A bumpy journey: exploring deep Gaussian mixture models

Margot Selosse
Université de Lyon, Lyon 2, ERIC UR3083
5 Avenue Pierre Mendès France, 69500 Bron
margot.selosse@gmail.com

Isobel Claire Gormley
School of Mathematics and Statistics
University College Dublin
Dublin 4
Ireland
claire.gormley@ucd.ie

Julien Jacques
Université de Lyon, Lyon 2, ERIC UR3083
5 Avenue Pierre Mendès France, 69500 Bron
julien.jacques@univ-lyon2.fr

Christophe Biernacki
Université de Lille, INRIA
40, av. Halley - Bât A - Park Plaza 59650 Villeneuve d'Ascq
christophe.biernacki@inria.fr

Abstract

The deep Gaussian mixture model (DGMM) is a framework directly inspired by the finite mixture of factor analysers model (MFA) and the deep learning architecture composed of multiple layers. The MFA is a generative model that considers a data point as arising from a latent variable (termed the score) which is sampled from a standard multivariate Gaussian distribution and then transformed linearly. The linear transformation matrix (termed the loading matrix) is specific to a component in the finite mixture. The DGMM consists of stacking MFA layers, in the sense that the latent scores are no longer assumed to be drawn from a standard Gaussian, but rather are drawn from a mixture of factor analysers model. Thus the latent scores are at one point considered to be the input of an MFA and also to have latent scores themselves. The latent scores of the DGMM's last layer only are considered to be drawn from a standard multivariate Gaussian distribution. In recent years, the DGMM has gained prominence in the literature: intuitively, this model should be able to capture complex distributions more precisely than a simple Gaussian mixture model. We show in this work that while the DGMM is an original and novel idea, in certain cases it is challenging to infer its parameters. In addition, we give some insights to the probable reasons of this difficulty. Experimental results are provided on github: https://github.com/ansubmissions/ICBINB, alongside an R package that implements the algorithm and a number of ready-to-run R scripts.

1 Mixture of Factor Analysers

The mixture of factor analysers model [3] is an extension of the factor analysis model to a finite mixture of G factor analysis models. We assume that an observation $\boldsymbol{x}_i \in \mathbb{R}^J$ can be expressed through a latent random vector $\boldsymbol{z}_i \in \mathbb{R}^R$ such that $R \leq J$. In addition, the latent labels $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ are assumed to be independent unobserved realisations of a categorical G-dimensional random vector,

1st I Can't Believe It's Not Better Workshop (ICBINB@NeurIPS 2020), Vancouver, Canada.

such that $v_{ig} = 1$ indicates that $\boldsymbol{x}_i$ is generated by the gth factor analyser. The generative process of this model is as follows:

$$\begin{aligned}
&\boldsymbol{v}_i \sim \mathcal{M}(1, (\pi_1, \ldots, \pi_G)),\\
&\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_R),\\
&\boldsymbol{u}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\psi}_g),\\
&\boldsymbol{x}_i = \boldsymbol{\eta}_g + \boldsymbol{\Lambda}_g \boldsymbol{z}_i + \boldsymbol{u}_i,
\end{aligned}$$

where g is the index of the element of $\boldsymbol{v}_i$ that is equal to 1, $\boldsymbol{\Lambda}_g$ is a $J \times R$ matrix and $\boldsymbol{\eta}_g \in \mathbb{R}^J$ is the mean vector of the gth factor analysis model. In addition, $\boldsymbol{\psi}_g$ is a diagonal covariance matrix of dimension J, and $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_G)$ are the mixing proportions. The latent variable $\boldsymbol{v}_i = (\boldsymbol{v}_{i1}, \ldots, \boldsymbol{v}_{iG})$ represents the partitions for observation i and $p(v_{ig} = 1) = \pi_g$. The marginal density of $\boldsymbol{x}_i$ is then a Gaussian mixture model (GMM):

$$f(\boldsymbol{x}_i; \boldsymbol{\theta}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\eta}_g, \boldsymbol{\Sigma}_g),$$

where $\boldsymbol{\theta} = (\boldsymbol{\eta}_g, \boldsymbol{\Lambda}_g, \boldsymbol{\psi}_g, \pi_g)_{g \in \{1,\ldots,G\}}$ denotes all parameters and:

- $f_g(.; \boldsymbol{\eta}_g, \boldsymbol{\Sigma}_g)$ is the Gaussian probability density function for component g with mean $\boldsymbol{\eta}_g$ and covariance matrix $\boldsymbol{\Sigma}_g$ and
- $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^T + \boldsymbol{\psi}_g$.

Hence, the observed-data log-likelihood of N samples $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ is given by:

$$l_o(\boldsymbol{\theta}; \boldsymbol{x}) = \sum_{i=1}^{N} \log \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\eta}_g, \boldsymbol{\Sigma}_g).$$

2 Deep Gaussian Mixture Models

2.1 Coupling Deep Learning and Gaussian Mixture Models

One of the earliest papers to define a model that stacks MFA layers in a deep learning fashion is that of [6] where the authors refer to the model as the "Deep Mixture of Factor Analysers". In this model, the layers are not fully-connected and the architecture is a tree: a neuron receives an input from only one neuron of the previous layer. In [7], the authors define a DGMM model with fully-connected layers, but the matrices of weights $\boldsymbol{\Lambda}$ are assumed to be square. In other words, there is no dimension reduction between each MFA layer, which leads to important unidentifiability issues. In the model proposed by [9], instead of using the MFA model as a layer, the authors suggest using a mixture of factor analysers model with common factor loadings (MCFA) [1]: the matrices $\boldsymbol{\Lambda}_g$ of the same layer are therefore assumed to be equal for all g. Finally, [8] defines the deep Gaussian mixture model (DGMM) with dimension reduction at each layer and provides an EM algorithm for the inference of the parameters. This present work aims at investigating the statistical efficiency of this algorithm.

2.2 Definition of the Deep Gaussian Mixture Model

This section relies on [8]. Let us assume that there are L layers and that the data $\boldsymbol{x} = (\boldsymbol{x}_i)_{i \in \{1,\ldots,N\}}$ are generated as follows:

$$\begin{aligned}
&\boldsymbol{x}_i = \boldsymbol{z}_i^{(0)} = \boldsymbol{\eta}_{g_1}^{(1)} + \boldsymbol{\Lambda}_{g_1}^{(1)} \boldsymbol{z}_i^{(1)} + \boldsymbol{u}_i^{(1)} \text{ with prob. } \pi_{g_1}^{(1)},\ g_1 \in \{1, \ldots, G^{(1)}\}\\
&\boldsymbol{z}_i^{(1)} = \boldsymbol{\eta}_{g_2}^{(2)} + \boldsymbol{\Lambda}_{g_2}^{(2)} \boldsymbol{z}_i^{(2)} + \boldsymbol{u}_i^{(2)} \text{ with prob. } \pi_{g_2}^{(2)},\ g_2 \in \{1, \ldots, G^{(2)}\}\\
&\ldots\\
&\boldsymbol{z}_i^{(l)} = \boldsymbol{\eta}_{g_{l+1}}^{(l+1)} + \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)} \boldsymbol{z}_i^{(l+1)} + \boldsymbol{u}_i^{(l+1)} \text{ with prob. } \pi_{g_{l+1}}^{(l+1)},\ g_{l+1} \in \{1, \ldots, G^{(l+1)}\} \qquad (1)\\
&\ldots\\
&\boldsymbol{z}_i^{(L-1)} = \boldsymbol{\eta}_{g_L}^{(L)} + \boldsymbol{\Lambda}_{g_L}^{(L)} \boldsymbol{z}_i^{(L)} + \boldsymbol{u}_i^{(L)} \text{ with prob. } \pi_{g_L}^{(L)},\ g_L \in \{1, \ldots, G^{(L)}\}, \text{ where:}
\end{aligned}$$

- $\boldsymbol{z}_i^{(L)}$ is a vector of size $R^{(L)}$ and is assumed to be drawn from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_{R^{(L)}})$,
- $\boldsymbol{u}_i^{(l)}$ is a specific random error that follows a Gaussian distribution with expectation $\mathbf{0}$ and covariance matrix $\boldsymbol{\psi}_{g_l}^{(l)}$,
- $\boldsymbol{\eta}_{g_l}^{(l)}$ is a vector of length $R^{(l)}$,
- $\boldsymbol{\Lambda}_{g_l}^{(l)}$ is a matrix of dimension $R^{(l-1)} \times R^{(l)}$,
- $J > \ldots > R^{(l-1)} > R^{(l)}$ for all l.

The vectors $\boldsymbol{u}_i^{(l)}$ are assumed to be independent of the scores $\boldsymbol{z}_i^{(l)}$. From these considerations, we see that at each layer l, the conditional distribution of $\boldsymbol{z}_i^{(l)}$ given $\boldsymbol{z}_i^{(l+1)}$ is a mixture of Gaussian distributions such that:

$$f(\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l+1)};\boldsymbol{\theta}) = \sum_{g_{l+1}=1}^{G^{(l+1)}} \boldsymbol{\pi}_{g_{l+1}} \mathcal{N}(\boldsymbol{z}_i^{(l)}; \boldsymbol{\eta}_{g_{l+1}}^{(l+1)} + \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)} \boldsymbol{z}_i^{(l+1)}, \boldsymbol{\psi}_{g_{l+1}}^{(l+1)}). \tag{2}$$

Figure 1 represents the graphical model of this DGMM in the case of $L = 2$.

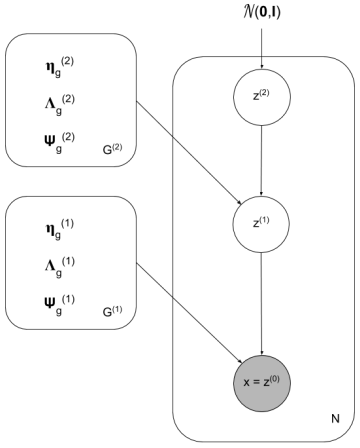

Figure 1: Graphical model of an example of the DGMM.

Therefore, the DGMM has parameters $\boldsymbol{\theta} = (\boldsymbol{\eta}_{g_l}^{(l)}, \boldsymbol{\Lambda}_{g_l}^{(l)}, \boldsymbol{\psi}_{g_l}^{(l)}, \pi_{g_l}^{(l)})_{l\in\{1,\ldots,L\};g_l\in\{1,\ldots,G^{(l)}\}}$. It also has the latent variables $\boldsymbol{v}^{(l)}$ and $\boldsymbol{z}^{(l)}$ where $\boldsymbol{v}^{(l)}$ corresponds to the partition of the lth layer.

In addition, it is important to notice that the marginal probability distribution $f(\boldsymbol{x}_i;\boldsymbol{\theta})$ is a GMM where each component corresponds to one of the possible paths of the network. Therefore, by noting $\mathcal{G}$ as the set of all possible paths through the network, the DGMM can be written:

$$f(\boldsymbol{x}_i;\boldsymbol{\theta}) = \sum_{\boldsymbol{g}\in\mathcal{G}} \pi_{\boldsymbol{g}} \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{g}}, \boldsymbol{\Sigma}_{\boldsymbol{g}}), \tag{3}$$

where $\boldsymbol{g} = (g_1, \ldots, g_L)$ is one of the possible paths of the network,

$$\pi_{\boldsymbol{g}} = \prod_{l=1}^{L} \pi_{g_l}^{(l)}, \tag{4}$$

$$\boldsymbol{\mu}_{\boldsymbol{g}} = \boldsymbol{\eta}_{g_1}^{(1)} + \boldsymbol{\Lambda}_{g_1}^{(1)} \left(\boldsymbol{\eta}_{g_2}^{(2)} + \boldsymbol{\Lambda}_{g_2}^{(2)} \big(\ldots (\boldsymbol{\eta}_{g_{L-1}}^{(L-1)} + \boldsymbol{\Lambda}_{g_{L-1}}^{(L-1)} \boldsymbol{\eta}_{g_L}^{(L)}) \big) \right) \text{ and} \tag{5}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{g}} = \boldsymbol{\Lambda}_{g_1}^{(1)} \left(\boldsymbol{\Lambda}_{g_2}^{(2)} \big(\ldots (\boldsymbol{\Lambda}_{g_L}^{(L)} \boldsymbol{\Lambda}_{g_L}^{(L)^T} + \boldsymbol{\psi}_{g_L}^{(L)}) \ldots \big) \boldsymbol{\Lambda}_{g_2}^{(2)^T} + \boldsymbol{\psi}_{g_2}^{(2)} \right) \boldsymbol{\Lambda}_{g_1}^{(1)^T} + \boldsymbol{\psi}_{g_1}^{(1)}. \tag{6}$$

We refer to the GMM of Equation (3) as the "global GMM" of the DGMM so as not to confuse it with the GMMs of the layers as expressed in Equation (2). In addition, the number of possible paths through the network is denoted as G. In fact, Equation (3) can be generalised to all layers: the marginal distributions of all the scores $\boldsymbol{z}_i^{(l)}$ are Gaussian mixtures. So, by noting $\tilde{\mathcal{G}}^{(l)}$ as all the possible paths from the lth layer, and by integrating out the latent variables of the layers that follow the lth layer, we have:

$$f(\boldsymbol{z}_i^{(l)};\boldsymbol{\theta}) = \sum_{\tilde{\boldsymbol{g}}^{(l)} \in \tilde{\mathcal{G}}^{(l)}} \tilde{\pi}_{\tilde{\boldsymbol{g}}^{(l)}} \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\tilde{\boldsymbol{g}}^{(l)}}^{(l+1)}, \tilde{\boldsymbol{\Sigma}}_{\tilde{\boldsymbol{g}}^{(l)}}^{(l+1)}), \tag{7}$$

where $\tilde{\boldsymbol{g}}^{(l)} = (g_{l+1}, \ldots, g_L)$ is a possible path of the network from the lth layer,

$$\tilde{\pi}_{\tilde{\boldsymbol{g}}^{(l)}} = \prod_{l'=l+1}^{L} \pi_{g_{l'}}, \tag{8}$$

$$\tilde{\boldsymbol{\mu}}_{\tilde{\boldsymbol{g}}^{(l)}} = \boldsymbol{\eta}_{g_{l+1}}^{(l+1)} + \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)} \left(\boldsymbol{\eta}_{g_{l+2}}^{(l+2)} + \boldsymbol{\Lambda}_{g_{l+2}}^{(l+2)} \big(\ldots (\boldsymbol{\eta}_{g_{L-1}}^{(L-1)} + \boldsymbol{\Lambda}_{g_{L-1}}^{(L-1)} \boldsymbol{\eta}_{g_L}^{(L)}) \big) \right) \text{ and} \tag{9}$$

$$\tilde{\boldsymbol{\Sigma}}_{\tilde{\boldsymbol{g}}^{(l)}} = \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)} \left(\boldsymbol{\Lambda}_{g_{l+2}}^{(l+2)} \big(\ldots (\boldsymbol{\Lambda}_{g_L}^{(L)} \boldsymbol{\Lambda}_{g_L}^{(L)^T} + \boldsymbol{\psi}_{g_L}^{(L)}) \ldots \big) \boldsymbol{\Lambda}_{g_{l+2}}^{(l+2)^T} + \boldsymbol{\psi}_{g_{l+2}}^{(l+2)} \right) \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)^T} + \boldsymbol{\psi}_{g_{l+1}}^{(l+1)}. \tag{10}$$

2.3 Inference of the model

Similar to MFA, the DGMM has parameters and latent variables to estimate. Therefore, the EM algorithm [2] is a candidate to optimise the log-likelihood function with respect to the parameters and latent variables. However, in the MFA model, the computation of the expectation of the complete data log-likelihood does not involve $p(\boldsymbol{z};\boldsymbol{\theta})$ since $\boldsymbol{z}$ is assumed to be drawn from the standard Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. For the DGMM, unless $l = L$, $\boldsymbol{z}_l^{(l)}$ is assumed to be drawn from an MFA with parameters $(\pi_{g_{l+1}}^{(l+1)}, \boldsymbol{\eta}_{g_{l+1}}^{(l+1)}, \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)}, \boldsymbol{\psi}_{g_{l+1}}^{(l+1)})_{g_{l+1} \in \{1,\ldots,G^{(l+1)}\}}$. Thus, the parameters of the model are involved in the computation of $p(\boldsymbol{z}^{(l)};\boldsymbol{\theta})$ which makes it difficult to optimise the auxiliary function with all the layers at the same time. A solution to this issue is to perform the optimisation layer by layer in iterations of the EM algorithm. In this case, when we optimise the auxiliary function with respect to the parameters of layer l, the fact that $p(\boldsymbol{z}^{(l)};\boldsymbol{\theta})$ depends on the parameters of layer $l+1$ is not a problem and an EM algorithm can be performed. We refer to [8] for further details about the EM algorithm steps. Finally, choosing the number of clusters for each layer can be done with a model selection criterion such as the BIC [4].

2.4 Interpreting the inferred partitions under the DGMM

There are two main ways to use the inferred partitions of the DGMM. The first way focuses on the first layer of the network, and then have a look at the partitions $\boldsymbol{v}^{(1)}$. In this case, the number of clusters is equal to $G^{(1)}$ and the distribution of the observations of a cluster is assumed not to be Gaussian since the latent scores $\boldsymbol{z}^{(1)}$ are assumed to be drawn through an MFA, as detailed in Equation (2). The second way of using the DGMM is to observe the clusters of the global GMM of Equation (3). In this case, the number of clusters is equal to $G^{(1)} \times \ldots \times G^{(L)}$, and the observations of a cluster are now assumed to follow a Gaussian distribution.

3 Experiments with the Deep Gaussian Mixture Model

We simulate data via the generative process of the DGMM with known parameters and partitions. Then, we run the EM algorithm to see if it estimates the parameters and partitions well. Because the number of parameters is large, we focus on the Adjust Rand Index (ARI), which gives an indicator of the performances of the algorithm on the estimation of the partitions and hence on the estimation of the parameters too. In these experiments we start by focusing on the partitions of the first layer, and then we focus on the partitions of the global GMM.

3.1 Settings for the simulated data set

For the data set $\boldsymbol{x}$, we have $N = 4,000$ and $J = 17$. The network was built with three layers ($L = 3$) and $G^{(1)} = 3$, $G^{(2)} = 3$ and $G^{(3)} = 2$. In addition, the latent scores dimensions were set to $R^{(1)} = 11$, $R^{(2)} = 6$ and $R^{(2)} = 2$ respectively; note this DGMM is identified in a factorial sense.. The parameters $\pi_g^{(l)}$ were set to $1/G^{(l)}$ for all g and for all l. The parameters $(\boldsymbol{\eta}_g^{(l)}, \boldsymbol{\Lambda}_g^{(l)}, \boldsymbol{\psi}_g^{(l)})_{g,l}$ were sampled from Gaussian distributions whose parameters can be found in the supplementary material files.

3.2 Results on the first layer

In this section, we focus on the partitions of the first layer $\boldsymbol{v}^{(1)}$. We run 20 times the DGMM EM algorithm and the classic GMM by using the `R` package `mclust` [5]. For the DGMM runs, we fix $G^{(2)}$ and $G^{(3)}$ to their ground-truth values and vary $G^{(1)}$ from 2 to 6. In addition, the algorithm is run with random starting values. For the GMM runs, we vary the number of components G from 2 to 6. Table 1 shows for each run the number of clusters with the best BIC value. We see that `mclust` was not able to find the true number of clusters. This result is expected since the clusters of the first layer are not supposed to be Gaussian: `mclust` overfits the data by finding more clusters than there truly are. However, the DGMM seems to yield the same behaviour, which is not expected since its first layer is supposed to capture non-Gaussian distributions.

DGMM	6	6	6	5	6	4	4	5	5	6	5	6	5	6	5	6	6	6	6	5
`mclust`	6	6	6	6	6	6	6	6	6	6	5	6	6	6	6	6	6	5	5	6

Table 1: Number of clusters chosen with BIC for each run of the DGMM and `mclust`.

Figure 2 represents the ARI of the runs with the best BIC value. We see that the DGMM yields better results than `mclust`. However, these results are not satisfying because the data was generated through the DGMM generative process: one would hope that at least some of the runs should achieve an ARI equal to 1. Let us note that similar behaviour was observed across a large number of data sets generated with a range of settings of the DGMM architecture but for brevity only one such is presented here.

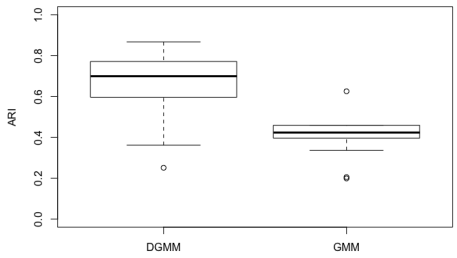

Figure 2: ARI of the DGMM's first layer and of the `mclust` result.

3.3 Results on the global GMM

For every experiment described below, the number of clusters $(G^{(l)})_l$ are fixed to their true value. The DGMM's inference algorithm is run 20 times with random starting values.

Figure 3 shows the resulting ARI. The DGMM yields poor results since the highest ARI value achieved is 0.67 even though the data set was generated through the DGMM's generative process.

In Figure 4, we plot the BIC values over the iterations of the EM algorithm for a single random initialisation; the BIC reaches a plateau, meaning that the EM algorithm achieves at least a local maximum of the likelihood.

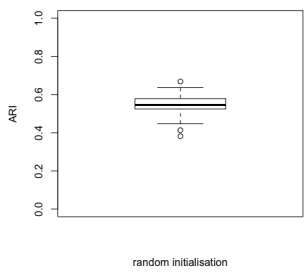

Figure 3: Boxplot of the ARI for the global GMM of the DGMM using random initialisations.

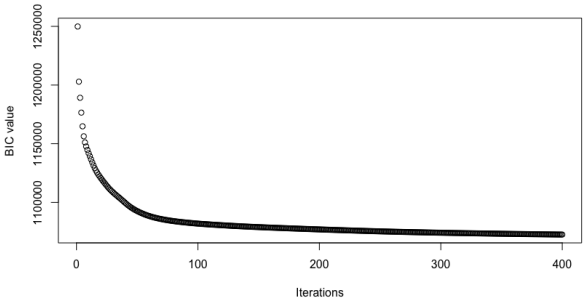

Figure 4: BIC values over the iterations of the EM algorithm using a single random initialisation.

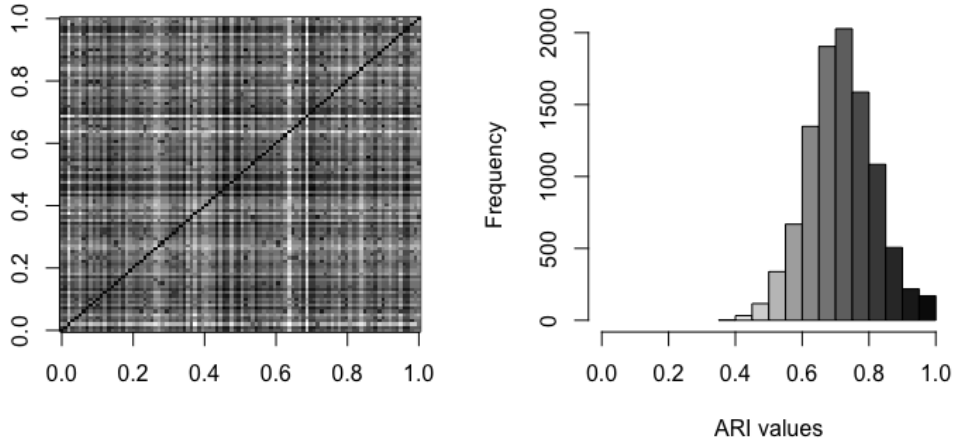

Figure 5: Mutual ARI of 100 runs of the EM algorithm. None of them gets the same final partition.

If the EM algorithm is able to find local maxima, but fails to find the right partitions, it might mean that there are many local maxima. To verify this, we run the EM algorithm a hundred times and compute the ARI of each resulting clustering with respect to every other resulting clustering. The mutual ARI are plotted in Figure 5 (left and right), where we see that none of the mutual ARI values are 1 (except the ARI of a result with respect to itself). This means that the EM algorithm found 100 *different* local maxima leading to very different partitions.

The last result we detail regards another unexpected behaviour of the DGMM's inference algorithm. When we list the resulting partition for each observation on each layer, we can see that all the clusters of every layer are represented. However, when we list the resulting partitions of the global GMM as in Table 2, we see that not all the clusters are represented. This means that even if each neuron of each layer has observations, not every combination of neurons of the three layers does. Therefore, there is a severe problem of empty clusters at the global GMM's level.

Finally, let us note that the package `mclust` was not able to find a solution for $G = \prod_l G^{(l)} = 18$. From this result, the DGMM can be seen as a candidate when the number of clusters is high, even if the estimation of partitions is not perfect.

Cluster numbers
1, 2, 4, 5, 6, 8, 10, 11, 12, 13, 14, 18
1, 2, 3, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16
1, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15, 16, 17
2, 3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16, 18
1, 2, 3, 5, 6, 7, 9, 10, 12, 14, 15, 16, 18
1, 2, 4, 6, 7, 9, 10, 11, 13, 14, 15, 16, 17, 18
1, 2, 5, 6, 8, 10, 11, 12, 14, 15, 17
2, 3, 4, 5, 6, 9, 11, 12, 13, 14, 16, 18
2, 3, 4, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18
1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18
1, 2, 3, 5, 6, 7, 9, 10, 11, 14, 15, 16, 18
2, 3, 5, 6, 7, 8, 9, 11, 12, 14, 15, 16
1, 2, 4, 5, 6, 7, 8, 10, 11, 13, 14, 15, 16, 17, 18
1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 14, 17, 18
1, 2, 4, 5, 8, 9, 10, 11, 13, 14, 17, 18
1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 17
1, 2, 3, 4, 6, 7, 9, 10, 11, 12, 15, 16, 17
1, 4, 5, 8, 10, 13, 14, 15, 17, 18
1, 3, 4, 7, 8, 9, 10, 11, 12, 13, 15, 17, 18
1, 5, 7, 8, 9, 10, 14, 16, 17, 18

Table 2: The non-empty clusters that are represented in the global GMM for the 20 runs.

ARI of layer 1	ARI of layer 2	ARI of layer 3
0.18	0.5	0
0.17	0.17	0.01
0.2	0.25	0.01
0.45	0.19	0
0.32	0.29	0.01
0.38	0.09	0
0.43	0.07	0
0.4	0.22	0
0.55	0.17	0.01
0.11	0.24	0.01
0.17	0.37	0
0.17	0.19	0
1	0.08	0.01
0.19	0.42	0.01
0.5	0.51	0
0.22	0.51	0
0.4	0.21	0
0.29	0.09	0
0.55	0.07	0
0.52	0.21	0

Table 3: ARI results for the GMM of each layer. We see that the latent scores $(\boldsymbol{z}^{(l)})_l$ do not seem to be estimated well.

3.4 Possible reasons for the poor results

There are several possible reasons for the poor results described in the previous section. First of all, the MFA model already suffers from unindentifiability issues that could worsen with the multilayer architecture of the DGMM. The most common solutions to tackle the unidentifiablity issues are:

- To constrain $\boldsymbol{\Lambda}_g^{(l)}$ so that it is orthonormal and to order the columns by decreasing variance of the corresponding latent scores.
- To constrain ${\boldsymbol{\Lambda}_g^{(l)}}^T \boldsymbol{\psi}_g^{(l)} \boldsymbol{\Lambda}_g^{(l)}$ to be diagonal.
- To constrain $\boldsymbol{\Lambda}_g^{(l)}$ so that it is a lower triangular full rank matrix with diagonal elements strictly positive.
- To choose an informative rotation matrix. Many heuristic methods try to find rotation matrices $\boldsymbol{P}$ that can be used to modify $\boldsymbol{\Lambda}_g^{(l)}$ (and the latent factors) so as to try to increase the interpretability.

While we already tried to apply these constraints (results not detailed in this document), we did not obtained better results regarding the partitions and parameters estimation.

Second, the DGMM has many latent variables, which can be a severe problem when it comes to estimation. Indeed, in the context of the configuration of Section 3.1, the latent scores alone represent $N \times (R^{(1)} + R^{(2)} + R^{(3)}) = 4,000 \times (11 + 6 + 2) = 76,000$ continuous values to estimate, which is very high. This large number of latent variables is for us one of the most important obstacles to correct estimation. To support this point, Table 3 shows the ARI that the EM algorithm achieved at each layer. At the first layer, the ARIs are globally higher than the other layers and can even be equal to 1. However, the ARI for the other layers worsens with the depth: the second layer's higher ARI value is 0.51 and the third layer's ARI value are all near to zero. This probably means that the latent scores are poorly estimated since the DGMM of layer 1 depends on the data $\boldsymbol{x}$, the DGMM of layer 2 depends on the scores $\boldsymbol{z}^{(1)}$ and the DGMM of layer 3 depends on the scores $\boldsymbol{z}^{(2)}$. A solution to bypass the latent score estimation is to use another kind of algorithm, such as the Newton-Raphson algorithm and its variants. However, we encountered different problems when using this kind of algorithm such as computational errors due to matrix inversions.

Finally, we have also observed that even though the clusters of each layer do not necessarily become empty, the clusters of the global GMM do so, which is also a problem for estimation of the partition.

4 Conclusion

In theory, the DGMM is a powerful model that can estimate complex distributions by stacking MFA layers in a deep learning fashion. We have seen in this Section 3.3 that this algorithm is able to provide an estimation of partitions and parameters in cases where mclust is not able to do so, for instance when the number of clusters is high. However, this estimation is not correct and we have seen that the EM algorithm suffers from drawbacks and can struggle to find the global maxima.

Acknowledgments and Disclosure of Funding

We would like to warmly thank Cinzia Viroli and Geoff McLachlan for their helpful discussions from which the work benefitted. ICG was supported by the Science Foundation Ireland funded Insight Centre for Data Analytics (SFI/12/RC/2289_P2).

References

[1] Jangsun Baek, G. Mclachlan, and Lloyd Flack. Mixtures of factor analyzers with common factor loadings: Applications to the clustering and visualization of high-dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 32:1298–309, 07 2010.

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

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

[4] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[5] Luca Scrucca, Michael Fop, Thomas Brendan Murphy, and Adrian E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):205–233, 2016.

[6] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep mixtures of factor analysers. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1123–1130, USA, 2012. Omnipress.

[7] Aaron van den Oord and Benjamin Schrauwen. Factoring variations in natural images with deep Gaussian mixture models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3518–3526. Curran Associates, Inc., 2014.

[8] Cinzia Viroli and Geoffrey J. McLachlan. Deep Gaussian mixture models. *Statistics and Computing*, 29(1):43–51, Jan 2019.

[9] Xi Yang, Kaizhu Huang, and Rui Zhang. Deep mixtures of factor analyzers with common loadings: A novel deep generative approach to clustering. In *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part I*, pages 709–719, 2017.