Mixed data Deep Gaussian Mixture Model: A clustering model for mixed datasets

Robin Fuchs
CNRS, Centrale Marseille, I2M, MIO, Aix-Marseille Univ. and
Denys Pommeret
SAF, Univ. Lyon 1.
and
Cinzia Viroli
Department of Statistical Sciences, Univ. of Bologna.

October 15, 2020

Abstract

Clustering mixed data presents numerous challenges inherent to the very heterogeneous nature of the variables. Two major difficulties lie in the initialisation of the algorithms and in making variables comparable between types. This work is concerned with these two problems. We introduce a two-heads architecture model-based clustering method called Mixed data Deep Gaussian Mixture Model (MDGMM) that can be viewed as an automatic way to merge the clusterings performed separately on continuous and non continuous data. We also design a new initialisation strategy and a data driven method that selects "on the fly" the best specification of the model and the optimal number of clusters for a given dataset. Besides, our model provides continuous low-dimensional representations of the data which can be a useful tool to visualize mixed datasets. Finally, we validate the performance of our approach comparing its results with state-of-the-art mixed data clustering models over several commonly used datasets.

Keywords: Mixed data; Deep Gaussian Mixture Model; MCEM algorithm.
Mixed datasets are made of variables of heterogeneous nature that can be split into two categories: i) the discrete data that contain binary data (that takes either the value 1 or the value 0), count data (taking values in $\mathbb{N}$), categorical data (non-ordered data taking a finite number of modalities), and ordinal data (ordered data taking a finite number of modalities); ii) the continuous data (generated by real-valued random variables). Due to their difference in nature, these variables do not share common scales and it is typically hard to characterise what similarity between observations is.

There has been a significant recent interest for mixed data clustering, which can be regrouped in four main categories according to Ahmad and Khan (2019): partitional algorithms, hierarchical algorithms, model-based algorithms and Neural Networks. Partitional algorithms try to find cluster centers and define a distance between observations and those centers. They optimise iteratively a metric based on the distance defined, as in K-modes or K-prototypes (Huang, 1998). Hierarchical algorithms perform nested clusterings and then find a way to merge them to create the final clustering (Philip and Ottaway, 1983; Chiu et al., 2001). Model-based models (Melnykov et al., 2010), as their name suggests it, rely on statistical foundations such as probability distributions or expectations. Finally, Neural Networks-based algorithms (Kohonen, 1990) design the clusters thanks to connected neurons that learn complex patterns contained in the data.

The Mixed data Deep Gaussian Mixture Model (MDGMM) we propose belongs to the model-based algorithms family and more precisely to the family of Generalized Linear Latent Variable Models (GLLVM) (Moustaki, 2003; Moustaki and Knott, 2000). This class of models supposes that there exists a link function between the discrete data space and a continuous latent space often chosen to be Gaussian. More recently, Cagnone and Viroli (2014) have extended this approach by considering a latent variable that is no more a Gaussian but a mixture of Gaussians (Fraley and Raftery, 2002). As a result, it enables the model to conduct clustering in the latent space. Since the latent dimension is chosen to be strictly inferior to the original dimension, the model also performs dimension reduction.

Our work generalizes this approach by considering a Deep Gaussian Mixture Model as latent variable. DGMMs have been introduced by Viroli and McLachlan (2019) and can be seen as a series of nested Mixture of Factor Analysis (MFA) (Ghahramani et al., 1996). As such, this approach also belongs to the mixture models literature.

To adapt the GLLVM to mixed data we propose a multilayer architecture inspired by Supervised Neural Networks and which relies on the same idea that composing simple functions enables to capture very complex patterns. Our model also borrows to Supervised Deep Learning methods the multi-inputs architecture used for instance in Deep Learning to train models with both images and text inputs. To analyze the two categories of variables together we define two "heads", one for discrete data (binary, categorical and ordinal data) and one for continuous data, and we add a common "tail" that merges the
information brought by the two heads. Note that the "heads and tail" denominations are here reversed compared to their common use in Supervised Deep Learning.

Our model implementation relies on automatic differentiation (Baydin et al., 2017) that helps keeping an acceptable running time even when the number of layers increases. Indeed, using auto-differentiation methods provided for instance by the autograd package or "Just-in-time" compilation as enabled by the numba package (or both methods as provided by JAX Bettencourt et al., 2019) truly cuts the computational running time. For instance, for the special case of GLLVM models, Niku et al. (2019) reported very significant computational gains from using auto-differentiation methods.

To summarize, this work has three main aims: it first generalizes the GLLVM and DGMM frameworks to deal with mixed data. Secondly, a new initialisation method is proposed to provide a suitable starting point for the MDGMM and more generally for GLLVM-based models. This initialization step combines Multiple Correspondence Analysis (MCA), GMM, MFA and the well known Partial Least Squares (PLS) algorithm. As mixed data are plunged into a multilayer continuous space we call this new initialisation Nested Space Embedding Procedure (NSEP). Thirdly, a model selection procedure is designed to identify the architecture of the model that best fit a given dataset.

The paper is organized as follows: Section 2 provides a detailed exposition of the model. In Section 3 we will be concerned with the EM algorithms used for estimation. Section 4 deals with the identifiability constraints of the model. Section 5 presents the initialization procedure NSEP and some practical considerations are given that can serve as a user manual. The performance of the model is compared to other competitor models in Section 6. In conclusion, Section 7 analyses the contributions of this work and highlights directions for future research.

2 Model presentation

2.1 The MDGMM as a generalization of existing models

In the sequel we assume that we observe \( n \) random variables, namely \( y_1, \ldots, y_n \), such that \( \forall i = 1, \ldots, n, y_i = (y^C_i, y^D_i) \), where \( y^C_i \) is a \( p_C \)-dimensional vector of continuous random variables and \( y^D_i \) is a \( p_D \)-dimensional vector of discrete random variables. Then each \( y_i \) is a a vector of dimension \( p_C + p_D \) of mixed variables. The architecture of the MDGMM is based on two models: first, the Mixture of Factor Analysis (MFA) (Ghahramani et al., 1996) generalized by the Deep Gaussian Mixture Models (DGMM) (Viroli and McLachlan, 2019), and second, the Generalized Linear Latent Variable Models (GLLVM) (Moustaki, 2003; Moustaki and Knott, 2000). MFA are mixture of Factor Analysis (FA) (Harman, 1976) models which represent the most elementary building block of our model and that can be first formulated as follows:
∀i ∈ [1, n] we have:

\[ y_i^C = \eta + \Lambda z_i + u_i, \]

where \( \eta \) is a constant vector of dimension \( p_C \), \( z_i \sim N(0, I_r) \), \( u_i \sim N(0, \Psi) \) and \( \Lambda \) is the factor loading matrix of dimension \( p_C \times r \), \( r \) being the dimension of the latent space. The underlying idea is to find a latent representation of the data of dimension \( r \), with \( r < p \). The loading matrix is then used to interpret the relationship existing between the data and their new representation. This model can be extended assuming that different groups of observations in the data have different latent representations, yielding to the following basic MFA model:

\[ y_i^C = \eta_{k_1} + \Lambda_{k_1} z_i + u_{ik_1}, \]

for \( k_1 \in [1, K_1] \), with \( z_i \sim N(0, I_p) \), \( u_{ik_1} \sim N(0, \Psi) \), and \( \Lambda_{k_1} \) is the factor loading of dimansion \( p \times r_1 \), with \( r_1 < p \).

The DGMM approach consists in extending once again the MFA model by assuming that \( z_i \) is no more drawn from a multivariate Gaussian but is itself a MFA. By repeating \( L \) times this hypothesis we obtain a \( L \)-layers DGMM defined by:

\[
\begin{align*}
    y_i^C &= \eta_{k_1}^{(1)} + \Lambda_{k_1}^{(1)} z_i^{(1)} + u_{ik_1}^{(1)}, \quad \text{with probability } \pi_{i,k_1}^{(1)} \\
    z_i^{(1)} &= \eta_{k_2}^{(2)} + \Lambda_{k_2}^{(2)} z_i^{(2)} + u_{ik_2}^{(2)}, \quad \text{with probability } \pi_{i,k_2}^{(2)} \\
    \vdots \notag \\
    z_i^{(L-1)} &= \eta_{k_{L}}^{(L)} + \Lambda_{k_{L}}^{(L)} z_i^{(L)} + u_{ik_{L}}^{(L)}, \quad \text{with probability } \pi_{i,k_{L}}^{(L)} \\
    z_i^{(L)} &\sim N(0, I_{r_L}), 
\end{align*}
\]

where, for \( \ell = 1, \ldots, L \), \( k_{\ell_1} \in [1, K_{\ell}] \), \( u_{ik_{\ell}} \sim N(0, \Psi_{k_{\ell}}) \), \( z_i^{(L)} \sim N(0, I_{r_L}) \) and where the factor loading matrices \( \Lambda_{k_{\ell}}^{\ell} \) have dimension \( r_{\ell-1} \times r_\ell \), with the constraint \( p > r_1 > r_2 > \ldots > r_L \). The constraints on the parameters \( \Lambda_{k_{\ell}}^{\ell} \) and \( \Psi_{k_{\ell}} \) will be discussed in Section 4.

The DGMM described in (1) can only handle continuous data. In order to apply a DGMM to discrete data we propose to integrate a Generalized Linear Latent Variable Model (GLLVM) framework within (1).

A GLLVM assumes that, \( \forall j \in [1, p_D] \), the discrete random variables \( y_j^D \) are associated to a continuous latent variable through an exponential family link (see the illustrations given in Cagnone and Viroli (2014)). Discrete random variables are then assumed to be independent from each other conditionally to the latent variable. We consider this latent variable as the first level of a DGMM, and mimicking the notation in (1) we denote it by \( z_i^{(1),D} \). Hence, one can use the previous DGMM architecture to treat discrete data and the GLLVM layer can be seen as a trainable embedding layer from discrete to continuous spaces.

Finally, in order to deal with mixed data, we propose to apply simultaneously (1) to continuous data and (1) combined with a GLLVM to discrete data. The two clustering processes can then be merged by...
specifying that the last layers are actually shared by the two networks. These ending layers, that will be referred to as the common tail layers, enable to synthesise information carried by both the discrete and continuous heads and to perform the clustering.

Intuitively, the two heads extract features from the data and pass them to the common tail. The tail reconciles both information sources, designs common features and performs the clustering. As such, any layer on the tail could in principle be used as clustering layer. As detailed in section 5.1, one could even use several tail layers to perform several clustering procedures (with different latent dimensions or number of clusters) in the same model run.

2.2 Formal definition

Let $y$ be the $n \times p$ matrix of the observed variables. We will denote by $i \in [1, n]$ the observation index and by $j \in [1, p]$ the variable index. We can decompose $y = (y^C|y^D)$ where $y^C$ is the $n \times p_c$ matrix of continuous variables and $y^D$ is the $n \times p_D$ matrix of discrete variables.

The global architecture of the MDGMM is similar to [1] with an additional GLLVM step for the discrete head as follows:

Discrete head:

\[
\begin{align*}
&z_i^{(1)D} = \eta_k^{(1)D} + \Lambda_k^{(1)D} z_i^{(2)D} + u_{i,k_1}^{(1)D} \text{ with probability } \pi_{i,k_1}^{(1)D} \\
&..., \\
&z_i^{(L_D)D} = \eta_k^{(L_D)D} + \Lambda_k^{(L_D)D} z_i^{(L_D+1)D} + u_{i,k_{L_D}}^{(L_D)D} \text{ with probability } \pi_{i,k_{L_D}}^{(L_D)D} \\
&y_i^C = \eta_k^{(1)C} + \Lambda_k^{(1)C} z_i^{(1)C} + u_{i,k_1}^{(1)C} \text{ with probability } \pi_{i,k_1}^{(1)C} \\
&z_i^{(1)C} = \eta_k^{(1)C} + \Lambda_k^{(1)C} z_i^{(2)C} + u_{i,k_1}^{(1)C} \text{ with probability } \pi_{i,k_2}^{(2)C} \\
&..., \\
&z_i^{(L_C)C} = \eta_k^{(L_C)C} + \Lambda_k^{(L_C)C} z_i^{(L_C+1)C} + u_{i,k_{L_C}}^{(L_C)C} \text{ with probability } \pi_{i,k_{L_C+1}}^{(L_C+1)C}.
\end{align*}
\]

Continuous head:

\[
\begin{align*}
&z_i^{(0+1)} = \eta_k^{(0+1)} + \Lambda_k^{(0+1)} z_i^{(0+2)} + u_{k_{L_0+1}}^{(0+1)} \text{ with probability } \pi_{i,(k_{L_0+1},k_{L_0+2})}^{(0+1)} \\
&..., \\
&z_i^{(L-1)} = \eta_k^{(L-1)} + \Lambda_k^{(L-1)} z_i^{(L)} + u_{k_{L-1}}^{(L-1)} \text{ with probability } \pi_{i,(k_{L-1},k_L)}^{(L-1)} \\
&z_i^{(L)} \sim \mathcal{N}(0, I_{r_L}).
\end{align*}
\]

It is assumed that the random variables $(u_{k_\ell}^{(\ell)})_{k_\ell}$ are all independent. The two heads only differ from each other by the fact that for the discrete head, a continuous representation of the data has first to be determined before information is fed through the layers. The GLLVM layer is parametrized by $(\lambda_0, A_0)$. 
\( \lambda_0 = (\lambda_{0\text{bin}}, \lambda_{0\text{count}}, \lambda_{0\text{ord}}) \) contains the intercept coefficients for each discrete data sub-type. We make the distinction between different sub-types of discrete data: binary, count, categorical and ordinal data. While the other sub-types are handled explicitly, the categorical variables are first converted to binary variables and treated as such. \( \Lambda_0 \) is a matrix of size \( p' \times D' \times r_1 \), with \( r_1 \) the dimension of the first Discrete DGMM layer and \( p' \times D' \) the dimension of the discrete data once categorical variables have been converted to dummies.

The notations remain the same as in the previous subsection and only a superscript is added to specify for each variable the head or tail to which it belongs. For instance \( z^C = (z^{(1)C}, ..., (z^{(L_C)C}) \) is the set of latent variables of the continuous head. This subscript is omitted for the common head. The \( \ell \)-th layer of the head \( h \) contains \( K^h_\ell \) components which is the number of components of the associated mixture. \( L_D \) and \( L_C \) are the number of layers of the discrete and continuous head, respectively. Representing these head in parallel as in Figure 1 we see that \( L_0 = \max(L_C, L_D) \), that is, the first layer of the common tail is the \( L_0 + 1 \)-th layers of the model. For simplicity of notation, we assume in the sequel that

\[ L_C = L_D = L_0, \]

but all the results are easily obtained in the general case.

Each path from one layer to the next is the realization of a mixture. In this sense we introduce, \( s^{(\ell)h} \in [1, K^h_\ell] \) the latent variable associated with the index of the component \( k^h_\ell \) of the layer \( \ell \) of the head \( h \). More generally, the latent variable associated with a path going from the first layer to the last layer of one head \( h \) is denoted by \( s^h = (s^{(1)h}, ..., s^{(L_0)h}) \). Similarly, the random variable associated to a path going through all the common tail layers is denoted by \( s^{(L_0+1:)} = (s^{(L_0+1)}, ..., s^{(L)}) \). By extension, the variable associated with a full path going from the beginning of head \( h \) to the end of the common tail is \( s^{(1h:L)} = (s^h, s^{L_0+1}) \). \( s^{(1h:L)} \) belongs to \( \Omega^h \) the set of all possible paths starting from one head of cardinal \( S^h = \prod_{\ell=1}^L K^h_\ell \). The variable associated with a path going from layer \( \ell \) of head \( h \) to layer \( L \) will be denoted \( s^{(\ell h:L)} \). Finally, by a slight abuse of notation a full path going through the component \( k^h_\ell \) of the \( \ell \)-th layer of head \( h \) will be denoted: \( s^{(1:k^h_\ell:L)} \) or more simply \( s^{(k^h_\ell:)}. \)

We can notice that the probability of one component does not depend on the components from previous layers, that is: for \( h \in \{C, D\} \),

\[ p(s^h) = p(s^{(1)h}, s^{(2)h}, ..., s^{(L_0)h}) = \prod_{\ell=1}^{L_0} p(s^{(\ell)h}). \]

In order to be the more concise as possible, we group the parameters of the model by defining:
\[ \Theta_D = (\Theta_{emb}, \Theta_{DGMM}) = ((\lambda_0, \Lambda_0), (\eta_k^{(t)}D, \Lambda_k^{(t)}D, \Psi_k^{(t)}D)_{k,t \in [1,K], t \in [1,L_0]}), \]
\[ \Theta_C = (\eta_k^{(t)}C, \Lambda_k^{(t)}C, \Psi_k^{(t)}C)_{k,t \in [1,K], t \in [1,L_0]}, \]
\[ \Theta_{L_0+1} = (\eta_k^{(t)}L, \Lambda_k^{(t)}L, \Psi_k^{(t)}L)_{k,t \in [1,K], t \in [L_0+1,L]}. \]

with \(emb\) standing for embedding.

As an illustration, the graphical model of a MDGMM with \(K_C = (5,4), K_D = (4,3), K = (2,1), L_C = L_D = L_0 = 2, S_C = 40\) and \(S_D = 24\) is provided in Figure 1. The decreasing size of the \((z^{(t)})_t\) illustrates the decreasing dimensions of the latent variables.

![Graphical model of a Mixed data DGMM (MDGMM)](image)

Figure 1: Graphical model of a Mixed data DGMM (MDGMM)

The complete density of the model is given by:

\[
L(y^C, y^D, z^C, z^D, z^{(L_0+1)}, s^C, s^D, s^{(L_0+1)} | \Theta_C, \Theta_D, \Theta_{L_0+1};)
\]
\[
= L(y^C|z^{(1)}C, s^C, s^{(L_0+1)}, \Theta_C, \Theta_{L_0+1};)L(z^C|z^{(L_0+1)}, s^C, s^{(L_0+1)}, \Theta_C, \Theta_{L_0+1};)
\]
\[
\times L(y^D|z^{(1)}D, s^D, s^{(L_0+1)}, \Theta_D, \Theta_{L_0+1};)L(z^D|z^{(L_0+1)}, s^D, s^{(L_0+1)}, \Theta_D, \Theta_{L_0+1};)
\]
\[
\times L(z^{(L_0+1)}|s^C, s^D, s^{(L_0+1)}, \Theta_C, \Theta_D, \Theta_{L_0+1};)
\]
\[
\times L(s^C, s^D, s^{(L_0+1)} | \Theta_C, \Theta_D, \Theta_{L_0+1};),
\]

which comes from the fact that we assume the two heads of the model to be conditionally independent with respect to the tail layers. Moreover, the layers of both heads and tail share the Markov property derived from the graphical model that \((z^{(t)})_t \perp \perp z^{(t+2)}_h, \ldots, z^{(L)}_h | z^{(t+1)}_h, \forall h \in \{C, D, (L_0 + 1 :)\}\).
The aim of the training is to maximize the expected log-likelihood, i.e. to maximize:

$$\mathbb{E}_{z^{(L_0+1)}, s^{L_0+1}, y, y_D} \left[ \log L(y_C, y_D, z^{(L_0+1)}, s^{L_0+1}, \Theta_C, \Theta_D, \Theta_{L_0+1}) \right]$$

$$= \mathbb{E}_{z^{(1)}, s^{(1)}, y_D} \left[ \log L(y_D | z^{(1)}, s^{(L_0+1)}, \Theta_D, \Theta_{L_0+1}) \right]$$

$$+ \mathbb{E}_{z^{(1)}, s^{(1)}, y_D} \left[ \log L(y_C | z^{(1)}, s^{(L_0+1)}, \Theta_C, \Theta_{L_0+1}) \right]$$

$$+ \sum_{h \in \{C,D\}} \sum_{l=1}^{L_0} \mathbb{E}_{z^{(l)}, y_D, \Theta_h} \left[ \log L(z^{(l)} | \Theta_C, \Theta_D, \Theta_{L_0+1}) \right]$$

$$+ \mathbb{E}_{y_C, y_D} \left[ \log L(y | z^{(1)}, s^{(L_0+1)}, \Theta_C, \Theta_D, \Theta_{L_0+1}) \right]$$

$$+ \mathbb{E}_{y_C, y_D} \left[ \log L(y | s^{(L_0+1)}, \Theta_C, \Theta_D, \Theta_{L_0+1}) \right],$$

with a slight abuse of notation in the double sum as in fact $z^{(L_0+1)} = z^{(L_0+1)}C = z^{(L_0+1)}D$. $\hat{\Theta}_h$ are the provisional estimate of $\Theta_h$ through the iterations of the algorithm.

### 3 Model training

The model is trained using a Monte Carlo version of the EM algorithm (MCEM) introduced by [Wei and Tanner (1990)](wei1990monte). Three types of layers have here to be trained: the GLLVM layer, the regular DGMM layers and the common tail layers that join the two heads. These three types of layers are described in this Section.

#### 3.1 Generalized Linear Latent Variable Model Embedding Layer

This section is based on [Cagnone and Viroli (2014)](cagnone2014generalized) which itself is based on seminal works by [Moustaki (2003)](moustaki2003generalized) and [Moustaki and Knott (2000)](moustaki2000generalized). It presents the canonical framework of GLLVMs for binary, count and ordinal data. We recall that categorical variables are one-hot encoded and then treated as a collection of binary variables.

By the conditional independence assumption between discrete variables, the likelihood can be written as:

$$f(y_D | \Theta_D, \Theta_{L_0+1}) = \int_{z^{(1)}} f(y_D | z^{(1)}, \Theta_D, \Theta_{L_0+1}) f(z^{(1)} | \Theta_D, \Theta_{L_0+1}) dz^{(1)}$$

$$= \int_{z^{(1)}} \prod_{j=1}^{P_d} f(y_j | z^{(1)}, \Theta_D, \Theta_{L_0+1}) f(z^{(1)} | \Theta_D, \Theta_{L_0+1}) dz^{(1)}.$$
where $y_j^D$ is the $j$th component of $y^D$. The density $f(y_j^D|z^{(1)D}, \Theta_D, \Theta_{L_0+1})$ belongs to an exponential family and in our empirical study we chose a Bernoulli distribution for binary variables, a binomial distribution for count variables and an ordered multinomial distribution for ordinal data. The whole expressions of the densities can be found in Cagnone and Viroli [2014].

In order to train the GLLVM layer, we maximize

$$E_{z^{(1)D},s^D,s(L_0+1)|y^D,\Theta_D,\Theta_{L_0+1}}[\log L(y^D|z^{(1)D}, s^D, s^{L_0+1}; \Theta_D, \Theta_{L_0+1})]$$

$$= E_{z^{(1)D}|y^D,\Theta_D,\Theta_{L_0+1}}[\log L(y^D|z^{(1)D}, \Theta_D, \Theta_{L_0+1})]$$

$$= \int f(z^{(1)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) \log L(y^D|z^{(1)D}, \Theta_D, \Theta_{L_0+1}) dz^{(1)D},$$

the second equality being due to the fact that $y^D$ is related to $(s^D, s^{(L_0+1)})$ only through $z^{(1)D}$.

### 3.1.1 MC Step

Draw $M^{(1)}$ observations from $f(z^{(1)D}|s^D, s^{(L_0+1)}, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})$.

### 3.1.2 E step

Hence the E-step consists in determining $f(z^{(1)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})$, which can be rewritten as:

$$f(z^{(1)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) = \sum_{s'} f(z^{(1)D}|y^D, s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) f(s^{(1D:L)} = s'| y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}).$$

The Bayes rule for the first term gives:

$$f(z^{(1)D}|y^D, s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) = \frac{f(z^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) f(y^D|z^{(1)D}, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})}{f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1})},$$

and we have

$$(z^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) \sim N(\mu_{s'}^{(1D:L)}, \Sigma_{s'}^{(1D:L)}),$$

where the mean and covariance parameters $(\mu_{s'}^{(1D:L)}, \Sigma_{s'}^{(1D:L)})$ are detailed in Appendix B. Moreover, $f(y^D|z^{(1)D}, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})$ belongs to an exponential family as stated in (4). Finally, $f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1})$ has to be numerically approximated. This is here performed by Monte Carlo estimation by simulating $M^{(1)}$ copies of $z^{(1)D}$ as follows

$$f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) = \int_{z^{(1)D}} f(y^D|z^{(1)D}, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) f(z^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) dz^{(1)D}$$

$$\approx \sum_{m=1}^{M^{(1)}} f(y^D|z_m^{(1D)}, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}), f(z_m^{(1D)}|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1}).$$

The second term of (5) can be written as a posterior density:

$$f(s^{(1D:L)} = s'| y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) = \frac{f(s^{(1D:L)} = s'| \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1})}{\sum_{s''} f(s^{(1D:L)} = s''| \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) f(y^D|s^{(1D:L)} = s'', \hat{\Theta}_D, \hat{\Theta}_{L_0+1})}.$$
and we have \((s^{(1:D:L)}|\hat{\Theta}_D, \hat{\Theta}_{L_0+1}) \sim M(\pi_s^{(1:D:L)})\) a multinomial distribution with parameters \(\pi_s^{(1:D:L)}\) which is the probability of a full path through the network starting from the discrete head. The density \(f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{L_0+1})\) is once again approached by Monte Carlo.

### 3.1.3 M step

There are no closed form solutions for the estimators of \((\lambda_0, \Lambda_0)\) that maximize

\[
E_{z^{(1)}D|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}}[\log L(y^D|z^{(1)}D, \Theta_D, \hat{\Theta}_{L_0+1})].
\]

We then use optimisation methods in order to compute those quantities. All methods belong to the Python scipy.optimize package (Virtanen et al., 2020). For binary, count and categorical variables, the optimisation program is unconstrained and the BFGS (Fletcher, 2013) algorithm is used. Concerning ordinal variables, the optimisation program is constrained as the intercept coefficients have to be ordered. The method used is a trust-region algorithm (Conn et al., 2000). All the gradients are computed by automatic differentiation using the autograd package (Maclaurin et al., 2015), which significantly speeds up the optimization process compared to hand-coded gradients.

**Remark 3.1** Other encoding mechanisms than GLLVM should be investigated to feed the latent layers with richer information. One of them could be to consider a mixture of GLLVM models as first layer rather than a single GLLVM. However, one has to assess if it is worth the additional computational burden.

### 3.2 Determining the parameters of the DGMM layers

In this section, we omit the subscript \(h \in \{C, D\}\) on the \(z^h, y^h\) and \(s^h\) variables because the reasoning is the same for both cases. For \(\ell \in [1, L_0]\), we aim to maximize

\[
E_{z^{(\ell)}, z^{(\ell+1)}, s|y, \hat{\Theta}}[\log L(z^{(\ell)}|z^{(\ell+1)}, s, \Theta)].
\]

Here the conditional distribution under which the expectation is taken depends on variables located in 3 different layers.

#### 3.2.1 MC Step

At each layer \(\ell\), one draws \(M^{(\ell)}\) pseudo-observations for each of the previously drawn \(\prod_{j=1}^{\ell-1} M^{(j)}\) pseudo-observations. Hence, in order to draw from \(f(z^{(\ell)}, z^{(\ell+1)}, s|y, \hat{\Theta})\) at layer \(\ell\):

- If \(\ell = 1\), reuse the \(M^{(1)}\) pseudo-observations drawn from \(f(z^{(1)}|s, \hat{\Theta})\),
- otherwise reuse the \(M^{(\ell-1)}\) pseudo-observations from \(f(z^{(\ell-1)}|y, s, \hat{\Theta})\) and the \(M^{(\ell)}\) pseudo-observations from \(f(z^{(\ell)}|z^{(\ell-1)}, s, \hat{\Theta})\) computed for each pseudo-observation of the previous DGMM layer.
• Draw \( M^{(\ell+1)} \) observations from \( f(z^{(\ell+1)}|z^{(\ell)}, s, \hat{\Theta}) \) for each previously sampled \( z^{(\ell)} \).

### 3.2.2 E Step

The conditional expectation distribution has the following decomposition:

\[
f(z^{(\ell)}, z^{(\ell+1)}, s|y, \hat{\Theta}) = f(z^{(\ell)}, s|y, \hat{\Theta}) f(z^{(\ell+1)}|z^{(\ell)}, s, y, \hat{\Theta}) = f(z^{(\ell)}|y, s, \hat{\Theta}) f(z^{(\ell+1)}|z^{(\ell)}, s, \hat{\Theta}).
\]

The first term can be rewritten and approximated as follows:

\[
f(z^{(\ell)}|y, s, \hat{\Theta}) = \int_{z^{(\ell-1)}} f(z^{(\ell)}|z^{(\ell-1)}, s, \hat{\Theta}) f(z^{(\ell-1)}|y, s, \hat{\Theta}) dz^{(\ell-1)} \approx \sum_{m=1}^{M^{(\ell-1)}} f(z^{(\ell)}|z^{(\ell-1)}_m, s, \hat{\Theta}) f(z^{(\ell-1)}_m|y, s, \hat{\Theta}).
\]

This expression is hence calculable in a recurrent manner \( \forall \ell \in [2, L_0] \), starting with \( f(z^{(1)}|y, s', \hat{\Theta}) \) given by (6). The second term of (8) can be expressed as in (7), and the last term is given by the Bayes rule:

\[
f(z^{(\ell+1)}|z^{(\ell)}, s, \hat{\Theta}) = \frac{f(z^{(\ell)}|z^{(\ell+1)}, s, \hat{\Theta}) f(z^{(\ell+1)}|s, \hat{\Theta})}{f(z^{(\ell)}|s, \hat{\Theta})}.
\]

Clearly, the denominator does not depend on \( z^{(\ell+1)} \) and is hence considered as a normalisation constant. Besides, we have that \( f(z^{(\ell)}|z^{(\ell+1)}, s, \hat{\Theta}) = N(\eta_{k\ell} + \Lambda_{k\ell}^{(\ell)} z^{(\ell+1)}, \Psi_{k\ell}^{(\ell)}) \). Finally, by construction of the DGMM, we have

\[
f(z^{(\ell+1)}|s, \hat{\Theta}) = f(z^{(\ell+1)}|s^{(\ell+1: L)}; \hat{\Theta}) = N(\mu_{s^{(\ell+1: L); k\ell+1}}, \Sigma^{(\ell+1)}_{s^{(\ell+1: L); k\ell+1}}).
\]

It follows that (10) is also a Gaussian distribution of parameters \( (\rho_{k\ell+1}^{(\ell+1)}, \xi_{k\ell+1}^{(\ell+1)}) \). All the formulas of the Gaussian parameters previously evoked are given in Appendix B.

### 3.2.3 M step

The estimators of the DGMM layer parameters \( \forall \ell \in [1, L_0] \) are given by:

\[
\begin{align*}
\hat{\eta}_{k\ell}^{(\ell)} &= \sum_{i=1}^{n} \sum_{s^{(k\ell)}_i} f(s^{(k\ell)}_i|y_i, \hat{\Theta}) \left[ E[z^{(\ell)}_i|s^{(k\ell)}_i] = s^{(k\ell)}_i, y_i, \hat{\Theta} \right] - \Lambda_{k\ell}^{(\ell)} E[z^{(\ell+1)}_i|s^{(k\ell)}_i, y_i, \hat{\Theta}] \\
\hat{\Lambda}_{k\ell}^{(\ell)} &= \sum_{i=1}^{n} \sum_{s^{(k\ell)}_i} f(s^{(k\ell)}_i|y_i, \hat{\Theta}) \left[ E[(z^{(\ell)}_i - \eta_{k\ell}^{(\ell)}) z^{(\ell+1)}_i|s^{(k\ell)}_i, y_i, \hat{\Theta}] = \Lambda_{k\ell}^{(\ell)} z^{(\ell+1)}_i|s^{(k\ell)}_i, y_i, \hat{\Theta} \right]^{-1} \\
\hat{\Psi}_{k\ell}^{(\ell)} &= \sum_{i=1}^{n} \sum_{s^{(k\ell)}_i} f(s^{(k\ell)}_i|y_i, \hat{\Theta}) E \left[ \left( z^{(\ell)}_i - (\eta_{k\ell}^{(\ell)} + \Lambda_{k\ell}^{(\ell)} z^{(\ell+1)}_i) \right) \left( z^{(\ell)}_i - (\eta_{k\ell}^{(\ell)} + \Lambda_{k\ell}^{(\ell)} z^{(\ell+1)}_i) \right)^T \right]_{s^{(k\ell)}_i, y_i, \hat{\Theta}},
\end{align*}
\]
with $z^{(k_L)} = (k_1, ..., k_{L-1}, k_L, \tilde{k}_{L+1}, ..., \tilde{k}_L)$, a path going through the network and reaching the component $k_L$. The details of the computation are given in Appendix A.1.

### 3.3 Training of the common tail layers

In this section, we aim to maximise $\forall \ell \in [L_0 + 1, L]$, the following expression:

$$
\mathbb{E}_{z^{(\ell)}, z^{(\ell+1)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}} \left[ \log L(z^{(\ell)} | z^{(\ell+1)}, s^C, s^{(L_0+1)}, \Theta_C, \Theta_D, \Theta_{L_0+1}) \right].
$$

#### 3.3.1 MC Step

The MC step remains the same as for regular DGMM layers except that the conditioning concerns both types of data ($y^C$ and $y^D$) and not only discrete or continuous data as in the heads layers.

#### 3.3.2 E Step

The distribution of the conditional expectation is $f(z^{(\ell)}, z^{(\ell+1)}, s^C, s^{(L_0+1)} | y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})$. As previously this quantity can be rewritten as:

$$
f(z^{(\ell)}, z^{(\ell+1)}, s^C, s^{(L_0+1)} | y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) = f(z^{(\ell)} | s^C, s^{(L_0+1)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) f(z^{(\ell+1)} | s^{(L_0+1)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) \times f(s^C, s^{(L_0+1)} | y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})
$$

$\forall \ell \in [L_0 + 1, L]$, the first term of (12) can be proportionally expressed as:

$$
f(z^{(\ell)} | s^C, s^{(L_0+1)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_{L_0+1}) \propto f(y^C | z^{(\ell)}, s^{(L_0+1)}, \hat{\Theta}_C, \hat{\Theta}_{L_0+1}) \times f(z^{(\ell)} | s^C, s^{(L_0+1)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_{L_0+1})
$$

One can compute $f(y^C | z^{(\ell)}, s^{(L_0+1)}, \hat{\Theta}_C, \hat{\Theta}_{L_0+1})$ using Bayes rule and $f(z^{(\ell)} | s^D, s^{(L_0+1)}, y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})$ is known from (9). Finally the second term of (12) can be computed as in (10) and the third term computations are given in subsection 3.4.

#### 3.3.3 M Step

The estimators of the junction layers keep the same form as the regular DGMM layers except once again that the two types of data and paths exist in the conditional distribution of the expectation.

### 3.4 Determining the paths probabilities

In this section, we determine the paths probabilities by optimizing the parameters of the last term of (3), i.e.

$$
\mathbb{E}_{s^C, s^D, s^{(L_0+1)}} \left[ \log L(s^C, s^D, s^{(L_0+1)}, \Theta_C, \Theta_D, \Theta_{L_0+1}) \right],
$$
with respect to \( \pi_s^h, \forall h \in \{C, D\} \) and \( \pi_s^{(L_0+1)} \).

### 3.4.1 E step

By mutual independence of \( s^C, s^D \) and \( s^{L_0+1} \), estimating the distribution of the expectation boils down to compute three densities: \( f(s^{(\ell)D} = k_\ell|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) \), \( f(s^{(\ell)C} = k_\ell|y^C, \hat{\Theta}_C, \hat{\Theta}_{L_0+1}) \), and \( f(s^{(\ell)} = k_\ell|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) \). The first density can be computed from \( \Omega^{(k_\ell)}D \) as

\[
f(s^{(\ell)D} = k_\ell|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}) = \sum_{\tilde{s} \in \Omega^{(k_\ell)}D} f(s^{(1D:L)} = \tilde{s}|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}),
\]

where \( \Omega^{(k_\ell)}D \) is the set of the full paths going through the component \( k_\ell \) of layer \( \ell \) of head \( D \). The second density can be computed similarly using the fact that \( f(y^C|s^C, s^{(L_0+1)}, \hat{\Theta}_C, \hat{\Theta}_{L_0+1}) \) is Gaussian with parameters \( (\mu^{(1C:L)}, \Sigma^{(1C:L)}) \). Concerning the last density, the computation details are given in Appendix A.3.

### 3.4.2 M step

Using again the conditional independence, we maximise:

\[
\begin{align*}
E_{s^C, s^D, s^{(L_0+1)}|y^C, y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}} & \left[ \log L(s^C, s^D, s^{(L_0+1)}|\Theta_C, \Theta_D, \Theta_{L_0+1}) \right] \\
& = \sum_{\ell=1}^{L_0} E_{s^{(\ell)C}|y^C, \hat{\Theta}_C, \hat{\Theta}_{L_0+1}} [\log L(s^{(\ell)C}|\Theta_C, \Theta_{L_0+1})] \\
& + \sum_{\ell=1}^{L_0} E_{s^{(\ell)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}} [\log L(s^{(\ell)D}|\Theta_D, \Theta_{L_0+1})] \\
& + \sum_{\ell=L_0+1}^{L} E_{s^{(\ell)}|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1}} [\log L(s^{(\ell)}|\Theta_C, \Theta_D, \Theta_{L_0+1})],
\end{align*}
\]

with respect to \( \pi_{k_\ell}^{(\ell)h}, \forall h \in \{C, D\}, \ell \in [1, L_0], k_\ell \in [1, K_\ell] \) and with respect to \( \pi_{k_\ell}^{(\ell)}, \forall \ell \in [L_0, L], k_\ell \in [1, K_\ell] \).

Following Appendix A.2, the probability estimator for each head \( h \) is :

\[
\hat{\pi}_{k_\ell}^{(\ell)h} = \frac{\sum_{i=1}^{n} f(s^{(\ell)h} = k_\ell|y^h, \hat{\Theta}_h, \hat{\Theta}_{L_0+1})}{n}.
\]

For the common tail the estimators are of the form, \( \forall \ell \in [L_0 + 1, L] \):

\[
\hat{\pi}_{k_\ell}^{(\ell)} = \frac{\sum_{i=1}^{n} f(s^{(\ell)} = k_\ell|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{L_0+1})}{n}.
\]

### 4 Identifiability

In this section, we combine both GLLVM and DGMM identifiability constraints proposed in [Cagnone and Viroli (2014)] and [Viroli and McLachlan (2019)] respectively, to make our model identifiable.
4.1 GLLVM identifiability constraints

Both the GLLVM model and the Factor Analysis model assume that the latent variable is centered and of unit variance. We define $z^{(1)h}_{new}$ the rescaled version of $z^{(1)h}$ for each head $h$ satisfying those constraints. This can be obtained by rescaling the parameters of the first layer at each EM step in the following way:

$$
\begin{align*}
&\eta_{k_1'}^{(1)h}_{new} = A^{-1T} \left[ \eta_{k_1'}^{(1)h} - \sum_{\tilde{z}_r} \pi_{\tilde{z}_r} \mu_{\tilde{z}_r} \right] \\
&\Lambda_{k_1'}^{(1)h}_{new} = A^{-1T} \Lambda_{k_1'}^{(1)h} \\
&\Psi_{k_1'}^{(1)h}_{new} = A^{-1T} \Psi_{k_1'}^{(1)h} A^{-1},
\end{align*}
$$

where $A = Var(z^{(1)h})$. The details are given in Appendix A.4. In the same way, the coefficients of $\Lambda^{(0)}$ are rescaled as follows: $\Lambda^{(0)\text{new}} = \Lambda^{(0)} A^{-1T}$.

In GLLVM models, the number of coefficients of $\Lambda^{(0)}$ matrix leads to a too high number of degrees of freedom. Thus, to ensure the identifiability of the model, one has to reduce the number of free coefficients. As in Cagnone and Viroli (2014) the upper triangular coefficients of $\Lambda^{(0)}$ are constrained to be zero. This constraint is explicitly taken into account during the optimisation phase, as the optimisation program is looking for solutions for $\Lambda^{(0)}$ that are upper triangular.

4.2 DGMM identifiability constraints

We assume first that the latent dimension is decreasing though the layers of each head and tail i.e. $p > r^h_1 > ... > r^h_L$. Secondly, we make the assumption that $\Lambda_{k_t}^{(\ell)hT} \Psi_{k_t}^{(\ell)-1h} \Lambda_{k_t}^{(\ell)h}$ is diagonal with elements in decreasing order $\forall \ell \in [1, L]$. Fruehwirth-Schnatter and Lopes (2018) obtained sufficient conditions for MFA identifiability, including the so called Anderson-Rubin (AR) condition, which requires that $r_{\ell-1} \leq \frac{r_{\ell-1}}{2}$. Enforcing this condition would prevent from defining a MDGMM for all datasets that present less than 7 variables of each type which is far too restrictive. Then, we implement a transformation to ensure the diagonality of $\Lambda_{k_t}^{(\ell)hT} \Psi_{k_t}^{(\ell)-1h} \Lambda_{k_t}^{(\ell)h}$ as follows: once all parameters have been estimated by the MCEM algorithm, the following transformation is applied over $\Lambda_{k_t}^{(\ell)h}$:

- Compute $B = \Lambda_{k_t}^{(\ell)hT} \Psi_{k_t}^{(\ell)-1h} \Lambda_{k_t}^{(\ell)h}$.
- Decompose $B$ according to the eigendecomposition $B = PDP^{-1}$, with $D$ the matrix of the eigenvalues and $P$ the matrix of eigenvectors.
- Define $\Lambda_{k_t}^{(\ell)h\text{new}} = \Lambda_{k_t}^{(\ell)h} P$. 

14
5 Practical considerations

5.1 Model and number of clusters selection

The selection of the best MDGMM architecture is performed using the pruning methodology which is widely used in the field of supervised neural networks [Blalock et al., 2020] but also for tree-based methods [Patil et al., 2010]. The idea is to determine the simplest architecture that could describe the data. In order to do so, one starts with a complex architecture, and delete the coefficients that do not carry enough information. Deleting those coefficients at some point during the training process is known as ”pre-pruning” and performing those deletions after full convergence is known as ”post-pruning”. In our case, we use a pre-pruning strategy to estimate the best number of components \( k_\ell \), the best number of factors \( r_\ell \) and the best number of layers for the heads and tails. The reason not to use post-pruning instead of pre-pruning is that very complex architectures training tend to show long running times and a higher propensity not to converge to good maxima.

Alternative approaches to pruning exist, based for instance on information criteria computations such as AIC [Akaike, 1998] or BIC [Schwarz et al., 1978]. However, these methods need to compute all the possible specifications of the model to evaluate the information criteria on each. In contrast, our approach needs only one model run to determine the best architecture which is far more computationally efficient.

In the following, we give a summary of our pruning strategy (extensive details are provided in Appendix C). Our pruning strategy determines the best number of components on each layer \( k_\ell \) by deleting the components associated with very low probabilities \( \pi_{k_\ell}^{(\ell)h} \) as they are the least likely to explain the data.

The choice of the latent dimensions of each layer \( r_\ell \) is performed by looking at the dimensions that carry the most important pieces of information about the previous layer. The goal is to ensure the circulation of relevant information through the layers without transmitting noise information. This selection is conducted differently for the GLLVM layer compared to the regular DGMM layers. For the GLLVM layer, we perform logistic regressions of \( y^C \) over \( z^{(1)}C \) and delete the dimensions that were associated with non-significant coefficients in a vast majority of paths. Concerning the regular DGMM layers, information carried by the current layer given the previous layer has been modeled using a Principal Component Analysis [Hotelling, 1933]. We compute the contribution of each original dimension to the first principal component analysis and keep only the dimensions that present a high correlation with this first principal component. Doing so we eliminate information of secondary importance carried out through the layers.

Finally, the choice of total number of layers is guided by the selected \( r_\ell \). For instance, if a dimension of 2 is selected for a head layer (or a dimension of 1 for a tail layer), then according to the identifiability constraint \( p^h > r_1^h > ... > r_\ell^h > ... > r_L \), the following head (or tails layers) are deleted.

Given that this procedure also selects the number of components on the tail layers, it can also be
used to automatically find the optimal number of clusters in the data. The user specifies a high number of components on the clustering layer and let the automatic selection operate. The optimal number of clusters is then the number of components remaining on the clustering layer at the end of the run. This feature of the algorithm is referred to as the "autochus mode" of the MDGMM in the following and in the code.

Alternatively, in case of doubt about the number of clusters in the data, the MDGMM could be used in "multi-clustering" mode. For example, if the number of clusters in the data is assumed to be two or three, one can define a MDGMM with three components on the first tail layer and two on the second tail layer. The first layer will output a three groups clustering and the second layer a two groups clustering. The two partitions obtained can then be compared to chose the best one. This can for instance be done with the silhouette coefficient (Rousseeuw, 1987) as implemented in our code. In the "multi-clustering" mode, the same described model selection occurs. The only exception is that the number of components of the tail layers remain frozen (as they correspond to one of the tested number of clusters in the data).

For all clustering modes of the MDGMM, the architecture selection procedure is performed at the end of some iterations chosen by the user before launching the algorithm. Note that once the optimal specification has been determined, it is better to refit the model using the determined specification rather than keeping the former output. Indeed, changing the architecture "on the fly" seems to disturb the quality of the final clustering. In our simulations, the embedding layer was specified as a GLLVM layer with $K_{LD} = 1$ and the model selection was performed on the dimensions of the components $(r^h)_{h \in \{C,D,L_0+1\}}$ and on the number of components of the tail layer $K_{L_0+1}$ after the first and second iterations of the algorithm.

### 5.2 Initialisation procedure

EM-based algorithms are known to be very sensitive to their initialisation values as shown for instance by Biernacki et al. (2003) for Gaussian Mixture models. In our case, using purely random initialization as in Cagnone and Viroli (2014) made the model diverge most of the time when the latent space was of high dimension. It can be explained by the fact that the clustering is performed in a projected continuous space of which one has no prior knowledge about. Initialising at random the latent variables $(\eta_{k_{LD}}, \Lambda^{(0)}_{k_{LD}}, \Psi^{(0)}_{k_{LD}}, \Lambda^{(0)}_{k_{LD}}, \Lambda^{(0)}_{k_{LD}})$ and the exponential family links parameters $(\lambda^{(0)}, \Lambda^{(0)})$ seems not to be a good practice. This problem gets even worse as the number DGMM layers grow. To stabilize our algorithm we propose the NESP approach which combines MCA, GMM, FA and PLS algorithm.

- For discrete head initialisation, the idea used here is to perform a Multiple Correspondence Analysis (MCA) (Nenadic and Greenacre, 2005) to determine a continuous low dimensional representation of the discrete data and use it as a first approximation of the latent variables $z^{(1)}_{D}$. The MCA considers all variables as categorical, thus the more the dataset actually contains this type of variables the better
the initialisation should be in theory. Once this is done, a Gaussian Mixture Model is fitted in order to determine groups in the continuous space. For each group a Factor Analysis Model (FA) is fitted to determine the parameters of the model \((\eta_{k\ell}^{(t)}, \Lambda_{k\ell}^{(t)}, \Psi_{k\ell}^{(t)}, \pi_{k\ell}^{(t)})\) and the latent variable of the following layer \(z^{(\ell+1)}\). Concerning the GLLVM parameters, logistic regressions of \(y_{jD}^{(l)}\) over \(z^{(1)}\) are fitted for each original variable of the discrete head: an ordered logistic regression for ordinal variables, an unordered logistic regression for categorical variables.

- For the continuous head and the common tail, the same described GMM coupled with FA procedure can be applied to determine the coefficients of the layer. The difficulty concerns the initialisation of the first tail layer with latent variable \(z^{(L_0+1)}\). Indeed, \(z^{(L_0+1)}\) has to be the same for both discrete and continuous last layers. As Factor Models are unsupervised models one cannot enforce such a constraint on the latent variable generated from each head. To overcome this difficulty, \(z^{(L_0+1)}\) has been determined by applying a PCA \(\text{[Pearson, 1901]}\) over the stacked variables \((z^{(L_0)}_C, z^{(L_0)}_D)\). Then the DGMM coefficients \((\eta_{k_{L_0}}^{(L_0)}h, \Lambda_{k_{L_0}}^{(L_0)}h, \Psi_{k_{L_0}}^{(L_0)}h)\) of each head have been separately determined using Partial Least Square (PLS) \(\text{[Wold et al., 2001]}\) of each head latent variables over \(z^{(L_0+1)}\).

### 5.3 Monte Carlo scheme

The number of Monte Carlo copies \(M^{(\ell)h}\) to draw at each layer has to be chosen before running the MCEM. \(\text{[Wei and Tanner, 1990]}\) advise to let \(M\) grow through the iterations starting with a very low \(M\). Doing so, one does not get stuck into very local optima at the beginning of the algorithm and ends up in a precisely estimated expectation state. The growth scheme of \(M^{(\ell)}_t\) through the iterations \(t\) implemented here is:

\[
M^{(\ell)}_t = \left\lfloor \frac{40 \times \log(n) \times t \times \sqrt{r^{(\ell)}}}{40} \right\rfloor.
\]

\(M^{(\ell)}_t\) grows linearly with the number of iterations \(t\) to follow \(\text{[Wei and Tanner, 1990]}\) advice. In order to explore the latent space at each layer, \(M^{(\ell)}\) also grows with the dimension of the layer. The square root rate just ensures that the running time remains affordable, whereas if there were no additional computational costs we would certainly have let \(M^{(\ell)}\) grow much more with \(r^{(\ell)}\). Finally, we make the hypothesis that the more observations in the dataset the stronger the signal is and hence the fewer draws of latent variables are needed to train the model.

**Remark 5.1** In this Monte Carlo version contrary to the regular EM algorithm, the likelihood does not increase necessary through the iterations. In classical EM-based models, the training is often stopped once the likelihood increases by less than a given threshold between two iterations. The stopping process had then to be adapted to account for temporary losses in likelihood. Hence, we have defined a patience parameter which is the number of iterations without log-likelihood increases to wait before stopping the algorithm.
Typically, we set this parameter to 1 iteration in the simulations.

6 Real Applications

The MDGMM model has been constructed on several building blocks introduced in this work that can be regarded as proper models. First our new initialisation procedure has improved the stability of the GLLVM with mixture latent variable (denoted GLMLVM thereafter) proposed by Cagnone and Viroli (2014). This has then been extended by adding several hidden layers at the model to obtain what we will call a Discrete data DGMM (DDGMM). Finally, a second head has been added in order for the MDGMM to support continuous data.

An alternative way to deal with both discrete and continuous data would have been to feed the GLLVM layer with the continuous variables by defining a link function between $y^C$ and $z^{(1)}$. Using the GLLVM framework would have imposed to assume that the continuous variables were mutually independent with respect to the latent variables. This is a stronger assumption than to assume that they are only independent of discrete data conditionally to the tail latent layers as in the MDGMM. The MDGMM is also more flexible than this alternative as it does not impose a strict parametric link between $y^C$ and $z^C$.

However, relaxing these hypotheses comes at the price of an additional computational complexity that have to be evaluated in terms of performance. Thus we will evaluate this one-head version of the MDGMM provided with a Gaussian link function and call it M1DGMM in the following.

As some of the sub-models can deal with discrete data only (GLLVMs, DDGMM) and other with mixed data (M1DGMM, MDGMM) we consider both types of datasets in this section.

This section will first present the continuous low dimensional representations of the data generated by the DDGMM and the MDGMM. Then the performance of the submodels will be properly evaluated by comparing them to state-of-the-art mixed clustering algorithms.

6.1 Data used

All the datasets used in the following come from the UCI repository. For the discrete data specification, we present results over 3 datasets: the Breast cancer, the Mushrooms and the Tic Tac Toe datasets.

For mixed datasets, we have used the Australian credit, the Heart (Statlog) and the Pima Indians diabetes Datasets.
6.1.1 Discrete data presentation

- The Breast cancer dataset is a dataset of 286 observations and 9 discrete variables. Most of the variables are ordinal.

- The Tic Tac Toe dataset is composed of 9 variables corresponding to each cell of a $3 \times 3$ grid of tic-tac-toe. The dataset presents the grids content at the end of 958 games. Each cell can be filled with one of the player symbol (x or o), or left blanked (b) if the play has ended before all cells were filled in. Hence all the variables are categorical in contrast with the Breast cancer data.
  The goal is here to retrieve which game led to victory of player 1 or of player 2 (no even games are considered here).

- Finally, the Mushrooms dataset is a two-class dataset with 22 attributes and 5644 observations once the missing data have been removed. The majority of the variables are categorical ones. Once the categorical data have been one-hot encoded, the final dataset has more than 70 variables and is hence a highly dimensional dataset.

6.1.2 Mixed data presentation

- The Heart (Stalog) dataset is composed of 270 observations, five continuous variables, three categorical variables, three binary variables and two ordinal variables.

- The Pima Indians Diabetes dataset presents several physiological variables (e.g. the blood pressure, the insulin rate, the age) of 768 Indian individuals. 267 individuals suffer from diabetes and the goal of classification tasks over this dataset is to distinguish the sound people from the sick ones. This dataset counts two discrete variables considered here respectively as binomial and ordinal and seven continuous variables.

- Finally, the Australian credit (Stalog) dataset is a binary classification dataset concerning credit cards. It is composed of 690 observations, 8 discrete categorical variables and 6 continuous variables. It is a small dataset with a high dimension.

  In the sequel, all of the continuous variables have been centered and reduced to ensure the numeric stability of the algorithms.

6.2 Data vizualisation

According to their multi-layer structures, the DDGMM and the MDGMM perform several dimension reductions of information while the signal goes through their layers. As such, they provide low dimensional
continuous representations of complex data than can be discrete, mixed or potentially highly dimensional. These representations are useful to understand how observations are clustered through the training process. They could also be reused to train other algorithms in the same spirit as for supervised Neural Network (Jogin et al., 2018).

Figure 2 shows the evolution of the latent representation during the training of the clustering layer of a DDGMM for the tic tac toe dataset. This clustering layer has a dimension of $r_\ell = 2$ and tries to distinguish $k_\ell = 2$ groups in the data. At at the beginning of the training at $t_1$, it is rather difficult to differentiate two clusters in the data. However, through the next iterations, one can clearly distinguish that two sets of points are pushed away from each other by the model. Moreover in $t_3$ the frontier between the two clusters can be drawn as a straight line in a two dimensional space. In $t_4$ at the end of the training, the model seems to have found a simpler frontier to separate the groups as only a vertical line, i.e. a separation in a one dimensional space is needed. This highlight the information sorting process occurring through the layers in order to keep only the simplest and the more discriminating parts of the signal.

The next two figures illustrate graphical properties of the MDGMM. Figure 3 presents two continuous representations of the Pima Diabetes data. These are obtained during the training of a MDGMM with two hidden tail layers of respectively $r_{L_0+1} = 3$ and $r_{L_0+1} = 2$ during the same iteration. Two clusters are looked for in each case ($K_{L_0+1} = K_{L_0+2} = 2$) and are associated with green and red colors on the figure.
On both layers the clusters are quite well separated. The signal carried seems coherent between the two layers with a very similar structure. For the same computational cost, i.e. one run of the model, several latent representations of the data in different dimensions can therefore be obtained.

Finally, the graphical representations produced by the MDGMM are useful tools to identify the right number of clusters in the data. Three MDGMMs have been run by setting $r_{L_0+1} = 2$ and with respectively $K_{L_0+1} = 2$, $K_{L_0+1} = 3$ and $K_{L_0+1} = 4$. The associated latent variables are presented in Figure 4 with a different color for each identified cluster.

The representations with three and four clusters present points that are intertwined, with no clear distinctions between clusters. On the contrary, when the number of clusters searched in the data is two this separation appears distinctly. Hence, this representation advocates for a two groups distinction in the data as it is suggested by the supervised labels of the dataset (absence or presence of heart disease). The four clusters representation also shows that the three points associated with the red cluster might be outliers potentially important to study.
As evoked in subsection 5.1, this visual diagnostic can be completed by using the "autoclus mode" of the MDGMM where the model automatically determine the best number of clusters in the data.

### 6.3 Performance comparison

#### 6.3.1 Models compared

As evoked in introduction, Ahmad and Khan (2019) have proposed a typology of clustering algorithms which is composed of four main families. In order to benchmark our models performance, we consider algorithms coming from each of the four families, namely: k-modes, k-Prototypes, Hierarchical Clustering and Self-Organising Maps (SOM). Ahmad and Khan (2019) actually defined a last family containing the models that did not fit into one of the four families mentioned. DBSCAN is one of them (Ester et al., 1996) and is used to represent this last kind of mixed data clustering algorithm.

For each dataset, we have set the number of unsupervised clusters to the "ground truth" classification number. In order to present a fair report, several specifications of the benchmark models have been run. For each specification, the models have been launched 30 times. The results reported correspond to the result achieved by the best specification of each benchmark models with respect to each metric. The set of specifications evaluated for each benchmark model is given in Appendix E. Concerning our models, the architectures were automatically selected and then fit 30 times on each dataset.

Here we use one unsupervised metric and two supervised metrics to assess the clustering quality: the silhouette coefficient, the micro precision and the macro precision. According to Ahmad and Khan (2019), most of the articles dealing with mixed data clustering use in priority the (micro) precision also called accuracy. However, the labels represent a way to partition the data in order to respond to a precise question, for instance "is this tumor malignant or benign ?" in the Breast cancer dataset. Yet, this is not the only question that one may ask from these data. For instance, one can wonder if the associated tumour is of big or small size. Hence, other valid partitions of the data exist and are penalized by these supervised metrics. That is why using unsupervised metrics such as the silhouette coefficient seemed necessary to us.

The silhouette coefficient measures how close on average a point is from the points of the same group with respect to the points of the other groups. The Euclidian distance cannot be used here due to the mixed feature space and hence the Gower distance (Gower, 1971) is used instead. The micro precision corresponds to the overall accuracy i.e. the proportion of correctly classified instances. The macro precision computes the proportion of correctly classified instances per class and then returns a non-weighted mean of those proportions. These two quantities tend to differ when the data are not balanced. The formal expressions of the metrics are given in Appendix D.
### 6.3.2 Results on discrete data

The following tables presents the best average results obtained by the algorithms and the associated standard error over the 30 runs in parenthesis. The best algorithm for a given dataset and metric is associated with a green cell and the worst with a red cell. An empty set symbol means that the metric was not defined for this algorithm on that dataset. For the special case of the k-prototypes algorithm, the empty set symbol means that the dataset contained only one type of discrete data which is a situation that the algorithm is not designed for.

![Table 1: Average results and standard errors over 30 runs of the best specification for each model over three discrete datasets](image)

From the standard errors computed, one can see that the new initialisation of the GLMLVM stabilizes the results, reducing the standard errors by a factor at least two. As the standard errors take into account only the complete runs of the algorithms, i.e. not the ones which have diverged, the standard errors of the randomly initialized algorithm are in fact much higher. The share of diverging launches was indeed as high as 40% and even more for the Mushrooms dataset. However, the new initialisation does not seem to enable the algorithm to explore better solutions as the average scores for the two algorithms are comparable. The results between the DDGMM and the new initialisation procedure are comparable. However, to find the best specification of the initialisation algorithm one has to explore all the possible specifications and chose the best one. Then to achieve such a performance one has to run the initialisation algorithm for each specification whereas only once for the DDGMM.

The DDGMNs results are also similar to the GLMLVM except concerning the Mushrooms data where the results become quite unstable. For this dataset, the number of Monte Carlo points was maintained constant through the iterations in order to keep an acceptable running time (less than three minutes per run on average). With lower Monte Carlo points the parameter space was less well explored. As a result, for some algorithm runs the DDGMM likelihood stops increasing after only one or two iterations which led to a wide variety of performance.

Compared to the benchmark algorithms, the DDGMM and the other sub-models remain in the average for...
nearly all datasets and all metrics. On the contrary, some benchmark algorithms seem very fit for some datasets but very poorly adapted for others. This is the case for instance of DBSCAN which performs best on the Breast cancer dataset but worst on the Mushrooms and the Tic Tac Toe datasets (the algorithm could find only one group in the data which explains that the silhouette score is not defined). Our suite of model seems in this regard applicable to a wide range of datasets.

Finally, among all results, the Hierarchical clustering is the algorithm that performs the best on a majority of metrics and datasets.

### 6.3.3 Results on mixed data

| Datasets          | Heart | Pima | Australian Credit |
|-------------------|-------|------|-------------------|
| Algorithms / Metrics | Silhouette | Micro | Macro | Silhouette | Micro | Macro | Silhouette | Micro | Macro |
| NESP               | 0.158 (0.040) | 0.735 (0.003) | 0.738 (0.059) | 0.189 (0.013) | 0.666 (0.056) | 0.651 (0.051) | 0.132 (0.039) | 0.718 (0.078) | 0.716 (0.088) |
| M1DGMM             | 0.142 (0.060) | 0.627 (0.095) | 0.706 (0.106) | 0.221 (0.058) | 0.679 (0.027) | 0.650 (0.068) | 0.069 (0.045) | 0.563 (0.046) | 0.570 (0.108) |
| M2DGMM             | 0.073 (0.035) | 0.592 (0.059) | 0.610 (0.071) | 0.117 (0.076) | 0.610 (0.053) | 0.560 (0.051) | 0.083 (0.032) | 0.566 (0.049) | 0.536 (0.110) |
| k-Modes            | 0.247 (0.000) | 0.811 (0.000) | 0.813 (0.000) | 0.643 (0.031) | 0.541 (0.000) | 0.482 (0.000) | 0.219 (0.008) | 0.783 (0.009) | 0.783 (0.007) |
| k-Prototypes       | 0.044 (0.000) | 0.592 (0.000) | 0.585 (0.000) | 0 (0) | 0 (0) | 0 (0) | 0.163 (0.000) | 0.562 (0.000) | 0.780 (0.000) |
| Hierarchical       | 0.263 (0.000) | 0.811 (0.000) | 0.809 (0.000) | 0.391 (0.000) | 0.656 (0.000) | 0.826 (0.000) | 0.354 (0.000) | 0.849 (0.000) | 0.847 (0.000) |
| SOM                | 0.043 (0.000) | 0.619 (0.000) | 0.614 (0.000) | 0.219 (0.000) | 0.663 (0.000) | 0.613 (0.000) | 0.050 (0.000) | 0.699 (0.000) | 0.730 (0.000) |
| DBSCAN             | 0.130 (0.000) | 0.558 (0.000) | 0.724 (0.000) | 0.391 (0.000) | 0.652 (0.000) | 0.826 (0.000) | 0 (0) | 0.575 (0.000) | 0.278 (0.000) |

Table 2: Average results and standard errors over 30 runs of the best specification for each model over three mixed datasets

The NESP seems here again to be a good starting point for both algorithms and certainly explain the fact that the M1DGMM reaches the best micro score on the Pima dataset.

The M1DGMM achieves better results on the Heart and Pima datasets, whereas the M2DGMM performs a little better on the Australian credit data for two metrics out of three. They also tend to present opposite patterns in terms of standard errors: when the M2DGMM results are stable the M1DGMM results tend to be more volatile and conversely. Hence, the two models seems complementary and using each one in turn could enable to conduct clustering on a very large diversity of datasets.

One can observe the same general pattern as for the discrete data results: our models give satisfactory performance on all datasets on average, whereas other models such as SOM, DBSCAN or k-modes are adapted only to some datasets. As on discrete data, the hierarchical clustering method seems to provide the best results on a large set of metrics and datasets.

### 7 Conclusion

This work aimed to bring together two recent methods, namely the extended GLLVM and the DGMM, to propose a new model able to deal with mixed datasets. In order to do so, several sub-models such as a new initialisation procedure NESP for GLLVM-based models, the DGMM for discrete data (DDGMM)
or the one-head MDGMM (M1DGMM) have also been introduced and could be used on their own. This suite of models take care of the usual clustering issues concerning architecture selection and the choice of the number of clusters in the data in an automated way.

From the experiments led on real data, the MDGMM performances are in line with the other state-of-the-art models compared. It can be regarded as a baseline model over a general class of data. Its use of nested Mixtures of Factor Analysis enables it to capture a very wide range of distributions and patterns. Despite of its complexity, the MDGMM remains interpretable. From a practical viewpoint, the structure of the latent space can be observed through the model training with the help of the graphical utilities presented above. Thus, they allow the user to perform visual diagnostics of the clustering process. From a theoretical standpoint, the parameters of the model remain interpretable as the link between parameters and clustering results is proper thanks to the identifiability of the model. The set of identifiability constraints presented here could seem quite restrictive. However, it forces the model to stay in a quite well delimited parameter space and avoid for instance a too significant explosion of the norm of the parameters values. The implementation of these constraints could however be improved by considering a Bayesian re-writting of our model on Variational principles. Indeed, it should make identification requirements easier to meet, as one can keep only the posterior draws that meet the identifiability requirements. Nikul et al. (2019) have rewritten the GLLVM model in a variational fashion and exhibit high running time and accuracy gains. Following their path, one could adapt the MDGMM to the variational framework. Finally considering the training process, the choice of an EM-based algorithm was motivated by its extensive use in the Gaussian Mixture Model literature. The EM-related algorithms are however very sensitive to the initialisation, which was in our case particularly tricky given the size of the parameter space. Combining Multiple Correspondance Analysis (MCA) with Gaussian Mixture Models (GMM), Factor Analysis (FA) and Partial Least Squares (PLS) into NESP has however enabled us to significantly stabilize the training process. Yet, new initialisation and training processes could be designed to help the model to better rationalize latent structures in the data within its very highly dimensional space and to make the model really competitive.

Acknowledgments

Thanks to the LIA LYSM (agreement between AMU, CNRS, ECM and INdAM).
Ahmad, A. and S. S. Khan (2019). Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access* 7, 31883–31902.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pp. 199–213. Springer.

Baydin, A. G., B. A. Pearlmutter, A. A. Radul, and J. M. Siskind (2017). Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research* 18(1), 5595–5637.

Bettencourt, J., M. J. Johnson, and D. Duvenaud (2019). Taylor-mode automatic differentiation for higher-order derivatives in jax. *Mathematics*.

Biernacki, C., G. Celeux, and G. Govaert (2003). Choosing starting values for the em algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computational Statistics & Data Analysis* 41(3-4), 561–575.

Blalock, D., J. J. G. Ortiz, J. Frankle, and J. Guttag (2020). What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*.

Cagnone, S. and C. Viroli (2014). A factor mixture model for analyzing heterogeneity and cognitive structure of dementia. *AStA Advances in Statistical Analysis* 98(1), 1–20.

Chiu, T., D. Fang, J. Chen, Y. Wang, and C. Jeris (2001). A robust and scalable clustering algorithm for mixed type attributes in large database environment. In *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 263–268.

Conn, A. R., N. I. Gould, and P. L. Toint (2000). *Trust region methods*, Volume 1. Siam.

Ester, M., H.-P. Kriegel, J. Sander, X. Xu, et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, Volume 96, pp. 226–231.

Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons.

Fraley, C. and A. E. Raftery (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association* 97(458), 611–631.

Fruehwirth-Schnatter, S. and H. F. Lopes (2018). Sparse bayesian factor analysis when the number of factors is unknown. *arXiv preprint arXiv:1804.04231*.
Ghahramani, Z., G. E. Hinton, et al. (1996). The em algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto.

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, 857–871.

Harman, H. H. (1976). *Modern factor analysis*. University of Chicago press.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology* 24(6), 417.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery* 2(3), 283–304.

Jogin, M., M. Madhulika, G. Divya, R. Meghana, S. Apoorva, et al. (2018). Feature extraction using convolution neural networks (cnn) and deep learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pp. 2319–2323. IEEE.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE* 78(9), 1464–1480.

Maclaurin, D., D. Duvenaud, and R. P. Adams (2015). Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop*, Volume 238, pp. 5.

Melnykov, V., R. Maitra, et al. (2010). Finite mixture models and model-based clustering. *Statistics Surveys* 4, 80–116.

Moustaki, I. (2003). A general class of latent variable models for ordinal manifest variables with covariate effects on the manifest and latent variables. *British Journal of Mathematical and Statistical Psychology* 56(2), 337–357.

Moustaki, I. and M. Knott (2000). Generalized latent trait models. *Psychometrika* 65(3), 391–411.

Nenadic, O. and M. Greenacre (2005). Computation of multiple correspondence analysis, with code in r.

Niku, J., W. Brooks, R. Herliansyah, F. K. Hui, S. Taskinen, and D. I. Warton (2019). Efficient estimation of generalized linear latent variable models. *PloS one* 14(5).

Patil, D. D., V. Wadhai, and J. Gokhale (2010). Evaluation of decision tree pruning algorithms for complexity and classification accuracy. *International Journal of Computer Applications* 11(2), 23–30.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2(11), 559–572.
A Calculus details

A.1 Estimators of the DGMM layers

We now turn on to the log-likelihood expression and give the estimators of the $\ell$-th DGMM layer parameters $\forall \ell \in [1, L_h], \forall h \in \{C, D\}$. In this section the $h$ superscripts are omitted for simplicity of notation.

$$
\log L(z_i^{(\ell)}|z_i^{(\ell+1)}, s_i, \Theta) = 
\frac{1}{2} \left[ \log(2\pi) + \text{log det}(\Psi_{k\ell}^{(\ell)}) + \left( z_i^{(\ell)} - (\eta_{k\ell}^{(\ell)} + \Lambda_{k\ell}^{(\ell)} z_i^{(\ell+1)}) \right)^T \Psi_{k\ell}^{(\ell)-1} \left( z_i^{(\ell)} - (\eta_{k\ell}^{(\ell)} + \Lambda_{k\ell}^{(\ell)} z_i^{(\ell+1)}) \right) \right].
$$

The derivatives of this quantity with respect to $\eta_{k\ell}^{(\ell)}, \Lambda_{k\ell}^{(\ell)}, \Psi_{k\ell}^{(\ell)}$ are given by
\[
\begin{aligned}
&\left\{ \begin{array}{c}
\frac{\partial \log L(z^{(\ell)}_{i}|z^{(\ell+1)}_{i},s_{i},\Theta)}{\partial \eta^{(\ell)}_{ki}} = \Psi^{(\ell)-1}_{k} \left( z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i}) \right) \\
\frac{\partial \log L(z^{(\ell)}_{i}|z^{(\ell+1)}_{i},s_{i},\Theta)}{\partial \Lambda^{(\ell)}_{ki}} = \Psi^{(\ell)-1}_{k} \left( z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i}) \right) z^{(\ell+1)T}_{i} \\
\frac{\partial \log L(z^{(\ell)}_{i}|z^{(\ell+1)}_{i},s_{i},\Theta)}{\partial \Psi^{(\ell)}_{ki}} = -\frac{1}{2} \Psi^{(\ell)-1}_{k} \left[ I_{r_{i}} - (z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i})) \right] (z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i}))^{T} \Psi^{(\ell)-1}_{k} \right].
\end{array} \right.
\end{aligned}
\]

Taking the expectation of the derivative with respect to \(\eta^{(\ell)}_{ki}\) and equalizing it to zero, it follows that:

\[
E_{z^{(\ell)}_{i},z^{(\ell+1)}_{i},s_{i}|y_{i},\hat{\Theta}} \left[ \frac{\partial \log L(z^{(\ell)}_{i}|z^{(\ell+1)}_{i},s_{i},\Theta)}{\partial \eta^{(\ell)}_{ki}} \right] = 0
\]

\[
\iff \Psi^{(\ell)-1}_{k} \sum_{i=1}^{n} E_{z^{(\ell)}_{i},z^{(\ell+1)}_{i},s_{i}|y_{i},\hat{\Theta}} \left[ z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i}) \right] = 0
\]

\[
\iff \sum_{i=1}^{n} E_{z^{(\ell)}_{i},z^{(\ell+1)}_{i},s_{i}|y_{i},\hat{\Theta}} \left[ z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i}) \right] = 0, \text{ since } \Psi^{(\ell)}_{ki} \text{ is positive semi-definite.}
\]

\[
\iff \sum_{i=1}^{n} \sum_{s_{i}} f(s_{i}^{(k_{c})}) = \hat{s}_{i}^{(k_{c})}|y_{i},\hat{\Theta} \left[ E_{z^{(\ell)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}} \left[ z^{(\ell)}_{i} \right] - \eta^{(\ell)}_{ki} - \Lambda^{(\ell)}_{ki} E_{z^{(\ell+1)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}} \left[ z^{(\ell+1)}_{i} \right] \right] = 0.
\]

Therefore, the estimator of \(\eta^{(\ell)}_{ki}\) is given by

\[
\hat{\eta}^{(\ell)}_{ki} = \frac{\sum_{i=1}^{n} \sum_{s_{i}} f(s_{i}^{(k_{c})}) = \hat{s}_{i}^{(k_{c})}|y_{i},\hat{\Theta} \left[ E_{z^{(\ell)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}} \left[ z^{(\ell)}_{i} \right] - \eta^{(\ell)}_{ki} - \Lambda^{(\ell)}_{ki} E_{z^{(\ell+1)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}} \left[ z^{(\ell+1)}_{i} \right] \right] }{\sum_{i=1}^{n} \sum_{s_{i}} f(s_{i}^{(k_{c})}) = \hat{s}_{i}^{(k_{c})}|y_{i},\hat{\Theta}}
\]

with

\[
E_{z^{(\ell+1)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}} = \int_{z^{(\ell+1)}_{i}} f(z^{(\ell+1)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}) \int_{z^{(\ell)}_{i}} f(z^{(\ell)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}) z^{(\ell+1)}_{i} d z^{(\ell+1)}_{i} d z^{(\ell)}_{i}
\]

\[
\approx \sum_{m_{\ell+1}=1}^{M^{(\ell+1)}} \sum_{m_{\ell}=1}^{M^{(\ell)}} f(z^{(\ell)}_{i,m_{\ell+1}}|s^{(k_{c})}_{i},y_{i},\hat{\Theta}) \sum_{m_{\ell+1}=1}^{M^{(\ell+1)}} z^{(\ell+1)}_{i,m_{\ell+1}}
\]

where \(z^{(\ell+1)}_{i,m_{\ell+1}}\) has been drawn from \(f(z^{(\ell+1)}_{i}|z^{(\ell)}_{i},s_{i},m_{\ell+1})\). Using the same reasoning for \(\Lambda^{(\ell)}_{ki}\) we obtain

\[
E_{z^{(\ell)}_{i},z^{(\ell+1)}_{i},s_{i}|y_{i},\hat{\Theta}} \left[ \frac{\partial \log L(z^{(\ell)}_{i}|z^{(\ell+1)}_{i},s_{i},\Theta)}{\partial \Lambda^{(\ell)}_{ki}} \right] = 0
\]

\[
\iff \Psi^{(\ell)-1}_{k} \sum_{i=1}^{n} E_{z^{(\ell)}_{i},z^{(\ell+1)}_{i},s_{i}|y_{i},\hat{\Theta}} \left[ (z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i})) z^{(\ell+1)T}_{i} \right] = 0
\]

\[
\iff \sum_{i=1}^{n} \sum_{s_{i}} f(s_{i}^{(k_{c})}) = \hat{s}_{i}^{(k_{c})}|y_{i},\hat{\Theta} \left[ E_{z^{(\ell)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}} \left[ (z^{(\ell)}_{i} - (\eta^{(\ell)}_{ki} + \Lambda^{(\ell)}_{ki} z^{(\ell+1)}_{i})) z^{(\ell+1)T}_{i} \right] - \Lambda^{(\ell)}_{ki} E_{z^{(\ell+1)}_{i}|s_{i}^{(k_{c})},y_{i},\hat{\Theta}} \left[ z^{(\ell+1)}_{i} z^{(\ell+1)T}_{i} \right] \right] = 0.
\]
Hence the estimator of $\Lambda_{k_\ell}$ is given by

$$
\hat{\Lambda}_{k_\ell} = \frac{\sum_{i=1}^{n} \sum_{z_i^{(k_\ell)}} f(s_i^{(k_\ell)} = \tilde{s}_i^{(k_\ell)} | y_i, \hat{\Theta}) \left[ E[(z_i^{(\ell)} - \hat{\eta}_{k_\ell}) z_i^{(\ell+1)T} | \tilde{s}_i^{(k_\ell)}, y_i, \hat{\Theta}] \right] E[z_i^{(\ell+1)} z_i^{(\ell+1)T} | \tilde{s}_i^{(k_\ell)}, y_i, \hat{\Theta}]^{-1},
$$

with

$$
E[(z_i^{(\ell)} - \hat{\eta}_{k_\ell}) z_i^{(\ell+1)T} | \tilde{s}_i^{(k_\ell)}, y_i, \hat{\Theta}] = \int_{z_i^{(\ell)}} f(z_i^{(\ell)} | \tilde{s}_i^{(k_\ell)}, y_i, \hat{\Theta}) \int_{z_i^{(\ell+1)}} f(z_i^{(\ell+1)} | z_i^{(\ell)}, \tilde{s}_i^{(k_\ell)}, \hat{\Theta})[(z_i^{(\ell)} - \hat{\eta}_{k_\ell}) z_i^{(\ell+1)T} d z_i^{(\ell)} d z_i^{(\ell+1)}] 
\approx \sum_{m_{i_1} \leq M^{(\ell+1)}} \sum_{m_{i_2} \leq M^{(\ell+1)}} f(z_i^{(\ell)} | \tilde{s}_i^{(k_\ell)}, y_i, \hat{\Theta}) \sum_{m_{i_3} \leq M^{(\ell+1)}} \sum_{m_{i_4} \leq M^{(\ell+1)}} [z_i^{(\ell)} - \hat{\eta}_{k_\ell}] z_i^{(\ell+1)T},
$$

where $z_{i,m_{i_1}}$ has been drawn from $f(z_i^{(\ell)} | s, \hat{\Theta})$ and $z_{i,m_{i_2}}$ from $f(z_i^{(\ell)} | z_i^{(\ell)}, s, \hat{\Theta})$.

Finally, we write

$$
E_{z_i^{(\ell)}, z_i^{(\ell+1)}, s_i | y_i, \hat{\Theta}} \left[ \frac{\partial \log L(z_i^{(\ell)} | z_i^{(\ell+1)}, s, \Theta)}{\partial \Psi_{k_\ell}^{(\ell)}} \right] = 0
$$

$$
\iff -\frac{1}{2} \psi_{k_\ell}^{(\ell)-1} \sum_{i=1}^{n} E_{z_i^{(\ell)}, z_i^{(\ell+1)}, s_i | y_i, \Theta} \left[ I_{R_1} - \left( z_i^{(\ell)} - (\eta_{k_\ell} + \Lambda_{k_\ell} z_i^{(\ell+1)}) \right) \left( z_i^{(\ell)} - (\eta_{k_\ell} + \Lambda_{k_\ell} z_i^{(\ell+1)}) \right)^T \psi_{k_\ell}^{(\ell)-1} \right] = 0
$$

$$
\iff \sum_{i=1}^{n} \sum_{s_i^{(k_\ell)}} f(s_i^{(k_\ell)} = \tilde{s}_i^{(k_\ell)} | y_i, \hat{\Theta}) I_{R_1} = \sum_{i=1}^{n} \sum_{s_i^{(k_\ell)}} f(s_i^{(k_\ell)} = \tilde{s}_i^{(k_\ell)} | y_i, \hat{\Theta}) E_{z_i^{(\ell)}, z_i^{(\ell+1)} | s_i^{(k_\ell)}, y_i, \hat{\Theta}} \left[ e^{(\ell)} e^{(\ell)T} \right] \psi_{k_\ell}^{(\ell)-1} = 0
$$

$$
\iff \sum_{i=1}^{n} \sum_{s_i^{(k_\ell)}} f(s_i^{(k_\ell)} = \tilde{s}_i^{(k_\ell)} | y_i, \hat{\Theta}) \psi_{k_\ell}^{(\ell)} = \sum_{i=1}^{n} \sum_{s_i^{(k_\ell)}} f(s_i^{(k_\ell)} = \tilde{s}_i^{(k_\ell)} | y_i, \hat{\Theta}) E_{z_i^{(\ell)}, z_i^{(\ell+1)} | s_i^{(k_\ell)}, y_i, \hat{\Theta}} \left[ e^{(\ell)} e^{(\ell)T} \right],
$$

with $e^{(\ell)} = \left( z_i^{(\ell)} - (\eta_{k_\ell} + \Lambda_{k_\ell} z_i^{(\ell+1)}) \right)$. Hence the estimator of $\Psi_{k_\ell}^{(\ell)}$ has the form

$$
\hat{\Psi}_{k_\ell}^{(\ell)} = \frac{\sum_{i=1}^{n} \sum_{s_i^{(k_\ell)}} f(s_i^{(k_\ell)} = \tilde{s}_i^{(k_\ell)} | y_i, \hat{\Theta}) E \left[ \left( z_i^{(\ell)} - (\eta_{k_\ell} + \Lambda_{k_\ell} z_i^{(\ell+1)}) \right) \left( z_i^{(\ell)} - (\eta_{k_\ell} + \Lambda_{k_\ell} z_i^{(\ell+1)}) \right)^T | \tilde{s}_i^{(k_\ell)}, y_i, \hat{\Theta} \right]}{\sum_{i=1}^{n} \sum_{s_i^{(k_\ell)}} f(s_i^{(k_\ell)} = \tilde{s}_i^{(k_\ell)} | y_i, \hat{\Theta})}.
$$
A.2 Head components and paths probabilities

In this section, we show how to estimate the probability of each component of each head layer. Let \( k_\ell \) the index of the component of layer \( \ell \) for which we want to derive an estimator and \( \tilde{k}_\ell \) another component index. The associated probabilities are respectively \( \pi_{k_\ell} \) and \( \pi_{\tilde{k}_\ell} \). We omit the head subscript \( h \) for better readability. We have

\[
E_{s^{(\ell)}|y,\hat{\Theta}}[\log L(s^{(\ell)}|\Theta)]
\]

\[
= \sum_{i=1}^{n} \sum_{k_{\ell}'=1}^{K_\ell} f(s^{(\ell)} = k_{\ell}'|y, \hat{\Theta}) \log L(s^{(\ell)} = k_{\ell}'|\Theta)
\]

\[
= \sum_{i=1}^{n} \sum_{k_{\ell}'=1}^{K_\ell} f(s^{(\ell)} = k_{\ell}'|y, \hat{\Theta}) \log L(s^{(\ell)} = k_{\ell}'|\Theta) + \sum_{i=1}^{n} f(s^{(\ell)} = \tilde{k}_{\ell}|y, \hat{\Theta}) \log L(s^{(\ell)} = \tilde{k}_{\ell}|\Theta)
\]

\[
= \sum_{i=1}^{n} \sum_{k_{\ell}'=1}^{K_\ell} f(s^{(\ell)} = k_{\ell}'|y, \hat{\Theta}) \log \pi_{k_{\ell}'} + \sum_{i=1}^{n} f(s^{(\ell)} = \tilde{k}_{\ell}|y, \hat{\Theta}) \log (1 - \sum_{k_{\ell}'=1}^{K_\ell} \pi_{k_{\ell}'})
\]

Taking the derivative with respect to \( \pi_{k_{\ell}} \) and equalizing to zero yields

\[
\frac{\partial E_{s^{(\ell)}|y,\hat{\Theta}}[\log L(s^{(\ell)}|\Theta)]}{\partial \pi_{k_{\ell}}^{(t)}} = 0 \iff \sum_{i=1}^{n} f(s^{(\ell)} = k_{\ell}|y, \hat{\Theta}) = \sum_{i=1}^{n} f(s^{(\ell)} = \tilde{k}_{\ell}|y, \hat{\Theta})
\]

\[
\iff \pi_{k_{\ell}}^{(t)} = \frac{\sum_{i=1}^{n} f(s^{(\ell)} = k_{\ell}|y, \hat{\Theta})}{\sum_{i=1}^{n} f(s^{(\ell)} = \tilde{k}_{\ell}|y, \hat{\Theta})} \pi_{k_{\ell}}^{(t)}
\]

Finally, summing over \( \tilde{k}_{\ell} \) we get

\[
\pi_{k_{\ell}}^{(t)} = \frac{\sum_{i=1}^{n} f(s^{(\ell)} = \tilde{k}_{\ell}|y, \hat{\Theta}) \pi_{k_{\ell}}^{(t)}}{\sum_{i=1}^{n} f(s^{(\ell)} = k_{\ell}|y, \hat{\Theta})} \implies \pi_{k_{\ell}}^{(t)} = \frac{n}{\sum_{i=1}^{n} f(s^{(\ell)} = k_{\ell}|y, \hat{\Theta})} \pi_{k_{\ell}}^{(t)} \iff \pi_{k_{\ell}}^{(t)} = \frac{\sum_{i=1}^{n} f(s^{(\ell)} = k_{\ell}|y, \hat{\Theta})}{n}.
\]

A.3 Common tail components probabilities

In this section, we highlight how to compute \( p(s^{(L_0+1)}|y^C, y^D, \hat{\Theta}_D, \hat{\Theta}_C, \hat{\Theta}_{(L_0+1)}) \). We are still making the two following conditional assumptions:

\[
(y^C \perp\!\!\!\perp y^D) | z^{(L_0+1)} \quad \text{and} \quad (z^D \perp\!\!\!\perp z^C) | z^{(L_0+1)},
\]

with \( z^{(L_0+1)} \) the first common tail layer. We then have:
\[
p(s^{(L_0+1)}|y^C, y^D, \hat{\Theta}_D, \hat{\Theta}_C, \hat{\Theta}_{(L_0+1)}) = \frac{p(s^{(L_0+1)}, y^C, y^D|\hat{\Theta}_D, \hat{\Theta}_C, \hat{\Theta}_{(L_0+1)})}{p(y^C, y^D)} \propto p(s^{(L_0+1)}, y^C, y^D|\hat{\Theta}_D, \hat{\Theta}_C, \hat{\Theta}_{(L_0+1)})
\]

\[
= \sum_s \sum_{s'} \int_{z^{(L_0+1)}} p(s^{(L_0+1)}, y^C, y^D, z^{(L_0+1)}, s', s^D|\hat{\Theta}_D, \hat{\Theta}_C, \hat{\Theta}_{(L_0+1)}) dz^{(L_0+1)}
\]

\[
= \sum_s \sum_{s'} \int_{z^{(L_0+1)}} p(y^C|s^C, z^{(L_0+1)}, \hat{\Theta}_C, \hat{\Theta}_{(L_0+1)}) p(y^D|s^D, s^{(L_0+1)}, z^{(L_0+1)}, \hat{\Theta}_D, \hat{\Theta}_{(L_0+1)})
\]

\[
\times \sum_{s'} p(s^{(L_0+1)}|s^{(L_0+1)}, \hat{\Theta}_{(L_0+1)}) \prod_{\ell=1}^{L_0} p(s^{(\ell)}|\hat{\Theta}_C, \hat{\Theta}_{(L_0+1)}) \prod_{\ell=L_0+1}^{L} p(s^{(\ell)}|\hat{\Theta}_{(L_0+1)}) dz^{(L_0+1)},
\]

by independence of the \((s^{(\ell)h})_{\ell,h}\). The first two terms are computed as for (12), the third term is a Gaussian given in (11) and each density of the products are multinomial densities whom coefficients have already been estimated.

### A.4 Latent variables identifiability rescaling

The GLLVM and Factor Analysis models assume that the latent variable is centered and of unit variance. We define \(z^{(1)h}_{new}\) the rescaled version of \(z^{(1)h}\) satisfying those constraints for each head \(h\) (simply called \(z^{(1)}\) in the following for each head).

Let \((\tilde{s}, \tilde{s}') \in \Omega^2\) be two paths through the network starting from the head \(h\) and \(A\) the Cholesky decomposition of \(Var(z^{(1)})\) such that \(Var(z^{(1)}) = AA^T\). We will first prove that the following rescaling:

\[
\begin{cases}
\Sigma^{new}_\tilde{s} = A^{-1} \Sigma_{\tilde{s}} A^{-1} \\
\mu^{new}_\tilde{s} = A^{-1} [\mu_{\tilde{s}} - \sum_{\tilde{s}'} \pi_{\tilde{s}'s'} \mu_{\tilde{s}'}],
\end{cases}
\]

\(\forall \tilde{s} \in \Omega\), leads to \(z^{(1)new}\) having unit variance and zero mean.

We have

\[
\begin{align*}
E(z^{(1)}) &= \sum_{\tilde{s}'} \pi_{\tilde{s}'s} \mu_{\tilde{s}'} \\
Var(z^{(1)}) &= \sum_{\tilde{s}} \pi_{\tilde{s}} (\Sigma_{\tilde{s}} + \mu_{\tilde{s}} \mu_{\tilde{s}}^T) - (\sum_{\tilde{s}} \pi_{\tilde{s}} \mu_{\tilde{s}})(\sum_{\tilde{s}} \pi_{\tilde{s}} \mu_{\tilde{s}})^T,
\end{align*}
\]

which gives, after the rescaling:
In our case, we aim to ensure these conditions by correcting only the parameters of the first layer, i.e. only \((\eta_{k_1}^{(1)}, \Lambda_k^{(1)}, \Psi_{k_1}^{(1)}) \; \forall k_1 \in [1, K_1]\) rather than all of the parameters of the following layers. We will then prove that rescaling \(\mu_{\tilde{s}}\) and \(\Sigma_{\tilde{s}} \; \forall \tilde{s} \in \Omega\) is equivalent to a rescaling of those parameters.

\[
\sum_{\tilde{s}}^{new} = A^{-1T} \Sigma_{\tilde{s}} A^{-1}
\]

\[
= A^{-1T} \left[ \Psi_{k_1}^{(1)} + \Lambda_k^{(1)} \left( \Psi_{k_2}^{(2)} + \sum_{\ell=3}^L \left( \prod_{m=2}^{\ell-1} \Lambda_k^{(m)} \right) \left( \prod_{m=1}^{\ell-1} \Lambda_k^{(m)} \right) \right) \right] A^{-1}
\]

\[
= A^{-1T} \Psi_{k_1}^{(1)} A^{-1} + A^{-1T} \Lambda_k^{(1)} \left( \prod_{m=2}^L \Lambda_k^{(m)} \right) \left( \prod_{m=1}^L \Lambda_k^{(m)} \right) \Lambda_k^{(1)} A^{-1}
\]

\[
= \Psi_{k_1}^{(1) new} + \Lambda_k^{(1) new} \left( \prod_{m=2}^L \Lambda_k^{(m)} \right) \left( \prod_{m=1}^L \Lambda_k^{(m)} \right) \Lambda_k^{(1) new}.
\]

Hence by identification, one has to perform the following rescaling: \(\Psi_{k_1}^{(1) new} = A^{-1T} \Psi_{k_1}^{(1)} A^{-1}\) and \(\Lambda_k^{(1) new} = \)
For the second transformation, we obtain:

\[
\mu^{\text{new}}_s = A^{-1T} \left[ \mu_{\tilde{s}} - \sum_{s'} \bar{\pi}_{s'} \mu_{s'} \right]
\]

\[
= A^{-1T} \left[ \eta^{(1)}_{k'_1} + \Lambda^{(1)}_{k'_1} \left( \eta^{(2)}_{k'_2} + \sum_{\ell=3}^{L} \left( \prod_{m=2}^{\ell-1} \Lambda^{(m)}_{k'_m} \right) \eta^{(\ell)}_{k'_\ell} \right) - \sum_{s'} \bar{\pi}_{s'} \mu_{s'} \right]
\]

\[
= A^{-1T} \left[ \eta^{(1)}_{k'_1} + \Lambda^{(1)}_{k'_1} \left( \eta^{(2)}_{k'_2} + \sum_{\ell=3}^{L} \left( \prod_{m=2}^{\ell-1} \Lambda^{(m)}_{k'_m} \right) \eta^{(\ell)}_{k'_\ell} \right) \right]
\]

\[
= \eta^{(1)\text{new}}_{k'_1} + \Lambda^{(1)\text{new}}_{k'_1} \left( \eta^{(2)}_{k'_2} + \sum_{\ell=3}^{L} \left( \prod_{m=2}^{\ell-1} \Lambda^{(m)}_{k'_m} \right) \eta^{(\ell)}_{k'_\ell} \right),
\]

and we deduce that \( \Lambda^{(1)\text{new}}_{k'_1} = A^{-1T} \Lambda^{(1)\text{new}}_{k'_1} \) but also that \( \eta^{(1)\text{new}}_{k'_1} = A^{-1T} \left[ \eta^{(1)}_{k'_1} - \sum_{s'} \bar{\pi}_{s'} \mu_{s'} \right] \).

To summarize things up at the end of each of the MCEM steps one must perform the following transformations to ensure identifiability of the model:

\[
\begin{align*}
\eta^{(1)\text{new}}_{k'_1} &= A^{-1T} \left[ \eta^{(1)}_{k'_1} - \sum_{s'} \bar{\pi}_{s'} \mu_{s'} \right] \\
\Lambda^{(1)\text{new}}_{k'_1} &= A^{-1T} \Lambda^{(1)}_{k'_1} \\
\Psi^{(1)\text{new}}_{k'_1} &= A^{-1T} \Psi^{(1)}_{k'_1} A^{-1},
\end{align*}
\]
B Gaussian Parameters expressions

A DGMM can be written at each layer as a regular Gaussian Mixture with a number of components equal to the number of paths starting from that layer. The Gaussian mean and covariance matrix of each path starting from the $k_\ell$ component of layer $\ell$ can be computed in the following way:

$$\mu^{(\ell)}_{\tilde{s}_\ell^{(k_\ell)}} = \eta_{k_\ell}^{(\ell+1)} + \sum_{j=\ell+1}^{L} \left( \prod_{m=\ell}^{j-1} \Lambda^{(m)}_{k_m'} \eta^{(j)}_{k_j'} \right),$$

and

$$\Sigma^{(\ell)}_{\tilde{s}_\ell^{(k_\ell)}} = \Psi^{(\ell)}_{k_\ell} + \sum_{j=\ell+1}^{L} \left( \prod_{m=\ell}^{j-1} \Lambda^{(m)}_{k_m'} \right) \left( \Psi^{(j)}_{k_j'} + \Lambda^{(j)}_{k_j'} \Lambda^{(j)}_{k_j'}^T \right) \left( \prod_{m=\ell}^{j-1} \Lambda^{(m)}_{k_m'} \right)^T.$$

In addition, we have that the random variable $(z^{(\ell+1)}|z^{(\ell)}, \tilde{s}, \tilde{\Theta})$ also follows a multivariate Gaussian distribution with mean and covariance parameters $(\rho^{(\ell+1)}_{k_{\ell+1}}, \xi^{(\ell+1)}_{k_{\ell+1}})$:

$$\rho^{(\ell+1)}_{k_{\ell+1}} = \xi^{(\ell+1)}_{k_{\ell+1}} \left( \Lambda^{(\ell+1)}_{k_{\ell+1}} \Psi^{(\ell+1)}_{k_{\ell+1}} - 1 \right) + \sum_{k_{\ell+1}} \left( \Sigma^{(\ell+1)}_{\tilde{s}_\ell^{(k_{\ell+1})}} \right) \mu^{(\ell+1)}_{\tilde{s}_\ell^{(k_{\ell+1})}},$$

and

$$\xi^{(\ell+1)}_{k_{\ell+1}} = \left( \sum_{k_{\ell+1}} \left( \Sigma^{(\ell+1)}_{\tilde{s}_\ell^{(k_{\ell+1})}} \right) + \Lambda^{(\ell+1)}_{k_{\ell+1}} \Psi^{(\ell+1)}_{k_{\ell+1}} - 1 \Lambda^{(\ell+1)}_{k_{\ell+1}} \right)^{-1}.$$

C Model selection details

This section gives additional details about the way model selection is performed on the fly.

A component of the $\ell$th layer is considered useless if its probability is inferior to $\frac{1}{4k_\ell}$, where $k_\ell$ denotes the number of components of the layer. For instance, if a layer is formerly composed of four components, the components associated with a probability inferior to 0.0625 are removed from the architecture.

For the GLLVM layer, logistic regressions are fitted to determine which of the dimensions had a significant effect over each $y_\ell^D_j$ for each path $\tilde{s}$. We have fitted a logistic LASSO for each binary and count variable and an ordinal logistic regression for each ordinal variable. The variables associated with coefficients identified as being zero (or not significant at a 10% level) for at least 25% of the paths were removed.

The same voting idea was used for the regular DGMM layers to determine the useless dimensions. As our algorithm generate draws of $(z^{(\ell+1)}|z^{(\ell)}, s)$ of dimension $r_{\ell+1}$, it is possible to perform a PCA on this variable for each path and each of the $M^{(\ell)}$ points simulated for $z^{(\ell)}$. Doing so, one can compute the average contribution of each dimension of $r_{\ell+1}$ to the first principal component and set a threshold under which a dimension is deleted.

We have set this threshold to 0.2 for our simulations. The intuition behind this is that the first component of the PCA conveys the majority of the pieces of information that $z^{(\ell+1)}$ has on $z^{(\ell)}$. If a dimension shares no common information with this first component, hence it is not useful to keep it.
The dimension of the junction layer (the first DGMM layer on the common tail) is chosen according to this procedure too. The two heads decide which dimensions of the junction layer is important and each dimension important for at least one head is kept. This is rather conservative but avoids that contradictory information coming from the two heads disrupt the global architecture.

The number of layers on the heads and tails is fully determined by the selection of the layers dimensions in order to keep the model identifiable. If the dimension of an intermediate tail layer $\ell$ is selected to be one then one cannot have $r_\ell > r_{\ell+1} > ... > r_L$. Thus, the following tail layers are deleted.

Similarly, if an head layer has a selected dimension of two, then the following head layers are deleted. Indeed, the tail has to have minimal dimensions of two and one on its last layers. This is not compatible with previous head layers of dimension inferior or equal to two.

In the case of head layers deletion, we restart the algorithm (initialisation and proper model run) with the new architecture. Otherwise it would be necessary to re-determine all the path and DGMM coefficients values to bridge the gap between the previous head layer and the junction layer. There were no easy way to do such thing and restarting the algorithm seemed the best to do. Note that in our simulations defining several heads layers did not give good results. Intuitively, it could too much dilute information before passing it to the common tail, resulting in poor performance. We advise to keep only one or two head layers before running the MDGMM. Doing so, this restarting procedure would not be often performed in practice.

### D Metrics

A true positive (TP) prediction of the model is an observation that has been assigned to the same class as the ”ground truth” label. On the contrary, a False Positive (FP) means that the class predicted by the model and the label do not match. $k$ denotes the class index and $K$ the cardinal of the set of all possible classes. $n_k$ is the number of points in the class $k$ and $y_{i,k}$ an observation of class $k$.

The formulas of the two precision metrics are:

\[
\text{Micro precision} = \frac{\sum_{k=1}^{K} \sum_{i=1}^{n} TP_{i,k}}{\sum_{k=1}^{K} \sum_{i=1}^{n} TP_{i,k} + FP_{i,k}}
\]

\[
\text{Macro precision} = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{i=1}^{n} TP_{i,k}}{\sum_{i=1}^{n} TP_{i,k} + FP_{i,k}}
\]

The formula of the silhouette coefficient is:

\[
\text{Silhouette coefficient} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i=1}^{n} \frac{d_{\text{inter}}(i,k) - d_{\text{intra}}(i,k)}{\max(d_{\text{intra}}(i,k),d_{\text{inter}}(i,k))}
\]

with $d_{\text{intra}}(i,k) = \frac{1}{n_k-1} \sum_{i' \neq i} d(y_{i,k}, y_{i',k})$ and $d_{\text{inter}}(i,k) = \min_{k' \neq k} \frac{1}{n_{k'}} \sum_{i' \neq i} \sum_{k' = 1}^{K} d(y_{i,k}, y_{i',k'})$

With $d$ a distance, the Gower distance (Gower, 1971) in our case.
E Benchmark models specifications

A standard Grid Search has been performed to find the best specification of the hyperparameters of the benchmark models. The following hyperparameters search spaces were used:

**K-modes** (from the kmodes package)
- Initialisation ∈ {'Huang', 'Cao', 'random'}.

**K-prototypes** (from the kmodes package)
- Initialisation ∈ {'Huang', 'Cao', 'random'}.

**Agglomerative clustering** (from the scikit-learn package)
- linkages ∈ {'complete', 'average', 'single'}.

**Self-Organizing Map** (from the SOMPY package)
- sigma ∈ [0.001, 0.751, 1.501, 2.250, 3.000]
- lr ∈ [0.001, 0.056, 0.111, 0.167, 0.223, 0.278, 0.333, 0.389, 0.444, 0.500].

**DBSCAN** (from the scikit-learn package)
- leaf_size ∈ {10, 20, 30, 40, 50}
- eps ∈ {0.01, 1.258, 2.505, 3.753, 5.000}
- min_samples ∈ {1, 2, 3, 4}
- Data used: 'scaled data', 'Gower Distance'.

**DBSCAN** was trained on two versions of the dataset. First on the scaled data as if they were all continuous, neglecting the mixed nature of the data. The second training was made by passing the Gower Distance Matrix computed on the data to DBSCAN. Each time the best performing specification was taken.

**GLMLVM**
- \( r \in [1, 5] \)
- \( k = 2 \).

**NESP (MCA + GMM + FA)**
- \( r \in [1, 13] \)
• $k = 2$.

**DDGMM**

The starting architecture over which automatic architecture selection was performed was:

• $r = \{7, 5, 3\}$
• $k = \{4, 2\}$
• Number of maximum iterations = 30.

**NESP (MCA + GMM + FA + PLS)**

The architectures considered had at most 2 layers on each head and 3 layers on the tail.

• $r$: All the minimal identifiable architectures.
• $k$: Random draws for each $k_\ell \in \{2, 3, 4\}$
• Number of maximum iterations = 30.

**M1DGMM**

The starting architecture over which automatic architecture selection was performed was:

• $r = \{5, 4, 3\}$
• $k = \{4, 2\}$
• Number of maximum iterations = 30.

**MDGMM**

The starting architecture over which automatic architecture selection was performed was:

• $r_c = \{p_c\}, r_d = \{5\}, r_t = \{4, 3\}$
• $k_c = \{1\}, k_d = \{1\}, k_{L_0+1} = \{2, 1\}$
• Number of maximum iterations = 30.

$k_c = \{1\}$ and $r_c = \{p_c\}$ are imposed by construction as the first layer of the continuous head are the data themselves.