstrained optimization methods.

Using the new expression for the parameters, we can rewrite the expression for the probability of a point given the parameters in the model as

$$p(\boldsymbol{x}_i \mid \boldsymbol{\theta}) = \sum_{j=1}^{M} \alpha_j(\boldsymbol{\omega}) g(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j). \tag{50}$$

From the EM update rules, we know already that the update rule for $\boldsymbol{\alpha}$ is

$$\boldsymbol{\alpha}^{(k+1)}(\boldsymbol{\omega}) = \frac{1}{N}\sum_{i=1}^{N} \hat{\boldsymbol{z}}_i^{(k)}. \tag{51}$$

Since $\boldsymbol{\alpha}$ is now a function of $\boldsymbol{\omega}$, to obtain an EM algorithm for this version of the problem, the update rule reduces to finding any root of the function

$$\boldsymbol{f}(\boldsymbol{\omega}) = \frac{N}{\sum_{k=1}^{M} e^{\omega_k}}[e^{\omega_1}, e^{\omega_2}, \ldots, e^{\omega_M}]^T - \sum_{i=1}^{N} \hat{\boldsymbol{z}}_i^{(k)}. \tag{52}$$

We then let $\boldsymbol{\omega}^{(k+1)}$ be such a root, which can be found using gradient ascent. For a fixed small constant γ, starting with some initial estimate for $\boldsymbol{\omega}^{(k+1)}$, we iterate on l as follows

$$\boldsymbol{\omega}^{(k+1),(l+1)} = \boldsymbol{\omega}^{(k+1),(l)} + \gamma \boldsymbol{f}\left(\boldsymbol{\omega}^{(k+1),(l)}\right), \tag{53}$$

and let $\boldsymbol{\omega}^{(k+1)}$ be the result from this process. An alternative is to just take one step of gradient ascent and use the result as the assignment to $\boldsymbol{\omega}^{(k+1)}$; that is,

$$\boldsymbol{\omega}^{(k+1)} = \boldsymbol{\omega}^{(k+1)} + \gamma \boldsymbol{f}\left(\boldsymbol{\omega}^{(k+1)}\right). \tag{54}$$

Noting that $\boldsymbol{f}(\boldsymbol{\omega}) = \nabla_{\boldsymbol{\omega}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$, for sufficiently small γ, the last update rule is a move up the gradient of Q and gives a value of $\boldsymbol{\omega}^{(k+1)}$ for which $Q(\boldsymbol{\theta}^{(k+1)} \mid \boldsymbol{\theta}^{(k)}) \geq Q(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\theta}^{(k)})$. Therefore, this is a *Generalized EM* method. A description of the GEM method that we used is in Algorithm 6. The use $\gamma = 0.0005$ because we want to try to make sure that the process monotonically increases. We understand that this is too conservative in some cases. However, increasing this value can produce non-monotonic behavior in other cases.

Algorithm 6 GEM

Initialize $\boldsymbol{\alpha}^{(0)}(\boldsymbol{\omega})$, and, for all $1 \leq j \leq M$, initialize $\boldsymbol{\mu}_j^{(0)}$ and $\boldsymbol{\Sigma}_j^{(0)}$
Compute $\boldsymbol{\omega}^{(0)}$ from $\boldsymbol{\alpha}^{(0)}(\boldsymbol{\omega})$
$k = 0$
repeat
  For all $1 \leq i \leq N$, $1 \leq j \leq M$, compute $\hat{z}_{i,j}^{(k)}$ using equation (37)
  For all $1 \leq j \leq M$, compute $\omega_j^{(k+1)}$, $\boldsymbol{\mu}_j^{(k+1)}$, and $\boldsymbol{\Sigma}_j^{(k+1)}$ using equations (54) with $\gamma = 0.0005$, (41), and (42), respectively.
  $k \leftarrow k + 1$
until stopping condition is met

8.1.3 AEM-1

AEM-1 is based on regular conjugate gradient. Algorithm 7 provides a general description of this method. The general idea behind all of the acceleration methods presented below is as follows:

repeat
run EM until we are *close to a solution*;
run the acceleration,
until stopping condition is met.

This algorithmic structure is similar to that proposed for the conjugate gradient acceleration of EM in the context of Bayesian networks [Thi95]. The reason for this basic structure in all of the acceleration methods is to keep the monotonicity property in all of the methods by using EM as a safeguard when a reduction in log likelihood occurs during the acceleration. Since the acceleration works best when we are close to a solution, we use EM to find initial parameters for the acceleration. We stop EM when the change in log likelihood is less than 0.5. In other words, we run EM "as long as the χ^2 statistic for testing the equality of two successive iterates is more than 1" [JJ93]. We say that the acceleration fails if there is a decrease in the log likelihood, which is typically caused by a failure during the line search. We discuss the fail conditions on the line search later in the paper. In addition, we would like to point out that we convert the parameters of the model into vector form by stacking each element of $\boldsymbol{\alpha}$, each element of $\boldsymbol{\mu}_j$, for $1 \leq j \leq M$, and each element in the upper triangle of $\boldsymbol{\Sigma}_j$, for $1 \leq j \leq M$.

We still need to deal with the original constraints on the parameters during the acceleration. With respect to the constraints on $\boldsymbol{\alpha}$, we use an approach similar to that used by Binder et al. [BKRK97] in the context of Bayesian networks with discrete variables. We project the part of the gradient relevant to $\boldsymbol{\alpha}$ into the constraint space. This will guarantee that the result from $\boldsymbol{\alpha}^{(k)} + \gamma^{(k)} \boldsymbol{r}_{\boldsymbol{\alpha}}^{(k)}$ satisfies the constraints on $\boldsymbol{\alpha}$ as long as $\gamma^{(k)}$ is sufficiently small. During the line search, before we evaluate $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)})$, we test that $\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)}$ satisfies the constraints on the parameters (i.e., $\sum_{j=1}^{M} \alpha_j = 1$, and for all j, $\alpha_j > 0$ and $\boldsymbol{\Sigma}_j$ is symmetric positive definite. If the resulting value of $\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)}$ does not satisfy the constraints, then we use a smaller value for γ until we find one such that $\boldsymbol{\theta}^{(k)} + \gamma \boldsymbol{d}^{(k)}$ satisfies them.

Algorithm 7 AEM-1

Initialize $\boldsymbol{\alpha}^{(0)}$, and, for all $1 \leq j \leq M$, initialize $\boldsymbol{\mu}_j^{(0)}$ and $\boldsymbol{\Sigma}_j^{(0)}$
repeat
Run EM starting from $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$ until the change in log likelihood in successive iterations is less than 0.5
if stopping condition is not met **then**
Convert resulting parameters from EM into vector form
Run regular conjugate gradient until the stopping condition is met or it fails.
Convert vector form of the parameters resulting from the conjugate gradient into their regular form and set the result to be $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$.
end if
until stopping condition is met

8.1.4 AEM-2

AEM-2 is based on the conjugate gradient acceleration of EM presented in Section 7.2. Algorithm 8 presents a general description of this method.

Algorithm 8 AEM-2

Initialize $\boldsymbol{\alpha}^{(0)}$, and, for all $1 \leq j \leq M$, initialize $\boldsymbol{\mu}_j^{(0)}$ and $\boldsymbol{\Sigma}_j^{(0)}$
repeat
  Run EM starting from $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$ until the change in log likelihood in successive iterations is less than 0.5
  if stopping condition is not met **then**
    Convert resulting parameters from EM into vector form
    Run conjugate gradient acceleration of EM until the stopping condition is met or it fails.
    Convert vector form of the parameters resulting from the conjugate gradient acceleration of EM into their regular form and set the result to be $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$.
  end if
until stopping condition is met

8.1.5 AEM-3

AEM-3 is as AEM-2, but we use the expression of the parameters that results from the reparametrization of the $\boldsymbol{\alpha}$ (i.e., the parameters are of the form given by equation (49)). Hence, we eliminate the constraints on $\boldsymbol{\alpha}$. Given that now we are using that expression for the parameters, we replace EM with GEM. The conjugate gradient acceleration of GEM is similar to that of EM except that we substitute the EM step by a GEM step and compute the gradients with respect to the new representation of the parameters.

8.1.6 PEM-1

PEM-1 is a method that uses Aitken acceleration but where we try to optimize the step size at every iteration. Algorithm 10 presents a description of this method.

8.1.7 PEM-2

PEM-2 is as PEM-1 but instead of trying to optimize the step size at every iteration, we use a constant step size. Algorithm 11 presents a description of this method.

8.2 Line search

All the methods presented above except for PEM-2 use the line search described in Algorithm 12. This line search method is a simple modification of that presented by Jamshidian and Jennrich [JJ93] and is based on the secant method.

PEM-2 uses a constant step size. However, since that step size might be too large resulting in parameters that lie outside the constraint space, we need to check the result from applying

Algorithm 9 AEM-3

Initialize $\boldsymbol{\alpha}^{(0)}(\boldsymbol{\omega})$, and, for all $1 \leq j \leq M$, initialize $\boldsymbol{\mu}_j^{(0)}$ and $\boldsymbol{\Sigma}_j^{(0)}$
Compute $\boldsymbol{\omega}^{(0)}$ from $\boldsymbol{\alpha}^{(0)}(\boldsymbol{\omega})$ using gradient ascent (i.e., equation (53))
repeat
  Run GEM starting from $\boldsymbol{\omega}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$ until the change in log likelihood in successive iterations is less than 0.5
  if stopping condition is not met **then**
    Convert resulting parameters from GEM into vector form
    Run conjugate gradient acceleration of GEM until the stopping condition is met or it fails.
    Convert vector form of the parameters resulting from the conjugate gradient acceleration of GEM into their regular form and set the result to be $\boldsymbol{\omega}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$.
  end if
until stopping condition is met

Algorithm 10 PEM-1

Initialize $\boldsymbol{\alpha}^{(0)}$, and, for all $1 \leq j \leq M$, initialize $\boldsymbol{\mu}_j^{(0)}$ and $\boldsymbol{\Sigma}_j^{(0)}$
repeat
  Run EM starting from $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$ until the change in log likelihood in successive iterations is less than 0.5
  if stopping condition is not met **then**
    Convert resulting parameters from EM into vector form
    Run Aitken-based acceleration of EM, performing the line search at each iteration to try to find the optimal step parameter, until the stopping condition is met or it fails.
    Convert vector form of the parameters resulting from the Aitken-based acceleration of EM into their regular form and set the result to be $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$.
  end if
until stopping condition is met

that step size. If that step size is too large, we keep halving that step size until we find an appropriate one [4]. The description of this "line search" is in Algorithm 13.

8.3 Stopping rule

We use the change in log likelihood from one iteration to the other as our stopping rule. We stop if the change in log likelihood is less than 10^{-5}.This is because we are interested in evaluating empirically how fast the methods converge to a stationary value of the log likelihood function. In most of the experimental runs, all of the methods converged to about both the same local maximum value in log likelihood and the same critical point in parameter

[4] We understand that because a step size of 1 is theoretically guaranteed to improve the likelihood, using a step size of value greater than or equal to 1 better. We can achieve that the value of the resulting step size is greater than or equal to 1 by getting monotonically decreasing values for the step size between 1 and the initial step size value until we find an appropriate one.

Algorithm 11 PEM-2

Initialize $\boldsymbol{\alpha}^{(0)}$, and, for all $1 \leq j \leq M$, initialize $\boldsymbol{\mu}_j^{(0)}$ and $\boldsymbol{\Sigma}_j^{(0)}$
repeat
  Run EM starting from $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$ until the change in log likelihood in successive iterations is less than 0.5
  if stopping condition is not met **then**
    Convert resulting parameters from EM into vector form
    Run Aitken-based acceleration of EM, with constant step parameter equal to 1.5, until the stopping condition is met or it fails.
    Convert vector form of the parameters resulting from the Aitken-based acceleration of EM into their regular form and set the result to be $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, and $\boldsymbol{\Sigma}_j^{(0)}$.
  end if
until stopping condition is met

space (modulo equivalent model results).

8.4 Initialization

We initialize the Gaussians as follows. In order to initialize the mixture proportions $\boldsymbol{\alpha}$, we generate a random distribution over M events. To initialize the means, we generate random points inside the hyper-cube of minimum volume containing all the data points [GJ94]. Finally, we initialize the covariance matrices as diagonal matrices with element (i, i) containing the square of the minimum of all distances between the mean of the i^{th} Gaussian and the other means [Bis96]. The reason for using this initialization method is that it seems likely to cover much of the parameter space relevant to the data (by the way in which we initialize the mixture proportions and the means) while it gives stability to the learning process because it seems to help avoid singularities (by the way we initialize the covariance matrices).

9 Results

We present results from two sets of experiments. In the first set of experiments, we fix all the parameters in the target models except the means of the Gaussian components and increase the difficulty of the problem by bringing the Gaussians closer and closer together. In the second set, we generate random models of different dimensions and number of Gaussian components to test the learning methods.

The first results are from running the methods above for 3 different data sets. The data sets are 2000 random samples of a 2-dimensional mixture of 2 Gaussians with parameters $\alpha_1 = \alpha_2 = 0.5$, $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{I}$, $\boldsymbol{\mu}_0 = [0, 0]^T$ and: (1) $\boldsymbol{\mu}_1 = [3, 3]^T$, (2) $\boldsymbol{\mu}_1 = [2, 2]^T$, (3) $\boldsymbol{\mu}_1 = [1, 1]^T$. Therefore, the estimation problem increases in difficulty.

Figures 5,6,9,10, 13 and 14 present the results from the experiments. The top left plot in each figure is a plot of the data set. For each data set, we generate 40 initial parameters using the initialization method discussed above and run each method starting from the same initial parameters. The top right plot in each figure is the number of iterations that running

Algorithm 12 JJsecant line search

Require: Current parameters $\boldsymbol{\theta}$ and direction $\boldsymbol{d}$

$\gamma_0 \leftarrow 0$
$\gamma_1 \leftarrow 2$
$n \leftarrow 1$
$\boldsymbol{r}_{\gamma_0} \leftarrow \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta} + \gamma_0 \boldsymbol{d}))$
$h_{\gamma_0} \leftarrow \boldsymbol{d}^T \boldsymbol{r}_{\gamma_0}$
while $n < 10$ **do**
  while the setting for the parameters $\boldsymbol{\theta} + \gamma_1 \boldsymbol{d}$ does not satisfy the constraints and $n < 10$ **do**
    $\gamma_1 \leftarrow \frac{\gamma_1}{2}$
    $n \leftarrow n + 1$
  end while
  if $n = 10$ **then**
    Raise fail condition and return
  end if
  $\boldsymbol{r}_{\gamma_1} \leftarrow \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta} + \gamma_1 \boldsymbol{d})$
  $h_{\gamma_1} \leftarrow \boldsymbol{d}^T \boldsymbol{r}_{\gamma_1}$
  if $n \neq 1$ and $|h_{\gamma_1}| < 0.1 h_0$ **then**
    return γ_1 as result
  end if
  if $\text{sign}(\gamma_1 - \gamma_0) \frac{h_{\gamma_0} - h_{\gamma_1}}{|h_{\gamma_0}| + |h_{\gamma_1}|} < 10^{-5}$ **then**
    Raise fail condition and return
  end if
  $\gamma^* \leftarrow \frac{\gamma_1 h_{\gamma_0} - \gamma_0 h_{\gamma_1}}{h_{\gamma_0} - h_{\gamma_1}}$
  $\gamma_0 \leftarrow \gamma_1$
  $h_{\gamma_0} = h_{\gamma_1}$
  $\gamma_1 = \gamma^*$
  $n \leftarrow n + 1$
end while
Raise fail condition and return

Algorithm 13 Constant line search

Require: Current parameters $\boldsymbol{\theta}$ and direction $\boldsymbol{d}$

$\gamma \leftarrow 1.5$

$n \leftarrow 1$

while the setting for the parameters $\boldsymbol{\theta} + \gamma \boldsymbol{d}$ does not satisfy the constraints and $n < 10$ **do**

$\gamma \leftarrow \frac{\gamma}{2}$

$n \leftarrow n + 1$

end while

if $n = 10$ **then**

Raise fail condition and return

else

Return γ as result.

end if

EM on the data took to *converge* starting from each initial parameterization. For each run from the same initial parameters, we computed the *acceleration speed-up* by dividing, for each method tested, the number of *EM-equivalent* iterations that EM took by those of the method. A value for this statistic greater than 1 means that the method was faster than EM by a factor equal to the value of the statistic. The rest of the plots in the figures are histograms of the value of the acceleration speed-up statistics for the 40 runs. Figures 7,8, 11,12, 15 and 16 are scatter plots of the results that show the number of iterations of each method with respect to the number of iterations for EM for each run. The line represents the identity function. Methods that are faster than EM would have plots such that the points lie to the left of the line, close to the bottom of the plot and far away from the line. Finally, Table 1 presents the average number of EM-equivalent iterations that each method took on each of the data sets.

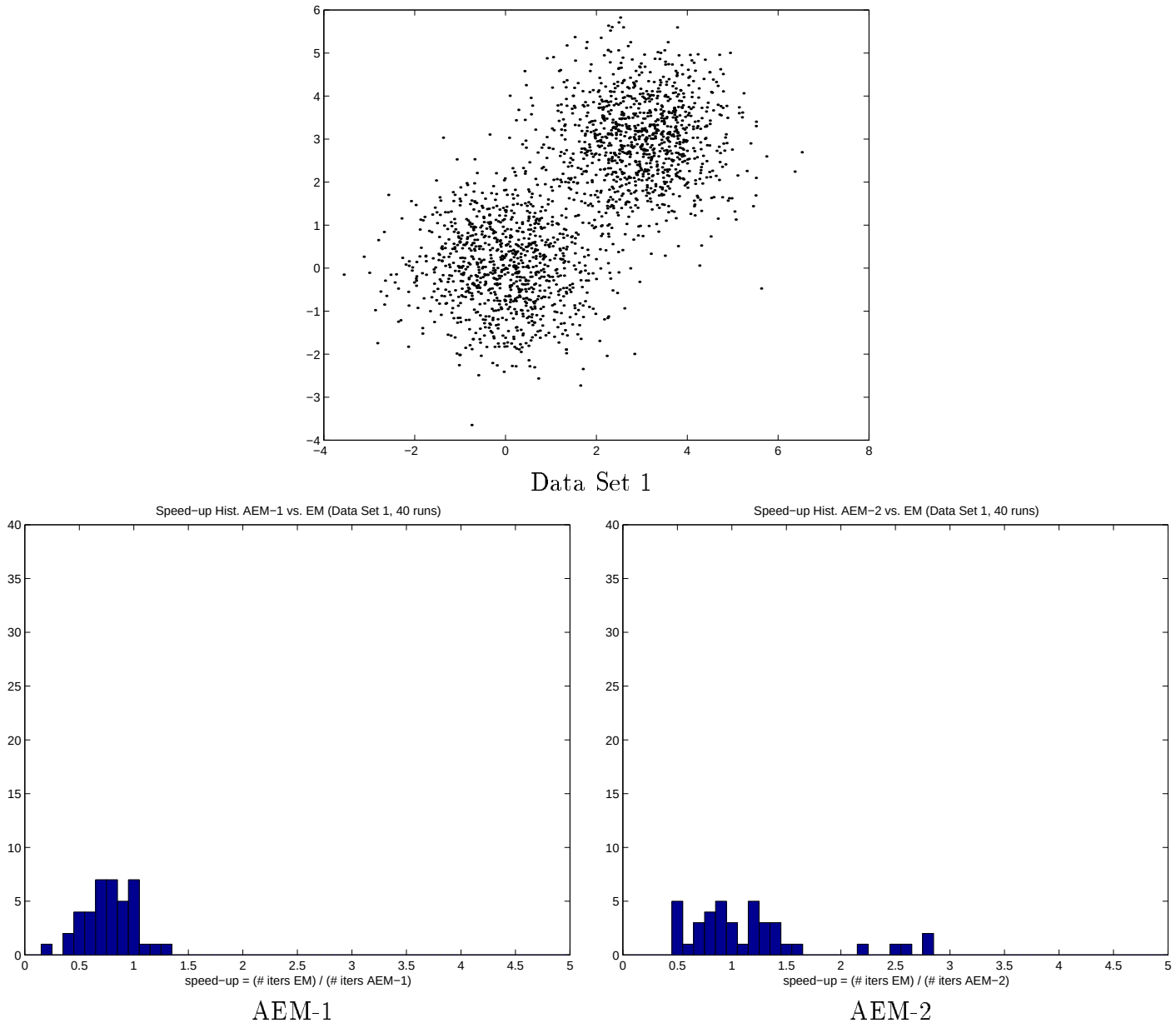


Figure 5: This figure presents the results for Data Set 1. The plots, except for the top plots, are histograms of the acceleration speed-up statistic for the 40 runs.

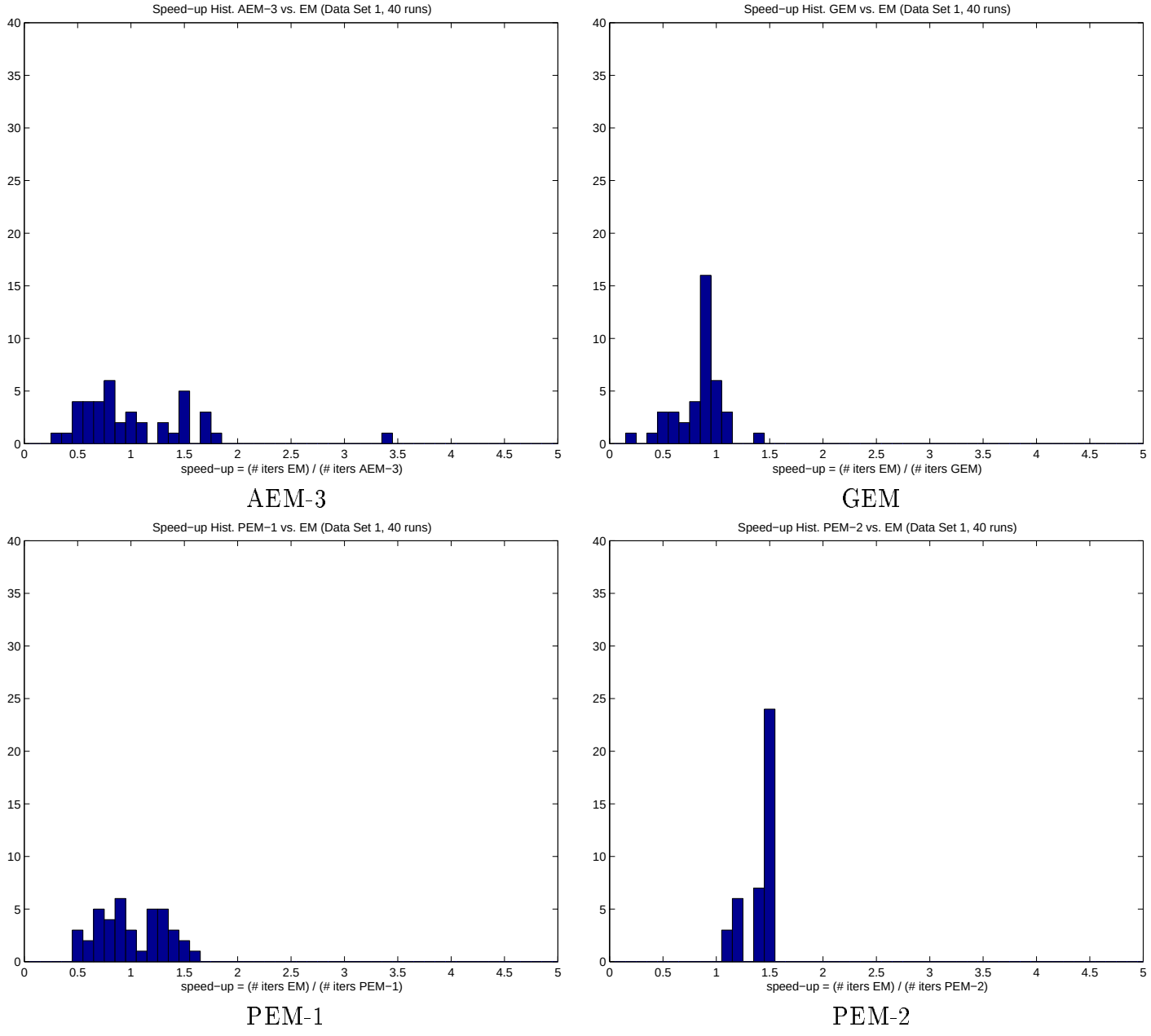


Figure 6: Continuation of Figure 5.

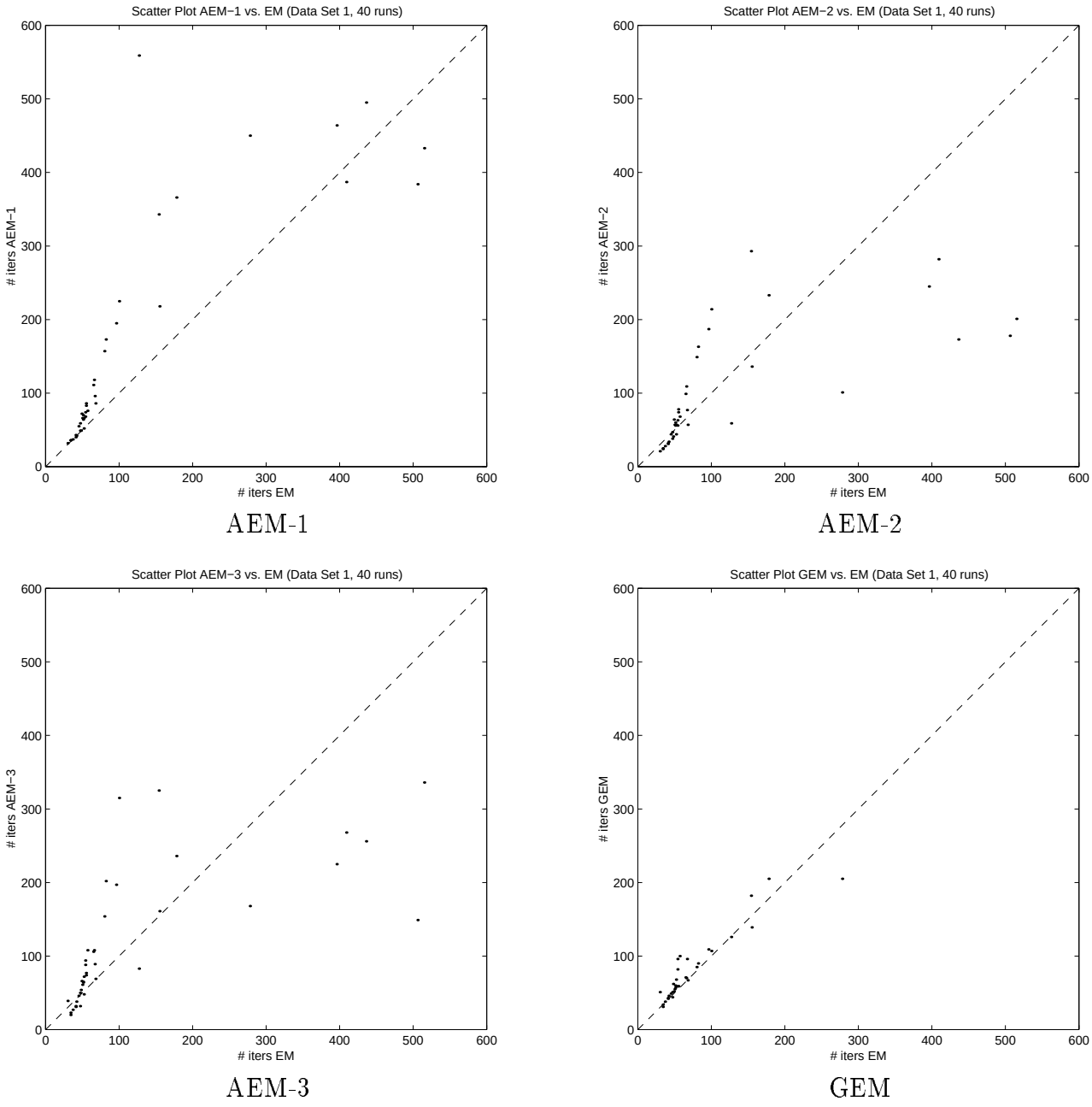
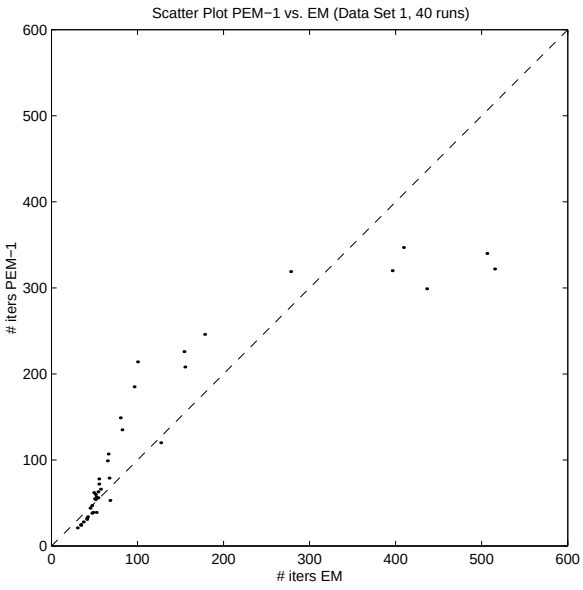
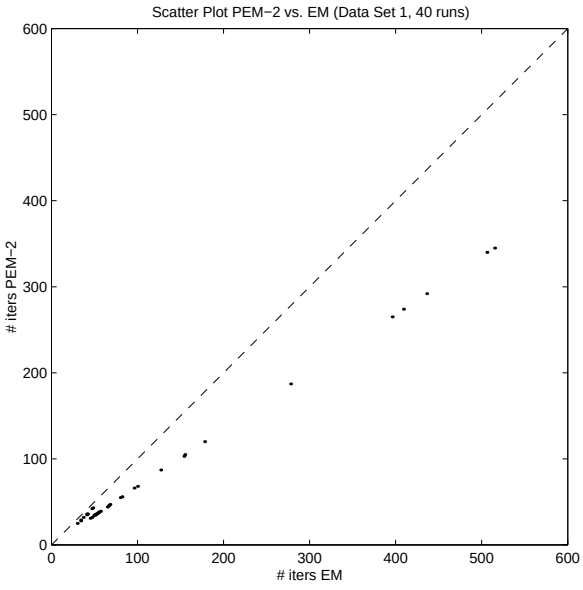


Figure 7: This figure presents the results for Data Set 1. The plots are scatter plots of the number of iterations of each method with respect to the number of iterations of EM for each run.



PEM-1



PEM-2

Figure 8: Continuation of Figure 7.

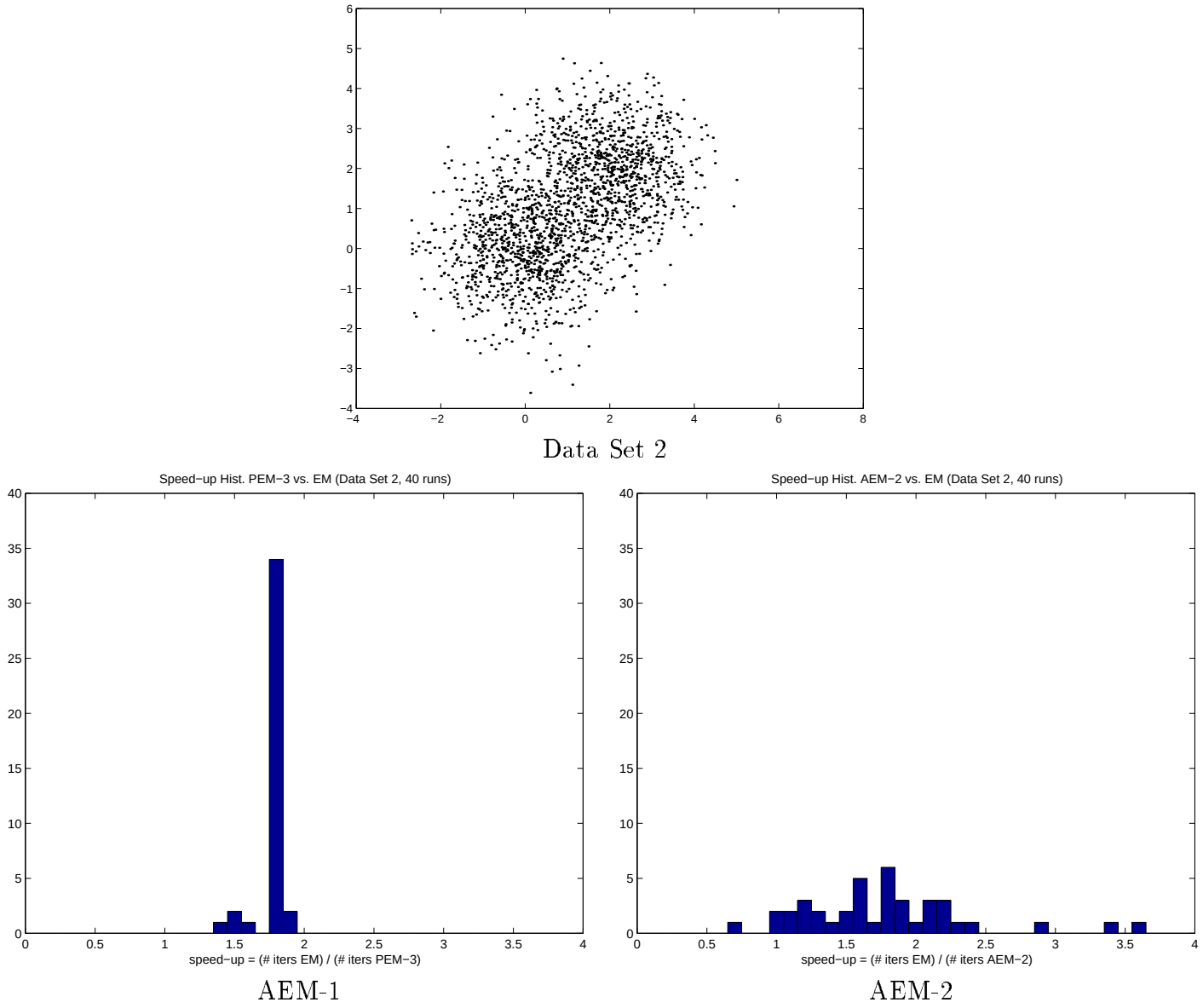


Figure 9: This figure presents the results for Data Set 2. The plots, except for the top plots, are histograms of the acceleration speed-up statistic for the 40 runs.

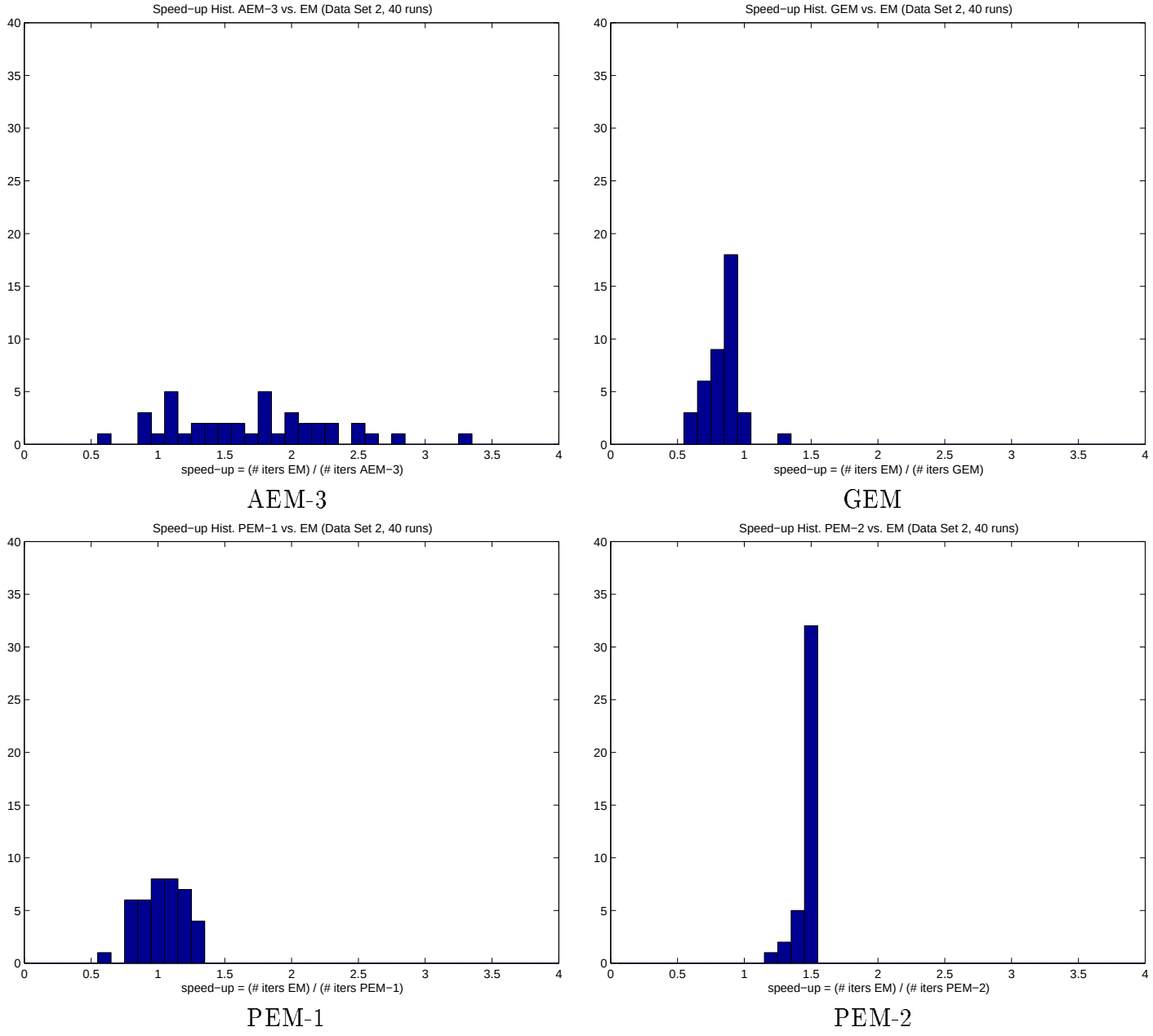


Figure 10: Continuation of Figure 9.

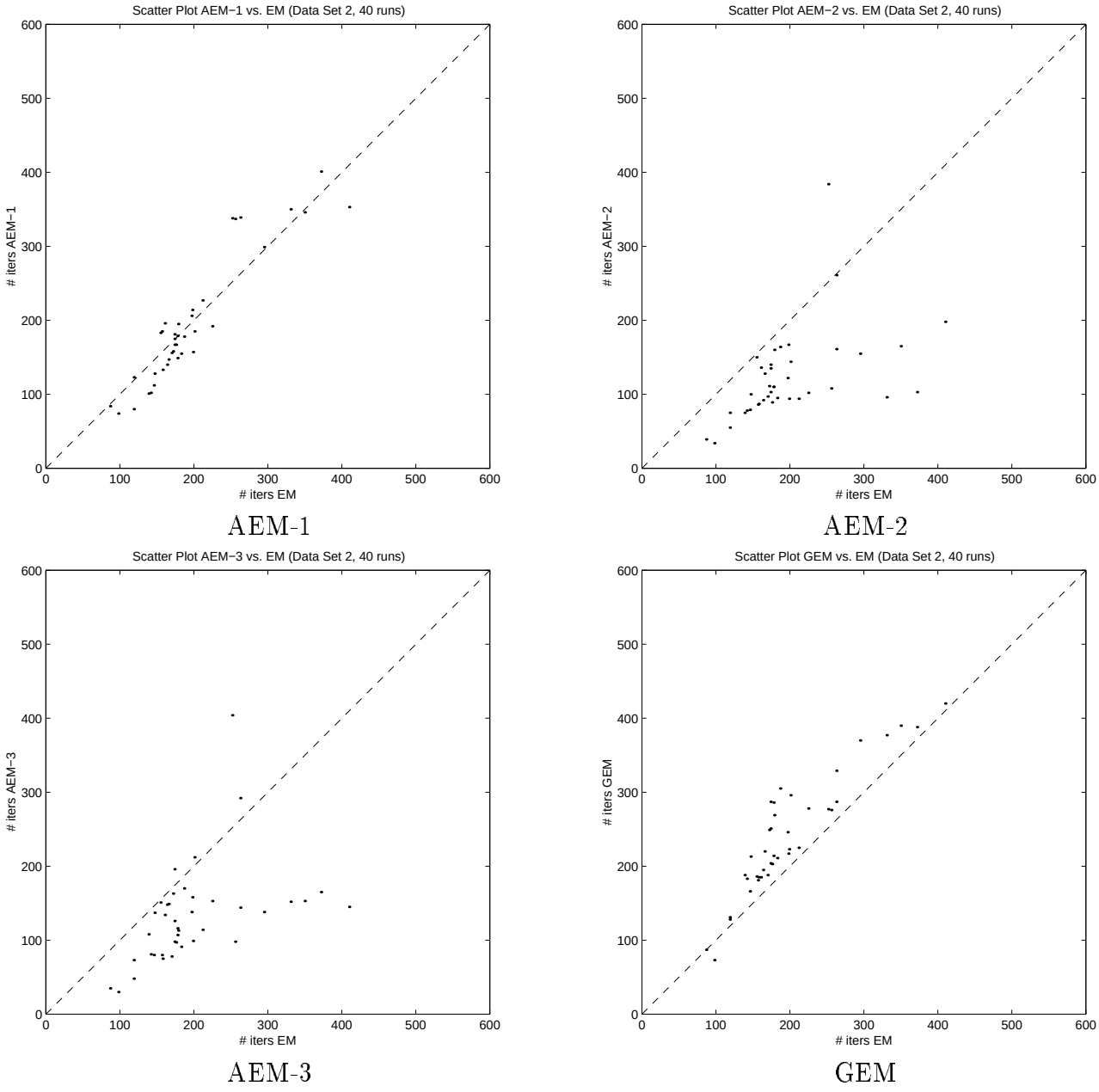
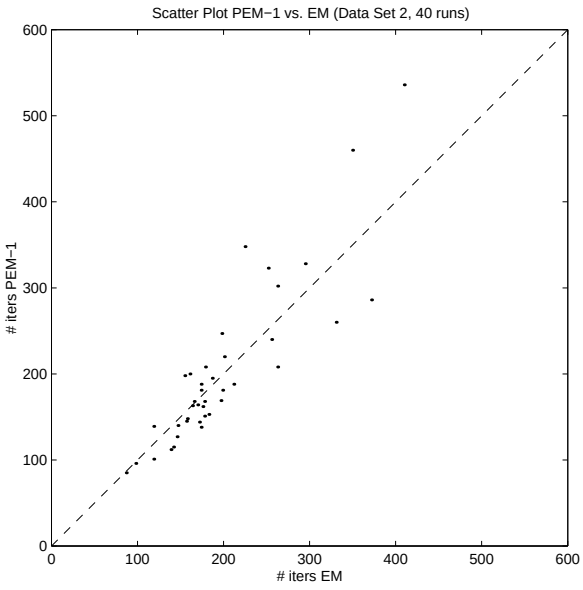
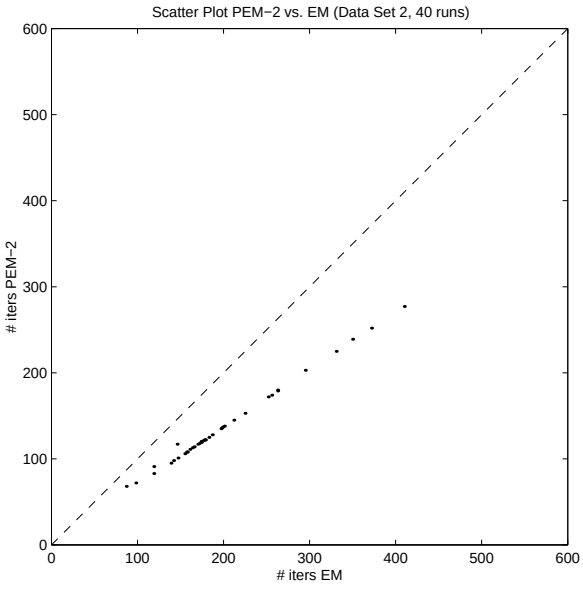


Figure 11: This figure presents the results for Data Set 2. The plots are scatter plots of the number of iterations of each method with respect to the number of iterations of EM for each run.



PEM-1



PEM-2

Figure 12: Continuation of Figure 11.

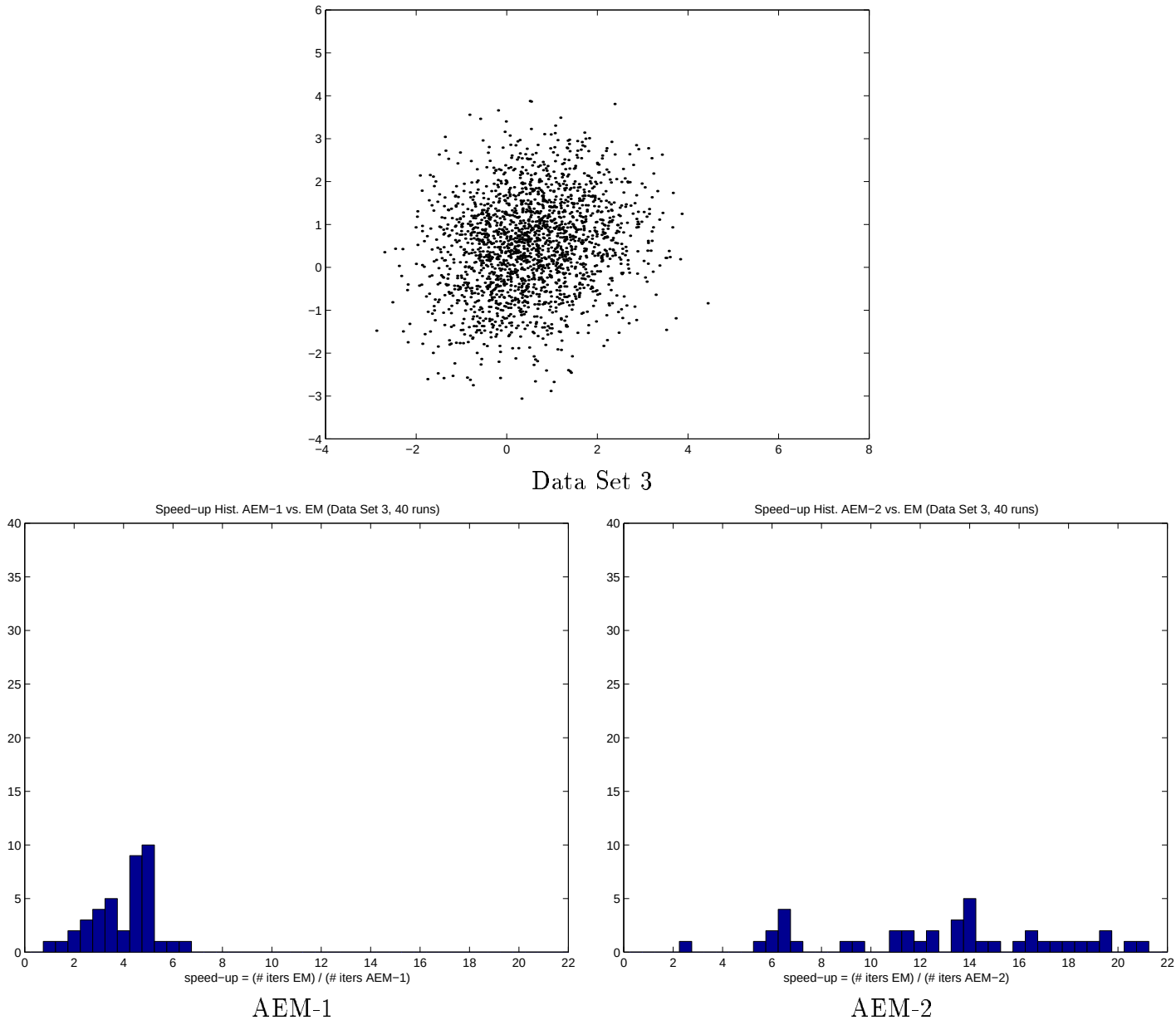
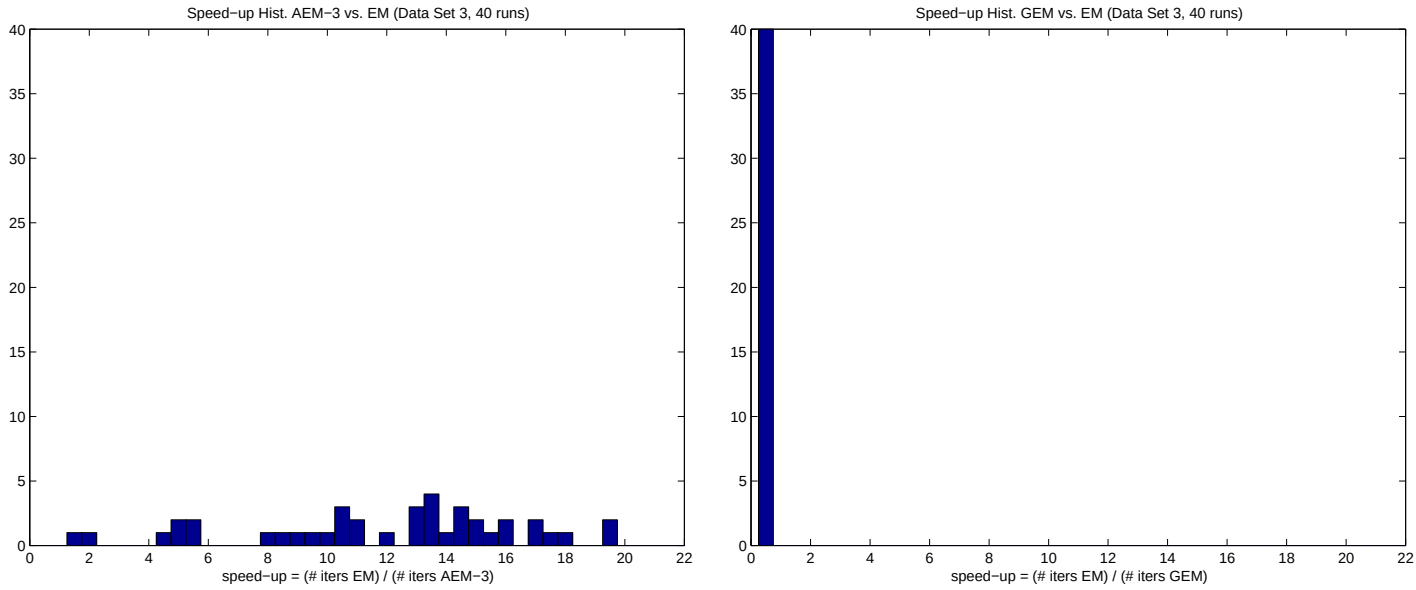
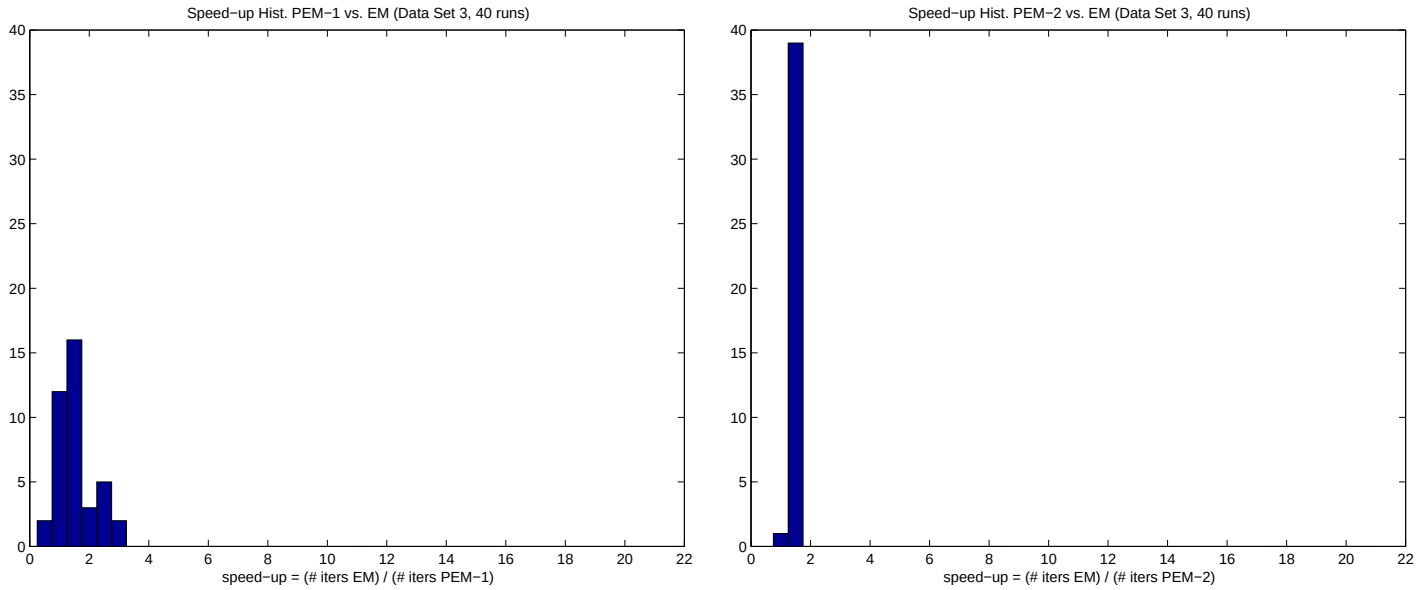


Figure 13: This figure presents the results for Data Set 3. The plots, except for the top plots, are histograms of the acceleration speed-up statistic for the 40 runs.



AEM-3 GEM



PEM-1 PEM-2

Figure 14: Continuation of Figure 13.

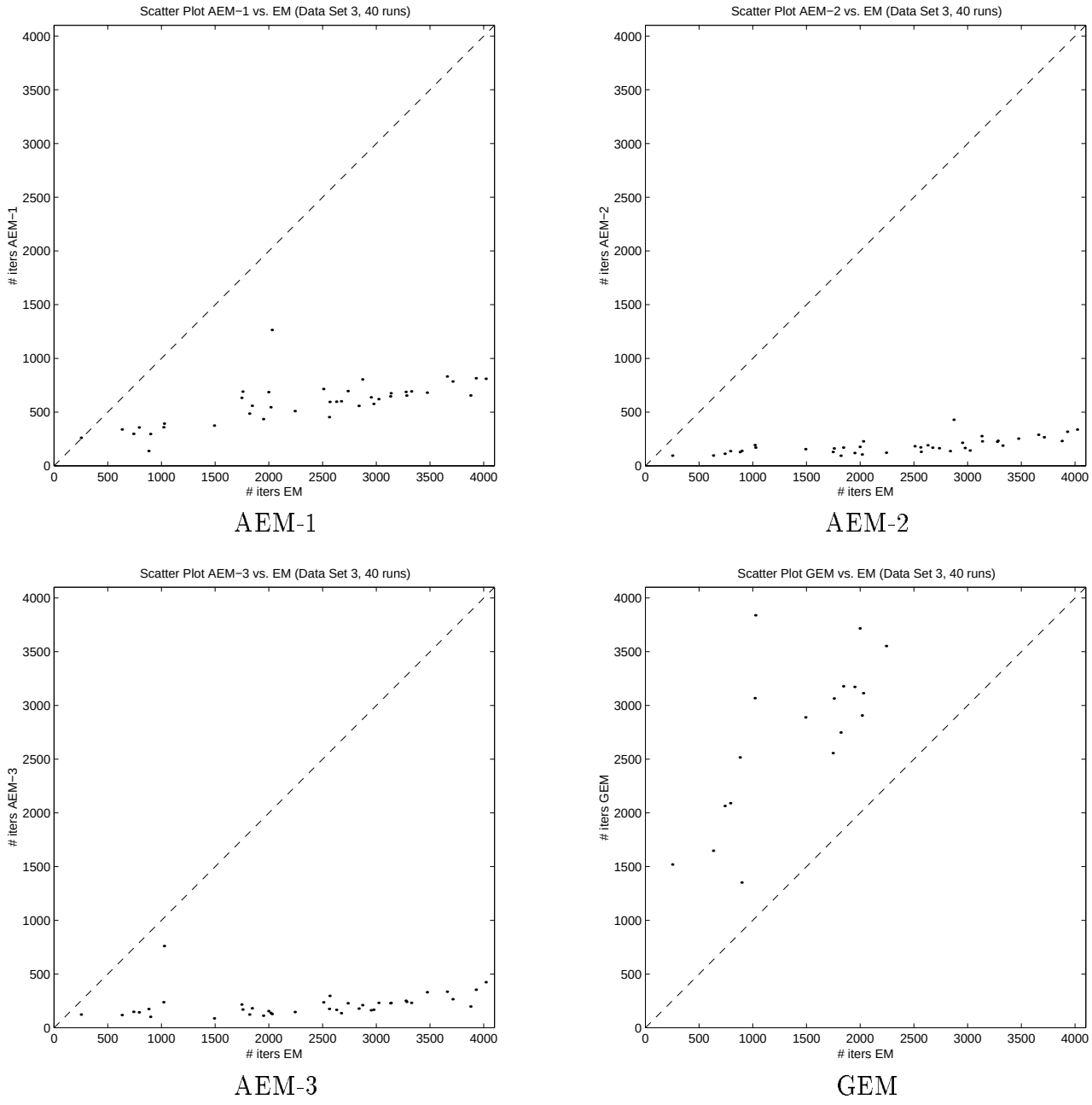
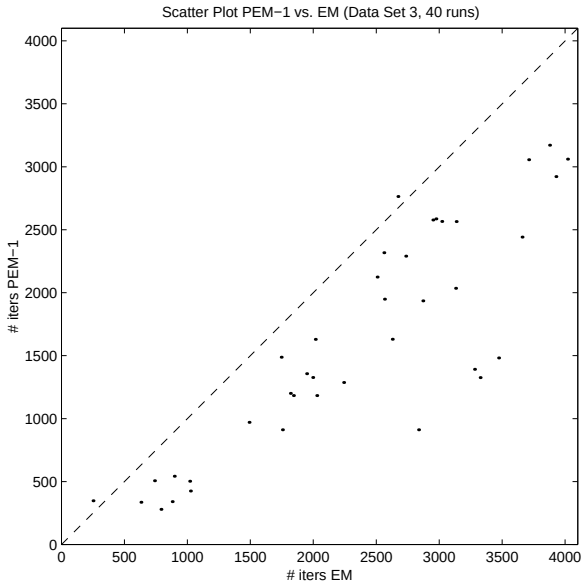
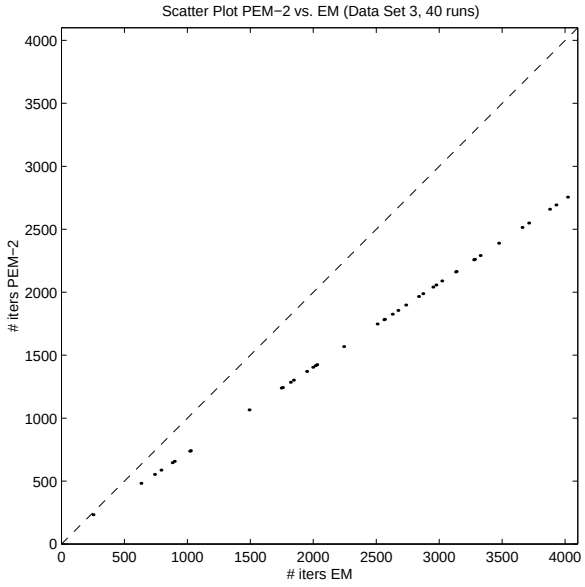


Figure 15: This figure presents the results for Data Set 3. The plots are scatter plots of the number of iterations of each method with respect to the number of iterations of EM for each run.



PEM-1



PEM-2

Figure 16: Continuation of Figure 15.

Method	Data Set		
	1	2	3
EM	120 ± 43	198 ± 23	2356 ± 321
AEM-1	163 ± 49	225 ± 66	585 ± 63
AEM-2	100 ± 24	122 ± 19	187 ± 23
AEM-3	116 ± 28	131 ± 21	214 ± 36
GEM	210 ± 122	240 ± 25	4308 ± 551
PEM-1	120 ± 33	202 ± 29	1967 ± 750
PEM-2	83 ± 29	137 ± 15	1642 ± 215

Table 1: This table presents the (approximate) 95% confidence intervals on the average number of EM-equivalent iterations that each method took to converge on the different data sets for the first set of experiments.

Method	Data Set			
	5	6	7	8
EM	136 ± 55	1672 ± 433	14 ± 1	122 ± 83
AEM-2	140 ± 65	195 ± 33	15 ± 1	140 ± 161
PEM-1	121 ± 37	1028 ± 336	15 ± 1	140 ± 133
PEM-2	111 ± 39	1160 ± 294	14 ± 1	91 ± 57

Table 2: This table presents the (approximate) 95% confidence intervals on the average number of EM-equivalent iterations that each method took to converge on the different data sets for the second set of experiments.

The first results from the second experiments are from data generated from a random model with 5 Gaussians in 2 dimensions ($N = 5000$). Figure 17 presents a plot of the data and scatter plots on the number of iterations of AEM-2, PEM-1 and PEM-2 as a function of the number of iterations of EM. The second results are from data generated from another model with 5 Gaussians in 2 dimensions ($N = 10000$), but this time the Gaussians are less distinguishable from each other. Figure 18 presents the second results. The third results are from data generated from 2 Gaussians in 5 dimensions ($N = 20000$). Figure 19 presents the third results. The fourth results are from data generated from 5 Gaussians in 5 dimensions ($N = 20000$). Figure 20 presents the fourth results. Finally, Table 2 presents the average number of EM-equivalent iterations that each method took on each of these data sets.

10 Discussion

The results show that accelerations of EM are useful in speeding up the learning process. However, there are cases when the proposed accelerations slow down convergence. That typically happened when the problem was inherently easy, and therefore the slow-down was not as severe. The cause of the slow-down was that the acceleration started when EM got close to a valley and not when it was close to a solution as it is supposed to. Again, note that

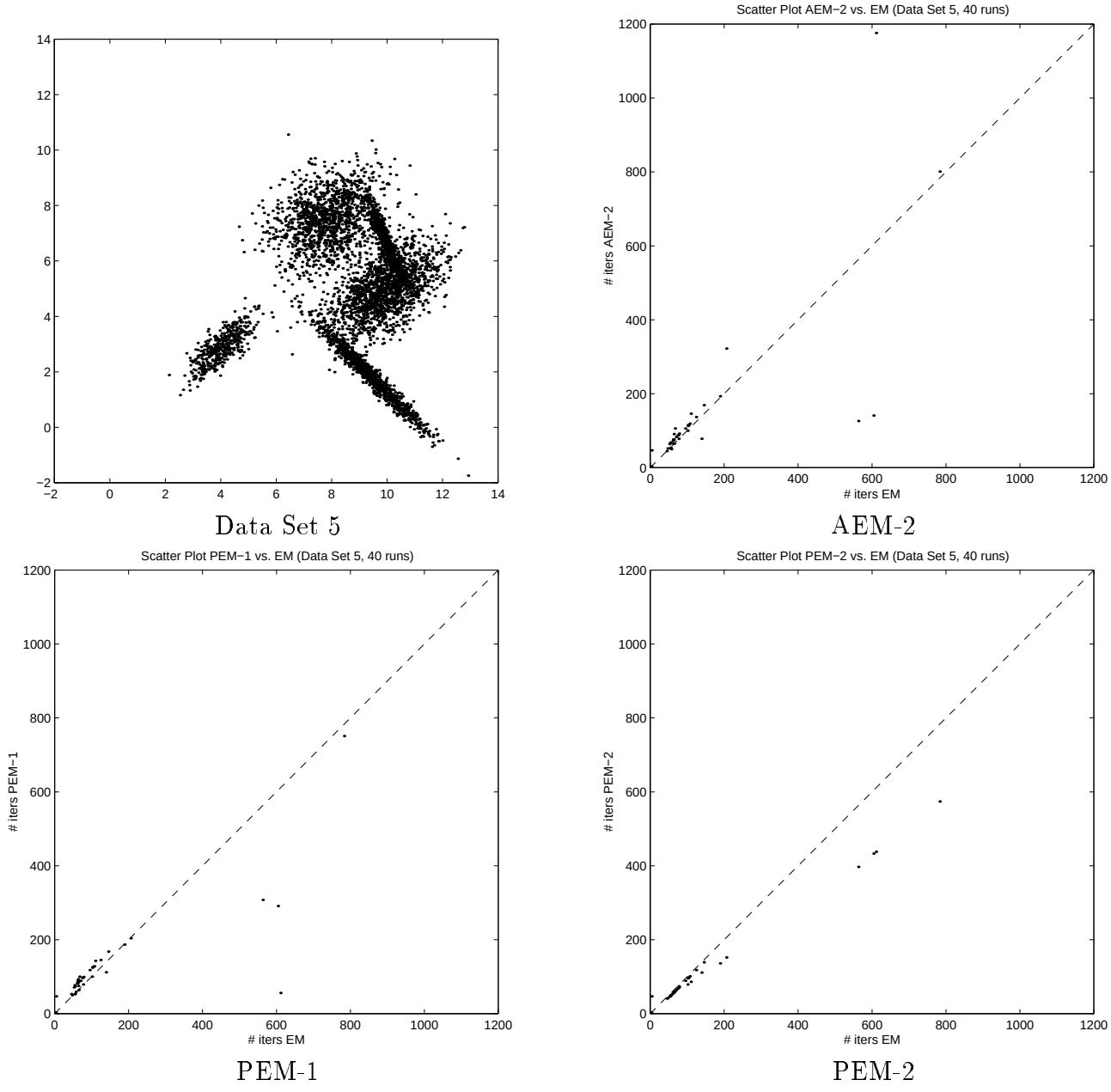
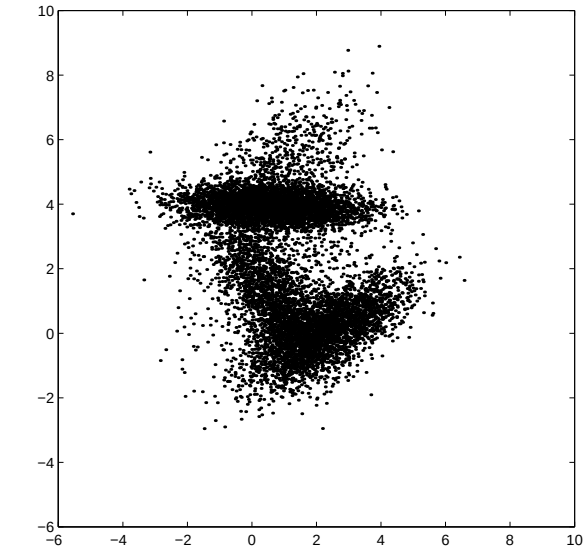
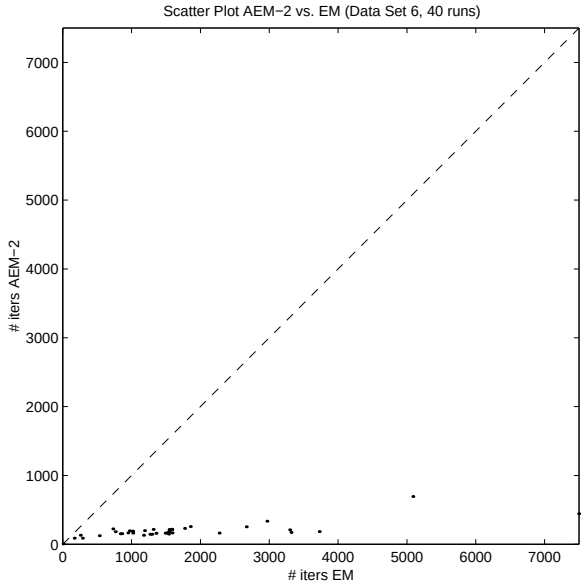


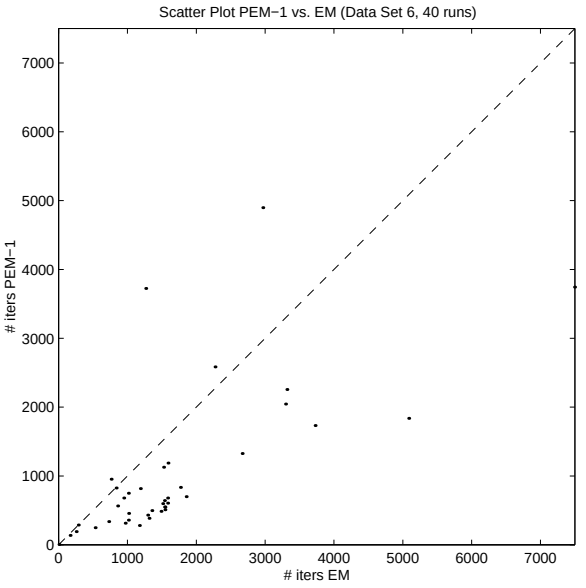
Figure 17: This figure presents the results for Data Set 5.



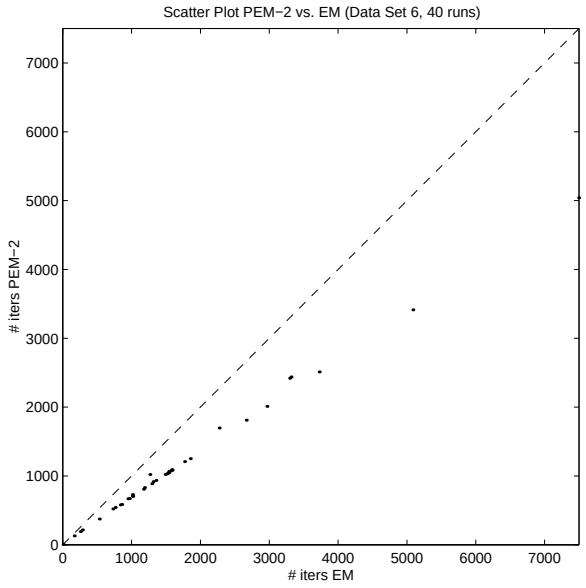
Data Set 6



AEM-2



PEM-1



PEM-2

Figure 18: This figure presents the results for Data Set 6.

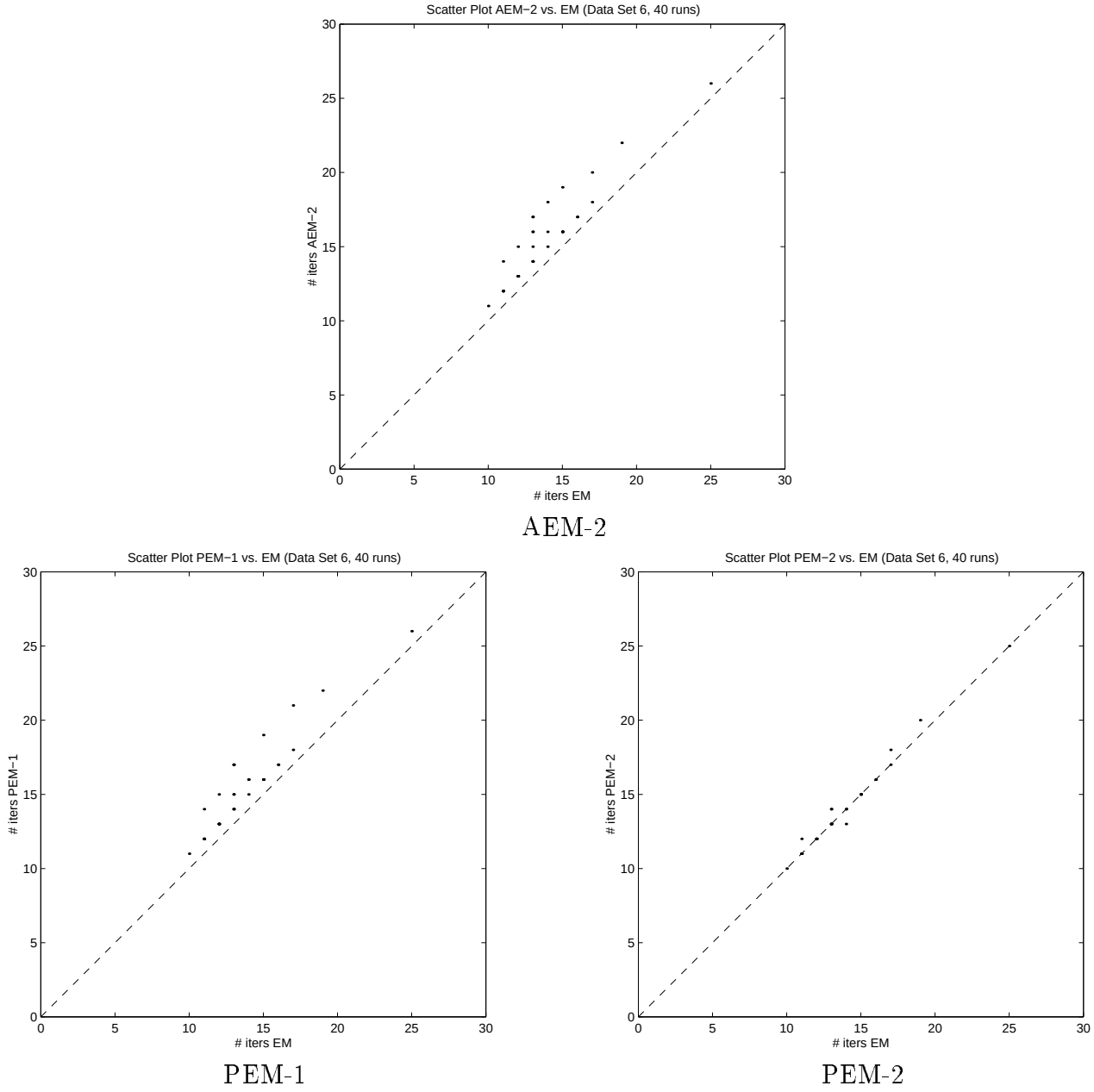


Figure 19: This figure presents the results for Data Set 7.

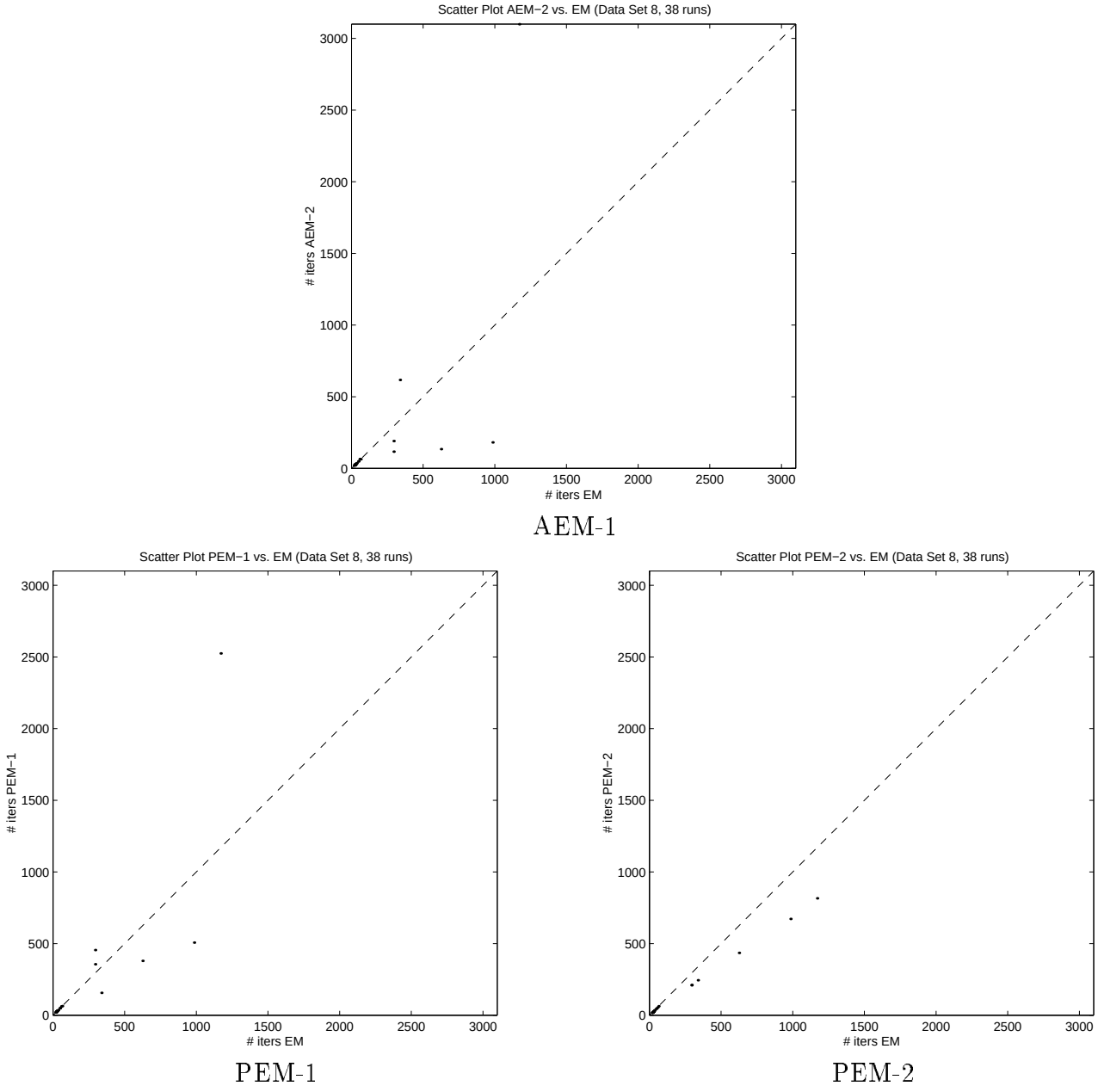


Figure 20: This figure presents the results for Data Set 8.

in such cases the slow-down is not as significant as the speed up the accelerations produce in hard instances. Evidence of this is the magnitude of the number of iterations as presented by the result tables.

Among the accelerations of EM, the conjugate gradient acceleration seems to provide the best speed-ups in hard instances. In easy instances, however, mainly because of false acceleration starts, those speed ups were not as significant, to the point that it sometimes produced slow-downs. It is interesting to note that the method based on parameterized EM with a constant parameter always sped up the process and the speed ups were about the same throughout (30% less iterations than regular EM). This speed up however might not be enough in hard instances, given the large amount of iterations EM required to converge in those cases. Trying to optimize the step size for parameterized EM using an inexact line search only seemed to pay off on average in hard instances.

Not surprisingly, GEM typically slows down convergence. Intuitively, this results from taking too small of a step at each iteration. We can try to increase the step size. However, we need to be careful that we do not take a step that decreases the likelihood. Therefore, the question of how to determine a good step size becomes important. We suspect that performing EM is typically better than performing GEM when we can actually maximize the Q function easily. Note however that the conjugate gradient accelerations of GEM (AEM-3) and EM (AEM-2) perform similarly in hard instances. The conjugate gradient acceleration of EM seems a little better than that of GEM in easier instances, but more experiments are necessary to establish significance.

We concede that in the runs where EM seems inferior to other methods, EM could have being stopped earlier since the flatness of the log likelihood function is such that we could have gotten a model that does relatively well in terms of log likelihood. It seems then that we need a different value for the change in log likelihood as our stopping rule for different problems. The problem is that we would not know what that is or how to determine it. If it is too large we can stop before we are even close to a solution. This is because the log likelihood function can have many hills and valleys and we sometimes do not know whether the small change in log-likelihood results from going by a valley or from being close to the top of a hill (i.e., a solution). If this threshold is too small we can perform more iterations than necessary. Then, from the standpoint of the AI community, the problem is being able to develop and study stopping rules that are more elaborated in that they use more information to make a determination of whether the method is close to a solution or not without sacrificing too much its efficiency.

We can also view the slowness of EM in problems where the Gaussians are not well separated as a problem of model selection [XJ96]. This is because in those problems, we are trying to fit a number of Gaussians to a cloud that looks more like a single cloud than a group of clouds that are close together. Therefore, we should only try to fit one Gaussian and not a multiple of them. The problem with this is that in cases when we know those subtle distinctions exists we want our model to learn them. A solution to this problem might be to use a small number of Gaussians to learn a general description of the data and use a different model to learn the subtle distinctions. Hierarchical models come to mind as an alternative. On the other hand, a program like AutoClass [CKS+88] requires that we find MLE for models with different number of components in order to estimate the probability of the data given the number of components and suggest the most likely number of components

in the data..

11 Future Work

First of all, we need to perform experiments on real datasets (as opposed to synthetic datasets as we do here) to corroborate the results we found. Second, experimental observation seems to support the idea that using *K-means* to find starting values for the parameters can improve the speed of convergence of the methods. This is because the problem when the Gaussians are too close together is that it takes the methods (EM in particular) some time to differentiate between what points belong to which Gaussian. This is something K-means is good at [KMN97]. Therefore, although K-means is optimizing a different function, starting from models that differentiate the data as much as possible and then trying to take care of shared points using EM might speed up the learning process. Actually, K-means is sometimes used in practice to find an initial value for the parameters from which to start EM. We suspect that all of the acceleration methods will take advantage of using K-means to find starting values for the parameters.

12 Conclusion

Our experiments show that acceleration methods can significantly improve the convergence speed of EM. Although those methods might slow down convergence on instances when the problem is simple, the slow down is not as significant as the speed up we obtain on instance when the problem is very hard.

Although we observe significant speed up in hard instances when we use the conjugate gradient acceleration of EM, we need to trade the speed up with the complexity of implementing such a method as efficiently as possible. It is not hard to see that EM and simple Aitken-based accelerations (i.e., parameterized EM) are simpler than the conjugate gradient acceleration.

References

[Ber95] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts, 1995.

[Bis96] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford, 1996.

[BKRK97] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 1997.

[BKS97] Eric Bauer, Daphne Koller, and Yoram Singer. Update rules for parameter estimation in Bayesian networks. In Dan Geiger and Prakash Pundalik Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 3–13, San Francisco, CA, 1997. Morgan Kaufmann.

[Cha54] K. C. Chanda. A note on the consistency and maxima of the roots of likelihood equations. *Biometrika*, 41:56–61, 1954.

[CKS+88] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, 1988.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 1991.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* Wiley-Interscience, 1973.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[Gha97] Zoubin Ghahramani. Learning dynamic Bayesian networks. In C. L. Giles and M. Gori, editors, *Adaptive Processing of Temporal Information*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1997. To Appear.

[GJ94] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6.* Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[JJ93] Mortaza Jamshidian and Robert I. Jennrich. Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Society*, 88(421):221–228, March 1993. Theory and Methods.

[JX93] Michael I. Jordan and Lei Xu. Convergence results for the EM approach to mixtures of expert architectures. Technical Report A.I. Memo No. 1458, C.B.C.L. Memo No. 87, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological an Computational Learning, Department of Brain and Cognitive Sciences, November 1993.

[KMN97] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In Dan Geiger and Prakash Pundalik Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 282–293, Brown University, Providence, Rhode Island, USA, August 1997. Morgan Kaufmann.

[Mei89] Isaac Meilijson. A fast improvement to the EM algorithm on its own terms. *J. R. Statist. Soc.*, 51(1):127–138, 1989.

[MK97] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions.* Wiley Series in Probability and Statistics. Wiley-Interscience, 1997.

[Pol71] E. Polak. *Computational Methods in Optimization: A Unified Approach*, volume 77 of *Mathematics in Science and Engineering*. Academic Press, New York, 1971.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[PW78] B. Charles Peters, Jr. and Homer F. Walker. An iterative procedure for obtaining maximum-likelihood estimates of the parameters for a mixture of normal distributions. *SIAM J. Appl. Math.*, 35(2):362–378, September 1978.

[RW84] Richard A. Redner and Homer F. Walker. Mixture densities, maximum likelihood and the EM algortihm. *SIAM Review*, 26(2):195–239, April 1984.

[She94] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method that even an idiot can understand. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, March 1994.

[Thi95] Bo Thiesson. Accelerated quantification of Bayesian networks with incomplete data. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge DIscovery and Data Mining*, pages 306–11. AAAI Press, 1995.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[Wu83] C. F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[XJ96] Lei Xu and Michael I. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8:129–151, 1996.

[Xu97] Lei Xu. Comparative analysis on convergence rates of the EM algorithm and its two modifications for Gaussians mixtures. *Neural Processing Letters*, 6:69–76, 1997.

A Gradient notation

We are using the following notation: for $f : \Re^d \to \Re$, $\boldsymbol{h} : \Re^d \to \Re^m$, $\boldsymbol{g} : \Re^m \to \Re^k$, and $\boldsymbol{s}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{h}(\boldsymbol{x}))$, the *gradient* of f with respect to $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_d} \right)^T .$$

For a vector-valued function $\boldsymbol{h}$,

$$\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_d(\boldsymbol{x}))^T ,$$

the *gradient matrix* of $\boldsymbol{h}$ with respect to $\boldsymbol{x}$ is

$$\begin{aligned}\nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}) &= (\nabla_{\boldsymbol{x}}h_1(\boldsymbol{x})\nabla_{\boldsymbol{x}}h_2(\boldsymbol{x})\cdots\nabla_{\boldsymbol{x}}h_m(\boldsymbol{x})) \\ &= \begin{pmatrix} \frac{\partial h_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial h_2(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial h_m(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial h_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial h_2(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial h_m(\boldsymbol{x})}{\partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_1(\boldsymbol{x})}{\partial x_d} & \frac{\partial h_2(\boldsymbol{x})}{\partial x_d} & \cdots & \frac{\partial h_m(\boldsymbol{x})}{\partial x_d} \end{pmatrix}.\end{aligned}$$

The *Jacobian* of $\boldsymbol{h}$ is $(\nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}))^T$. By the *chain rule* for differentiation,

$$\nabla_{\boldsymbol{x}}\boldsymbol{s}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})\nabla_{\boldsymbol{h}(\boldsymbol{x})}\boldsymbol{g}(\boldsymbol{h}(\boldsymbol{x})).$$

From the above definitions we can derive some additional useful relations such as

$$\nabla_{\boldsymbol{x}}\left(\boldsymbol{y}^T\boldsymbol{x}\right) = \boldsymbol{y},$$

$$\nabla_{\boldsymbol{x}}\left(\boldsymbol{x}\right) = \boldsymbol{I},$$

$$\nabla_{\boldsymbol{x}}\left(\boldsymbol{A}\boldsymbol{x}\right) = \boldsymbol{A}^T,$$

$$\nabla_{\boldsymbol{x}}\left(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}\right) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^T\boldsymbol{x}.$$

We also use the following notation. Let $\boldsymbol{A}$ be an $(m \times n)$ matrix and $a_{i,j}$ be its element in row i column j. Then

$$\text{vec}\left(\boldsymbol{A}\right) = (a_{1,1}, a_{2,1}, a_{3,1}, \ldots, a_{m,1}, a_{1,2}, a_{2,2}, \ldots, a_{m,n})^T.$$

That is, $\text{vec}\left(\boldsymbol{A}\right)$ is the vector that results from stacking the columns of matrix $\boldsymbol{A}$ from right to left. Let $f : \Re^{mn} \rightarrow \Re$ and $\text{vec}(\boldsymbol{A}) \in \Re^{mn}$. We denote $f(\boldsymbol{A}) \equiv f(\text{vec}(\boldsymbol{A}))$. Also,

$$\begin{aligned}\nabla_{\boldsymbol{A}}f(\boldsymbol{A}) &\equiv \nabla_{\text{vec}(\boldsymbol{A})}f(\boldsymbol{A}) \\ &= \left(\frac{\partial f(\boldsymbol{A})}{\partial a_{1,1}}, \frac{\partial f(\boldsymbol{A})}{\partial a_{2,1}}, \ldots, \frac{\partial f(\boldsymbol{A})}{\partial a_{m,n}}\right)^T \\ &= \text{vec}\left(\begin{pmatrix} \frac{\partial f(\boldsymbol{A})}{\partial a_{1,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,n}} \\ \frac{\partial f(\boldsymbol{A})}{\partial a_{2,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{2,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(\boldsymbol{A})}{\partial a_{m,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{m,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{m,n}} \end{pmatrix}\right).\end{aligned}$$

Using the notation presented above, we can derive the following relations:

$$\begin{aligned}\nabla_{\boldsymbol{A}}\left(\mid \boldsymbol{A} \mid\right) &= \mid \boldsymbol{A} \mid \text{vec}\left(\left(\boldsymbol{A}^{-1}\right)^T\right), \\ \nabla_{\boldsymbol{A}}\left(\boldsymbol{x}^T\boldsymbol{A}^{-1}\boldsymbol{x}\right) &= -\text{vec}\left(\left(\boldsymbol{A}^{-1}\right)^T\boldsymbol{x}\boldsymbol{x}^T\left(\boldsymbol{A}^{-1}\right)^T\right).\end{aligned}$$

NOTE: I can verify the above relations for (2×2) matrices. It seems that they should hold in general but I do not know how to prove it.

B Hessian matrix

The *Hessian matrix* is the matrix of second order partial derivatives of a function with respect to its variables. For instance, if we have a function $f : \Re^n \rightarrow \Re$ then the Hessian matrix of f with respect to the variables $\boldsymbol{x} \in \Re^n$ is

$$\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x}) \equiv \boldsymbol{H}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1 \partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_1 \partial x_n} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2 \partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_2 \partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n \partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_n \partial x_n} \end{pmatrix}.$$

C Fisher information matrix

The *Fisher information matrix* $\boldsymbol{F}(\boldsymbol{\theta})$ is related to the the Hessian of the log likelihood function. The Fisher information matrix is by definition [RW84, Mei89]

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{\theta}) &= \int_{\boldsymbol{x}} \left(\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x} \mid \boldsymbol{\theta})\right) \left(\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x} \mid \boldsymbol{\theta})\right)^T p(\boldsymbol{x} \mid \boldsymbol{\theta}) d\boldsymbol{x} \\ &= E_{\boldsymbol{X}} \left[\left(\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right) \left(\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)^T\right]. \end{aligned}$$

Let the Hessian of the log likelihood of a random data point $\boldsymbol{X}$ with respect to the parameters $\boldsymbol{\theta}$ be

$$\begin{aligned} \boldsymbol{H}(\boldsymbol{\theta}; \boldsymbol{X}) &= \nabla^2_{\boldsymbol{\theta}} \left(\ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right) \\ &= \nabla_{\boldsymbol{\theta}} \left(\nabla_{\boldsymbol{\theta}} \left(\ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)\right) \\ &= \nabla_{\boldsymbol{\theta}} \left(\frac{1}{p(\boldsymbol{X} \mid \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} p(\boldsymbol{X} \mid \boldsymbol{\theta})\right) \\ &= \frac{1}{p(\boldsymbol{X} \mid \boldsymbol{\theta})} \nabla^2_{\boldsymbol{\theta}} \left(p(\boldsymbol{X} \mid \boldsymbol{\theta})\right) - \frac{1}{\left(p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)^2} \nabla_{\boldsymbol{\theta}} p(\boldsymbol{X} \mid \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} p(\boldsymbol{X} \mid \boldsymbol{\theta}) \\ &= \frac{1}{p(\boldsymbol{X} \mid \boldsymbol{\theta})} \nabla^2_{\boldsymbol{\theta}} \left(p(\boldsymbol{X} \mid \boldsymbol{\theta})\right) - \left(\nabla_{\boldsymbol{\theta}} \left(\ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)\right) \left(\nabla_{\boldsymbol{\theta}} \left(\ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)\right)^T \\ E_{\boldsymbol{X}} \left[\boldsymbol{H}(\boldsymbol{\theta}; \boldsymbol{X})\right] &= -E_{\boldsymbol{X}} \left[\left(\nabla_{\boldsymbol{\theta}} \left(\ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)\right) \left(\nabla_{\boldsymbol{\theta}} \left(\ln p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)\right)^T\right] + E_{\boldsymbol{X}} \left[\frac{1}{p(\boldsymbol{X} \mid \boldsymbol{\theta})} \nabla^2_{\boldsymbol{\theta}} \left(p(\boldsymbol{X} \mid \boldsymbol{\theta})\right)\right]. \end{aligned}$$

But, the second term in the last equation evaluated at the true value of the parameters $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ is $\boldsymbol{0}$ [RW84]. Therefore, rearranging the last equation, we have

$$F(\boldsymbol{\theta}^*) = E_{\boldsymbol{X}} \left[-\boldsymbol{H}(\boldsymbol{\theta}^*; \boldsymbol{X})\right].$$

Note also that in general people call the gradient of the log likelihood with respect to the parameters the *score* and the negative of the Hessian the *information* (or *observed information* when it is evaluated at the MLE).

D Convergence speed

The convergence speed of a method is said to be of order c if for the sequence of parameters $\boldsymbol{\theta}^{(k)}$ generated by the method, the converging value $\boldsymbol{\theta}^N$, some sequence of values $s^{(k)}$ and some norm $\|\cdot\|$ on the parameter space,

$$\left\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^N\right\| \leq s^{(k)} \left\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^N\right\|^c.$$

If $c = 1$, we say that the method has linear convergence rate. If $c = 2$, we say that the method has quadratic convergence rate. Since we assume that the method converges, $0 \leq s^{(k)} < 1$. Also, if $c = 1$ and $\lim_{k\to\infty} s^{(k)} = 0$, then we say that the method has super-linear convergence rate.

E Norms

In this paper we deal primarily with the Euclidean norm; that is, for a vector $\boldsymbol{x}$,

$$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}.$$

By a *matrix norm* we mean the norm of a matrix that is induced by the Euclidean norm; that is, for a vector $\boldsymbol{x}$ and a matrix $\boldsymbol{A}$,

$$\|\boldsymbol{A}\| = \max_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}.$$