MDGMM: A clustering model for mixed datasets
8th draft

Robin Fuchs*[1], Denys Pommeret †[2], and Cinzia Viroli ‡[3]

[1,2] Institut de Mathématiques de Marseille,163 Avenue de Luminy, 13009 Marseille
[3] Department of Statistical Sciences, University of Bologna, Via Belle Arti 41, 40126 Bologna, Italy

Abstract

Clustering mixed datasets presents numerous challenges inherent to the very heterogeneous nature of the data. Existing works mention the difficulty of finding proper initialisation strategies and to merge clusterings when several pre-clusterings are performed on the different data types before building a unique clustering.

In this work, we introduce a two-heads architecture model-based clustering method called Mixed data Deep Gaussian Mixture Models (MDGMM) that can be viewed as an automatic way to merge the clusterings performed separately on discrete and continuous data. We also design a new initialisation strategy and a model selection method that selects "on the fly" the best specification of the model 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.

*robin.fuchs@univ-amu.fr
†denys.pommeret@univ-amu.fr
‡cinzia.viroli@unibo.it

1 Introduction

Mixed datasets are made of variables of heterogeneous nature: 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), ordinal data (ordered data taking a finite number of modalities) or continuous data (generated by real-valued random variables). Due to they difference in nature, these variables does not share common scales and it is typically hard to characterise what similarity between observations is.
They have been a significant recent interest for mixed data clustering, which can be regrouped in four main categories according to [1]: partional algorithms, hierarchical algorithms, model-based, Neural Networks based.

Partitional algorithms try to find cluster centers and define a distance between observations and those centers. Finally, they optimise iteratively a metric based on the distance defined. Example of those algorithms are K-modes or K-prototypes.[16][17]
Hierarchical algorithms perform nested clusterings and then find a way to merge them to create the final clustering.
On the other hand, as their name suggests it, model-based algorithms relies on statistical fundations such as probability distributions, expectations etc...
Finally, Neural Networks-based algorithms design the clusters thanks to connected neurons that learns complex patterns contained in the data.

Our Mixed data Deep Gaussian Mixture Model (DGMM) belongs to the model-based algorithms family and more precisely to the family of Generalized Linear Latent Variable Models (GLLVM) [20][21]. This class of models supposes that their exists a link function between the discrete data space and a continuous latent space often chosen to be Gaussian.
Cagnone and Viroli [7] extend this approach by considering a latent variable that is no more a Gaussian but a mixture of Gaussians [11]. As a result, it enables the model to conduct clustering in the latent space. As the latent dimension is chosen to be strictly inferior to the original dimension, the model also performs dimension reduction by definition.
Our work generalizes this approach by considering a Deep Gaussian Mixture model as latent variable. DGMMs have been introduced by Viroli and McLachlan (2019) [28] and can be seen as a series of nested Gaussian Mixtures and more precisely as a series of Mixture of Factor Analyser (MFA) [13]. We also introduce a new initialisation method for such models which increases the performance and stabilizes the algorithm through its run.
As such, our work also belong to the mixture models literature.

The model-based clustering algorithms are often described as slow compared to partitional algorithms [1]. However, using auto-differentiation methods [3] 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 [5]) truly cuts the computational running time. For instance, for the special case of GLLVM models, Niku and al. (2019) [23] report very significant computational gains from using auto-differentiation methods.
Our model implementation relies on automatic differentiation that helps keeping an acceptable running time even when the number of layers increases.

The multilayer architecture is inspired by Supervised Neural Networks and relies on the same idea that composing simple functions enable 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. In our case, we define two "heads", one for discrete data (binary, categorical and ordinal data) and one for continuous data as well as 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.

The GLLVM framework enables to include continuous variables into account by specifying a distribution belonging to the exponential family as link function. As a result, we could in principle have treated both

discrete and continuous data into the same "head".
Specifying a parametric distribution for discrete data enables to structure very heterogeneous sources of information. However, in the case of continuous variables, the variables are more easy comparable and specifying a wrong distribution might be very harmful (for heavy tailed data for instance) even more than for discrete data.

Thus, we introduce a two-heads architecture in order not to use a parametric distribution specification for continuous data and perform clustering in the common tail. Each layer of the network is trained using a Monte-Carlo Expectation-Maximisation (MCEM) algorithm [29].

2 Model presentation

2.1 The MDGMM as a generalization of existing models

[Be careful of C, D superscripts]. [Add the sources]
The model is build upon several models: The Mixture of Factor Analysers (MFA) generalized by the Deep Gaussian Mixture Models (DGMM) and the Generalized Linear Latent Variable Models (GLLVM).

The Factor Analyser (FA) [14] model that is the most elementary building block of our model can be formulated as follows:

$$y_i^C = \eta + \Lambda z_i + u_i$$

with y^C some continuous data made of n observations and p_C variables, η a constant vector of size p_C, $z_i \sim \mathrm{N}(0, I_r)$ $\forall i \in [1, n]$, $u_i \sim \mathrm{N}(0, \Psi)$ and Λ the factor loading matrix of dimension $p_C \times r$.
The underlying idea, is to find a latent representation of the data of dimension r with $p < r$. 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 representation, the model then becomes a Mixture of Factor Analyser (MFA) [13].

$$y_i^C = \eta_{k_1} + \Lambda_{k_1} z_i + u_{ik_1} \text{ with probability } \pi_{i,(k_0,k_1)}$$

For $k_0 \in \{1\}$, $k_1 \in [1, K_1]$ and with $z_i \sim \mathrm{N}(0, I_p)$ and $p < r_1$.

It is possible to extend once again the MFA model by assuming that z_i is no more drawn from a multivariate Gaussian but is itself a MFA. The corresponding model is a two hidden layers DGMM [28]:

$$\begin{cases} y_i^C = \eta_{k_1}^{(1)} + \Lambda_{k_1}^{(1)} z_i^{(1)} + u_{ik_1}^{(1)} \text{ with probability } \pi_{i,(k_0,k_1)}^{(1)} \\ z_i^{(1)} = \eta_{k_2}^{(2)} + \Lambda_{k_2}^{(2)} z_i^{(2)} + u_{ik2}^{(2)} \text{ with probability } \pi_{i,(k_1,k_2)}^{(2)} \end{cases} \tag{1}$$

with $z_i^{(2)} \sim \mathrm{N}(0, I_{r_2})$, $k_0 \in \{1\}$, $k_1 \in [1, K_1]$ et $k_2 \in [1, K_2]$ and $p < r_1 < r_2$.

Deeper DGMMs can be defined by rewriting iteratively the last latent variables as MFAs. Doing so, one ends up with the following L-layers deep DGMM:

$$\begin{cases} y_i^C = \eta_{k_1}^{(1)} + \Lambda_{k_1}^{(1)} z_i^{(1)} + u_{ik_1}^{(1)} \text{ with probability } \pi_{i,(k_o,k_1)}^{(1)} \\ z_i^{(1)} = \eta_{k_2}^{(2)} + \Lambda_{k_2}^{(2)} z_i^{(2)} + u_{ik_2}^{(2)} \text{ with probability } \pi_{i,(k_1,k_2)}^{(2)} \\ ... \\ z_i^{(L-1)} = \eta_{k_L}^{(L)} + \Lambda_{k_L}^{(L)} z_i^{(L)} + u_{ik_L}^{(L)} \text{ with probability } \pi_{i,(k_{L-1},k_L)}^{(L)} \\ z_i^{(L)} \sim \mathcal{N}(0, I_{r_L}) \end{cases} \tag{2}$$

For $k_0 \in \{1\}$, $k_1 \in [1, K_1]$ and $k_2 \in [1, K_2]$, ... and $p < r_1 < r_2 < ... < r_L$.

However, in order to use the DGMM one needs the data to be continuous. In order to apply a DGMM to a discrete dataset, one would first have to find a continuous representation of discrete data. In order to do so, we have integrated the Generalized Linear Latent Variable Model (GLLVM) framework within the DGMM framework.

The GLLVM assumes that the discrete variables $y_j^D \in \mathbb{R}^n$, $\forall j \in [1, p_D]$ are mutually independent with respect to the latent variables $z^{(1)D}$. The function linking each original variable to the latent variable belongs to the exponential family and has the following form:

$$f(y_j^D | z^{(1)D}) = \exp\Big(\frac{y_j^D - b_j(\theta_j)}{\phi_j} + c_j(y_j^D, \phi_j)\Big)$$

with θ_j, ϕ_j, c_j coefficients to estimate that indirectly depend on $z^{(1)D}$.

Hence, defining $z^{(1)D}$ as the first level of a DGMM one can use the 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 datasets that present both discrete and continuous variables, we make the additional assumption that discrete and continuous data are independent. The continuous data can be handled as in (2) and the discrete data by a DGMM with a GLLVM embedding layer. The two clustering processes can then be merged by specifying that the last layers are actually shared by the two networks. These ending layers, called common tail layers in the following, enable to synthesise information carried by both the discrete and continuous heads and to perform the clustering.

2.2 Formal definition

Let y be the $n \times p$ matrix of the observed variables, $i \in [1, n]$ the observation index and $j \in [1, p]$ the variable index. Using the notations defined earlier we have that $y = y^C \cup y^D$ with $y^C \in \mathbb{R}^n \times \mathbb{R}^{p_c}$ and $y^D \in \mathbb{R}^n \times \mathbb{R}^{p_D}$. For all $h \in \{C, D\}$, y_j^h is therefore a vector of dimension n.

The following equations present the global architecture of the model introduced in the previous section:

$$\begin{cases}
\text{Discrete head} : \begin{cases} y_i^D \to z_i^{(1)D} \text{ through GLLVM link via } (\lambda^{(0)}, \Lambda^{(0)}) \\ z_i^{(1)D} = \eta_{k_1}^{(1)D} + \Lambda_{k_1}^{(1)D} z_i^{(2)D} + u_{i,k_1}^{(1)D} \text{ with probability } \pi_{i,(k_1,k_2)}^{(1)D} \\ \dots \\ z_i^{(\underline{L})D} = \eta_{k_{\underline{L}}}^{(\underline{L})D} + \Lambda_{k_{\underline{L}}}^{(\underline{L})D} z_i^{(\underline{L}+1)} + u_{k_{\underline{L}}}^{(\underline{L})D} \text{ with probability } \pi_{i,(k_{\underline{L}},k_{\underline{L}+1})}^{(\underline{L})D} \end{cases} \\
\text{Continuous head} : \begin{cases} y_i^C = \eta_{k_1}^{(1)C} + \Lambda_{k_1}^{(1)C} z_i^{(2)C} + u_{i,k_1}^{(1)C} \text{ with probability } \pi_{i,(k_1,k_2)}^{(1)C} \\ z_i^{(1)C} = \eta_{k_1}^{(1)C} + \Lambda_{k_1}^{(1)C} z_i^{(1)C} + u_{i,k_1}^{(1)C} \text{ with probability } \pi_{i,(k_1,k_2)}^{(1)C} \\ \dots \\ z_i^{(\underline{L})C} = \eta_{k_{\underline{L}}}^{(\underline{L})C} + \Lambda_{k_{\underline{L}}}^{(\underline{L})C} z_i^{(\underline{L}+1)} + u_{k_{\underline{L}}}^{(\underline{L})} \text{ with probability } \pi_{i,(k_{\underline{L}},k_{\underline{L}+1})}^{(\underline{L})C} \end{cases} \\
\text{Common tail :} \\
z_i^{(\underline{L}+1)} = \eta_{k_{\underline{L}+1}}^{(\underline{L}+1)} + \Lambda_{k_{\underline{L}+1}}^{(\underline{L}+1)} z_i^{(\underline{L}+2)} + u_{k_{\underline{L}+1}}^{(\underline{L}+1)} \text{ with probability } \pi_{i,(k_{\underline{L}+1},k_{\underline{L}+2})}^{(\underline{L}+1)} \\
\dots \\
z_i^{(L-1)} = \eta_{k_{L-1}}^{(L-1)} + \Lambda_{k_{L-1}}^{(L-1)} z_i^{(L)} + u_{k_{L-1}}^{(L-1)} \text{ with probability } \pi_{i,(k_{L-1},k_L)}^{(L-1)} \\
z_i^{(L)} \sim \mathcal{N}(0, I_{r_L})
\end{cases}$$

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 coefficients

of the GLLVM layer are (λ_0, Λ_0). The $\lambda_0 = (\lambda_{0bin}, \lambda_{0count}, \lambda_{0ord})$ coefficients are intercept coefficients for each discrete data sub-type. We make the distinction between four subtypes 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. If the categorical variable has for instance 5 modalities we convert it to 4 dummy variables.
Λ_0 is a matrix of size $p' \times r_1$, with r_1 the dimension of the first Discrete DGMM layer and p' the dimension of the 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 $z^C = (z^{(1)C}, ..., (z^{(\underline{L})C})$ is the set of latent variables of the continuous head. Notice that by an abuse of notation, we use $\underline{L}$ as the last layer index for both heads even if each head can have a different number of layers. We omit the h subscript to lighten the notations.

In addition we introduce, $s^{(l)h} \in [1, K_l^h]$ the latent variable associated with the index of the component k_l^h of the layer l 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 $s^h = (s^{(1)h}, ..., s^{(\underline{L})h})$. Similarly, the random variable associated to a path going through all the common tail layers is $s^{(\underline{L}+1:)} = (s^{(\underline{L}+1)}, ..., s^{(L)})$. By extension, the variable associated with a full path going from the beginning of head h to the end of the common tail is $s^{(1h:L)} = (s^h, s^{\underline{L}+1})$. $s^{(1h:L)}$ belongs to Ω^h the set of all possible paths starting from one head of cardinal $S^h = \prod_{l=1}^{L} K_l^h$. The variable associated with a path going from layer l of head h to layer L will be denoted $s^{(lh:L)}$. Finally, by a slight abuse of notation a full path going through the component k_l^h of the l-th layer of head h will be denoted: $s^{(1:k_l^h:L)}$ or more simply $s^{(:k_l^h:)}$.

In order to be the more concise as possible, we group the parameters of the model by defining:

$$\begin{aligned}
\Theta^D &= (\Theta_{emb}, \Theta_{DGMM}) = ((\lambda_0, \Lambda_0), (\eta_{k_l}^{(l)D}, \Lambda_{k_l}^{(l)D}, \Psi_{k_l}^{(l)D})_{k_l \in [1, K_l^D], l \in [1, \underline{L}_D]}) \\
\Theta^C &= (\eta_{k_l}^{(l)C}, \Lambda_{k_l}^{(l)C}, \Psi_{k_l}^{(l)C})_{k_l \in [1, K_l^C], l \in [1, \underline{L}_C]} \\
\Theta^{(\underline{L}+1:)} &= (\eta_{k_l}^{(l)(\underline{L}+1)}, \Lambda_{k_l}^{(l)(\underline{L}+1)}, \Psi_{k_l}^{(l)(\underline{L}+1)})_{k_l \in [1, K_l^{(\underline{L}+1)}], l \in [\underline{L}_C, L]}
\end{aligned}$$

with *emb* standing for embedding.

As an illustration, the graphical model of a simple MDGMM is provided in the following figure:

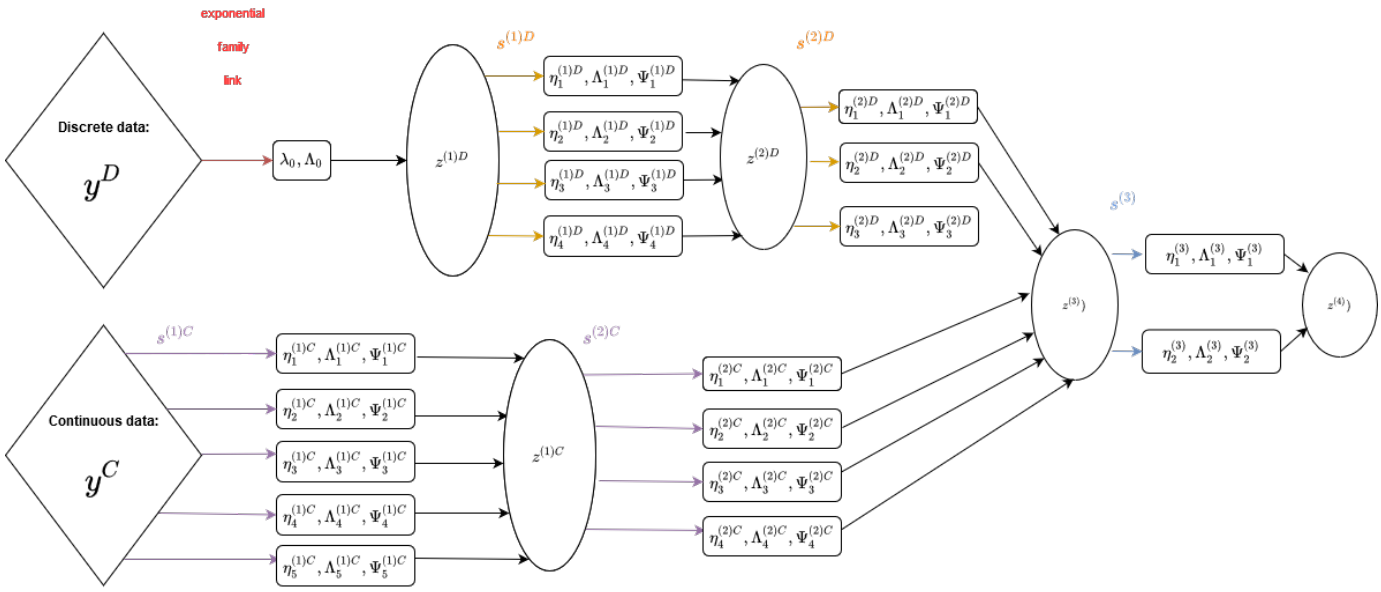


Figure 1: Graphical model of a Mixed data DGMM (MDGMM)

The complete density of the model is:

$$\begin{aligned}
&L(y^C, y^D, z^C, z^D, z^{(\underline{L}+1:)}, s^C, s^D, s^{(\underline{L}+1:)} | \Theta_C, \Theta_D, \Theta_{\underline{L}+1:}) \\
&\quad = L(y^C | z^{(1)C}, s^C, s^{(\underline{L}+1:)}, \Theta_C, \Theta_{\underline{L}+1:}) L(z^C | z^{(\underline{L}+1:)}, s^C, s^{(\underline{L}+1:)}, \Theta_C, \Theta_{\underline{L}+1:}) \\
&\quad \times L(y^D | z^{(1)D}, s^D, s^{(\underline{L}+1:)}, \Theta_D, \Theta_{\underline{L}+1:}) L(z^D | z^{(\underline{L}+1:)}, s^D, s^{(\underline{L}+1:)}, \Theta_D, \Theta_{\underline{L}+1:}) \\
&\quad \times L(z^{(\underline{L}+1:)} | s^C, s^D, s^{(\underline{L}+1:)}, \Theta_C, \Theta_D, \Theta_{\underline{L}+1:}) \\
&\quad \times L(s^C, s^D, s^{(\underline{L}+1:)} | \Theta_C, \Theta_D, \Theta_{\underline{L}+1:})
\end{aligned}$$

which comes from the fact that the two heads of the model are independent. Moreover, the layers of both heads and tail share the Markov property derived from the graphical model that $(z^{(l)h} \perp\!\!\!\perp z^{(l+2)h}, ..., z^{(L)}h) \Big| z^{(l+1)h}$, $\forall h \in \{C, D, (\underline{L}+1)\}$.

The aim of the training is to maximize the expected log-likelihood, i.e. to maximize :

$$\begin{aligned}
&E_{z^C, z^D, z^{(\underline{L}+1)}, s^C, s^D, s^{(\underline{L}+1:)} | y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}} [\log L(y^C, y^D, z^C, z^D, z^{(\underline{L}+1)}, s^C, s^D, s^{(\underline{L}+1:)}, \Theta_C, \Theta_D, \Theta_{\underline{L}+1:})] \\
&= E_{z^{(1)D}, s^D, s^{(\underline{L}+1:)D} | y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}} [\log L(y^D | z^{(1)D}, s^D, s^{(\underline{L}+1:)}, \Theta_D, \Theta_{\underline{L}+1:})] \\
&+ E_{z^{(1)C}, s^C, s^{(\underline{L}+1:)C} | y^C, \hat{\Theta}_C, \hat{\Theta}_{\underline{L}+1:}} [\log L(y^C | z^{(1)C}, s^C, s^{(\underline{L}+1:)}, \Theta_C, \Theta_{\underline{L}+1:})] \\
&+ \sum_{h \in \{C,D\}} \sum_{l=1}^{\underline{L}} E_{z^{(l)h}, z^{(l+1)h}, s^h, s^{(\underline{L}+1:)} | y^h, \hat{\Theta}_h, \hat{\Theta}_{\underline{L}+1:}} [\log L(z^{(l)h} | z^{(l+1)h}, s^h, s^{(\underline{L}+1:)}, \Theta_h, \Theta_{(\underline{L}+1:)})] \\
&+ \sum_{l=\underline{L}+1}^{L-1} E_{z^{(l)}, z^{(l+1)}, s^C, s^D, s^{(\underline{L}+1:)} | y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}} [\log L(z^{(l)} | z^{(l+1)}, s^C, s^D, s^{(\underline{L}+1:)}, \Theta_C, \Theta_D, \Theta_{(\underline{L}+1:)})] \\
&+ E_{z^{(L)} | y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}} [\log L(z^{(L)} | \Theta_C, \Theta_D, \Theta_{\underline{L}+1}))] \\
&+ E_{s^C, s^D, s^{(\underline{L}+1:)} | y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}} [\log L(s^C, s^D, s^{(\underline{L}+1:)} | \Theta_C, \Theta_D, \Theta_{\underline{L}+1})] \qquad (3)
\end{aligned}$$

with a slight abuse of notation in the double sum : $z^{(\underline{L}+1)} = z^{(\underline{L}+1)C} = z^{(\underline{L}+1)D}$. One also defines $\hat{\Theta}^h$ as the provisional estimate of Θ^h at iteration t of the algorithm.
The following sections expose how to compute the estimators related to these expectations.

3 Model training

The training of the model involves training three types of layers by MCEM: The GLLVM layer, the regular DGMM layers and the common tail layers that unit the two heads.

3.1 Generalized Linear Latent Variable Model Embedding Layer

This section is based on Cagnone and Viroli (2014) [7] which itself is based on seminal works by Moustaki (2003) [20] and Moustaki and Knot (2000) [21]. 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 mutual independence assumption between discrete variables evoked earlier, the likelihood of the data can be written as:

$$\begin{aligned} f(y^D|\Theta_D, \Theta_{\underline{L}+1:}) &= \int_{z^{(1)D}} f(y^D|z^{(1)D}, \Theta_D, \Theta_{\underline{L}+1:}) f(z^{(1)D}|\Theta_D, \Theta_{\underline{L}+1:}) dz^{(1)D} \\ &= \int_{z^{(1)D}} \prod_{j=1}^{p_d} f(y_j^D|z^{(1)D}, \Theta_D, \Theta_{\underline{L}+1:}) f(z^{(1)D}|\Theta_D, \Theta_{\underline{L}+1:}) dz^{(1)D} \end{aligned} \tag{4}$$

with y_j^D a discrete variable of the dataset.

For each discrete variable, $f(y_j^D|z^{(1)D})$ belongs to the exponential family distributions. In our simulations, the Bernoulli distribution has been used for binary variables, the binomial distribution for count variables and the ordered multinomial distribution for ordinal data. The exact expressions of the densities by data type can be found in Cagnone and Viroli (2014) [7].

In order to train the GLLVM layer, one maximizes

$$\begin{aligned} &E_{z^{(1)D}, s^D, s^{(\underline{L}+1:)}|y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}}[\log L(y^D|z^{(1)D}, s'^D, s'^{\underline{L}+1:}, \Theta_D, \Theta_{\underline{L}+1:})] \\ &= E_{z^{(1)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}}[\log L(y^D|z^{(1)D}, \Theta_D, \Theta_{\underline{L}+1:})] \\ &= \int f(z^{(1)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \log L(y^D|z^{(1)D}, \Theta_D, \Theta_{\underline{L}+1:}) dz^{(1)D} \end{aligned}$$

as y^D is related to $(s^D, s^{(\underline{L}+1:)})$ only through $z^{(1)D}$. Note that s'^h is simply a realisation of s^h.

3.1.1 MC Step

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

3.1.2 E step

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

$$f(z^{(1)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) = \sum_{s'^D} f(z^{(1)D}|y^D, s'^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) f(s^{(1D:L)} = s'|y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \tag{5}$$

Bayes rule for the first term gives :

$$f(z^{(1)D}|y^D, s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) = \frac{f(z^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) f(y|z^{(1)D}, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})}{f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})} \tag{6}$$

As in the DGMM paper, $(z^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \sim N(\mu_{s'}^{(1D:L)}, \Sigma_{s'}^{(1D:L)})$, with $(\mu_{s'}^{(1D:L)}, \Sigma_{s'}^{(1D:L)})$ the mean and covariance parameters detailed in Appendix B.

$f(y^D|z^{(1)D}, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})$ is a distribution from the exponential family as stated in (4). Finally, $f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})$ has to be numerically approached, by Monte Carlo estimation in our case:

$$\begin{aligned} f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) &= \int_{z^{(1)D}} f(y^D|z^{(1)D}, s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) f(z^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) dz^{(1)D} \\ &= \int_{z^{(1)D}} f(y^D|z^{(1)D}, \hat{\Theta}) f(z^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) dz^{(1)D} \\ &\approx \sum_{m=1}^{M^{(1)}} f(y^D|z_m^{(1)D}, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}, \hat{\Theta}) f(z_m^{(1)D}|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \end{aligned}$$

The second term can be written as a posterior density:

$$f(s^{(1D:L)} = s'|y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) = \frac{f(s^{(1D:L)} = s'|\hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) f(y^D|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})}{\sum_{s''} f(s^{(1D:L)} = s''|\hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) f(y^D|s^{(1D:L)} = s'', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})} \tag{7}$$

$(s^{(1D:L)}|\hat{\Theta}) \sim M(\pi_s^{(1D:L)})$ is a multinomial distribution of parameters $\pi_s^{(1D:L)}$ the probability of a full path through the network starting from the discrete head and $f(y|s', \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})$ that is once again approximated by Monte Carlo.

3.1.3 M step

There are no closed form solutions for the estimators of (λ_0, Λ_0) that maximize $E_{z^{(1)D}|y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1}}[\log L(y^D|z^{(1)D}, \Theta_D, \hat{\Theta}_{\underline{L}+1})]$. Hence, one has to use optimisation methods in order to compute those estimations. The methods used here all belong to the Python scipy.optimize package.

For categorical variables, the optimisation program is unconstrained (except to match the identifiability constraints detailed in section 4). The algorithm used is BFGS [10] and the gradients is computed with autograd.

Concerning ordinal variables, the optimisation program is constrained as the intercept coefficients have to be ordered. The method used is a trust-region algorithm [8] coupled once again with autograd.

3.2 Determining the parameters of the DGMM layers

In this section, we omit the subscript $h \in \{C, D\}$ on z^h and y^h variables to make notations more readable. In this section only, s' refers here to a full path (including common layers) and not only path going through one head.
For $l \in [1, \underline{L}]$, we aim to maximize

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

The conditional distribution under which the expectation is taken is quite complex given that it depends on variables located in 3 different layers. McLachlan and Viroli (2019) make the assumption that:

$$\begin{aligned} &\arg\max_{\Theta} E_{z^{(l)}, z^{(l+1)}, s|y, \hat{\Theta}}[\log L(z^{(l)}|z^{(l+1)}, s, \Theta)] \\ &\approx \arg\max_{\Theta} E_{z^{(l+1)}|z^{(l)}, s, \hat{\Theta}}[\log L(z^{(l)}|z^{(l+1)}, s, \Theta)] \end{aligned}$$

Doing so, we loose the conditioning over y which may be very problematic as it is the most important information bridge between discrete information and the following continuous layers (even if the embedding coefficients, Θ_{emb}, also account for a part of that link).
Thus, we are not making this assumption here and keep the full conditional distribution of the expectation.

3.2.1 MC Step

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

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

3.2.2 E Step

The conditional expectation distribution has the following decomposition:

$$\begin{aligned} f(z^{(l)}, z^{(l+1)}, s = s'|y, \hat{\Theta}) &= f(z^{(l)}, s = s'|y, \hat{\Theta}) f(z^{(l+1)}|z^{(l)}, s', y, \hat{\Theta}) \\ &= f(z^{(l)}|y, s', \hat{\Theta}) f(s = s'|y, \hat{\Theta}) f(z^{(l+1)}|z^{(l)}, s', \hat{\Theta}) \end{aligned} \tag{8}$$

The first term can be rewritten as :

$$\begin{aligned} f(z^{(l)}|y, s', \hat{\Theta}) &= \int_{z^{(l-1)}} f(z^{(l)}|z^{(l-1)}, s', \hat{\Theta}) f(z^{(l-1)}|y, s', \hat{\Theta}) dz^{(l-1)} \\ &\approx \sum_{m=1}^{M^{(l-1)}} f(z^{(l)}|z_m^{(l-1)}, s', \hat{\Theta}) f(z_m^{(l-1)}|y, s', \hat{\Theta}) \end{aligned} \tag{9}$$

This term is hence calculable in a recurrent manner $\forall l \in [2, \underline{L}]$, starting with $f(z^{(1)}|y, s', \hat{\Theta})$ which expression is given in in (6).

The second term is detailed in (7).
Finally, the last term corresponds to equation (7) of the DGMM paper.

By Bayes rule:

$$f(z^{(l+1)}|z^{(l)}, s', \hat{\Theta}) = \frac{f(z^{(l)}|z^{(l+1)}, s', \hat{\Theta}) f(z^{(l+1)}|s', \hat{\Theta})}{f(z^{(l)}|s', \hat{\Theta})} \tag{10}$$

The denominator does not depend on $z^{(l+1)}$ and is hence considered as a normalisation constant. Besides, we have that $f(z^{(l)}|z^{(l+1)}, s', \hat{\Theta}) = N(\eta_{k_l'}^{(l)} + \Lambda_{k_l'}^{(l)} z^{(l+1)}, \Psi_{k_l'}^{(l)})$.

Concerning $f(z^{(l+1)}|s', \hat{\Theta})$, as in the DGMM paper we are forward looking: the previous paths that have led to $z^{(l+1)}$ does not matter given the following paths. Hence the conditioning in previous $s'^{(1:l)}$ can be removed.

$$\begin{aligned} f(z^{(l+1)}|s = s', \hat{\Theta}) &= f(z^{(l+1)}|s'^{(l+1:L)}, \hat{\Theta}) \\ &= N(\mu_{s'^{(:k_{l+1}:)}}^{(l+1)}, \Sigma_{s'^{(:k_{l+1}:)}}^{(l+1)}) \end{aligned}$$

With the mean and covariance parameters can be computed as in Appendix B.

Hence, one can show as in equation (8) of Viroli and McLachlan (2019) [28] that $(z^{(l+1)}|z^{(l)}, s', \hat{\Theta}) \sim N(\rho_{k_{l+1}}^{(l+1)}, \xi_{k_{l+1}}^{(l+1)})$, with:

$$\rho_{k_{l+1}}^{(l+1)} = \xi_{k_{l+1}}^{(l+1)}\Big(\Lambda_{k_{l+1}}^{(l+1)T}(\Psi_{k_{l+1}}^{(l+1)})^{-1}(z^{(l)} - \eta_{k_{l+1}}^{(l+1)}) + \Sigma_{s'^{(:k_{l+1}:)}}^{(l+1)}\mu_{s'^{(:k_{l+1}:)}}^{(l+1)}\Big)$$

and

$$\xi_{k_{l+1}}^{(l+1)} = \Big(\Sigma_{s'^{(:k_{l+1}:)}}^{(l+1)} + \Lambda_{k_{l+1}}^{(l+1)T}(\Psi_{k_{l+1}}^{(l+1)})^{-1}\Lambda_{k_{l+1}}^{(l+1)}\Big)^{-1}$$

3.2.3 M step

The estimators of the DGMM layer parameters $\forall l \in [1, L-1]$ are given by :

$$\begin{cases} \hat{\eta}_{k_l}^{(l)} = \dfrac{\sum_{i=1}^n \sum_{s'^{(:k_l:)}_i} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y, \hat{\Theta})\left[E[z_i^{(l)}|s_i^{(:k_l:)} = s_i'^{(:k_l:)}, y_i, \hat{\Theta}] - \Lambda_{k_l}^{(l)} E[z_i^{(l+1)}|s_i'^{(:k_l:)}, y_i, \hat{\Theta}]\right]}{\sum_{i=1}^n \sum_{s'^{(:k_l:)}_i} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y_i, \hat{\Theta})} \\ \hat{\Lambda}_{k_l}^{(l)} = \dfrac{\sum_{i=1}^n \sum_{s'^{(:k_l:)}_i} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y_i, \hat{\Theta})\left[E[(z_i^{(l)} - \hat{\eta}_{k_l}^{(l)}) z_i^{(l+1)T}|s_i^{(:k_l:)} = s_i'^{(:k_l:)}, y_i, \hat{\Theta}]\right]}{\sum_{i=1}^n \sum_{s'^{(:k_l:)}_i} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y_i, \hat{\Theta})} E[z_i^{(l+1)} z_i^{(l+1)T}|s_i'^{(:k_l:)}, y_i, \hat{\Theta}]^{-1} \\ \hat{\Psi}_{k_l}^{(l)} = \dfrac{\sum_{i=1}^n \sum_{s'_i} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y_i, \hat{\Theta}) E\left[\left(z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right)\left(z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right)^T |s_i'^{(:k_l:)}, y_i, \hat{\Theta}\right]}{\sum_{i=1}^n \sum_{s'^{(:k_l:)}_i} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y_i, \hat{\Theta})} \end{cases}$$

With $s_i'^{(:k_l:)} = (k_1', ..., k_{l-1}', k_l, k_{l+1}', ..., k_L')$, a path going through 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 $\forall l \in [\underline{L}+1, L]$,

$$E_{z^{(l)}, z^{(l+1)}, s^C, s^D, s^{(\underline{L}+1:)}|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}}[\log L(z^{(l)}|z^{(l+1)}, s^C, s^D, s^{(\underline{L}+1:)}, \Theta_C, \Theta_D, \Theta_{(\underline{L}+1:)})]$$

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 and not only discrete or continuous data as in the heads layers.

3.3.2 E Step

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

$$\begin{aligned} f(z^{(l)}, z^{(l+1)}, s^C, s^D, & s^{(\underline{L}+1:)}|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \\ &= f(z^{(l)}|s^C, s^D, s^{(\underline{L}+1:)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \\ &\times f(z^{(l+1)}|z^{(l)}, s^{(\underline{L}+1:)}, \hat{\Theta}_{\underline{L}+1:}) \\ &\times f(s^C, s^D, s^{(\underline{L}+1:)}|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \end{aligned} \tag{11}$$

$\forall l \in [\underline{L}+2, L]$:

$$\begin{aligned} & f(z^{(l)}|s^C, s^D, s^{(\underline{L}+1:)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \\ &= \int_{z^{(l-1)}} f(z^{(l)}|z^{(l-1)}, s^C, s^D, s^{(\underline{L}+1:)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) f(z^{(l-1)}|s^C, s^D, s^{(\underline{L}+1:)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) dz^{(l-1)} \end{aligned}$$

Hence the first term can be computed in a recurrent manner as in (9) starting from layer $\underline{L}+1$. The recurrent relationship start with the density of layer $\underline{L}+1$, which can be calculated in the following way:

$$
\begin{aligned}
&f(z^{(\underline{L}+1)}|s^C, s^D, s^{(\underline{L}+1:)}, y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \\
&\propto f(z^{(\underline{L}+1)}, y^C|s^C, s^D, s^{(\underline{L}+1:)}, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \\
&= f(y^C|z^{(\underline{L}+1)}, s^C, s^{(\underline{L}+1:)}, \hat{\Theta}_C, \hat{\Theta}_{\underline{L}+1:}) f(z^{(\underline{L}+1)}|s^C, s^D, s^{(\underline{L}+1:)}, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}) \\
&= f(y^C|z^{(\underline{L}+1)}, s^C, s^{(\underline{L}+1:)}, \hat{\Theta}_C, \hat{\Theta}_{\underline{L}+1:}) f(z^{(\underline{L}+1)}|s^D, s^{(\underline{L}+1:)}, y^D, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:})
\end{aligned}
$$

Where the last equality comes from the fact that $z^{\underline{L}+1}$ is linked to y^c only through s^C and not s^D.
The first density is simply a Gaussian of parameters $(\mu^{(1C:\underline{L}+1)}, \Sigma^{(1C:\underline{L}+1)})$ [ABUSIVE NOTATION] and the second density is calculable as in (9).

Finally the second term of (11) can be computed as in (10) and the third term computation is 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.

$$E_{s^C, s^D, s^{(\underline{L}+1)}|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}}[\log L(s^C, s^D, s^{(\underline{L}+1:)}, \Theta_C, \Theta_D, \Theta_{\underline{L}+1:})]$$

with respect to $\pi_s^h \ \forall h \in \{C, D\}$ and $\pi_s^{(\underline{L}+1:)}$.

In order to be the most precise as possible, there are actually three different multinomial distributions to be distinguished.
The full paths going from one head to the end of the common tail has distribution $p(s^h, s^{(\underline{L}+1:)}|y^h, \Theta_h, \Theta_{L+1:})$ of parameter $\pi_s^{(1:L)h}$ with support in Ω^h.
The paths going from one head to the first common layer where the end of the paths is known, has distribution $p(s^h|y^h, \Theta_h, \Theta_{L+1:})$. The parameters of this distribution will be denoted $\tilde{\pi}_s^h$ and are of cardinal $S - \prod_{l=\underline{L}+1}^{L} K_l^h$.
Finally at the layer level, the probability of the component k_l to be reached by a path is also a multinomial distribution of parameters $\pi_{k_l}^{(l)}$ of cardinal K_l.

3.4.1 E step

$f(s^h, s^{(\underline{L}+1:)}|y^j, \hat{\Theta}_h, \hat{\Theta}_{(\underline{L}+1:)})$ is given in (7). Note that $f(y^h|s^h, s^{(\underline{L}+1:)}, \hat{\Theta}_j, \hat{\Theta}_{(\underline{L}+1:)})$ has to be approximated for the discrete head but is directly available as a Gaussian distribution of parameters $(\mu^{(1C:L)}, \Sigma^{(1C:L)})$ for the continuous head.

3.4.2 M step

The estimator of the path probabilities starting from both layers is maximizing

$$E_{s^C, s^D, s^{(\underline{L}+1:)}|y^C, y^D, \hat{\Theta}_C, \hat{\Theta}_D, \hat{\Theta}_{\underline{L}+1:}}[\log L(s^C, s^D, s^{(\underline{L}+1:)}|\Theta_C, \Theta_D, \Theta_{\underline{L}+1})]$$

$$
\begin{aligned}
E_{s^C,s^D,s^{(\underline{L}+1:)}|y^C,y^D,\hat{\Theta}_C,\hat{\Theta}_D,\hat{\Theta}_{\underline{L}+1:}}[\log L(s^C,s^D,s^{(\underline{L}+1:)}|\Theta_C,\Theta_D,\Theta_{\underline{L}+1})] \\
= E_{s^C,s^{(\underline{L}+1:)}|y^C,\hat{\Theta}_C,\hat{\Theta}_{\underline{L}+1:}}[\log L(s^C,s^{(\underline{L}+1:)}|\Theta_C,\Theta_{\underline{L}+1})] \\
+ E_{s^D,s^{(\underline{L}+1:)}|y^D,\hat{\Theta}_D,\hat{\Theta}_{\underline{L}+1:}}[\log L(s^D|s^{(\underline{L}+1:)},\Theta_D,\Theta_{\underline{L}+1})]
\end{aligned}
$$

[Give more calculus explanations]

Following the calculus given in Appendix A.2, one ends up with the estimator of the path probability $\pi_s^{(1C:L)}$ for the continuous head for a given path $s'^{(1C:L)}$ being

$$\hat{\pi}_{s'}^{(1C:L)} = \frac{\sum_{i=1}^n f(s_i'^{(1C:L)}|y_i^C,\hat{\Theta}_C,\hat{\Theta}_{\underline{L}+1:})}{n}$$

and for the path starting from the discrete head with the ending common layers known :

$$\hat{\hat{\pi}}_{s'}^{D} = \frac{\sum_{i=1}^n f(s_i'^{(1:k'_{\underline{L}+1},k'_{\underline{L}+2},\ldots,k'_L)D},|y_i^D,\hat{\Theta}_D,\hat{\Theta}_{\underline{L}+1:})}{n}$$

[Provide calculus details for these estmators]

Which has to be read as the mean of all the probabilities starting from the discrete head and finishing by the known $k_{\underline{L}+1},...,k_L$ components of the common layers.

From this path probabilities it is then direct to compute the probability of each component of each layer. The probability of a component is indeed simply the sum of the probabilities of the paths going through this component.

For instance, $\pi_{k_l}^{(l)}$ the probability of the component k_l of layer l is simply:

$$\pi_{k_l}^{(l)} = \sum_{s'^{(:k_l:)}} \pi_{s'^{(:k_l:)}}$$

4 Identifiability

In this section, we make the assumption that the identifiability of the model is guaranteed by enforcing both GLLVM identifiability constraints and DGMM constraints. Further work should give proof that this union of constraints is sufficient.

4.1 GLLVM identifiability constraints

The GLLVM model assumes that the latent variable is centered and of unit variance. We define $z^{(1)Dnew}$ the rescaled version of $z^{(1)D}$ satisfying those constraints. For the continuous head, no operations are needed in this respect.

This can be obtained by rescaling the parameters of the first layer at each EM step in the following way:

$$
\begin{cases}
\eta_{k'_1}^{(1)Dnew} = A^{-1T}\left[\eta_{k'_1}^{(1)D} - \sum_{s''}\pi_{s''}\mu_{s''}\right] \\
\Lambda_{k'_1}^{(1)Dnew} = A^{-1T}\Lambda_{k'_1}^{(1)D} \\
\Psi_{k'_1}^{(1)Dnew} = A^{-1T}\Psi_{k'_1}^{(1)D}A^{-1}
\end{cases}
$$

with $A = Var(z^{(1)D})$.
The proof is given in Appendix B.1.

As $z^{(1)D}$ has been rescaled, the coefficients of $\Lambda^{(0)}$ have also to be rescaled: $\Lambda^{(0)new} = \Lambda^{(0)} A^{-1T}$.

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) [7] 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.* $p < r_1^h < ... < r_L$. Secondly, we make the assumption that $\Lambda_{k_l}^{(l)} \Psi_{k_l}^{(l)} \Lambda_{k_l}^{(l)T}$ is diagonal with elements in decreasing order $\forall l \in [1, L]$. In practice for now, once all parameters have been estimated by the MCEM algorithm, we apply the following transformation:

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

A more rigourous way would be to add explicitly this diagonal constraint into the Maximisation program of the M Step of the MCEM algorithm and derive the associated estimators. This should be done in further research.

[to rewrite:] Finally, Fruehwirth-Schnatter and al. (2018) [12] give sufficient conditions for MFA identifiability that could unfortunately not be applied here. One of them is indeed the so-called Anderson-Rubin condition (AR condition) that would imply in our case that $r_{l-1} \leq \frac{r_l - 1}{2}$. Enforcing this condition would prevent from using MDGMMs unless the network is composed of at least 7 variables of each type which is far too restrictive. As a result we choose to use the set of the presented necessary conditions without explicitly prove that they are sufficient, bearing in mind that it is a limit to our work.

5 Model selection

The selection of the MDGMM best architecture is performed using the pruning methodology which is very used for supervised neural networks [6] but also for tree-based methods [25]. 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_l, the best number of factors r_l 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 [2] or BIC [27]. 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 and provide extensive details in Appendix C.

Our pruning strategy determines the best number of component on each layer k_l^h by deleting the components associated with very low probabilities $\pi_{k_l}^{(l)h}$ as they are the least likely to explain the data.

The choice of the latent dimensions of each layer r_l^h is performed by looking at the dimensions that carry the more 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.
The selection is conducted differently for the GLLVM layer compared to the regular DGMM layers.
For the GLLVM layer, we have performed logistic regressions [4] of y^C over $z^{(1)C}$ and delete the dimensions that were associated with zero coefficients or coefficients that were not significant in a vast majority of paths.
Concerning the regular DGMM layers, information carried by the current layer given the previous layer and the paths has been modeled using a Principal Component Analysis [15]. 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_l. If layer l as dimension 1, then according to the identifiability constraint $p^h < r_1^h < r_l^h < ... < r_L$, the following layers are deleted.

These operations are performed during the first two iterations of the algorithm as the first iterations are determinant for EM-based algorithms, then periodically through the iterations. Once the optimal specification have been determined, the model is refitted with this specification as architecture changing through the training seemed to lead to poorer final results in our simulations.

6 Real Applications

EM-based algorithms are known to be very sensitive to its initialisation points (source). In our case, using purely random initialization as in Cagnone and Viroli (2014) [7] made the model diverge most of the time, especially 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 $(\eta_{k_l}^{(l)h}, \Lambda_{k_l}^{(l)h}, \Psi_{k_l}^{(l)h}, s^{(l)h}, z^{(l)h})_{k_l,l,h}$ and the exponential family links parameters $(\lambda^{(0)}, \Lambda^{(0)})$ seems not to be a good practice. This problem might get even worse as the number DGMM layers grow.

For discrete head initialisation, the idea used here is to perform a Multiple Correspondence Analysis (MCA) [22] 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 consider all variables as categorical, thus the more the dataset actually contains this type of variables the better the initialisation.
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 $(\eta_{k_l}^{(l)}, \Lambda_{k_l}^{(l)}, \Psi_{k_l}^{(l)}, \pi_{k_l}^{(l)})$ and the latent variable of the following layer $z^{(l+1)}$.
Concerning the GLLVM parameters, logistic regressions of y_j^h over $z^{(1)h}$ are fitted for each original variable of each 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 last layers of both head and of $z^{(\underline{L}+1)}$. Indeed, $z^{(\underline{L}+1)}$ 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, $z^{(\underline{L}+1)}$ has been determined by applying a PCA over the stacked variable $(z^{(\underline{L})D}, z^{(\underline{L})D})$. Then the DGMM coefficients $(\eta_{k_{\underline{L}}}^{(\underline{L})h}, \Lambda_{k_{\underline{L}}}^{(\underline{L})h}, \Psi_{k_{\underline{L}}}^{(\underline{L})}h)$ of each head have been separately determined using Partial Least Square (PLS) [30].

6.1 Models compared

In order to illustrate the performance of our algorithm, several models are fitted to data. As our model is on the first hand a generalisation of the model of Viroli and Cagnone (2014) [7] for discrete data only, we also provide results on discrete data only with a one discrete head-architecture of our model that will be denoted by Discrete Data DGMM (DDGMM). Then, we give results for our complete two-heads specification on mixed datasets. Note that we do not give results of the MDGMM fitted only on continuous data as it is actually the DGMM itself evaluated by Viroli and McLachlan (2019) [28].

As evoked in introduction, Ahmad and al. (2018) [1] have proposed a typology of clustering algorithms which is composed of four main families. In order to benchmark our models performance, they will be compared to algorithms coming from the other four families. k-modes and k-Prototypes [16][17] represent the partitional algorithms, Agglomerative Clustering [26] is part of the hierarchical algorithms, Self-Organising Maps (SOM) [18] account for Neural-Network Based approaches. Ahmad and al. actually define a last family with models unclassifiable in the four main families. In order to consider also one of this model we also use DBSCAN [9] has a benchmark for this last type of algorithm.

In addition to these models, two other models have also been adjusted when benchmarking the one-head discrete specification: A GMM fitted on the MCA-transformed data, which is the model used to initialise our model. The idea is to verify that our model provide an additional clustering power compared to its initialisation point.
Finally, we report the performance of the algorithm by Cagnone and Viroli [7] (2014) in order to assess the additional clustering power provided when the network is made deeper. In the results the model is named GLMLVM for Generalized Linear Mixture of Latent Variables Model.

In order for the results to be reproducible, all of the specifications used for each model are given in Appendix D.

6.2 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 Abalone Datasets.

6.2.1 Discrete data presentation

The breast cancer dataset is a dataset of 286 observations and 9 discrete variables. Most of the variables are ordinal, which might undermine the quality of the initialisation by MCA.

The Tic Tac Toe dataset is composed of 9 variables corresponding to each cell of a $3x3$ grid of tic-tac-toe. The dataset present 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 Mushroom 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-ht encoded, the final dataset has more than 70 variables and is hence a highly dimensional dataset. The relatively large size and high-dimensionaly of this dataset aims at proving the scalability of our algorithm.

6.2.2 Mixed data presentation

The Australian credit dataset is a binary classification dataset concerning credit cards. It is composed of 690 observations, 8 discrete categorical variables and 6 numeric variables. It is a small dataset with a high dimension. The few missing data existing in the dataset have been removed [add the number of NAs].

The heart (Stalog) dataset is composed of 270 observations composed by five continuous variables, three categorical variables, three binary variables and two ordinal variables. This dataset designed for binary classification has hence diverse data types among its variables.

The abalone dataset presents eight attributes to predict the age of 4,177 abalones that are small Sea Water molluscs. They are six continuous attributes, one binary variable and one count variable. On the contrary of the other datasets that have been designed for binary classification, the number of classes is here of 29 as the ages to predict range from 1 to 29 years old.

6.3 Results

For each dataset, we have set the number of unsupervised clusters to the true number. Each model specification has been run 30 times and the results reported correspond to the specification that has given the best results for each model.

We use here three supervised metrics to assess the clustering quality: the micro precision, the macro precision and the cluster purity measure. 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. Finally, the cluster purity [19] measures how much the ground-truth labelling and predicted labelling overlap [be more precise here].

6.3.1 Results on discrete data

Datasets	Breast Cancer			Tic Tac Toe dataset			Mushrooms dataset		
Algorithms / Metrics	Micro	Macro	Purity	Micro	Macro	Purity	Micro	Macro	Purity
DDGMM	0.603 (0.096)	0.539 (0.106)	0.711 (0.007)	0.566 (0.043)	0.515 (0.048)	0.653 (0.000)	0.738 (0.116)	0.717 (0.243)	0.741 (0.112)
GLMLVM (random init)									
GLMLVM (new init)	0.729 (0.006)	0.661 (0.058)	0.729 (0.006)	0.582 (0.042)	0.542 (0.041)	**0.655** (0.006)	0.852 (0.002)	**0.904** (0.001)	0.852 (0.002)
MCA + GMM	0.686 (0.089)	0.645 (0.052)	0.728 (0.012)	0.604 (0.005)	0.599 (0.004)	0.653 (0.000)	0.822 (0.087)	0.870 (0.062)	0.830 (0.058)
k-Modes	0.592 (0.000)	0.534 (0.000)	0.708 (0.000)	0.610 (0.000)	0.586 (0.000)	0.653 (0.000)	0.852 (0.000)	0.898 (0.000)	0.852 (0.000)
k-Prototypes	0.730 (0.014)	0.667 (0.011)	0.731 (0.005)	∅(∅)	∅(∅)	∅(∅)	0.851 (0.000)	0.895 (0.001)	0.851 (0.000)
Hierarchical	**0.754** (0.000)	0.712 (0.000)	**0.754** (0.000)	**0.654** (0.000)	**0.827** (0.000)	0.654 (0.000)	**0.854** (0.000)	**0.904** (0.000)	**0.854** (0.000)
SOM	0.662 (0.066)	0.500 (0.126)	0.711 (0.008)	0.612 (0.028)	0.488 (0.051)	0.653 (0.000)	0.853 (0.003)	0.902 (0.002)	0.853 (0.003)
DBSCAN	0.726 (0.009)	**0.860** (0.253)	0.726 (0.009)	0.653 (0.000)	0.327 (0.000)	0.653 (0.000)	0.618 (0.000)	0.310 (0.000)	0.618 (0.000)

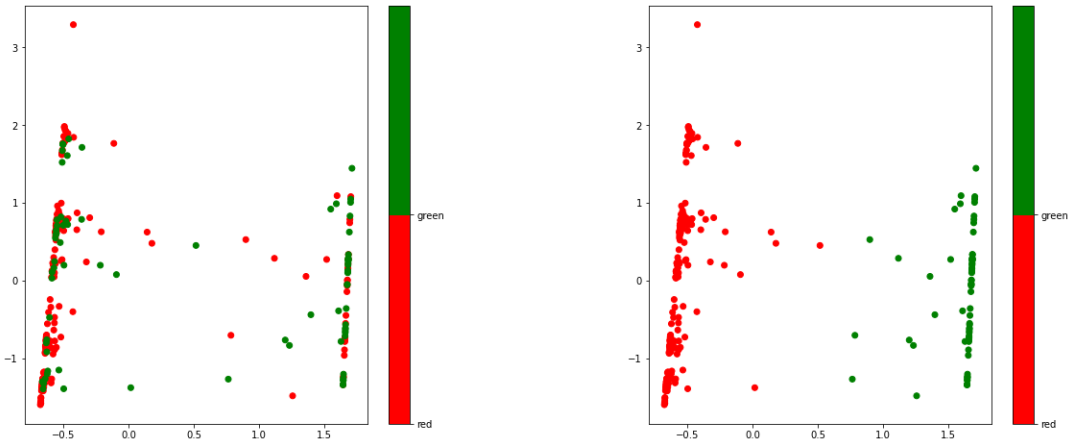
[Last macro std of DBSCAN is weird...],[DDGM VERY VARIABLE ON MUSHROOMS...]

6.3.2 Results on mixed data

BENCHMARK FAMD + GMM ON CONTINUOUS

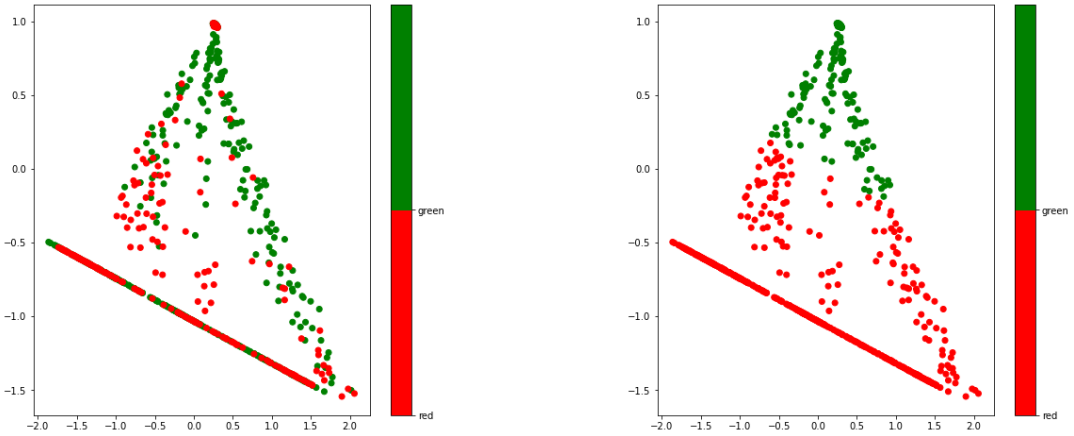
6.3.3 Results visualisation

This section presents some of the continuous representation of the variables that one could obtain after running the MDGMM. [change the color legends on the figure].



(a) Actual labels (b) Predicted labels

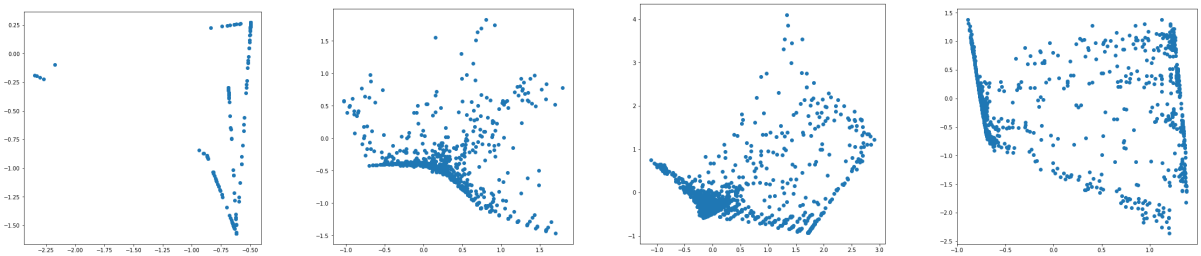
Figure 2: Continuous representation of the breast cancer dataset



(a) Actual labels (b) Predicted labels

Figure 3: Continuous representation of the Tic Tac Toe cancer dataset

Finally, we illustrate the training of our algorithm with successive 2D-representations existing through the iterations.



(a) Training at t1 (b) Training at t2 (c) Training at t3 (d) Training at t4

Figure 4: Continuous representation of the Tic Tac Toe dataset

7 Discussion

In this work, we have chosen to approximate unknown expectations and densities using Monte Carlo simulations. This choice was motivated more by the fact that it enabled to have a homogeneous methodology to train all the layers rather than by performance concerns. Cagnone and Viroli [7] have rather used Gauss-Hermite approximations which also performed well at least in low dimensions. Alternative methods such as low discrepancy series have not been tried here. Variational rewritting of our model seem also to be a promising research path. Indeed, it should make identification requirements easier to meet, as one can keep only the posterior draws that meet the requirements, a possibility that does not exist in our frequentist framework. One could extend for instance extend Niku & al. (2019) work [23]. Indeed, the authors propose to rewrite the GLLVM in a variational fashion and demonstrate the running time and accuracy gains of this solution.

In our Monte Carlo simulations, one of the very important parameters of the algorithm is the number of Monte Carlo copies $M^{(l)}h$ to draw at each layer.
Wei and Tanner (1990) [29] advise to let M grow through the iteration 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.
Hence, in our simulations, we let M^l grow linearly with the number of iterations t. Typically, we advise to take $M^{lt} = a \times t \times r_l$ with $a \in [1, 2]$. The effect of other growth patterns could be investigate in further research.

Classically, the model is stopped once the log-likelihood does not increase anymore. The clustering associated with the best likelihood is chosen as the final one. Yet, due to the additional error brought by the MC steps of the MCEM algorithm, one does not have the guarantee that the log-likelihood is strictly increasing through the iterations anymore. 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 2.

We have here made the choice to take a GLLVM model to encode discrete features into a continuous space. It appears to us that the flexibility in the design of the latent space coupled with the easily interpretable coefficients of the link functions made it an interesting choice for discrete data encoding. However, this very simple and parametric link might also act as an information bottleneck, destroying the complexity of the initial data and making additional latent layers less useful. Moreover, the optimization steps needed to compute the link coefficients is the slowest part of the algorithm and adds an additional degree of error. Finally, assuming the independence between the discrete variable is a very strong assumption which left an important part of the information brought by the data out.
In this regard, other encoding mechanisms 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 as here. However as mentioned, one has to assess if it worths the additional computational burden.

One can notice that the "clustering link" has here been chosen to be the one existing between $z^{(1)}$ and $z^{(2)}$ as one can see on Figure 1. Another choice could have been to write the model as a mixture of GLLVM. However, fitting a GLLVM is very computational intensive due to the optimization step it involves. Fitting a mixture of GLLVM could then have led to a too expensive computing time.

The training of our model can also be modified by considering for instance training the model with gradient-based techniques rather that MCEM methods. The EM-related algorithms are indeed very sensitive to the initialisation, which was in our case particularly tricky given the size of the parameter space. Combining Multiple Correspondance Analysis (MCA) [22] or Factor Analysis for Mixed Data (FAMD) [24] with Gaussian Mixture Models (GMM) has however enables us to reduce dramatically the number of times where the model diverged and seems to be a good starting point in this respect.

One of the main advantage of the MDGMM is that despite its complexity, it provides simultaneously (in one model fit) several continuous representations of the data of different dimensions. Indeed, every layer has

a different dimension and is a representation of the data that could bring useful insights once plotted. Going further, the model enables to perform several clusterings with different numbers of clusters at the same time. For instance, if the true number clusters is assumed to be between 2 and 4 one can compare the clusters obtained on the layer with 4 components, 3 components and 2 components and pick the one that makes the more sense. Hence, using the first common layer as clustering layer is a natural choice, but nothing prevents one from using one of the following common layers or even the set of all possible paths through the common layers as clustering mechanism.

8 Acknowledgments

Thanks to the LIA LYSM (agreement between AMU, CNRS, ECM and INdAM).

References

[1] Amir Ahmad and Shehroz S Khan. "Survey of state-of-the-art mixed data clustering algorithms". In: *IEEE Access* 7 (2019), pp. 31883–31902.

[2] Hirotogu Akaike. "Information theory and an extension of the maximum likelihood principle". In: *Selected papers of hirotugu akaike*. Springer, 1998, pp. 199–213.

[3] Atılım Günes Baydin et al. "Automatic differentiation in machine learning: a survey". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 5595–5637.

[4] Joseph Berkson. "Why I prefer logits to probits". In: *Biometrics* 7.4 (1951), pp. 327–339.

[5] Jesse Bettencourt, Matthew J Johnson, and David Duvenaud. "Taylor-Mode Automatic Differentiation for Higher-Order Derivatives in JAX". In: (2019).

[6] Davis Blalock et al. "What is the state of neural network pruning?" In: *arXiv preprint arXiv:2003.03033* (2020).

[7] Silvia Cagnone and Cinzia Viroli. "A factor mixture model for analyzing heterogeneity and cognitive structure of dementia". In: *AStA Advances in Statistical Analysis* 98.1 (2014), pp. 1–20.

[8] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. *Trust region methods*. Vol. 1. Siam, 2000.

[9] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[10] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[11] Chris Fraley and Adrian E Raftery. "Model-based clustering, discriminant analysis, and density estimation". In: *Journal of the American statistical Association* 97.458 (2002), pp. 611–631.

[12] Sylvia Fruehwirth-Schnatter and Hedibert Freitas Lopes. "Sparse Bayesian factor analysis when the number of factors is unknown". In: *arXiv preprint arXiv:1804.04231* (2018).

[13] Zoubin Ghahramani, Geoffrey E Hinton, et al. *The EM algorithm for mixtures of factor analyzers*. Tech. rep. Technical Report CRG-TR-96-1, University of Toronto, 1996.

[14] Harry H Harman. *Modern factor analysis*. University of Chicago press, 1976.

[15] Harold Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6 (1933), p. 417.

[16] Zhexue Huang. "Clustering large data sets with mixed numeric and categorical values". In: 1997.

[17] Zhexue Huang. "Extensions to the k-means algorithm for clustering large data sets with categorical values". In: *Data mining and knowledge discovery* 2.3 (1998), pp. 283–304.

[18] Teuvo Kohonen. "The self-organizing map". In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[19] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[20] Irini Moustaki. "A general class of latent variable models for ordinal manifest variables with covariate effects on the manifest and latent variables". In: *British Journal of Mathematical and Statistical Psychology* 56.2 (2003), pp. 337–357.

[21] Irini Moustaki and Martin Knott. "Generalized latent trait models". In: *Psychometrika* 65.3 (2000), pp. 391–411.

[22] Oleg Nenadic and Michael Greenacre. "Computation of multiple correspondence analysis, with code in R". In: (2005).

[23] Jenni Niku et al. "Efficient estimation of generalized linear latent variable models". In: *PloS one* 14.5 (2019).

[24] Jérôme Pagès. "Analyse factorielle de donnees mixtes: principe et exemple d'application". In: *Montpellier SupAgro, http://www. agro-montpellier. fr/sfds/CD/textes/pages1. pdf* (2004).

[25] Dipti D Patil, VM Wadhai, and JA Gokhale. "Evaluation of decision tree pruning algorithms for complexity and classification accuracy". In: *International Journal of Computer Applications* 11.2 (2010), pp. 23–30.

[26] G Philip and BS Ottaway. "Mixed data cluster analysis: an illustration using Cypriot hooked-tang weapons". In: *Archaeometry* 25.2 (1983), pp. 119–133.

[27] Gideon Schwarz et al. "Estimating the dimension of a model". In: *The annals of statistics* 6.2 (1978), pp. 461–464.

[28] Cinzia Viroli and Geoffrey J McLachlan. "Deep gaussian mixture models". In: *Statistics and Computing* 29.1 (2019), pp. 43–51.

[29] Greg CG Wei and Martin A Tanner. "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms". In: *Journal of the American statistical Association* 85.411 (1990), pp. 699–704.

[30] Svante Wold, Michael Sjöström, and Lennart Eriksson. "PLS-regression: a basic tool of chemometrics". In: *Chemometrics and intelligent laboratory systems* 58.2 (2001), pp. 109–130.

A Appendix

A.1 Calculus details of the estimators of the DGMM layers

We now turn on to the log-likelihood expression and gives the estimators of the l-th DGMM layer parameters $\forall l \in [1, L-1]$.

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

The derivatives of this quantity with respect to $\eta_{k_l}^{(l)}, \Lambda_{k_l}^{(l)}, \Psi_{k_l}^{(l)}$ are :

$$\begin{cases} \frac{\partial \log L(z_i^{(l)}|z_i^{(l+1)}, s_i, \Theta)}{\partial \eta_{k_l}^{(l)}} = \Psi_{k_l}^{(l)-1}\left(z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right) \\ \frac{\partial \log L(z_i^{(l)}|z_i^{(l+1)}, s_i, \Theta)}{\partial \Lambda_{k_l}^{(l)}} = \Psi_{k_l}^{(l)-1}\left(z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right) z_i^{(l+1)T} \\ \frac{\partial \log L(z_i^{(l)}|z_i^{(l+1)}, s_i, \Theta)}{\partial \Psi_{k_l}^{(l)}} = -\frac{1}{2}\Psi_{k_l}^{(l)-1}\left[I_{r_1} - \left(z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right)\left(z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right)^T \Psi_{k_l}^{(l)-1}\right] \end{cases}$$

Taking the expectation of the derivative with respect to $\eta_{k_l}^{(l)}$ and equalizing it to zero, it comes that :

$$E_{z^{(l)}, z^{(l+1)}, s|y, \hat{\Theta}}\left[\frac{\partial \log L(z^{(l)}|z^{(l+1)}, s, \Theta)}{\partial \eta_{k_l}^{(l)}}\right] = 0$$

$$\iff \Psi_{k_l}^{(l)-1} \sum_{i=1}^{n} E_{z_i^{(l)}, z_i^{(l+1)}, s_i|y_i, \hat{\Theta}}\left[z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right] = 0$$

$$\iff \sum_{i=1}^{n} E_{z_i^{(l)}, z_i^{(l+1)}, s_i|y_i, \hat{\Theta}}\left[z_i^{(l)} - (\eta_{k_l}^{(l)} + \Lambda_{k_l}^{(l)} z_i^{(l+1)})\right] = 0 \text{ because } \Psi_{k_l}^{(l)} \text{ is positive semi-definite}$$

$$\iff \sum_{i=1}^{n} \sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y_i, \hat{\Theta})\left[E_{z_i^{(l)}|s_i'^{(:k_l:)}, y_i, \hat{\Theta}}[z_i^{(l)}] - \eta_{k_l}^{(l)} - \Lambda_{k_l}^{(l)} E_{z_i^{(l+1)}|s_i'^{(:k_l:)}, y_i, \hat{\Theta}}[z_i^{(l+1)}]\right] = 0$$

Therefore, the estimator of $\eta_{k_l}^{(l)}$ is:

$$\hat{\eta}_{k_l}^{(l)} = \frac{\sum_{i=1}^{n} \sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y, \hat{\Theta})\left[E[z_i^{(l)}|s_i^{(:k_l:)} = s_i'^{(:k_l:)}, y_i, \hat{\Theta}] - \Lambda_{k_l}^{(l)} E[z_i^{(l+1)}|s_i'^{(:k_l:)}, y_i, \hat{\Theta}]\right]}{\sum_{i=1}^{n} \sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)} = s_i'^{(:k_l:)}|y_i, \hat{\Theta})}$$

with

$$E[z_i^{(l+1)}|s_i^{(:k_l:)} = s_i'^{(:k_l:)}, y_i, \hat{\Theta}] = \int_{z_i^{(l)}} f(z_i^{(l)}|s_i'^{(:k_l:)}, y_i, \hat{\Theta}) \int_{z_i^{(l+1)}} f(z_i^{(l+1)}|z_i^{(l)}, s_i'^{(:k_l:)}, \hat{\Theta}) z_i^{(l+1)} dz_i^{(l+1)} dz_i^{(l)}$$

$$\approx \sum_{m_l=1}^{M^{(l)}} f(z_{i,m_l}^{(l)}|s_i'^{(:k_l:)}, y_i, \hat{\Theta}) \sum_{m_{l+1}=1}^{M^{(l+1)}} z_{i,m_{l+1}}^{(l+1)}$$

Where $z_{i,m_{l+1}}^{(l+1)}$ has been drawn from $f(z_{i,m_{l+1}}^{(l+1)}|z_{i,m_l}^{(l)}, s)$.

Using the same reasoning for $\Lambda_{k_l}^{(l)}$:

$$E_{z^{(l)},z^{(l+1)},s|y,\hat{\Theta}}\left[\frac{\partial \log L(z^{(l)}|z^{(l+1)},s,\Theta)}{\partial \Lambda_{k_l}^{(l)}}\right]=0$$

$$\iff \Psi_{k_l}^{(l)-1}\sum_{i=1}^{n}\left[E_{z_i^{(l)},z_i^{(l+1)},s_i|y_i,\hat{\Theta}}[(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)}))z_i^{(l+1)T}]\right]=0$$

$$\iff \sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})\left[E_{z_i^{(l)},z_i^{(l+1)}|s_i'^{(:k_l:)},y_i,\hat{\Theta}}[(z_i^{(l)}-\eta_{k_l}^{(l)})z_i^{(l+1)T}]-\Lambda_{k_l}^{(l)}E_{z_i^{(l+1)}|s_i'^{(:k_l:)},y_i,\hat{\Theta}}[z_i^{(l+1)}z_i^{(l+1)T}]\right]=0$$

Hence the estimator of $\Lambda_{k_l}^{(l)}$ is :

$$\hat{\Lambda}_{k_l}^{(l)}=\frac{\sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})\left[E[(z_i^{(l)}-\hat{\eta}_{k_l}^{(l)})z_i^{(l+1)T}|s_i'^{(:k_l:)},y_i,\hat{\Theta}]\right]}{\sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})}E[z_i^{(l+1)}z_i^{(l+1)T}|s_i'^{(:k_l:)},y_i,\hat{\Theta}]^{-1}$$

with

$$\begin{aligned}E[(z_i^{(l)}-\hat{\eta}_{k_l}^{(l)})z_i^{(l+1)T}|s_i'^{(:k_l:)},y_i,\hat{\Theta}] &= \int_{z_i^{(l)}} f(z_i^{(l)}|s_i'^{(:k_l:)},y_i,\hat{\Theta})\int_{z_i^{(l+1)}} f(z_i^{(l+1)}|z_i^{(l)},s_i'^{(:k_l:)},\hat{\Theta})[(z_i^{(l)}-\hat{\eta}_{k_l}^{(l)})z_i^{(l+1)T}]dz_i^{(l+1)}dz_i^{(l)}\\ &\approx \sum_{m_l=1}^{M^{(l)}} f(z_{i,m_l}^{(l)}|s_i'^{(:k_l:)},y_i,\hat{\Theta})\sum_{m_{l+1}=1}^{M^{(l+1)}}[(z_{i,m_l}^{(l)}-\hat{\eta}_{k_l}^{(l)})z_{i,m_{l+1}}^{(l+1)T}]\end{aligned}$$

Where $z_{i,m_l}^{(l)}$ has been drawn from $f(z_{i,m_l}^{(l)}|s,\hat{\Theta})$ and $z_{i,m_{l+1}}^{(l+1)}$ from $f(z_{i,m_{l+1}}^{(l+1)}|z_{i,m_l}^{(l)},s,\hat{\Theta})$.

Finally, the last estimator:

$$E_{z^{(l)},z^{(l+1)},s|y,\hat{\Theta}}\left[\frac{\partial \log L(z^{(l)}|z^{(l+1)},s,\Theta)}{\partial \Psi_{k_l}^{(l)}}\right]=0$$

$$\iff -\frac{1}{2}\Psi_{k_l}^{(l)-1}\sum_{i=1}^{n}E_{z_i^{(l)},z_i^{(l+1)},s_i|,y_i,\hat{\Theta}}\left[I_{r_1}-\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)^T\Psi_{k_l}^{(l)-1}\right]=0$$

$$\iff \sum_{i=1}^{n}E_{z_i^{(l)},z_i^{(l+1)},s_i|y_i,\hat{\Theta}}\left[I_{r_1}-\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)^T\Psi_{k_l}^{(l)-1}\right]=0$$

$$\iff \sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})I_{r_1}=\sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})E_{z_i^{(l)},z_i^{(l+1)}|s_i'^{(:k_l:)},y_i,\hat{\Theta}}\left[\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)^T\right]$$

$$\iff \sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})\Psi_{k_l}^{(l)}=\sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})E_{z_i^{(l)},z_i^{(l+1)}|s_i'^{(:k_l:)},y_i,\hat{\Theta}}\left[\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)^T\right]$$

And hence the estimator of $\Psi_{k_l}^{(l)}$ is:

$$\hat{\Psi}_{k_l}^{(l)}=\frac{\sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})E\left[\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)\left(z_i^{(l)}-(\eta_{k_l}^{(l)}+\Lambda_{k_l}^{(l)}z_i^{(l+1)})\right)^T|s_i'^{(:k_l:)},y_i,\hat{\Theta}\right]}{\sum_{i=1}^{n}\sum_{s_i'^{(:k_l:)}} f(s_i^{(:k_l:)}=s_i'^{(:k_l:)}|y_i,\hat{\Theta})}$$

A.2 Calculus details of path probabilities

Let s' be a given path through the network, we have that:

$$\frac{\partial}{\partial \pi_{s'}} \log L(s|\hat{\Theta}) = \frac{\partial}{\partial \pi_{s'}} \sum_{s'} \delta_{s'} \log(\pi_{s'}) = \frac{\delta_{s'}}{\pi_{s'}}$$

with $\delta_{s'} = 1$ if $s = s'$ and 0 if not.

Hence,

$$\frac{\partial}{\partial \pi_{s'}} E_{s|y,\hat{\Theta}}[\log L(s|\Theta)] = \sum_{i}^{n} \frac{f(s_i'|y_i, \hat{\Theta})}{\pi_{s'}}$$

Consider s'', a second path through the network we have that :

$$\begin{cases} \frac{\partial}{\partial \pi_{s'}} E_{s|y,\hat{\Theta}}[\log L(s|\Theta)] = 0 \\ \frac{\partial}{\partial \pi_{s''}} E_{s|y,\hat{\Theta}}[\log L(s|\Theta)] = 0 \end{cases}$$

Hence, equalizing the two equations:

$$\begin{aligned} \sum_{i}^{n} \frac{f(s_i'|y_i, \hat{\Theta})}{\pi_{s'}} &= \sum_{i=1}^{n} \frac{f(s_i''|y_i, \hat{\Theta})}{\pi_{s''}} \\ \Leftrightarrow \pi_{s''} &= \frac{\sum_i^n f(s_i''|y_i, \hat{\Theta})}{\sum_{i=1}^n f(s_i'|y_i, \hat{\Theta})} \pi_{s'} \\ \Leftrightarrow \sum_{s''} \pi_{s''} &= \frac{\sum_{s''} \sum_{i=1}^n f(s_i''|y_i, \hat{\Theta})}{\sum_{i=1}^n f(s_i'|y_i, \hat{\Theta})} \pi_{s'} \\ \Leftrightarrow 1 &= \frac{n}{\sum_{i=1}^n f(s_i'|y_i, \hat{\Theta})} \pi_{s'} \\ \Leftrightarrow \pi_{s'} &= \frac{\sum_{i=1}^n f(s_i'|y_i, \hat{\Theta})}{n} \end{aligned}$$

B Gaussian Parameters computation details

The DGMM can be written as a regular Gaussian Mixture with a number of components equal to the number of paths. The Gaussian coefficients on each path can be computed in the following way:

$$\begin{aligned}\mu^{(1)}_{s'^{(:k_1:)}} &= \eta^{(1)}_{k'_1} + \Lambda^{(1)}_{k'_1}(\eta^{(2)}_{k'_2} + \Lambda^{(2)}_{k'_2}(...(\eta^{(L-1)}_{k'_{L-1}} + \Lambda^{(L-1)}_{k'_{L-1}}\eta^{(L)}_{k'_L})))\\ &= \eta^{(1)}_{k_1} + \sum_{l=2}^{L}(\prod_{m=1}^{l-1}\Lambda^{(m)}_{k'_m})\eta^{(l)}_{k'_l}\end{aligned}$$

and

$$\begin{aligned}\Sigma^{(1)}_{s'^{(:k_1:)}} &= \Psi^{(1)}_{k'_1} + \Lambda^{(1)}_{k'_1}(\Psi^{(2)}_{k'_2} + \Lambda^{(2)}_{k'_2}(...(\Psi^{(L)}_{k'_L} + \Lambda^{(L)}_{k'_L}\Lambda^{(L)T}_{k'_L})...)\Lambda^{(2)T}_{k'_2})\Lambda^{(1)T}_{k_1}\\ &= \Psi^{(1)}_{k'_1} + \sum_{l=2}^{L}(\prod_{m=1}^{l-1}\Lambda^{(m)}_{k'_m})(\Psi^{(l)}_{k'_l} + \Lambda^{(l)}_{k'_l}\Lambda^{(l)T}_{k'_l})(\prod_{m=1}^{l-1}\Lambda^{(m)}_{k'_m})\Lambda^{(1)T}_{k_1}\end{aligned}$$

More generally, each $z^{(l)}$ $\forall l \in [1, L]$ can be written as a Gaussian mixture over all partial paths $s'^{(l+1:L)}$ starting at layer l. On each partial path, the mean and covariance coefficients are:

MAKE k_l APPARENT IN S !!!

$$\mu_{s'^{(l:L)}} = \eta^{(l+1)}_{k'_l} + \sum_{j=l+1}^{L}(\prod_{m=l}^{j-1}\Lambda^{(m)}_{k'_m})\eta^{(j)}_{k'_j}$$

and

$$\Sigma_{s'^{(l:L)}} = \Psi^{(l)}_{k'_l} + \sum_{j=l+1}^{L}(\prod_{m=l}^{j-1}\Lambda^{(m)}_{k'_m})(\Psi^{(j)}_{k'_j} + \Lambda^{(j)}_{k'_l}\Lambda^{(j)T}_{k'_j})(\prod_{m=l}^{j-1}\Lambda^{(m)}_{k'_m})^T$$

B.1 Calculus details of GLLVM identifiabiliy rescaling

The GLLVM model assumes that the latent variable is centered and of unit variance. We define $z^{(1)new}$ the rescaled version of $z^{(1)}$ satisfying those constraints.

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

$$\begin{cases} \Sigma_{s'}^{new} = A^{-1T} \Sigma_{s'} A^{-1} \\ \mu_{s'}^{new} = A^{-1T}[\mu_{s'} - \sum_{s''} \pi_{s''} \mu_{s''}] \end{cases}$$

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

We have that :

$$\begin{cases} E(z^{(1)}) = \sum_{s''} \pi_{s''} \mu_{s''} \\ Var(z^{(1)}) = \sum_{s'} \pi_{s'} (\Sigma_{s'} + \mu_{s'} \mu_{s'}^T) - (\sum_{s'} \pi_{s'} \mu_{s'})(\sum_{s'} \pi_{s'} \mu_{s'})^T \end{cases}$$

After the rescaling :

$$\begin{cases} E(z^{(1^{new})}) = \sum_{s'} \pi_{s'} \mu_{s'}^{new} \\ Var(z^{(1^{new})}) = \sum_{s'} \pi_{s'} (\Sigma_{s'}^{new} + \mu_{s'}^{new} \mu_{s'}^{newT}) - (\sum_{s'} \pi_{s'} \mu_{s'}^{new})(\sum_{s'} \pi_{s'} \mu_{s'}^{new})^T \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = \sum_{s'} \pi_{s'} A^{-1T}[\mu_{s'} - \sum_{s''} \pi_{s''} \mu_{s''}] \\ Var(z^{(1^{new})}) = \sum_{s'} \pi_{s'} (\Sigma_{s'}^{new} + \mu_{s'}^{new} \mu_{s'}^{newT}) - (\sum_{s'} \pi_{s'} \mu_{s'}^{new})(\sum_{s'} \pi_{s'} \mu_{s'}^{new})^T \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = A^{-1T}[\sum_{s'} \pi_{s'} \mu_{s'} - (\sum_{s'} \pi_{s'}) \sum_{s''} \pi_{s''} \mu_{s''}] \\ Var(z^{(1^{new})}) = \sum_{s'} \pi_{s'} (\Sigma_{s'}^{new} + \mu_{s'}^{new} \mu_{s'}^{newT}) - (\sum_{s'} \pi_{s'} \mu_{s'}^{new})(\sum_{s'} \pi_{s'} \mu_{s'}^{new})^T \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = 0 \\ Var(z^{(1^{new})}) = \sum_{s'} \pi_{s'} (\Sigma_{s'}^{new} + \mu_{s'}^{new} \mu_{s'}^{newT}) - (\sum_{s'} \pi_{s'} \mu_{s'}^{new})(\sum_{s'} \pi_{s'} \mu_{s'}^{new})^T \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = 0 \\ Var(z^{(1^{new})}) = \sum_{s'} \pi_{s'} (\Sigma_{s'}^{new} + \mu_{s'}^{new} \mu_{s'}^{newT}) - 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = 0 \\ Var(z^{(1^{new})}) = \sum_{s'} \pi_{s'} \Big(A^{-1T} \Sigma_{s'} A^{-1} + \big(A^{-1T}[\mu_{s'} - \sum_{s''} \pi_{s''} \mu_{s''}] \big) \big(A^{-1T}[\mu_{s'} - \sum_{s''} \pi_{s''} \mu_{s''}] \big)^T \Big) \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = 0 \\ Var(z^{(1^{new})}) = A^{-1T}[\sum_{s'} \pi_{s'} \Big(\Sigma_{s'} + \big(\mu_{s'} - \sum_{s''} \pi_{s''} \mu_{s''} \big) \big(\mu_{s'} - \sum_{s''} \pi_{s''} \mu_{s''} \big)^T \Big)] A^{-1} \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = 0 \\ Var(z^{(1^{new})}) = A^{-1T}[\sum_{s'} \pi_{s'} (\Sigma_{s'} + \mu_{s'} \mu_{s'}^T) - (\sum_{s'} \pi_{s'} \mu_{s'})(\sum_{s''} \pi_{s''} \mu_{s''})^T] A^{-1} \end{cases}$$

$$\Leftrightarrow \begin{cases} E(z^{(1^{new})}) = 0 \\ Var(z^{(1^{new})}) = I_{r_1} \end{cases}$$

In our case, we aim to ensure these conditions by correcting only the parameters of the first layer *i.e.* only $(\eta_{k'_1}^{(1)}, \Lambda_{k'_1}^{(1)}, \Psi_{k'_1}^{(1)})$ $\forall k'_1 \in [1, K_1]$ rather than all of the parameters of the following layers. We will then prove that rescaling μ'_s and Σ'_s $\forall s' \in \Omega$ is equivalent to a rescaling of those parameters.

$$
\begin{aligned}
\Sigma_s^{new} &= A^{-1T}\Sigma_s A^{-1} \\
&= A^{-1T}\left[\Psi_{k_1'}^{(1)} + \Lambda_{k_1'}^{(1)}\Big(\Psi_{k_2'}^{(2)} + \sum_{l=3}^{L}\big(\prod_{m=2}^{l-1}\Lambda_{k_m'}^{(m)}\big)\big(\Psi_{k_l'}^{(l)} + \Lambda_{k_l'}^{(l)}\Lambda_{k_l'}^{(l)T}\big)\big(\prod_{m=1}^{l-1}\Lambda_{k_m'}^{(m)}\big)\Big)\Lambda_{k_1'}^{(1)T}\right]A^{-1} \\
&= A^{-1T}\Psi_{k_1'}^{(1)}A^{-1} + A^{-1T}\Lambda_{k_1'}^{(1)}\Big(\Psi_{k_2'}^{(2)} + \sum_{l=3}^{L}\big(\prod_{m=2}^{l-1}\Lambda_{k_m'}^{(m)}\big)\big(\Psi_{k_l'}^{(l)} + \Lambda_{k_l'}^{(l)}\Lambda_{k_l'}^{(l)T}\big)\big(\prod_{m=1}^{l-1}\Lambda_{k_m'}^{(m)}\big)\Big)\Lambda_{k_1}^{(1)T}A^{-1} \\
&= \Psi_{k_1'}^{(1)new} + \Lambda_{k_1'}^{(1)new}\Big(\Psi_{k_2'}^{(2)} + \sum_{l=3}^{L}\big(\prod_{m=2}^{l-1}\Lambda_{k_m'}^{(m)}\big)\big(\Psi_{k_l'}^{(l)} + \Lambda_{k_l'}^{(l)}\Lambda_{k_l'}^{(l)T}\big)\big(\prod_{m=1}^{l-1}\Lambda_{k_m'}^{(m)}\big)\Big)\Lambda_{k_1'}^{(1)Tnew}
\end{aligned}
$$

Hence one has to perform the following rescaling : $\Psi_{k_1'}^{(1)new} = A^{-1T}\Psi_{k_1'}^{(1)}A^{-1}$ and $\Lambda_{k_1'}^{(1)new} = A^{-1T}\Lambda_{k_1'}^{(1)}$.

For the second transformation, it comes:

$$
\begin{aligned}
\mu_{s'}^{new} &= A^{-1T}[\mu_{s'} - \sum_{s''}\pi_{s''}\mu_{s''}] \\
&= A^{-1T}\left[\eta_{k_1'}^{(1)} + \Lambda_{k_1'}^{(1)}\Big(\eta_{k_2'}^{(2)} + \sum_{l=3}^{L}\big(\prod_{m=2}^{l-1}\Lambda_{k_m'}^{(m)}\big)\eta_{k_l'}^{(l)}\Big) - \sum_{s''}\pi_{s''}\mu_{s''}\right] \\
&= A^{-1T}\left[(\eta_{k_1'}^{(1)} - \sum_{s''}\pi_{s''}\mu_{s''}) + \Lambda_{k_1'}^{(1)}\Big(\eta_{k_2'}^{(2)} + \sum_{l=3}^{L}\big(\prod_{m=2}^{l-1}\Lambda_{k_m'}^{(m)}\big)\eta_{k_l'}^{(l)}\Big)\right] \\
&= \eta_{k_1'}^{(1)new} + \Lambda_{k_1'}^{(1)new}\Big(\eta_{k_2'}^{(2)} + \sum_{l=3}^{L}\big(\prod_{m=2}^{l-1}\Lambda_{k_m'}^{(m)}\big)\eta_{k_l'}^{(l)}\Big)
\end{aligned}
$$

We find again that $\Lambda_{k_1}^{(1)new} = A^{-1T}\Lambda_{k_1}^{(1)new}$ but also that $\eta_{k_1}^{(1)new} = A^{-1T}\left[\eta_{k_1}^{(1)} - \sum_{s''}\pi_{s''}\mu_{s''}\right]$

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

$$
\begin{cases}
\eta_{k_1'}^{(1)new} = A^{-1T}\left[\eta_{k_1'}^{(1)} - \sum_{s''}\pi_{s''}\mu_{s''}\right] \\
\Lambda_{k_1'}^{(1)new} = A^{-1T}\Lambda_{k_1'}^{(1)} \\
\Psi_{k_1'}^{(1)new} = A^{-1T}\Psi_{k_1'}^{(1)}A^{-1}
\end{cases}
$$

C Model selection details

D Benchmark models specifications

ADD THE MODELS SPECIFICATIONS HERE

A component is considered useless if its probability is inferior to $\frac{1}{4k_l}$. For instance, if a layer is formerly composed of 3 components, one or more components are deleted if their probability is inferior to 8.33%.

For the GLLVM layer, we have chosen to perform penalized logistic regressions in order to determine which of the dimensions had a significant effect over each y_j^D for each path s'. We have fitted a regular logistic LASSO for binary and count data and an ordinal logistic regression for ordinal data as no implementation of Ordinal Logistic LASSO regression currently exists in Python. The coefficients identified as being zero (or not significant) in the majority of times over the paths and original variables were removed.

For the regular DGMM layers, we have used the same idea of majority voting to determine the useless dimensions. As our algorithm generate draws of $z^{(l+1)|z^{(l)},s}$ of dimension r_{l+1}, it is possible to perform a PCA [source] on this variable for each path and each of the $M^{(l)}$ points simulated for $z^{(l)}$. Doing so, one can compute the average contribution of each dimension of r_{l+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 $z^{(l+1)}$ has on $z^{(l)}$. If a dimension shares no common information with this first component, hence it is not useful to keep it.

TO FINISH

For the choice of the dimension of the regular DGMM layers (including $z^{(1)C}$), one could perform a PCA over the variable $z^{(l+1)}|z^{(l)}$ and compute the contribution of each dimension of the new layer to the previous one. Deleting the dimensions with the fewest pieces of information, enable to delete minor information and to keep only the strongest signal from one layer to another.

Architecture looking scheme: Then if the algorithm have not converged until iterations the 10th iteration.

The selection is made through the iterations, each layer level at a time starting from the first layer of each head. While simplifying the architecture, the likelihood will decrease mechanically and hence the likelihood cannot be used to track model fitness while performing model selection. As a result, we advise to perform a dimension/number of components selection, then wait for a few iterations and perform the next selection.

one can use PCA to quantify the information held by each latent dimension of each layer given the previous layer and drop the less important ones.

For the first discrete head dimension r_1^D, one could conduct logistic LASSO regressions to select the dimensions that carry the information. For all the other layers data, the regular LASSO could be use to perform such a task.

Finally, in order to select k_l^h, one can delete the components with very low probabilities through the iterations.

Mention the likelihood monitoring.