Introducing parsimony to analyse complex data with model-based clustering

Margot Selosse

N° d'ordre NNT : 2020LYSE2106

THESE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de

L'UNIVERSITÉ LUMIÈRE LYON 2

École Doctorale : ED 512

Informatique et Mathématiques

Discipline : Mathématiques appliquées

Soutenue publiquement le 13 novembre 2020, par :

Margot SELOSSE

Introducing parsimony to analyse complex data with model-based clustering.

Devant le jury composé de :

Stéphane CHRÉTIEN, Professeur des Universités, Université Lumière Lyon 2, Président

Pierre LATOUCHE, Professeur des Universités, Université de Paris, Rapporteur

Bettina GRÜN, Associate professor, Vienna University of Economics and Business, Rapporteuse

Charles BOUVEYRON, Professeur des Universités, INRIA et Université Côte d'Azur, Examinateur

Claire GORMLEY, Associate professor, University College Dublin, Examinatrice

Charlotte LACLAU, Maître de conférences, Université de Lyon, Examinatrice

Julien JACQUES, Professeur des Universités, Université Lumière Lyon 2, Directeur de thèse

Christophe BIERNACKI, Professeur des Universités, INRIA, Co-Directeur de thèse

Contrat de diffusion

Université Lumière Lyon 2

École doctorale InfoMaths Lyon (ED 512)

Établissement d'inscription : Université Lumière Lyon 2

Laboratoire d'accueil : Laboratoire ERIC (UR 3083)

THÈSE DE DOCTORAT EN MATHÉMATIQUES

Discipline : Mathématiques appliquées

Margot SELOSSE

Introducing parsimony to analyse complex data with model-based clustering

Date de soutenance : 13/11/2020

Après avis des rapporteurs :

Bettina Grün (Vienna University of Economics and Business)
Pierre Latouche (Université de Paris)

Jury de soutenance :

Christophe Biernacki	INRIA Lille, Université de Lille	Directeur de thèse
Charles Bouveyron	INRIA Sophia-Antipolis, Université Côte d'Azur	Invité
Stéphane Chrétien	Université Lumière Lyon 2	Président du jury
Bettina Grün	Vienna University of Economics and Business	Rapporteure
Claire Gormley	University College Dublin	Examinatrice
Julien Jacques	Université Lyon 2	Directeur de thèse
Charlotte Laclau	Université de Saint-Etienne	Examinatrice
Pierre Latouche	Université de Paris	Rapporteur

Acknowledgments

Tout d'abord, je souhaite remercier du fond du coeur mes directeurs Julien Jacques et Christophe Biernacki. Merci pour votre écoute et votre patience infinie. Julien, merci de m'avoir initiée à la recherche, merci pour ta confiance et ton soutien infaillible lors de ces dernières années. Christophe, merci pour tes conseils et tes encouragements. Vous m'avez appris énormément et je vous en suis sincèrement reconnaissante.

I would like to warmly thank Pierre Latouche and Bettina Grün for agreeing to review my thesis. Thank you both for the excellent discussions we had in the Working Group, and thank you to Pierre for all the conversations we had. I really enjoyed the research talks we had in my thesis comittee, your advices and remarks were always a great help. Thank you to Claire Gormley, Charles Bouveyron, Stéphane Chrétien and Charlotte Laclau for having accepted to be part of my jury. Claire, thank you for your support, your advices and all the nice talks we had. I feel really lucky to have had the chance to work with you.

Merci aux membres du laboratoire ERIC pour leur bonne humeur, leur solidarité et tous les moments de détente hors-boulot. Merci en particulier à Antoine, Clément, Robin, Jean, Jairo, les deux Julien et Habiba.

Merci à mes étudiants, merci à Mathieu et Florian que j'ai suivi en projets, et à ceux que j'ai eu en cours en général. C'est aussi avec vos questions que j'ai progressé, et le temps passé avec vous tous a confirmé mon goût pour l'enseignement.

Merci évidemment aux membres de ma famille: Eric et Catherine, mes parents à qui je dois tout, Léa et Emeric, Garance et Yann, Monique et Bernard, Kiko, Jacques. Merci pour votre amour, merci de me soutenir dans tous mes projets, je vous aime et je suis fière de vous avoir pour famille.

Gracias a mis queridos suegros Eugenia y Miguel y a mis cuñados Luz y Felicitas, Dámaso y Inés, Rocío y Lucano y Francisca, por lo geniales que son. Gracias por hacerme sentir parte de esta familia hermosa, gracias por esos momentos maravillosos que compartimos. Los quiero mucho y los extraño todos los dias.

Merci/Gracias/Thanks/Danke à mes amis: ceux de longue date, mes différents colocataires, ceux du lycée, de l'INSA, de Worldline, de DM, du hand, de la planche à voile, de Gre, de la Juntada et de la MLSS. Merci pour tous ces moments de rires/sport/détente/bouffe. Merci particulièrement à Elise, Maxime, Roxane, Antoine, Thu, Nicolas, Sandra et Sofiane, Julie, Sarah, Reda, Pauline et Guillaume, Audrey et Maxime, Cécile T., Cyndie, Loïc, Jonathan, Xavier et Solène, Manu, Cécile F. et Benoît, Maxime F., David et Mélanie, Juliette et Guilhem, Juan et Pao, Lucia, Santi L. et Ceci, Nico et Sofi, Javi, Ivan K., Santi G., Annika, Kasia, Séphora, Sebastian, German, Vanina et Horacio, Josefina et Francisco, Fernando et Gaby, Sole et Roger et la famille Palacios en général. Merci de faire partie de ma vie et de l'illuminer tel que vous le faîtes.

Enfin, merci à Miguel, qui me comble de bonheur. Gracias por tu amor, tu apoyo y tu optimismo indefectibles. Te dedico esta tesis, y todos los futuros capitulos de mi vida.

A Miguel.

Introducing parsimony to analyse complex data with model-based clustering

In recent years, the production of digitized information has increased exponentially. Websites, social media, smartphones and the Internet of Things in general have contributed to a massive production of data of all kinds. This overproduction has also led to more complex data sets in the sense that they are high-dimensional, sparse, heterogeneous or that they contain missing values. Traditional algorithms and statistical models are no longer sufficient to handle this kind of data since they do not take into account the already mentioned particularities and they can not scale to the "big data" phenomenon. Fortunately, the access to increasing computation power has allowed practitioners to design more complex algorithms that are being adapted to the complexity of the data.

In statistical analysis (or machine learning), unsupervised learning refers to a type of algorithms that brings new insights about the data to the user. Some examples include dimension reduction, pattern recognition and cluster analysis. The goal of cluster analysis is to find meaningful groups of observations in a data set. These groups are referred to as "clusters". In each cluster, the members have something in common that they do not share with members of other clusters. Performing cluster analysis helps finding a structure in the data set, which can be helpful in different domains such as medicine, marketing or computer vision.

Model-based clustering is an unsupervised learning domain that designs probabilistic models for cluster analysis. Defining a probabilistic model brings many advantages such as interpretability, model selection criteria and credibility intervals in the Bayesian framework. Due to their flexibility, those approaches have proved to be efficient in many domains and they are widely used for the analysis of data. A disadvantage of classical model-based clustering methods is the high number of parameters to be estimated, which can slow the inference algorithms and lead to poor results in case of complex data. Designing more parsimonious models (i.e reducing the number of parameters) is an efficient way to tackle this problem.

This thesis gathers contributions to introduce parsimony in model-based clustering methods for complex data sets. In the first two chapters, we focus on co-clustering. Co-clustering consists in clustering simultaneously the rows and columns of a matrix (or the observations and the features of a data set). We describe two novel approaches of model-based co-clustering to handle heterogeneous data sets and textual data sets. The third contribution is a thorough investigation of a Deep Gaussian Mixture Model which combines model-based clustering techniques in a deep learning fashion. We detail methodological aspects and provide numerical experiments on simulated and real data sets for all the contributions.

Keywords: model-based clustering – mixture models – co-clustering – factor analysis – parsimony.

Parcimonie dans les modèles probabilistes pour l'analyse de données complexes

Ces dernières années, la production d'informations numériques a fortement augmenté. Les sites web, les réseaux sociaux et l'Internet des Objets en général ont contribué à une production massive de données de tous genres. Cette sur-production a aussi conduit à des jeux de données plus complexes, dans la mesure où ils sont devenus de haute dimension, sparses, hétérogènes, ou encore qu'ils contiennent des valeurs manquantes. Les algorithmes et méthodes traditionnelles ne sont donc plus suffisants pour gérer ce type de données car ils ne prennent pas en compte ces particularités, et ne peuvent se mettre à l'échelle du phénomène "Big-Data". Heureusement, l'accès à des machines de plus en plus puissantes a permis aux experts de concevoir des algorithmes plus complexes, adaptés à la complexité de ces données.

L'apprentissage non-supervisé concerne un type d'algorithmes en analyse statistique (ou apprentissage automatique) qui apporte à l'utilisateur un nouveau point de vue sur les données. Quelques exemples de méthodes dites non-supervisées sont la réduction de dimension, la détection de motifs, ou encore la segmentation de données (ou analyse de clusters). L'objectif de la segmentation de données est de trouver des groupes d'observations dans un jeu de données. Ces groupes sont appelés "clusters". Dans chaque cluster, les membres ont quelque chose en commun qu'ils ne partagent pas avec les membres des autres clusters. Utiliser un algorithme de clustering aide à trouver une structure dans le jeu de données, ce qui peut-être utile dans différents domaines comme la médecine, le marketing ou la vision par ordinateur.

Les modèles probabilistes pour le clustering regroupent des méthodes d'apprentissage non-supervisé pour la segmentation de données. Définir un modèle probabiliste apporte de nombreux avantages comme l'interprétabilité, la sélection de modèle ou la possibilité d'estimer des intervalles de confiance. Grâce à leur flexibilité, ces approches ont prouvé leur efficacité dans différents domaines, et elles sont souvent utilisées pour analyser des données. Un désavantage des modèles probabilistes pour le clustering classiques, c'est qu'ils requièrent le calcul de nombreux paramètres, ce qui peut ralentir les différents algorithmes d'inférence et donner de mauvais résultats dans le contexte de données complexes. Introduire de la parcimonie (i.e. réduire le nombre de paramètres à estimer) est une manière efficace de pallier ce problème.

Cette thèse regroupe plusieurs contributions pour introduire de la parcimonie dans les modèles probabilistes pour le clustering dans le cadre de données complexes. Dans les deux premiers chapitres, nous nous concentrons sur le co-clustering. Le co-clustering consiste à effectuer un clustering simultané des lignes et des colonnes d'une matrice (ou des observations et des variables d'un jeu de données). Nous décrivons deux approches innovantes de co-clustering probabiliste pour gérer des données hétérogènes et des données textuelles. La troisième contribution investigue les modèles de mélange gaussiens profonds (ou Deep Gaussian Mixture Model), qui couple les modèles probabilistes et des techniques de réseau profond, plus communément appelées Deep Learning. Les aspects méthodologiques sont détaillés, et des expériences numériques sont réalisées sur des jeux de données simulés et réels.

Mots-Clefs : modèles probabilistes – clustering – modèles de mélange – co-clustering – analyse de facteurs – parcimonie.

Contents

Notations

Acronyms

- ML: Machine Learning,
- FMM: Finite Mixture Model,
- GMM: Gaussian Mixture Model,
- FA: Factor Analysis,
- MFA: Mixture of Factor Analysers,
- LBM: Latent Block Model,
- MLBM: Multiple Latent Block Model,
- SOCC: Self-Organised Co-Clustering,
- DGMM: Deep Gaussian Mixture Model,
- LGM: Latent Gaussian Model,
- IC: Information Criterion,
- BIC: Bayesian Information Criterion,
- ICL: Integrated Completed Likelihood,
- ARI: Adjusted Rand Index,
- NN: Neural Network,
- EM algorithm: Expectation Maximisation algorithm,
- NR algorithm: Newton-Raphson algorithm,
- IS: Importance Sampling.

Subscripts

- N: number of rows,
- i: index for rows (from 1 to N),
- J: number of columns,
- j: index for columns (from 1 to J),
- G: number of row-clusters,
- g: index for row-clusters (from 1 to G),
- H: number of column-clusters,
- h: index for column-clusters (from 1 to H),
- q: iteration,
- R: dimension of the latent scores in FA,
- Specific to the MLBM:
  - D: number of data matrices for the MLBM,
  - d: index for data matrices (from 1 to D) for the MLBM,
  - H_d: number of column-clusters of dth matrix for the MLBM,
  - J_d: number of columns of dth matrix for the MLBM,
- Specific to the DGMM:
  - L: number of layers of a DGMM,
  - l: index for layers of a DGMM,
  - $\boldsymbol{R}$: dimensions of the latent scores for each layer, vector of length L,
  - $R^{(l)}$: lth element of R, dimension of the latent scores at the lth layer in DGMM,
  - $\boldsymbol{G}$: number of clusters for each layer, vector of length L,
  - $G^{(l)}$: lth element of $\boldsymbol{G}$, number of clusters for layer l in the DGMM.
  - $\boldsymbol{g}$: one of the path of the DGMM network, is also an index for the clusters of the global GMM of the DGMM, vector of length l,
  - g_l: lth element of $\boldsymbol{g}$, index for clusters of layer l.

Data (observed and latent)

- $\boldsymbol{x}$: data set, matrix of dimension $(N \times J)$,
- x_{ij}: cell of $\boldsymbol{x}$, corresponds to the ith row and jth column,
- $\boldsymbol{v}$: set of row partitions, N vectors of length G,
- $\boldsymbol{v}_i$: ith element of $\boldsymbol{v}$, which represents the partition of row i, vector of length G,

- $\boldsymbol{w}$: set of column partitions, J vectors of length H,
- $\boldsymbol{w}_j$: jth element of $\boldsymbol{w}$, which represents the partition of column j, vector of length H,
- $\boldsymbol{w}^d$: column partitions of the dth matrix for the MLBM, J_d vectors of length H_d,
- $\boldsymbol{w}_j^d$: jth element of $\boldsymbol{w}^d$, which represents the partition of column j of the dth matrix, vector of length H_d.
- $\boldsymbol{z}$: scores in FA,
- $\boldsymbol{z}_i$: ith row of $\boldsymbol{z}$, score of row i, vector of length R,
- $\boldsymbol{z}^{(l)}$: scores of the lth layer in Deep GMMs.
- $\boldsymbol{z}_i^{(l)}$: ith row of $\boldsymbol{z}^{(l)}$, score of row i at layer l, vector of length $R^{(l)}$

Parameters

- $\boldsymbol{\theta}$: all the parameters of a model,
- ν: number of parameters,
- $\boldsymbol{\pi}$: row mixing proportions, vector of length G,
- π_g: gth element of $\boldsymbol{\pi}$, proportion of the gth row-cluster, scalar,
- $\boldsymbol{\rho}$: column mixing proportions, vector of length H,
- ρ_h: hth element of $\boldsymbol{\rho}$, proportion of the hth column-cluster, scalar,
- $\boldsymbol{\rho}^d$: column mixing proportions of the dth matrix for the MLBM, vector of length H_d,
- ρ_h^d: hth element of $\boldsymbol{\rho}^d$, for the MLBM, proportion of the hth column-cluster, scalar,
- $\boldsymbol{\alpha}$: parameters of the distributions of the blocks for the LBM,
- $\boldsymbol{\alpha}_{gh}$: element of $\boldsymbol{\alpha}$, parameters of block (g, h) for the LBM,
- $\boldsymbol{\alpha}^d$: parameters of the distributions of the blocks of the dth matrix for the MLBM,
- $\boldsymbol{\alpha}_{gh}^d$: element of $\boldsymbol{\alpha}^d$, parameters of block (g, h) of the dth matrix for the MLBM,
- $\mathcal{M}(.,.)$ is the Multinomial distribution:
  - m: number of levels,
  - $\boldsymbol{\beta}$: proportion parameters of the Multinomial distribution, vector of length m whose elements are positive and sum to 1.
  - β^r: rth element of $\boldsymbol{\beta}$, proportion of the rth level for a categorical variable.
- BOS(.,.) is the BOS ditribution:
  - μ: mode parameter of the BOS distribution, integer,
  - τ: precision parameter of the BOS distribution, scalar.
- $\mathcal{P}(.)$ is the Poisson distribution:

  - δ: part of the parameter for the Poisson distribution, scalar,
  - $n_{i.}$: part of the parameter for the Poisson distribution number of occurrences in row i,
  - $n_{.j}$: part of the parameter for the Poisson distribution number of occurrences in column j.
- $\mathcal{N}(.)$ Gaussian distribution:
  - μ: mean parameter of the unidimensional Gaussian distribution, scalar,
  - σ: standard deviation parameter of the unidimensional Gaussian distribution, scalar.
- Factor Analysis:
  - $\mathbf{\Lambda}$: factor loadings, matrix of dimension $(J \times R)$,
  - $\boldsymbol{\eta}$: mean, vector of length J,
  - $\boldsymbol{\psi}$: noise co-variance matrix, diagonal matrix of dimension J.
- Deep GMM:
  - $\pi_{\boldsymbol{g}}$: mixing proportion of the $\boldsymbol{g}$th cluster of the global GMM,
  - $\pi_{gl}^{(l)}$: mixing proportion of gth cluster of layer l,
  - $\mathbf{\Lambda}_g^{(l)}$: factor loadings of the gth cluster of lth layer, matrix of dimension $(R^{(l-1)} \times R^{(l)})$,
  - $\boldsymbol{\eta}_g^{(l)}$: mean of the gth cluster of lth layer, vector of length $R^{(l-1)}$,
  - $\boldsymbol{\psi}_g^{(l)}$: noise co-variance matrix of the gth cluster of lth layer, diagonal matrix of dimension $R^{(l-1)}$.

Functions

- $l_o(\boldsymbol{\theta}; \boldsymbol{x})$: observed-data log-likelihood, sometimes written $f(\boldsymbol{x}; \boldsymbol{\theta})$ or $p(\boldsymbol{x}; \boldsymbol{\theta})$,
- $l_c(\boldsymbol{\theta}; \boldsymbol{x})$: complete-data log-likelihood, sometimes written $f(\boldsymbol{x}, \boldsymbol{v}; \boldsymbol{\theta})$ or $p(\boldsymbol{x}, \boldsymbol{v}; \boldsymbol{\theta})$,
- $\mathbb{1}(< expr >)$: indicator function, is equal to 1 when $expr$ is true, and 0 otherwise.

1

Introduction

1.1 Scientific context

1.1.1 Machine Learning

In recent times, there has been an increased interest in machine learning. The term "machine learning" itself is quite broad and not well defined. Statistical learning, data mining, applied statistics, data science, artificial intelligence, pattern recognition, among others, are all fields that have been related to machine learning; however, depending on the user and their background, the meaning of the term "machine learning" may vary. In this thesis, we use this term to refer to the field in which we use mathematical models coupled with data to solve complex problems such as predictive maintenance, face recognition, natural language processing, weather forecasting and so on. These tasks are usually hard to solve as there is no explicit set of rules that can be used to perform them. Machine learning models are said to learn from the data since their behavior depends on the data samples that have been fed into the program as an input. In addition, these algorithms can be used on different data sets to solve different problems and this is the reason why we consider them as intelligent.

The growing interest in machine learning is due to two factors. First, the availability of digitised information has increased in recent years and now private companies and institutions have more access to massive data flows through social networks, smartphones, websites and purchase platforms. Second, this data could not have been stored, preprocessed or analysed without the enormous increase in computing power, which allows more complex and powerful models to be designed.

Machine learning algorithms often have a mathematical model of the data that can be used to describe the interactions between the different features. These models have parameters that can be tuned by the algorithm so that the observed interactions on the data samples are consistent with those of the model. The process of changing the parameters in the model based on data is referred to as "fitting" or "training", which consists in minimising a loss function of these parameters, also referred to as the "objective" function.

There are several paradigms within machine learning, each using data in a different way and for different tasks. We describe now the three main families of paradigms.

1.1.2 Three families of paradigms

1.1.2.a Supervised Learning

In supervised learning, we have two sets of variables. The input variables $\boldsymbol{x}_i$ and the label variables $\boldsymbol{y}_i$. The goal is to learn a mapping f from $\boldsymbol{x}_i$ to $\boldsymbol{y}_i$ given a training set made of pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in\{1,\dots,N\}}$. By noting $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \boldsymbol{\theta})$ the model's prediction of $\boldsymbol{y}_i$ given the model parameters $\boldsymbol{\theta}$, then the loss function $\mathcal{L}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)$ defines the score based on how precise the predictions of the model are. The parameters $\boldsymbol{\theta}$ are chosen to minimise the loss function on a data set of given samples $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in\{1,\dots,N\}}$:

$$\sum_i^N \mathcal{L}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = \sum_i^N \mathcal{L}(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})).$$

The choice of the loss function depends on the problem that needs to be solved and the nature of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$. An important property of a "good" machine learning algorithm in supervised learning is generalization. Generalization means that a model fitted on a certain data set is also consistent with samples it was not trained with (i.e. unseen samples that were not part of the training data set).

Many supervised algorithms already exist, such as Linear Regression, Logistic Regression, Decision Trees and its variants (such as Random Forests) and Support Vector Machine (Hastie et al., 2001). Lately, the deep learning algorithms also yield state-of-the-art performances on several tasks involving computer vision, speech recognition and many others (Goodfellow et al., 2016).

1.1.2.b Unsupervised learning

Unsupervised learning is broader and less well defined than supervised learning because it can serve many purposes. Basically, the goal of unsupervised learning is to find interesting structures in the data $\boldsymbol{x}$. Usually, there are no label variables $\boldsymbol{y}$ provided, only the input variables $\boldsymbol{x}$ are. Some examples of unsupervised learning tasks include density estimation, dimension reduction, feature extraction, generative modeling and cluster analysis.

- Density estimation is the construction of an estimate based on observed data of an unobservable underlying probability density function. The unobservable density function is thought of as the density according to which a large population is distributed;

the data are usually thought of as random samples from that population. The most basic form of density estimation is a rescaled histogram.

- Dimension reduction consists in finding a lower dimensional representation of the data set while keeping the most important information in the data. This can be useful for many purposes: for storing the data in a more compact way or for visualising the data set, for instance.

- Generative modeling consists in assuming that the data are sampled from a process through a density function $p(\boldsymbol{x}; \boldsymbol{\theta})$ whose parameters $\boldsymbol{\theta}$ are to be estimated (or inferred). Once the parameters are obtained, we can generate new data through sampling. The most common technique for this purpose is to maximise the log-likelihood associated to the data.

- Cluster analysis is the task of gathering the $\boldsymbol{x}_i$ samples into different groups (or "clusters") of similar observations. Observations that turn out to be in the same cluster are, then, supposed to belong to the same category. For a number G of clusters, the aims is assigning every observation of the data set $\boldsymbol{x}$ to one of the G clusters. Clustering is useful for finding structures in the data set and, thus, for a better understanding of the data.

1.1.2.c Semi-supervised learning

Semi-supervised learning (Chapelle et al., 2010) is halfway between supervised and unsupervised learning in the sense that the labels $\boldsymbol{y}_i$ are not available for all the samples $\boldsymbol{x}_i$. In this case, the goal of a semi-supervised algorithm can be to design a model that uses the remaining unlabelled samples to get a better performance in the learning task when compared to using only the labelled samples. An example of semi-supervised learning is transfer learning (Pan and Yang, 2010), which are models that use the information learnt from a certain data set (referred to as the "source" data set) to improve the performance on a different data set (referred to as "target" data set). Finally, semi-supervised learning is particularly used in the presence of graph data, where the prediction of a label for a node can be deducted from other labelled nodes and from other attributes of its neighborhood.

1.1.3 Complex data

Most machine learning techniques are well defined and efficient whenever the data set $\boldsymbol{x}$ is "easy", which means that it is structured, low-dimensional and that it does not contain missing data. However, real life data sets are often much more complex than the "toy" data sets or the data sets easily available on different platforms. The properties that make us consider a data set as "complex" in this thesis are:

- High-dimensionality, sometimes referred to as the "curse of dimensionality", relates to the phenomena that arise when handling data in a high-dimensional space (with many variables). The common issue is that when the dimensionality increases, the volume of the space increases so fast that the data become sparse (Bellman, 1966). In addition, estimation problems arise when the number of observations N is lower than the number of variables J.

- Heterogeneity, also referred to as "mixed data sets", relates to the data whose variables are not of the same nature. For instance, a simple data set with information about

the clients of a company could contain the social status (a categorical variable), the age (a count variable) and the height and weight (continuous variables) of the clients. Such a variety of data can be difficult to model mathematically since the variables do not take values in the same space. Therefore, it is difficult to define a distribution common to all variables.

- Sparse data refers to data sets where information is rare. Usually, it relates to data sets with a large majority of zero values. For instance, when we model the interactions of the users of a large social network by counting the number of messages sent to each other, the resulting matrix is usually sparse. Indeed, a lot of users never talk to each other, which results in a large amount of cells equal to zero.
- Data sets with missing values refer to the fact that, sometimes, certain cells do not have a value. For example, when analysing surveys, it is common to find questions that were not replied to by certain participants. This can be modelled in different ways, if we consider wether the respondant did not answer on purpose or unintentionally.
- Streaming data sets are those whose data come as a flow. The most common example of streaming is when the data come from sensors that actualise values of certain measures at different times. This kind of data requires special algorithms that are able to receive new data over time but this topic will not be covered in this thesis.

1.2 Contributions of the thesis

1.2.1 Focus of the thesis

This thesis focuses on unsupervised learning and, more specifically, on model-based clustering which marries generative modeling and cluster analysis to perform the clustering task. Model-based clustering designs probabilistic models for cluster analysis. Defining a probabilistic model brings many advantages such as interpretability and model selection criteria. Due to their flexibility, these approaches have proved their efficiency in many domains (Bouveyron et al., 2019), and they are methods widely used for the analysis of data. A disadvantage of classical model-based clustering methods is the high number of parameters to be estimated, which can slow the inference algorithms and lead to poor results in case of complex data. Designing more parsimonious models (i.e reducing the number of parameters) is an efficient way to tackle this problem. This thesis aims at designing novel model-based approches for complex data. In this work, we handle high-dimensional data, but we also focus on heterogeneous data sets (data with variables of different nature), missing data and sparse data sets such as textual data.

1.2.2 Content of the thesis

Chapter 2 recalls the notions necessary for a good understanding of the contributions of this thesis. First, it details the mathematical aspects of Finite Mixture Models (FMMs), which are the basis of the model-based clustering approaches. Such notions will be useful for all the chapters of this thesis. Second, this chapter contains a revision of the factor analysis paradigm and, particularly, the Mixture of Factor Analysers (MFA) as the basis for Chapter 5. Finally, this chapter defines the Latent Block Model (LBM), which is a co-clustering technique. Co-clustering is the task of clustering simultaneously the rows and the columns of a data set. These notions will be helpful for Chapter 3 and Chapter 4.

All the notions seen in this chapter will rely on the Expectation-Maximisation algorithm (EM) (Dempster et al., 1977), which is an efficient optimisation algorithm for some model-based approaches.

Chapter 3 presents an extension of the Multiple Latent Block Model (MLBM) (Robert, 2017) to mixed data. As mentioned above, mixed data sets are hard to model through probability distributions since the values that are taken by the variables do not lie in the same space. In the co-clustering case, it is particularly difficult because the algorithm has to cluster the variables too. Intuitively, it seems odd to cluster variables of different nature since the purpose of clustering is to gather elements that share something in common. The MLBM approach consists in extending the Latent Block Model (LBM), a model-based co-clustering approach, so that it is able to take mixed data into account.

Chapter 4 presents the Self-Organised Co-clustering model (SOCC) for textual data. The SOCC model was specifically designed for document-term matrices. Document-term matrices represent textual data sets with documents as observations and all the terms that are used as variables. Then, for the cell ($document_i$,$term_j$), it counts how many times $term_j$ was used in $document_i$. This representation has the advantage of being easy to read and fast to build. However, it usually results in high-dimensional and extremely sparse matrices that are difficult to exploit. The SOCC model adapts to these particularities and defines a model to cluster terms and documents that offers user-friendly results.

Chapter 5 investigates the Deep Gaussian Mixture Model (Deep GMM) (Viroli and McLachlan, 2019) and its properties. This model is based on the Mixture of Factor Analysis (MFA) model and consists in stacking MFA layers in a deep learning fashion. This is made possible by considering that the latent scores of a layer are the data input of the MFA of the next layer. In this chapter, we empirically show the difficulties to properly estimate the parameters of the model and we discuss the possible reasons and solutions to tackle these problems.

1.2.3 List of publications and softwares

PUBLICATIONS

- Margot Selosse, Julien Jacques, Christophe Biernacki, Florence Cousson-Gélie. Analysing a quality of life survey using a co-clustering model for ordinal data and some dynamic implications. Journal of the Royal Statistical Society: Series C Applied Statistics, Wiley, 2019, 68, pp.1327-1349.
- Margot Selosse, Julien Jacques, Christophe Biernacki. Model-based co-clustering for mixed type data. Computational Statistics and Data Analysis, Elsevier, 2020, 144, pp.106866.
- Margot Selosse, Julien Jacques, Christophe Biernacki, Textual data summarization using the Self-Organized Co-Clustering model, Pattern Recognition, 103, 2020.
- Margot Selosse, Julien Jacques, Christophe Biernacki. ordinalClust: an R package for analysing ordinal data. 2020. *to appear in the R journal.*
- Margot Selosse, Claire Gormley, Julien Jacques, Christophe Biernacki. A bumpy journey: exploring deep Gaussian mixture models, NeurIPS 2020 Workshop ICBINB.

R PACKAGES AVAILABLE ON CRAN

- `ordinalClust`: implementation of the method of Chapter 3 exclusively for ordinal data.
- `mixedClust`: implementation of the method of Chapter 3 for 5 types of data.

R PACKAGES AVAILABLE ON REQUEST

- `SOCC`: implementation of the method of Chapter 4.
- `deepMFA`: implementation and experiments scripts of Chapter 5.

2

State-of-the-art

2.1 Model-based clustering

2.1.1 Introduction

Model-based clustering refers to a statistical domain that proposes probabilistic models for cluster analysis. The goal of cluster analysis is to find meaningful groups of observations in a data set. These groups are referred to as "clusters". In each cluster, the members have something in common that they do not share with members of other clusters. Cluster analysis belongs to the unsupervised type of methods and it can be helpful in many applications. For example, marketing firms have data about consumers, and it is too time-consuming to customise shopping offers specific to every consumer. Clustering the clients is a good solution since it defines a low number of consumer's clusters. Each cluster can represent a typical consumption profile. It is then easier to develop an offer or a product for every profile.

The biological classification system or taxonomy of Linnaeus, applied to plants and animals in the year 1735, is an early example of the grouping of observations (Bouveyron et al., 2019). Linnaeus divided plants into 24 classes, including flowers with one stamen (Monandria), flowers with two stamens (Diandria) and flowerless plants (Cryptogamia). His methods of grouping were based on data; however, the criteria to separate the flowers into groups were subjective. Cluster analysis is something more: the search for groups in quantitative data using automatic methods. From the early 1930s onwards, a range of automatic algorithms was proposed, most of which were based on a matrix to measure the similarities between the observations. The purpose of those automatic algorithms was to divide or partition the data into groups such that the observations within a certain group were similar among themselves but dissimilar to the observations in other groups with similar characteristics among themselves as well. The most famous method, referred to as the "k-means" algorithm, was proposed in Steinhaus (1956) and, nowadays, it is still largely used. However, these methods leave several practical questions unresolved such as how many clusters we should use, how to compare the results of the many available clustering methods or how certain we are of a clustering partition. The probabilistic approaches, which specify probabilistic models for the full data set, have the potential to answer these questions. The main statistical models for clustering are the finite mixture models in which each group is modelled by its own probability distribution. In the following section, we describe the finite mixture models and the Expectation-Maximisation algorithm, which is widely used to infer their parameters.

2.1.2 Finite mixture models

Finite mixture models are used to model the probability density function of random variables with a weighted linear combination of G component densities (or cluster densities). They were first proposed in Pearson and Henrici (1894) where the author modelled the distribution of ratios between forehead width and body length for 1000 Neapolitan crabs with $G = 2$ univariate Gaussian distributions.

To define these models, we assume that the data $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ consist of N multivariate observations, each of dimension J, such that $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iJ})$. A Finite Mixture Model (FMM) represents the probability distribution (or density function) of an observation

$\boldsymbol{x}_i$ as a finite mixture of weighted average of G probability density functions:

$$f(\boldsymbol{x}_i) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g). \tag{2.1}$$

In Equation (2.1), $\pi_g \geq 0$, $\forall g$ and $\sum_{g=1}^{G} \pi_g = 1$. In addition, $f_g(.; \boldsymbol{\alpha}_g)$ is the density of the gth component given the value of its parameter $\boldsymbol{\alpha}_g$. The parameters $\pi_1, \ldots, \pi_G$ are referred to as "mixing proportions" and $f_1, \ldots, f_G$ are referred to as "component densities". We can characterise this model via a hierarchical construction by considering a G-dimensional latent variable $\boldsymbol{v}_i$, that is encoded such that $v_{ig} = 1$ when $\boldsymbol{x}_i$ belongs to the gth component, and $v_{ig} = 0$ otherwise. We give the probability $p(v_{ig} = 1) = \pi_g$. If we assume that the conditional density of $\boldsymbol{x}_i$ given $v_{ig} = 1$ equals $f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g)$, then, the joint density of $(\boldsymbol{x}_i, \boldsymbol{v}_i)$ can be written as follows:

$$f(\boldsymbol{x}_i, \boldsymbol{v}_i; \boldsymbol{\theta}) = \prod_{g=1}^{G} [\pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g)]^{v_{ig}}. \tag{2.2}$$

Hence, the marginal density of $\boldsymbol{x}_i$ is:

$$\begin{aligned} f(\boldsymbol{x}_i; \boldsymbol{\theta}) &= \sum_{\boldsymbol{v}_i \in \mathcal{V}} f(\boldsymbol{x}_i, \boldsymbol{v}_i) \\ &= \sum_{\boldsymbol{v}_i \in \mathcal{V}} \Big(\prod_{g=1}^{G} [\pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g)]^{v_{ig}} \Big) \\ &= \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g), \end{aligned} \tag{2.3}$$

where $\mathcal{V}$ represents all the possible values of $\boldsymbol{v}_i$. Therefore, the data set $\boldsymbol{x}$ can be seen as an i.i.d sample generated from a FMM with a probability density function $f(\boldsymbol{x}; \boldsymbol{\theta})$ where $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_G, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_G)$ represents all the parameters of the model, and the observed-data log-likelihood noted as $l_o(\boldsymbol{\theta}; \boldsymbol{x})$ is given by:

$$\begin{aligned} l_o(\boldsymbol{\theta}; \boldsymbol{x}) &= \log \prod_{i=1}^{N} f(\boldsymbol{x}_i; \boldsymbol{\theta}) \\ &= \sum_{i=1}^{N} \log \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g). \end{aligned} \tag{2.4}$$

In this thesis, we focus on the homogeneous parametric FMMs whose component densities arise from the same parametric family such as the Gaussian Mixture Models (GMM) (Celeux and Govaert, 1995), the t-distribution Mixture Models (Peel and McLachlan, 2000) and the skew normal Mixture Models (Lin et al., 2007); however, there are other types of FMMs. In the heterogeneous parametric FMMs, the components densities come from different parametric families (see Coretto and Hennig (2011)). The non-parametric FMMs refer to the cases in which no assumptions are made about the form of $f_g(.)$ (e.g. Benaglia et al. (2009)).

There are several inference methods for the FMMs but we focus on the maximum likelihood framework (McLachlan and Peel, 2000). The aim of this approach is to maximise

the log-likelihood as a function with respect to the model parameters $\boldsymbol{\theta}$. There is no way to maximise the log-likelihood function of Equation (2.4) in a closed form. However, it can be maximised using iterative procedures such as the Newton-Raphson method and the EM-algorithm (McLachlan and Peel, 2000). At each iteration, the Newton-Raphson method requires computing the Hessian matrix of the log-likelihood function and the solution of ν linear equations, where ν is the number of components of the vector of parameters $\boldsymbol{\theta}$. When $\boldsymbol{\theta}$ is high-dimensional, an iteration of the Newton-Raphson method can become very expensive. Furthermore, this algorithm is extremely sensitive to the starting value and the convergence from an arbitrary starting value is not guaranteed. These are the reasons why the Expectation-Maximisation algorithm is more popular and widely used for the inference of FMMs. In this thesis, we focus on the EM algorithm and we describe it in the next section.

2.1.3 Expectation Maximisation algorithm for FMMs

In this framework, we consider the sample $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and it is assumed that the observation $\boldsymbol{x}_i$ is associated to the G-dimensional latent variable $\boldsymbol{v}_i$. The set of vectors $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N)$ is a latent variable and will often be referred to as the "row partitions" or "partitions". The complete-data log-likelihood function $l_c(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{v})$ can be written as follows:

$$\begin{aligned} l_c(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{v}) &= \log \prod_{i=1}^{N} f(\boldsymbol{x}_i, \boldsymbol{v}_i; \boldsymbol{\theta}) \\ &= \log \prod_{i=1}^{N} \prod_{g=1}^{G} \left(\pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g)\right)^{v_{ig}} \\ &= \sum_{i=1}^{N} \sum_{g=1}^{G} v_{ig} \{\log \pi_g + \log f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g)\}. \end{aligned} \tag{2.5}$$

The EM-algorithm consists of iterations composed of two steps: the Expectation (E) step and the Maximisation (M) step that are iteratively run until convergence. We note $\boldsymbol{\theta}^{(q)}$ as the value of the parameters vector $\boldsymbol{\theta}$ after the qth iteration. The EM algorithm starts with some starting value $\boldsymbol{\theta}^{(0)}$.

E-STEP At the qth iteration, this step consists in computing the expectation of $l_c(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{v})$ conditional to $\boldsymbol{x}$ using $\boldsymbol{\theta}^{(q-1)}$. This expectation is sometimes referred to as "the auxiliary function" in the literature, and we denote it by $\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(q-1)})$:

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(q-1)}) &= \mathbb{E}[l_c(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{v}) | \boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}] \\ &= \sum_{i=1}^{N} \sum_{g=1}^{G} \mathbb{E}[v_{ig} | \boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}] \{\log \pi_g^{(q-1)} + \log f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g^{(q-1)})\} \\ &= \sum_{i=1}^{N} \sum_{g=1}^{G} t_{ig}^{(q)} \{\log \pi_g^{(q-1)} + \log f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g^{(q-1)})\}, \end{aligned} \tag{2.6}$$

where

$$t_{ig}^{(q)} = \mathbb{E}[v_{ig} | \boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}] = \frac{\pi_g^{(q-1)} f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g^{(q-1)})}{\sum_{g'=1}^{G} \pi_{g'}^{(q-1)} f_{g'}(\boldsymbol{x}_i | \boldsymbol{\alpha}_{g'}^{(q-1)})}. \tag{2.7}$$

$t_{ig}^{(q)}$ is the posterior probability that $\boldsymbol{x}_i$ belongs to the gth component at iteration q. In the case of FMMs, the E-step requires computing of $t_{ig}^{(q)}$ for all $i \in \{1, \ldots, N\}$ and for all $g \in \{1, \ldots, G\}$.

M-STEP On the qth iteration, this step computes the estimate of $\boldsymbol{\theta}$ that maximises the function $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$:

$$\boldsymbol{\theta}^{(q)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(q-1)}). \tag{2.8}$$

The estimation of the mixing proportions $\boldsymbol{\pi}^{(q)} = (\pi_1^{(q)}, \ldots, \pi_G^{(q)})$ is evaluated independently of the estimation of the parameters $\boldsymbol{\alpha}^{(q)} = (\boldsymbol{\alpha}_1^{(q)}, \ldots, \boldsymbol{\alpha}_G^{(q)})$ that corresponds to the parameters of the component densities of the FMM. This is possible since $\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(q-1)})$ can be written:

$$\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(q-1)}) = \sum_{i=1}^{N}\sum_{g=1}^{G} t_{ig}^{(q)} \log \pi_g^{(q-1)} + \sum_{i=1}^{N}\sum_{g=1}^{G} t_{ig}^{(q)} \log f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g^{(q-1)}). \tag{2.9}$$

By taking into account the constraint $\sum_g \pi_g = 1$, we can maximise $\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(q-1)})$ with respect to $\boldsymbol{\pi}$ using the method of the Lagrange multipliers. We obtain the following updates:

$$\pi_g^{(q)} = \frac{\sum_{i=1}^{N} t_{ig}^{(q)}}{N}, \forall g \in \{1, \ldots, G\}. \tag{2.10}$$

From Equation (2.9), we see that we can obtain $\boldsymbol{\alpha}^{(q)}$ by finding the root of:

$$\sum_{i=1}^{N}\sum_{g=1}^{G} t_{ig}^{(q)} \frac{\partial \log f_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g)}{\partial \boldsymbol{\alpha}}. \tag{2.11}$$

In Appendix 2.4.1, we explain why an iteration of the EM-algorithm causes the log-likelihood to increase.

2.1.4 Gaussian Mixture Models

The most common FMMs are those defined by Equation (2.4) whose component densities arise from multivariate Gaussian densities referred to as "Gaussian Mixture Models" or GMMs. In this context, a component parameter $\boldsymbol{\alpha}_g$ corresponds to the mean vector $\boldsymbol{\mu}_g$ and covariance matrix $\boldsymbol{\Sigma}_g$. The GMMs are popular since many natural measurements and processes tend to have Gaussian distributions; thus, populations containing subpopulations of these measurements will tend to have densities similar to GMMs. The GMMs can be written as follows:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}; \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g). \tag{2.12}$$

The main criticism of the GMMs is that they require the estimation of a large number of parameters. A J-variate GMM with G components requires computing $G \times J(J+1)/2$ parameters to estimate the covariance matrices: $G \times J$ for the means and $G - 1$ for the mixing proportions. In Banfield and Raftery (1993) and Celeux and Govaert (1995), the authors propose to reduce the parametric complexity of the model from Equation (2.12)

Eigen decomposition of $\boldsymbol{\Sigma}_g$	# of parameters for $(\boldsymbol{\Sigma}_g)_g$	Sphericity	Volume	Shape	Orientation	Name
$\lambda \boldsymbol{I}$	1	Spherical	Equal	Equal	None	EII
$\lambda_g \boldsymbol{I}$	G	Spherical	Variable	Equal	None	VII
$\lambda_g \boldsymbol{A}$	$G + J - 1$	Diagonal	Variable	Equal	Coordinate axes	VEI
$\lambda \boldsymbol{A}_g$	$G(J-1)+1$	Diagonal	Equal	Variable	Coordinate axes	EVI
$\lambda_g \boldsymbol{A}_g$	GJ	Diagonal	Variable	Variable	Coordinate axes	VVI
$\lambda \boldsymbol{A}$	J	Diagonal	Equal	Equal	Coordinate axes	EEI
$\lambda \boldsymbol{D}\boldsymbol{A}\boldsymbol{D}^T$	$J(J+1)/2$	Elliptical	Equal	Equal	Equal	EEE
$\lambda_g \boldsymbol{D}\boldsymbol{A}\boldsymbol{D}^T$	$J(J+1)/2 + G - 1$	Elliptical	Variable	Equal	Equal	VEE
$\lambda \boldsymbol{D}\boldsymbol{A}_g\boldsymbol{D}^T$	$J(J+1)/2 + (G-1)(J-1)$	Elliptical	Equal	Variable	Equal	EVE
$\lambda_g \boldsymbol{D}\boldsymbol{A}_g\boldsymbol{D}^T$	$J(J+1)/2 + (G-1)J$	Elliptical	Variable	Variable	Equal	VVE
$\lambda \boldsymbol{D}_g\boldsymbol{A}\boldsymbol{D}_g^T$	$GJ(J+1)/2 - (G-1)J$	Elliptical	Equal	Equal	Variable	EEV
$\lambda_g \boldsymbol{D}_g\boldsymbol{A}\boldsymbol{D}_g^T$	$GJ(J+1)/2 - (G-1)(J-1)$	Elliptical	Variable	Equal	Variable	VEV
$\lambda \boldsymbol{D}_g\boldsymbol{A}_g\boldsymbol{D}_g^T$	$GJ(J+1)/2 - (G-1)$	Elliptical	Equal	Variable	Variable	EVV
$\lambda_g \boldsymbol{D}_g\boldsymbol{A}_g\boldsymbol{D}_g^T$	$GJ(J+1)/2$	Elliptical	Variable	Variable	Variable	VVV

Table 2.1 – The fourteen models for parameterizations of the covariance matrix $\boldsymbol{\Sigma}_g$ in GMMs context.

by introducing parsimony. They consider the eigenvalue decomposition of the covariance matrix:

$$\boldsymbol{\Sigma}_g = \lambda_g \boldsymbol{D}_g \boldsymbol{A}_g \boldsymbol{D}_g^T, \tag{2.13}$$

where $\lambda_g \in \mathbb{R}$ is equal to $|\boldsymbol{\Sigma}_g|^{1/J}$, $\boldsymbol{D}_g \in \mathbb{R}^{J\times J}$ is the matrix of eigenvectors of $\boldsymbol{\Sigma}_g$, and $\boldsymbol{A}_g \in \mathcal{R}^{J\times J}$ is a diagonal matrix such that $|\boldsymbol{A}_g| = 1$, with the normalised eigenvalues of $\boldsymbol{\Sigma}_g$ in a decreasing order. The parameter λ_g determines the volume of the gth cluster, $\boldsymbol{D}_g$ its orientation and $\boldsymbol{A}_g$ its shape. The total number of parameters can be decreased by imposing various restrictions on λ_g, $\boldsymbol{D}_g$ or $\boldsymbol{A}_g$, such as setting $\lambda_g = \lambda$, $\boldsymbol{D}_g = \boldsymbol{D}$, $\boldsymbol{A}_g = \boldsymbol{A}$ or fixing $\boldsymbol{D}_g\boldsymbol{A}_g\boldsymbol{D}_g^T = \boldsymbol{I}$. In total, fourteen models arise from the variations, namely "`EII`'", "`VII`'", "`VEI`'", "`EVI`'", "`VVI`'", "`EEI`'", "`EEE`'", "`VEE`'", "`EVE`'", "`VVE`'", "`EEV`'", "`VEV`'", "`EVV`'", "`VVV`'". Table 2.1 reports all the possible combinations of these restrictions with the corresponding number of parameters for $\boldsymbol{\Sigma}_g$, sphericity, volume, shape orientation and model names. In Figure 2.1, the geometric characteristics are shown graphically.

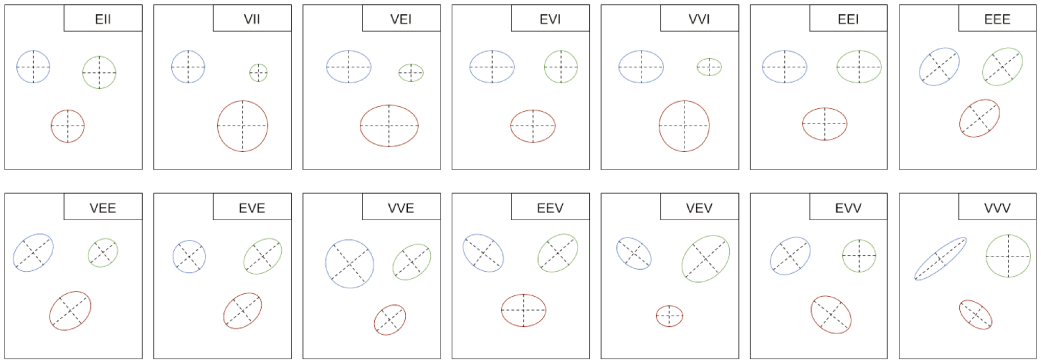

Figure 2.1 – Graphical representations of the fourteen models for GMMs, in dimension $J = 2$.

2.1.5 Model Selection

Until now, we have considered the Finite Mixture Models and the EM-algorithm to estimate the parameters in the context where G is known. However, in practice, we do not always know the number of components G. In addition, we have described the fourteen models to

introduce parsimony and to reduce the number of parameters to be estimated in GMMs, but we do not necessarily know which of the fourteen models is the best one. At this point, we have to choose the number of components and the most parsimonious model that best fits the data. These decisions turn out to be a model selection problem. It is still an active research topic and for reviews on the matter, we refer to Fernández and Arnold (2016), Fonseca and Cardoso (2007) and McLachlan and Peel (2000). The most popular methodology for model selection is to use an information criterion (IC) that aims at balancing fitness (trying to maximise the log-likelihood function) and parsimony (using penalties associated with measures of model complexity) to avoid overfitting the data. When using an IC to compare several models fitted to the same data set, we select the model with the lowest IC value. The best known information criteria is the Bayesian Information Criterion (Schwarz, 1978) referred to as "BIC" and the Integrated Classification Likelihood criterion (Biernacki et al., 2000) referred to as "ICL". We present the details of these criteria in this section.

2.1.5.a Deriving the Bayesian Information Criterion

In this section, we present how the BIC criterion is derived based on Lebarbier and Mary-Huard (2004) and Raftery (1995). Let us consider the sample $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ of independent variables with unkonwn density f. The aim is to estimate f, and we have a finite set of models $\{M_1, \ldots, M_t\}$ available for this goal. A model M_l corresponds to a density h_{M_l} with parameters $\boldsymbol{\theta}_l$. We denote the space for values of $\boldsymbol{\theta}_l$ as Θ_l and we denote the dimension of $\boldsymbol{\theta}_l$ as ν_l. Model selection consists in choosing a model among the set of models. The BIC criterion is set in a Bayesian context, that is to say, M_l and $\boldsymbol{\theta}_l$ are seen as random variables and have prior distributions noted as $p(M_l)$ and $p(\boldsymbol{\theta}_l|M_l)$. Such an approach is interesting when the user wants to give more weight to some models through $p(M_l)$. However, the distribution $p(M_l)$ is usually assumed to be non-informative (i.e. uniform). In addition, we will see that the prior distribution $p(\boldsymbol{\theta}_l|M_l)$ will not appear in the BIC expression. The BIC criterion aims at choosing the model M_l that maximises the posterior distribution $p(M_l|\boldsymbol{x})$:

$$M_{BIC} = \underset{M_l}{\operatorname{argmax}}\, p(M_l|\boldsymbol{x}). \tag{2.14}$$

From the Bayes formula, $P(M_l|\boldsymbol{x})$ is given by:

$$P(M_l|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|M_l)p(M_l)}{p(\boldsymbol{x})}. \tag{2.15}$$

From now, we suppose that $p(M_l)$ is non-informative, i.e. $p(M_1) = \ldots = p(M_t)$. From Equations (2.14) and (2.15), we need to maximise $p(\boldsymbol{x}|M_l)$ that can be expressed as follows:

$$\begin{aligned} p(\boldsymbol{x}|M_l) &= \int_{\Theta_l} p(\boldsymbol{x}, \boldsymbol{\theta}_l|M_l) d\boldsymbol{\theta}_l \\ &= \int_{\Theta_l} h_{M_l}(\boldsymbol{x}; \boldsymbol{\theta}_l) p(\boldsymbol{\theta}_l|M_l) d\boldsymbol{\theta}_l, \end{aligned} \tag{2.16}$$

where $h_{M_l}(\boldsymbol{x}; \boldsymbol{\theta}_l)$ is the density corresponding to model M_l:

$$h_{M_l}(\boldsymbol{x}; \boldsymbol{\theta}_l) = p(\boldsymbol{x}|\boldsymbol{\theta}_l, M_l). \tag{2.17}$$

We write the integral of Equation (2.16) with another form:

$$p(\boldsymbol{x}|M_l) = \int_{\Theta_l} \exp\{h(\boldsymbol{\theta}_l)\} d\boldsymbol{\theta}_l$$

with

$$h(\boldsymbol{\theta}_l) = \log\big(h_{M_l}(\boldsymbol{x}, \boldsymbol{\theta}_l) p(\boldsymbol{\theta}_l|M_l)\big). \tag{2.18}$$

Usually, $p(\boldsymbol{x}|M_l)$ is intractable. However, we can use the Laplace approximation and write:

$$p(\boldsymbol{x}|M_l) = \exp\{h(\boldsymbol{\theta}_l^\star)\}\big(\frac{2\pi}{N}\big)^{\nu_l/2} |H_{\boldsymbol{\theta}_l^\star}|^{-1/2} + \mathcal{O}(N^{-1}), \tag{2.19}$$

with:

- $\boldsymbol{\theta}_l^\star = \underset{\boldsymbol{\theta}_l}{\operatorname{argmax}} \frac{h(\boldsymbol{\theta}_l)}{N}$ and
- $H_{\boldsymbol{\theta}_l^\star}$ is the Hessian matrix defined by:

$$H_{\boldsymbol{\theta}_l^\star} = -\Big[\frac{\partial^2 h(\boldsymbol{\theta}_l)/N}{\partial \boldsymbol{\theta}_{l_i} \partial \boldsymbol{\theta}_{l_j}}\Big]_{i,j} | \boldsymbol{\theta}_l = \boldsymbol{\theta}_l^\star,$$

  where $\boldsymbol{\theta}_{l_i}$ is the ith coordinate of $\boldsymbol{\theta}_l$.

From Equations (2.18) and (2.19), we can write:

$$\log p(\boldsymbol{x}|M_l) = \log h_{M_l}(\boldsymbol{x}, \boldsymbol{\theta}_l^\star) + \log p(\boldsymbol{\theta}_l^\star|M_l) + \frac{\nu_l}{2}\log(2\pi) - \frac{1}{2}\log(|H_{\boldsymbol{\theta}_l^\star}|) + \mathcal{O}(N^{-1}). \tag{2.20}$$

The difficulty is now to evaluate $\boldsymbol{\theta}_l^\star$ and $H_{\boldsymbol{\theta}_l^\star}$. $\boldsymbol{\theta}_l^\star$ is asymptotically equal to:

$$\hat{\boldsymbol{\theta}}_l = \underset{\boldsymbol{\theta}_l}{\operatorname{argmax}} \frac{1}{N} h_{M_l}(\boldsymbol{x}, \boldsymbol{\theta}_l). \tag{2.21}$$

Asymptotically, the Hessian matrix $H_{\boldsymbol{\theta}_l^\star}$ can be replaced by the Fisher's Information matrix $F_{\hat{\boldsymbol{\theta}}_l}$:

$$F_{\hat{\boldsymbol{\theta}}_l} = -\mathbb{E}\bigg(\Big[\frac{\partial^2 \log g_{M_l}(\boldsymbol{x}, \boldsymbol{\theta}_l)}{\partial \boldsymbol{\theta}_{l_i} \partial \boldsymbol{\theta}_{l_j}}\Big]_{i,j} | \boldsymbol{\theta}_l = \hat{\boldsymbol{\theta}}_l\bigg), \tag{2.22}$$

because when N increases, $\log g_{M_l}(\boldsymbol{x}, \boldsymbol{\theta}_l)$ increases too, whereas $\log p(\boldsymbol{\theta}_l|M_l)$ stays constant. We substitute $\boldsymbol{\theta}_l^\star$ by $\hat{\boldsymbol{\theta}}_l$ and $H_{\boldsymbol{\theta}_l^\star}$ by $F_{\hat{\boldsymbol{\theta}}_l}$ in Equation (2.20) and get:

$$\begin{aligned}\log p(\boldsymbol{x}|M_l) = &\overbrace{\log h_{M_l}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_l) - \frac{\nu_l}{2}\log N}^{\text{tends to } -\infty \text{ when } N\to\infty} \\ &\overbrace{+ \log p(\boldsymbol{\theta}_l^\star|M_l) + \frac{\nu_l}{2}\log(2\pi) - \frac{1}{2}\log(|F_{\hat{\boldsymbol{\theta}}_l}|)}^{\text{stays bounded when } N\to\infty:\ \mathcal{O}(1)} + \mathcal{O}(N^{-1/2}).\end{aligned} \tag{2.23}$$

We neglect the error terms $\mathcal{O}(1)$ and $\mathcal{O}(N^{-1/2})$ and get:

$$\log p(\boldsymbol{x}|M_l) \approx \log h_{M_l}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_l) - \frac{\nu_l}{2}\log N. \tag{2.24}$$

The BIC criterion arises from these approximations. More precisely, the BIC of the model M_l (noted as BIC_{M_l}) corresponds to the approximations of $-2\log p(\boldsymbol{x}|M_l)$ and is defined by:

$$\begin{aligned}\text{BIC}_{M_l} &= -2\log h_{M_l}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_l) + \nu_l \log N \\ &= -2\log p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_l, M_l) + \nu_l \log N. \end{aligned} \tag{2.25}$$

The model M_{BIC} chosen by this criterion is:

$$M_{BIC} = \underset{M_l}{\text{argmin}}\ \text{BIC}_{M_l}. \tag{2.26}$$

2.1.5.b Integrated Classification Likelihood criterion

In Biernacki et al. (2000), the authors introduce the ICL criterion, which is a variant of the BIC method in the sense that the ICL approximates the complete data log-likelihood of the model, which is written as follows:

$$\log p(\boldsymbol{x}, \boldsymbol{v}|M_l) = \log \int_{\Theta_l} p(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\theta}_l|M_l) d\boldsymbol{\theta}_l. \tag{2.27}$$

With a similar reasoning to that of BIC, we find the ICL to be equal to:

$$\text{ICL}_{M_l} = -2\log p(\boldsymbol{x}, \boldsymbol{v}|\hat{\boldsymbol{\theta}}_l, M_l) + \nu_l \log N. \tag{2.28}$$

The performances of ICL compared to those of BIC in choosing the number of components have been assessed in multiple works such as Biernacki et al. (2000) and Baudry et al. (2010). In fact, the ICL turns out to be equal to the BIC penalised by the estimated mean entropy:

$$\text{ICL}_{M_l} = \text{BIC}_{M_l} - \sum_{g}^{G_{M_l}} \sum_{i}^{N} p(v_{ig} = 1|\boldsymbol{x}; \hat{\boldsymbol{\theta}}_l) \log p(v_{ig} = 1|\boldsymbol{x}; \hat{\boldsymbol{\theta}}_l). \tag{2.29}$$

In comparison to the BIC, the ICL introduces an additional term that penalises clustering configurations that exhibit overlapping groups: low-entropy solutions with well-separated groups are preferred to configurations that give the best match with regard to the distributional assumptions. In addition, the ICL criterion will often be chosen over the BIC criterion when the complete-data log-likelihood is easier to compute than the log-likelihood itself. For example, this is the case of the Stochastic Block Model (Latouche et al., 2010) or the Latent Block Model (Nadif and Govaert, 2008), which will be described later.

2.1.6 Conclusion

The notions presented in this section are the basis of the model-based clustering and will be of help in subsequent chapters. Finite Mixture Models are a powerful framework for estimating densities and performing clustering on data. However, they can suffer from some drawbacks due to the high number of parameters to be estimated in high-dimensionality. As we saw in Section 2.1.4, practitioners tried to find solutions to introduce parsimony in the classical models. Many other models exist to address this issue and they use different approaches such as selection of variables (Fop and Murphy, 2018) and dimension reduction (McNicholas and Murphy, 2008; Bouveyron et al., 2007). In this chapter, we will describe two of these approaches.

In the next section, we will study Mixture of Factor Analysers (MFA) (McLachlan and Peel, 2000) which is a model-based approach for clustering and reducing the dimension of the data at the same time. In Section 2.3, we will describe the Latent Block Model (Nadif and Govaert, 2008), which is a model-based co-clustering approach. Co-clustering consists in clustering simultaneously the rows and the columns of a matrix. We will develop the idea that, by gathering the features into clusters, co-clustering can be seen as a tool to bring parsimony to clustering.

2.2 Factor Analysis

2.2.1 Introduction

As explained in Section 2.1, model-based clustering is a popular tool to group data into several homogeneous groups. In this section, we describe Factor Analysis (FA) methods, which rely on the assumption that the observed data was generated from latent variables with a lower dimension. In Section 2.2.2, we present the single factor analysis, a statistical method for dimension reduction. In Section 2.2.5, we detail the mixture of factor analysers, which performs clustering and dimension reduction at the same time.

2.2.2 Single Factor Analyser

In maximum likelihood factor analysis, a vector $\boldsymbol{x}_i$ of $\boldsymbol{x}$ is modelled using an R-dimensional vector of real valued factors, $\boldsymbol{z}_i$. Usually, R is much smaller than J. The generative model is given by:

$$\boldsymbol{x}_i = \boldsymbol{\Lambda}\boldsymbol{z}_i + \boldsymbol{u}_i, \tag{2.30}$$

where $\boldsymbol{\Lambda}$ is a $J \times R$ matrix and is referred to as the "factor loadings". The R-dimensional latent variables $\boldsymbol{z}_i$ are referred to as "scores". They are assumed to follow the standard mutivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_R)$, where $\boldsymbol{I}_R$ is R-squared identity matrix. The J-dimensional random variable $\boldsymbol{u}_i$ follows the multivariate Gaussian distribution $\mathcal{N}(0, \boldsymbol{\psi})$, where $\boldsymbol{\psi}$ is a diagonal matrix. The aim of factor analysis is to find $\boldsymbol{\theta} = (\boldsymbol{\Lambda}, \boldsymbol{\psi})$ that best models the covariance structure of $\boldsymbol{x}_i$. The scores $\boldsymbol{z}_i$ model correlations between the elements of $\boldsymbol{x}_i$ and play the same role as the principal component scores in Principal Component Analysis: they are informative projections of the data. The variables $\boldsymbol{u}_i$ represent the independent noise in each element of $\boldsymbol{x}_i$. Considering the whole matrix $\boldsymbol{x}$, we can write the model in the matricial notation:

$$\boldsymbol{x} = \boldsymbol{\Lambda}\boldsymbol{z} + \boldsymbol{u}, \tag{2.31}$$

where $\boldsymbol{z}$ is an $R \times N$ matrix, whose ith column corresponds to $\boldsymbol{z}_i$, and where $\boldsymbol{u}$ is a $J \times N$ matrix, whose ith row corresponds to $\boldsymbol{u}_i$. Let us note that the factor analysis model, as it was defined, is deeply related to the probabilistic PCA (Tipping and Bishop, 1999a) whose noise covariance matrix is assumed to be isotropic ($\boldsymbol{\psi} = \sigma^2 \boldsymbol{I}_R$, with $\sigma \in \mathbb{R}$). The observed-data log-likelihood of the model is written as follows:

$$l_o(\boldsymbol{\theta}; \boldsymbol{x}) = \log \int_{\boldsymbol{z}} f(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) d\boldsymbol{z}, \tag{2.32}$$

since the scores $\boldsymbol{z}$ are continuous multivariate variables.

2.2.3 Inference for the Single Factor Analyser

Once again, the Single Factor Analyser is a problem with latent variables (the scores $\boldsymbol{z}$) and parameters ($\boldsymbol{\Lambda}$ and $\boldsymbol{\psi}$) that has to be resolved iteratively since no closed-form expression exists for the maximum likelihood estimation of $\boldsymbol{\Lambda}$ and $\boldsymbol{\psi}$ (McLachlan and Krishnan, 1997). However, the model can be fitted using the Expectation-Maximisation (EM) algorithm as considered in Rubin and Thayer (1982). Let us recall that the EM-algorithm requires, in the first place, computing the expectation of the complete-data log-likelihood. From the model definition, we use the property of affine transformations of normal random variables and the property of the sum of normal random vectors to conclude that $\boldsymbol{x}$ is also Gaussian distributed with mean vector $\mathbf{0}$ and covariance matrix $\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\psi}$. Since the data $\boldsymbol{x}$ and factors $\boldsymbol{z}$ are both Gaussian distributed, their joint density is also Gaussian. In other words, the complete-data log-likelihood is equal to:

$$l_c(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{z}) = \log \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \tag{2.33}$$

where $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\psi} & \boldsymbol{\Lambda} \\ \boldsymbol{\Lambda}^T & \boldsymbol{I}_R \end{bmatrix}$.
Actually, the Single Factor Analyser can be seen as a way of specifying a joint density model on $\boldsymbol{x}$ using a small number of parameters. Indeed, the case of a Gaussian density with classic full covariance matrix requires computation $J \times (J+1)/2$ whereas FA requires computation of $J \times (R+1)$ parameters. This model allows a flexible trade-off between a full covariance matrix and a diagonal covariance matrix.

From now on, we are going to use convenient properties of the joint Gaussians to show why maximising the expectation of the complete-data log-likelihood is equivalent to maximising the expectation of the conditional probability density function $f(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta})$. We can write the joint distribution as follows:

$$l_c(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{z}) = \log f(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta}) + \log f(\boldsymbol{z}; \boldsymbol{\theta}). \tag{2.34}$$

With the conditional distribution expressed by:

$$f(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\Lambda}\boldsymbol{z}, \boldsymbol{\psi}). \tag{2.35}$$

Let us focus on the prior $f(\boldsymbol{z}; \boldsymbol{\theta})$. We have:

$$\begin{aligned} \log f(\boldsymbol{z}) &= -\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{z}_i - \mathbf{0})^T \boldsymbol{I}(\boldsymbol{z}_i - \mathbf{0}) - \frac{N}{2}\ln|\boldsymbol{I}_R| - \frac{NJ}{2}\log(2\pi) \\ &= -\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{z}_i - \mathbf{0})^T \boldsymbol{I}(\boldsymbol{z}_i - \mathbf{0}) - \frac{N}{2}\ln|\boldsymbol{I}_R| + \text{const.} \end{aligned} \tag{2.36}$$

We notice that none of the parameters intervenes in the expression of the prior $f(\boldsymbol{z})$, meaning that its derivative with respect to $\boldsymbol{\theta}$ is null: $\log f(\boldsymbol{z})$ is irrelevant in the maximisation of the expectation of the complete-data log-likelihood.

E-STEP As the authors of Ghahramani and Hinton (1997), we focus on the conditional distribution $f(\boldsymbol{x}|\boldsymbol{z})$. That term is written as follows:

$$\log f(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta}) = -\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\Lambda}\boldsymbol{z}_i)^T\boldsymbol{\psi}^{-1}(x_i - \boldsymbol{\Lambda}\boldsymbol{z}_i) - \frac{N}{2}\ln|\boldsymbol{\psi}| - \frac{NJ}{2}\log(2\pi)$$
$$= -\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\Lambda}\boldsymbol{z}_i)^T\boldsymbol{\psi}^{-1}(x_i - \boldsymbol{\Lambda}\boldsymbol{z}_i) - \frac{N}{2}\ln|\boldsymbol{\psi}| + \text{const.} \tag{2.37}$$

In this context, the auxiliary function $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$ is defined as the expectation of $\log f(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ with respect to $p(\boldsymbol{z}|\boldsymbol{x})$:

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) &= \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\Big[-\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\Lambda}\boldsymbol{z}_i)^T\boldsymbol{\psi}^{-1}(\boldsymbol{x}_i - \boldsymbol{\Lambda}\boldsymbol{z}_i) - \frac{N}{2}\ln|\boldsymbol{\psi}| + \text{const}\Big] \\ &= -\frac{1}{2}\sum_{i=1}^{N}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\Big[(\boldsymbol{x}_i - \boldsymbol{\Lambda}\boldsymbol{z}_i)^T\psi^{-1}(\boldsymbol{x}_i - \boldsymbol{\Lambda}\boldsymbol{z}_i)\Big] - \frac{N}{2}\ln|\boldsymbol{\psi}| + \text{const} \\ &= -\frac{1}{2}\sum_{i=1}^{N}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{x}_i^T\psi^{-1}\boldsymbol{x}_i - \boldsymbol{z}_i^T\boldsymbol{\Lambda}^T\psi^{-1}\boldsymbol{x}_i - \boldsymbol{x}_i^T\psi^{-1}\boldsymbol{\Lambda}\boldsymbol{z}_i + \boldsymbol{z}_i^T\boldsymbol{\Lambda}^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\boldsymbol{z}_i\big] \\ &\quad - \frac{N}{2}\ln|\boldsymbol{\psi}| + \text{const.}\end{aligned} \tag{2.38}$$

The red term can be simplified since $\boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\boldsymbol{z}_i$ is a scalar; we have:

$$\boldsymbol{z}_i^T\boldsymbol{\Lambda}^T\boldsymbol{\psi}^{-1}\boldsymbol{x}_i + \boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\boldsymbol{z}_i = 2\boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\boldsymbol{z}_i.$$

The blue term simplifies too because of the trace trick:

$$\boldsymbol{z}_i^T\boldsymbol{\Lambda}^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\boldsymbol{z}_i = \text{tr}(\boldsymbol{\Lambda}^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\boldsymbol{z}_i\boldsymbol{z}_i^T).$$

By considering these simplifications and by applying the expectation only on the random parts of $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$ with respect to $p(\boldsymbol{z}|\boldsymbol{x})$, we can write the auxiliary function as follows:

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) = -\frac{1}{2}\sum_{i=1}^{N}\big(&\boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{x}_i - 2\boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i\big] \\ &+ \text{tr}(\boldsymbol{\Lambda}^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\big]\big) - \frac{N}{2}\ln|\boldsymbol{\psi}| + \text{const.}\end{aligned} \tag{2.39}$$

Therefore, the E-step requires computation of $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}_i|\boldsymbol{x}_i]$ and $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i]$. By definition, in the case of Gaussian multivariate, we have:

$$\begin{aligned}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}_i|\boldsymbol{x}_i] &= \mathbb{E}[\boldsymbol{z}_i] + \boldsymbol{\Lambda}^T(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \psi)^{-1}(\boldsymbol{x}_i - \mathbb{E}[\boldsymbol{x}_i]) \\ &= \boldsymbol{\Lambda}^T(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \psi)^{-1}\boldsymbol{x}_i &(2.40)\\ &= \boldsymbol{\beta}\boldsymbol{x}_i, &(2.41)\end{aligned}$$

with $\boldsymbol{\beta} = \boldsymbol{\Lambda}^T(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \psi)^{-1}$. Also by definition, we get:

$$\begin{aligned}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i] &= \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}_i|\boldsymbol{x}_i]\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i\big])^T + \mathrm{Var}(\boldsymbol{z}_i|\boldsymbol{x}_i)\\ &= \boldsymbol{\beta}\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{\beta}^T + \mathrm{Var}(\boldsymbol{z}_i|\boldsymbol{x}_i)\\ &= I - \boldsymbol{\Lambda}^T(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\psi})^{-1}\boldsymbol{\Lambda} + \boldsymbol{\beta}\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{\beta}^T\\ &= I - \boldsymbol{\beta}\boldsymbol{\Lambda} + \boldsymbol{\beta}\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{\beta}^T. \end{aligned} \tag{2.42}$$

M-STEP For the M-step, we split $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$ in several terms for convenience:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) = -\frac{1}{2}\sum_{i=1}^{n}\Big(\underbrace{\boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{x}_i}_{A}\underbrace{-2\boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i\big]}_{B} + \underbrace{\mathrm{tr}(\boldsymbol{\Lambda}^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\big])}_{C}\Big)\underbrace{-\frac{N}{2}\ln|\boldsymbol{\psi}|}_{D} + \text{const.}$$

Maximisation of $\mathcal{Q}$ with respect to $\boldsymbol{\Lambda}$.
The derivatives of A and D are null because they do not depend on $\boldsymbol{\Lambda}$. So,

$$\frac{\partial\mathcal{Q}}{\partial\boldsymbol{\Lambda}} = -\frac{1}{2}\sum_{i=1}^{N}(\frac{\partial B}{\partial\boldsymbol{\Lambda}} + \frac{\partial C}{\partial\boldsymbol{\Lambda}}).$$

We have:

$$\begin{aligned}\frac{\partial B}{\partial\boldsymbol{\Lambda}} &= \frac{\partial}{\partial\boldsymbol{\Lambda}}\Big(-2\boldsymbol{x}_i^T\boldsymbol{\psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i\big]\Big)\\ &= -2\boldsymbol{\psi}^{-1}\boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i\big]^T,\end{aligned}$$

because $\frac{\partial\boldsymbol{a}^T\boldsymbol{X}\boldsymbol{b}}{\partial\boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^T$ and $\boldsymbol{\psi}^{-1} = \boldsymbol{\psi}^{-1^T}$ since $\boldsymbol{\psi}$ is diagonal.

$$\begin{aligned}\frac{\partial C}{\partial\boldsymbol{\Lambda}} &= \frac{\partial}{\partial\boldsymbol{\Lambda}}\Big(\mathrm{tr}(\boldsymbol{\Lambda}^T\psi^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\big]\Big)\\ &= (\psi^{-1}\boldsymbol{\Lambda} + \psi^{-1}\boldsymbol{\Lambda})\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\big]\\ &= 2\psi^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\big],\end{aligned}$$

because $\frac{\partial\mathrm{tr}(\boldsymbol{X}^T\boldsymbol{M}\boldsymbol{X})}{\boldsymbol{X}} = \boldsymbol{M}\boldsymbol{X} + \boldsymbol{M}^T\boldsymbol{X}$. Therefore, we have to resolve:

$$\frac{\partial Q}{\partial\boldsymbol{\Lambda}} = -\sum_{i=1}^{N}\psi^{-1}\boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i\big]^T + \sum_{i=1}^{N}\psi^{-1}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\big] = \boldsymbol{0}.$$

We can then update $\boldsymbol{\Lambda}$ with:

$$\boldsymbol{\Lambda} = (\sum_{i=1}^{N}\boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i\big]^T)(\sum_{i=1}^{N}\mathbb{E}[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i])^{-1}. \tag{2.43}$$

Maximisation of $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$ with respect to ψ^{-1}.

The expression of $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$ contains more terms with $\boldsymbol{\psi}^{-1}$ than with $\boldsymbol{\psi}$. It is more convenient to derivate with respect to $\boldsymbol{\psi}^{-1}$:

$$\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\psi}^{-1}} = -\frac{1}{2}\sum_{i=1}^{N}\left(\frac{\partial A}{\partial \boldsymbol{\psi}^{-1}} + \frac{\partial B}{\partial \boldsymbol{\psi}^{-1}} + \frac{\partial C}{\partial \boldsymbol{\psi}^{-1}}\right) + \frac{\partial D}{\partial \boldsymbol{\psi}^{-1}}$$

$$\frac{\partial A}{\partial \boldsymbol{\psi}^{-1}} = \boldsymbol{x}_i \boldsymbol{x}_i^T,$$

since $\frac{\partial a^T X a}{\partial X} = aa^T$.

$$\frac{\partial B}{\partial \boldsymbol{\psi}^{-1}} = -2\boldsymbol{x}_i \left(\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i|\boldsymbol{x}_i\right]\right)^T,$$

since $\frac{\partial a^T X b}{\partial X} = ab^T$.

$$\frac{\partial C}{\partial \boldsymbol{\psi}^{-1}} = \boldsymbol{\Lambda}\left(\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\right]\right)^T,$$

since $\frac{\partial \mathrm{tr}(\boldsymbol{M}_1 \boldsymbol{X} \boldsymbol{M}_2)}{\partial \boldsymbol{X}} = \boldsymbol{M}_1^T \boldsymbol{M}_2^T$.

$$\frac{\partial D}{\partial \boldsymbol{\psi}^{-1}} = \frac{-N}{2}\frac{\partial \ln|\boldsymbol{\psi}|}{\partial \boldsymbol{\psi}^{-1}} = \frac{-N}{2}\frac{\partial \ln|\boldsymbol{\psi}\boldsymbol{\psi}|^{1/2}}{\partial \boldsymbol{\psi}^{-1}} = \frac{-N}{4}\frac{\partial \ln|\boldsymbol{\psi}\boldsymbol{\psi}|}{\partial \boldsymbol{\psi}^{-1}} = \frac{N}{2}\boldsymbol{\psi},$$

because $\boldsymbol{\psi}$ is diagonal, and we have $\frac{\partial \log|\boldsymbol{X}\boldsymbol{X}|}{\partial \boldsymbol{X}^{-1}} = -2\boldsymbol{X}^T = -2\boldsymbol{X}$.
Therefore, we have to find $\boldsymbol{\psi}$ such that:

$$\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\psi}^{-1}} = -\frac{1}{2}\sum_{i=1}^{N}\left(\boldsymbol{x}_i\boldsymbol{x}_i^T - 2\boldsymbol{x}_i\left(\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i|\boldsymbol{x}_i\right]\right)^T + \boldsymbol{\Lambda}\left(\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\right]\right)^T\right) - \frac{N}{2}\boldsymbol{\psi} = 0.$$

This leads to:

$$\boldsymbol{\psi} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T - \frac{1}{N}\sum_{i=1}^{N}2\boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i|\boldsymbol{x}_i\right]^T\boldsymbol{\Lambda}^T + \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\right]^T\boldsymbol{\Lambda}^T. \quad (2.44)$$

Furthermore, from $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\right]^T = \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\right]$ because $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\right]$ is symmetric, and from Equation (2.43), we have:

$$\boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i\right]^T = \boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i|\boldsymbol{x}_i\right].$$

So, we get:

$$\begin{aligned}\boldsymbol{\psi} &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T - \frac{1}{N}\sum_{i=1}^{N}2\boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i|\boldsymbol{x}_i\right]^T\boldsymbol{\Lambda}^T + \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i|\boldsymbol{x}_i\right]^T\boldsymbol{\Lambda}^T \\ &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T - \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}\left[\boldsymbol{z}_i|\boldsymbol{x}_i\right]^T\boldsymbol{\Lambda}^T. \quad (2.45)\end{aligned}$$

To constrain $\boldsymbol{\psi}$ to be diagonal, we impose:

$$\boldsymbol{\psi} = \frac{1}{N}\text{diag}\{\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T - \boldsymbol{\Lambda}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}_i|\boldsymbol{x}_i]\boldsymbol{x}_i^T\}, \tag{2.46}$$

where $\text{diag}(\boldsymbol{M})$ means that all the off-diagonal elements of the matrix $\boldsymbol{M}$ are set to zero.

CONVERGENCE The EM-algorithm is numerically stable (McLachlan and Krishnan, 1997) and has good convergence properties in that they ensure that the likelihood is not decreased after each iteration. However, it is also known to be slow to converge. This could be due to the typically large fraction of latent variables.

2.2.4 Unidentifiability

In Factor Analysis, the parameters are unidentifiable. To see this, it is necessary to consider an arbitrary orthogonal matrix $\boldsymbol{Q}$ of dimensions $(R \times R)$, satisfying $\boldsymbol{Q}\boldsymbol{Q}^T = I$. Let us define $\widetilde{\boldsymbol{\Lambda}} = \boldsymbol{\Lambda}\boldsymbol{Q}$. Then, the likelihood function of this model is the same since:

$$\begin{aligned}\text{cov}(\boldsymbol{x}) &= \widetilde{\boldsymbol{\Lambda}}\widetilde{\boldsymbol{\Lambda}}^T + \boldsymbol{\psi} \\ &= \boldsymbol{\Lambda}\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{\Lambda}^T = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T.\end{aligned} \tag{2.47}$$

Therefore, multiplying $\boldsymbol{\Lambda}$ by an orthogonal matrix is like rotating $\boldsymbol{z}$ before generating $\boldsymbol{x}$ but, since $\boldsymbol{z}$ is distributed according to an isotropic Gaussian, it makes no difference to the likelihood. We can not uniquely identify $\boldsymbol{\Lambda}$ and, therefore, we can not uniquely identify the latent scores $\boldsymbol{z}$ either. The common solutions to tackle this problem are listed below:

- To constrain $\boldsymbol{\Lambda}$ so that it is orthonormal and to order the columns by decreasing variance of the corresponding latent scores. This approach is used in the case of Principal Component Analysis.
- To constrain $\boldsymbol{\Lambda}^T\boldsymbol{\psi}\boldsymbol{\Lambda}$ to be diagonal.
- To constrain $\boldsymbol{\Lambda}$ so that it is a lower triangular full rank matrix with diagonal elements strictly positive. This form is used, for instance, in Geweke and Zhou (1996) and Aguilar and West (2000). It provides identification and often useful interpretation of the model.
- To choose an informative rotation matrix. Many heuristic methods try to find rotation matrices $\boldsymbol{P}$ that can be used to modify $\boldsymbol{\Lambda}$ (and the latent factors) so as to try to increase the interpretability. Among them, *varimax* is a well-known method for this purpose (Kaiser, 1958).

2.2.5 Mixture of Factor Analysers

The Mixture of Factor Analysers (G.J. McLachlan and Bean, 2003) is an extension of the factor analysis model to a mixture with G factor analysers. We assume that $\boldsymbol{x}_i$ can be expressed from a latent random vector $\boldsymbol{z}_i \in \mathbb{R}^R$ such that $R \leq J$. In addition, the latent labels $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ are assumed to be independent unobserved realisations of a categorical G-dimensional random vector, such that $v_{ig} = 1$ indicates that $\boldsymbol{x}_i$ is generated by the gth factor analyser. The generative process of this model is as follows:

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

where $\boldsymbol{\Lambda}_g$ is a $J \times R$ matrix and $\boldsymbol{\eta}_g \in \mathbb{R}^J$ is the mean vector of the gth factor analyser. In addition, $\boldsymbol{\psi}_g$ is a diagonal square matrix of dimension J, and $\boldsymbol{\pi} = (\pi_g)_g$ are the mixing proportions. The latent variable $\boldsymbol{v}_i$ represents the partitions and $p(v_{ig} = 1) = \pi_g$. The marginal density of $\boldsymbol{x}_i$ is written as follows:

$$
f(\boldsymbol{x}_i; \boldsymbol{\theta}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\eta}_g, \boldsymbol{\Sigma}_g), \tag{2.48}
$$

such that:

- $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_G)$ are the mixing proportions,
- $f_g(.; \boldsymbol{\eta}_g, \boldsymbol{\Sigma}_g)$ is the Gaussian probability density function of parameters $\boldsymbol{\eta}_g$ and $\boldsymbol{\Sigma}_g$ and
- $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^T + \boldsymbol{\psi}_g$.

Hence, the observed-data log-likelihood $l_o(\boldsymbol{\theta}; \boldsymbol{x})$, is given by:

$$
l_o(\boldsymbol{\theta}; \boldsymbol{x}) = \sum_{i=1}^{N} \log \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\eta}_g, \boldsymbol{\Sigma}_g). \tag{2.49}
$$

Let us note that the conditional distribution of $\boldsymbol{x}$ is also Gaussian:

$$
p(\boldsymbol{x}_i|\boldsymbol{z}_i, v_{ig} = 1) = \mathcal{N}(\boldsymbol{\eta}_g + \boldsymbol{\Lambda}_g \boldsymbol{z}_i, \boldsymbol{\psi}_g). \tag{2.50}
$$

2.2.6 Inference of the Mixture of Factor Analysers

Different kinds of EM-algorithms were proposed for the MFA model. Originally, the authors of G.J. McLachlan and Bean (2003) proposed an Alternated Expectation Conditional Maximisation algorithm (referred to as "AECM algorithm"). Ghahramani and Hinton (1997) proposed an EM-algorithm for the MFA, but for the special case with $\boldsymbol{\psi}_g = \boldsymbol{\psi}$. Zhao and Yu (2008) developed an Expectation Conditional Maximisation algorithm (referred to as "ECM algorithm") to reduce the execution time. In this section, we develop the EM-algorithm in the same fashion as Ghahramani and Hinton (1997), but we do not apply the equality constraint on matrix $\boldsymbol{\psi}_g$ in order to keep the model as comprehensive as possible. The latent variables in this model are the scores $\boldsymbol{z}$ and the partitions $\boldsymbol{v}$.

COMPUTING THE EXPECTATION The latent variables are $\boldsymbol{z}$ and $\boldsymbol{v}$. Therefore, the E-step is made of two steps (one for each latent variable). The complete data log-likelihood is:

$$
l_c(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{z}) = \log p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{v}) + \log p(\boldsymbol{z}|\boldsymbol{v}) + \log p(\boldsymbol{v}),
$$

where $p(\boldsymbol{z}|\boldsymbol{v}) = p(\boldsymbol{z})$ because $\boldsymbol{z}$ does not depend on $\boldsymbol{v}$. With a similar reasoning as in Section 2.2.3, Equation 2.36, we get rid of the term $p(\boldsymbol{z}|\boldsymbol{v})$ and write :

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) = \mathbb{E}_{p(\boldsymbol{z},\boldsymbol{v}|\boldsymbol{x})}\Big[\sum_{i=1}^{N}\big(\log p(\boldsymbol{x}_i|\boldsymbol{z}_i, \boldsymbol{v}_i) + \log p(\boldsymbol{v}_i)\big)\Big].$$

We can develop:

$$\begin{aligned}
\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) &= \mathbb{E}_{p(\boldsymbol{z},\boldsymbol{v}|\boldsymbol{x})}\Big[\sum_{i=1}^{N}\log\big(p(\boldsymbol{x}_i|\boldsymbol{z}_i, \boldsymbol{v}_i)p(\boldsymbol{v}_i)\big)\Big] \\
&= \sum_{i=1}^{N}\mathbb{E}_{p(\boldsymbol{z},\boldsymbol{v}|\boldsymbol{x})}\Big[\log\prod_{g=1}^{G}\big(p(\boldsymbol{x}_i|\boldsymbol{z}_i, v_{ig}=1)p(v_{ig}=1)\big)^{v_{ig}}\Big] \\
&= \sum_{i=1}^{N}\sum_{g=1}^{G}\mathbb{E}_{p(\boldsymbol{z},\boldsymbol{v}|\boldsymbol{x})}\Big[v_{ig}\log\big(p(\boldsymbol{x}_i|\boldsymbol{z}_i, v_{ig}=1)p(v_{ig}=1)\big)\Big] \\
&= \sum_{i=1}^{N}\sum_{g=1}^{G}\mathbb{E}_{p(\boldsymbol{v}|\boldsymbol{x})}\Big[v_{ig}\Big]\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\Big[\log\big(p(\boldsymbol{x}_i|\boldsymbol{z}_i, v_{ig}=1)p(v_{ig}=1)\big)\Big] \\
&= \sum_{i=1}^{N}\sum_{g=1}^{G}t_{ig}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\Big[\log\big(p(\boldsymbol{x}_i|\boldsymbol{z}_i, v_{ig}=1)p(v_{ig}=1)\big)\Big],
\end{aligned}$$

with $t_{ig} \propto p(\boldsymbol{x}_i|v_{ig}=1)p(v_{ig}=1)$. Therefore, we have:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) = \sum_{i=1}^{N}\sum_{g=1}^{G}t_{ig}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\Big[\text{const} - \frac{1}{2}\ln|\boldsymbol{\psi}_g| - \frac{1}{2}\big(\boldsymbol{x}_i - \boldsymbol{\eta}_g - \boldsymbol{\Lambda}_g\boldsymbol{z}_i\big)^T\boldsymbol{\psi}_g^{-1}\big(\boldsymbol{x}_i - \boldsymbol{\eta}_g - \boldsymbol{\Lambda}_g\boldsymbol{z}_i\big)\Big].$$

If we get rid of the term const and develop:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{g=1}^{G}t_{ig}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\Big[\ln|\boldsymbol{\psi}_g| + \big(\boldsymbol{x}_i - \boldsymbol{\eta}_g - \boldsymbol{\Lambda}_g\boldsymbol{z}_i\big)^T\boldsymbol{\psi}_g^{-1}\big(\boldsymbol{x}_i - \boldsymbol{\eta}_g - \boldsymbol{\Lambda}_g\boldsymbol{z}_i\big)\Big].$$

The EM-algorithm for the MFA model is as follows for iteration q (see Appendix 2.4.2 for more details):

(A) First E-step Compute $t_{ig}^{(q)} = \mathbb{E}_{p(\boldsymbol{v}|\boldsymbol{x})}\big[\boldsymbol{v}_i|\boldsymbol{x}_i\big]$ for each data point $\boldsymbol{x}_i$, given $\boldsymbol{\Lambda}^{(q-1)}$ and $\boldsymbol{\psi}^{(q-1)}$.

(B) Second E-step Compute $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\big[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{v}_i\big]$ and $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i, \boldsymbol{v}_i\big]$ for each data point $\boldsymbol{x}_i$, given $\boldsymbol{\Lambda}_g^{(q-1)}$ and $\boldsymbol{\psi}_g^{(q-1)}$.

$$\begin{aligned}
\mathbb{E}[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{v}_i] &= t_{ig}^{(q)} \times (\boldsymbol{\Lambda}_g^{(q-1)^T}(\boldsymbol{\Lambda}_g^{(q-1)}\boldsymbol{\Lambda}_g^{(q-1)^T} + \boldsymbol{\psi}_g^{(q-1)})^{-1})(\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q-1)}) \\
&= t_{ig}^{(q)}\boldsymbol{\beta}_g(\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q-1)}),
\end{aligned}$$

with $\boldsymbol{\beta}_g = \boldsymbol{\Lambda}_g^{(q-1)^T}(\boldsymbol{\Lambda}_g^{(q-1)}\boldsymbol{\Lambda}_g^{(q-1)^T} + \boldsymbol{\psi}_g^{(q-1)})^{-1}$.

$$\mathbb{E}[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i, \boldsymbol{v}_i] = t_{ig}^{(q)}(I - \boldsymbol{\beta}_g\boldsymbol{\Lambda}_g^{(q-1)} + \boldsymbol{\beta}_g(\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q-1)})(\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q-1)})^T\boldsymbol{\beta}_g^T).$$

(C) M STEP We update the parameters as follows:

$$\boldsymbol{\eta}_g^{(q)} = \frac{\sum_{i=1}^N t_{ig}^{(q)}\left(\boldsymbol{x}_i - \boldsymbol{\Lambda}_g^{(q-1)}\mathbb{E}\left[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{v}^{(q)}\right]\right)}{\sum_{i=1}^N t_{ig}^{(q)}},$$

$$\boldsymbol{\Lambda}_g^{(q)} = \frac{\sum_{i=1}^N t_{ig}^{(q)}\left((\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q)})\mathbb{E}\left[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{v}\right]^T\mathbb{E}\left[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i, \boldsymbol{v}^{(q)}\right]^{-1}\right)}{\sum_{i=1}^N t_{ig}^{(q)}},$$

$$\boldsymbol{\psi}_g^{(q)} = \frac{\sum_{i=1}^N t_{ig}^{(q)}\left((\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q)})(\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q)})^T - (\boldsymbol{x}_i - \boldsymbol{\eta}_g^{(q)})\mathbb{E}\left[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{v}^{(q)}\right]^T\boldsymbol{\Lambda}_g^{(q)^T}\right)}{\sum_{i=1}^N t_{ig}^{(q)}},$$

$$\pi_g^{(q)} = \frac{1}{N}\sum_{i=1}^N t_{ig}^{(q)}.$$

2.2.7 Specific MFA models

Mixture of factor analysers are widely used to perform clustering and dimension reduction at the same time. In the last decades, many models have been built and based on the model presented in Section 2.2.5. As mentioned above, Ghahramani and Hinton (1997) propose an EM-algorithm for a very similar model, where $\boldsymbol{\psi}_g = \boldsymbol{\psi}$, which makes the model more parsimonious. The mixture of probabilistic principal component analyis (Tipping and Bishop, 1999b) is also a special case of the mixture factor analysers, where $\boldsymbol{\psi}_g = \boldsymbol{\psi}$ and $\boldsymbol{\psi} = \sigma\boldsymbol{I}$ so that the noise covariance is considered isotropic. A general framework that includes all these models was proposed in McNicholas and Murphy (2008). The authors created a family of models known as the parsimonious Gaussian mixture model (PGMM) family. The PGMM family is composed of 8 models that are coded with three letters that take the value "U" (unconstrained) or "C" (constrained). The first letter refers to the loading matrix, which can be common to all groups $\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$ (C..) or not (U..). The second letter indicates if the noise covariance matrix is common to all groups $\boldsymbol{\psi}_g = \boldsymbol{\psi}$ (.C.) or not (.U.). The third letter refers to the noise covariance matrix, which can be isotropic $\boldsymbol{\psi}_g = \sigma_g I$ (..C) or not (..U). For example, the mixture of PPCA is the model UCC, because the loading matrices are different for each group (U..), the noise covariance is isotropic (..C) and common to all groups (.C.).

The authors of Baek et al. (2010) proposed the MCFA (Mixture of Common Factor Analysers) and constrained the model so that the parameters of the resulting GMM share a common matrix. That is to say, in the standard MFA we have:

$$\boldsymbol{\mu}_g = \boldsymbol{\eta}_g \text{ and } \boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^T + \boldsymbol{\psi}_g;$$

but in the MCFA we have:

Model Name	covariance structure	# of parameters
MFA-M	$\mathbf{\Lambda}_g\mathbf{\Lambda}_g^T + \boldsymbol{\psi}_g$	$(G-1) + GD + GJ + GR(J-(R-1)/2) + GJ$
MFA-G	$\mathbf{\Lambda}_g\mathbf{\Lambda}_g^T + \boldsymbol{\psi}$	$(G-1) + GD + GJ + GR(J-(R-1)/2) + J$
PPCA	$\mathbf{\Lambda}_g\mathbf{\Lambda}_g^T + \sigma\boldsymbol{I}$	$(G-1) + GD + GJ + GR(J-(R-1)/2) + 1$
MCFA	$AD_gA^T + \boldsymbol{\psi}$	$(G-1) + JR - R(R+1)/2 + GR + GR(R+1)/2 + J$

Table 2.2 – Number of parameters for different MFA models. MFA-G corresponds to MFA described in Ghahramani and Hinton (1997) and MFA-M corresponds to the MFA described in G.J. McLachlan and Bean (2003).

$$\boldsymbol{\mu}_g = A\boldsymbol{\zeta}_g \text{ and } \mathbf{\Sigma}_g = AD_gA^T + \boldsymbol{\psi},$$

such that A is a $J \times R$ orthonormal matrix ($A^TA = I$), $\boldsymbol{\zeta}_g$ is a R-dimensional vector, D_g is an $R \times R$ positive definite symmetric matrix and $\boldsymbol{\psi}$ a diagonal $J \times J$ matrix. This model aims both to lower the complexity of the MFA model and to ease the visualisation of the clustered data.

2.2.8 Model selection

The BIC criterion applies to Factor Analyis models. As in Section 2.1, it is equal to the penalised quantity:

$$\text{BIC} = -2\log p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) + \nu \log N.$$

and we refer to Table 2.2 to get the number of parameters ν of some of the FA models mentioned above.

2.2.9 Conclusion

In this section, we have reviewed the basics for factor analysis and the mixture of factor analysers. These notions will be helpful in the contribution of Chapter 5 where we will study the Deep GMM, an approach that consists in stacking MFA layers in a deep learning fashion.

2.3 Model-based co-clustering

2.3.1 Introduction

Finite Mixture Models (as described in Section 2.1) are a popular tool for performing clustering. Since the clustering algorithms bring out groups of rows into the data set, they also highlight a structure inherent to the data. However, the recent "big-data" phenomenon has greatly increased the number of features, leading to the emergence of high-dimensional data sets. The analysis of a cluster relies on the representative of the cluster (mean, mode,...). However, in high-dimensional contexts, this representative is described by a large number of features that makes the clustering more difficult to interpret and makes the summary of the data set less useful. From this consideration comes the need to also "summarise" the features, which can be done by gathering them into clusters in parallel with the usual clustering of observations. Co-clustering methods seem to be a good option for performing this task because they perform clustering of rows and columns simultaneously. Figure 2.2 illustrates well the concept of co-clustering (Govaert and Nadif, 2013). In Image (1), a binary dataset

is shown and it is composed of ten rows and seven columns. In Image (2), a clustering is performed (some rows are swapped in order to gather the ones that are similar) and we can perceive three clusters among these rows. In Image (3), a co-clustering is performed (the rows *and* the columns are swapped) and we perceive three clusters in lines and three clusters in columns. The crossing of a row-cluster and of a column-cluster is called a block. The blocks highlight a structure in the data that was not necessarily expected before; the dataset can be summarised as in Image (4).

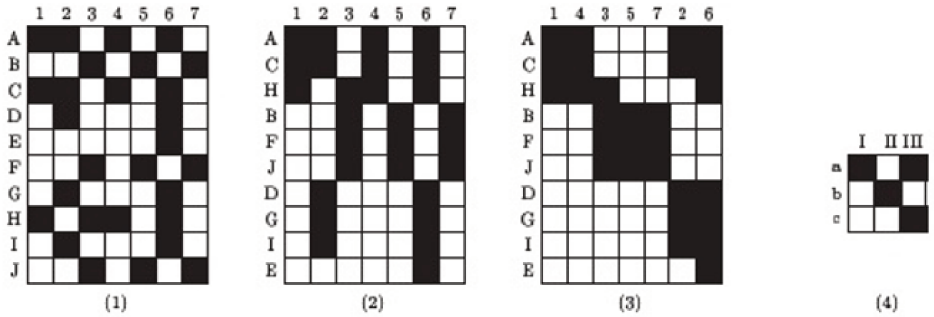

Figure 2.2 – Simple example of co-clustering with binary data from Govaert and Nadif (2013).

Co-clustering techniques can also be seen as an efficient alternative method to selecting variables thanks to its parsimony, especially in very high dimensions. In addition, it can produce interpretable sets of variables since it can group redundant variables or noisy variables. In this way, a first naive answer is to manually select the informative blocks, but Ailem et al. (2017) alternatively defines a model that automatically distinguishes the informative blocks for textual data sets. For mixed data, variable selection is more challenging. McParland et al. (2017) performs clustering while incorporating variable selection and this method can produce homogeneous row-clusters. However, compared to co-clustering, it does not provide interpretable column-clusters, which may be essential for the summary of the data set, particularly when there is a large number of variables.

Among the most famous co-clustering techniques, the Non-Negative Matrix Trifactorisation consists in factoring the $N \times J$ data matrix $\boldsymbol{x}$ into three matrices $\boldsymbol{a}$ $(N \times G)$, $\boldsymbol{b}$ $(G \times H)$ and $\boldsymbol{c}$ $(H \times J)$ with the constraint that all three matrices have non-negative elements (Buono and Pio, 2015). More specifically, the approximation of $\boldsymbol{x}$ by $\boldsymbol{x} \approx \boldsymbol{abc}$ is achieved by minimising the error function $\min_{(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})} ||\boldsymbol{x} - \boldsymbol{abc}||$ with the constraint that all elements of $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are greater than or equal to 0, and $||.||$ being a matrix norm to be chosen (e.g. the Frobenius norm). The matrix $\boldsymbol{b}$ represents the *block* matrix: an element b_{gh} of $\boldsymbol{b}$ summarises the observations belonging to row-cluster g and column-cluster h. Despite the non-negative property of the matrices, it is not always easy to interpret the resulting matrices. For instance, matrices $\boldsymbol{a}$ and $\boldsymbol{c}$ are not always normalised, which makes it difficult to interpret them in terms of rows and columns belonging to corresponding clusters. Furthermore, this technique depends on the choice of the distance measure. Conversely, probabilistic approaches propose normalised membership matrices and do not require the user to choose a particular distance measure. In the Latent Block Model, referred to as "LBM" (Govaert and Nadif, 2013), the elements of a block are modelled by a parametric distribution. Therefore, the results give more information than a simple scalar, as mentioned in the previous methods. Each block is, then, interpretable via the parameters of the block-distribution. Moreover, model selection

criterion such as the ICL criterion (Biernacki et al., 2000) can be used for model selection purposes including the choice of the number of co-clusters. This technique has proved its efficiency in the co-clustering of several types of data: continuous (Nadif and Govaert, 2008), nominal (Singh Bhatia et al., 2017), binary (Laclau et al., 2017), ordinal (Jacques and Biernacki, 2018) and functional data (Slimen et al., 2018; Bouveyron et al., 2018).

2.3.2 The Latent Block Model

The Latent Block Model (LBM) is a widely used model to perform co-clustering (Govaert and Nadif, 2013). Basically, it assumes that all elements of a block follow the same distribution. In this section, the assumptions used for the LBM are defined and the mathematical details are given.

Consider the data matrix $\boldsymbol{x} = (x_{ij})_{i,j}$, where $i \in \{1, \ldots, N\}$ is the row (observation) index and $j \in \{1, \ldots, J\}$ is the column (feature) index. It is assumed that there are G row-clusters and H column-clusters that correspond to a partition $\boldsymbol{v} = (v_{ig})_{i,g}$ of the rows, and a partition $\boldsymbol{w} = (w_{jh})_{j,h}$ of the columns with $1 \leq g \leq G$ and $1 \leq h \leq H$, where v_{ig} is equal to 1 if row i belongs to cluster g or to 0 otherwise. Similarly, w_{jh} is equal to 1 when column j belongs to cluster h, and to 0 otherwise.

The first LBM assumption is that the univariate random variables x_{ij} are conditionally independent given the row and column partitions $\boldsymbol{v}$ and $\boldsymbol{w}$. Therefore, the conditional probability density function of $\boldsymbol{x}$ given $\boldsymbol{v}$ and $\boldsymbol{w}$ is written:

$$p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\alpha}) = \prod_{i,j,g,h} p(x_{ij}; \alpha_{gh})^{v_{ig} w_{jh}}, \tag{2.51}$$

where $\boldsymbol{\alpha} = (\alpha_{gh})_{g,h}$ are the distribution parameters of block (g, h).

The second LBM assumption is that the latent variables $\boldsymbol{v}$ and $\boldsymbol{w}$ are independent so $p(\boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\pi}, \boldsymbol{\rho}) = p(\boldsymbol{v}; \boldsymbol{\pi}) p(\boldsymbol{w}; \boldsymbol{\rho})$ with:

$$p(\boldsymbol{v}; \boldsymbol{\pi}) = \prod_{i,g} \pi_g^{v_{ig}} \quad \text{and} \quad p(\boldsymbol{w}; \boldsymbol{\rho}) = \prod_{j,h} \rho_h^{w_{jh}},$$

where $\pi_g = p(v_{ig} = 1)$ and $\rho_h = p(w_{jh} = 1)$. This implies that, for all i, the distribution of $\boldsymbol{v}_i$ is the multinomial distribution $\mathcal{M}(\pi_1, \ldots, \pi_G)$ and it does not depend on i. Similarly, for all j, the distribution of $\boldsymbol{w}_j$ is the multinomial distribution $\mathcal{M}(\rho_1, \ldots, \rho_H)$ and it does not depend on j.

From these considerations, the LBM parameter is defined as $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\alpha})$, where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_G)$ and $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_H)$ are respectively the rows and columns mixing proportions. Therefore, if $\mathcal{V}$ and $\mathcal{W}$ are the sets of all possible labels $\boldsymbol{v}$ and $\boldsymbol{w}$, the probability density function of $\boldsymbol{x}$ is written:

$$p(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{(\boldsymbol{v}, \boldsymbol{w}) \in \mathcal{V} \times \mathcal{W}} \prod_{i,g} \pi_g^{v_{ig}} \prod_{j,h} \rho_h^{w_{jh}} \prod_{i,j,g,h} p(x_{ij}; \alpha_{gh})^{v_{ig} w_{jh}}. \tag{2.52}$$

2.3.3 Inference of the Latent Block Model

As explained in Section 2.1, the EM-algorithm is a well-known method for estimating parameters with latent variables in the model. In the expectation step, the EM algorithm requires computing the auxiliary function which is the expectation of the complete data

log-likelihood. In the Latent Block Model framework, the complete data log-likelihood can be written as follows:

$$\begin{aligned} l_c(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}) &= \log p(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\theta}) \\ &= \log p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{w}) + \log p(\boldsymbol{v}) + \log p(\boldsymbol{w}) \\ &= \sum_{i,j,g,h} \log p(x_{ij}|v_{ig}, w_{jh}; \boldsymbol{\theta})^{v_{ig}w_{jh}} \sum_{i,g} \log p(v_{ig}; \boldsymbol{\theta})^{v_{ig}} \sum_{j,h} \log p(w_{jh}; \boldsymbol{\theta})^{w_{jh}}. \end{aligned} \tag{2.53}$$

From Equation (2.53), we can write the auxiliary function $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$ as follows:

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) = &\sum_{i,g} p(v_{ig} = 1|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}) \log \pi_g + \\ &\sum_{j,h} p(w_{jh} = 1|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}) \log \rho_h + \\ &\sum_{i,j,g,h} p(v_{ig}w_{jh} = 1|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}) \log p(x_{ij}|v_{ig}w_{jh} = 1). \end{aligned} \tag{2.54}$$

This expression contains the probability $p(v_{ig}w_{jh} = 1|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)})$ which needs to consider all possible values for $\boldsymbol{v}_{i'}$ and $\boldsymbol{w}_{j'}$ with $i' \neq i$ and $j' \neq j$. The E-step would require calculation of $G^N \times H^J$ terms: for example, if $G = 2$, $H = 2$, $N = 20$ and $J = 20$; each E-step of the EM algorithm would need to compute $2^{20} \times 2^{20} \approx 10^{12}$ terms, which is not feasible. In 2.3.3.a and 2.3.3.b, we describe two variants of the EM algorithm to tackle this issue.

2.3.3.a Variational EM algorithm

In Nadif and Govaert (2008), the authors propose using a variational approximation for the distribution $p(\boldsymbol{v}, \boldsymbol{w}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)})$. Let us consider a free distribution $q(\boldsymbol{v}, \boldsymbol{w})$. Similar to Equation (2.70), we can write the observed data log-likelihood as the sum of two terms:

$$l_o(\boldsymbol{\theta}; \boldsymbol{x}) \log p(\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q||p), \tag{2.55}$$

where

- $\mathcal{L}(q, \boldsymbol{\theta})$ represents the expectation of the complete data log-likelihood with respect to the distribution $q(\boldsymbol{v}, \boldsymbol{w})$:

$$\mathbb{E}_{q(\boldsymbol{v},\boldsymbol{w})}\left[\log\left(\frac{p(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\theta})}{q(\boldsymbol{v}, \boldsymbol{w})}\right)\right], \tag{2.56}$$

- $\mathrm{KL}(q||p)$ is the KL divergence between $q(\boldsymbol{v}, \boldsymbol{w})$ and $p(\boldsymbol{v}, \boldsymbol{w}|\boldsymbol{x}; \boldsymbol{\theta})$.

E-STEP Once again, since $\mathrm{KL}(q||p)$ is greater or equal to 0, the first term of the RHS of Equation (2.55) is a lower bound of the log-likelihood, and it is equal to the log-likelihood if and only if the distribution $q(\boldsymbol{v}, \boldsymbol{w})$ is equal to the distribution $p(\boldsymbol{v}, \boldsymbol{w}|\boldsymbol{x}; \boldsymbol{\theta})$. However, as mentioned above, this term is not tractable. In variational inference, the mean-field approximation assumes that:

$$q(\boldsymbol{v}, \boldsymbol{w}) = q(\boldsymbol{v})q(\boldsymbol{w}).$$

This results in an independence of the partitions given the data. Therefore, the lower bound to maximise at the E-step is :

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) = &\sum_{i,g} t_{ig}^{(q)} \log \pi_g + \sum_{j,h} s_{jh}^{(q)} \log \rho_h + \sum_{i,j,g,h} t_{ig}^{(q)} s_{jh}^{(q)} \log p(x_{ij}|\boldsymbol{\alpha}_{gh}) \\ &- \sum_{i,g} t_{ig}^{(q)} \log t_{ig}^{(q)} - \sum_{j,h} s_{jh}^{(q)} \log s_{jh}^{(q)},\end{aligned} \tag{2.57}$$

where

$$t_{ig}^{(q)} = \frac{\pi_g^{(q-1)} \prod_{h=1}^{H} f(\boldsymbol{x}_i; \boldsymbol{\alpha}_{gh}^{(q-1)})}{\sum_{g'=1}^{G} \pi_{g'}^{(q-1)} \prod_{h=1}^{H} f(\boldsymbol{x}_i|\boldsymbol{\alpha}_{g'h}^{(q-1)})} \text{ and} \tag{2.58}$$

$$s_{jh}^{(q)} = \frac{\rho_h^{(q-1)} \prod_{g=1}^{G} f(\boldsymbol{x}_i; \boldsymbol{\alpha}_{gh}^{(q-1)})}{\sum_{h'=1}^{H} \rho_{h'}^{(q-1)} \prod_{g=1}^{G} f(\boldsymbol{x}_i|\boldsymbol{\alpha}_{gh'}^{(q-1)})}. \tag{2.59}$$

M-STEP The M-step consists in updating the co-cluster parameters $\boldsymbol{\theta}^{(q)}$ to maximise the complete data log-likelihood. Once again, the Lagrange multipliers are used to update the row mixing proportions by:

$$\pi_g^{(q)} = \frac{1}{N} \sum_{i=1}^{N} t_{ig}^{(q)}, \tag{2.60}$$

and the column mixing proportions by:

$$\rho_h^{(q)} = \frac{1}{J} \sum_{j=1}^{J} s_{jh}^{(q)}. \tag{2.61}$$

Likewise, the parameters $\boldsymbol{\alpha}^{(q)}$ are updated by finding the root of:

$$\sum_{i=1}^{N} \sum_{g=1}^{G} \sum_{j=1}^{J} \sum_{h=1}^{H} t_{ig}^{(q)} s_{jh}^{(q)} \frac{\partial \log f_{gh}(\boldsymbol{x}_{ij}; \boldsymbol{\alpha}_{gh})}{\partial \boldsymbol{\alpha}}. \tag{2.62}$$

Let us recall that the parameters $\boldsymbol{\alpha}$ depend on the type of the data matrix $\boldsymbol{x}$. For instance, with Gaussian data we have $\boldsymbol{\alpha} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

CONVERGENCE The term of Equation (2.56) is a lower bound of the log-likelihood, whose relevancy depends on the KL divergence which cannot be estimated. Therefore, we need to know if:

- the parameters $\boldsymbol{\theta}$ that maximise (2.56) are close to the parameters that maximise the log-likelihood,
- the maximum value of (2.56) is close to the maximum of the log-likelihood.

For classical finite mixture models, Keribin (2009) lists several results regarding the matter. Wang and Titterington (2004) show that, in the context of FMM with simple exponential distribution family, the estimator obtained through the mean-field estimator converges to the value of the true parameters. However, Humphreys and Titterington (2000) show that the approximation is poor when the number of observations is not high enough. These experimental results and the convergence of the partition shown in Mariadassou and Matias (2015) let us think that the mean-field approximation will yield satisfying results asymptotically.

NUMBER OF ITERATIONS The VEM-algorithm is deterministic and will increase the approximation of the lower bound of the log-likelihood. To be able to know when to stop the algorithm, the user should fix a threshold ϵ, and the algorithm is stopped whenever the difference between this approximation at iteration q and at iteration $q-1$ is lower than ϵ. In practice, users also fix a maximum number of iterations equal to *nb.iter* so that the algorithm is stopped when $q = nb.iter$ even if the difference is not lower than ϵ. We denote the iteration in which one of the two conditions is satisfied as $q.final$. The final estimation of the partitions $\hat{\boldsymbol{v}}$ (or $\hat{\boldsymbol{w}}$) is equal to $\boldsymbol{v}^{(q.final)}$ (or $\boldsymbol{w}^{(q.final)}$) and the final estimation of the parameters $\hat{\boldsymbol{\theta}}$ is $\boldsymbol{\theta}^{(q.final)}$.

CONCLUSIONS ON VEM ALGORITHM The mean-field variational approaches have many advantages. First, the computation of $t_{ig}^{(q)}$ and $s_{jh}^{(q)}$ is easy. Second, the VEM algorithm converges fast to a local maximum. The drawback of this deterministic approach is that it is strongly dependent on the initialisation of the algorithm. Moreover, a stationary point in this algorithm can only be a stationary point in the log-likelihood if the model satisfies the conditions of the variational approximation. This can be true in some asymptotical conditions, but it is not always the case with finite distance. In Keribin et al. (2010), the authors propose the SEM-Gibbs algorithm which simulates the density $p(\boldsymbol{v}, \boldsymbol{w}|\boldsymbol{x}; \boldsymbol{\theta})$ using a Gibbs sampler (Gelfand and Smith, 1990) instead of approaching it.

2.3.3.b Stochastic EM-Gibbs algorithm

The SEM-Gibbs algorithm is another option to tackle the difficulties regarding the computation of $p(v_{ig}w_{jh} = 1|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)})$. In the VEM algorithm of section 2.3.3.a, this density is approximated. With the SEM-Gibbs algorithm, another strategy is considered. It aims at estimating the partitions $(\boldsymbol{v}, \boldsymbol{w})$ using a Gibbs sampler. The Gibbs sampler consists in sampling $v_{ig}^{(q)}$ and $w_{jh}^{(q)}$ according to the quantities $p(v_{ig}^{(q)}|\boldsymbol{x}, \boldsymbol{w}^{(q-1)}; \boldsymbol{\theta}^{(q-1)})$ and $p(w_{jh}^{(q)}|\boldsymbol{x}, \boldsymbol{v}^{(q)}; \boldsymbol{\theta}^{(q-1)})$ respectively. Therefore, we repeat the following Expectation and Maximisation steps *nb.iter* times.

E-STEP The estimation step is replaced by the generation of partitions through a Gibbs sampler. So the algorithm repeats *n.iter.gibbs* times:

- Sample the row partitions $\boldsymbol{v}_i$ for all $i \in \{1, \dots, N\}$:

$$\boldsymbol{v}_i^{(q)} \sim \mathcal{M}(1, \boldsymbol{t}_i^{(q)}),$$

  with $\boldsymbol{t}_i^{(q)}$ a G-dimensional vector such that:

$$t_{ig}^{(q)} = \frac{\pi_g^{(q-1)} \prod_{h=1}^{H} f(\boldsymbol{x}_i; \boldsymbol{\alpha}_{gh}^{(q-1)})}{\sum_{g'=1}^{G} \pi_{g'}^{(q-1)} \prod_{h=1}^{H} f(\boldsymbol{x}_i|\boldsymbol{\alpha}_{g'h}^{(q-1)})}.$$

- Sample the column partitions $\boldsymbol{w}_j$ for all $j \in \{1, \ldots, J\}$:

$$\boldsymbol{w}_j^{(q)} \sim \mathcal{M}(1, \boldsymbol{s}_j^{(q)}),$$

with $\boldsymbol{s}_j^{(q)}$ a H-dimensional vector such that:

$$s_{jh}^{(q)} = \frac{\rho_h^{(q-1)} \prod_{g=1}^{G} f(\boldsymbol{x}_i; \boldsymbol{\alpha}_{gh}^{(q-1)})}{\sum_{h'=1}^{H} \rho_{h'}^{(q-1)} \prod_{g=1}^{G} f(\boldsymbol{x}_i | \boldsymbol{\alpha}_{gh'}^{(q-1)})}.$$

In practice, Keribin et al. (2010) finds that a unique iteration of this step is enough. Therefore, it is recommended to choose $n.iter.gibbs = 1$ to fasten the algorithm.

M-STEP The M-step consists in updating the co-cluster parameters $\boldsymbol{\theta}^{(q)}$ to maximise the complete data log-likelihood. Once again, the Lagrange multipliers are used to update the row mixing proportions by:

$$\pi_g^{(q)} = \frac{1}{N} \sum_{i=1}^{N} v_{ig}^{(q)}, \tag{2.63}$$

and the column mixing proportions by:

$$\rho_h^{(q)} = \frac{1}{J} \sum_{j=1}^{J} w_{jh}^{(q)}. \tag{2.64}$$

The parameters $\boldsymbol{\alpha}^{(q)}$ are also updated by finding the root of:

$$\sum_{i=1}^{N} \sum_{g=1}^{G} \sum_{j=1}^{J} \sum_{h=1}^{H} v_{ig}^{(q)} w_{jh}^{(q)} \frac{\partial \log f_{gh}(\boldsymbol{x}_{ij}; \boldsymbol{\alpha}_{gh})}{\partial \boldsymbol{\alpha}}. \tag{2.65}$$

CONVERGENCE For a sufficiently large number of iterations, the samples obtained can be considered as a realisation of the posterior $p(\boldsymbol{v}, \boldsymbol{w} | \boldsymbol{x}; \boldsymbol{\theta}^{(q-1)})$. However, the estimations of the partitions and parameters at each iteration q fluctuate around the values of the built Markov Chain after several iterations. We need to make sure that a sufficiently high number of iterations *nb.burnin* occur before regarding samples as realisations of the target distribution. This is referred to as the "burn-in period". In addition, the final estimation of the parameters $\hat{\boldsymbol{\theta}}$ is not $\boldsymbol{\theta}^{(nb.iter)}$ but it is equal to the mean (or the mode) of the samples over the last $nb.after.burnin$ iterations. The Gibbs sampler is then executed for $nb.final.gibbs$ to sample $\boldsymbol{v}$ and $\boldsymbol{w}$ with $\boldsymbol{\theta}$ fixed to $\hat{\boldsymbol{\theta}}$. The final estimations $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{w}}$ are the modes of this chain. To summarise this discussion, we have:

$$nb.iter = nb.burnin + nb.after.burnin,$$

where we must ensure that *nb.burnin* and $nb.after.burnin$ are high enough to have a correct estimation $\hat{\boldsymbol{\theta}}$. In addition, the Gibbs sampler is run $nb.final.gibbs$ times in order to estimate the partitions.

NUMBER OF ITERATIONS In this thesis, we use plots to determine empirically if the numbers of iterations *nb.burnin* and $nb.after.burnin$ are high enough. This consists in running the algorithm with fixed *nb.burnin* and $nb.after.burnin$ and plotting the values of different parameters over all the iterations. At the end of the algorithm execution, we check that the values of the parameters stabilise after *nb.burnin* iterations, and that $nb.after.burnin$ iterations are high enough to estimate $\boldsymbol{\theta}$ with the mean (or mode) of the samples. We illustrate our point with Figure 2.3 where we see the evolution of a parameter over a total number of iterations (*nb.iter*) equals to 400. We observe that the burn-in number of iterations (*nb.burnin*) should be at a minimum of 100 (so that the right samples are taken into account in the estimation of the parameter) and at a maximum of 200 (so that there are enough samples to correctly estimate $\boldsymbol{\theta}$). However, let us note that this figure is an example for one parameter, but it can be useful to plot the evolution of several parameters to ensure that the algorithm really reaches the stationary distribution for a given *nb.burnin*.

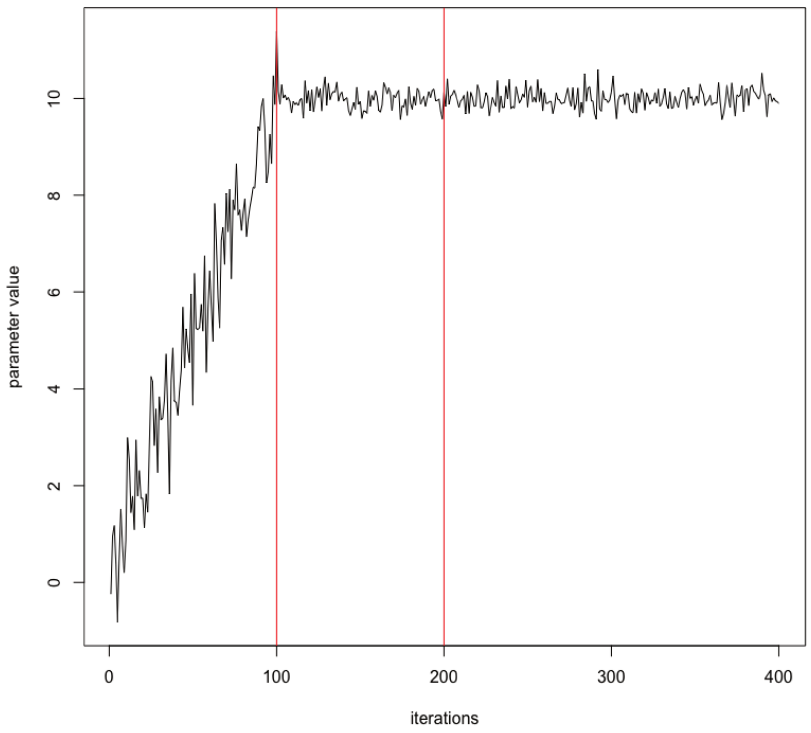

Figure 2.3 – Example of a parameter evolution over iterations: we see that the burn-in period is over at the 100th iteration.

Less subjective ways exist to evaluate if the stationary distribution has been achieved. The authors of Gelman and Rubin (1992) propose a general approach to monitoring the convergence of the Markov Chain Monte Carlo (MCMC) output in which parallel chains are run with starting values that are spread relatively to the posterior distribution. Convergence is confirmed when the output from all chains is indistinguishable. We could have used this method in this thesis; however, in Section 3.4, we show that we can obtain satisfactory

results wiithout it.

CONCLUSION ON THE SEM-GIBBS ALGORITHM The SEM-Gibbs algorithm is easy to implement and is less sensitive to the initialisation than the VEM-algorithm. In addition, it is less likely to find spurious solutions (Keribin et al., 2010). However, it requires the user to choose the right number of iterations to correctly estimate the parameters $\boldsymbol{\theta}$.

2.3.4 Model Selection

In the previous Section, we have presented the methods to estimate the parameters and partitions of the LBM. The number of row-clusters G and column-clusters H were assumed to be known. Similar to Section 2.1.5, we need a criterion to choose the adequate (G, H).

The BIC criterion is widely used for model-based clustering. However, it requires computing the log-likelihood $p(\boldsymbol{x}; \hat{\boldsymbol{\theta}})$, which is not tractable in the context of the LBM (see Section 2.3.3). In Keribin et al. (2013), the authors develop an asymptotic approximation of the ICL for the LBM that is written as follows:

$$\begin{aligned}\text{ICL-BIC}(G, H) = &\log p(\boldsymbol{x}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}; \hat{\boldsymbol{\theta}}) \\ &- \frac{1}{2}(G-1)\log N - \frac{1}{2}(H-1)\log J - \frac{1}{2}\nu \times GH \log(NJ),\end{aligned} \tag{2.66}$$

where ν is the number of components of $\boldsymbol{\alpha}_{gh}$. We obtain a criterion based on the penalization by three terms of the complete data log-likelihood. The first one is due to the row-clusters, the second one is due to the column-cluster and the last one regards the parameters of each block. We will use this approximation in this thesis.

2.3.5 Conclusion

In this section, we have detailed the Latent Block Model, which is a model-based approach for co-clustering. Contrary to the MFA (Section 2.2), the LBM does not assume a latent representation of the data in a lower dimension. However, gathering the variables can be seen as a way of reducing the dimension because, once the co-clustering is performed, the row-clusters are represented by the parameters of each column-cluster but not by the parameters of each column.

The notions reviewed here will be useful for contributions of Chapters 3 and 4. Until now, we have defined the model in a generic way where $\boldsymbol{\alpha}$ represents the parameters of a distribution as in the case of Gaussian data, $\boldsymbol{\alpha}_{gh} = (\boldsymbol{\mu}_{gh}, \boldsymbol{\Sigma}_{gh})$. However, data sets are often made of different types of data (count, ordinal, functional, and so on…). Such data sets are referred to as "mixed data sets". In chapter 3, we extend the LBM so that it is able to take into account the mixed data. In chapter 4, we apply the LBM to count data in the specific context of textual data, which represent a challenge because they are high-dimensional and sparse. In this contribution, we use the Poisson distribution with constrained parameters to address these particularities.

2.4 Appendices

2.4.1 Proof that the EM-algorithm causes the log-likelihood to increase

We introduce an arbitrary distribution $q(\boldsymbol{v})$ defined over the latent variables space. In addition, we define the functional $\mathcal{L}(q, \boldsymbol{\theta})$ as follows:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\boldsymbol{v} \in \mathcal{V}} q(\boldsymbol{v}) \log \left[\frac{p(\boldsymbol{x}, \boldsymbol{v}; \boldsymbol{\theta})}{q(\boldsymbol{v})} \right]. \tag{2.67}$$

We can write:

$$\begin{aligned} \mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\boldsymbol{v} \in \mathcal{V}} q(\boldsymbol{v}) \log \left[\frac{p(\boldsymbol{x}, \boldsymbol{v}; \boldsymbol{\theta})}{q(\boldsymbol{v})} \right] \\ &= \sum_{\boldsymbol{v} \in \mathcal{V}} q(\boldsymbol{v}) \log \left[\frac{p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}) p(\boldsymbol{x}; \boldsymbol{\theta})}{q(\boldsymbol{v})} \right] \\ &= \sum_{v \in \mathcal{V}} q(\boldsymbol{v}) \log \left[\frac{p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta})}{q(\boldsymbol{v})} \right] + \log p(\boldsymbol{x}; \boldsymbol{\theta}) \underbrace{\sum_{v \in \mathcal{V}} q(\boldsymbol{v})}_{=1}. \end{aligned} \tag{2.68}$$

Now, we define the Kullback-Leibler divergence $\mathrm{KL}(q||p)$ between $q(\boldsymbol{v})$ and the posterior distribution $p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta})$ as follows:

$$\mathrm{KL}(q||p) = \sum_{v \in \mathcal{V}} q(\boldsymbol{v}) \log \left[\frac{q(\boldsymbol{v})}{p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta})} \right]. \tag{2.69}$$

Therefore, from Equation (2.68), for any distribution $q(\boldsymbol{v})$ the following decomposition holds:

$$\log p(\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q||p). \tag{2.70}$$

The KL divergence satisfies $\mathrm{KL}(q||p) \geq 0$, with equality, if and only if $q(\boldsymbol{v}) = p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta})$. In other words, $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound of $\log p(\boldsymbol{x}; \boldsymbol{\theta})$.

Let us consider the steps of the EM-algorithm. In the E-step, the lower bound $\mathcal{L}(q, \boldsymbol{\theta}^{(q-1)})$ is maximised with respect to $q(\boldsymbol{v})$ while holding $\boldsymbol{\theta}^{(q-1)}$ fixed. From Equation (2.70), we note that the value of $\log p(\boldsymbol{x}; \boldsymbol{\theta})$ does not depend on $q(\boldsymbol{v})$ and, consequently, the largest value of $\mathcal{L}(q, \boldsymbol{\theta}^{(q-1)})$ will occur when the KL divergence $\mathrm{KL}(q||p)$ is equal to 0, i.e., when $q(\boldsymbol{v})$ is equal to the posterior distribution $p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)})$. If we substitute $q(\boldsymbol{v})$ by $p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)})$ into Equation (2.67), we see that after the E-step, the lower bound takes the form:

$$\begin{aligned} \mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\boldsymbol{v} \in \mathcal{V}} p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}) p(\boldsymbol{x}, \boldsymbol{v}; \boldsymbol{\theta}) - \sum_{\boldsymbol{v} \in \mathcal{V}} p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}) p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}) \\ &= \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) - \sum_{\boldsymbol{v} \in \mathcal{V}} p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}) p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}). \end{aligned} \tag{2.71}$$

It can be noted that the second term of Equation (2.71) is independent of $\boldsymbol{\theta}$; so, the lower bound can be written:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)}) + \text{const}. \tag{2.72}$$

In the M-step, the distribution $q(\boldsymbol{v})$ is held fixed, and we maximise $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(q-1)})$ with respect to $\boldsymbol{\theta}$. From Equation (2.71), we see that maximising the auxiliary function is equivalent to maximising $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. This will cause the lower bound to increase unless it is at a maximum already. After the parameter $\boldsymbol{\theta}$ is updated to $\boldsymbol{\theta}^{(q)}$, the posterior distribution $p(\boldsymbol{v}|\boldsymbol{x}; \boldsymbol{\theta}^{(q)})$ is no longer equal to the distribution $q(\boldsymbol{v})$ since it was defined on the parameter $\boldsymbol{\theta}^{(q-1)}$. Therefore, the KL divergence is no longer equal to zero. The next E-step will change $q(\boldsymbol{v})$ so that the KL is null, and so forth again. Both the E and M steps of the EM algorithm increase the value of a bound on the log-likelihood, and the EM steps change the model parameters in such a way as to cause the log-likehood to increase (or remain unchanged in case it is at a maximum already).

2.4.2 EM-algorithm computations for MFA

We focus on parameters of cluster g:

$$\begin{aligned}
\mathcal{Q}_g &= -\frac{1}{2}\sum_{i=1}^{N} t_{ig}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\Big[(\boldsymbol{x}_i - \boldsymbol{\eta}_g - \boldsymbol{\Lambda}_g\boldsymbol{z}_i)^T\psi_g^{-1}(\boldsymbol{x}_i - \boldsymbol{\eta}_g - \boldsymbol{\Lambda}_g\boldsymbol{z}_i) + \ln|\psi_g|\Big] \\
&= -\frac{1}{2}\sum_{i=1}^{N} t_{ig}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\Big[\boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{z}_i - \boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{\eta}_g - \boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i - \\
&\eta_g^T\psi_g^{-1}\boldsymbol{z}_i + \eta_g^T\psi_g^{-1}\boldsymbol{\eta}_g + \eta_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i - \\
&\boldsymbol{z}_i^T\Lambda_g^T\psi_g^{-1}\boldsymbol{x}_i + \boldsymbol{z}_i^T\Lambda_g^T\psi_g^{-1}\boldsymbol{\eta}_g + \boldsymbol{z}_i^T\Lambda_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i + \ln|\psi_g|\Big].
\end{aligned}$$

To the blue terms, we apply the trace trick:

$$\boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{z}_i = tr(\psi_g^{-1}\boldsymbol{z}_i\boldsymbol{x}_i^T),$$

$$\eta_g^T\psi_g^{-1}\boldsymbol{\eta}_g = tr(\psi_g^{-1}\boldsymbol{\eta}_g\eta_g^T) \text{ and}$$

$$\boldsymbol{z}_i^T\Lambda_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i = tr(\Lambda_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i\boldsymbol{z}_i^T).$$

The terms in orange, red and violet are scalars, so :

$$\boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{\eta}_g + \eta_g^T\psi_g^{-1}\boldsymbol{z}_i = 2\boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{\eta}_g,$$

$$\boldsymbol{z}_i^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i + \boldsymbol{z}_i^T\Lambda_g^T\psi_g^{-1}\boldsymbol{z}_i = 2\boldsymbol{z}_i^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i \text{ and}$$

$$\eta_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i + \boldsymbol{z}_i^T\Lambda_g^T\psi_g^{-1}\boldsymbol{\eta}_g = 2\eta_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i.$$

Therefore, we have:

$$\begin{aligned}
\mathcal{Q}_g = -\frac{1}{2}\sum_{i=1}^{N} t_{ig}\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}\Big[&tr(\psi_g^{-1}\boldsymbol{z}_i\boldsymbol{x}_i^T) - 2\boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{\eta}_g - \\
&2\boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i + tr(\psi_g^{-1}\boldsymbol{\eta}_g\eta_g^T) + \\
&2\eta_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i + tr(\Lambda_g^T\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i\boldsymbol{z}_i^T) + \ln|\psi_g|\Big].
\end{aligned}$$

We apply the expectation to the concerned terms. For convenience, we note $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{v})}$ as $\mathbb{E}$:

$$\mathcal{Q}_g = -\frac{1}{2}\sum_{i=1}^{N} t_{ig}\Big(tr(\psi_g^{-1}\boldsymbol{z}_i\boldsymbol{x}_i^T) - 2\boldsymbol{x}_i^T\psi_g^{-1}\boldsymbol{\eta}_g - 2\boldsymbol{z}_i^T\psi_g^{-1}\boldsymbol{\Lambda}_g\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{v}\big] + tr(\psi_g^{-1}\boldsymbol{\eta}_g\eta T_g) + 2\eta T_g\psi_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{z}_i + tr(\Lambda T_g\psi_g^{-1}\boldsymbol{\Lambda}_g\mathbb{E}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i,\boldsymbol{v}\big]) + \ln|\psi_g|\Big).$$

MAXIMISATION OF $\mathcal{Q}_g$ W.R.T. $\boldsymbol{\eta}_g$

$$\frac{\partial\mathcal{Q}_g}{\boldsymbol{\eta}_g} = \sum_{i=1}^{N} t_{ig}\Big(\psi_g^{-1}\boldsymbol{z}_i - \psi_g^{-1}\boldsymbol{\eta}_g - \psi_g^{-1}\boldsymbol{\Lambda}_g\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{v}\big]\Big),$$

$$\frac{\partial\mathcal{Q}_g}{\partial\boldsymbol{\eta}_g} = 0 \iff \boldsymbol{\eta}_g = \frac{\sum_{i=1}^{N} t_{ig}\Big(\boldsymbol{x}_i - \boldsymbol{\Lambda}_g\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{v}\big]\Big)}{\sum_{i=1}^{N} t_{ig}}.$$

MAXIMISATION OF $\mathcal{Q}_g$ W.R.T. $\boldsymbol{\Lambda}_g$

$$\frac{\partial\mathcal{Q}_g}{\partial\boldsymbol{\Lambda}_g} = \sum_{i=1}^{N} t_{ig}\Big(\psi_g^{-1}(\boldsymbol{x}_i - \boldsymbol{\eta}_g)\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{v}\big]^T - \psi_g^{-1}\boldsymbol{\Lambda}_g\mathbb{E}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i,\boldsymbol{v}\big]^T\Big).$$

Note that $\mathbb{E}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i,\boldsymbol{v}\big]^T = \mathbb{E}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i,\boldsymbol{v}\big]$. So, we have:

$$\frac{\partial\mathcal{Q}_g}{\partial\boldsymbol{\Lambda}_g} = 0 \iff \boldsymbol{\Lambda}_g = \frac{\sum_{i=1}^{N} t_{ig}\Big((\boldsymbol{x}_i - \boldsymbol{\eta}_g)\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{z}_i,\boldsymbol{x}_i,\boldsymbol{v}\big]^T\mathbb{E}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i,\boldsymbol{v}\big]^{-1}\Big)}{\sum_{i=1}^{N} t_{ig}}.$$

MAXIMISATION OF $\mathcal{Q}_g$ W.R.T. ψ_g^{-1} For convenience, we derive with respect to ψ_g^{-1} and not with respect to ψ_g. Favorably, the term ψ_g is going to appear in the expression of $\frac{\partial\mathcal{Q}_g}{\partial\psi_g^{-1}}$.

$$\frac{\partial\mathcal{Q}_g}{\psi_g^{-1}} = -\frac{1}{2}\sum_{i=1}^{N} t_{ig}\Big((\boldsymbol{x}_i - \boldsymbol{\eta}_g)(\boldsymbol{x}_i - \boldsymbol{\eta}_g)^T - 2(\boldsymbol{x}_i - \boldsymbol{\eta}_g)\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{v}\big]^T\Lambda T_g + \boldsymbol{\Lambda}_g\mathbb{E}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i,\boldsymbol{v}\big]^T\Lambda_g^T - \psi_g\Big).$$

Let us focus on the blue term. From the maximisation of $\mathcal{Q}_g$ w.r.t. $\boldsymbol{\Lambda}_g$, we have:

$$\sum_{i=1}^{N} t_{ig}\boldsymbol{\Lambda}_g\mathbb{E}\big[\boldsymbol{z}_i\boldsymbol{z}_i^T|\boldsymbol{x}_i,\boldsymbol{v}\big]^T = \sum_{i=1}^{N} t_{ig}(\boldsymbol{x}_i - \boldsymbol{\eta}_g)\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{v}\big]^T,$$

therefore, by replacing it in $\frac{\partial \mathcal{Q}_g}{\psi_g^{-1}}$:

$$\frac{\partial \mathcal{Q}_g}{\psi_g^{-1}} = -\frac{1}{2}\sum_{i=1}^{N} t_{ig}\Big((\boldsymbol{x}_i - \boldsymbol{\eta}_g)(\boldsymbol{x}_i - \boldsymbol{\eta}_g)^T - (\boldsymbol{x}_i - \boldsymbol{\eta}_g)\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{v}\big]^T \Lambda_g^T - \psi_g\Big).$$

Therefore, we get:

$$\frac{\partial \mathcal{Q}_g}{\partial \psi_g^{-1}} = 0 \iff \psi_g = \frac{\sum_{i=1}^{N} t_{ig}\Big((\boldsymbol{x}_i - \boldsymbol{\eta}_g)(\boldsymbol{x}_i - \boldsymbol{\eta}_g)^T - (\boldsymbol{x}_i - \boldsymbol{\eta}_g)\mathbb{E}\big[\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{v}\big]^T \Lambda_g^T\Big)}{\sum_{i=1}^{N} t_{ig}}.$$

3

Multiple Latent Block Model for mixed data

3.1 Introduction

In this chapter, we present our work published in Selosse et al. (2020b) and Selosse et al. (2019). In Section 2.3, we presented the Latent Block Model approach, which has proved its efficiency in co-clustering several types of data separately: continuous (Nadif and Govaert, 2008), nominal (Singh Bhatia et al., 2017), binary (Laclau et al., 2017), ordinal (Jacques and Biernacki, 2018), and functional (Slimen et al., 2018; Bouveyron et al., 2018). However, this model as it was defined does not allow using different types of data in the data sets. Indeed, we saw that a block contains cells x_{ij} that are assumed to be drawn from a distribution of parameter $\boldsymbol{\alpha}_{gh}$. If the column-cluster corresponding to a block gathers features of different types, the cells of this block will be different, which makes impossible the assumption they were drawn from the same distribution. In this chapter, we present an extension of the Latent Block Model which permits taking heterogeneous data into account in order to tackle this difficulty.

Heterogeneous data sets are composed of features of different types. For example, in medicine, a patient's file can be composed of images (X-rays), text (medical reports), continuous data (age, blood test results…), categorical data (social category, pregnancy, drug addiction…), and even functional data (pulse, blood pressure…). Several clustering frameworks have been developed to address this particularity. The latent class model (Everitt, 1984) is frequently used. It assumes that the variables are conditionally independent upon the row-cluster membership. Consequently, the joint probability distribution function (p.d.f) of the features of different types is obtained by the product of the p.d.f of each individual feature (see an implementation using Mixtcomp software, Biernacki et al. (2015)). However, when the variables are inherently correlated in a row-cluster, this model is not suitable. To overcome this issue, the authors of Marbac et al. (2017) want to conserve standard marginal distributions but also try to loosen the conditional independence of the variables. For this purpose, they use copula, which allow definition of both the dependence model and the type of marginal distributions. The proposed model relies on the main assumption that each cluster follows a Gaussian copula. However, the authors note that model complexity increases with the number of variables, which is not suitable in a big-data context. Another way to address the issues of heterogeneous data is to see some variables as the manifestation of a latent vector. For example, in McParland and Gormley (2016), the clustMD model considers continuous and categorical data (nominal and ordinal) and assumes that a categorical variable is the representation of an underlying latent continuous variable. Then, it is assumed that the continuous variables (observed and unobserved) follow a multivariate Gaussian mixture model. Until now, these methods have proposed models for basic data such as categorical (nominal or ordinal) and continuous data. In Bouveyron et al. (2015), the authors allow the introduction of more complex data such as functional data or networks by projecting the data set into a reproducing kernel Hilbert space. Regarding the analysis of variables, multiblock methods, widely used in Chemistry and Biology, handle data sets that share the same observations but have variables measured differently. They aim at finding underlying relationships between these data sets. In particular, multiblock component models use latent variables to summarise the relevant information between and within the sets (see Smilde et al. (2003) for a complete survey).

However, none of these techniques were developed in a co-clustering framework. To the best of our knowledge, the only work to co-cluster heterogeneous data is Bouchareb et al. (2017), which extends the LBM for data sets with continuous and binary data. The present work goes further by proposing an extension that can take into account four types of data: categorical, continuous, count and ordinal data. Furthermore, the inference algorithm

can deal with missing values and proposes a way to impute them. Finally, the Integrated Completed Likelihood (ICL) criterion (Biernacki et al., 2000) is adapted to the proposed model in order to select the number of row-clusters and column-clusters.

The chapter is organised as follows. Section 3.2 gives an overview of the LBM to help understanding of this paper. Then, it proposes an extension to a new LBM version that allows heterogeneous data sets. Section 3.2.2 proposes an algorithm for model inference, based on a Stochastic Expectation Maximization algorithm coupled with a Gibbs sampler as presented in Section 2.3.3.b. In Section 3.3, a description of the different types of data that can be taken into account with this method is given, and formulas for model inference are presented. Section 3.4 assesses the efficiency of the proposed method on simulated data. Section 3.5 and Section 3.6 show how the method performs on real data sets. Section 3.7 provides a conclusion.

3.2 Multiple Latent Block Model

In this section, we detail the Multiple Latent Block Model.

3.2.1 Definition of the Multiple Latent Block Model

Now, consider a matrix $\boldsymbol{x}$ composed of D different sets of features. It has N rows and $J = \sum_{d=1}^{D} J_d$ columns, J_d being the number of features of the d-th set:

$$\boldsymbol{x} = (\boldsymbol{x}^1, ..., \boldsymbol{x}^D), \text{ with } \boldsymbol{x}^d = (x_{ij}^d)_{i=1,...,N;\ j=1,...,J_d}.$$

Here, the idea of "sets" of features is introduced to define the features we potentially want to group together in a column-cluster, and those we do not want to be together. Thus, features of a same set can be grouped together in an *intra-set* column-cluster; features of different sets cannot. There are two reasons for separating features into different sets: a technical one and a semantic one. Firstly, two features of different types (*e.g.* a categorical feature and a continuous one) are chosen so as not to be modeled with a similar probability distribution, but rather with a standard distribution suitable to their type. Since it will be assumed later that all the features in a column-cluster have the same p.d.f., such an assumption is not suitable for features of different types. This is the reason for this work. Secondly, the user can consider, for practical reasons, that some features necessarily have to be separated because it does not make sense to gather them in a same column cluster. This case is explored in Section 3.6 and the reader can refer Selosse et al. (2019) for more details. The sets of elements $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^D)$ are annotated $(\boldsymbol{x}^d)_d$ with $d \in \{1, \ldots, D\}$.

In the co-clustering framework, it is assumed that G row-clusters and $H = H_1 + \ldots + H_D$ column-clusters exist, and that they are inherent to the matrix $\boldsymbol{x}$. Moreover, the sums and the products relating to sets of features will be written in subscript by the letter d. Again, the underlying range of variation will be omitted in the sums and products, thus they are written $\sum_d$ and $\prod_d$.

Finally, a data set may have missing data. This aspect is specific to this chapter since it was not taken into account in Section 2.3. To handle this particularity, the d^{th} matrix $\boldsymbol{x}^d$ is said to be made up of two sets $\check{\boldsymbol{x}}^d$ and $\hat{\boldsymbol{x}}^d$, where $\check{\boldsymbol{x}}^d$ is the observed data, and $\hat{\boldsymbol{x}}^d$ is the missing data. An element of $\boldsymbol{x}^d$ will be annotated $\check{x}_{ij}^d$ if x_{ij}^d is observed, and $\hat{x}_{ij}^d$ otherwise. To model missing values, three main processes exist in data analysis (we refer to Little and Rubin (1986) for a complete review). The Missing Completely At Random (MCAR) process assumes that the missing data mechanism is unrelated to the values of any variables: for

example, in a survey, participants accidentally skipped questions. The Missing At Random (MAR) process supposes that a missing value has nothing to do with the variable whose value is missing, but it does have to do with the values of other variables. For example, males are less likely to fill in a depression survey but this has nothing to do with their level of depression, after accounting for maleness. The last process is called Missing Not At Random (MNAR) and occurs when the missing value is directly influenced by the variable itself. For instance, a drug addict may not answer a question about drugs precisely because of their addiction. In the present work, it is assumed that the whole missing process is MAR, because it is the most frequent situation encountered in practice Donders et al. (2006).

The LBM relies on the assumption that the block's elements are the realizations of a random variable that follows a distribution with parameter $\boldsymbol{\alpha}$. In this work, we chose to adopt a standard distribution for each kind of feature (e.g. Gaussian for continuous data and Poisson for count data). In this context, if the elements of the blocks are not of the same type, it is not possible to consider that they were sampled from the same distribution. The Multiple Latent Block Model (MLBM) was defined in Robert (2017) for two matrices of binary data separated into two blocks for semantic reasons. In this chapter, the MLBM is extended for $D \geq 1$ matrices such that each matrix may have continuous, categorical, ordinal or count data. In this model, the columns of the matrix $\boldsymbol{x}$ are reordered such that $\boldsymbol{x}$ is composed of D matrices put side by side, each matrix containing features of homogeneous type as described above. The co-clustering is performed in such a way that features of different types cannot be part of a same column-cluster. Consequently, it is possible to define a distribution on each block because it is made of variables of the same type. Figure 3.1 illustrates the idea behind this model.

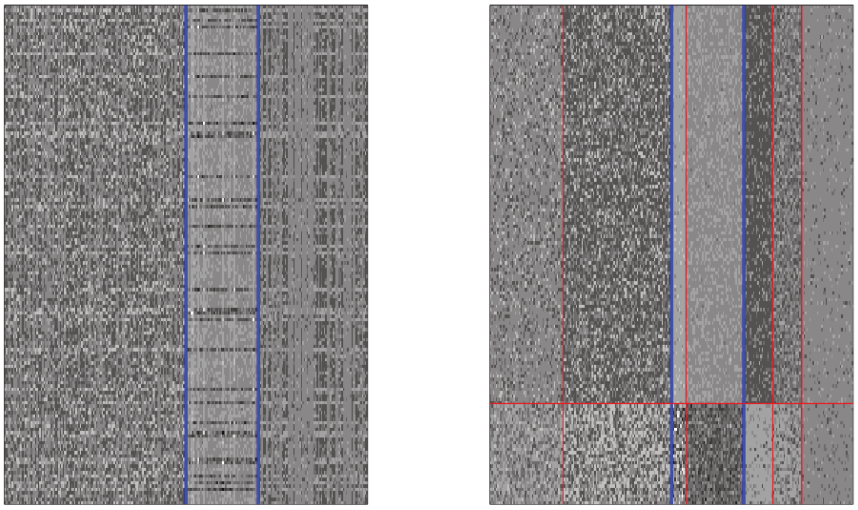

Figure 3.1 – (a) is the matrix $\boldsymbol{x}$. The blue lines represent the separation of the features that are not of same type. (b) is the matrix after having performed a co-clustering. The red lines represent the co-clusters limits.

Similar to Section 2.3, $\boldsymbol{v}$ represents the row partition. Let $\boldsymbol{w}^d$ denote the column partitions

of the d-th matrix ($1 \leq d \leq D$), $\boldsymbol{\rho}^d = (\rho_1^d, \ldots, \rho_{H_d}^d)$ the corresponding mixing proportions, and let us introduce the notations $\boldsymbol{w} = (\boldsymbol{w}^d)_d$ and $\boldsymbol{\rho} = (\boldsymbol{\rho}^d)_d$.

The MLBM relies on several assumptions. The first one states that the D matrices data are independent conditional on the row and column partitions, and specifically that, for all $t \neq d$ the matrix $\boldsymbol{x}^d$ does not depend on the column partitions $\boldsymbol{w}^t$:

$$p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{w}) = p(\boldsymbol{x}^1|\boldsymbol{v}, \boldsymbol{w}^1) \times \ldots \times p(\boldsymbol{x}^D|\boldsymbol{v}, \boldsymbol{w}^D).$$

The other assumptions of the MLBM are similar to those of the LBM. Firstly, the univariate random variables x_{ij}^d are assumed to be conditionally independent on partitions $\boldsymbol{v}$ and $\boldsymbol{w}^d$. Thus, the conditional probability function of $\boldsymbol{x}$ given $\boldsymbol{v}$ and $(\boldsymbol{w}^d)_d$ is expressed as:

$$p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\alpha}) = \prod_{i,j,g,h,d} p(x_{ij}^d; \alpha_{gh}^d)^{v_{ig} w_{jh}^d},$$

where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^d)_d$ with $\boldsymbol{\alpha}^d = (\alpha_{gh}^d)_{g,h}$ is the distribution parameters of block (g, h) of matrix $\boldsymbol{x}^d$.

Second, the latent variables $\boldsymbol{v}, \boldsymbol{w}^1, \ldots, \boldsymbol{w}^D$ are assumed to be independent, so: $p(\boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\pi}, \boldsymbol{\rho}) = p(\boldsymbol{v}; \boldsymbol{\pi}) \prod_d p(\boldsymbol{w}^d; \boldsymbol{\rho}^d)$, where:

$$p(\boldsymbol{v}; \boldsymbol{\pi}) = \prod_{i,g} \pi_g^{v_{ig}} \text{ and } p(\boldsymbol{w}^d; \boldsymbol{\rho}^d) = \prod_{j,h} {\rho_h^d}^{w_{jh}^d}.$$

The MLBM parameter is thus defined by $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\alpha})$. Moreover, if V and $(W^d)_d$ are the sets of all possible labels $\boldsymbol{v}$ and $(\boldsymbol{w}^d)_d$, the probability density function $p(\boldsymbol{x}; \boldsymbol{\theta})$ is written:

$$p(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{(\boldsymbol{v}, (\boldsymbol{w}^d)_d) \in V \times (W^d)_d} \prod_{i,g} \pi_g^{v_{ig}} \prod_d \prod_{j,h} {\rho_h^d}^{w_{jh}^d} \prod_{i,j,g,h} p(x_{ij}^d; \alpha_{gh}^d)^{v_{ig} w_{jh}^d}. \tag{3.1}$$

Note that so far the type of $p(x_{ij}^d; \alpha_{gh}^d)$ has not been defined. It will be in Section 3.3, based on the type of x_{ij}^d (nominal, ordinal, continuous, ...).

Finally, the MAR assumption in the Latent Block Model indicates that we have:

$$p(\hat{x}_{ij}^d|\boldsymbol{v}, \boldsymbol{w}^1, \ldots, \boldsymbol{w}^D) = \prod_{g,h} p(x_{ij}^d; \alpha_{gh}^d)^{v_{ig} w_{jh}^d}.$$

3.2.2 Model Inference

The MLBM inference aims at estimating $\boldsymbol{\theta}$ that maximises the observed log-likelihood:

$$l(\boldsymbol{\theta}; \check{\boldsymbol{x}}) = \sum_{\hat{\boldsymbol{x}}} \log p(\boldsymbol{x}; \boldsymbol{\theta}). \tag{3.2}$$

We saw in Section 2.3 that the classical EM algorithm cannot be used in the LBM framework. It is still the case in the MLBM context. We saw that different alternatives to the EM algorithm exist, such as the variational EM algorithm, the SEM-Gibbs algorithm, and other algorithms linked to Bayesian inference Govaert and Nadif (2013). The SEM-Gibbs version is used in this chapter because in addition to being known to be less sensitive to initialization, it is simple to implement. Furthermore, it easily handles missing values $\hat{\boldsymbol{x}}$ in $\boldsymbol{x}$, which is an important advantage for real data sets.

3.2.2.a SEM-Gibbs algorithm

The SEM-Gibbs algorithm begins with an initialization of partitions, parameters and missing values $\boldsymbol{v}^{(0)}, \boldsymbol{w}^{(0)}, \boldsymbol{\theta}^{(0)}, \hat{\boldsymbol{x}}^{(0)}$. This initialization process is described in more details later. The following five steps describe the q-th iteration, with $q \in (1, ..., nbSEM)$. The choice of the number of iterations ($nbSEM$) will also be described later.

(A) SAMPLING ROW PARTITIONS Generate the row partitions with:

$$p(v_{ig}^{(q)} = 1 \mid \boldsymbol{x}, \boldsymbol{w}^{(q-1)}; \boldsymbol{\theta}^{(q-1)}) \propto \pi_g^{(q-1)} \times \prod_d t_g^d(\boldsymbol{x}_{i.}^d | \boldsymbol{w}^{d^{(q-1)}}; \boldsymbol{\alpha}^{d^{(q-1)}}), \tag{3.3}$$

where $t_g^d(\boldsymbol{x}_{i.}^d | \boldsymbol{w}^{d(q-1)}; \boldsymbol{\alpha}^{d^{(q-1)}}) = \prod_{j,h} f(x_{ij}^d; {\alpha_{gh}}^{d^{(q-1)}})^{w_{jh}^{d\ (q-1)}}$ with $\boldsymbol{x}_{i.}^d = (x_{ij}^d)_j$.
Note that this probability depends on the data type of the d-th matrix through the p.d.f $f(x_{ij}^d; {\alpha_{gh}}^{d^{(q-1)}})$, whose exact expression will be given in Section 3.3.

(B) FIRST M-STEP This first M-step consists in updating the co-cluster parameters $\boldsymbol{\theta}^{(q)}$ to maximise the completed log-likelihood (3.2). The row mixing proportions are consequently updated by:

$$\pi_g^{(q)} = \tfrac{1}{N} \sum_i v_{ig}^{(q)},$$

and the parameter $\boldsymbol{\alpha}^{d^{(q)}}$ is updated as well. However, the computations depend on the type of matrix $\boldsymbol{x}$ features. Section 3.3 describes how to update $\boldsymbol{\alpha}^{d^{(q)}}$ according to the type of variables.

(C) SAMPLING COLUMN PARTITIONS For all $d \in \{1, \ldots, D\}$ generate the column partitions for the d-th matrix $\boldsymbol{x}^d$ with:

$$p(w_{jh}^{d\ (q)} = 1 \mid \boldsymbol{x}^d, \boldsymbol{v}^{(q)}; \boldsymbol{\theta}^{(q)}) \propto \rho_h^{d(q)} \times s_h^d(\boldsymbol{x}_{.j}^d | \boldsymbol{v}^{(q)}; \boldsymbol{\alpha}^{d^{(q-1)}}), \tag{3.4}$$

where $s_h^d(\boldsymbol{x}_{.j}^d | \boldsymbol{v}^{(q)}; \boldsymbol{\alpha}^{d^{(q)}}) = \prod_{i,g} f(x_{ij}^d; {\alpha_{gh}}^{d^{(q-1)}})^{v_{ig}^{(q)}}$ with $\boldsymbol{x}_{.j}^d = (x_{ij}^d)_i$.
Here, note that s_h^d obviously depends on the type of the d-th matrix (see Section 3.3).

(D) SECOND M-STEP In this second M-step, the column mixing proportions are updated by:

$$\rho_h^{d(q)} = \tfrac{1}{J_d} \sum_j w_{jh}^{d\ (q)},$$

and the parameter $\boldsymbol{\alpha}^{d^{(q)}}$ is also updated depending on the data type of the d-th matrix (see Section 3.3).

The SEM-Gibbs algorithm is iterated for a given number of iterations. The first part of these iterations is called the burn-in period, meaning that the parameters of $\boldsymbol{\theta}$ are not yet simulated according to its stationary distribution. Consequently, only iterations that occurred after this burn-in period are taken into account and are referred to as the sampling distribution hereafter. While the final estimations of discrete parameters give the mode of the sampling distribution, the final estimations of the continuous parameters give the mean of the sample distribution. This leads to a final estimation of $\boldsymbol{\theta}$ called $\hat{\boldsymbol{\theta}}$. Then, a sample

of $(\hat{\boldsymbol{x}}, \boldsymbol{v}, \boldsymbol{w})$ is simulated by iterating steps (A) and (B) of the SEM-Gibbs algorithm with $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. The final partitions $(\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}})$ and the missing observations $\hat{\boldsymbol{x}}$ are estimated using the mode of their marginal sampled distribution.

PROPOSAL FOR INITIALIZATION OF THE MLBM The algorithm starts with an initialization of the partitions. Then the mixing proportions and the block parameters are estimated with regard to these partitions. In the case of $D = 1$, this initialization can be made randomly or with the k-means algorithm (Brault, 2014). However, when $D > 1$, these options often lead to empty clusters. In this thesis, we propose a new specific initialization strategy was worked out to tackle this issue. It begins with an initial random initialization. However, for the first *nb.init* iterations (such that *nb.init* is less than or equal *nb.burnin*), whenever a row-cluster becomes empty, a percentage of the row partitions is sampled from the Multinomial distribution $\mathcal{M}(1/G, \ldots, 1/G)$. Concretely, it means that at iteration q, with $q \leq nb.init$, if a row-cluster does not have any element, a percentage of the rows of matrix $\boldsymbol{v}^{(q)}$ are erased, and randomly re-sampled. Similarly when a column-cluster becomes empty on the d^{th} matrix, a percentage of the column partitions is sampled from the multinomial distribution $\mathcal{M}(1/H_d, \ldots, 1/H_d)$. Therefore, if a column-cluster of the d^{th} matrix does not have any element at iteration q $(q \leq I)$, a percentage of the rows of matrix $\boldsymbol{w}^{d(q)}$ are erased, and randomly re-sampled.

CHOICE OF THE NUMBER OF ITERATIONS As explained in Section 2.3.3.b, the SEM-algorithm can be slow to reach its stationary state. After having arbitrarily chosen the total number of iterations, the stability of the algorithm has to be checked. To accomplish that, the evolution of the parameters through the iterations can simply be graphically analysed. If the parameters are "stable" between the burn-in period and the last iteration then the number of iterations was well chosen.

3.2.2.b Model Selection

To select the number of blocks $(G, H_1, \ldots, H_D)$, a model selection criterion must be used. The most standard ones, like Bayesian Information Criterion (BIC) Schwarz (1978), rely on penalizing the maximum log-likelihood value $l(\hat{\boldsymbol{\theta}}; \check{\boldsymbol{x}})$. However, due to the dependency structure of the observed data $\check{\boldsymbol{x}}$, this value is not available.

Alternatively, an approximation of the ICL information criterion Biernacki et al. (2000), called here ICL-BIC, can be invoked to overcome the previous problem due to the dependency structure in the missing variables $(\check{\boldsymbol{x}}, \boldsymbol{v}, \boldsymbol{w})$. The key point is that this latter vanishes since ICL relies on the completed latent block information $(\boldsymbol{v}, \boldsymbol{w})$, instead of integrating on it as it is the case in BIC. In particular, Keribin et al. (2013) detailed how to express ICL-BIC for the general case of categorical data. It is possible to straightforwardly transpose the ICL-BIC expression given by these authors by following their work step by step, with no new technical material. As proved in Robert (2017), the resulting MLBM-specific ICL-BIC is expressed by:

$$\text{ICL-BIC}(G, H_1, \ldots, H_D) =$$
$$\log p(\check{\boldsymbol{x}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}; \hat{\boldsymbol{\theta}}) - \tfrac{1}{2}(G-1)\log N - \sum_d \tfrac{1}{2}(H_d - 1)\log J_d - \sum_d \tfrac{1}{2}\nu_d G H_d \log(N \times J_d),$$

where ν_d is the number of parameters to estimate for the d-th matrix $\boldsymbol{x}^d$. It will depend on G, H_d and the type of the variables of $\boldsymbol{x}^d$. Table 3.1 in Section 3.3 gives ν_d for each type of distribution.

3.2.2.c Proposal of a heuristic strategy to avoid a greedy search on the ICL-BIC values

In theory, to find the best number of blocks $(G, H_1, \ldots, H_D)$, the co-clustering has to be executed for each possible value and the result with the highest ICL-BIC has to be retained. Let n_G be the number of candidate values for G, while n_{H_d} is the number of candidate values for H_d, $d \in \{1, \ldots, D\}$. Thus, the number of co-clustering to execute is $n_G \times n_{H_1} \times \ldots \times n_{H_D}$. For example, if $D = 3$ and the user wants to try 10 values for G and for each H_d, then it would require execution of 10^4 co-clusterings. Depending on the data set, it might take too much time to find the best solution. In practice, a good set $(G, H_1, \ldots, H_D)$ is found using the following heuristic. Let (G_{min}) be the minimum of the candidate values for G. Then, $(H_{d_{min}})_d$ is the minimum of the candidate values for $(H_d)_d$. The algorithm starts with the set $(G_{min}, H_{1_{min}}, \ldots, H_{D_{min}})$. At iteration p, the current best set $(G, H_1, \ldots, H_D)$ is called $(G, H_1, \ldots, H_D)^{(p)}$ and is made of values: $(G^{(p)}, H_1^{(p)}, \ldots, H_D^{(p)})$. At the p^{th} iteration, $(D + 1)$ co-clusterings are performed with sets $(G^{(p)} + 1, H_1^{(p)}, \ldots, H_D^{(p)})$, $(G^{(p)}, H^{(p)} + 1, \ldots, H_D^{(p)})$,...,$(G^{(p)}, H_1^{(p)}, \ldots, H_D^{(p)} + 1)$. Then, the ICL-BIC is computed for each result. If none of the ICL-BIC values are better than for the set $(G, H_1, \ldots, H_D)^{(p)}$, the algorithm finishes and $(G, H_1, \ldots, H_D)^{(p)}$ is the set to use. Otherwise, the set with the highest ICL-BIC is retained, and becomes $(G, H_1, \ldots, H_D)^{(p+1)}$. The algorithm then reiterates the same steps.

3.3 Modeling of the different types of data

Representing the data as a mathematical object is challenging and requires compromise. Often the user has to find a trade-off between information loss, interpretability and feasibility for their representation. The model described here can work with the following types of data: categorical data (nominal, ordinal, binary), count data, continuous data and document-term matrices. While the probability distributions for nominal (Multinomial), binary (Bernoulli), count (Poisson) and continuous (Gaussian) data are widely accepted, several ways to model textual and ordinal data exist.

The simplest way to represent textual data is as a Document-Term count matrix where a cell counts how many times a term appears in a document. The Poisson distribution is a good distribution for modeling this matrix because it models the occurrences of an event (in this case, the appearances of a word). In a more advanced way, the Document-Term TF-IDF matrix, counts the times a term appears, but penalises the result if this same term appears in the other documents Jones (1972). The resulting score is continuous numeric which implies the usage of the Gaussian distribution. In the latter, the "stop-words" terms are discarded. However, even with the TF-IDF normalisation, the Gaussian distribution is not the best way to handle Document-Term matrices Salah et al. (2018). Lots of other Document-Term matrix types exist, and they have proven their efficiency in many applications Ailem et al. (2017); Laclau et al. (2017). In this work, a simple Document-Term matrix representation is considered. When handling Document-Term matrix data only (and no other kind of data), diagonal LBM or equivalent approaches are more appropriate since the matrix is sparse Ailem et al. (2017, 2016).

Ordinal data is also a sensitive data type. It may seem very easy to model them as if they were nominal, but doing that would spoil the order between the different levels, which is an intrinsic property of this type of data. In some applications, it can be interpreted as continuous Lubke and Muthén (2004) but in other cases it is not an option. For example, for clinical surveys, psychologists sometimes spend years defining ordinal scales on abstract

concepts like pain, perception of control or anxiety MaloneBeach and Zarit (1995); Zigmond and Snaith (1983); it is therefore difficult to project their results onto other scales or into a continuous space. In the present work, a recent distribution for ordinal data (BOS for Binary Ordinal Search model, Biernacki and Jacques (2016)) is used. It has proven its efficiency for modeling and clustering ordinal data. The main advantages of the BOS model are its parsimony and the interpretability of its parameters.

This section describes the expression of the p.d.f $f(x_{ij}^d; \boldsymbol{\alpha}^{d^{(q-1)}})$ and the way to update $\boldsymbol{\alpha}^{d^{(q-1)}}$, in the SEM-Gibbs algorithm, depending on the type of the matrix $\boldsymbol{x}^d$. The superscripts (q) and (d) are omitted to simplify the expressions.

3.3.1 Modeling nominal data

A nominal variable is a variable that can take on one of a limited, fixed, number of possible values. Each of the possible values of a categorical variable is referred to as a level. For a block (g,h) of nominal data, we consider the multinomial distribution $\mathcal{M}(1, \boldsymbol{\beta}_{gh})$, where $\boldsymbol{\beta}_{gh} = (\beta_{gh}^r)_{r=1,\dots,m}$, and $\sum_r^m \beta_{gh}^r = 1$. Therefore, with this type of data, the MLBM block parameter α_{gh} is quoted as $\boldsymbol{\beta}_{gh}$, and the p.d.f is given by:

$$f(x_{ij}; \boldsymbol{\beta}_{gh}) = \prod_r^m (\beta_{gh}^r)^{\mathbb{1}(x_{ij}=r)},$$

where $\mathbb{1}(x_{ij} = r) = 1$ if $x_{ij} = r$, and 0 otherwise.

The update of each β_{gh}^r is:

$$\beta_{gh}^r = \frac{1}{n_{gh}} \sum_{i,j} v_{ig} w_{jh} \mathbb{1}(x_{ij} = 1),$$

where n_{gh} is the number of elements belonging to block (g, h).

Firstly, note that if two nominal variables do not have the same number of levels m, then their distribution are not defined on the same support. Consequently, such variables should be separated into different matrices $\boldsymbol{x}^d$ of $\boldsymbol{x}$. Secondly, the co-clustering we propose is dependent on the order of the levels. For example two categorical features with $m = 3$ levels having respective parameters $\boldsymbol{\beta} = (0.1, 0.7, 0.2)$ and $\boldsymbol{\beta} = (0.7, 0.2, 0.1)$ won't be detected as two variables following the same distribution. Consequently they won't be grouped together in a similar column cluster, whereas a simple switch in the order of the levels could change this and lead to grouping these variables together. Note that this problem is not specific to co-clustering and is also present in clustering Biernacki and Lourme (2019). While the user should be aware that the results are conditional on the encoding of levels, this is not an issue addressed in this work.

3.3.2 Modeling ordinal data

Ordinal data is a special case of nominal data, where the order between the levels has a meaning.

In the present work, the BOS model (Biernacki and Jacques, 2016) is chosen to model ordinal data. It is a probability distribution parameterized by a position parameter $\mu_{gh} \in \{1, \dots, m\}$ and a precision parameter $\tau_{gh} \in [0, 1]$. This distribution has interesting properties from an interpretation standpoint: it rises from the uniform distribution when $\tau_{gh} = 0$ to a more peaked distribution around the mode μ_{gh} when τ_{gh} increases, and it reaches a Dirac distribution at the mode μ_{gh} when $\tau_{gh} = 1$. It is shown in Biernacki and Jacques (2016) that

the BOS distribution is a polynomial function of τ_{gh} with degree $m-1$ whose coefficients depend on the position parameter μ_{gh}.

Therefore, with this type of data, the MLBM block parameter α_{gh} is quoted as (μ_{gh}, τ_{gh}), and the p.d.f is given by:

$$f(x_{ij}; \mu_{gh}, \tau_{gh}) = \sum_{r=0}^{m-1} C_r(\mu_{gh}, x_{ij}) \tau_{gh}^r,$$

where $C_r(\mu_{gh}, x_{ij})$ is a constant depending on μ_{gh} and x_{ij}.

Since BOS inference relies on an EM-algorithm, the update of parameter (μ_{gh}, τ_{gh}) is obtained through an EM-algorithm. For further details on this algorithm, see Biernacki and Jacques (2016). Similarly to the nominal variables case, if two ordinal variables do not have the same number of levels, they have to be separated into different matrices $\boldsymbol{x}^d$ of $\boldsymbol{x}$.

3.3.3 Modeling continuous data

In the continuous case, the unidimensional Gaussian distribution $\mathcal{N}(\mu_{gh}, \sigma_{gh}^2)$ is considered. Thus, the MLBM block parameter α_{gh} is here (μ_{gh}, σ_{gh}) and the p.d.f is given by:

$$f(x_{ij}; \mu_{gh}, \sigma_{gh}) = \exp\{\tfrac{-1}{2\sigma_{gh}^2}(x_{ij} - \mu_{gh})^2\} / \sqrt{2\pi\sigma_{gh}^2}.$$

The update of parameters $(\mu_{gh}, \sigma_{gh}^2)$ is:

$$\mu_{gh} = \frac{1}{n_{gh}} \sum_{i,j} v_{ig} w_{jh} x_{ij} \quad \text{and} \quad \sigma_{gh}^2 = \frac{1}{n_{gh}} \sum_{i,j} v_{ig} w_{jh} (x_{ij} - \mu_{gh})^2.$$

3.3.4 Modeling count data

Count variables are modeled by the Poisson distribution. For a block (g,h) of count data, a Poisson distribution with a specific parametrization is considered: $\mathcal{P}(n_{i.} n_{.j} \delta_{gh})$, where $n_{i.} = \sum_j x_{ij}$ and $n_{.j} = \sum_i x_{ij}$ are the number of occurrences in row i and the number of occurrences in column j. The parameters $n_{i.}$ and $n_{.j}$ are independent of the co-clustering and are consequently preliminary estimated from the count data matrix. Consequently, the MLBM parameter α_{gh} are only the parameter δ_{gh}, which is the effect of the block (g,h) Govaert and Nadif (2018). The p.d.f is given by:

$$f(x_{ij}; \delta_{gh}) = \tfrac{1}{x_{ij}!} e^{-n_{i.} n_{.j} \delta_{gh}} (n_{i.} n_{.j} \delta_{gh})^{x_{ij}}.$$

The update of each parameter δ_{gh} is obtained by:

$$\delta_{gh} = \frac{1}{n_{g.} n_{.h}} \sum_{i,j} v_{ig} w_{jh} x_{ij},$$

where $n_{g.} = \sum_{i,j} v_{ig} x_{ij}$ and $n_{.h} = \sum_{i,j} w_{jh} x_{ij}$.

Finally, Table 3.1 summarises the number of parameters ν for each type of data described above.

Table 3.1 – Number of parameters (ν) of the distribution properties

Data type	Distribution	α_{gh}	ν
Nominal	Multinomial	$\boldsymbol{\beta}_{gh} = (\beta^r_{gh})_{r=1,\dots,m}$	$(m-1)$
Ordinal	BOS	(μ_{gh}, τ_{gh})	2
Continuous	Gaussian	(μ_{gh}, σ_{gh})	2
Count	Poisson	$(\mu_i, \nu_j, \delta_{gh})$	1

3.4 Numerical experiments on artificial data

This section has two goals. The first is to show that the proposed inference algorithm works appropriately. The second is to evaluate the model selection strategy: the efficiency of the ICL-BIC criterion in selecting the true numbers of clusters and the ability of the heuristic search to sparsely explore the space of numbers of clusters.

3.4.1 Simulation settings

Two simulation settings are considered. While they both have the same parameters, the first is built such that $(N = J_1 = J_2 = J_3 = J_4 = 100)$, and the second is built with $(N = J_1 = J_2 = J_3 = J_4 = 500)$.

PARAMETERS SETUP Both settings were simulated with four types of distribution: nominal (with $m = 5$ levels), continuous, ordinal (with $m = 3$ levels), and count data. The number of blocks was set to $(G, H_1, H_2, H_3, H_4) = (3, 3, 3, 3, 3)$. Furthermore, the mixing row proportions were $\boldsymbol{\pi} = (0.2, 0.3, 0.5)$ and the mixing column proportions were equal to: $\boldsymbol{\rho}_1 = (0.25, 0.3, 0.45)$, $\boldsymbol{\rho}_2 = (0.2, 0.35, 0.45)$, $\boldsymbol{\rho}_3 = (0.25, 0.35, 0.4)$, $\boldsymbol{\rho}_4 = (0.25, 0.35, 0.4)$. Table 3.2 details the parameters that were assigned to each block.

Table 3.2 – Value of block parameters. For the count data, parameters are not equal between the first and second simulations because they depend on the margins.

	Nominal $m = 5$ $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$		
	col-cluster 1	col-cluster 2	col-cluster 3
row-cluster 1	0.05,0.05,0.8,0.05,0.05	0.1,0.25,0.3,0.3,0.05	0.1,0.2,0.4,0.2,0.1
row-cluster 2	0.05,0.1,0.7,0.1,0.05	0.8,0.05,0.05,0.05,0.05	0.4,0.05,0.1,0.05,0.4
row-cluster 3	0.2,0.5,0.2,0.05,0.05	0.8,0.05,0.05,0.05,0.05	0.05,0.8,0.05,0.05,0.05

	Continuous μ, σ			Ordinal $m = 5$ μ, π		
	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 1	col-cluster 2	col-cluster 3
row-cluster 1	100,1	0.5,5	-90,5	3,0.4	1,0.2	3,0.7
row-cluster 2	10,4	-15,1	-95,1	2,0.1	3,0.5	2,0.8
row-cluster 3	-20,1	-30,3	500,4	2,0.5	1,0.8	2,0.2

	Count 100 $\delta \times 10^{-5}$			Count 500 $\delta \times 10^{-7}$		
	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 1	col-cluster 2	col-cluster 3
row-cluster 1	1.2	5.5	1.2	4.6	20.5	4.9
row-cluster 2	8.3	5.5	0.5	30.0	20.5	1.6
row-cluster 3	1.3	1.3	3.5	5.5	5.6	14.5

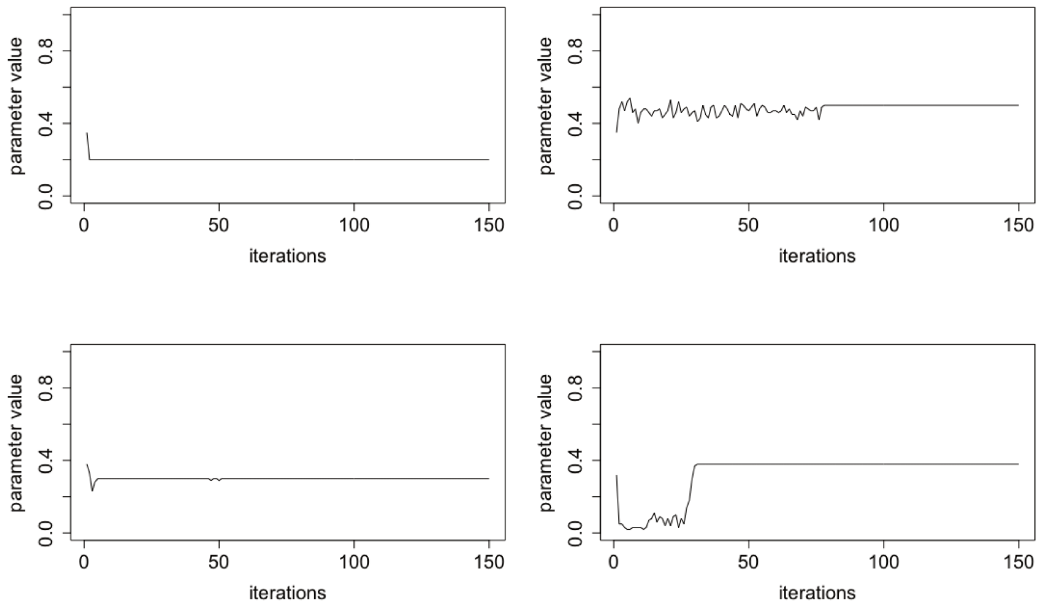

Figure 3.2 – Evolution of parameters ρ through the SEM-Gibbs algorithm iterations. From left to right, and from top to bottom, the graph represents the evolution of the first element of each vector ρ_1, ρ_2, ρ_3 and ρ_4.

EXPERIMENTAL SETUP For both settings, 20 data sets are simulated, and the same process is run through. Firstly, the co-clustering is performed on the 20 data sets with the true numbers of clusters $(G_1, H_1, \ldots, H_D)$ and the correctness of the parameter estimation is evaluated. Then, we assess the efficiency of the ICL-BIC criteria by using an exhaustive search among possible values for the number of clusters. In order to reduce the number of ICL-BIC values to compute, we consider only the number of clusters obtained by adding or removing one to each of the elements of the true $(G, H_1, \ldots, H_D)$. Therefore, for each simulation, $3^5 = 243$ co-clusterings are executed, because 3 values are tested for G,H_1,H_2,H_3 and H_4. Then, the set $(G, H_1, \ldots, H_D)$ with the best ICL-BIC value is retained. Afterward, the heuristic search from Section 3.2.2.b is evaluated. In this case, the number of co-clusterings to be performed is not fixed because the algorithm stops once it can't find a better ICL-BIC value.

CHOICE FOR THE NUMBER OF ITERATIONS The number of iterations for the SEM-Gibbs algorithm was set to 150 and the burn-in period was considered to take 100 iterations. To check if this number of iterations is enough, the evolution of the parameters is graphically observed, as in Figure 3.2. Here, only a few parameters are represented as an example, but it is useful to check for several parameters. We notice in this example that some of the parameters reached their stationary state from the beginning of the algorithm, and that other parameters needed 50 to 100 iterations to get stable. Therefore, in order to ensure that all parameters have achieved their stationary distribution, a burn-in period of 100 iterations is considered (over a total number of 150 iterations). Numerical results in Section 3.4.2 show that these numbers of iterations are large enough since the parameters are well estimated with this particular setting.

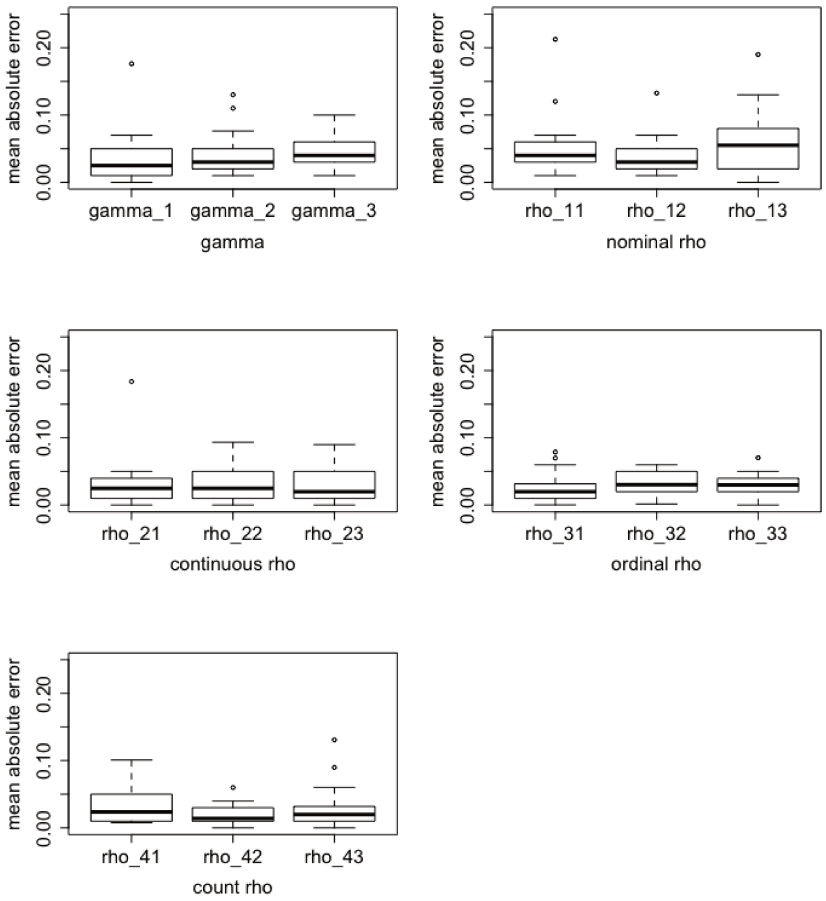

Figure 3.3 – Mean absolute error for the mixing proportions with $N = J_d = 100$.

CHOICES FOR INITIALIZATION The number *nb.init* corresponds to the number of iterations a certain percentage of the partitions are randomly sampled when a cluster becomes empty, as explained in Section 3.2.2.a. Here, *nb.init* is tuned to be equal to the number of iterations for burn-in, while the percentage value was fixed to 20.

3.4.2 Parameter and partition estimation

3.4.2.a Parameter estimation

The co-clustering was performed on 20 data sets, with the true numbers of clusters. The mean absolute errors for the mixing proportions are shown in Figure 3.3 and Figure 3.4. The mean absolute errors between the parameter values and their estimation are given in Table 3.3 and Table 3.4 for the continuous, ordinal and count data. For the nominal data, all the mean absolute errors were less than 0.01. These errors are extremely low, which means that the model parameters are correctly estimated.

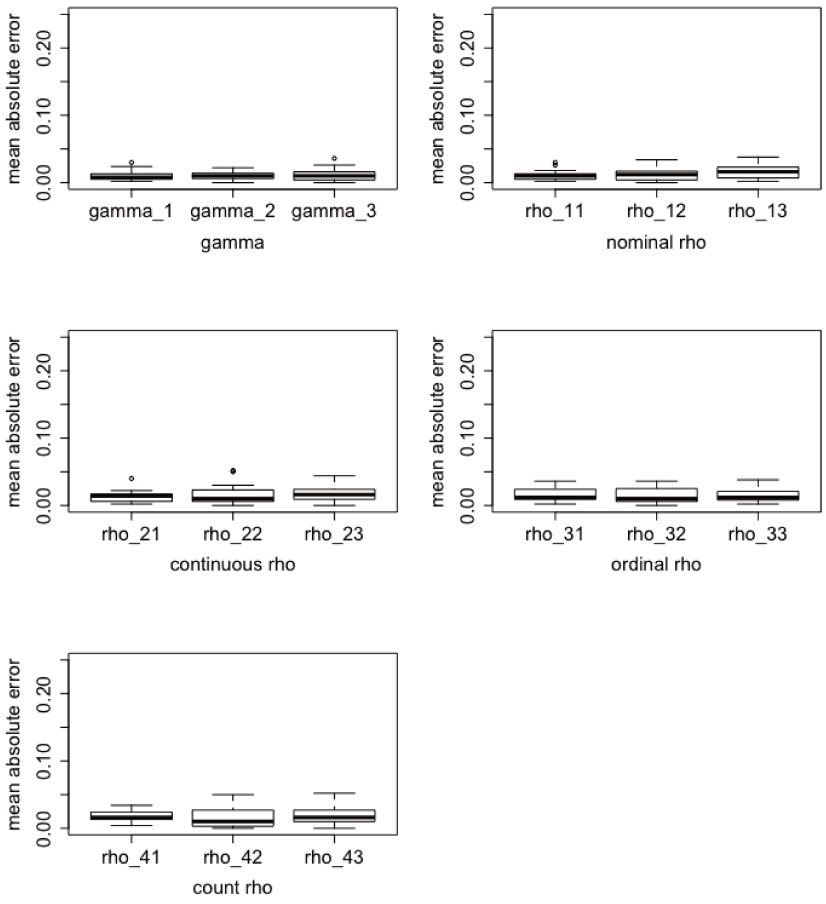

Figure 3.4 – Mean absolute error for the mixing proportions with $N = J_d = 500$.

Table 3.3 – Value of the block parameters mean absolute error on simulation with $N = J_d = 100$ for the continuous, ordinal and count matrices.

	Continuous μ, σ			Ordinal $m = 5$ μ, π			Count $\delta \times 10^{-5}$		
	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 1	col-cluster 2	col-cluster 3
row-cluster 1	0.01,0.01	0.03,0.02	0.02,0.02	0.00,0.05	0.00,0.03	0.00,0.05	0.16	1.89	1.33
row-cluster 2	0.04,0.02	0.00,0.00	0.00,0.00	0.00,0.04	0.00,0.03	0.00,0.05	0.87	1.97	1.4
row-cluster 3	0.00,0.00	0.01,0.01	0.01,0.01	0.00,0.02	0.00,0.03	0.00,0.03	0.34	0.83	1.06

Table 3.4 – Value of the blocks parameters mean absolute error on simulation with $N = J_d = 500$ for the continuous, ordinal and count matrices.

	Continuous μ, σ			Ordinal $m = 5$ μ, π			Count $\delta \times 10^{-7}$		
	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 1	col-cluster 2	col-cluster 3
row-cluster 1	0.2,0.03	0.3,0.09	0.1,0.08	0.00,0.04	0.00,0.03	0.00,0.01	0.1	0.2	0.1
row-cluster 2	0.01,0.02	0.1,0.1	0.1,0.06	0.00,0.02	0.00,0.03	0.00,0.04	0.5	0.1	0.1
row-cluster 3	0.00,0.00	0.01,0.01	0.01,0.01	0.00,0.03	0.00,0.02	0.00,0.03	0.3	0.2	0.3

3.4.2.b Partition estimation

The partition estimation is assessed using the Adjusted Rand Index, referred to as "ARI" Hubert and Arabie (1985). The ARIs for the row and column partitions, on the two simulated data sets are given in Table 3.5. We see that the co-clustering algorithm succeeds in finding the true partitions for the rows and columns.

Table 3.5 – Mean (standard deviation) ARIs for two data sets $N = 100$ and $N = 500$.

N	Rows	Categorical	Continuous	Ordinal	Count
100	0.98 (0.09)	0.95 (0.14)	0.98 (0.07)	1 (0.01)	0.98 (0.09)
500	1 (0.00)	1 (0.00)	1 (0.00)	1 (0.00)	1 (0.00)

3.4.3 Model selection

In this section, the ICL-BIC criterion's efficiency is assessed for choosing the right number of clusters by row and by column. Furthermore, the heuristic search described in 3.2.2.b is evaluated. The complexity of the problem should be emphasised here. Usually, criteria such as BIC or ICL are used to find the right number of clusters for the row partitions only. In the case of co-clustering, they are extended to find the right number of clusters for the row partitions and the column partitions. In the present work, it is used to find $(D+1)$ numbers of clusters (one for the rows, and one for each kind of feature). Mathematically, the search space is much larger which makes the problem more complex.

EXHAUSTIVE SEARCH Table 3.6 presents which sets (G, H_1, H_2, H_3, H_4) had the best ICL-BIC value in the exhaustive search. The number of occurrences indicates how many times the sets were chosen. Note that the right numbers of clusters $(G, H_1, H_2, H_3, H_4) = (3, 3, 3, 3, 3)$ has been chosen more often for the larger data set with $(N = J_d = 500)$. This result is consistent because the proposed ICL-BIC is based on asymptotic approximations. For the data set with $(N = J_d = 100)$, the model with $(G, H_1, H_2, H_3, H_4) = (3, 3, 3, 3, 3)$ is only the fourth one to be chosen. However, the average means of the ARI for the co-clustering when the model chosen was $(G, H_1, H_2, H_3, H_4) \neq (3, 3, 3, 3, 3)$, are equal to (0.94 (0.10), 0.93 (0.08), 0.98 (0.02), 0.98 (0.02), 0.98 (0.02)). This means that when the criterion for model selection does not find the true model, the algorithm still finds good partitions.

Table 3.6 – Exhaustive search results on 20 simulations results.

$N = J_d = 100$									
(G, H_1, H_2, H_3, H_4)	34333	34334	43333	**33333**	44333	34433	34443	44334	44433
number of occurrences	6	3	3	2	2	1	1	1	1

$N = J_d = 500$										
(G, H_1, H_2, H_3, H_4)	**33333**	33332	33323	43333	34342	32323	33343	34332	34322	44332
number of occurrences	6	3	2	2	2	1	1	1	1	1

Table 3.7 – Heuristic search results on 20 simulations results.

$N = J_d = 100$												
(G, H_1, H_2, H_3, H_4)	**33333**	22223	22233	22234	22243	22244	22334	23223	32223	33334	34334	37334
number of occurrences	5	3	3	1	1	1	1	1	1	1	1	1

$N = J_d = 500$					
(G, H_1, H_2, H_3, H_4)	**33333**	32233	22333	23234	32333
number of occurrences	10	5	3	1	1

HEURISTIC SEARCH Table 3.7 presents which sets (G, H_1, H_2, H_3, H_4) were chosen by the heuristic search. Once again, the algorithm works better for the larger data set with $(N = J_d = 500)$, although the results for $(N = J_d = 100)$ are good too.

SEARCH COMPUTATION TIME The simulations were run using a Linux 4.9.0-3-amd64 server, on Debian 9. For the data set with $N = J_d = 100$, the exhaustive search took 23 minutes, while the heuristic search took at most 18 minutes. For the data set with $N = J_d = 500$, the exhaustive search lasted 33 hours whereas the heuristic search took at most 2 hours. This means that in the case of a small data set, it can be interesting to run an exhaustive search, as it does not take much more time than the heuristic search. However, an exhaustive search as it was performed in this simulation requires knowledge of the neighborhood of the right set (G, H_1, H_2, H_3, H_4). For a larger data set, the heuristic search is recommended as it is very efficient and up to 15 times faster than the exhaustive search. Furthermore, we can expect it to be even more than 15 times faster in case of larger data sets than the ones used in these simulations.

3.4.4 More challenging data sets

In Section 3.4.1 the parameter settings for the continuous variables generate well separated clusters since the means are separate and the variances small. Besides, the optimal ARIs (ARIs obtained while knowing the parameters) in line and in column were always equal to 1. In this section, we change these parameters so that the clusters are not well separated with regard to the continuous variables. We used the simulated data set of Section 3.4.1 with 100 rows and 400 columns and changed the parameters of the continuous variables. In each block of the diagonal, we have $\mu = \epsilon$ and $\sigma = 1$. On the other blocks, $\mu = 0$ and $\sigma = 1$ (see Table 3.8). We performed the co-clustering algorithm 20 times for ϵ equals to 0.5 and 0.2. Then, we performed the co-clustering 20 times with the data set made of the continuous variables only. The optimal ARIs of the data set and the ARIs resulting from the co-clustering are given in Table 3.9. In this simulation, we see that when the co-clustering algorithm is performed only on the continuous variables, it does not distinguish the different blocks well. Indeed, the row-clusters are too mixed. However, when the other variables

Table 3.8 – Mean and Standard deviation for the blocks of continuous variables for the more challenging data sets cases.

	col-cluster 1	col-cluster 2	col-cluster 3
row-cluster 1	ϵ,1	0,1	0,1
row-cluster 2	0,1	ϵ,1	0,1
row-cluster 3	0,1	0,1	ϵ,1

Table 3.9 – ARIs for the more challenging data set case.

		optimal ARI	Rows ARI	Categorical ARI	Continuous ARI	Ordinal ARI	Count ARI
$\epsilon = 0.5$	all variables	(1,1,0.89,1,1)	0.96 (0.1)	1 (0)	0.85 (0.11)	0.98 (0.09)	0.96 (0.11)
	only continuous	(0.86,0.90)	0.69 (0.2)	-	0.7 (0.16)	-	-
$\epsilon = 0.2$	all variables	(1,1,0.28,1,1)	1 (0)	0.94 (0.17)	0.19 (0.12)	0.94 (0.16)	1 (0)
	only continuous	(0.14,0.28)	0 (0.03)	-	0 (0.03)	-	-

(categorical, ordinal and counting) are taken into account, the co-clustering succeeds in finding the true partitions. In addition, the good estimation of the row partitions obtained thanks to the non-continuous variables improves the column partitions estimation for the continuous variables.

3.4.5 Missing data

In this section, we investigate the behavior of the ARIs when missing values are introduced into the data. Again, we used the data set with 100 rows and 400 columns. We performed the co-clustering algorithm 20 times on the data set with 10%, 20%, 30%, 50% and 75% of missing values. Resulting ARIs for row and column partitions are given in Table 3.10. We see that up to 30% of missing values, the ARI does not changes significantly. However, with more missing values, the ARI for the partitions reduces.

Table 3.10 – ARIs for a data set with missing values.

ARI type	Rows	Categorical	Continuous	Ordinal	Counting
Original Simulation	0.98 (0.09)	0.95 (0.14)	0.98 (0.07)	1 (0.01)	0.98 (0.09)
10% NA	1 (0)	1 (0)	0.87 (0.2)	1 (0.01)	1 (0.01)
20% NA	1 (0)	1 (0)	0.88 (0.21)	0.99 (0.01)	0.99 (0.02)
30% NA	0.99 (0.04)	0.98 (0.1)	0.98 (0.07)	0.94 (0.14)	0.87 (0.14)
50% NA	0.59 (0.08)	0.99 (0.01)	0.98 (0.07)	0.93 (0.11)	0.76 (0.18)
75% NA	0.23 (0.13)	0.71 (0.07)	0.77 (0.14)	0.46 (0.08)	0.38 (0.2)

3.4.6 Conclusion

As a conclusion for this simulation study, the SEM-Gibbs algorithm is efficient in estimating the model parameters and the partitions. Regarding model selection, while we know that the ICL-BIC criterion leads to a consistent estimation of the number of blocks when the number of rows and column tends to infinity (see Keribin et al. (2013)), its behavior for finite sample size remains robust. Moreover, using the proposed heuristic search enables drastic reduction in computing time without significantly decreasing the performance of the estimation.

When the continuous variables have poorly separated clusters, the co-clustering succeeds in finding the true row partitions and the true column partitions of the other variables.

When up to 30% of missing values are introduced, the co-clustering succeeds in finding the true row and column partitions. When there are more than 30% of missing values, it is more difficult for the co-clustering to find the true partitions.

3.5 Real data applications

In this section, two real data sets are considered. The first one concerns the famous TED talks* and contains the transcripts and ratings of TED Talks uploaded to the official TED.com website until September 21st, 2017. It is a mixed data set because the transcripts are textual data whereas the ratings are numbers. The second data set is the result of a survey that Slovakian Statistic students gave to people around them. The responses were categorical with different numbers of levels and some of them were ordinal.

3.5.1 Co-clustering of count and continuous data

THE TED TALKS DATA SET TED is a non-profit organization which posts conferences on-line for free distribution. The conferences address a wide range of topics, including science, culture and innovation. The TED talks data set† contains information about 2 467 TED Talks. This work is focused on their transcripts and their ratings given by the users. The rating system is particular on this website. A list of fourteen words was defined (beautiful, inspiring, persuasive, fascinating, ok, longwinded, confusing, informative, courageous, ingenious, funny, obnoxious, unconvincing, jaw-dropping). A user wanting to rate a talk is asked to choose the three words that best describe the talk.

DATA SET PRE-PROCESSING First of all, a couple of TED talks were actually a musical performance. Their transcripts were of the form "(Applause)(Music)(Applause)", which is informationless, so these talks were removed from the data set. Then, the other transcripts were projected into a Document-Term matrix, each cell counting the occurrences of a term in a talk. It appears that some terms that occurred only once were onomatopoeia such as "aargh" and "aaaaaaaargh". These terms were removed: we assumed that these words do not bring valuable information, and that at the same time, removing them reduces the dimension of the matrix. The ratings variables were used without any changes: no normalization was performed as a pre-processing. In contrast with the Document-Term matrix, the ratings matrix is not sparse since only 1% of the values are equal to 0. The mean and standard deviation of the ratings matrix are equal to 175.2 and 538.2 respectively. The resulting matrix is therefore of dimension $(2\,464 \times (40\,137 + 14))$, in other words, $N = 2\,464$, $J_1 = 40\,137$ $J_2 = 14$. The data set is seen as two matrices of different types ($D = 2$). The first one is the Document-Term matrix of the transcripts whose occurrences are modeled by a Poisson distribution. The second matrix represents, for each talk, the number of users that voted for each of the words in the proposed adjectives list. Given the high number of votes, this number is modeled by a Normal distribution. This matrix could be seen as a counting matrix as well. However, the proposed Poisson model takes into account margins on rows and columns (see $n_{i\cdot}$ and $n_{\cdot j}$ in Section 3.3.4). These margins make sense on document-term matrices. However, on the rating matrix of this application, they are not as

*https://www.ted.com/talks

†https://www.kaggle.com/rounakbanik/ted-talks/data

relevant. Furthermore, the Gaussian distribution is more suitable because with a Poisson distribution, the mean is equal to the variance: over large numbers like those in the rating matrix, the Poisson parameters are less informative.

CO-CLUSTERING AS A PARSIMONIOUS CLUSTERING The main motivation on this data set is to cluster the TED talks to distinguish the different kinds of talks, and to observe the ratings of each row-cluster. Using a classical clustering technique is not conceivable because of the high dimension of the data set. The latent class model, for example, would define a distribution for each of the 40 151 variables and for each class, which is definitely over-parameterised and not interpretable. With a co-clustering technique not only will the talks be clustered, but the variables will be clustered as well, which will result in a small number of interpretable blocks.

CO-CLUSTERING RESULTS After having searched for the highest ICL-BIC as explained in Section 3.2.2.b with $(G_{min}, H_{1_{min}}, H_{2_{min}}) = (2, 2, 1)$, the best set (G, H_1, H_2) was found to be equal to $(8, 6, 2)$. Figure 3.5 gives a representation of the block parameters. For the Document-Term matrix, the δ parameters are represented by shades of gray. The lighter the block, the lower its corresponding δ parameter. When a block's δ parameter is high, this means that the column-cluster terms of this block are quite specific to the corresponding row-cluster. For the ratings matrix, the shades of gray represent the μ parameter of the resulting blocks. The darker the block, the higher the μ parameter.

First of all, we focus on the row-clusters of the document-term-matrix. Note from the titles of talks with the same row-cluster number that the co-clustering grouped talks with similar topics. For example the third group seems to be about high technology and science with titles such as "A robot that runs and swims like a salamander", "A mobile fridge for vaccines" and "The hunt for a supermassive black hole"; whereas the fourth group refers to politics, with talks called as "Why Brexit happened – and what to do next", "How ideas trump crises" , and "Aid for Africa? No thanks.". From the ratings row-cluster parameters, it can be seen that the seventh row-cluster's talks were rated about ten times more than the documents of the other row-clusters. It is interesting to observe that this corresponds to a row-cluster closely related to psychology and introspection. Table 3.11 gives an overview of the Document-Term matrix row-clusters, giving some titles and the topic that was deduced from them. On the other hand, two row-clusters were more difficult to interpret. For example, the eighth row-cluster gathers talks with titles such as "Dare to educate Afghan girls", "Averting the climate crisis", "Fighting with nonviolence" and "What it's like to be a parent in a war zone". While the talks tend to be about education and parenting, the inherent topic is not obvious nor unique. The same issue was observed with the third row-cluster: with titles such as "The magic of Fibonacci numbers", "A new equation for intelligence" and "New thinking on the climate crisis", it is hard to define a unique subject for this group.

It is not easy to interpret directly the terms clusters because these column-clusters contain on-average about 6 000 variables. However, we have extracted some of the 100 most frequent words for some notable blocks with high δ parameters to check if they are relevant to the row-clusters' topics of Table 3.11. Firstly, from Figure 3.5, on the left, block $(6, 1)$, corresponding to the 6^{th} row-cluster and 1^{st} column cluster, was noted. Among the most frequent words are "knowledge", "future", "company", "information", "community", "working", and"imagine", which are relevant to the 6^{th} row-cluster topic about innovation and high-technology. Similarly, block $(4, 5)$ was noted. Some of the most frequent terms are "phenomenon", "coffee", "discovery", "organisms", and "suffering", which correspond to

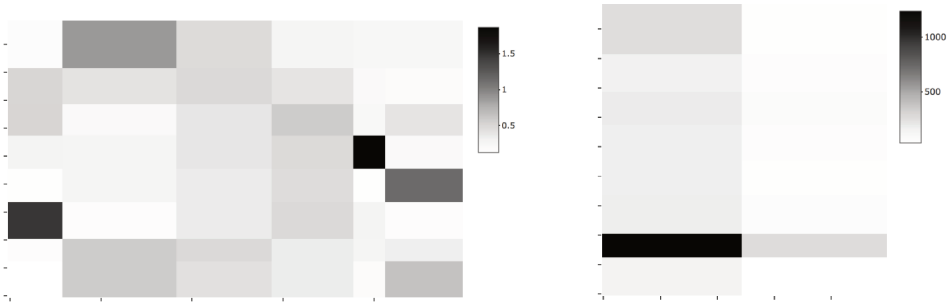

Figure 3.5 – Block representation of the Document-Term matrix (left) and of the ratings matrix (right). The shades of gray represent the δ parameter of each block for the Document-Term matrix. For the rating matrix, they represent the μ parameter.

the 4^{th} row-cluster topic about medicine and health. Finally, block $(5, 6)$ was investigated. It appears that its column-cluster terms are specific to the 5^{th} row-cluster about politics, with the words "india", "history", "technology", "program", and "impact" among the most frequent ones.

We now consider the column-clusters of the ratings for the TED talks. The adjectives were split into two groups. The first column cluster is composed of the following adjectives: "Inspiring", "Beautiful", "Courageous", "Persuasive", "Fascinating", "Informative", and "Funny". These adjectives were on average voted for more than those of the second column-cluster, and this for all the row-clusters. The second column-cluster is made up of the adjectives "Ingenious", "Confusing", "Jaw-dropping" "Obnoxious", "Longwinded", "Unconvincing" ,and "OK".

From these observations, we can conclude that the co-clustering results helped provide understanding and a summary of the data set. Firstly, it clustered the TED-talks documents. The resulting classes were relevant regarding the titles topics and corresponding term column-clusters. Furthermore, the rating matrix gives information about the kinds of talks preferred. Overall, the co-clustering results gave an overview of a big data set that cannot be done easily by a human.

CO-CLUSTERING OF EACH SET SEPARATELY In this section we perform the co-clustering algorithm on the Document-Term matrix and on the ratings matrix separately. We also compare these results with the co-clustering performed with both matrices thanks to the MLBM. On the Document-Term matrix, the row and column partitions ARIs were on average equal to 0.36 (0.04) and 0.30 (0.04). On the ratings matrix, the row and column partitions ARIs were on average equal to 0.04 (0.00) and 0.27 (0.16). This means that the co-clusters obtained with separate matrices are very different from the co-clusters obtained using the MLBM. In particular, co-clustering of each set separately is not relevant to providing a unified row partition.

3.5.2 Co-clustering of ordinal and nominal data

YOUNG PEOPLE RESPONSES TO QUESTIONNAIRES In 2013, Slovakian students of a statistics class were asked to invite their friends to participate in a survey that concerned several

Table 3.11 – Row-cluster interpretation for the TED talks data set.

Row-cluster number	Example titles	Interpreted topics
1	"My year of living biblically", "My journey from Marine to actor", "How I'm preparing to get Alzheimer's", "12 truths I learned from life and writing", "The year I was homeless"	Story-telling
2	"Art that craves your attention", "Building a museum of museums on the web", "How to engineer a viral music video", "A one-man orchestra of the imagination", "Moving sculpture"	Art, Culture
3	"The magic of Fibonacci numbers", "How behavioral science can lower your energy bill", "New thinking on the climate crisis", "A new equation for intelligence", "Winning the oil endgame"	Energy, Climate, Mathematics
4	"A map of the brain", "Your brain hallucinates your conscious reality", "Is anatomy destiny?", "Growing new organs", "A doctor's case for medical marijuana"	Medicine, Health
5	"Why Brexit happened – and what to do next", "How ideas trump crises", "Aid for Africa? No thanks.", "The surprising way groups like ISIS stay in power", "The attitudes that sparked Arab Spring"	Politics
6	"A robot that runs and swims like a salamander", "A mobile fridge for vaccines", "The hunt for a supermassive black hole", "Hands-on science with squishy circuits", "How we'll find life on other planets"	High technology, Science, Innovation
7	"Who are you, really? The puzzle of personality", "How to succeed? Get more sleep", "Your body language may shape who you are", "A kinder, gentler philosophy of success", "What really matters at the end of life"	Psychology, Introspection
8	"What it's like to be a parent in a war zone", "Teachers need real feedback", "Averting the climate crisis", "Dare to educate Afghan girls", "Fighting with nonviolence"	Education, Crisis

aspects of their life [‡]. The responses were defined on different scales; for example, a question such as "I enjoy listening to music." could be answered from 1 ("Don't enjoy at all") to 5 ("Enjoy very much"). The questions regarding music preferences, movie preferences, hobbies and interests, spending habits and phobias are seen as 5 levels ordinal data, not only because the answers are on a scale, but also because two answers can be compared. For example, questions concerning the music preferences could be : "I enjoy classical music." or "I enjoy rock music.", and both could have a reply on a scale from 1 ("Don't enjoy at all") to 5 ("Enjoy very much"). In this case, the order in the responses is clear, and one can easily compare the two answers of a same user. However, in the case of personality traits, views on life and opinion, questions could be: "I have to be well prepared before public speaking."or "I always keep my promises.", still on a 5 level scale from 1 ("Strongly disagree") to 5 ("Strongly agree"). The order of the responses can not be compared, so considering them to be ordinal makes their interpretation too arbitrary. That is why these questions were considered to be categorical variables, with a number of levels equal to 5. Furthermore, demographic questions such as "What is my gender?", with responses "Female" and "Male" are modeled as categorical variables with 2 levels. This survey was completed by 1 010 people.

Thus, the resulting matrix is of dimension $(1\,010 \times (80+5+54))$, so $N = 1\,010$, $J_1 = 80$, $J_2 = 5$ and $J_3 = 54$. The data set is seen as three matrices of different types. The first contains the 80 questions with answers considered as ordinal, with 5 levels. The second contains the 5 questions with answers considered as nominal, with 2 levels. Finally, the third contains the 54 questions with answers considered as nominal, with 5 levels.

Finally, the data set had a small amount of missing data (0.4%), which will be estimated using the SEM-Gibbs algorithm as described in Section 3.2.2.

CO-CLUSTERING RESULTS The SEM-Gibbs algorithm used 150 iterations and the burn-in period was set at 100 iterations. These numbers were defined using the same technique as in Section 3.5.1, by checking the evolution of several parameters through the SEM-Gibbs iterations. The best set (G, H_1, H_2, H_3) was found to be equal to $(3, 4, 2, 4)$. Figure 3.6 shows the resulting co-clustering, and Table 3.12 gives the estimated parameters of each block.

First of all, we notice that the first row-cluster has the lowest position parameter μ on the first column cluster of ordinal data. This means that people from this group have less overall enjoyment – or are less interested in, or are less afraid of – the topics of this column cluster's questions. These topics included classical music, branded clothing, psychology, politics and dangerous dogs. In addition, the parameters show that this row-cluster is quite heterogeneous. This row-cluster has the lowest position parameters π on the two first column-clusters of ordinal data, and they systematically have the highest β_1 and β_5 on categorical data with 5 levels. We will now consider the second row-cluster. We notice that it has a β_3 parameter equal to 0.5 on the personality questions first column-clusters, which is high. It means that people from this row-cluster are quite indecisive about the topics of these column-clusters. The questions included "I am 100% happy with my life.", "I believe all my personality traits are positive.", "I have lot of friends.", and "My moods change quickly.".

Finally, we analyse the fourth column-cluster of the ordinal variables. We notice that it has the highest position parameter for all the row-clusters with $\mu = 5$. The questions of this column-cluster are: "I enjoy listening to music", "I enjoy watching movies", "I enjoy comedies", "I am interested in internet", and "I am interested in socializing". It means that

[‡] https://www.kaggle.com/cardot/se-young-people-survey/data

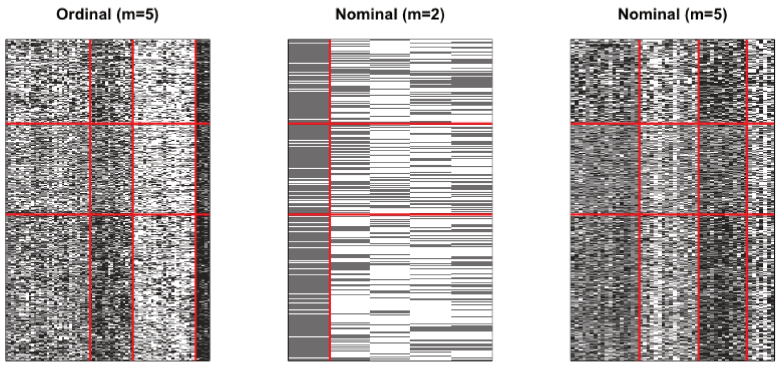

Figure 3.6 – Co-clustering result on Young People Survey.

the interviewed people are in overall agreement about being very interested in these topics.

Table 3.12 – Resulting co-clustering parameters for the student survey data set.

	Ordinal ($m = 5$) μ, π				Nominal ($m = 2$) β_1, β_2	
	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 4	col-cluster 1	col-cluster 2
row-cluster1	1,0.08	5,0.16	1,0.41	5,0.69	0.1,0.9	0.63,0.37
row-cluster 2	3,0.25	3,0.21	1,0.31	5,0.52	0.11,0.89	0.62,0.38
row-cluster 3	3,0.14	5,0.28	1,0.24	5,0.74	0.1,0.9	0.7,0.3

	Nominal ($m = 5$) $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$			
	col-cluster 1	col-cluster 2	col-cluster 3	col-cluster 4
row-cluster1	0.10,0.13,0.32,0.2,0.25	0.3,0.15,0.25,0.12,0.18	0.10,0.10,0.19,0.16,0.45	0.39,0.10,0.15,0.09,0.27
row-cluster 2	0.02,0.14,0.5,0.28,0.06	0.13,0.29,0.38,0.16,0.04	0.02,0.13,0.34,0.33,0.18	0.23,0.23,0.26,0.16,0.12
row-cluster 3	0.03,0.11,0.36,0.35,0.15	0.16,0.25,0.31,0.18,0.10	0.03,0.08,0.20,0.31,0.38	0.24,0.16,0.19,0.18,0.23

3.6 Analysing a quality of life survey in oncology - Use case

This work was published in Selosse et al. (2019). In this Section, we use the MLBM with a different point of view as it was used until now. The data set contained variables that were not necessarily different in terms of nature, but that were different in term of semantics. Therefore, running the co-clustering without imposing any constraint on these variables did not make any sense since some variables could be gathered into the same group, which could not make any sense to interpret the results. The data set is a survey carried out among women affected by breast cancer (Cousson-Gélie (2014)). Individuals with cancer usually experience traumatizing hardships such as chemotherapy and intense stress. The disease and its treatment have an impact on different domains of their environment, such as their social life or emotional state. In psychology, these domains are divided into dimensions. For example, in Table 3.13, the domain *quality of life* is divided into six dimensions (physical functioning, role functioning, social functioning, emotional functioning, cognitive functioning and global health evaluation). In contrast, the domain *emotional state* is defined using two dimensions: anxiety and depression (Zigmond and Snaith (1983)). Other psychological

dimensions have been identified as a quality of life predictor, such as perceived control of the illness, which corresponds to the general belief whereby evolution of the disease depends either on internal factors (action, effort or personal abilities) or on external factors (luck or destiny) (Cousson-Gélie (2014)), or social support, which assesses perceived availability (number of people on whom the individual thinks they can count if necessary) and the degree of satisfaction relating to this support (Sarason et al. (1983)).

The patients were asked to reply to various questionnaires related to distinct dimensions, the answers being of the ordinal kind with different numbers of levels (Agresti (2010)). They repeated this task at six different stages of their treatment. Therefore, the resulting dataset comprises a set of six tables, with the rows representing the patients and the columns representing the questions. First of all, the psychologists sought to identify psychological profiles. In particular, they wanted to analyse the mutual influence of the different dimensions for each profile. To help them with this task, constrained co-clustering was performed.

3.6.1 Data set

3.6.1.a Description of survey population

Several questionnaires were given to $N = 161$ women having their first surgery for suspicious breast tumour. These patients were between 31 and 77 years old with an average age of 56.25 years (standard deviation = 9.99). Most were married or living maritally (77.0%). Nearly half of the patients were active professionally (49.7%) and 38.5% were retired when they started the study. These 161 patients were asked to answer several questionnaires at different stages of their treatment: one at their first surgery, and followed by a questionnaire 1, 4, 7, 10, 13 months after this assessment. This means that the patients replied six times to 134 questions and each answer was given on an ordinal scale (with between four and seven levels). Therefore, the dataset comprises a set of six matrices of ordinal data such that the observations (rows) correspond to the patients, and the variables (columns) correspond to the questions.

The dataset also contains missing values, for which we distinguish two types. The first type occurred when some patients did not answer any of the questions at one of the six stages (i.e. they did not return the questionnaire at this stage). In this case, when co-clustering was performed solely on the answers for this stage, the rows corresponding to these patients were placed in a special row-cluster called “did not answer”. Co-clustering was then performed without taking them into account. The second type occurred when some patients failed to answer to only a couple of questions (i.e. they returned an incomplete questionnaire). In this case, the patient was taken into account for co-clustering, and the missing values (18 values in total) were estimated by the algorithm. The way the algorithm deals with this type of missing data is described later on.

3.6.1.b Psychological dimensions

The questionnaires given to the patients were detailed; indeed, the design of questionnaires is a highly specialized undertaking in psychology. Each questionnaire relates to domains of life, and each domain is itself divided into dimensions (e.g. MaloneBeach and Zarit (1995)). Table 3.13 lists the domains and the corresponding dimensions included in the study. In the questionnaires, most of the questions are associated with a dimension. The few questions that are not related to one of these psychological dimensions concern the symptoms of the disease and its treatment (nausea, tiredness, etc.).

Table 3.13 – Table of domains and dimensions raised in the questionnaires.

Domains				
Quality of life (Aaronson et al. (1993))	Social Support (Sarason et al. (1983))	Specific Social Support (Pierce et al. (1997))	Emotional State (Zigmond and Snaith (1983))	Control perception (Cousson-Gélie (2014))
Dimensions				
Physical functioning, Role functioning, Emotional functioning, Cognitive functioning, Social functioning, Global health evaluation.	Satisfaction, Quantity.	Intensity, Perception of availability, Conflict.	Anxiety, Depression.	Causal attribution, Control perception, Religion control.

Table 3.14 – Co-clustering result on anxiety, depression and symptom dimensions: estimated BOS parameters (μ_{gh}, τ_{gh}) for each cluster (g, h).

	Anxiety		Depression		Symptom	
	Col.-cluster 1	Col.-cluster 2	Col.-cluster 1	Col.-cluster 2	Col.-cluster 1	Col.-cluster 2
Row-cluster 1	(2,0.77)	(2,0.77)	(1,0.70)	(2,0.83)	(2,0.46)	(1,0.74)
Row-cluster 2	(2,0.68)	(3,0.72)	(2,0.47)	(2,0.79)	(3,0.39)	(1,0.42)
Row-cluster 3	(1,0.64)	(2,0.44)	(1,0.77)	(2,0.70)	(2,0.58)	(1,0.71)
Row-cluster 4	(1,0.67)	(2,0.47)	(1,0.79)	(2,0.71)	(1,0.80)	(1,0.93)
Row-cluster 5	(2,0.72)	(3,0.55)	(2,0.64)	(2,0.75)	(2,0.66)	(1,0.77)

3.6.2 Application to the survey dataset

3.6.2.a Constrained co-clustering with different dimensions

Several constrained co-clustering operations were performed on the dataset, with different dimensions and at different times. This section presents some significant results that were obtained. In the following experiments, the heuristic search described in Section 3.2.2.b was executed with $G_{min} = 3$ and $H_{d_{min}} = 1$ to select the number of row-clusters and column-clusters (G and $(H_d), d \in \{1, ..., D\}$). All the ICL-BIC values are available in the appendix. The choice of a sufficient number of iterations for the SEM algorithm and for the burn-in period was made empirically. It was noticed that the parameters would stabilize after 150 iterations (or fewer). Therefore, the burn-in period was set to 400 iterations and the total number of iterations was fixed at 500.

3.6.2.b Anxiety, depression and symptom.

As a first experiment, it was decided to investigate the responses that were given at time $T5$, at the end of the treatment. The questions regarding the symptoms of the treatment are interesting at this time because it marks the point at which the patients had been receiving chemotherapy for one year. Constrained co-clustering was performed by taking the questions related to the anxiety, depression and symptom dimensions. In this case, all the questions have a number of levels m equal to 4. Therefore, the only constraint is the separation of the questions that are related to different dimensions. The execution time of this set-up is about 12 seconds with a 2.00GHz Intel Xeon E5 2620 CPU and 8 Go of RAM. The result of the constrained co-clustering operation is illustrated by Figure 3.7. For all the figures, clusters are read from left to right and from top to bottom. Table 3.14 details the estimated BOS parameters (μ_{gh} and τ_{gh}) for $g \in \{1, ..., G\}$ and $h \in \{1, ..., H_d\}, \forall d \in \{1, ..., D\}$.

Five row-clusters are highlighted by the co-clustering results. Table 3.14 shows that the position parameters of the second row-cluster $(\mu_{2h})_{d,h}$ are generally greater than (or equal

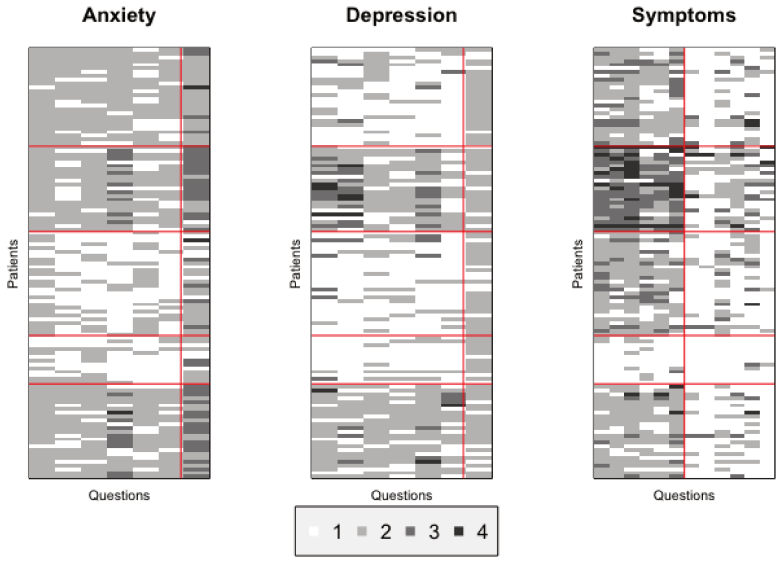

Figure 3.7 – Results of constrained co-clustering on anxiety, depression and symptom dimensions.

to) those of the other row-clusters. This means that the second group feels more anxiety and depression, and feels the disease symptoms more intensively than the other groups. It can also be seen that the fourth row-cluster is less inclined to anxiety and depression and suffers less from the symptoms than the others groups; indeed, parameters $(\mu_{4h})_{d,h}$ are generally the lowest. Furthermore, the precision parameters $(\tau_{4h})_{d,h}$ are quite high for this row-cluster, which means that the answers show limited spread around the position $(\mu_{4h})_{d,h}$. By observing the results for these two groups, it is possible to establish that the degree to which symptoms are felt is closely linked to signs of anxiety and depression, which is a fairly logical and intuitive result. However, the first, third and fifth groups provide more information. They are effectively very similar in terms of the degree to which they suffer the disease symptom dimensions. However, they differ a great deal in the first column-cluster of anxiety, and the first of depression. This means that even if a link between symptoms, anxiety and depression can be deduced from initial observations, it is not fully confirmed when people do not experience the symptoms at the extremes ("Very much" or "Not much"). The column-clusters offer interesting results as well: there is a clear separation between the symptoms. By examining the questions in each cluster, it becomes clear that the questions in the first cluster exclusively deal with pain and fatigue, while the second cluster deals with other symptoms such as nausea or loss of appetite. The co-clustering operation therefore detected two sub-dimensions for the symptoms dimension. Furthermore, there is a big difference in how the patients feel these two clusters: it is clearly noticeable that the position parameters $(\mu_{g1})_{(symptoms)}$ are generally higher than $(\mu_{g2})_{(symptoms)}$. Therefore, all the patients in general suffer more from pain and fatigue than the other symptoms.

3.6.2.c Social support: satisfaction, availability, intensity and conflict.

As a second experiment, the questions related to social support were used. The responses are taken from the fourth stage of the survey. This is in the middle of the patients' treatment, so they have already experienced a great deal, but know that they have to keep going for a few

Table 3.15 – Co-clustering result on social support dimensions: estimated BOS parameters (μ_{gh}, τ_{gh}) for each cluster (g, h).

	Satisfaction	Availability		Intensity		Conflict		
	Col.-cluster 1	Col.-cluster 1	Col.-cluster 2	Col.-cluster 1	Col.-cluster 2	Col.-cluster 1	Col.-cluster 2	Col.-cluster 3
Row-cluster 1	(2,0.90)	(1,0.72)	(1,0.96)	(3,0.48)	(1,0.59)	(4,0.80)	(3,0.24)	(1,0.62)
Row-cluster 2	(3,0.87)	(1,0.64)	(1,0.50)	(3,0.46)	(2,0.48)	(4,0.47)	(3,0.42)	(1,0.49)
Row-cluster 3	(1,0.73)	(2,0.72)	(1,0.86)	(3,0.52)	(2,0.63)	(4,0.59)	(3,0.51)	(1,0.44)
Row-cluster 4	(2,0.79)	(1,0.61)	(2,0.64)	(3,0.31)	(2,0.50)	(3,0.27)	(3,0.32)	(1,0.44)
Row-cluster 5	(1,0.93)	(1,0.78)	(1,0.91)	(3,0.27)	(1,0.68)	(4,0.71)	(3,0.18)	(1,0.63)

more months. Their perception of the social support they receive is therefore interesting at this point. This aspect includes questions relating to four dimensions: satisfaction (where the number of levels $m = 6$), perception of availability, intensity and conflict (where the number of levels $m = 4$). The questions that relate to the same dimension have the same number of levels. Again, the only constraint is the separation of the questions that are not related to the same dimensions. The result of the constrained co-clustering operation is illustrated by Figure 3.8. Furthermore, Table 3.15 details the estimated BOS parameters (μ and τ) for each co-cluster.

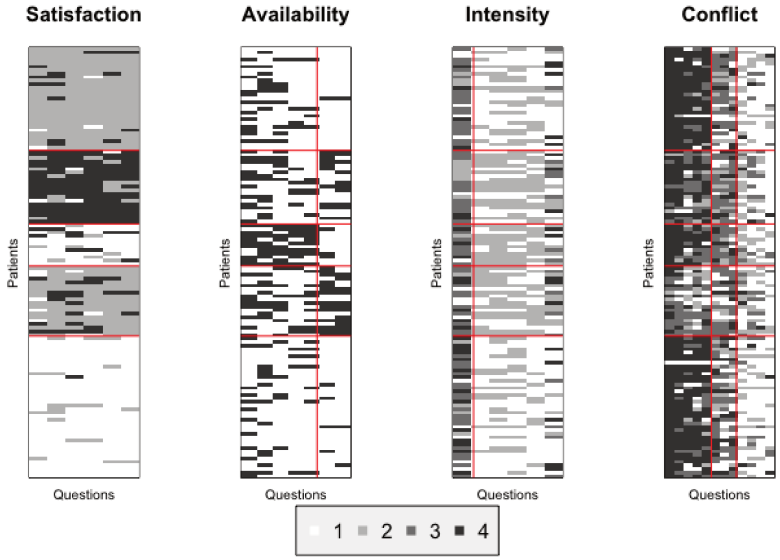

Figure 3.8 – Result of constrained co-clustering on dimensions related to social support.

The co-clustering operation detected five row-clusters. The third and the fifth are clearly satisfied with the social support they receive. Indeed, their position parameters $\mu_{31(satisfaction)}$ and $\mu_{51(satisfaction)}$ are equal to 1. Furthermore, the precision parameters $\tau_{31(satisfaction)}$ and $\tau_{51(satisfaction)}$ are very high, which means that most of the patients effectively gave the most positive response to the questions regarding their satisfaction. In contrast, the women in the first group are quite dissatisfied with their social support compared to the others. Another result is that the third group, one of the most satisfied, has one of the worst levels of perception of availability among their close family and friends ($\mu_{31(availability)} \geq \mu_{g1(availability)}$). Furthermore, it is also interesting to observe the column-clusters that were detected by the co-clustering operation for the conflict dimension. The first group of questions is about the effort the patient has to make to avoid conflict with their loved ones.

Table 3.16 – Co-clustering results for the symptoms dimension, at three different times: estimated BOS parameters (μ_{gh}, τ_{gh}) for each co-cluster (g, h).

	T0 Symptoms		T2 Symptoms		T25 Symptoms	
	Col.-cluster 1	Col.-cluster 2	Col.-cluster 1	Col.-cluster 2	Col.-cluster 1	Col.-cluster 2
Row-cluster 1	(2,0.20)	(1,0.67)	(2,0.62)	(1,0.72)	(2,0.64)	(1,0.74)
Row-cluster 2	(2,0.09)	(1,0.62)	(3,0.43)	(1,0.40)	(3,0.42)	(1,0.46)
Row-cluster 3	(1,0.66)	(1,0.84)	(1,0.58)	(1,0.85)	(1,0.54)	(1,0.84)

The second group comprises questions about changes in relationships, while the last cluster concerns feelings of anger towards close family and friends.

3.6.2.d Symptoms at different times

In this experiment, the questions related to symptoms were selected at different stages (at times $T0$, $T2$ and $T5$). The constraint is therefore not to separate the questions from different dimensions, but to separate the questions that are from different stages. The point of performing co-clustering on such a dataset is that the row-clusters group together individuals who evolved in a similar way regarding this dimension. Furthermore, the column-clusters provide information about how the patients' symptoms generally worsened (or improved) throughout the treatment. BOS parameters for this experiment are available in Table 3.16, and Figure 3.9 illustrates the results.

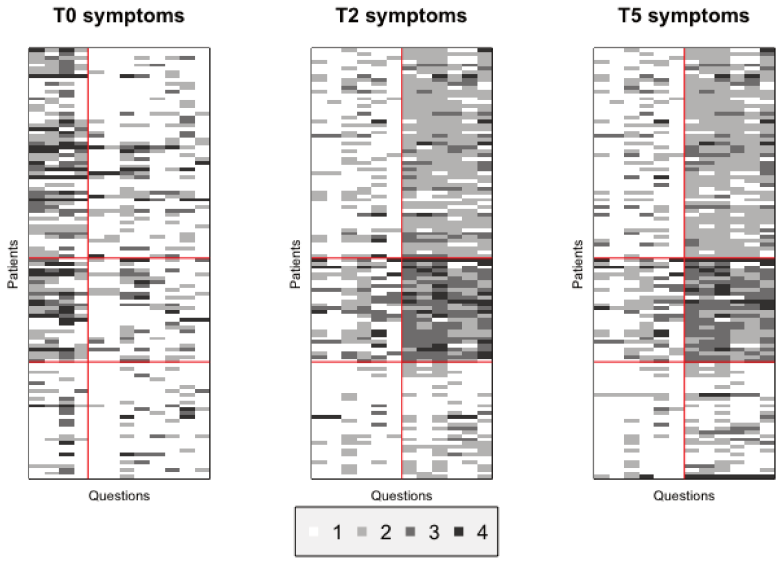

Figure 3.9 – Co-clustering results for questions related to symptoms, at three different times.

The co-clustering operation highlights three row-clusters. The third groups together people who felt the symptoms of the disease to a lesser degree than the others: the position parameters $(\mu_{3h})_{d,h}$ are all equal to 1. Furthermore, the precision parameters $(\tau_{3h})_{d,h}$ are fairly high, which implies that the responses show limited spread around the value 1. It is also interesting to investigate how the column-clusters evolve. To begin with, for each time, the symptoms are separated into two column-clusters: the first is systematically worse overall than the second, because $(\mu_{g1})_{(T0,T2,T5)} \geq (\mu_{g2})_{(T0,T2,T5)}, \forall g \in \{1, ..., G\}$. It is observed

that at time $T0$ there are fewer symptoms in column-cluster 1 than in column-cluster 2, whereas they are equally shared at times $T2$ and $T5$.

3.7 Conclusion and perspectives

This chapter presents a model-based co-clustering model for data sets made of mixed type data. It relies on the latent block model and inference is performed using an SEM-Gibbs algorithm. The method has the great advantage of having an efficient criterion to select the number of row and column clusters. Furthermore, the parameters that are estimated on each block allow the user to easily interpret the partitions. Finally, missing data is handled, which is often useful in the case of real data sets. The efficiency of the algorithm was illustrated on a simulated data set and then on real data. Moreover, if a user is interested in clustering the observations, the co-clustering algorithm proposed gives a parsimonious way to do this, by grouping all the features into a small number of clusters. The R package mixedCoclust, which implements the MLBM for mixed data is available on CRAN. Furthermore, if the reader is particularly interested in handling ordinal data, as in Section 3.6, the package ordinaClust is also available on CRAN, and is able to cluster and classify ordinal data. We implemented both packages in C++ so that the algorithms run faster.

The proposed model has certain limitations. A major issue is that the variables of different types cannot be part of the same column-cluster as the model is based on the assumption that the elements of a same block share the same distribution. It would be interesting to find an approach to overcome this limitation. Furthermore, as noted in Section 3.3.1, the way the data is encoded can have a strong impact on the resulting co-clustering partition. Although there are ways to address the matter in some cases, as detailed in Biernacki and Lourme (2019), the user should be aware of it. Additionally, the influence of each kind of feature on the resulting row partitions is to be investigated more deeply in a future work. Indeed, certain types of data will have more impact on the probability for a row belonging to a particular row-cluster, even if the D matrices have the same number of features J_d. Also, the case where the J_d are highly unbalanced should be studied. An interesting approach could be to give the same importance to the D matrices, even if they do not have the same number of features. Finally, the way nominal and ordinal variables are modeled can raise the dimensionality of the problem. When the number of nominal and/or ordinal variables with differing levels increases, the number of sets x^d increases. However, the number of parameters will not significantly increase, because the proposed model is very parsimonious. In addition, even though it may significantly increase the number of competing models, the negative impact on the model selection process time will be limited thanks to the heuristic search procedure introduced in Section 3.2.2.b.

4

Self-Organised Co-Clustering

4.1 Introduction

While textual data has existed for centuries, its occurence, use and ease of access has exploded in recent years, thanks in particular to the Internet. Social networks have largely driven this phenomenon: in 2019, Twitter had almost 474,000 tweets per minute and Facebook reported 4.3 billion messages posted per day. Access to an infinite number of resources via forums, the digitisation of newspapers and the creation of websites are also other important factors.

However, since text is an unstructured type of data, its analysis is not trivial and requires the use of special methods. The representation of text alone is a challenge, as various recent papers have shown Wu et al. (2018); Thongtan and Phienthrakul (2019). Most problems related to the analysis of textual data are still open issues, and are challenged by strong technological obstacles. Therefore, when users deal with a large unknown corpus, they often need - as a first step - a global overview of their data set. In other words, users often need to summarise their data, for example by knowing which documents share the same topics and the main topics of each cluster. The most famous way to do this is probably the Latent Dirichlet Allocation model (LDA, Blei et al. (2003)), which proposes a probabilistic modelling of the words appearing in the documents. Many extensions of LDA have been proposed over the years. For instance, recently, Mantyla et al. (2018) combines LDA and clustering algorithms to highlight the main topics of their clusters. In Drosatos and Kaldoudi (2019), the authors analyse scientific literature related to the field of e-Health. In Yan et al. (2013), the authors describe the Biterm Topic Model (BTM). It outperforms LDA on short texts (such as instant messages and tweets) for which LDA performs poorly, due to the sparsity of the data. In Zhu et al. (2018), the authors propose another version of the BTM: they represent the biterms (word-pairs) as graphs and use a deep convolutional network to encode word co-relationships.

This chapter presents the Self-Organised Co-Clustering model (SOCC), published in Selosse et al. (2020c). It aims at providing a tool to summarise large document-term matrices, whose rows correspond to documents and columns correspond to terms. Co-clustering was already successfully used to analyse textual data sets. For example, in Buono and Pio (2015) and Salah et al. (2018), the authors use distance based methods such as the Non-Negative Matrix Factorisation to cluster the data set. In Laclau and Nadif (2016) the authors propose a co-clustering using a double k-means, and impose that the meaningful blocks are on the diagonal. However, as mentioned in Section 2.3 of Chapter 2, these kinds of methods require choosing the metric $||.||$ that best fits the structure of the underlying latent blocks based on available data, which can be difficult. Furthermore, to the best of our knowledge, these methods do not propose a way to select the correct number of blocks.

Model-based co-clustering approaches were also proposed to analyse textual data. For example, in Singh Bhatia et al. (2017), the Latent Block Model is used with the Poisson distribution. In Selosse et al. (2020b), we also used the Multiple Latent Block Model to analyse textual data along with continuous data.

However, when dealing with high-dimensional sparse data, several blocks may be mainly sparse (composed of zeros) and cause inference issues. In addition, highlighting homogeneous blocks is not always sufficient to obtain easy-to-interpret results. Indeed, despite being homogeneous, these sparse blocks are not relevant from an interpretation perspective, and we need a new step to select the pertinent blocks. In other words, it is left to the user to choose the most useful co-clusters and to determine which term clusters (column-clusters) are more specific to which document clusters (row-clusters). This task is not straightforward even with a reasonable number of row and column-clusters. Therefore, it is necessary to work on a co-clustering technique that offers ready-to-use results.

We can address this problem by imposing a pattern on the co-clustering structure. Such an approach directly produces the most meaningful co-clusters, and significantly simplifies the results and their analysis. In the present work, we propose a co-clustering approach based on the Latent Block Model Govaert and Nadif (2013), in which we impose a structure wherein column-clusters (clusters of terms) are separated into three parts. In the first part, each cluster of terms is specific to one cluster of documents. In the second part, each cluster of terms is specific to two clusters of documents. The third part contains only one

column-cluster and gathers terms that are common to all clusters of documents. The main motivation of this paper is to provide a tool with high comprehensibility: having three sections offers explicable results, with a reasonable number of co-clusters. The choice to constrain our model to pairwise interactions between clusters is essentially motivated by the classical ANOVA modelling, which is usually limited to the two-way analysis. Furthermore, pairwise interactions are more interpretable than higher order interactions, and interactions between more than three factors are expected to be infrequent. Figure 4.1 illustrates the proposed structure. On the left, we present a usual co-clustering with the Poisson Latent Block Model. On the right, we show a co-clustering with the SOCC structure: thin separations between the three parts of column-clusters were added, with the noisy blocks as the lighter ones.

Other works have introduced a structure in their related co-clustering. In Laclau and Nadif (2017) and Ailem et al. (2017), the authors propose block diagonal co-clustering techniques, with binary and counting data respectively. Firstly, this consists of constraining the co-clustering such that the number of row-clusters is equal to the number of column-clusters. Secondly, the blocks out of the diagonal are considered to be noisy and share the same parameter. In fact, these models are particular cases of the model we propose: they constrain the structure to only the first part of column-clusters mentioned above. While these methods proved their efficiency in the case of document-term matrices, they assume that a cluster of terms is specific to only one cluster of documents. However, a group of terms could be specific to several groups of documents. Let us assume for instance that the documents are research papers, with one cluster related to computer science and another one related to mathematics. Each cluster has its own specific terms but many terms (for instance those related to probability distributions) will appear in both communities. In this work, we address this issue by defining a more complete structure among blocks without losing interpretability.

Section 4.3 describes the novel method referred to as 'Self-Organized Co-Clustering' (SOCC). In Section 4.4, we assess the efficiency of our solution in three ways. Firstly, we use simulated data, to evaluate the partition estimation of the SOCC model and state-of-the-art competing models. Secondly, we use real textual data sets to compare the proposed approach with these models, regarding both document clustering and term clustering. Thirdly, we describe a use case of the SOCC model on a real labelled data set. In Section 4.5, we detail an example for using the SOCC model in a truly unsupervised context. The last section concludes the paper and discusses topics for possible future research.

4.2 Reminders on the Latent Block Model for counting data

In this section, we briefly recall the backgroung for the Poisson Latent Block Model (PLBM). A more detailed explaination is available in Section 3.3.4 of Chapter 3.

4.2.1 The Poisson Latent Block Model (PLBM)

Counting data, such as those present in document-term matrices, can be modelled using the Poisson distribution. For a x_{ij} belonging to block (g, h) the Poisson distribution is parameterised with λ_{ij} such that $\lambda_{ij} = n_{i.}n_{.j}\delta_{gh}$. Here, the values $n_{i.}$, $n_{.j}$ correspond to a 'row effect' and a 'column effect' respectively, and are computed as follows:

$$n_{i.} = \sum_j x_{ij} \text{ and } n_{.j} = \sum_i x_{ij}.$$

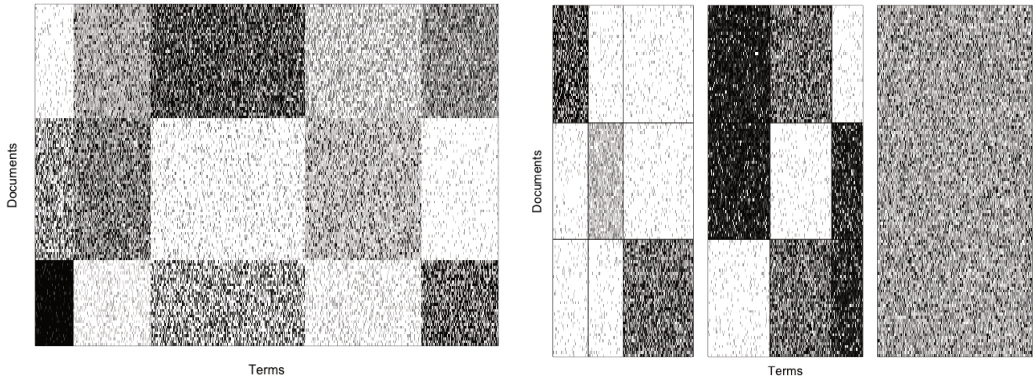

Figure 4.1 – On the left, the usual Poisson Latent Block Model: we see that some blocks are not easily classifiable into noisy or significant blocks. On the right, the SOCC approach: we can easily distinguish betwenn the noisy blocks (shown in a lighter shade) and the significant blocks.

The parameter δ_{gh} is referred to as 'the effect of block (g, h)' Govaert and Nadif (2010). The probability density function is therefore given by:

$$f(x_{ij}; \delta_{gh}) = \mathcal{P}(n_{i.} n_{.j} \delta_{gh}) = \frac{1}{x_{ij}!} e^{-n_{i.} n_{.j} \delta_{gh}} (n_{i.} n_{.j} \delta_{gh})^{x_{ij}}. \tag{4.1}$$

4.2.2 Inference

We recall here the steps of the SEM-Gibbs algorithm for the PLBM.

(A) SAMPLING ROW PARTITIONS Generate the row partitions with:

$$p(v_{ig}^{(q)} = 1 | \boldsymbol{x}, \boldsymbol{w}^{(q-1)}; \boldsymbol{\theta}^{(q-1)}) \propto \pi_g^{(q-1)} \times \prod_{jh} f(x_{ij}; \delta_{gh}^{(q-1)})^{w_{jh}^{(q-1)}}.$$

(B) FIRST M STEP Update:

$$\pi_g^{(q)} = \frac{1}{N} \sum_i v_{ig}^{(q)}$$

and

$$\delta_{gh}^{(q)} = \frac{1}{n_{g.} n_{.h}} \sum_{ij} v_{ig}^{(q)} w_{jh}^{(q-1)} x_{ij},$$

with $n_{g.} = \sum_{ij} v_{ig}^{(q)} x_{ij}$ and $n_{.h} = \sum_{ij} w_{jh}^{(q-1)} x_{ij}$.

(C) SAMPLING COLUMN PARTITIONS Generate the column partitions with:

$$p(w_{jh}^{(q)} = 1 | \boldsymbol{x}, \boldsymbol{v}^{(q)}; \boldsymbol{\theta}^{(q)}) \propto \rho_h^{(q-1)} \times \prod_{ig} f(x_{ij}; \delta_{gh}^{(q)})^{v_{ig}^{(q)}}.$$

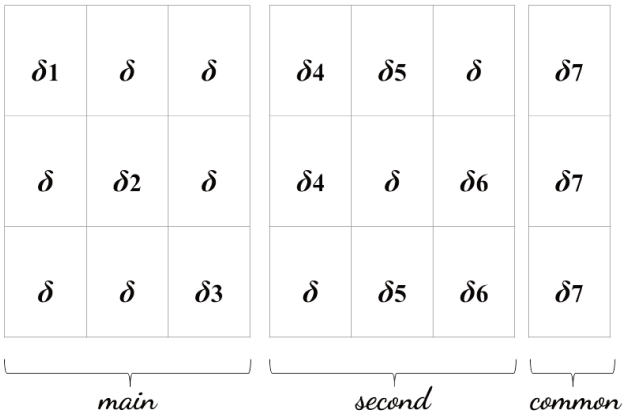

Figure 4.2 – Co-clustering structure of the Self-Organised Co-Clustering model, with block effect parameters, in the case $G = 3$.

(D) SECOND M STEP Update:

$$\rho_h{}^{(q)} = \frac{1}{J}\sum_j w_{jh}^{(q)}$$

and $\delta_{gh}^{(q)}$ as in Step (B).

4.3 Self-Organised Co-Clustering

4.3.1 An easy-to-read structure

In the latter section, all the δ_{gh} are unrelated, and consequently, each block should be interpreted separately from each other. In the model we propose, this independence does not hold true anymore: a structure is forced among the blocks so that the result is easier to read. Thus, for a given block (g, h), the corresponding block effect δ_{gh} will either be specific to column-cluster h with $\delta_{gh} = \delta_h$, or non-specific, with $\delta_{gh} = \delta$. In the case of non-specific block effect $\delta_{gh} = \delta$, the block (g, h) is considered as a noisy block. We refer to it as a 'non-meaningful' block, and it shares the same δ with all the other non-meaningful blocks. In the case of $\delta_{gh} = \delta_h$, the block (g, h) is 'meaningful', and shares the same δ_h with all the meaningful blocks of the same column-cluster h. In this case, the terms of the h^{th} column-cluster are considered to be specific to the documents of one or several row-clusters.

To organise these meaningful and non-meaningful blocks, several rules are given. First of all, after choosing the number of row-clusters G, the co-clustering necessarily has $H = G + \binom{G}{2} + 1$ column-clusters. Moreover, the column-clusters are divided into three sections called *main*, *second* and *common*. The purpose of these sections is explained here.

The *main* section concerns the first G column-clusters, for $h \in \{1, ..., G\}$. In each column-cluster h of this section, only one block is meaningful and parameterised by δ_h. All the other blocks are non-meaningful and parameterised by δ. Consequently, for each cluster of documents (row-cluster), the meaningful block indicates the terms that are specific to these documents. As a result, in the *main* section, the meaningful blocks are located on the diagonal, and the other ones are the non-meaningful ones.

The *second* section concerns the following $\binom{G}{2}$ column-clusters ($h \in \{G+1, ..., G+\binom{G}{2}\}$). In each column-cluster h of this section, two blocks are meaningful. Consequently, each column-cluster contains terms that are specific to two clusters of documents (row-clusters).

Finally, the *common* section consists of only one column-cluster and gathers the terms that are common to all documents.

This structure, as well as the corresponding block effect δ, are illustrated by Figure 4.2, in which we clearly see the meaningful blocks with $\delta_{gh} = \delta_h$ and non-meaningful blocks with $\delta_{gh} = \delta$. We also discern the organization among these blocks and the three different sections *main*, *second* and *common*. For instance, in the *main* section, the first column cluster is considered to be specific to first row-cluster, thus only the column cluster's first block has its own specific distribution with δ_1. On the other hand, the other blocks of this column-cluster are considered to be non-meaningful, and have a block effect parameter δ, which is common to all non-meaningful blocks. In the *second* section, we note, for example, that for $h = 4$ blocks $(1, 4)$ and $(2, 4)$ are meaningful, and share the same block effect δ_4. This means that terms from column-cluster 4 are specific to documents from row-clusters 1 and 2. Moreover, block $(4, 3)$ is non-meaningful and has the same effect δ as the other non-meaningful blocks. The *common* section is a bit particular insofar that it contains only one column-cluster, so $h = 7$. This column-cluster contains the terms that are specific to all groups of documents and its corresponding blocks all share the same δ_7.

4.3.2 The SOCC model and its inference

From Section 4.3.1, knowing the column-cluster h we can write: $g \in \mathcal{C}_h \cup \overline{\mathcal{C}}_h$, such that $\mathcal{C}_h$ are the meaningful blocks of column-cluster h and $\overline{\mathcal{C}}_h$ are the non-meaningful blocks of column-cluster h. In this case, the probability of the SOCC model is written as:

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{\substack{(\boldsymbol{v},\boldsymbol{w}) \\ \in V\times W}} \prod_{ig} \pi_g^{v_{ig}} \prod_{jh} \rho_h^{w_{jh}} \prod_{ijh} \prod_{g\in\mathcal{C}_h} f(x_{ij};\delta_h)^{v_{ig}w_{jh}} \prod_{g\in\overline{\mathcal{C}}_h} f(x_{ij};\delta)^{v_{ig}w_{jh}}. \tag{4.2}$$

The complete data log-likelihood is given by:

$$\begin{aligned} & l_c(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{v},\boldsymbol{w}) = \\ & \sum_{ig} v_{ig}\log\pi_g + \sum_{jh} w_{jh}\log\rho_h + \sum_{ijh}\Bigg(\\ & \sum_{g\in\mathcal{C}_h} v_{ig}w_{jh}\left[x_{ij}\log(n_{i.}n_{.j}\delta_h) - n_{i.}n_{.j}\delta_h - \log(x_{ij}!)\right] + \\ & \sum_{g\in\overline{\mathcal{C}}_h} v_{ig}w_{jh}\left[x_{ij}\log(n_{i.}n_{.j}\delta) - n_{i.}n_{.j}\delta - \log(x_{ij}!)\right]\Bigg). \end{aligned} \tag{4.3}$$

As in Section 4.2.1, the SEM-Gibbs algorithm is chosen to estimate the partitions $(\boldsymbol{v}, \boldsymbol{w})$ and parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\delta})$ with $\boldsymbol{\delta} = (\delta, \delta_1, \ldots, \delta_H)$. In contrast with the Poisson LBM, the Poisson distribution $f(x_{ij};\delta_{gh})$ of block (g, h) will depend on the meaningfulness of block (g, h). For all $h \in \mathcal{H}$ if $g \in \mathcal{C}_h$, then $f(x_{ij};\delta_{gh}) = f(x_{ij};\delta_h)$, and if $g \in \overline{\mathcal{C}}_h$, then $f(x_{ij};\delta_{gh}) = f(x_{ij};\delta)$, where f is the Poisson p.d.f. given by Equation (4.1).

The SEM-Gibbs algorithm proposed for the Self-Organised Co-Clustering model inference is summarised below. It iterates the partitions sampling and the update of the parameters

(steps 1 to 4) during a given number of iterations (nb.iter). The final parameter estimation, now denoted by $\hat{\boldsymbol{\theta}}$, is obtained by averaging the model parameters over the sample distribution (after a burn-in period). Last, the final partitions $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{w}}$ are estimated with $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$, using another Gibbs sampler.

(A) SAMPLING ROW PARTITIONS Generate the row partitions with:

$$p(v_{ig}^{(q)} = 1|\boldsymbol{x}, \boldsymbol{w}^{(q-1)}; \boldsymbol{\theta}^{(q-1)}) \propto \pi_g^{(q-1)} \times \prod_{jh} f(x_{ij}; \delta_{gh}^{(q-1)})^{w_{jh}^{(q-1)}}.$$

(B) FIRST M STEP Update the parameters:

$$\pi_g^{(q)} = \frac{1}{N} \sum_i v_{ig}^{(q)},$$

$$\delta^{(q)} = \frac{\sum\limits_{ijhg \in \overline{\mathcal{C}}_h} v_{ig}^{(q)} w_{jh}^{(q-1)} x_{ij}}{\sum\limits_{ijhg \in \overline{\mathcal{C}}_h} v_{ig} w_{jh}^{(q-1)} n_{i.} n_{.j}}$$

and

$$\delta_h^{(q)} = \frac{\sum\limits_{ijg \in \mathcal{C}_h} v_{ig}^{(q)} w_{jh}^{(q-1)} x_{ij}}{\sum\limits_{ijg \in \mathcal{C}_h} v_{ig}^{(q)} w_{jh}^{(q-1)} n_{i.} n_{.j}}.$$

(C) SAMPLING COLUMNS PARTITIONS Generate the columns partitions with:

$$p(w_{jh}^{(q)} = 1|\boldsymbol{x}, \boldsymbol{v}^{(q)}; \boldsymbol{\theta}^{(q)}) \propto \rho_h^{(q-1)} \times \prod_{ig} f(x_{ij}; \delta_{gh}^{(q)})^{v_{ig}^{(q)}}.$$

(D) SECOND M STEP Update the parameters:

$$\rho_h^{(q)} = \frac{1}{J} \sum_j w_{jh}^{(q)},$$

$\delta^{(q)}$ and $\delta_h^{(q)}$ as in Step (B).

CHOICE OF THE NUMBER OF ITERATIONS For the SEM-Gibbs algorithm, two numbers must be chosen: the total number of SEM-Gibbs iterations (*nb.iter*) and the number of iterations for the burn-in period. These numbers are graphically chosen by visualizing the values of the model's parameters along the SEM-Gibbs iterations. The parameters must reach their stationary state after the burn-in period, and the remaining number of iterations until the end must be sufficient to compute their respective means.

4.3.3 Model selection

The definition of a model selection criterion has two purposes. Firstly, in the context of unsupervised methods, choosing the number of row-clusters G is an issue. One of the great advantages of the SOCC model is that the number of column-clusters H is directly fixed by the number of row-clusters G. Indeed, as explained before, $H = G + \binom{G}{2} + 1$. However, the choice for the number of row-clusters G is still a problem. Second, as described in the algorithm, the SEM-Gibbs algorithm starts with a random initialization of partitions $(\boldsymbol{v}, \boldsymbol{w})$. However, this initialization has an impact on the convergence of the algorithm and on the resulting estimations. It is therefore recommended to execute the algorithm several times with different initializations and to have a criterion to choose the best solution.

As detailed in Section 2.3, we use the ICL-BIC approximation. For the SOCC model, is is given by:

$$\begin{aligned} \text{ICL-BIC}(G) = &\log p(\hat{\boldsymbol{\theta}}; \boldsymbol{x}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}) \\ &- \frac{1}{2}(G-1)\log N - \frac{1}{2}(H-1)\log J - \frac{1}{2}(H+1)\log(NJ). \end{aligned} \tag{4.4}$$

The number G of row-clusters maximizing this criterion must be retained.

4.4 Numerical Experiments

In this section, we assess the quality of the SOCC model. First of all, we chose seven clustering, co-clustering and topic-modelling methods to compare the results: we list them in Section 4.4.1 and refer to them as 'baselines'. In Section 4.4.2, we simulate data through the SOCC model's process generation. We run the baselines algorithms and compare their results with those of the SOCC model, in terms of partition estimation. We also evaluate the behaviour of the ICL criterion for choosing the number of row-clusters. In Section 4.4.3, we used real textual data sets whose documents are known to belong to some predefined classes and compared the row-clustering (or column-clustering) quality with the baseline methods. We conclude this section by illustrating with a use case how the SOCC model can be helpful for interpreting the co-clustering results.

4.4.1 Baselines

Seven clustering, co-clustering and topic-modelling methods were selected as baselines to compare our results. Two of them are based on the Latent Block Model. The Poisson Latent Block Model (PLBM,Govaert and Nadif (2010)), as detailed in Section 4.2, is a co-clustering algorithm that uses the direct application of the Latent Block Model. The Sparse Poisson Latent Block Model Ailem et al. (2017), referred to as 'SPLBM', is a constrained version of the Poisson Latent Block Model, which was also developed to co-cluster document-term matrices. This model, already described in the introduction, constrains its structure to the *main* structure of our model. Both models were implemented in `C++` from the pseudo-code of their respective papers. The Information Theory Co-Clustering method, referred to as 'ITCC' Dhillon et al. (2003), is a co-clustering technique that uses information theory and the mutual information to discover the blocks. We used the `C++` implementation provided by their authors. The Orthogonal Non-negative Matrix Tri-Factorization method, referred to as 'ONMTF' Ding et al. (2006), is a co-clustering algorithm based on matrix factorization. We implemented the pseudo-code provided in `R`. The Non-negative Matrix Factorization NMF

Table 4.1 – Simulated parameters $\delta_{gh} \times 10^{-7}$. For each cell x_{ij} the Poisson parameter is equal to $n_{i.}n_{.j}\delta_{gh}$, with row margins $n_{i.}$ equal to 2455 on average, and columns margins $n_{.j}$ equal to 249 on average.

Cluster	1	2	3	4	5	6	7
1	8.6	2.9	2.9	49.8	47.8	2.9	34.0
2	2.9	9.0	2.9	49.8	2.9	52.9	34.0
3	2.9	2.9	9.4	2.9	47.8	52.9	34.0

Paatero and Tapper (1994) is a clustering algorithm based on matrix factorization. The R Package `NMF` Gaujoux and Seoighe (2010) was used for the experiments. The Spherical Kmeans clustering method ('Skmeans') is the implementation of the kmeans algorithm, but with embedding of the Cosine similarity (and not the Euclidean distance). The R Package `skmeans` Hornik et al. (2012) was used for the experiments. Latent Dirichlet Allocation (LDA) Blei et al. (2003) is a generative statistical model for topic modelling. The R package `textmineR` implementation was used to use it on the data sets. To assess the quality of the row-clusters, all of these seven methods were used. To assess the quality of the column-clusters, we obviously only selected the four co-clustering methods.

4.4.2 Simulated data set

4.4.2.a Simulation setting

A data set with $N = 120$, $J = 1\,200$, $G = 3$ and $H = 7$ was simulated. The parameters were chosen arbitrarily: the row mixing proportions $\boldsymbol{\pi}$ are equal to $(.33, .33, .33)$ and the column mixing proportions $\boldsymbol{\rho}$ are equal to $(.08, .08, .17, .17, .17, .08, .25)$. The block effects are given in Table 4.1.

For the SOCC inference, the total number of iterations of the SEM-Gibbs algortihm was fixed to 50 with a burn-in period of size 35. In Figure 4.3, the evolution of parameters δ and δ_h for the *main* section is plotted. We see that the parameters stabilise in less than ten iterations. The numbers of fixed iterations are therefore enough to reach the stationary state.

4.4.2.b Results

The SOCC model was run on 100 simulations, and the Adjusted Rand Index, referred to as 'ARI' Hubert and Arabie (1985) between the true partitions and the estimated partitions were computed. The ARI for row-clusters was always equal to 1. Regarding the column-clusters, the mean ARI was equal to .99. It shows that the inference algorithm for SOCC functions appropriately. It is worth noting that 25% of runs failed to reach a valid solution, systematically leading to empty clusters solutions. Such behaviour is a well-known drawback of co-clustering procedures Brault (2014); Selosse et al. (2020b). Nevertheless, this relative frequency of failures is not too high and not detrimental for the use of the SOCC model. When we obtain a solution with some empty clusters, we just have to restart the algorithm with another random initialization.

Furthermore, we executed the competitors' algorithms on this data set: the ARI boxplots for all methods are available in Figure 4.4. We see that on this simple data set, most of the methods perform well in terms of row clustering. This is the reason why we challenge the

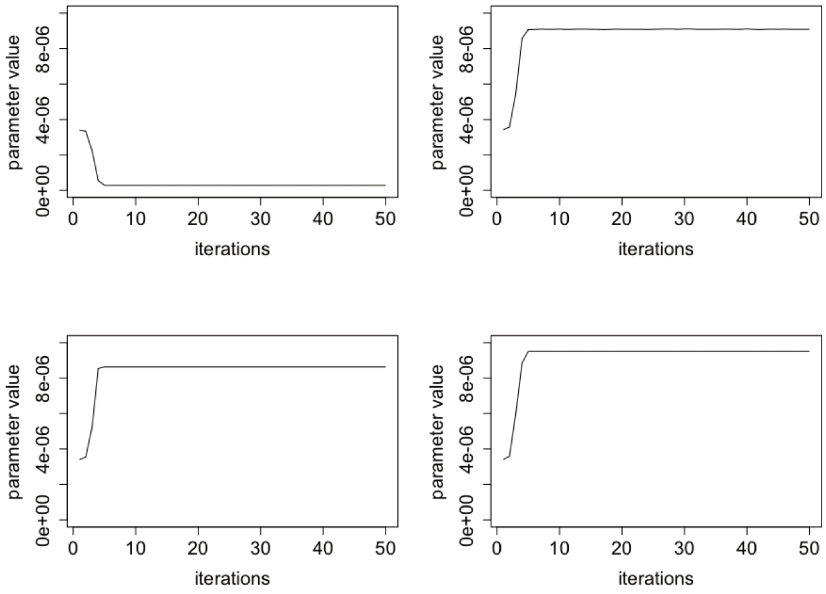

Figure 4.3 – From left to right, and from top to bottom: change in parameters δ, δ_1, δ_2, δ_3 when executing the algorithm on the simulated data set. The parameters reach their stationary state in less than 10 iterations.

Table 4.2 – Number of row and column-clusters (G, H) selected by ICL-BIC on the 100 simulated data sets, the right one being $(3, 7)$.

(G, H)	(2,6)	**(3,7)**	(4,11)	(5,16)
# chosen	25	**75**	0	0

methods using real and more difficult data sets in Section 4.4.3.

4.4.2.c Selection for the number of row clusters

For each of the 100 simulations, the co-clustering was run for $G = \{2, 3, 4, 5\}$ and the ICL-BIC criterion was computed. Table 4.2 presents the number of times each G was selected: the right number was selected in 75% of the cases. For the remaining 25% executions, $G = 2$ was selected.

4.4.3 Real data sets experiments

In this section, real labelled data sets are used to assess the quality of the proposed method. We describe the data sets that were used, the methods the SOCC was compared to and the results.

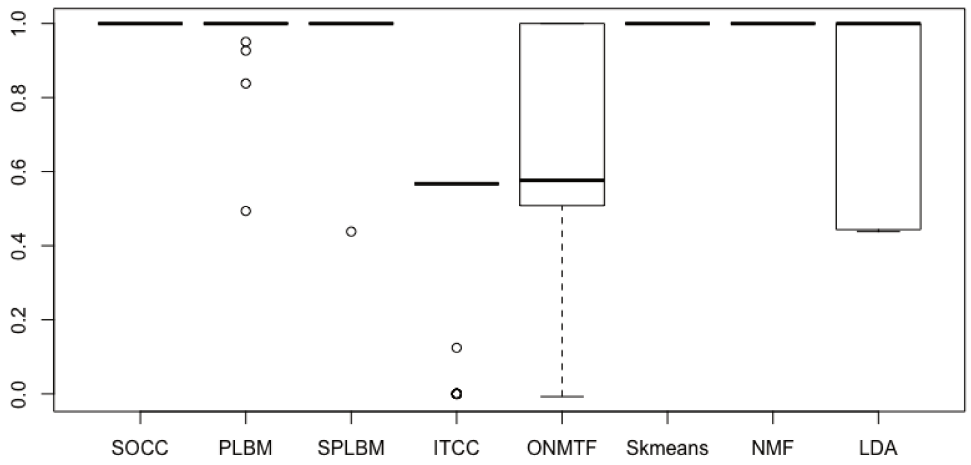

Figure 4.4 – ARI for SOCC model and competitors models on simulated data set.

4.4.3.a Real data sets

Eight data sets were retained for this Section. The **classic3** data set (dimensions $3,891 \times 5,236$) and the **classic4** data set* (dimensions $7,094 \times 5,896$) consist respectively of 3 different document collections (CISI, CRANFIELD, and MEDLINE) and 4 different document collections (CACM, CISI, CRANFIELD, and MEDLINE). **Pubmed5** ($12,648 \times 8,863$), **Pubmed4** ($11,131 \times 8,257$) and **Pubmed3** ($9,582 \times 7,454$) were built from the collection Pubmed10 Chen et al. (2009), with approximately $15,500$ medical abstracts from the Medline database, partitioned across 10 different diseases and published between 2000 and 2008. Pubmed3 contains the three largest classes, while Pubmed4 (and Pubmed5) contains the four (and five) largest classes. The classes, ranked from the largest to the smallest, include documents about otitis, migraine, age-related macular degeneration, kidney calculi and hay fever. **Pubmed4min** ($2,121 \times 3,660$) was also extracted from the Pubmed10 data set. However, only the four smallest classes were extracted. The documents are about jaundice, Raynaud disease, chickenpox and gout. The **sports** ($8,580 \times 14,870$) and **yahoo** ($2,340 \times 10,431$) data sets were obtained from the CLUTO toolkit Karypis (2002). The yahoo data set contains 6 different document categories with each document corresponding to a web page listed in the subject hierarchy of *Yahoo!*. The sports data set contains documents regarding 7 different sports including baseball, basketball, bicycling, boxing, football, golfing and hockey.

DISCUSSION ON THE NUMBER OF CLUSTERS The baselines data sets never have more than 7 known clusters in line, when other methods such as Ailem et al. (2017) execute their algorithm on data sets with up to 50 row-clusters. A limitation of the SOCC model is its difficulty in using it when the number of classes G is greater than 10. When $G = 10$,

*http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/

we have $G = G + \binom{G}{3} + 1 = 56$. With 56 column-clusters, the resulting co-clustering loses its interpretability, which is supposed to be a strength of the model. Therefore, it is recommended not to use the model when G is superior to 7.

4.4.3.b Assessing the quality of row-clusters

To assess the document clustering quality, the ARI between the known partitions and those estimated were computed. For each data set, each method was executed 30 times. Figure 4.5 plots the ARIs boxplots for all data sets and methods. We can see on these boxplots that the SOCC approach obtains the highest median ARIs for the classic3, pubmed4min and sports data sets. On the classic3 data set, the SOCC model obtains a median ARI of 0.96, and so does the NMF method. The model with the second highest median ARI (0.95) is the SPLBM model. On the pubmed4min data set, the median ARI for the SOCC model is equal to 0.55. The PLBM method yields the second highest ARI value with 0.46. Finally, on the sports data set, the SOCC obtains the highest median ARI value (0.44), and the NMF methods ranks second with an ARI value equal to 0.43.

On the other data sets, the SOCC model obtains satisfactory results and ranks as the second-best method in terms of ARI after Skmeans. This latter clustering method yields better results on data sets pubmed3, pubmed4, and pubmed5 but it presents one of the worst performances for classic4, pubmed4min and sports. Therefore, even if it obtains good results on some data sets, its inconsistency on the other data sets makes it an unreliable method. For this reason, SOCC seems to be the best method from a document clustering standpoint. The reason for this success is probably due to the model's parsimony.

4.4.3.c Assessing the quality of column-clusters

In most studies, the evaluation of co-clustering algorithms is only based on resulting row-clusters. This is due to the lack of public data sets providing the true partitions for both observations and features. In document clustering, for example, popular benchmarks provide the true document labels, while the term clusters remain unknown. To overcome this problem and improve over currently used evaluation methods, we propose the following strategy. For a given column-cluster, the ten most frequents terms are extracted. We compute the average Jaccard similarity between these terms on all the documents: this value is considered as a proximity measure between terms of the column-cluster. We average this proximity measure over all the column-clusters. In terms of interpretation, this criterion based on Jaccard similarities is used to assess how a co-clustering gathers terms that often occur in the same document. We report the scores obtained by the methods on the data sets in Table 4.3. From these results, it can be seen that for the classic4, pubmed3, pubmed4, pubmed4min, pubmed5, sports and yahoo data sets, all algorithms perform equally well but the SOCC model has the highest averaged score. Regarding the classic3 data set, ONMTF yields a better result (.89), but is closely followed by the SOCC model (.88).

4.4.3.d pubmed4min use case

In this section, we demonstrate using the Pubmed4min data set that the SOCC results are easy-to-interpret. Regarding the *main* section, when we seek the 10 most frequent terms of the first column-cluster, we get 'varicella', 'vaccin', 'ag', 'children', 'year', 'immun', 'zoster', 'hospit', 'chickenpox' and 'adult'. These terms are closely related to chickenpox (or varicella), so we can easily guess that the first row-cluster's documents are about chickenpox. When we

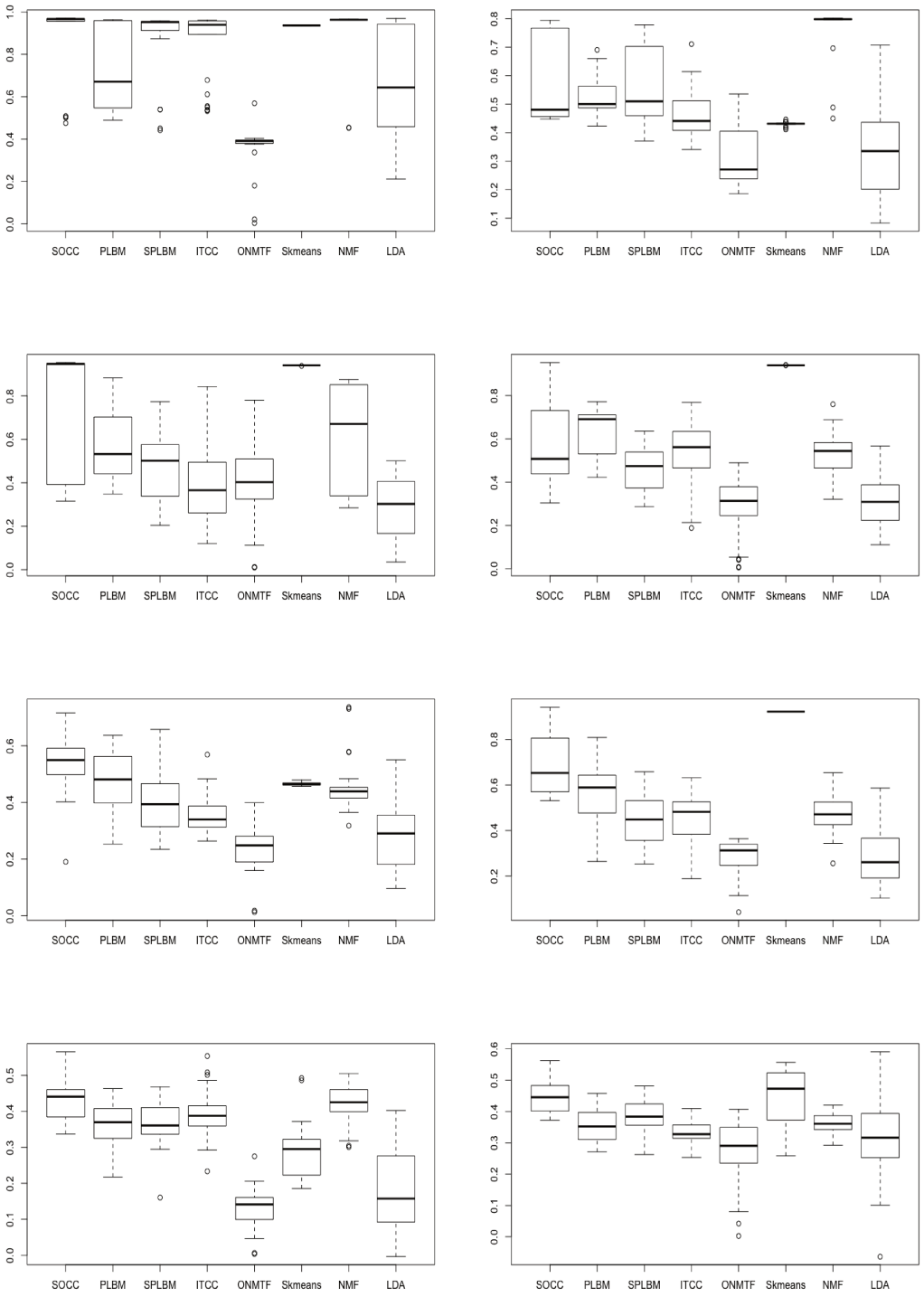

Figure 4.5 – ARIs for document clustering. From left to right and top to bottom: classic3, classic4, pubmed3, pubmed4, pubmed4min, pubmed5, sports, yahoo.

Table 4.3 – Average similarity measurements between the top 10 terms of each column-cluster.

Data set	SOCC	PLBM	SPLBM	ITCC	ONTMF
Classic3	.88 (.07)	.86 (.08)	.86 (.08)	.86 (.08)	**.89** (.07)
Classic4	**.91** (.06)	.88 (.07)	.88 (.07)	.87 (.07)	.87 (.07)
Pubmed3	**.85** (.13)	.77 (.13)	.79 (.12)	.76 (.13)	.80 (.08)
Pubmed4	**.88** (.12)	.80 (.15)	.80 (.13)	.80 (.14)	.81 (.09)
Pubmed4min	**.87** (.11)	.79 (.13)	.81 (.09)	.80 (.13)	.84 (.08)
Pubmed5	**.90** (.12)	.78 (.13)	.81 (.13)	.83 (.13)	.85 (.08)
Sports	**.88** (.11)	.79 (.11)	.79 (.11)	.77 (.11)	.78 (.10)
YahooKB1	**.85** (.20)	.67 (.31)	.70 (.33)	.69 (.31)	.69 (.31)

seek the 10 most frequent terms of the second column-cluster, we get 'jaundic', 'obstruct', 'liver', 'bile', 'biliari', 'hepat', 'duct', 'rat', 'stent' and 'bilirubin'. Again, we can easily assert that the second row-cluster's documents are about jaundice. Regarding the *second* section, if we look at column-cluster 5, which corresponds to the terms specific to row-clusters 1 and 2, we get: 'rate', 'complic', 'neg', 'mortal', 'morbid', 'infant', 'neonat', 'bacteri', 'safe', 'inva'. These terms are mostly related to children, which seems consistent since jaundice and chickenpox are very common in toddlers and newborns. Furthermore, jaundice can occur as a complication of chickenpox, justifying the presence of 'complic' in the list.

4.5 Harry Potter use case

In this section, we use the SOCC model on the **Harry Potter** data set. For each stage of performing a co-clustering, we show the difficulties encountered by the classical co-clustering methods and how the SOCC model overcomes them. The Harry Potter data set contains the first three volumes of the famous series (Rowling (1997, 1998, 1999)), entitled 'Harry Potter and the Philosopher's Stone', 'Harry Potter and the Chamber of Secrets' and 'Harry Potter and the Prisoner of Azkaban'. In the resulting Document-Term matrix, each line represents a chapter, and each column represents a term.

4.5.1 Co-clustering set up

DATA SET PRE-PROCESSING The original text was changed. Firstly, the punctuation and numbers were removed. Secondly, the terms that appeared only once were removed because they do not often add useful information. The whole was then transformed to a classic Document-Term frequency matrix. The resulting matrix is of dimensions $N = 57$ and $J = 6,884$.

Table 4.4 – Maximum ICL values for each G tested.

number of row clusters G	2	3	4	5	6	7	8
max ICL value	-231774.9	-228133.4	-226650.7	-225895.4	-226709.2	**-225072.6**	-226035.7

SETTING THE NUMBER OF ITERATIONS When dealing with a new data set, the user must choose the total number of iterations and the number of burn-in iterations. For this, they must execute the SEM-Gibbs algorithm with the different numbers of clusters they want to test (see paragraph 'Finding the right numbers of clusters' below) with an arbitrary number of iterations. Then, they must check that the parameters reached their stationary state before the number of burn-in iterations. For the Harry Potter data set, and with $G = 7$, we see in Figure 4.6 that the parameters reached their stationary state before the 75th iteration. The total number of iterations can then be fixed to 100 and the number of burn-in iterations to 75.

FINDING THE RIGHT NUMBER OF CLUSTERS For the baselines PLBM, ONMTF and ITCC, the user has to define two numbers of clusters G and H at this stage. Furthermore, the ONMTF and ITCC methods have no criteria to define these numbers. The SOCC model induces H from G so the user only has to choose G. Furthermore, the ICL criterion defines the best number of clusters once the algorithm is run on the different possibilities. On the Harry Potter data set, we ran the SEM-Gibbs algorithm for $G = \{2, 3, 4, 5, 6, 7, 8\}$, and got the corresponding ICL values. The largest ICL value was obtained with $G = 7$. Table 4.4 presents the maximum ICL values for each number of row-clusters tested. Figure 4.7 plots the Document-Term matrix sorted by row-clusters and column-clusters.

4.5.2 Interpretation of the results

At this stage, the user has a co-clustered Document-Term matrix. Using the methods ONMTF, ITCC and PLBM, they are able to obtain the chapters of the books that are gathered into the same group. However, they cannot easily know the main topic of each group. For example, for the PLBM method, they should find the highest block effect and observe the corresponding row-cluster and column-cluster to obtain the relevant chapters and terms. With the SOCC model, the user can directly know which blocks are of interest. In this section, we studied the terms belonging to column-clusters and found the main underlying topic. We do not list every term but chose the ones that are most related to the topic concerned. Here, we develop an interpretation of the column-clusters.

INTERPRETATION OF COLUMN-CLUSTERS FOR THE *main* SECTION Seven clusters in line were detected by the SOCC model. Therefore, there are also seven column-clusters in the main section. The first contains the terms specific to the chapters of the first row-cluster, the second contains the terms specific to the chapters of the second row-cluster, and so on. We highlight below that this specific co-clustering structure is easily readable for to users.

- Cluster 1: Some terms specific to the chapters of this row-cluster are 'agony', 'hewho-mustnotbenamed', 'pain', 'quirrell' and 'serpent'. These terms refer to Harry Potter's enemy, called Lord Voldemort. People are so afraid of him that they never say his name aloud and refer to him as 'he-who-must-not-be-named'. He loves serpents and torturing his opponents. Quirrell is his servant in Volume 1. We propose for this cluster the label 'Voldemort' for this cluster.

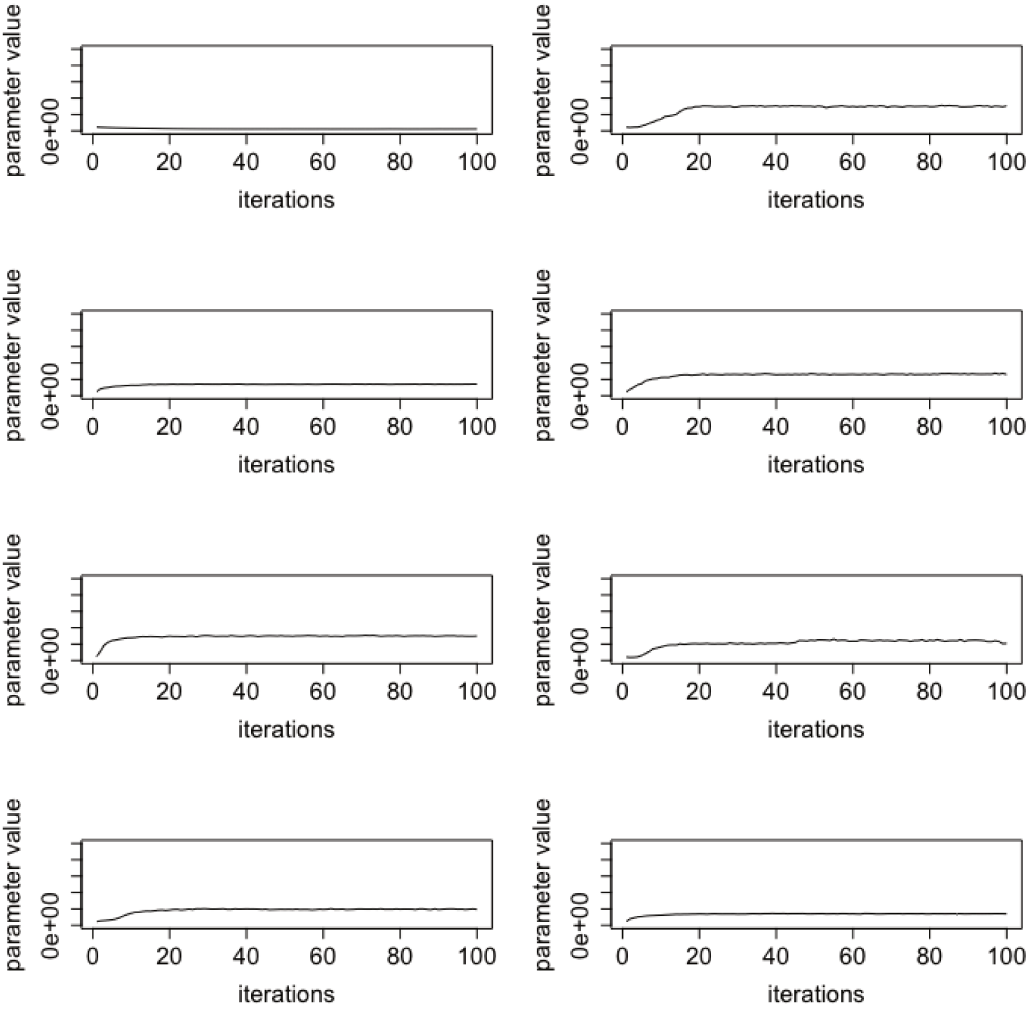

Figure 4.6 – Changes in parameters for the Harry Potter data set for δ, δ_1, δ_2, δ_3, δ_4, δ_5, δ_6, δ_7.

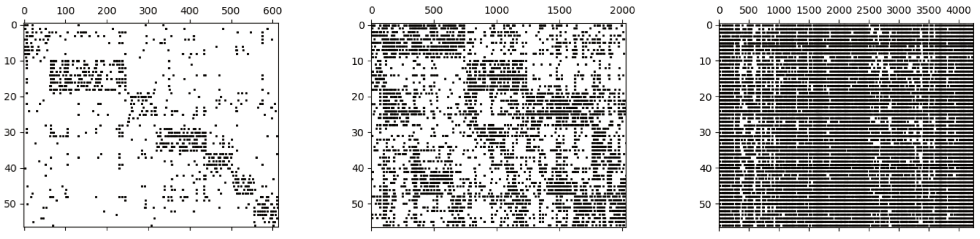

Figure 4.7 – Co-clustering of the Harry Potter data set with the SOCC method. From left to right: the $main$, the $second$ and the $common$ sections. The graphic was produced using the `Python` function $spy()$ with argument $markersize$ set to 1.2.

- Cluster 2: Some terms specific to the chapters of this row-cluster are 'animagus', 'black', 'dementors', 'godfather','james','lupin', 'murderer', 'peter', 'pettigrew', 'remus', 'scabbers','sirius', 'transform' and 'werewolf'. These terms relate to friendships of Harry's father. James Potter, Sirius Black, Remus Lupin and Peter Pettigrew were friends in Hogwart. Remus was a werewolf so his friends learnt how to transform into animals to be able to handle his strength when he turned into a a werewolf. Wizards with this capacity are called animagus. Finally, Pettigrew betrayed their friends and delivered James to Voldemort. Proposed label: Animagus.

- Cluster 3: Specific related terms here are 'alicia', 'angelina', 'beater', 'broom', 'captain', 'championship', 'chaser', 'cheers', 'commentary', 'game', 'goalposts', 'johnson', 'jordan', 'katie', 'lee', 'locker', 'match', 'quaffle', 'refereeing', 'scores', 'spinnet', 'teams' and 'win'. These terms relate to Quidditch, a sport where wizard must score points while flying on magic brooms. Alicia Spinnet, Angelina Johnson and Katie Bell are players on Harry's team. Lee Jordan is the match commentator of the school. Proposed label: Quidditch.

- Cluster 4: Here, specific related terms are 'birthday', 'cousin', 'drive', 'dudley', 'dursley', 'figg', 'moustache', 'petunia', 'privet', 'relative', 'television', 'uncle', 'vernon'. These terms refer to Harry's family. When his parents died, his aunt and uncle (Petunia and Vernon Dursley) adopted him. They have a child named Dudley, and the family lives in the Privet Drive street. Proposed label: the Dursleys.

- Cluster 5: Some terms specific to the chapters of this row-cluster are 'arthur', 'booklist', 'bookshop', 'burrow', 'molly', 'mum', 'supplies', 'shop' and 'weasley'. These terms relate to the Weasleys. They are members of the family of Ron Weasley, Harry's best friend. They live in a house called the Burrow. Arthur and Molly Weasley are Ron's parents. Every summer, Harry spends a part of summer with them, and they go to shop for the supplies for the following year. Proposed label: the Weasleys.

- Cluster 6: Some terms specific to the chapters of this row-cluster are 'bulstrode', 'crabbes', 'dueling', 'finchfletchey', 'goyles', 'greenhouse', 'justin', 'longbottoms', 'mandrakes', 'millicent' and 'sprout'. These terms are related to Harry's courses, and in particular his classmates. Crabbes, Goyles, Justin Finch-Fletchey, Milicent Bulstrode and Longbottom are all Harry's classmates. Ms. Sprout is the botany teacher, and the mandrakes are a special kind of magical plants. Proposed label: classmates.

- Cluster 7: Some terms specific to the chapters of this row-cluster are 'aragog', 'bane', 'centaurs', 'dragon', 'firenze', 'fluffy', 'forest', 'giant', 'goblins', 'hagrid', 'norbert', 'spider' and 'unicorn'. These terms refer to magical creatures that live in Harry' world. His friend Hagrid (a half giant wizard) has a passion about them. He owns a three-headed dog called Fluffy. In his childhood, he also raised Aragog, a giant spider. Firenze and Bane are centaurs living in the forest near Harry's school. Proposed label: magic creatures.

Therefore, the *main* section highlights seven main clusters of chapters that are related to: Voldemort, animagus, Quidditch, the Dursleys, the Weasleys, classmates and magical creatures.

A NOTE ON THE *main* SECTION COMPARED TO THE SPLBM MODEL Until now, most of the other co-clustering techniques have shown weaknesses in the overall process: ONMTF and ITCC

do not have a criterion to choose the number of blocks. For PLBM, the two numbers G and H have to be chosen and interpretation is difficult once the co-clustering is performed. The SPLBM model does not have these problems. In fact, the SPLBM is similar to the *main* section in the sense that it considers the meaningful blocks as being on the diagonal of the matrix. However, the *main* section is more selective and interpretable. Indeed, when running the SPLBM on the Harry Potter data set with $G = 7$, there will be 983 terms per column-clusters on average. It is therefore difficult to read them all and grasp what each row-cluster is about. In our case, the *second* and *common* sections get a large majority of the terms. In the same example, on the Harry Potter data set, the *main* section has 78 terms on average. Therefore, it is easier to read them quickly and get the topic of each row-cluster, as we just demonstrated above.

INTERPRETATION OF COLUMN-CLUSTERS FOR THE *second* SECTION With regard to the *second* section, as mentioned before, its corresponding column-clusters have terms that are related to two row-clusters. Since we now know what each row-cluster is about individually, from the *main* section, we can see the terms that link them. The SOCC model looks for common words for every row-cluster pair. This can be a limitation: for example, the chapters related to the Dursleys and the chapters related to Quidditch do not have a lot in common and the column-cluster related to these two groups of chapters contains only the word 'card', which is unrelated to both. However, most of the column-clusters that relates to two clusters of chapters are of interest to users. Here are some examples:

- Row clusters 1 and 4, which are about Voldemort and the Dursleys, share meaningful blocks in column-cluster 10. The corresponding terms include 'mother', 'nephew', 'petunias' and 'scar'. Petunia Dursley is Harry's aunt. She is connected to Voldemort because he killed her sister. He also attempted to kill Harry as a young boy, but he survived, and he was left with a scar on his forehead. Petunia then adopted her nephew.

- Row-clusters 1 and 5, which are about Voldemort and the Weasleys share meaningful blocks in column-cluster 11. This column-cluster has terms such as 'basilisks', 'tom', 'riddle' and 'ginny'. This makes sense because Ginny is Mr. and Ms. Weasley's daughter. She is closely connected to Voldemort in Volume 2. The wizard finds a way to bring Tom Riddle to life. Tom is the past version of himself, when he was a normal teenager in the school. Tom casts a spell on Ginny so that she wakes the giant basilisk serpent up in the Chamber of Secrets. Then, this snake attacks the school's students.

- Row-clusters 1 and 6, which are about Voldemort and Harry's classmates share meaningful blocks in column-cluster 12. The correspond terms include 'ernie', 'petrified' and 'serpents'. In Volume 2, Ernie is Harry's classmate. In duelling class, Harry speaks to a serpent, an ability both he and Voldemort hold. Ernie thinks that he is ordering the snake to attack Justin Finch-Fletchey. His suspicions grow when Justin is found petrified in the corridor. He spreads the rumour that Harry's destiny was to become a powerful dark wizard and that is why Voldemort wanted to kill him.

- Row-clusters 3 and 5, which are about quidditch and the Weasleys share meaningful blocks in column-cluster 20. It contains only three words, for which the two most frequent are 'fred' and 'george'. Fred and George are twins and they are also members of the Weasley family. Both of them are 'beaters' on Harry's Quidditch team.

- Row-clusters 3 and 6, which are about Quidditch and the Harry's classmates share meaningful blocks in column-cluster 21. The column-cluster contains the terms 'crabbe', 'goyle', 'malefoy' and 'slytherins'. Crabbe, Goyle and Malefoy belong to the Slytherin house at the school. They are Harry's classmates and hate him. In Volume 3, Harry and his classmates discover that he faints in the presence of dementors (a creature that can absorb your soul). Later on in the year, Harry fainted while playing in a Quidditch match, when Crabbe, Goyle and Malefoy arrived on the field disguised as dementors.
- Row-clusters 4 and 5, which are about the Dursleys and the Weasleys share meaningful blocks in column-cluster 23. The corresponding terms include 'auntie', 'bedroom', 'brothers', 'errol', 'ink', 'letters', 'september', 'sons', 'summer' and 'written'. The vocabulary related to a family context connects the two row-clusters because both of them relate to families. The terms 'summer' and 'september' relate to the fact that Harry spends part of his summer vacations at his aunt's place and the other part at the Weasley's. The terms 'errol', 'ink' and 'letters' refers to Errol, Ron Weasley's owl, which he uses to write to Harry when he is at his aunt's.

A NOTE ON THE *common* SECTION The *common* section is composed by a unique column-cluster. However, this cluster contains the majority of the terms, with $\rho_{29} = 0.63$ (thus, 63% of terms). The corresponding terms include 'harry', 'potter', 'ron', 'hermiones', 'granger' and 'hogwarts'. These terms are very important for the Harry Potter story, and at first, it seems odd that they are not in the *main* section. However, this phenomenon is explained by considering that the *common* section includes the terms that are frequent to all row-clusters. Furthermore, if the term 'harry' appeared in a column-cluster of the *main* section, it would not bring any valuable information about the chapters of this row-cluster, since Harry is present in all chapters.

4.5.3 Conclusions on the study of the Harry Potter data set

This section brought an insight on how to use the SOCC model on a completely unsupervised data set. Furthermore, for each stage of the process of co-clustering, we indicated how the tasks left to the user are easier with the SOCC model in comparison with the other co-clustering methods.

4.6 Conclusion and perspectives

In this work, we proposed the SOCC model, a novel approach to easily co-cluster textual data sets. The model offers easy-to-read results, and quickly shows the terms that are specific to one group of documents, the terms that are specific to two groups of documents, the terms that are common to all documents. The resulting algorithm is not only more accurate than other state-of-the-art methods but it is also able to detect the number of co-clusters, as a result of the ICL-BIC criterion. An R package `SOCC` is available upon request to perform these functionalities.

In future work, we could define other structures, for example with clusters of terms specific to 3 or more groups of documents. The first concern here is the increasing number of column-clusters (which would require at least $\binom{G}{3}$ more column-clusters). Also, it would be interesting to investigate a more developed model selection: we can allow the structure to not have all $G + \binom{G}{2} + 1$ column clusters. For example, in Figure 4.8, we see the pubmed4min

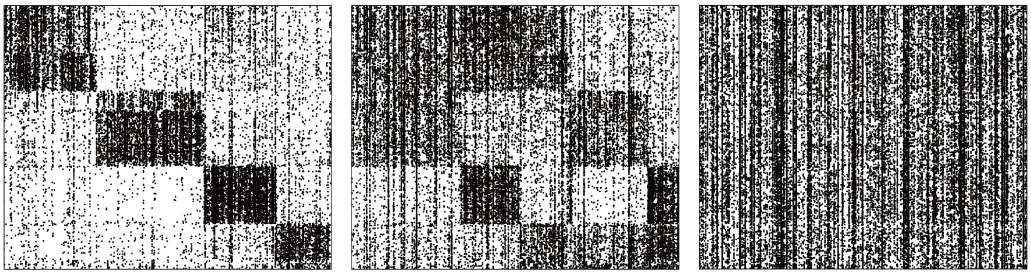

Figure 4.8 – Co-clustering of pubmed4min data set with the SOCC method. From left to right: the *main*, the *second* and the *common* sections. The graphic was produced using the `Python` function *spy*() with the argument *markersize* set to 1.3.

SOCC co-clustering with $G = 4$. We know that the *second* part comprises $\binom{4}{2} = 6$ column-clusters. We can easily notice five of them, but the sixth one is very small: is this column-cluster necessary? We could use the ICL criterion to dispose of the irrelevant column-clusters. However, loosening the strict structure assumption would result in other issues arising: testing all solutions could significantly increase the overall execution time.

5

Investigations on the Deep Gaussian Mixture Model

5.1 Introduction

This chapter is a work in collaboration with Claire Gormley from the University College Dublin, it was published in Selosse et al. (2020a). It consists in a thorough investigation of

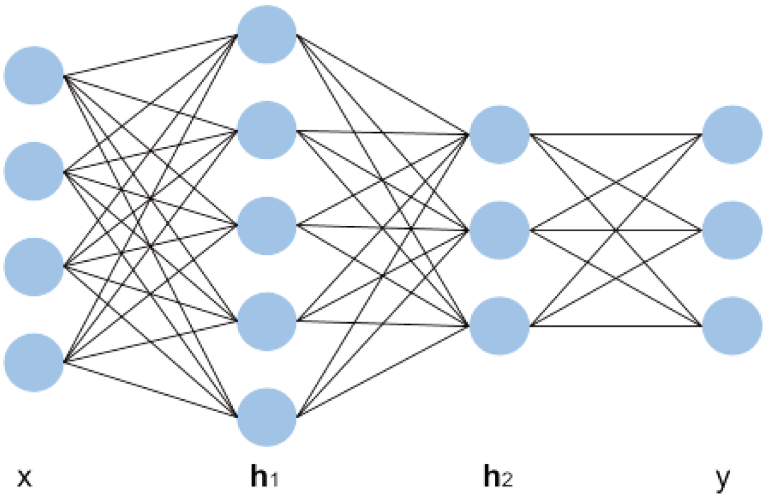

Figure 5.1 – Visualization of a neural network with two layers.

the Deep Gaussian Mixture Model (Viroli and McLachlan, 2019). This model is recent and it combines deep learning and model-based clustering. First, we describe the deep learning (or neural networks) framework and we explain how its particular multilayered architecture can be applied to the Mixture of Factor Analysers. Second, we describe the mathematical aspects and detail the inference algorithm of the model. Then, we highlight the limitations of the model on simulated data sets and try to explain why those limitations exist and what the perspectives to address the different issues are. All experiments can be found in the package `deepMFA`.

5.1.1 Neural networks

Deep learning is a broad framework that organises elements of inference in a hierarchical multilayered architecture. A layer is composed of elements called "neurons" that make simple operations (or activations) on the input they are fed with. Then, the results of these operations are passed to the neurons of the following layer. In other words, the input of a layer is the output of the previous one. An example is shown in Figure 5.1, where we see the input data $\boldsymbol{x}$ with $J = 4$, a first layer with five neurons and output $\boldsymbol{h}_1$, a second layer with three neurons with output $\boldsymbol{h}_2$ and the last output $\boldsymbol{y}$ with three dimensions. The vocabulary around deep learning is often associated to neurobiology since this architecture is directly inspired by the neural network of the human brain. Since each layer can learn upon the features extracted in the previous layer, the composition of subsequent multiple layers makes the whole system able to describe complex relationships between variables. Usually, the deep learning framework – and more specifically the deep neural networks – is used in a supervised context.

To compute the outputs $\boldsymbol{h}_l$ of a layer l, the outputs of the previous layer $\boldsymbol{h}_{l-1}$ are transformed with a linear transformation $\boldsymbol{\Lambda}_l$ (and bias $\boldsymbol{\eta}_l$) and subsequently transformed with a non-linear function σ_l, also referred to as the "activation function":

$$\boldsymbol{h}_l = \sigma_l(\boldsymbol{\Lambda}_l \boldsymbol{h}_{l-1} + \boldsymbol{\eta}_l),$$

with $\boldsymbol{h}_l$ (or $\boldsymbol{h}_{l-1}$) a vector of size $R^{(l)}$ (or $R^{(l-1)}$), $\boldsymbol{\Lambda}_l$ a matrix of size $R^{(l)} \times R^{(l-1)}$ and $\boldsymbol{\eta}_l$ a vector of size $R^{(l)}$. The outputs of a L-layers network is a linear combination of the activations in the last layer:

$$\boldsymbol{y} = \boldsymbol{\Lambda}_y \boldsymbol{h}_L + \boldsymbol{\eta}_y,$$

where the size J_y of the vector $\boldsymbol{y}$ depends on the task of prediction. The parameter $\boldsymbol{\Lambda}_y$ is a matrix of size $J_y \times R^{(L)}$ and $\boldsymbol{\eta}_y$ is of size J_y. Sometimes, $\boldsymbol{y}$ is also transformed with a certain non-linear function. When the neural network must predict the class of an observation, among a finite number of classes, then the "softmax" function will be used so that the last output corresponds to probabilities for an input to belong to each class. Therefore, the parameters to estimate are the matrices $(\boldsymbol{\Lambda}_l)_{l\in\{1,\dots,L\}}$ and the bias $(\boldsymbol{\eta})_{l\in\{1,\dots,L\}}$. The loss function will also depend on the task and on the nature of $\boldsymbol{y}$. The most common loss functions are the mean squared error loss (for a continuous target), the binary cross-entropy (for a binary target) and the multi-class cross-entropy (for a categorical target). Neural networks are usually optimised using gradient-based methods such as the gradient descent algorithm. At the qth iteration, the algorithm computes the gradient of the loss function with respect to the parameters $\boldsymbol{\theta} = (\boldsymbol{\Lambda}_l, \boldsymbol{\eta}_l)_{l\in\{1,\dots,L\}}$ and updates the parameters with a small step in the opposite direction of the gradient:

$$\boldsymbol{\theta}^{(q)} = \boldsymbol{\theta}^{(q-1)} - \alpha \nabla \boldsymbol{\theta},$$

where α is the step size, also referred to as "learning rate" and $\nabla\boldsymbol{\theta}$ is the gradient of the loss function with respect to the parameters. Training a neural network is the iterative process of updating the parameters many times. The most general deep learning architecture is referred to as the "Multi-Layer Perceptron" (MLP) which is made of "fully-connected" layers. This means that all the neurons of a layer are connected to all the neurons of the previous layer.

These concepts are the basis of the neural networks but many alternatives exist. Neural networks have been intensively studied over the last years and they reach the state-of-the-art performances in supervised and unsupervised learning in many domains. Some well-known variations of the neural networks are listed below:

- Activation functions: a lot of activation functions exist, some of them are the sigmoid function, the tanh function, the ReLU function and the Leaky-ReLU function.
- Auto-encoders: this particular family of neural networks learns to copy its input to its output. It has an internal layer that describes a code used to represent the input, and it is composed of two main parts: an encoder that maps the input into the code, and a decoder that maps the code to a reconstruction of the original input. A recent version of the auto-encoder is the Variational Auto-Encoder (Rezende et al., 2014; Kingma and Welling, 2013), whose internal layer representation is not fully-connected but are samples from a standard Gaussian multivariate distribution.
- Dropout: it consists in randomly setting activations to zero in the network during the training phase. This is a technique to avoid overfitting the data set (Srivastava et al., 2014).

- Convolutional Neural Networks (Lecun et al., 1998) (CNN): these neural networks were also inspired by biological processes and they try to mimic the visual cortex. First of all, the layers have neurons arranged in 3 dimensions. In addition, the layers are not fully connected. Each neuron inside a convolutional layer is connected to only a small region of the previous layer. Therefore, the CNN exploits spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers.

- Residual Neural Networks (He et al., 2016): they are a special kind of neural networks that allows neurons outputs to jump over some layers. A motivation for skipping over layers is to avoid the problem of vanishing gradients, which occurs when the gradient value becomes too small. It is known that the brain has structures similar to residual nets in the cerebral cortex.

- Generative Adversarial Nets (Goodfellow et al., 2014): this framework has two networks. The "generative network" is trained to generate samples and the "discriminative network" evaluates their probability to be real (i.e. their probability to not be made by the generative network). These two networks are adversarial because the objective of the generative network is to increase the error rate of the discriminative network (i.e. by producing novel candidates that the discriminative network thinks they are part of the true data distribution).

5.1.2 Coupling Deep Learning and Gaussian Mixture Models

Neural networks have shown great efficiency in supervised and unsupervised tasks. However, other methods such as model-based clustering should not be put aside since they still can be of help in different perspectives. In fact, recent studies show that the Mixture of Factor Analysers can be effectively estimated from image data and it is able to describe a higher spectrum of data density than Generative Adversarial Nets (Richardson and Weiss, 2018; Śmieja et al., 2020). The Deep Gaussian Mixture Model aims at exploiting advantages of both multi-layer architectures and MFA models. The idea behind that model is to consider a layer as an MFA. First, the input of the first layer is the data $\boldsymbol{x}$ and each neuron corresponds to a component of the first mixture. The resulting latent scores $\boldsymbol{z}^{(1)}$ are seen as the output $\boldsymbol{h}_1$ of the first layer and they are the input of the second layer, whose neurons are also components of a mixture, and so on, until the last layer whose scores $\boldsymbol{z}_L$ are assumed to be sampled from a standard multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ (as in the classical MFA model).

One of the earliest papers to define a model with the idea of stacking MFA layers is that of Tang et al. (2012) where the author refers to the model as the "Deep Mixture of Factor Analyser". In this model, the layers are not fully-connected and the architecture is a tree: a neuron receives an input from only one neuron of the previous layer. In van den Oord and Schrauwen (2014), the authors define a DGMM model with fully-connected layers, but the matrices of weights $\boldsymbol{\Lambda}$ are assumed to be squared. In other words, there is no dimension reduction between each MFA layer, which leads to important unidentifiability issues. In the model proposed by Yang et al. (2017), instead of using the MFA model as a layer, the authors suggest using the MCFA model (Baek et al., 2010): the matrices $\boldsymbol{\Lambda}_g$ of a same layer are therefore assumed to be equal for all g. Finally, Viroli and McLachlan (2019) define the Deep Gaussian Mixture Model (DGMM) with dimension reduction at each layer and provide an EM-algorithm for the inference of the parameters. This chapter aims at investigating the properties of the DGMMs as they were described in Viroli and McLachlan (2019).

5.2 Deep Gaussian Mixture Models

5.2.1 Definition of the Deep Gaussian Mixture Model

This section relies on Viroli and McLachlan (2019). Let us assume that there are L layers and that the data $\boldsymbol{x} = (\boldsymbol{x}_i)_{i\in\{1,\dots,N\}}$ is generated as follows:

$$
\begin{aligned}
&\boldsymbol{x}_i = \boldsymbol{z}_i^{(0)} = \boldsymbol{\eta}_{g_1}^{(1)} + \boldsymbol{\Lambda}_{g_1}^{(1)}\boldsymbol{z}_i^{(1)} + \boldsymbol{u}_i^{(1)} \text{ with prob. } \pi_{g_1}^{(1)},\ g_1 \in \{1,\dots,G^{(1)}\}\\
&\boldsymbol{z}_i^{(1)} = \boldsymbol{\eta}_{g_2}^{(2)} + \boldsymbol{\Lambda}_{g_2}^{(2)}\boldsymbol{z}_i^{(2)} + \boldsymbol{u}_i^{(2)} \text{ with prob. } \pi_{g_2}^{(2)},\ g_2 \in \{1,\dots,G^{(2)}\}\\
&\dots\\
&\boldsymbol{z}_i^{(l)} = \boldsymbol{\eta}_{g_{l+1}}^{(l+1)} + \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)}\boldsymbol{z}_i^{(l+1)} + \boldsymbol{u}_i^{(l+1)} \text{ with prob. } \pi_{g_{l+1}}^{(l+1)},\ g_{l+1} \in \{1,\dots,G^{(l+1)}\}\\
&\dots\\
&\boldsymbol{z}_i^{(L-1)} = \boldsymbol{\eta}_{g_L}^{(L)} + \boldsymbol{\Lambda}_{g_L}^{(L)}\boldsymbol{z}_i^{(L)} + \boldsymbol{u}_i^{(L)} \text{ with prob. } \pi_{g_L}^{(L)},\ g_L \in \{1,\dots,G^{(L)}\},
\end{aligned}
\tag{5.1}
$$

where:

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

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

$$
f(\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l+1)};\boldsymbol{\theta}) = \sum_{g_{l+1}=1}^{G^{(l+1)}} \boldsymbol{\pi}_{g_{l+1}}\mathcal{N}(\boldsymbol{\eta}_{g_{l+1}}^{(l+1)} + \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)}\boldsymbol{z}_i^{(l+1)}, \boldsymbol{\psi}_{g_{l+1}}^{(l+1)}). \tag{5.2}
$$

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

$$
f(\boldsymbol{x}_i;\boldsymbol{\theta}) = \sum_{\boldsymbol{g}\in\mathcal{G}} \pi_{\boldsymbol{g}}\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{g}}, \boldsymbol{\Sigma}_{\boldsymbol{g}}), \tag{5.3}
$$

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

$$
\pi_{\boldsymbol{g}} = \prod_{l=1}^{L} \pi_{g_l}^{(l)}, \tag{5.4}
$$

$$\boldsymbol{\mu_g} = \boldsymbol{\eta}_{g_1}^{(1)} + \boldsymbol{\Lambda}_{g_1}^{(1)}\left(\boldsymbol{\eta}_{g_2}^{(2)} + \boldsymbol{\Lambda}_{g_2}^{(2)}\big(\ldots(\boldsymbol{\eta}_{g_{L-1}}^{(L-1)} + \boldsymbol{\Lambda}_{g_{L-1}}^{(L-1)}\boldsymbol{\eta}_{g_L}^{(L)})\big)\right) \text{ and} \tag{5.5}$$

$$\boldsymbol{\Sigma_g} = \boldsymbol{\Lambda}_{g_1}^{(1)}\left(\boldsymbol{\Lambda}_{g_2}^{(2)}\big(\ldots(\boldsymbol{\Lambda}_{g_L}^{(L)}\boldsymbol{\Lambda}_{g_L}^{(L)^T} + \boldsymbol{\psi}_{g_L}^{(L)})\ldots\big)\boldsymbol{\Lambda}_{g_2}^{(2)^T} + \boldsymbol{\psi}_{g_2}^{(2)}\right)\boldsymbol{\Lambda}_{g_1}^{(1)^T} + \boldsymbol{\psi}_{g_1}^{(1)}. \tag{5.6}$$

We refer to the GMM of Equation (5.3) as the "global GMM" of the DGMM so as not to confuse it with the GMM of the layers. In addition, the number of possible paths through the network is denoted as G. In fact, Equation (5.3) can be generalised to all layers: the margin distributions of all the scores $\boldsymbol{z}_i^{(l)}$ are Gaussian mixtures. So, by denoting $\tilde{\mathcal{G}}$ as all the possible paths from the lth layer, and by integrating out the latent variables of the layers that follow the lth layer, we have:

$$f(\boldsymbol{z}_i^{(l)};\boldsymbol{\theta}) = \sum_{\tilde{\boldsymbol{g}}\in\tilde{\mathcal{G}}} \tilde{\pi}_{\tilde{\boldsymbol{g}}}\mathcal{N}(\tilde{\boldsymbol{\mu}}_{\tilde{\boldsymbol{g}}}^{(l+1)}, \tilde{\boldsymbol{\Sigma}}_{\tilde{\boldsymbol{g}}}^{(l+1)}) \tag{5.7}$$

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

$$\tilde{\pi}_{\tilde{\boldsymbol{g}}} = \prod_{l'=l+1}^{L} \pi_{g_{l'}}, \tag{5.8}$$

$$\tilde{\boldsymbol{\mu}}_{\tilde{\boldsymbol{g}}} = \boldsymbol{\eta}_{g_{l+1}}^{(l+1)} + \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)}\left(\boldsymbol{\eta}_{g_{l+2}}^{(l+2)} + \boldsymbol{\Lambda}_{g_{l+2}}^{(l+2)}\big(\ldots(\boldsymbol{\eta}_{g_{L-1}}^{(L-1)} + \boldsymbol{\Lambda}_{g_{L-1}}^{(L-1)}\boldsymbol{\eta}_{g_L}^{(L)})\big)\right) \text{ and} \tag{5.9}$$

$$\tilde{\boldsymbol{\Sigma}}_{\tilde{\boldsymbol{g}}} = \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)}\left(\boldsymbol{\Lambda}_{g_{l+2}}^{(l+2)}\big(\ldots(\boldsymbol{\Lambda}_{g_L}^{(L)}\boldsymbol{\Lambda}_{g_L}^{(L)^T} + \boldsymbol{\psi}_{g_L}^{(L)})\ldots\big)\boldsymbol{\Lambda}_{g_{l+2}}^{(l+2)^T} + \boldsymbol{\psi}_{g_{l+2}}^{(l+2)}\right)\boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)^T} + \boldsymbol{\psi}_{g_{l+1}}^{(l+1)}. \tag{5.10}$$

To conclude, the DGMM has parameters $\boldsymbol{\theta} = (\boldsymbol{\eta}_{g_l}^{(l)}, \boldsymbol{\Lambda}_{g_l}^{(l)}, \boldsymbol{\psi}_{g_l}^{(l)}, \pi_{g_l}^{(l)})_{l\in\{1,\ldots,L\};g_l\in\{1,\ldots,G^{(l)}\}}$. It also has the latent variables $\boldsymbol{v}^{(l)}$ and $\boldsymbol{z}^{(l)}$ where $\boldsymbol{v}^{(l)}$ corresponds to the partition of the lth layer. In Table 5.1, we list the number of parameters to be estimated for the DGMM and other model-based clustering methods: the GMMs `VVV` and `EEE` models and the MFA model.

model name	mixing proportions	means	covariance matrix	factor loadings	noise co-variance matrix
GMM VVV	$G-1$	GR	$GJ(J+1)/2$	none	none
GMM EEE	$G-1$	GR	$J(J+1)/2$	none	none
MFA	$G-1$	GR	none	$GR[R-(R-1)/2]$	GR
DGMM	$\sum_{l=1}^{L} G^{(l)} - 1$	$\sum_{l=1}^{L} G^{(l)}R^{(l-1)}$	none	$\sum_{l=1}^{L} G^{(l)}R^{(l)}[R^{(l-1)} - (R^{(l)}-1)/2]$	$\sum_{l=1}^{L} G^{(l)}R^{(l-1)}$

Table 5.1 – Number of parameters for four model-based clustering methods.

5.2.2 Inference of the model

Similar to MFA, the DGMM has parameters and latent variables to estimate. Therefore, the EM algorithm is a candidate to optimise the log-likelihood function with respect to the parameters and latent variables. However, in the MFA model, the computation of the expectation of the complete data log-likelihood does not involve $p(\boldsymbol{z};\boldsymbol{\theta})$ since $\boldsymbol{z}$ is assumed

to be drawn from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ (see Section 2.4.2 for further details). For the DGMM, unless $l = L$, $\boldsymbol{z}_l^{(l)}$ is assumed to be drawn from an MFA with parameters $(\pi_{g_{l+1}}^{(l+1)}, \boldsymbol{\eta}_{g_{l+1}}^{(l+1)}, \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)}, \boldsymbol{\psi}_{g_{l+1}}^{(l+1)})_{g_{l+1} \in \{1,\dots,G^{(l+1)}\}}$. Thus, the parameters of the model are involved in the computation of $p(\boldsymbol{z}^{(l)}; \boldsymbol{\theta})$ which makes it difficult to optimise the auxiliary function with all the layers at the same time. A solution to this issue is to perform the optimisation layer by layer in an iteration of the EM algorithm. In this case, when we optimise the auxiliary function with respect to the parameters of layer l, the fact that $p(\boldsymbol{z}^{(l)}; \boldsymbol{\theta})$ depends on the parameters of layer $l+1$ is not a problem. Therefore, an EM algorithm similar to that in Section 2.2.6 can be performed. We detail the computations made at iteration q.

(A) First E-step Before iterating on the layers, the EM algorithm has to compute the expectation of partitions $\boldsymbol{v}$ of the **global** GMM of Equation (5.3). It corresponds to estimating which path $\boldsymbol{g} = (g_1, \dots, g_L)$ was followed in the network by each observation. This requires the computation of $\boldsymbol{\pi}_{\boldsymbol{g}}^{(q)}$, $\boldsymbol{\mu}_{\boldsymbol{g}}^{(q)}$ and $\boldsymbol{\Sigma}_{\boldsymbol{g}}^{(q)}$ from Equations 5.4, 5.5 and 5.6. Then, we compute $t_{i\boldsymbol{g}}^{(q)}$:

$$t_{i\boldsymbol{g}}^{(q)} = \mathbb{E}[v_{i\boldsymbol{g}} | \boldsymbol{x}; \boldsymbol{\theta}^{(q-1)}] = \frac{\pi_{\boldsymbol{g}}^{(q-1)} f_{\boldsymbol{g}}(\boldsymbol{x}_i; \boldsymbol{\mu}_{\boldsymbol{g}}^{(q-1)}, \boldsymbol{\Sigma}_{\boldsymbol{g}}^{(q-1)})}{\sum_{\boldsymbol{g}'=1}^{G} \pi_{\boldsymbol{g}'}^{(q-1)} f_{\boldsymbol{g}'}(\boldsymbol{x}_i | \boldsymbol{\mu}_{\boldsymbol{g}'}^{(q-1)}, \boldsymbol{\Sigma}_{\boldsymbol{g}}^{(q-1)})}. \tag{5.11}$$

Once $t_{i\boldsymbol{g}}^{(q)}$ is known for all i and $\boldsymbol{g}$, it is straightforward to compute the quantities $t_{ig_l}^{(l,q)}$, which correspond to the partitions of the GMM of each layer.
Then, the EM algorithm iterates on the layers $l \in \{1, \dots, L\}$ the second E-step and the M-step.

(B) Second E-step For layer l, this step consists in computing the expectations of $\boldsymbol{z}_i^{(l)}$ given $\boldsymbol{z}_i^{(l-1)}$ and $\boldsymbol{v}_i^{(q)}$:

- $\mathbb{E}_{p(\boldsymbol{z}^{(l)} | \boldsymbol{z}^{(l-1)}, \boldsymbol{v})}[\boldsymbol{z}_i^{(l)} | \boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}]$ and
- $\mathbb{E}_{p(\boldsymbol{z}^{(l)} | \boldsymbol{z}^{(l-1)}, \boldsymbol{v})}[\boldsymbol{z}_i^{(l)} {\boldsymbol{z}_i^{(l)}}^T | \boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}]$.

According to Equation (5.2) and (5.7), we have:

$$f(\boldsymbol{z}_i^{(l-1)} | \boldsymbol{z}_i^{(l)}, \boldsymbol{v}_i^{(q)}; \boldsymbol{\theta}^{(q-1)}) = \mathcal{N}(\boldsymbol{\eta}_{g_l}^{(l,q-1)} + \boldsymbol{\Lambda}_{g_l}^{(l,q-1)} \boldsymbol{z}_i^{(l)}, \boldsymbol{\psi}_{g_l}^{(l,q-1)}) \text{ and} \tag{5.12}$$

$$f(\boldsymbol{z}_i^{(l)} | \boldsymbol{v}_i; \boldsymbol{\theta}^{(q-1)}) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\tilde{\boldsymbol{g}}}^{(l+1)}, \tilde{\boldsymbol{\Sigma}}_{\tilde{\boldsymbol{g}}}^{(l+1)}). \tag{5.13}$$

By using the property of conditional multivariate Gaussian distributions, we get:

$$f(\boldsymbol{z}_i^{(l)} | \boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}; \boldsymbol{\theta}^{(q-1)}) = \mathcal{N}(\boldsymbol{\kappa}_{g_l}(\boldsymbol{z}_i^{(l-1)}), \boldsymbol{\zeta}_{g_l}), \tag{5.14}$$

where

$$\boldsymbol{\kappa}_{g_l}(\boldsymbol{z}_i^{(l-1)}) = \boldsymbol{\zeta}_{g_l} \left({\boldsymbol{\Lambda}_{g_l}^{(l,q-1)}}^T {\boldsymbol{\psi}_{g_l}^{(l,q-1)}}^{-1} (\boldsymbol{z}_i^{(l-1)} - \boldsymbol{\eta}_{g_l}^{(l,q-1)}) + (\tilde{\boldsymbol{\Sigma}}_{\tilde{\boldsymbol{g}}}^{(l+1)})^{-1} \tilde{\boldsymbol{\mu}}_{\tilde{\boldsymbol{g}}}^{(l+1)} \right) \text{ and} \tag{5.15}$$

$$\boldsymbol{\zeta}_{g_l} = \left((\tilde{\boldsymbol{\Sigma}}_{\tilde{\boldsymbol{g}}}^{(l+1)})^{-1} + \boldsymbol{\Lambda}_{g_l}^{(l,q-1)^T} \boldsymbol{\psi}_{g_l}^{(l,q-1)^{-1}} \boldsymbol{\Lambda}_{g_l}^{(l,q-1)} \right)^{-1}. \tag{5.16}$$

Viroli and McLachlan (2019) suggest using a stochastic E-step and generating S samples $\boldsymbol{z}_i^{(l)^{[s]}}$ from Equation (5.14). Then, the expectations can be approximated as follows:

$$\mathbb{E}_{p(\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l-1)},\boldsymbol{v})}[\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}] \approx \frac{\sum_{s=1}^{S} \boldsymbol{z}_i^{(l)^{[s]}}}{S} \text{ and} \tag{5.17}$$

$$\mathbb{E}_{p(\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l-1)},\boldsymbol{v})}[\boldsymbol{z}_i^{(l)} \boldsymbol{z}_i^{(l)^T}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}] \approx \frac{\sum_{s=1}^{S} \boldsymbol{z}_i^{(l)^{[s]}} \boldsymbol{z}_i^{(l)^{[s]^T}}}{S}. \tag{5.18}$$

However, another option is to approximate the expectations directly with the parameters $\boldsymbol{\kappa}_{g_l}$ and $\boldsymbol{\zeta}_{g_l}$:

$$\mathbb{E}_{p(\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l-1)},\boldsymbol{v})}[\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}] \approx \boldsymbol{\kappa}_{g_l}(\boldsymbol{z}_i^{(l-1)}) \text{ and} \tag{5.19}$$

$$\mathbb{E}_{p(\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l-1)},\boldsymbol{v})}[\boldsymbol{z}_i^{(l)} \boldsymbol{z}_i^{(l)^T}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}] \approx \boldsymbol{\kappa}_{g_l}(\boldsymbol{z}_i^{(l-1)}) \boldsymbol{\kappa}_{g_l}(\boldsymbol{z}_i^{(l-1)})^T + \boldsymbol{\zeta}_{g_l}. \tag{5.20}$$

Both alternatives were implemented, but the experiments we run in this chapter used the last one. Indeed, the Stochastic version seems to yield similar results when $S > 5$ and sometimes struggles to run all the iterations when $S < 5$, due to computational errors.

(C) M STEP For convenience, we note $\mathbb{E}_{p(\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l-1)},\boldsymbol{v})}$ as $\mathbb{E}$. We update the parameters as follows:

$$\boldsymbol{\eta}_{g_l}^{(l,q)} = \frac{\sum_{i=1}^{N} t_{ig_l}^{(l,q)} \left(\boldsymbol{z}_i^{(l-1)} - \boldsymbol{\Lambda}_{g_l}^{(l,q)} \mathbb{E}[\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}] \right)}{\sum_{i=1}^{N} t_{ig_l}^{(l,q)}},$$

$$\boldsymbol{\Lambda}_{g_l}^{(l,q)} = \frac{\sum_{i=1}^{N} t_{ig_l}^{(l,q)} \left((\boldsymbol{z}_i^{(l-1)} - \boldsymbol{\eta}_{g_l}^{(l,q)}) \mathbb{E}\left[\boldsymbol{z}_i|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}\right]^T \mathbb{E}[\boldsymbol{z}_i^{(l)} \boldsymbol{z}_i^{(l)^T}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}]^{-1} \right)}{\sum_{i=1}^{N} t_{ig_l}^{(l,q)}},$$

$$\boldsymbol{\psi}_{g_l}^{(l,q)} = \frac{\sum_{i=1}^{N} t_{ig_l}^{(l,q)} \left((\boldsymbol{z}_i^{(l-1)} - \boldsymbol{\eta}_{g_l}^{(l,q)})(\boldsymbol{z}_i^{(l-1)} - \boldsymbol{\eta}_{g_l}^{(l,q)})^T - (\boldsymbol{z}_i^{(l-1)} - \boldsymbol{\eta}_{g_l}^{(l,q)}) \mathbb{E}[\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i^{(q)}]^T \boldsymbol{\Lambda}_{g_l}^{(l,q)^T} \right)}{\sum_{i=1}^{N} t_{ig_l}^{(l,q)}},$$

$$\pi_g^{(l,q)} = \frac{1}{N} \sum_{i=1}^{N} t_{ig_l}^{(l,q)}.$$

5.2.3 Model selection

The BIC criterion applies to DGMM with the same expression $-2\log p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) + \nu \log N$ (see Section 2.1.5 for further details). The number of parameters ν is given by:

$$\nu = \sum_{l=1}^{L} \underbrace{(G^{(l)} - 1)}_{\text{mixing proportions}} + \underbrace{G^{(l)} R^{(l-1)}}_{\text{means}} + \underbrace{G^{(l)} R^{(l)} [R^{(l-1)} - (R^{(l)} - 1)/2]}_{\text{factor loadings}} + \underbrace{G^{(l)} R^{(l-1)}}_{\text{noise covariance matrix}},$$

with $R^{(l-1)} = J$ when $l = 1$.

5.3 Properties of the Deep GMM

To our knowledge, there is no assessment of the properties of the estimators of the DGMM in the literature. In this section, we propose to assess empirically the EM algorithm detailed above. We simulate data via the generative process of the DGMM with known parameters and partitions. Then, we run the algorithm to see if it estimates the parameters and partitions well. This investigation is useful because estimating the latent parameters is not easy due to the multi-layer architecture of the model and to the many latent variables. Furthermore, trying the inference algorithm on a model is a good way to know if the algorithm needs adjustments (e.g. different initialisations). All the experiments made here are available with the `deepMFA` package. In addition, Tables 5.7 and 5.8 list the correspondences between sections and scripts in Appendix 5.6.2.

5.3.1 Preliminary analysis: simulated data

5.3.1.a Settings for the simulated data set

The data set $\boldsymbol{x}$ was set with $N = 4000$ and $J = 8$. The network was built with two layers ($L = 2$) such that each layer has three neurons ($G^{(1)} = G^{(2)} = 3$). In addition, the latent scores were set to be of dimension $R^{(1)} = 5$ and $R^{(2)} = 2$ respectively. The parameters $\pi_g^{(l)}$ were set to $1/3$ for all g and for all l. The parameters $(\boldsymbol{\eta}_g^{(l)}, \boldsymbol{\Lambda}_g^{(l)}, \boldsymbol{\psi}_g^{(l)})_{g,l}$ were sampled from Gaussian whose parameters can be found in the file `configuration.R` of the R package `deepMFA`. For every experiment described below, the EM-algorithm is run twenty times on the same data set. We compare our results with baselines methods. We use the `mclust` package (Scrucca et al., 2016), which performs clustering with the fourteen GMMs (see Table 2.1) and returns the model with the lower BIC value. We also use the function `kmeans`.

5.3.1.b Results on the simulated data sets

We run the EM algorithm with two different initialisations of the partitions: random and kmeans. Figure 5.2 shows the resulting ARI for the DGMM and its two different initialisations, the `mclust` package and the `kmeans` function, with respect to the true partitions.
First, there is a small difference between the random initialisation and the kmeans initialisation. Second, the DGMM yields poor results since the higher ARI value it gets is equal to 0.80 (with random initialisation) even though the data set was generated through its generative process. In addition, we note that the `mclust` algorithm outperforms the DGMM by getting higher ARI values and by reaching an ARI equal to 1 on three simulations. It

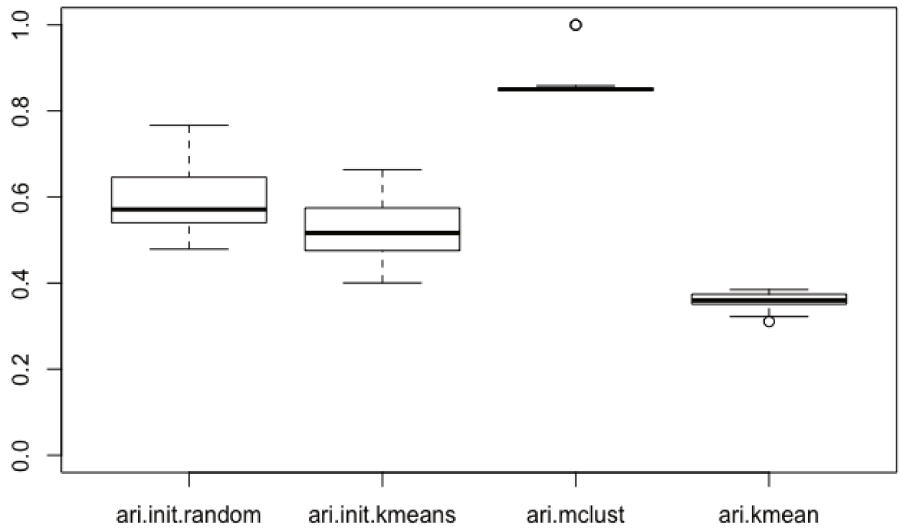

Figure 5.2 – Boxplots of the ARI for DGMM with random initialisation, DGMM with kmeans initialisation, `mclust` and `kmeans`.

is interesting to note that `mclust` yields these results with the model `VVV` (i.e. the most complex model) and a minimum BIC value equal to 192234.7 whereas the minimum BIC value of the DGMM is equal to 213528.6. In Figure 5.3, we plot all the BIC values for both, the DGMM and the `mclust` results and we see that `mclust` systematically gets lower BIC values.
In Figure 5.4, we plot the BIC values over the iterations of the DGMM and we see that they reach a plateau, meaning that the EM algorithm found a local maximum of the likelihood. If the EM algorithm is able to find local maxima, but fails to find the right partitions, it might mean that there are too many local maxima. To verify this, we run the EM algorithm a hundred times and compute the ARI of each result with respect to every other result. The mutual ARI are plotted in Figure 5.5, where we see that none of the mutual ARI is equal to 1 (except the ARI of a result with respect to itself). This means that the EM algorithm found 100 *different* local maxima.
The ARI mentioned above concerned the ARI for the global GMM. However, the generative process of the DGMM includes GMM for every layer. In Table 5.2, we show the ARI that the EM algorithm got for each layer. At the first layer, the ARI are high and can even be equal to 1. However, the ARI for the second layer are very low. This can mean that the latent scores are poorly estimated since the DGMM of layer 1 depends on the data $\boldsymbol{x}$, and the DGMM of layer 2 depends on the scores $\boldsymbol{z}^{(1)}$.
The last result we detail regards the partitions for the GMM of each layer and the partitions of the global GMM. Table 5.3 lists the clusters that are represented at layer 1, layer 2 and for the global GMM. We see that for several runs, the clusters got empty in some layers; for

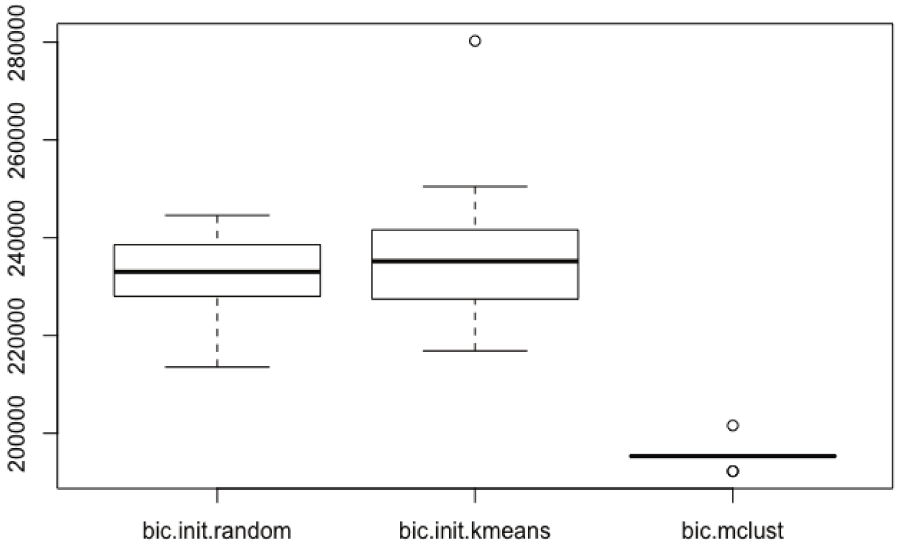

Figure 5.3 – Boxplots of the BIC values for DGMM with random initialisation, DGMM with kmeans initialisation and `mclust`.

example, in the first run, in the second layer, no observation was affiliated to clusters 1 and 2. In addition, even in the cases where all the clusters of layers 1 and 2 have observations (e.g. runs number 4 and 19), the global GMM does not have observations in its 9 clusters, meaning that even if each neuron of each layer has observations, not all the combinations of neurons of the two layers have observations. Therefore, there is a severe problem of clusters that get empty at the layers' level and at the global GMM's level too.

5.3.1.c Possible reasons for the poor results

There are several possible reasons for the poor results described in the previous section. First of all, the MFA model already suffers from unindentifiability issues that could worsen with the multilayer architecture of the DGMM. Second, the DGMM has many latent variables, which can be a severe problem when it comes to estimation. Indeed, in the context of configuration of Section 5.3.1.a, the latent scores alone represent $N \times (R^{(1)} + R^{(2)}) = 4000 \times (5+2) = 28000$ continuous values to estimate, which is very high, especifically given the fact that the network is not truly deep since it has only two layers. Finally, we have also observed that even though the clusters of each layer do not necessarily get empty, the clusters of the global GMM do so, which is also a problem for the partition estimation.

5.3.2 More experiments

In this section, we implement different versions of the EM algorithm for the DGMM to understand, in a first place, why the algorithm does not retrieve the true parameters and to

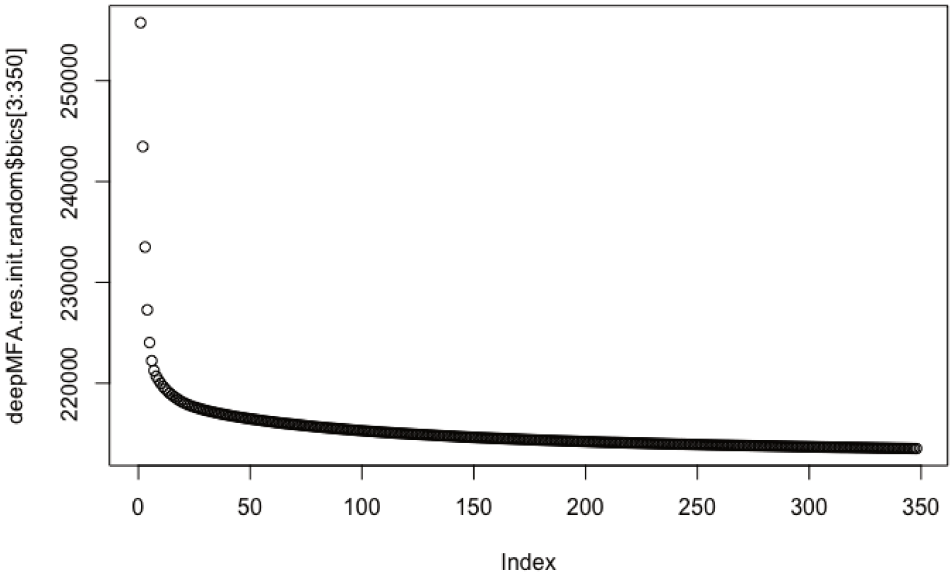

Figure 5.4 – BIC values over the iterations of the DGMM with random initialisation. The first two iterations are removed so that the plot's scale is not too big.

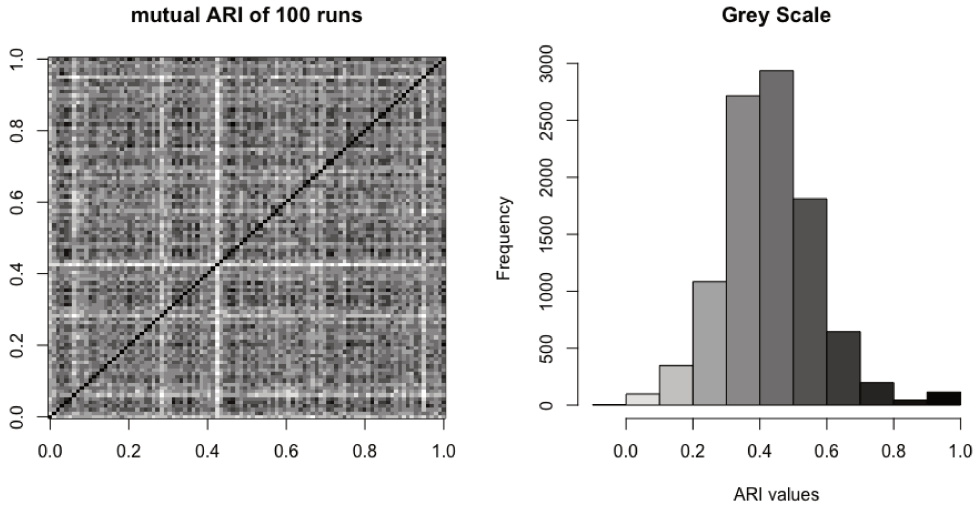

Figure 5.5 – Mutual ARI of 100 runs of the EM-algorithm. None of them gets the same final partitions.

see if other alternatives can yield better results.

run number	ARI of layer 1	ARI of layer 2
1	0.71	0.06
2	0.49	0.05
3	1	0.06
4	0.4	0.17
5	0.54	0.06
6	0.39	0.08
7	0.51	0.22
8	0.53	0.06
9	0.72	0.25
10	0.1	0.06
11	0.55	0
12	0.73	0.08
13	0.5	0.18
14	1	0.21
15	0.29	0.08
16	0.71	0.11
17	0.16	0
18	0.5	0.17
19	0.57	0.06
20	0.61	0.06

Table 5.2 – ARI results for the GMM of each layer. We see that the latent scores $\boldsymbol{z}^{(1)}$ do not seem to be estimated well.

run number	layer 1	layer 2	global GMM
1	1, 2, 3	3	7, 8, 9
2	1, 2, 3	1, 3	1, 2, 3, 7, 8, 9
3	1, 2, 3	1, 2	1, 2, 3, 4, 5, 6
4	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 7, 8
5	1, 2, 3	1, 3	1, 2, 3, 7, 8, 9
6	1, 2, 3	1	1, 2, 3
7	1, 2, 3	1	1, 2, 3
8	1, 2, 3	2	4, 5, 6
9	1, 2, 3	2, 3	4, 5, 6, 7, 8, 9
10	1, 2, 3	2, 3	4, 5, 6, 7, 8, 9
11	1, 2, 3	1	1, 2, 3
12	1, 2, 3	1	1, 2, 3
13	1, 2, 3	1, 2	1, 2, 3, 4, 5, 6
14	1, 2, 3	1, 2	1, 2, 3, 5, 6
15	1, 2, 3	2, 3	4, 5, 6, 7, 8, 9
16	1, 2, 3	2, 3	4, 5, 6, 7, 8, 9
17	1, 2, 3	1, 2	1, 2, 3, 4, 6
18	1, 2, 3	2, 3	4, 5, 6, 7, 8, 9
19	1, 2, 3	1, 2, 3	1, 2, 3, 4, 7, 8, 9
20	1, 2, 3	2	4, 5, 6

Table 5.3 – The clusters that are present at the end of the EM algorithm for the 20 runs.

5.3.2.a Fixing the scores $(\boldsymbol{z}^{(l)})_l$

The goal of these experiments is to verify the reason why the fact that the DGMM has many latent variables is an issue for the estimation of the partitions. We fix a part of the scores $\boldsymbol{z}$ to their true values. At each iteration, a percentage of the scores is fixed to its true value. In the experiment fix.z.100, all the scores are fixed to their true value. In the experiments fix.z.75 and fix.z.50 we respectively fix 75% and 50% of the scores. Figure 5.6 shows the results obtained in this configuration and we see that fixing the scores to their true value largely improves the performances of the EM algorithm. However, not all the ARI are equal to 1, meaning that even though the latent scores are known by the algorithm, it still may have issues in finding the true parameters and thus the true partitions. This indicates that the estimation task of this model is truly difficult.

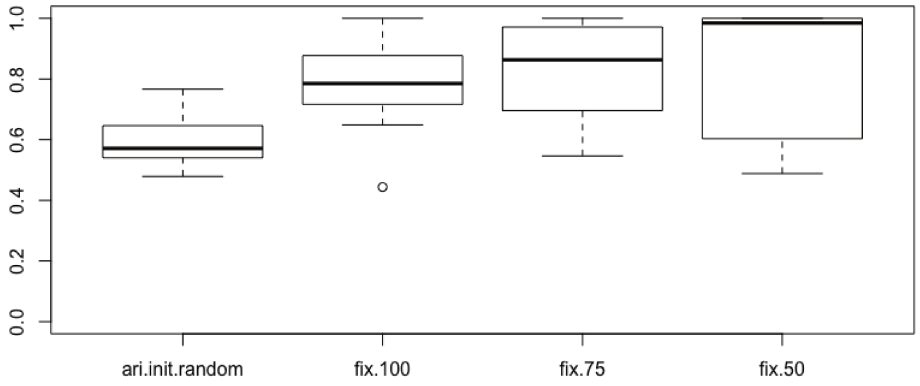

Figure 5.6 – Boxplots of the ARI for DGMM with random initialisation, and DGMM with fixed scores: fix.z.100, fix.z.75 and fix.z.50 have respectively 100%, 75% and 50% of their scores fixed to their true value at each iteration.

5.3.2.b Fixing the parameters $\boldsymbol{\theta}$

In this section, the EM algorithm is initialised with some parameters $\boldsymbol{\theta}_{\text{start}}$. In the first experiment (start.1), we initialised the parameters with the true parameters (the ones that were used to generate the data). Then, we initialise the algorithm with the true parameters with added noise. In the experiments start.2, start.3 and start.4, the parameters are the true parameters with added noise respectively sampled from the Gaussian distributions $\mathcal{N}(0, 0.5)$, $\mathcal{N}(0, 1)$ and $\mathcal{N}(0, 2.5)$. The aim of these experiments is to check whether the inference algorithm finds the right partitions when given the true parameters and whether a bad parameter initialisation deteriorates the performance. Figure 5.7 shows the results for start.1, start.2, start.3 and start.4; when the algorithm is initialised with the true parameters, it is not a problem to find the true partitions. However, when the initialisation parameters are a bit far from reality, the performances deteriorate fast. This indicates that the log-likelihood has too many local maxima so that if the EM algorithm is not initialised with

parameters near their true values, it will find another local maximum which is not the global maximum.

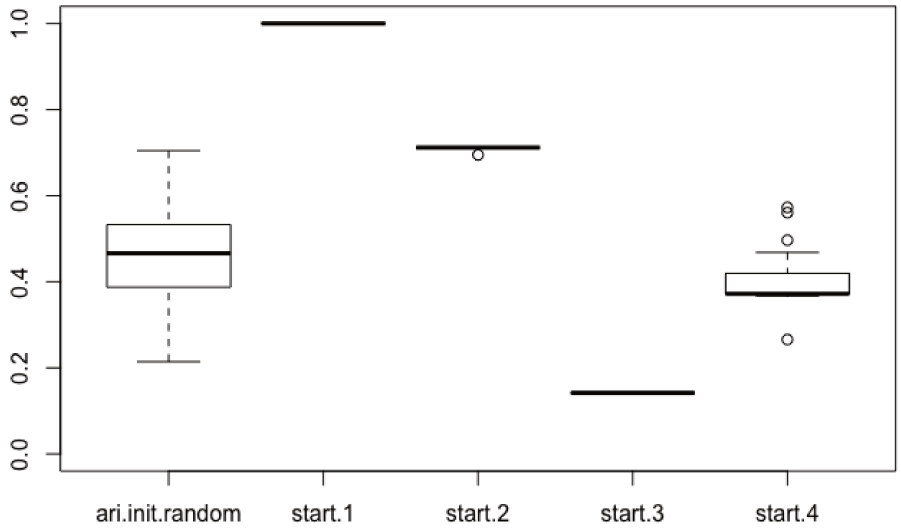

Figure 5.7 – Boxplots of the ARI for DGMM with random initialisation, and DGMM with initialisation of the true parameters (start.1) and true parameters with noise (start.2, start.3, start.4).

5.3.2.c Using a strategy to avoid empty clusters

In Table 5.3, we saw that the clusters of the layers and of the global DGMM get empty. Therefore, we implemented the same strategy as in Section 3.2.2.a. For the record, it consists in starting the algorithm with a random initialisation of the partitions. Then, for the first *nb.burnin* iterations, whenever a cluster of the global DGMM becomes empty, a percentage of the partitions is sampled from the Multinomial distribution $\mathcal{M}(1/G, \ldots, 1/G)$. Concretely, it means that at iteration q, with $q \leq nb.burnin$, if a cluster does not have any element, a percentage of the rows of matrix $\boldsymbol{v}^{(q)}$ is erased and randomly re-sampled. We experiment this strategy on the DGMM. To compute the percentage *nb.percent* of re-sampled partitions, we count the number of empty clusters *nb.empty* and compute:

$$nb.percent = rate \times N \times \frac{nb.empty}{\boldsymbol{G}}.$$

This formula is arbitrary; so we also arbitrarily fixed $nb.burnin = 200$. We tried this strategy with three different values of *rate* (0.2, 0.5 and 0.7). Figure 5.8 shows the results obtained and it can be seen that this strategy significantly improves the performance of the EM algorithm but it does not reach the same performances as the classic `mclust`. Furthermore, runs with $rate = 0.2$ yield better results.

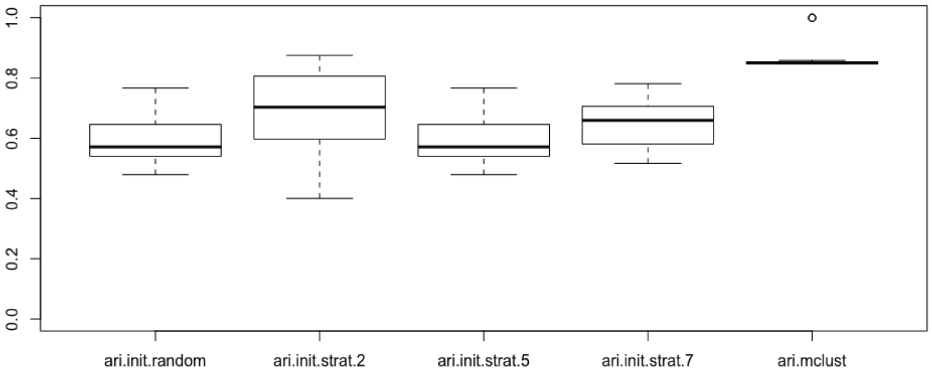

Figure 5.8 – Boxplots of the ARI for DGMM with random initialisation, DGMM with the strategy of initialisation strat.2 corresponds to a rate equal to 0.2, strat.2 corresponds to a rate equal to 0.5 and strat.7 corresponds to a rate equal to 0.7, and `mclust`.

Table 5.4 lists the clusters that are represented at layer 1, layer 2 and for the global GMM when using the strategy. It can be observed that even though some runs obtained some empty clusters, six of the runs succeeded in filling all the clusters. In model-based clustering, emptiness of clusters happens when the space of parameters is too sparse, which leads to spurious solutions. Forcing the algorithm in its first iterations not to get empty clusters helps reducing this space and it can lead to better results. However, in the case of DGMM, it does not solve all the estimation difficulties since it does not yield better results than the `mclust` function. In addition, it requires choosing the value of *rate*, which is not easy when the ground truth is unknown.

5.3.2.d Fixing the loading matrices after a burn-in period

In this experiment, we try to tackle the unindentifiability issues. For this purpose, we change the EM algorithm so that after a number *nb.burnin* of iterations, the parameters $(\mathbf{\Lambda}_{gl}^{(l)})_{g\in\{1,\dots,G^{(l)}\};\, l\in\{1,\dots,L\}}$ are fixed to their value at iteration $q = nb.burnin$. Here, we hope that the loading matrices reach their true value after a burn-in period and that the latent scores $(\boldsymbol{z}^{(l)})_l$ adapt to this value in the iterations that follow. We tried with different values of *nb.burnin* and got always the same result. In Figure 5.9, we show that this method worsens significantly the performances. If the unindentifiability issue is an obstacle to the good estimation of the parameters, then fixing the loading matrices after a burn-in period is not the solution to tackle this issue.

5.3.2.e Rotating the loading matrices

In this experiment, we try to tackle the unindentifiability issue of the MFA model by imposing a rotation on the $(\mathbf{\Lambda}_{gl}^{(l)})_{g\in\{1,\dots,G^{(l)}\};\, l\in\{1,\dots,L\}}$ parameters. We try two different rotations:

- The varimax rotation (Kaiser, 1958), the most common rotation option, is an orthogonal rotation of the factor axes to maximise the variance of the squared loadings of a

run number	layer 1	layer 2	global GMM
1	1, 2, 3	1, 2, 3	2, 3, 4, 5, 7, 8, 9
2	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 8, 9
3	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 9
4	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 8, 9
5	1, 2, 3	1, 2, 3	1, 3, 4, 7, 8, 9
6	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 9
7	1, 2, 3	1, 2, 3	1, 2, 3, 6, 7, 8, 9
8	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 9
9	1, 2, 3	1, 2, 3	1, 2, 5, 6, 7, 9
10	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 9
11	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 8, 9
12	1, 2, 3	1, 2, 3	1, 2, 3, 5, 6, 8, 9
13	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 7, 8, 9
14	1, 2, 3	1, 2, 3	1, 2, 3, 5, 6, 8
15	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 8, 9
16	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 8, 9
17	1, 2, 3	1	1, 2, 3
18	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 7, 8, 9
19	1, 2, 3	1, 2, 3	1, 2, 3, 5, 6, 7, 8, 9
20	1, 2, 3	1, 2, 3	1, 2, 3, 4, 5, 6, 7, 8, 9

Table 5.4 – The clusters that are present at the end of the EM algorithm for the 20 runs with the initialisation strategy; we see that several runs fill all the clusters of the layers and of the global GMM.

factor (column) on all the variables (rows) in a factor loading matrix.

- The procrustean rotation (Jackson, 2005) is a rotation with two matrices as input. It is desired to find the rotation that will best approximate one to the other. In our work, we use this rotation so that $\mathbf{\Lambda}_{g_l}^{(l,q)}$ best approximates $\mathbf{\Lambda}_{g_l}^{(l,q-1)}$, for the loading matrices not to change radically from an iteration to the other.

It is obvious that when the loading matrices are rotated, the scores $(\boldsymbol{z}^{(l)})_l$ also need to be rotated. Figure 5.10 shows the results obtained. We see that using the varimax rotation does not improve the performances at all whereas using the procrustes rotation does not systematically get better results than the original EM algorithm but is able to get slightly better ARIs for some runs. Overall, using rotations does not get better results than the `mclust` function, which can mean that the unindentifiability of the DGMM is not the only issue of the model.

5.3.2.f Constraining $\mathbf{\Lambda}_{g_l}^{(l)}$ to be a lower triangular full rank matrix

We implement the EM algorithm to force some values of $\mathbf{\Lambda}_{g_l}^{(l)}$ to be equal to 0, so that it is a lower triangular full rank matrix. The aim of this experiment is to reduce the number of parameters to be estimated and to decomplexify the model. The simulation settings were slightly changed so that the parameters respected the same constraint. These settings can be found in the file `configuration-upper-tri.R`. In this case, we also had to run again the original EM algorithm, the `mclust` and `kmeans` functions because the parameters values had changed. Figure 5.11 shows the ARI for every function. We see that applying this constraint does not yield better results. If simplifying the parameters of the model does not improve

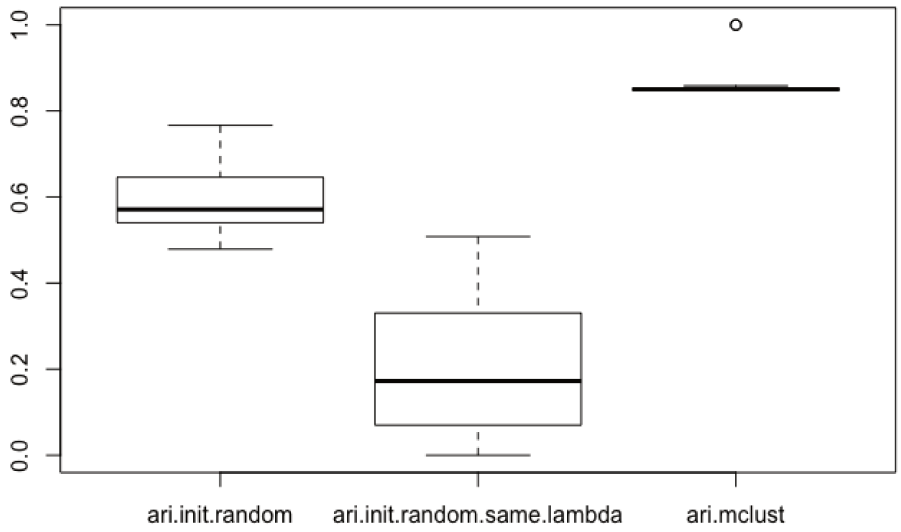

Figure 5.9 – Boxplots of the ARI for DGMM with random initialisation, DGMM with loading matrices fixed after a burn-in period and `mclust`.

the performances, it can mean that the estimation problems are also due to the many latent variables of the models; therefore, constraining some values of the loading matrices is not enough to address the high dimensionality of the latent scores $(\boldsymbol{z}^{(l)})_l$.

5.3.2.g Constraining $\boldsymbol{\Lambda}_{g_l}^{(l)^T}\boldsymbol{\psi}_{g_l}^{(l)^{-1}}\boldsymbol{\Lambda}_{g_l}^{(l)}$ to be diagonal

We implement the EM algorithm with the constraint that $\boldsymbol{\Lambda}_{g_l}^{(l)^T}\boldsymbol{\psi}_{g_l}^{(l)^{-1}}\boldsymbol{\Lambda}_{g_l}^{(l)}$ is diagonal, as it is suggested in Viroli and McLachlan (2019). In Figure 5.12, it can be seen that the algorithm worsens its performance when applying this constraint. If the unindentifiability issue is an obstacle to the good estimation of the parameters, then, constraining $\boldsymbol{\Lambda}_{g_l}^{(l)^T}\boldsymbol{\psi}_{g_l}^{(l)^{-1}}\boldsymbol{\Lambda}_{g_l}^{(l)}$ to be diagonal is not the solution to tackle this issue.

5.3.2.h Using the Newton-Raphson algorithm in one of two iterations

The EM algorithm is not the only optimisation algorithm able to infer the MFA and DGMM parameters. Other gradient-based algorithms such as the Newton-Raphson (NR) algorithm can be used. An advantage of this algorithm is that it does not require computing the expectation of the latent scores $(\boldsymbol{z}^{(l)})_l$ whereas its main disadvantage is that the convergence is not guaranteed, and sometimes it can not run all the iterations because of computational errors due to matrix inversions. We were not able to make the Newton-Raphson algorithm work on the DGMM; however, we tried to alternate two M-steps to estimate the loading factors $\boldsymbol{\Lambda}_{g_l}^{(l)}$. When q was even, we used the M-step described in Section 5.2.2. When q was

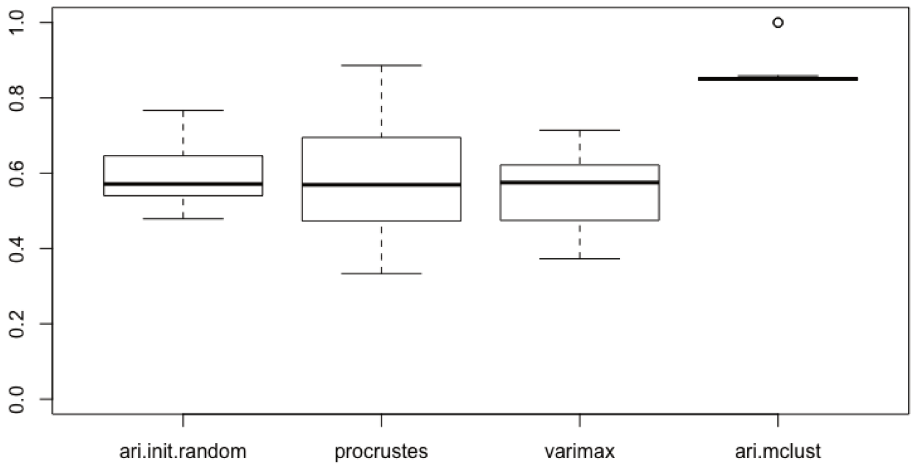

Figure 5.10 – Boxplots of the ARI for DGMM with random initialisation, DGMM with two different rotations of the loading matrices.

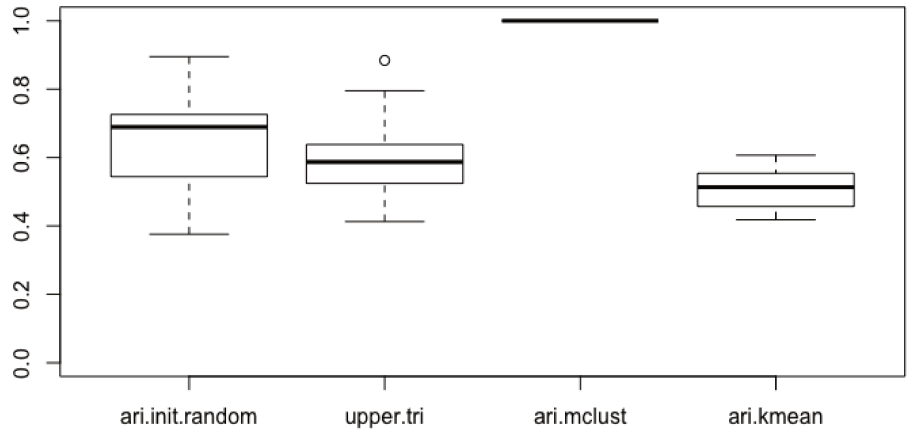

Figure 5.11 – Boxplots of the ARI for DGMM with random initialisation, DGMM with the constraint on the factor loading matrices, `mclust` and kmeans.

odd, we used the same M-step to update the parameters $(\boldsymbol{\eta}_{g_l}^{(l)})_{g_l,l}, \boldsymbol{\psi}_{g_l}^{(l)})_{g_l,l})_{g_l,l}$ and we used a variant NR algorithm, referred to as the "Nelder-Mead" algorithm (Nelder and Mead, 1965), to update the parameters $(\boldsymbol{\Lambda}_{g_l}^{(l)})_{g_l,l}$. We refer to this variant of the EM algorithm as the

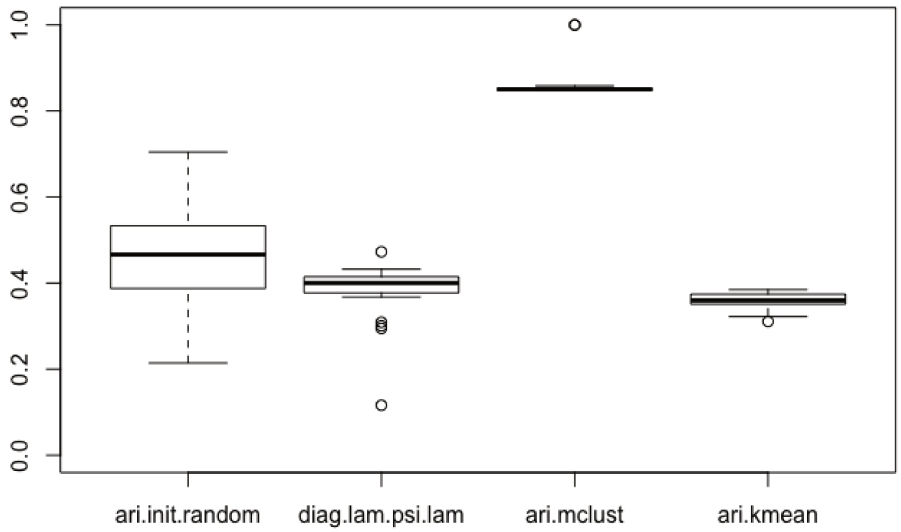

Figure 5.12 – Boxplots of the ARI for DGMM with random initialisation, DGMM with the constraint on the factor loading matrices, `mclust` and kmeans.

"hybrid EM algorithm".

The Nelder-Mead algorithm can be very slow to estimate the parameters. That is the reason why we reduced the number of observations ($N = 2000$) and the number of iterations ($nb.iter = 150$) for this experiment. We also had to run the standard EM algorithm, `mclust` and the `kmeans` function to be able to compare the results with these new settings. In this context (Figure 5.13), the hybrid EM algorithm yields better results than the original EM algorithm. Avoiding calculating the expectation of the scores for certain iterations improves the results, which may reinforce the idea that the poor results of the EM algorithm are due to too many latent variables. However, using the NR algorithm does not yield satisfying results either, which means that it is not able to tackle all the estimation issues for the DGMM. In addition, this method runs slowly and requires choosing the optimisation method to compute the loading matrices.

5.3.2.i Applying the hybrid EM with Procrustes rotation and the initialisation strategy

Until now, three modifications of the original EM algorithm have seemed to improve the performances: the use of the initialisation strategy (Section 5.3.2.c), the use of the procrustean rotation on the factor loadings (Section 5.3.2.e) and the hybrid EM algorithm (Section 5.3.2.h). In this experiment, we propose to use these three tweaks. Figure 5.14 shows the ARI obtained for the configurations: the random initialisation, the procrustes rotation, the mix of the hybrid EM algorithm, the procrustean rotation and the initialisation strategy, and the `mclust` function. We see that using the three modifications does not yield better results and that `mclust` is still the best method to find the true partitions.

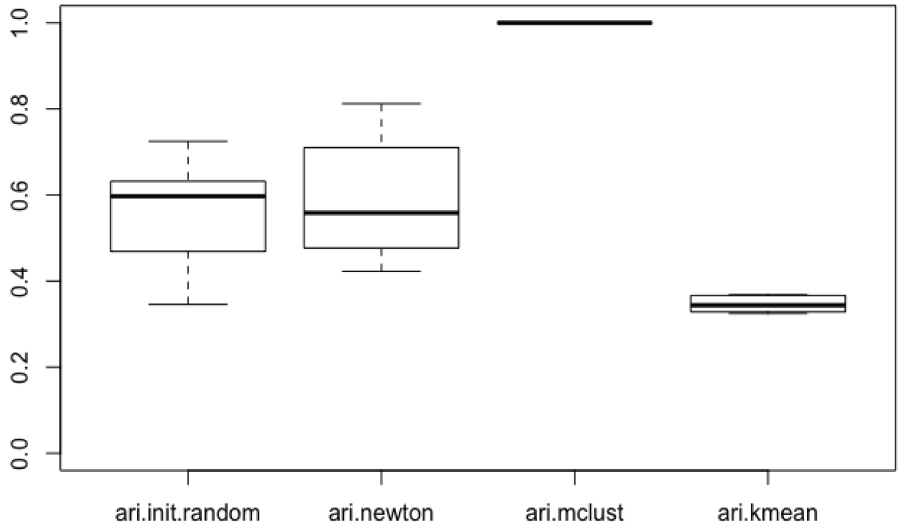

Figure 5.13 – Boxplots of the ARI for DGMM with random initialisation, DGMM with Newton-Raphson, `mclust` and kmeans.

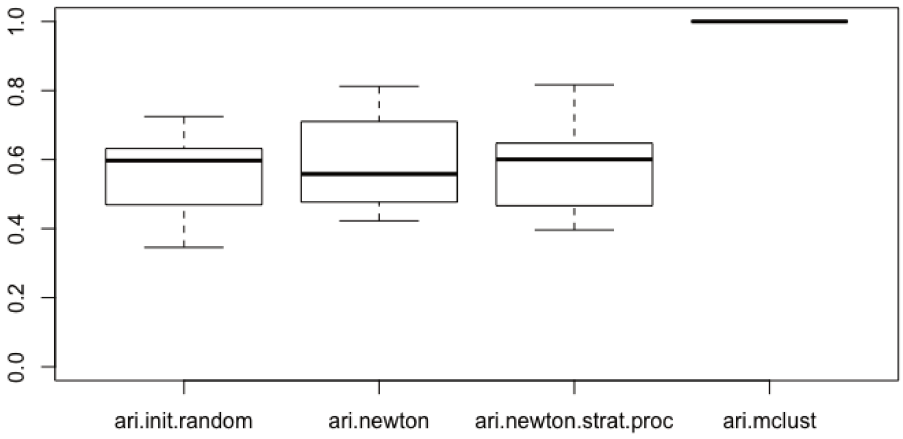

Figure 5.14 – Boxplots of the ARI for DGMM with random initialisation, DGMM with the hybrid EM only, DGMM with hybrid EM, the strategy of initialisation and the procrustean rotation, and `mclust`.

5.3.3 Applying the DGMM to real data sets

In the previous section, we have seen that the EM algorithm has trouble finding the true partitions of data simulated with the DGMM generative process. In this section, we run the algorithm with real data sets.

5.3.3.a The MNIST data set

PRESENTATION OF THE DATASET The MNIST dataset (LeCun and Cortes, 2010) is the most popular data set in deep learning. It has 60000 images of handwritten digits numbers with grayscale levels. Each image was normalized to fit into a 28×28 pixels box. The 28×28 matrices were then reshaped as vectors of length 784. The result is a 60000×784 matrix with the grayscale level of each pixel from each image. The dataset contains a 60000 long vector as well, which indicates the actual number represented by the images. Figure 5.15 shows how some images in the dataset look like.

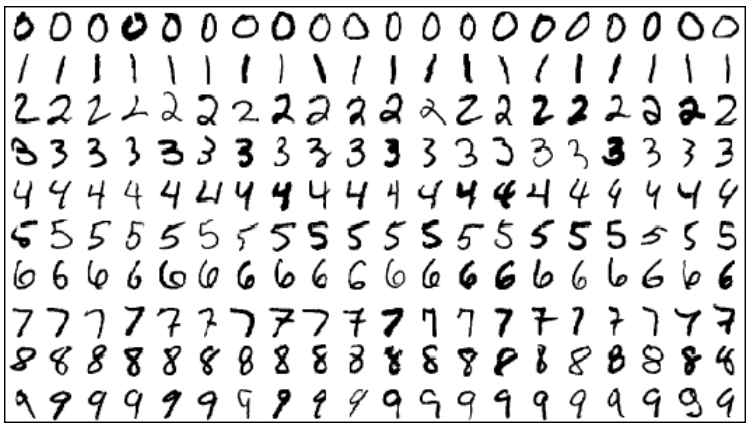

Figure 5.15 – Samples of the MNIST dataset.

PREPROCESSING AND ALGORITHM SETTINGS We run a Principal Component Analysis on the data set and kept the 20 principal features of the PCA. Then, we randomly sampled 1000 observations. Therefore, we have $N = 1000$ and $J = 20$. We used a random initialisation and fixed $G^{(1)} = 5$ and $G^{(2)} = 2$ (so that we have $5 \times 2 = 10$ clusters); and $R^{(1)} = 15$, $R^{(1)} = 10$ and $nb.init = 50$. We run the EM algorithm 10 times with these settings. We also used the `mclust` package and run the `kmeans` function to compare our results.

RESULTS Figure 5.16 shows the ARI obtained on the MNIST data set. We see that the DGMM yields better results than `mclust` and `kmeans`.
Another interesting result is that the EM algorithm does not seem to suffer from emptiness of clusters in the case of the MNIST data set. Table 5.5 lists the clusters that are represented at layer 1, layer 2 and for the global GMM where all the clusters are always filled.

USING THE DGMM TO SAMPLE NEW DIGITS WITH DIFFERENT NUMBERS OF LAYERS Here, the goal was to sample new MNIST images from the DGMM generative process. To this purpose, we had to use the MNIST data set without performing a PCA beforehand. We run the EM algorithm once, with the settings $L = 2$, $G^{(1)} = 5$, $G^{(2)} = 2$, $R^{(1)} = 10$, $R^{(2)} = 3$. We drew

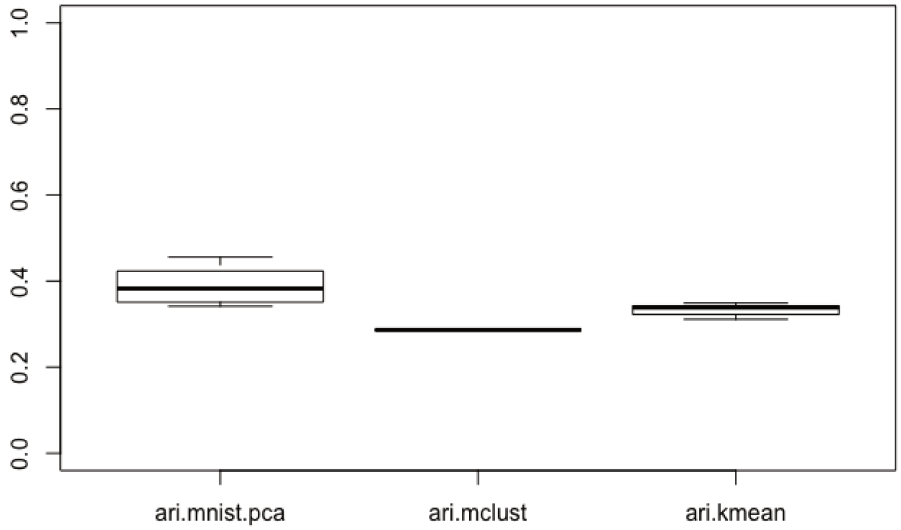

Figure 5.16 – Boxplots of the ARI for DGMM with random initialisation, `mclust` and kmeans on the MNIST data set.

samples from each of the clusters of these three DGMMs. We represent these samples in Figure 5.17. It can be observed that, for some of the clusters, the samples are readable and we can recognise a number while in some other cases, it is more difficult to distinguish it.

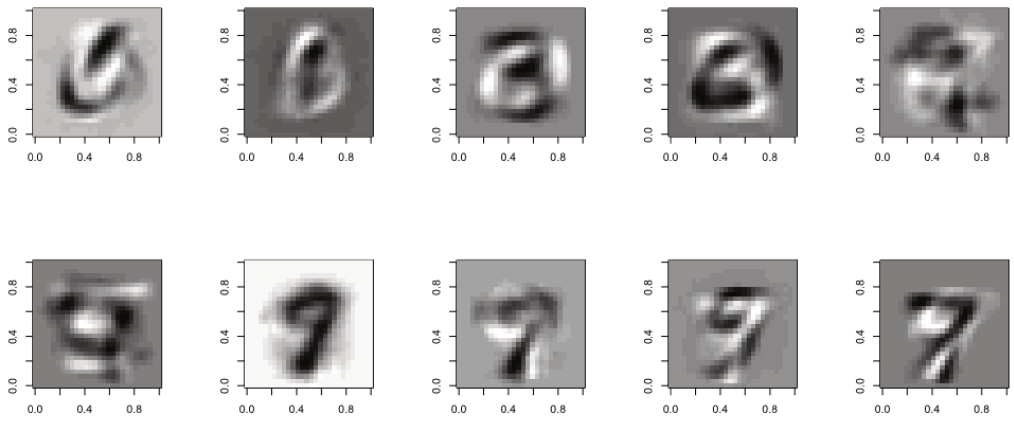

Figure 5.17 – Samples from the 10 clusters with the DGMM generative process and parameters estimated by the EM algorithm.

run number	layer 1	layer 2	global GMM
1	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
2	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
3	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
4	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
5	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
6	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
7	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
8	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
9	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
10	1, 2, 3, 4, 5	1, 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Table 5.5 – The clusters that are present at the end of the EM algorithm for the 10 runs with the MNIST data set. None of the clusters is empty.

5.3.3.b The ecoli data set

The ecoli data set consists of $N = 336$ proteins classified into their various cellular localization sites based on their amino acid sequences. There are $J = 7$ variables and $G = 8$ really unbalanced groups that make the clustering task rather difficult: cp cytoplasm (143), inner membrane without signal sequence (77), periplasm (52), inner membrane, uncleavable signal sequence (35), outer membrane (20), outer membrane lipoprotein (5), inner membrane lipoprotein (2), inner membrane, cleavable signal sequence (2). These data are available from the UCI machine learning repository*. We run the `mclust` and `kmeans` function 10 times to compare our results, and we run the EM-algorithm with the initialisation strategy and the Procrustes rotation. In addition, we set $L = 2$, $R^{(1)} = 3$, $R^{(2)} = 2$, $G^{(1)} = 4$, $G^{(2)} = 2$, $nb.iter = 30$, $nb.burnin = 15$ and $rate = 0.1$. Results are shown in Figure 5.18, where we see that the DGMM is able to get better results than its competitors. However, it is also less stable, in the sense that it can also get much worse results regarding the partition estimation.

In Table 5.6, we list the ARI and BIC values for each of the 10 runs on the data set. We see that the 7th run got the lowest BIC value and also yielded the highest ARI score. Similarly, we see that the run with the highest BIC value also corresponds to the run with the lowest ARI value. This means that even though the DGMM model gets less stable results than the two other methods, the BIC seems to be a good criterion to choose among the runs when the ground-truth is not known.

5.3.4 Conclusion on the experiments

The DGMM was ingeniously designed to stack MFA layers in a deep learning fashion. In theory, this would allow capturing complex structures in the data and its nature as a factor analyser would allow performing clustering on data sets where there are more variables than observations. However, in practice, the architecture of DGMM involves many latent variables and parameters resulting in a model that is difficult to infer.

In this section, we simulate a data set with the DGMM generative process and try to modify the original EM algorithm so that it can address these issues. However, none of these implementations yields satisfying results in the sense that they are not able to estimate the true parameters and that a classical GMM always obtains better performances to find

*https://archive.ics.uci.edu/ml/datasets/Ecoli

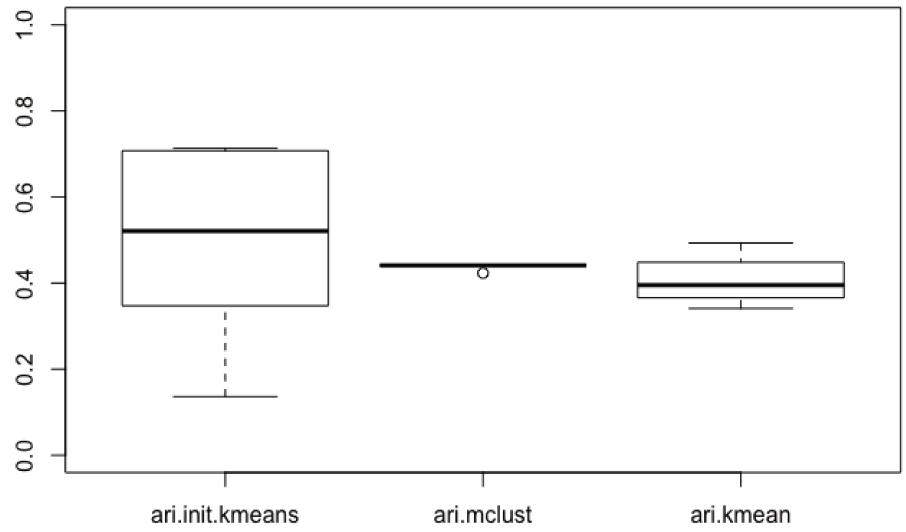

Figure 5.18 – Boxplots of the ARI for DGMM with random initialisation, `mclust` and kmeans on the ecoli data set.

the true partitions. Nonetheless, we also run the EM algorithm on real data sets, whose underlying generative process is unknown. In this case, the DGMM seems to yield better results than the GMMs and the kmeans algorithm. This can be explained by its deep learning architecture. Indeed, the neural networks experimentally reach the state-of-the-art performances in many domains and particularly on complex data such as image data sets. However, the mathematical proof of their efficiency is still an active research topic together with the theoretical aspects of their behaviour.

5.4 Suggestion of extension of DGMM to categorical data

The original purpose of studying DGMM was to build a new model that would have extended the DGMM to categorical and mixed data. In this section, we define the mathematical aspects of this model but we do not go further with experiments given the previous results of Section 5.3.

5.4.1 Latent Gaussian Models for discrete data

Here, we consider the data are discrete. Similar to other sections, we have N data observations with ith observed vector denoted by $\boldsymbol{x}_i$. However, each element of $\boldsymbol{x}_i$ can take values from a discrete set. We consider latent Gaussian vectors $\boldsymbol{z}_i$ that follow a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The probability of each element $\boldsymbol{x}_i$ is parameterized in terms of the linear predictor $\boldsymbol{\chi}_i$. The predictor $\boldsymbol{\chi}_i$ is an m-long vector,

run number	ARI score	BIC value
1	0.51	-16155
2	0.648	-16411.03
3	0.424	-16528.57
4	0.532	-16268.73
5	0.25	-16176.52
6	0.347	-16292.69
7	**0.713**	**-16581.54**
8	0.709	-16337.49
9	0.707	-16181.97
10	*0.136*	*-14694.59*

Table 5.6 – ARI and BIC for the 10 runs of the DGMM on the ecoli data set.

whose rth element is written as $\boldsymbol{\chi}_{ir}$; $\boldsymbol{\chi}_i$ is defined through $\boldsymbol{W}$, a real valued matrix of size $m \times L$ and $\boldsymbol{w}$, a J length real valued vector. Therefore, the LGM for discrete data follows the generative process:

$$\begin{aligned}\boldsymbol{z}_i &\sim \mathcal{N}(\widetilde{\mu}, \widetilde{\Sigma}),\\ \boldsymbol{\chi}_i &= \boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{w},\\ \boldsymbol{x}_i &\sim \mathcal{M}(1, \boldsymbol{\phi}(\boldsymbol{\chi}_i)),\end{aligned}$$

where $\boldsymbol{\phi}(\boldsymbol{\chi}_i)$ is an m-long vector whose rth element $\phi_r(\boldsymbol{\chi}_i)$ is:

$$\phi_r(\boldsymbol{\chi}_i) = \exp\left\{\boldsymbol{\chi}_{ir} - \log\left(\sum_{r'=1}^{m}\exp(\boldsymbol{\chi}_{ir'})\right)\right\}.$$

So, we have:

$$p(\boldsymbol{x}_i = r|\boldsymbol{z}_i;\boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\chi}_{ir})}{\sum_{r'=1}^{m}\exp(\boldsymbol{\chi}_{ir'})}. \tag{5.21}$$

5.4.2 LGM and DGMM

In the DGMM, we can use the Latent Gaussian Model described above by considering that the latent variables ($\boldsymbol{z}_i$) were drawn from a DGMM. In this case, we assume that an observation $\boldsymbol{x}_i$ was generated as folllows:

$$
\begin{aligned}
&\boldsymbol{x}_i \sim \mathcal{M}(1, \boldsymbol{\phi}(\boldsymbol{\chi}_i)), \text{ with} \\
&\boldsymbol{\chi}_i = \boldsymbol{W}\boldsymbol{z}_i^{(0)} + \boldsymbol{w} \\
&\boldsymbol{z}_i^{(0)} = \boldsymbol{\eta}_{g_1}^{(1)} + \boldsymbol{\Lambda}_{g_1}^{(1)}\boldsymbol{z}_i^{(1)} + \boldsymbol{u}_i^{(1)} \text{ with prob. } \pi_{g_1}^{(1)},\ g_1 \in \{1, \dots, G^{(1)}\} \\
&\boldsymbol{z}_i^{(1)} = \boldsymbol{\eta}_{g_2}^{(2)} + \boldsymbol{\Lambda}_{g_2}^{(2)}\boldsymbol{z}_i^{(2)} + \boldsymbol{u}_i^{(2)} \text{ with prob. } \pi_{g_2}^{(2)},\ g_2 \in \{1, \dots, G^{(2)}\} \\
&\dots \\
&\boldsymbol{z}_i^{(l)} = \boldsymbol{\eta}_{g_{l+1}}^{(l+1)} + \boldsymbol{\Lambda}_{g_{l+1}}^{(l+1)}\boldsymbol{z}_i^{(l+1)} + \boldsymbol{u}_i^{(l+1)} \text{ with prob. } \pi_{g_{l+1}}^{(l+1)},\ g_{l+1} \in \{1, \dots, G^{(l+1)}\} \\
&\dots \\
&\boldsymbol{z}_i^{(L-1)} = \boldsymbol{\eta}_{g_L}^{(L)} + \boldsymbol{\Lambda}_{g_L}^{(L)}\boldsymbol{z}_i^{(L)} + \boldsymbol{u}_i^{(L)} \text{ with prob. } \pi_{g_L}^{(L)},\ g_L \in \{1, \dots, G^{(L)}\}.
\end{aligned} \tag{5.22}
$$

From $\boldsymbol{z}_i^{(0)}$, the generative model is exactly the same as for the DGMM for continuous data. However, in the latter, we have $\boldsymbol{z}_i^{(0)} = \boldsymbol{x}_i$ and not having this equality in the categorical case can arise some issues for the model inference.

5.4.2.a Why are categorical data a problem for the inference?

In the DGMM, it is important to recall that the EM algorithm requires computing the expectations $\mathbb{E}_{p(\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l-1)},\boldsymbol{v})}[\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i]$ and $\mathbb{E}_{p(\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l-1)},\boldsymbol{v})}[\boldsymbol{z}_i^{(l)}{\boldsymbol{z}_i^{(l)}}^T|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i]$.
We can do that because:

$$
p(\boldsymbol{z}_i^{(l)}|\boldsymbol{z}_i^{(l-1)}, \boldsymbol{v}_i; \boldsymbol{\theta}) = p(\boldsymbol{z}_i^{(l)}|\boldsymbol{v}_i; \boldsymbol{\theta}) \times \frac{p(\boldsymbol{z}_i^{(l-1)}|\boldsymbol{z}_i^{(l)}, \boldsymbol{v}_i; \boldsymbol{\theta})}{p(\boldsymbol{z}_i^{(l-1)}|\boldsymbol{v}_i; \boldsymbol{\theta})}, \tag{5.23}
$$

and these three terms are Gaussians, which leads to Equation (5.14).
However, in the categorical case, for $l = 0$, we need to compute $p(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i; \boldsymbol{\theta})$ with:

$$
p(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i, \boldsymbol{v}_i; \boldsymbol{\theta}) = p(\boldsymbol{z}_i^{(0)}|\boldsymbol{v}_i; \boldsymbol{\theta}) \times \frac{p(\boldsymbol{x}_i|\boldsymbol{z}_i^{(0)}; \boldsymbol{\theta})}{p(\boldsymbol{x}_i; \boldsymbol{\theta})}, \tag{5.24}
$$

and not all these terms are Gaussians. The only Gaussian term is $p(\boldsymbol{z}_i^{(0)}|\boldsymbol{v}_i; \boldsymbol{\theta})$ and we cannot find an analytic solution.

5.4.3 Solution for categorical data

5.4.3.a Importance Sampling

GENERIC IDEA OF IMPORTANCE SAMPLING Importance sampling (IS) is a useful technique when we want to approximate an expectation of the form:

$$
\mathbb{E}_{p(\boldsymbol{z})}[h] = \int h(\boldsymbol{z}_i)p(\boldsymbol{z}_i)d\boldsymbol{z}_i. \tag{5.25}
$$

The idea is to choose a proposal distribution $q(\boldsymbol{z}_i)$ from which it is easy to draw samples and rewrite the integral as follows:

$$
\mathbb{E}_{p(\boldsymbol{z})}[h] = \int h(\boldsymbol{z}_i)\frac{p(\boldsymbol{z}_i)}{q(\boldsymbol{z}_i)}q(\boldsymbol{z}_i)d\boldsymbol{z}_i. \tag{5.26}
$$

We can approximate the expectation with:

$$\mathbb{E}_{p(\boldsymbol{z})}[h] \approx \frac{1}{S}\sum_{s=1}^{S}\frac{p(\boldsymbol{z}_i^{[s]})}{q(\boldsymbol{z}_i^{[s]})}h(\boldsymbol{z}_i^{[s]}), \tag{5.27}$$

where $z_i^{[s]} \sim q(z)$.

APPLYING IS TO OUR PROBLEM We want to compute:

$$\mathbb{E}_{p(\boldsymbol{z}^{(0)}|\boldsymbol{x},\boldsymbol{v})}[\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i, \boldsymbol{v}_i].$$

Compared to the definition of IS above, we have $\boldsymbol{z}_i = \boldsymbol{z}_i^{(0)}$, $h(\boldsymbol{z}_i) = \boldsymbol{z}_i^{(0)}$ and $p(\boldsymbol{z}_i) = p(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i, \boldsymbol{v}_i; \boldsymbol{\theta})$. Let us consider we know a proposal distribution $q(\boldsymbol{z}_i^{(0)})$. Replacing the corresponding terms in Equation (5.27), we have:

$$\mathbb{E}_{p(\boldsymbol{z}^{(0)}|\boldsymbol{x},\boldsymbol{v})}[\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i, \boldsymbol{v}_i] \approx \frac{1}{S}\sum_{s=1}^{S}\frac{p(\boldsymbol{z}_i^{(0)^{[s]}}|\boldsymbol{x}_i)}{q(\boldsymbol{z}_i^{(0)^{[s]}})}\boldsymbol{z}_i^{(0)^{[s]}}, \tag{5.28}$$

with $\boldsymbol{z}_i^{(0)^{[s]}} \sim q(\boldsymbol{z}_i^{(0)})$.
IS seems to be a good candidate to compute the expectation but, in our case, we still have two issues:

- IS requires computing $p(\boldsymbol{z}_i^{(0)^{[s]}}|\boldsymbol{x}_i; \boldsymbol{\theta})$, which is not feasible for us to do and,
- we do not know how to choose proposal distribution $q(\boldsymbol{z}_i^{(0)})$.

5.4.3.b Handling unnormalised distribution

IS can handle cases where the distribution $p(\boldsymbol{z}_i^{(0)^{[s]}}|\boldsymbol{x}_i; \boldsymbol{\theta})$ can only be evaluated up to a normalisation constant, so that $p(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i; \boldsymbol{\theta}) = \widetilde{p}(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i; \boldsymbol{\theta})/C_p$ where $\widetilde{p}(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i; \boldsymbol{\theta})$ can be evaluated easily, whereas C_p is unknown. Similarly, IS also handles the case where the proposal distribution $q(\boldsymbol{z}_i^{(0)}) = \widetilde{q}(\boldsymbol{z}_i^{(0)})/C_q$ has the same property. We then have:

$$\begin{aligned}\mathbb{E}_{p(\boldsymbol{z}^{(0)}|\boldsymbol{x},\boldsymbol{v})}[\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i, \boldsymbol{v}_i] &= \frac{C_q}{C_p}\int \boldsymbol{z}_i^{(0)}\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i; \boldsymbol{\theta})}{\widetilde{q}(\boldsymbol{z}_i^{(0)})}q(\boldsymbol{z}_i^{(0)})d\boldsymbol{z}_i^{(0)} \\ &\approx \sum_{s=1}^{S}\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)^{[s]}}|\boldsymbol{x}_i)/\widetilde{q}(\boldsymbol{z}_i^{(0)^{[s]}})}{\sum_{s'=1}^{S}\widetilde{p}(\boldsymbol{z}_i^{(0)^{[s']}}|\boldsymbol{x}_i)/\widetilde{q}(\boldsymbol{z}_i^{(0)^{[s']}})}\boldsymbol{z}_i^{(0)^{[s]}},\end{aligned} \tag{5.29}$$

with $\boldsymbol{z}_i^{(0)^{[s]}} \sim q(\boldsymbol{z}_i^{(0)})$. In Appendix 5.6.1, we show how to find this result.

5.4.3.c Using the Laplace's method as a proposal distribution

LAPLACE'S METHOD Laplace's method approximates the integral of the form $\int \exp\{h(\boldsymbol{z}_i)\}d\boldsymbol{z}_i$ where $h(\boldsymbol{z}_i)$ is a function satisfying some regularity conditions. In this method, we first compute the mode $\boldsymbol{m}_i$ of $h(\boldsymbol{z}_i)$ and its curvature $\boldsymbol{V}$ at the mode; these quantities are defined as:

$$\boldsymbol{m}_i = \underset{\boldsymbol{z}_i}{\text{argmax}}\ h(\boldsymbol{z}_i) \text{ and} \tag{5.30}$$

$$\boldsymbol{V} = -\left[\frac{\partial^2 h(\boldsymbol{z}_i)}{\partial \boldsymbol{z}_i \partial \boldsymbol{z}_i^T}\right]^{-1}_{\boldsymbol{z}_i=\boldsymbol{m}_i}. \tag{5.31}$$

Then, we take the first order Taylor series expansion of $h(\boldsymbol{z}_i)$ around $\boldsymbol{m}_i$, as shown in Equation (5.32). The second term in the expansion corresponds to a Gaussian whose normalising constant is known, which gives us the approximation of Equation (5.33):

$$\int \exp\{h(\boldsymbol{z}_i)\}d\boldsymbol{z}_i \approx \int \exp\left[h(\boldsymbol{m}_i) - \frac{1}{2}(\boldsymbol{z}_i - \boldsymbol{m}_i)\boldsymbol{V}^{-1}(\boldsymbol{z}_i - \boldsymbol{m}_i)\right]d\boldsymbol{z}_i \tag{5.32}$$

$$= \exp[h(\boldsymbol{m}_i)](|2\pi\boldsymbol{V}|)^{1/2}. \tag{5.33}$$

APPLYING THE LAPLACE'S METHOD TO OUR PROBLEM We define the function as follows:

$$h(\boldsymbol{z}_i^{(0)}) = \log p(\boldsymbol{x}_i, \boldsymbol{z}_i^{(0)}; \boldsymbol{\theta}). \tag{5.34}$$

For LGM, computing $\boldsymbol{m}_i$ is easy since $h(\boldsymbol{z}_i^{(0)})$ is a concave function , and $\boldsymbol{V}_i$ is usually available in closed form. These quantities can be used to approximate the posterior as follows:

$$p(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i;\boldsymbol{\theta}) \propto p(\boldsymbol{x}_i, \boldsymbol{z}_i^{(0)}, \boldsymbol{\theta}) = \exp[h(\boldsymbol{z}_i^{(0)})] \tag{5.35}$$

$$\approx \exp[h(\boldsymbol{m}_i)] \exp\big[-\frac{1}{2}(\boldsymbol{z}_i^{(0)} - \boldsymbol{m}_i)^T \boldsymbol{V}_i^{-1}(\boldsymbol{z}_i^{(0)})\big] \tag{5.36}$$

$$\propto \mathcal{N}(\boldsymbol{z}_i^{(0)}|\boldsymbol{m}_i, \boldsymbol{V}_i). \tag{5.37}$$

5.4.3.d Estimating $\boldsymbol{W}$ and $\boldsymbol{w}$

The parameters $\boldsymbol{W}$ and $\boldsymbol{w}$ also have to be estimated. Since we have:

$$\boldsymbol{\chi}_i = \boldsymbol{W}\boldsymbol{z}_i^{(0)} + \boldsymbol{w} \text{ and } p(\boldsymbol{x}_i = r|\boldsymbol{z}_i;\boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\chi}_{ir})}{\sum_{r'=1}^{m}\exp(\boldsymbol{\chi}_{ir'})},$$

finding the parameters W and w_0 that maximise $p(x = k|z^{(0)})$ is like finding the weights of a softmax function (in other words a neural network with only one neuron).

5.4.4 Remarks on the model

In this section, we presented a solution to extend the DGMM to categorical data. We chose to use Importance Sampling coupled with the Laplace's method as a proposal distribution to compute the expectations we needed; however, other methods of the literatures could have been used such as the Hamiltonian Monte Carlo algorithm (Mohamed et al., 2009) or methods based on Local Variational Bounds (Khan et al., 2010). The DGMM for categorical data entirely relies on the DGMM for continuous data. Because we could not find a way to properly estimate the parameters of the latter, we cannot compare the alternatives of the DGMM for categorical data.

5.5 Conclusion and perspectives

In theory, the DGMM is a powerful model that can estimate complex distributions by stacking MFA layers in a deep learning fashion. However, we saw through different experiments that the EM algorithm, deployed to infer its parameters, suffers from drawbacks and struggles to find the global maxima, due to the many latent variables and the local maxima. Nevertheless, on some real data sets, the DGMM yields satisfying results and offers a nice generative process to visualise the representant of the clusters in the context of image data sets. In this thesis, the DGMM was mostly described in the context of continuous variables but, if the inference drawbacks get solved, it can be easily extended to categorical data. Finally, the model selection aspects were not further investigated but they could also have a great impact on the performance of the model. As in neural networks, it would be interesting to understand the differences between shallow and deep networks and be able to build tools to choose architectures less arbitrarily.

5.6 Appendices

5.6.1 Importance sampling and unnormalised distributions

Here, we prove how to find the result of Equation (5.29). From the first part of this equation, we have:

$$\begin{aligned}\mathbb{E}_{p(\boldsymbol{z}^{(0)}|\boldsymbol{x},\boldsymbol{v})}[\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i,\boldsymbol{v}_i] &= \frac{C_q}{C_p}\int \boldsymbol{z}_i^{(0)}\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i;\boldsymbol{\theta})}{\widetilde{q}(\boldsymbol{z}_i^{(0)})}q(\boldsymbol{z}_i^{(0)})d\boldsymbol{z}_i^{(0)} \\ &\approx \frac{C_q}{C_p}\frac{1}{S}\sum_{s=1}^{S}\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)^{[s]}}|\boldsymbol{x}_i)}{\widetilde{q}(\boldsymbol{z}_i^{(0)^{[s]}})}\boldsymbol{z}_i^{(0)^{[s]}}.\end{aligned} \tag{5.38}$$

We can use the same sample set to estimate the ratio $\frac{C_q}{C_p}$ with the result:

$$\begin{aligned}\frac{C_p}{C_q} &= \frac{C_p}{C_q}\underbrace{\int\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i;\boldsymbol{\theta})}{C_p}d\boldsymbol{z}_i^{(0)}}_{=1} \\ &= \frac{1}{C_q}\int\frac{C_q\widetilde{p}(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i;\boldsymbol{\theta})}{\widetilde{q}(\boldsymbol{z}_i^{(0)})}q(\boldsymbol{z}_i^{(0)})d\boldsymbol{z}_i^{(0)} \\ &= \int\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i;\boldsymbol{\theta})}{\widetilde{q}(\boldsymbol{z}_i^{(0)})}q(\boldsymbol{z}_i^{(0)})d\boldsymbol{z}_i^{(0)} \\ &\approx \sum_{s=1}^{S}\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)^{[s]}}|\boldsymbol{x}_i)}{\widetilde{q}(\boldsymbol{z}_i^{(0)^{[s]}})}.\end{aligned}$$

Therefore, we get:

$$\mathbb{E}_{p(\boldsymbol{z}^{(0)}|\boldsymbol{x},\boldsymbol{v})}[\boldsymbol{z}_i^{(0)}|\boldsymbol{x}_i,\boldsymbol{v}_i] \approx \sum_{s=1}^{S}\frac{\widetilde{p}(\boldsymbol{z}_i^{(0)^{[s]}}|\boldsymbol{x}_i)/\widetilde{q}(\boldsymbol{z}_i^{(0)^{[s]}})}{\sum_{s'=1}^{S}\widetilde{p}(\boldsymbol{z}_i^{(0)^{[s']}}|\boldsymbol{x}_i)/\widetilde{q}(\boldsymbol{z}_i^{(0)^{[s']}})}\boldsymbol{z}_i^{(0)^{[s]}}. \tag{5.39}$$

5.6.2 Tables of correspondences between scripts and sections

script name	section number
`script-1.R`	Sections 5.3.1.a and 5.3.1.b
`script-2.R`	Section 5.3.1.b
`script-3.R`	Section 5.3.2.a
`script-4.R`	Section 5.3.2.b
`script-5.R`	Section 5.3.2.c
`script-6.R`	Section 5.3.2.d
`script-7.R`	Section 5.3.2.e
`script-8.R`	Section 5.3.2.f
`script-9.R`	Section 5.3.2.g
`script-10.R`	Section 5.3.2.h and Section 5.3.2.i
`script-11.R`	Section 5.3.3.a

Table 5.7 – Scripts and their corresponding sections.

section number	script name
Section 5.3.1.a	`script-1.R`
Section 5.3.1.b	`script-1.R` and `script-2.R`
Section 5.3.2.a	`script-3.R`
Section 5.3.2.b	`script-4.R`
Section 5.3.2.c	`script-5.R`
Section 5.3.2.d	`script-6.R`
Section 5.3.2.e	`script-7.R`
Section 5.3.2.f	`script-8.R`
Section 5.3.2.g	`script-9.R`
Section 5.3.2.h	`script-10.R`
Section 5.3.2.i	`script-10.R`
Section 5.3.3.a	`script-11.R`

Table 5.8 – Sections and their corresponding scripts.

6

Conclusion and Perspectives

6.1 Conclusion

This thesis presented several contibutions to bring parsimony to model-based clustering methods. In Chapter 2, we introduced model-based clustering and some of its most famous techniques so that the reader gets the necessary background for the next chapters. We first described the Finite Mixture Model as the basis of all the model-based methods. Then, we explained in detail the Mixture of Factor Analysers, which are the basis of Chapter 5. Finally, we reviewed the Latent Block Model, a probabilistic approach of co-clustering, which is the basis model for the contributions of Chapters 3 and 4.

Our first contribution is detailed in Chapter 3. It defines a co-clustering approach that is able to take mixed data sets into account. It extends the Latent Block Model and more particularly the Multiple Latent Block Model (MLBM) (Robert, 2017) to separate column-clusters of variables of different nature. We detailed the SEM-algorithm to infer the parameters of the model and we showed that this method can be of help to analyse different real data sets.

When we used the MLBM on real data sets, we handled textual data via document-term matrices, which are sparse and high-dimensional. We observed that interpreting the LBM and the MLBM was not always easy, which made us design the SOCC model that we described in Chapter 4. The SOCC model adapts to the complexity of textual data by

defining and organising noise blocks. Once the co-clustering is run, the user already knows what the noisy blocks are, hence the user knows which blocks contain specific information.

Chapter 5 is a thorough investigation of the Deep Gaussian Mixture Model (Deep GMM) (Viroli and McLachlan, 2019) and its properties. This model is based on the Mixture of Factor Analysis (MFA) model and consists in stacking MFA layers in a deep learning fashion. This is made possible by considering that the latent scores of a layer are the data input of the MFA of the next layer. In this chapter, we empirically show the difficulties to properly estimate the parameters of the models and we discuss the possible reasons and solutions to tackle these problems.

6.2 Perspectives of the MLBM

In Chapter 3, we presented the MLBM model for mixed data, which is a co-clustering model able to take heterogeneous data into account. As detailed in Section 3.7, this model has certain limitations. One of the main flaws is that each kind of variable will have a different impact on the resulting row partitions even if the D matrices have the same number of features J_d. Works such as Wang (2001) could be a good avenue to start investigating the possibility of affiliating weights to the different matrices $(\boldsymbol{x}^d)_{d\in\{1,\dots,D\}}$.

Finally, in Section 3.6, we saw that the need of multiple latent blocks is not necessarily due to the data heterogeneity and that it can be semantic (i.e. the user may need certain variables not to be into the same column-cluster). We can imagine that some applications could need a similar separation for the rows too. In other words, the user could need similar constraints where certain rows cannot be together, resulting in D sets of variables (as in MLBM), and E sets of observations (the extension). Imposing such a constraint is quite straightforward, but it makes the model selection more complicated since there would not be 1 but E numbers of row-clusters to define.

6.3 Perspectives of the SOCC model

In Chapter 4, we defined the SOCC model, a co-clustering algorithm for textual data sets. We saw that the main perspective of the model is to define other more flexible structures where some column-clusters could be removed when they are useless. This could be done using the ICL but would cost a lot of time of computation.

One of the most important advantages of this model is the interpretability of its results. For years, the Machine Learning community has tried to improve the performance of their techniques and has reached State-of-the-Art accuracies with "black box" frameworks such as neural networks that are hard to explain. However, recently, a part of the community has shown the importance of explainability and interpretability in ML (Gilpin et al., 2018). Many domains such as healthcare, insurances, banks and so on, look for deploying ML systems; so, if we are unable to deploy improved interpretability in our algorithms, the potential impact of ML will be limited. Therefore, the perspectives of this chapter are not only about the SOCC model itself but also about the need of designing models with better interpretability and explainability.

6.4 Perspectives of the DGMM

In Chapter 5, we investigated the properties of the DGMM (Viroli and McLachlan, 2019). We saw that the estimation of the parameters of the DGMM is difficult, and the first perspective of this chapter would be to find a robust and reliable algorithm to find correct estimators. We also saw that when this problem is solved, the DGMM for categorical and mixed data could be investigated so that it is able to take into account more complex data. In addition, and still considering that inference problems are solved, the fact that DGMM is both a neural network and a probabilistic model offers many avenues of investigation.

6.4.1 From a probabilistic point of view

The DGMM is based on the MFA model, which is itself based on latent variables $\boldsymbol{z}$, assumed to be drawn from the standard multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. In fact, many probabilistic models rely on such latent variables, we can cite, among others, the Probabilistic Partial Least Squares (el Bouhaddani et al., 2018) and the Probabilistic Canonical Correlation Analysis (Bach and Jordan, 2005). Such models could possibly be thought in a deep learning fashion too, and we could investigate the feasibility of a multi-layer architecture to describe the distribution of their latent variables.

6.4.2 From a deep learning point of view

Because the DGMM is somehow a neural network, an interesting perspective would be to transfer some of the neural networks extensions that were recently found to the model. The DGMM architecture, for example, can be designed with layers that are not fully-connected in order to resemble more to a Convolutional Neural Network. This could possibly improve its performance with image data sets. The dropout technique could also be implemented in the DGMM context; removing a neuron at layer l would be the equivalent to removing a cluster at layer l. The ResNets architecture could also be an inspiration for an extension of the DGMM, although it would require to investigate how to handle the dimension reduction of the scores $\boldsymbol{z}$, which is not straightforward.

Bibliography

N. K. Aaronson, S. Ahmedzai, B. Bergman, M. Bullinger, A. Cull, N. J. Duez, A. Filiberti, H. Flechtner, S. B. Fleishman, J. C. J. M. d. Haes, S. Kaasa, M. Klee, D. Osoba, D. Razavi, P. B. Rofe, S. Schraub, K. Sneeuw, M. Sullivan, and F. Takeda. The european organization for research and treatment of cancer qlq-c30: A quality-of-life instrument for use in international clinical trials in oncology. *JNCI: Journal of the National Cancer Institute*, 85(5):365–376, 1993. doi: 10.1093/jnci/85.5.365.

A. Agresti. *Analysis of Ordinal Categorical Data, 2nd Ed.* John Wiley & Sons, Inc., 2010.

O. Aguilar and M. West. Bayesian dynamic factor models and portfolio allocation. *Journal of Business & Economic Statistics*, 18(3):338–357, 2000.

M. Ailem, F. Role, and M. Nadif. Graph modularity maximization as an effective method for co-clustering text data. *Know.-Based Syst.*, 109(C):160–173, Oct. 2016.

M. Ailem, F. Role, and M. Nadif. Sparse poisson latent block model for document clustering. *IEEE Trans. Knowl. Data Eng.*, 29(7):1563–1576, 2017.

F. R. Bach and M. I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical report, 2005.

J. Baek, G. Mclachlan, and L. Flack. Mixtures of factor analyzers with common factor loadings: Applications to the clustering and visualization of high-dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 32:1298–309, 07 2010. doi: 10.1109/TPAMI.2009.149.

J. D. Banfield and A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49(3):803–821, 1993.

J.-P. Baudry, A. E. Raftery, G. Celeux, K. Lo, and R. Gottardo. Combining mixture components for clustering. *Journal of Computational and Graphical Statistics*, 19(2): 332–353, 2010.

R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

T. Benaglia, D. Chauveau, and D. R. Hunter. An em-like algorithm for semi- and non-parametric estimation in multivariate mixtures. *Journal of Computational and Graphical Statistics*, 18(2):505–526, 2009.

C. Biernacki and J. Jacques. Model-Based Clustering of Multivariate Ordinal Data Relying on a Stochastic Binary Search Algorithm. *Statistics and Computing*, 26(5):929–943, 2016.

C. Biernacki and A. Lourme. Unifying data units and models in (co-)clustering. *Adv. Data Anal. Classif.*, 13(1):7–31, Mar. 2019. ISSN 1862-5347.

C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 22(7):719–725, July 2000.

C. Biernacki, T. Deregnaucourt, and V. Kubicki. Model-based clustering with mixed/missing data using the new software MixtComp. In *CMStatistics 2015 (ERCIM 2015)*, London, United Kingdom, Dec. 2015.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

A. Bouchareb, M. Boullé, and F. Rossi. Co-clustering de données mixtes à base des modèles de mélange. In *Actes de la 17ème Conférence Internationale Francophone sur l'Extraction et gestion des connaissances (EGC'2017)*, pages 141–152, Grenoble, France, 2017.

C. Bouveyron, S. Girard, and C. Schmid. High-Dimensional Data Clustering. *Computational Statistics and Data Analysis*, 52(1):502–519, 2007.

C. Bouveyron, M. Fauvel, and S. Girard. Kernel discriminant analysis and clustering with parsimonious gaussian process models. *Statistics and Computing*, 25(6):1143–1162, 2015. ISSN 0960-3174.

C. Bouveyron, L. Bozzi, J. Jacques, and F. Jollois. The functional latent block model for the co-clustering of electricity consumption curves. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(4):897–915, 2018.

C. Bouveyron, G. Celeux, T. B. Murphy, and A. E. Raftery. *Model-Based Clustering and Classification for Data Science: With Applications in R.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

V. Brault. *Estimation et sélection de modèle pour le modèle des blocs latents.* PhD thesis, Université Paris Sud-Paris XI, 2014.

N. D. Buono and G. Pio. Non-negative matrix tri-factorization for co-clustering: An analysis of the block matrix. *Information Sciences*, 301:13 – 26, 2015. ISSN 0020-0255.

G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781 – 793, 1995. ISSN 0031-3203.

O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning.* The MIT Press, 1st edition, 2010. ISBN 0262514125.

Y. Chen, L. Wang, M. Dong, and J. Hua. Exemplar-based visualization of large document corpus (infovis2009-1115). *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1161–1168, Nov. 2009.

P. Coretto and C. Hennig. Maximum likelihood estimation of heterogeneous mixtures of gaussian and uniform distributions. *Journal of Statistical Planning and Inference*, 141(1): 462 – 473, 2011. ISSN 0378-3758.

F. Cousson-Gélie. Évolution du contrôle religieux la première année suivant l'annonce d'un cancer du sein : quels liens avec les stratégies de coping, l'anxiété, la dépression et la qualité de vie ? *Psychologie Francaise*, 59(4):331 – 341, 2014. ISSN 0033-2984.

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

I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 89–98, New York, NY, USA, 2003. ACM.

C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 126–135, New York, NY, USA, 2006. ACM.

A. R. T. Donders, G. J. van der Heijden, T. Stijnen, and K. G. Moons. Review: A gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, 59(10): 1087 – 1091, 2006.

G. Drosatos and E. Kaldoudi. A probabilistic semantic analysis of ehealth scientific literature. *Journal of Telemedicine and Telecare*, 2019.

S. el Bouhaddani, H.-W. Uh, C. Hayward, G. Jongbloed, and J. Houwing-Duistermaat. Probabilistic partial least squares model: Identifiability, estimation and application. *Journal of Multivariate Analysis*, 167(C):331–346, 2018.

B. S. Everitt. *Introduction to Latent Variable Models*. Chapman and Hall. 1984.

D. Fernández and R. Arnold. Model selection for mixture-based clustering for ordinal data. *Australian & New Zealand Journal of Statistics*, 58, 08 2016. doi: 10.1111/anzs.12179.

J. Fonseca and M. Cardoso. Mixture-model cluster analysis using information theoretical criteria. *Intell. Data Anal.*, 11:155–173, 04 2007. doi: 10.3233/IDA-2007-11204.

M. Fop and T. B. Murphy. Variable selection methods for model-based clustering. *Statist. Surv.*, 12:18–65, 2018. doi: 10.1214/18-SS119.

R. Gaujoux and C. Seoighe. A flexible r package for nonnegative matrix factorization. *BMC Bioinformatics*, 11(1):367, 2010.

A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

A. Gelman and D. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992. ISSN 08834237.

J. Geweke and G. Zhou. Measuring the Pricing Error of the Arbitrage Pricing Theory. CEMA Working Papers 276, China Economics and Management Academy, Central University of Finance and Economics, 1996.

Z. Ghahramani and G. E. Hinton. The em algorithm for mixtures of factor analyzers. 1997.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, 2018.

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

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618.

G. Govaert and M. Nadif. Latent block model for contingency table. *Communications in Statistics - Theory and Methods*, 39(3):416–425, 2010.

G. Govaert and M. Nadif. *Co-Clustering*. Computing Engineering series. ISTE-Wiley, 2013.

G. Govaert and M. Nadif. Mutual information, phi-squared and model-based co-clustering for contingency tables. *Advances in Data Analysis and Classification*, 12(3):455–488, Sep 2018.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

K. Hornik, I. Feinerer, M. Kober, and C. Buchta. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22, 2012.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec 1985.

K. Humphreys and D. M. Titterington. Approximate bayesian inference for simple mixtures. In J. G. Bethlehem and P. G. M. van der Heijden, editors, *COMPSTAT*, pages 331–336, Heidelberg, 2000. Physica-Verlag HD. ISBN 978-3-642-57678-2.

J. E. Jackson. *Procrustes Rotation*. American Cancer Society, 2005. ISBN 9780470011812. doi: 10.1002/0470011815.b2a13072.

J. Jacques and C. Biernacki. Model-based co-clustering for ordinal data. *Computational Statistics & Data Analysis*, 123(C):101–115, 2018.

K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

H. F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200, 1958. doi: 10.1007/BF02289233.

G. Karypis. CLUTO a clustering toolkit. Technical Report 02-017, Dept. of Computer Science, University of Minnesota, 2002.

C. Keribin. Les méthodes bayésiennes variationnelles et leur application en neuroimagerie : une étude de l'existant. Research Report RR-7091, INRIA, 2009.

C. Keribin, G. Govaert, and G. Celeux. Estimation d'un modèle àblocs latent par l'algorithme sem. *42èmes Journées de Statistique*, 05 2010.

C. Keribin, V. Brault, G. Celeux, and G. Govaert. Estimation and Selection for the Latent Block Model on Categorical Data. Research Report RR-8264, INRIA, Nov. 2013.

M. E. E. Khan, G. Bouchard, K. P. Murphy, and B. M. Marlin. Variational bounds for mixed-data factor analysis. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1108–1116. Curran Associates, Inc., 2010.

D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. cite arxiv:1312.6114.

C. Laclau and M. Nadif. Hard and fuzzy diagonal co-clustering for document-term partitioning. *Neurocomput.*, 193(C):133–147, June 2016.

C. Laclau and M. Nadif. Diagonal latent block model for binary data. *Statistics and Computing*, 27(5):1145–1163, 2017.

C. Laclau, I. Redko, B. Matei, Y. Bennani, and V. Brault. Co-clustering through optimal transport. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia*, pages 1955–1964, 2017.

P. Latouche, E. Birmelé, and C. Ambroise. Bayesian methods for graph clustering. In A. Fink, B. Lausen, W. Seidel, and A. Ultsch, editors, *Advances in Data Analysis, Data Handling and Business Intelligence*, pages 229–239, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

E. Lebarbier and T. Mary-Huard. Le critère BIC : fondements théoriques et interprétation. Research Report RR-5315, INRIA, 2004.

Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

T. Lin, J. Lee, and S. Yen. Finite mixture modelling using the skew normal distribution. *Statistica Sinica*, 17:909–927, 07 2007.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., New York, NY, USA, 1986. ISBN 0-471-80254-9.

G. Lubke and B. Muthén. Applying multigroup confirmatory factor models for continuous outcomes to likert scale data complicates meaningful group comparisons. *Structural Equation Modeling: A Multidisciplinary Journal*, 11(4):514–534, 2004.

E. E. MaloneBeach and S. H. Zarit. Dimensions of social support and social conflict as predictors of caregiver depression. *International Psychogeriatrics*, 7(1):25–38, 1995.

M. V. Mantyla, M. Claes, and U. Farooq. Measuring lda topic stability from clusters of replicated runs. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450358231.

M. Marbac, C. Biernacki, and V. Vandewalle. Model-based clustering of gaussian copulas for mixed data. *Communications in Statistics - Theory and Methods*, 46(23), 2017.

M. Mariadassou and C. Matias. Convergence of the groups posterior distribution in latent or stochastic block models. *Bernoulli*, 21(1):537–573, 02 2015. doi: 10.3150/13-BEJ579.

G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley New York, 1997. ISBN 0471123587.

G. J. McLachlan and D. Peel. *Finite mixture models*. Wiley Series in Probability and Statistics, New York, 2000.

P. D. McNicholas and T. B. Murphy. Parsimonious gaussian mixture models. *Statistics and Computing*, 18(3):285–296, 2008.

D. McParland and I. Gormley. Model based clustering for mixed data: Clustmd. *Adv. Data Anal. Classif.*, 10(2):155–169, June 2016. ISSN 1862-5347.

D. McParland, C. M. Phillips, L. Brennan, H. M. Roche, and I. C. Gormley. Clustering high-dimensional mixed data to uncover sub-phenotypes: joint analysis of phenotypic and genotypic data. *Statistics in Medicine*, 36(28):4548–4569, 2017.

S. Mohamed, Z. Ghahramani, and K. A. Heller. Bayesian exponential family pca. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1089–1096. Curran Associates, Inc., 2009.

M. Nadif and G. Govaert. Algorithms for model-based block gaussian clustering. In *DMIN'08, the 2008 International Conference on Data Mining*, Las Vegas, Nevada, USA, July 14-17 2008.

J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 06 1994.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

K. Pearson and O. M. F. E. Henrici. Iii. contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. (A.)*, 185:71–110, 1894.

D. Peel and G. J. McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10(4):339–348, 2000.

G. R. Pierce, I. G. Sarason, B. R. Sarason, J. A. Solky-Butzel, and L. C. Nagle. Assessing the quality of personal relationships. *Journal of Social and Personal Relationships*, 14(3): 339–356, 1997.

A. E. Raftery. Bayesian model selection in social research. *Sociological Methodology*, 25: 111–163, 1995.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

E. Richardson and Y. Weiss. On gans and gmms. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5852–5863, Red Hook, NY, USA, 2018. Curran Associates Inc.

V. Robert. *Classification croisee pour l'analyse de bases de donnees de grandes dimensions de pharmacovigilance*. PhD thesis, Université Paris-Sud, 2017.

J. K. Rowling. *Harry Potter and the Philosopher's Stone*, volume 1. Bloomsbury Publishing, London, 1 edition, June 1997.

J. K. Rowling. *Harry Potter and the Chamber of Secrets*, volume 1. Bloomsbury Publishing, London, 1 edition, June 1998.

J. K. Rowling. *Harry Potter and the Prisoner of Azkaban*, volume 1. Bloomsbury Publishing, London, 1 edition, June 1999.

D. B. Rubin and D. T. Thayer. Em algorithms for ml factor analysis. *Psychometrika*, 47 (1):69–76, 1982.

A. Salah, M. Ailem, and M. Nadif. Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering. In *Proceedings of the Thirty-Second International Conference on Artificial Intelligence (AAAI'18)*, 2018.

I. G. Sarason, H. M. Levine, R. B. Basham, and B. R. Sarason. Assessing social support: The social support questionnaire. *Journal of Personality and Social Psychology*, page 139, 1983.

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

L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):205–233, 2016. URL https://journal.r-project.org/archive/2016-1/scrucca-fop-murphy-etal.pdf.

M. Selosse, J. Jacques, C. Biernacki, and F. Cousson-Gélie. Analysing a quality-of-life survey by using a coclustering model for ordinal data and some dynamic implications. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, in press, 2019.

M. Selosse, C. Gormley, J. Jacques, and C. Biernacki. A bumpy journey: exploring deep gaussian mixture models. In *NeurIPS 2020 Workshop ICBINB*, 2020a.

M. Selosse, J. Jacques, and C. Biernacki. Model-based co-clustering for mixed type data. *Computational Statistics & Data Analysis*, 144:106866, 2020b. ISSN 0167-9473.

M. Selosse, J. Jacques, and C. Biernacki. Textual data summarization using the self-organized co-clustering model. *Pattern Recognition*, 103:107315, 2020c. ISSN 0031-3203.

P. Singh Bhatia, S. Iovleff, and G. Govaert. blockcluster: An R package for model-based co-clustering. *Journal of Statistical Software*, 76(9):1–24, 2017.

Y. B. Slimen, S. Allio, and J. Jacques. Model-based co-clustering for functional data. *Neurocomputing*, 291:97 – 108, 2018. ISSN 0925-2312.

M. Śmieja, M. Przewięźlikowski, and L. Struski. Estimating conditional density of missing values using deep gaussian mixture model. 2020.

A. K. Smilde, J. A. Westerhuis, and S. d. Jong. A framework for sequential multiblock component methods. *Journal of Chemometrics*, 17(6):323–337, 6 2003. ISSN 1099-128X.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1(804): 801, 1956.

Y. Tang, R. Salakhutdinov, and G. Hinton. Deep mixtures of factor analysers. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1123–1130, USA, 2012. Omnipress. ISBN 978-1-4503-1285-1. URL http://dl.acm.org/citation.cfm?id=3042573.3042718.

T. Thongtan and T. Phienthrakul. Sentiment classification using document embeddings trained with cosine similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 407–414, Florence, Italy, July 2019. Association for Computational Linguistics.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 61(3):611–622, 1999a.

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Comput.*, 11(2):443–482, 1999b. ISSN 0899-7667.

A. van den Oord and B. Schrauwen. Factoring variations in natural images with deep gaussian mixture models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3518–3526. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5227-factoring-variations-in-natural-images-with-deep-gaussian-mixture-models.pdf.

C. Viroli and G. J. McLachlan. Deep gaussian mixture models. *Statistics and Computing*, 29(1):43–51, Jan 2019.

B. Wang and D. M. Titterington. Convergence and asymptotic normality of variational bayesian approximations for exponential family models with missing values. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, page 577–584, Arlington, Virginia, USA, 2004. AUAI Press. ISBN 0974903906.

S. X. Wang. *Maximum weighted likelihood estimation*. PhD thesis, University of British Columbia, 2001.

L. Wu, I. E.-H. Yen, K. Xu, F. Xu, A. Balakrishnan, P.-Y. Chen, P. Ravikumar, and M. J. Witbrock. Word mover's embedding: From Word2Vec to document embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*,

pages 4524–4534, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

X. Yan, J. Guo, Y. Lan, and X. Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 1445–1456, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320351.

X. Yang, K. Huang, and R. Zhang. Deep mixtures of factor analyzers with common loadings: A novel deep generative approach to clustering. In *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part I*, pages 709–719, 2017. doi: 10.1007/978-3-319-70087-8_73. URL https://doi.org/10.1007/978-3-319-70087-8_73.

J. Zhao and P. Yu. Fast ml estimation for the mixture of factor analyzers via an ecm algorithm. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 19:1956–61, 12 2008. doi: 10.1109/TNN.2008.2003467.

Q. Zhu, Z. Feng, and X. Li. GraphBTM: Graph enhanced autoencoded variational inference for biterm topic model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4663–4672, Brussels, Belgium, 2018. Association for Computational Linguistics.

A. S. Zigmond and R. P. Snaith. The hospital anxiety and depression scale. *Acta Psychiatrica Scandinavica*, 67(6):361–370, 1983.

List of Figures

List of Tables

Résumé long : Parcimonie dans les modèles probabilistes pour l'analyse de données complexes

Apprentissage automatique

Ces dernières années, l'apprentissage automatique (aussi appelé "machine learning" en anglais) a reçu beaucoup d'intérêt de la part de la communauté scientifique et du grand publique. "Intelligence artificielle", "apprentissage statistique", "science des données", sont tous des termes qui représentent une branche de l'apprentissage automatique ou qui y sont fortement liés.

On considère que les modèles d'apprentissage automatique apprennent à partir des données puisque leur comportement dépend des échantillons de données qui ont été introduits dans le programme en entrée. En outre, ces algorithmes peuvent être utilisés sur différents ensembles de données pour résoudre différents problèmes et c'est la raison pour laquelle nous les considérons comme intelligents. L'intérêt croissant pour l'apprentissage automatique est dû à deux facteurs. Premièrement, la production d'informations numériques a fortement augmenté ces dernières années, les entreprises et institutions privées ont désormais davantage accès à des flux de données massifs via les réseaux sociaux, les smartphones, les sites web et les plateformes d'achat. Deuxièmement, ces données n'auraient pas pu être stockées, prétraitées ou analysées sans l'énorme croissance de la puissance de calcul, qui permet de concevoir des modèles plus complexes et plus puissants.

Trois familles de paradigmes

Il existe plusieurs paradigmes en apprentissage automatique, qui utilisent les données de manière différentes, et qui réalisent différents type de tâches. Nous décrivons maintenant les trois grandes familles de paradigmes de l'apprentissage automatique.

Apprentissage supervisé

En apprentissage supervisé, nous avons deux ensembles de variables. Les variables d'entrée $\boldsymbol{x}_i$, et les variables labellisées $\boldsymbol{y}_i$. Le but est d'apprendre une application f de $\boldsymbol{x}_i$ vers $\boldsymbol{y}_i$, avec le jeu de données fait de paires $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in\{1,\dots,N\}}$. En notant $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \boldsymbol{\theta})$ la prédiction du modèle pour $\boldsymbol{y}_i$ sachant les paramètres $\boldsymbol{\theta}$, alors la fonction de perte $\mathcal{L}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)$ définit à quel point les prédicitions du modèle sont précises. Les paramètres $\boldsymbol{\theta}$ sont choisis pour minimiser cette fonction de perte sur un jeu de données avec les échantillons $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in\{1,\dots,N\}}$ donnés :

$$\sum_i^N \mathcal{L}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = \sum_i^N \mathcal{L}(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta})).$$

Le choix de fonction de perte dépend du problème à résoudre et de la nature de $\boldsymbol{x}_i$ et $\boldsymbol{y}_i$. de nombreux algorithmes supervisés existent déjà, tels que la régression linéaire, la régression logistique, les arbres de décisions, les machines à vecteur de support. Dernièrement, les algorithmes impliquant les réseaux de neurones atteignent les performances de l'état de l'art pour différentes tâches telles que la vision par ordinateur et la reconnaissance vocale.

Apprentissage non-supervisé

L'apprentissage non-supervisé a un sens plus large et moins bien défini que l'apprentissage supervisé, car il peut servir dans plusieurs cas. Globalement, le rôle de l'apprentissage non-

supervisé est de trouver des structures au sein des données $\boldsymbol{x}$. En général, les variables $\boldsymbol{y}$ ne sont pas fournies, seulement les variables d'entrées $\boldsymbol{x}$ le sont. Les principaux exemples de tâches non supervisées sont l'estimation de densité, la réduction de dimension, l'extraction de caractéristques, la modélisation générative et l'analyse de clusters.

- L'estimation de densité est la construction d'un estimateur de la desnité de probabilité, basé sur les données observées.
- La réduction de dimension consiste à trouver une représentation pour un jeu de données, et ce dans un espace de plus petite dimension.
- Les modèles génératifs consistent à considérer que les observations d'un jeu de données ont été échantillonés selon un processus par la fonction de densité $p(\boldsymbol{x}; \boldsymbol{\theta})$, dont les paramètres $\boldsymbol{\theta}$ doivent être estimés.
- L'analyse de clusters est la tâche qui consiste à regrouper les observations $\boldsymbol{x}_i$ dans différents groupes (ou clusters). Les observations qui se trouvent dans le même groupe sont alors supposées appartenir à une même catégorie.

Apprentissage semi-supervisé

L'apprentissage semi-supervisé est à mi-chemin entre l'apprentissage supervisé et l'apprentissage non-supervisé, dans le sens où tous les labels $\boldsymbol{y}_i$ ne sont pas nécessairement disponibles pour toutes les observations $\boldsymbol{x}_i$. Dans ce cas, l'objectif d'un algorithme semi-supervisé peut être de concevoir un modèle qui utilise les échantillons non-labellisés pour obtenir de meilleures performances en prédictions, en comparaison avec un algorithme qui n'utiliserait que les échantillons labellisés.

Données complexes

La plupart des techniques d'apprentissage automatiques sont très efficaces lorsque le jeu de données est dit "facile", ce qui signifie qu'il est structuré, en petite dimension, et qu'il ne contient pas de valeurs manquantes. Cependant, les jeux de données qui représentent la réalité sont souvent plus compliqués. Les propriétés qui nous font considérer un jeu de donénes comme "complexe" sont :

- La haute dimension, souvent associée à "la malédiction de la dimension", qui concerne les phénomènes qui apparaissent lors de la manipulation de jeux de données avec de nombreuses variables. Le principal problème est que lorsque le nombre de variables augmente, le volume de l'espace augmente si rapidement que les données deviennent sparses. De plus, de nombreux algorithmes ne peuvent pas estimer leurs paramètres lorsque le nombre d'observations N est supéreur au nombre de variables J.
- L'hétérogénéité des données, ou la mixité des données, concerne les données qui ne sont pas de même nature. Par exemple, un simple jeu de données sur les clients d'une entreprise pourrait contenir le statut social (une variable catégorielle), l'âge (une variable de comptage), la taille et le poids du client (des variables continues). Une telle diversité de type de données peut être difficile à modéliser mathématiquement car les valeurs des variables ne font pas parti du même espace. C'est donc difficile de choisir une distribution commune à toutes ces variables.

- La sparsité se réfère aux jeux de données avec peu d'information. Souvent, cela concerne les données qui contiennent une majorité de valeurs nulles. Par exemple, lorsque nous modélisons les intéractions entre utilisateurs d'un réseau social en comptant le nombre de messages qu'ils s'envoient, la matrice résultante est généralement sparses (beaucoup d'utilisateurs ne s'envoient jamais de messages).
- Les valeurs manquantes se réfèrent au fait que, parfois, certains éléments d'un jeu de données n'ont pas de valeur. Par exemple, lorsque l'on analyse un questionnaire auquel des personnes ont répondu, il est très courant d'observer que certaines questions n'ont pas été répondues par certains participants. Cela peut être modélisé de différentes manières, selon si on considère que le participant n'a pas répondu de manière intentionelle ou non.
- Les jeux de données en continu sont ceux dont les données arrivent en flux. L'exemple le plus courant concerne les données venant de capteurs dont les valeurs s'actualisent à différents instants. Ce genre données requièrent des algorithmes spéciaux capables de recevoir de nouvelles données au fil du temps, mais ce sujet ne sera pas abordé dans cette thèse.

Contenu de la thèse

Cette thèse se concentre sur l'apprentissage non-supervisé, et plus spécifiquement sur la parcimonie de modèles de clustering probabilistes dans le cadre de données complexes. Les modèles de clustering probabilistes marient les modèles génératifs et l'analyse de clusters. Ce type de modèle apporte de nombreux avantages tels que l'interprétabilité et la sélection de modèle. Grâce à leur flexibilité, ces techniques ont prouvé leur efficacité dans de nombreux domaines, et sont largement utilisées pour l'analyse de données. Un inconvénient des méthodes de clustering probabilistes classiques est le nombre élevé de paramètres à estimer, ce qui peut ralentir les algorithmes d'inférence et conduire à de mauvais résultats dans le cas de données complexes. Concevoir des modèles plus parcimonieux (c'est à dire avec moins de paramètres) est un moyen efficace de surmonter ce problème. Cette thèse a pour objectif de concevoir de nouvelles approches probabilstes adaptées aux données complexes. Nous nous intéressons à des données en grande dimension, mais aussi à des données hétérogènes, des données avec des valeurs manquantes et des données sparses telles que les données textuelles.

Le Chapitre 2 rappelle les notions nécessaires pour une bonne compréhension des contributions de la thèse. Premièrement, il détaille les aspects mathématiques des modèles de mélange finis, qui sont à la base des approches probabilistes de clustering. Ces notions seront utiles pour tous les autres chapitres de cette thèse. Deuxièmement, ce chapitre décrit l'anayse factorielle, et plus particulièrement le modèle de mélange d'analyse de facteurs, qui est la base du Chapitre 5. Finalement, ce chapitre définit le modèle des blocs latents (LBM), qui est une technique de co-clustering. Le co-clustering est une tâche qui consiste à réaliser le clustering simultané des lignes et des colonnes d'un jeu de données. Ces notions seront utiles pour le Chapitre 3 et pour le Chapitre 4.

Le Chapitre 3 présente une extension du modèle des blocs latents multiples (MLBM) (Robert, 2017) aux données hétérogènes. Ces données sont difficiles à modéliser avecune seule et même distribution car les valeurs des variables ne se trouvent pas dans le même espace. Dans le cas du co-clustering, c'est particulièrement compliqué, car l'agorithme doit regrouper les variables aussi. De plus, il peut sembler contre-intuitif de regrouper des variables de nature différente car l'objectif du clustering est de regrouper des éléments qui ont quelque chose

en commun. L'approche MLBM consiste à étendre le modèle des blocs latents (LBM) pour qu'il soit capable de prendre des données hétérogènes en compte.

Le Chapitre 4 présente le modèle SOCC (Self-Organised Co-Clustering model) pour les données textuelles et plus précisément pour les matrices document-terme. Ces matrices représentent des données textuelles tel que la cellule ($document_i$,$terme_j$) compte combien de fois le terme $terme_j$ a été utiisé dans le document $document_i$. Cette représentation a l'avantage d'être facile à construire et à lire. Cependant, les matrices qui en résultent sont de très haute dimension et extrêmement sparses, ce qui les rend difficiles à exploiter. Le modèle SOCC s'adapte à ces particularités et définit un modèle pour le clustering des termes et des documents qui offre des résultats simples à exploiter.

Le Chapitre 5 investigue le modèle de mélange Gaussian profond (DGMM) (Viroli and McLachlan, 2019) et ses propriétés. Ce modèle consiste à empiler des couches de MFA, ce qui résulte en une architecture imitant les réseaux de neurones. Cela est rendu possible en considérant les scores latents d'une couche comme étant l'entrée du MFA de la couche d'après. Dans ce chapitre, nous montrons empiriquement les difficultés pour estimer les paramètres du modèle, puis nous discutons les raisons possibles et les solutions à ces problèmes.

Mots-Clefs : modèles probabilistes – clustering – modèles de mélange – co-clustering – analyse de facteurs – parcimonie.