Mixture Models for Multidimensional Positive Data Clustering with Applications to Image Categorization and Retrieval

Taoufik Bdiri

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy (Electrical & Computer Engineering) at
Concordia University
Montréal, Québec, Canada

June 2015

© Taoufik Bdiri, 2015

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Taoufik Bdiri**

Entitled: **Mixture Models for Multidimensional Positive Data Clustering with Applications to Image Categorization and Retrieval**

and submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (Electrical & Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Ashutosh Bagchi	Chair
Dr. Olfa Nasraoui	External Examiner
Dr. Lyes Kadem	External to Program
Dr. Anjali Awasthi	Examiner
Dr. Jamal Bentahar	Examiner
Dr. Nizar Bouguila	Thesis Supervisor
Dr. Djemel Ziou	Thesis Co-Supervisor

Approved by _____
Dr. Abdel R. Sebak
Chair of Department or Graduate Program Director

June 22, 2015 _____
Dr. Amir Asif
Dean of Faculty of Engineering & Computer Science

ABSTRACT

Mixture Models for Multidimensional Positive Data Clustering with Applications to Image Categorization and Retrieval

Taoufik Bdiri, Ph.D.
Concordia University, 2015

Model-based approaches have become important tools to model data and infer knowledge. Such approaches are often used for clustering and object recognition which are crucial steps in many applications, including but not limited to, recommendation systems, search engines, cyber security, surveillance and object tracking. Many of these applications have the urgent need to reduce the semantic gap of data representation between the system level and the human being understandable level. Indeed, the low level features extracted to represent a given object can be confusing to machines which cannot differentiate between very similar objects trivially distinguishable by human beings (e.g. apple vs tomato). Such a semantic gap between the system and the user perception for data, makes the modeling process hard to be designed basing on the features space only. Moreover those models should be flexible and updatable when new data are introduced to the system. Thus, apart from estimating the model parameters, the system should be somehow informed how new data should be perceived according to some criteria in order to establish model updates. In this thesis we propose a methodology for data representation using a hierarchical mixture model basing on the inverted Dirichlet and the generalized inverted Dirichlet distributions. The proposed approach allows to model a given object class by a set of components deduced by the system and grouped according to labeled training data representing the human level semantic. We propose an update strategy to the system components that takes into account adjustable metrics representing users perception. We also consider the "page zero" problem in image retrieval systems when a given user does not possess adequate tools and semantics to express what he/she is looking for, while he/she can visually identify it. We propose a statistical framework that enables users to start a search process and interact with the system in order to find their target "mental image". Finally we propose to improve our models by using a variational Bayesian inference to learn generalized inverted Dirichlet mixtures with features selection. The merit of our approaches is evaluated using extensive simulations and real life applications.

To my parents,
Without whom none of my success would have been possible,
I love you endlessly.

ACKNOWLEDGEMENTS

I would never have been able to finish my Ph.D. without the guidance of many people and support from friends and family.

I would like to express my deepest gratitude to my supervisor, Dr. Nizar Bouguila, for his excellent guidance, caring and valuable support which provided me with an excellent environment to do research. It was a pleasure to work with him during all my graduate studies.
I am also thankful to my co-supervisor Dr. Djemel Ziou whose expertise and insightful comments helped me a lot throughout this thesis and taught me many things even beyond the research work.

Many thanks to my committee members, namely Dr. Olfa Nasraoui, Dr. Jamal Bentahar, Dr. Lyes Kadem, Dr. Anjali Awasthi, and Dr. Ashutosh Bagchi. Thank you for accepting to evaluate this thesis, your guidance and comments have significantly helped me to improve my work.

I would also like to thank the Tunisian Government who gave me the opportunity to study in Canada through the national excellence scholarship program, and Quebec government for the FQRNT doctoral scholarship.

I would not forget to thank my friends with whom I have spent many years through which there were many ups and downs. Especially, I would like to thank Dr. Mohamed Al Mashrgy, Dr. Abdulmotaleb Zidan, Dr. Abdulblbaset Alemam, Muhammad Azam Nazir, Basem Alghabashi, Ali Sefidpour, Dr. Ali Mosleh, Dr. Ali Shojaee, Elise Epaillard, Imane Chatri, Dr. Michael Osadebey, and Walid Masoudi. I would not mention by name all my friends outside the university but I would like to thank all of them as well.

Naturally, best of my thanks go to my family. I would never have completed this thesis without the endless support and love of my father Dr. Mohamed Bdiri, and my mother Naima Toumi. None of my success would have been possible without them being beside me. I want to especially thank my father who has always wanted me to do a Ph.D. thesis. I dedicate all this work to him and I hope that he will always be proud of me. I also thank my brother Dr. Hatem Bdiri, and my two sisters Rym Bdiri and Yosra Bdiri for their continuous support and encouragements. Many thanks go to my beloved wife Dr. Feriel Ben Abdallah Bdiri for her support and love, and for being beside me at the most difficult times during this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Data analysis is crucial for any business survival as it represents the basis of data-driven decisions that improve and maintain the quality of a given provided service or production. Indeed, with the increase of calculations power, data analysis is now present in many sectors including but not limited to health care, IT services, drug manufactures and finance. Amongst others, data analysis is strongly applied in modern applications that cover computer vision and object recognition. As a matter of fact, data generation has become a daily routine in many industries and entertainment services, e.g. 300 hours of video are uploaded, every minute, to YouTube that has more than 1 billion users [1]. One challenging crucial aspect in data analysis is clustering, which is defined as the process of assigning observations sharing similar characteristics to subgroups, such that their heterogeneity is minimized within a given subgroup and maximized between the subgroups. Still, data homogeneity is relative as classification processes are context dependent. This fact has triggered the urgent need to design tools capable of analyzing complex and multidimensional data ensuring a smooth interaction with users whose intentions are not necessarily well represented in system-level features' space. Indeed during such interactions, the system is not always aware of the users intentions, and in many cases there is a semantic gap between the users high level modeling of data and the system-level data model. Consider for instance search engines or object recognition tasks, where a user is looking for a specific element among a huge amount of data. The system is usually not aware of what the user is looking for, and naturally not aware of the user's perception toward the different data elements e.g. if a user's intention is to classify objects and creatures, he/she would assign a car to the object class and a lion to the creature class, whereas a different user may put the car and the lion in the same class having another intention like classifying moving and motionless elements. Hence we talk about the user ontological model and intention which differ from one user to another. Many clustering techniques have been explored by researchers [2] especially model-based clustering that has been extensively adopted in order to perform data

analysis in many disciplines such as bioinformatics [3–5], computer vision [6–8], recommending systems [9–11] and social networks [12–14]. Indeed, mixture based approaches are a powerful tool that serve to infer knowledge and abstract the complexity of a huge amount of information. They are a well formed mathematical representation that assumes that data are generated by a set of probabilistic components representing subpopulations and that every observation belongs to one of them. Such an assumption establishes a clustering process basing on posteriors probabilities. Yet, the adoption of model based approaches still under extensive research, and the task of their deployment faces many challenges namely the choice of the appropriate distribution to model data, how to estimate the parameters of the mixture and how to select the model that is appropriate to better model data in terms of number of components. In this thesis, we focus on multidimensional data clustering and classification which are considered as a key component for many real life applications using mixture models. In the following we introduce some notions that constitute the basis of the statistical frameworks that we develop throughout this thesis.

1.1 Generative and Discriminative Learning

In machine learning, generative and discriminative learning are the two major approaches to classify data and establish predictions in terms of assigning it to their appropriate categories or classes. Generative learning tends to model the data using a joint distribution on inputs and outputs, whereas discriminative learning tends to establish a sort of mapping from inputs to outputs using a conditional distribution or a prediction function. Consider for instance a set of observations with features X that we want to classify into classes labeled by a categorical variable Y. A generative classifier models the joint probability $p(X,Y)$, factorized in the form of $p(X|Y)p(Y)$ where $p(X|Y)$ is a data generating process. The parameters of the model are learned by the maximization of the likelihood with respect to $p(X|Y)p(Y)$. A discriminative classifier, models the conditional distribution $p(Y|X)$ of the class labels given the features and learn the model by maximizing the conditional likelihood with respect to $p(Y|X)$. Thus, the generative classifier is termed as sampling paradigm, and the discriminative classifier is termed as diagnostic paradigm [15].

1.2 Model-based Approach : Finite Mixture Models

Finite mixture models are used in generative learning processes and are often used to model data sampled from a population that is suspected to be composed of a finite number of homogenous subpopulations. Each subpopulation is modeled by a probability density, and the whole model is formed by a weighted sum of those densities [16]. Thus, if we consider a random variable X, a

finite mixture model decomposes a probability function $p(X)$ into the sum of M class probability functions. Let $p_i(X)$ denotes the probability representing the i^{th} class probability function. The finite mixture model can be written as :

$$p(X|\Theta) = \sum_{i=1}^{M} \pi_i p_i(X|\theta_i) \tag{1.1}$$

where Θ represents the set of parameters such that $\Theta = \{\pi_1, \theta_1, \pi_2, \theta_2, ..., \pi_M, \theta_M\}$ and $\pi_i > 0$ such that $\sum_{i=1}^{M} \pi_i = 1$. π_i denotes the mixing weight of the i^{th} class and is considered as a prior probability of observing a sample from the class i. If we consider N samples $\mathscr{X} = \{\vec{X}_1, \vec{X}_2, ..., \vec{X}_n\}$ drawn from a probability density function $p(X|\Theta)$ parameterized by the elements of the parameters vector Θ, the joint probability of the finite mixture model is given by :

$$p(\mathscr{X}|\Theta) = \prod_{n=1}^{N} \sum_{i=1}^{M} \pi_i p_i(\vec{X}_n|\theta_i) \tag{1.2}$$

1.3 Parameters Estimation

Most of approaches to estimate parameters of mixture models consider either deterministic or Bayesian techniques [17]. Deterministic techniques often use the EM algorithm which is a popular iterative method that tries to maximize the log-likelihood $\log p(\mathscr{X}|\Theta)$ with respect to Θ [18, 19]. On the other hand, the Bayesian techniques [20] have been proposed to mitigate some drawbacks of the deterministic techniques such as dependency on initialization and the difficulty to deal with multi-modal distributions. Most of the Bayesian methods that have been proposed consider Laplace's approximation [21] or Markov chain Monte Carlo (MCMC) [22]. Still, the Laplace's approximation often makes a too strong assumption considering that the likelihood is not multi-model which is often not the case for finite mixture model, while MCMC is computationally costly. Another emerging Bayesian technique is variational inference [23] which approximates the model posterior distribution by minimizing the Kullback-Leibler divergence between the true posterior and a distribution which represents its lower bound. Variational inference is used in Chapter 5 as an alternative solution to the EM.

1.4 Disribution Fitting and Model Selection

As a matter of fact, using probability is a form of dealing with uncertainty, and the choice of the probability density function that serves to model data can have a significant impact on the accuracy of the model in terms of clustering and classification. Distribution fitting is the procedure of

selecting a distribution that better describes a given dataset generated by some random process. The normal distribution has been extensively adopted for modeling, however in many real-world applications it is not an adequate choice to model non-Gaussian data. Indeed the normal distribution is symmetric which makes it inappropriate to be used to model data skewness. It also has a constant shape that does not depend on the distribution parameters in contrast with heavy-tailed distributions, and it is defined on the entire real axis which makes other bounded or non negative distributions fit more when dealing with bounded or positive data. In this thesis we propose to use the inverted Dirichlet distribution and the generalized inverted Generalized distribution that will be introduced throughout the thesis. As for the model selection, it consists of selecting a statistical model from a set of candidate models, given the data. When using mixture models, it is a crucial issue to choose the optimal number of components that are needed to represent the data. Many techniques have been proposed to establish a model selection, and most of them are based on the likelihood function and some information theoretic criteria such as the minimum message length (MML) [24], Akaike information criterion (AIC) [25], minimum description length (MDL) [26], mixture MDL (MMDL) [27], and LEC [17].

1.5 Contributions

The aim of this thesis is to propose several approaches to model and cluster multidimensional data using mixture models, and explore them to establish image categorization and retrieval. The overall contributions of this thesis are as follows

- ☞ **A statistical framework using hierarchical mixture models alterable according to users context**

  We show that it is not adequate to represent an object class by a single probability distribution, and we build a hierarchical model where every object class is composed of submixtures. The proposed approach gives the possibility to redefine the hierarchy without the need of establishing a new parameters estimation. We develop the statistical model using the inverted Dirichlet mixture models.

- ☞ **A new approach for online learning of mixture models using adjustable model selection criteria**

  We propose a new methodology to update mixture models with the consideration of users perceptions basing on probabilistic metrics that inform the system how it should construct the mixture and establish updates when new data are introduced on line. The proposed

methodology is implemented using the generalized inverted Dirichlet (GID) distribution.

☞ **A statistical framework for mental targets search using mixture models**
We propose a framework using (GID) mixture models and a Bayesian formulation in order to design a search engine for "Mental Images" targets, with the possibility of searching multi-targets without having an explicit query and basing on visual search only.

☞ **Variational Bayesian inference for infinite generalized inverted Dirichlet mixtures with features selection**
We improve the modeling of the GID mixture by considering an infinite nonparametric Bayesian framework namely the Dirichlet process. We use variational Bayesian inference to estimate the mixture parameters and establish an appropriate model selection. We also integrate a "features selection" capability in order to improve the clustering accuracy.

1.6 Thesis Overview

The organization of this thesis is as follows:

- ❑ Chapter 1, contains an introduction to mixture models and an overview over the thesis.
- ❑ In Chapter 2, we introduce a novel hierarchical mixture model where each component is composed of a set of finite probability densities forming a super class mixture. Our proposed model can be viewed as a mixture of mixtures to support multi-level hierarchies where the structure of the hierarchy can be altered according to users' ontological models within costless computational time. The proposed approach is generalized to adopt any probability density function and an algorithm to learn the model is proposed. We adopt the inverted Dirichlet distribution to build the model, and a simulation is performed to validate the proposed approach using synthetic and a real world application namely visual object clustering and recognition. This work is published in Expert Systems with Applications journal [28].
- ❑ In Chapter 3, we propose a new methodology to update a mixture model that takes into account simultaneously users perception and the dynamic nature of real-world data. We implement the proposed approach using the GID distribution. Experiments on synthetic data as well as real data generated from different publicly available image datasets indicates that the proposed approach has merits and provides promising results. This work is under review by Engineering Applications of Artificial Intelligence journal [29].

- ❏ In Chapter 4, we propose a statistical framework that enables users to start a search process and interact with the system in order to find their target “mental image", using visual features only. Our bayesian formulation provides the possibility of searching multi target classes within the same search process. Data are modeled by a GID mixture that also serves to quantify the similarities between images. We run experiments including real users and we present a case study of a search process that gives promising results in terms of number of iterations needed to find the mental target classes within a given dataset. This work is accepted and is to be published in Artificial-Intelligence Applications in Information and Communication Technologies as a book chapter [30].

- ❏ In Chapter 5, we develop a variational Bayesian framework for the infinite GID mixture models that has proven its capability to model complex multidimensional data. We also integrate a "features selection" approach to highlight the features that are most informative in order to construct an appropriate model in terms of clustering accuracy. Experiments on synthetic data as well as real data generated from visual scenes and handwritten digits datasets validate the proposed approach. This work is under review by Applied Intelligence journal [31].

- ❏ Chapter 6, summarizes our contributions and present some potential future works.

Chapter 2

Object Clustering and Recognition Using Multi-Finite Mixtures for Semantic Classes and Hierarchy Modeling

2.1 Introduction

In many real-life application, clustering is considered as a form of knowledge extraction from data [32, 33]. In the previous chapter, we have described one major approach for clustering based on statistics which is model-based clustering using finite mixture models. As mentioned previously, finite mixture models can be defined as a weighted sum of probability distributions where each distribution represents the population of a given subgroup. The authors in [34] traced the use of finite mixture models back to the 1960s and 1970s, citing amongst others, works in [35, 36]. Although their use backs at least as far as 1963, it is only in the recent decades that mixture models applications started to cover many fields including, but not limited to, digital image processing and computer vision [37–39], social networks [40–42], medicine [43–47], and bioinformatics [48–50]. The consideration of mixture models is practical for many applications. In many cases, however, the complexity of the observed data may render the use of one single distribution to represent a given class insufficient for inference. Many techniques have been proposed to select the best number of mixture components that best represents the data. Examples include Bayesian inference criterion (BIC) [51], minimum description length (MDL) [52] and minimum message length (MML) [24] criteria. These criteria are mainly used in unsupervised algorithms where the system handles data modeling without any intervention during the learning process. Still, in many cases,

the system representation does not necessarily fit with a human comprehensible semantic. Consider for instance an object recognition application where the system has to recognize different objects according to the user need. In most cases, basing on the features space only, the system considers classes with important visual similarities (e.g., apple and tomato), as being the classes that should be represented by a single component in the mixture. This is not always the case in real applications, where a human being or an application would need to differentiate between different classes even when they have close visual properties and similarities, thus, we talk here about a semantic meaning of the mixture components as the gap between the system representation and the human representation of data is still high. An eventual solution is to form a hierarchical model based on some ontology as in [53–56] where the data are grouped into clusters and sub clusters (i.e. tree-structured clustering).Yet, this approach still form the model using the visual similarities and groups data according to the system choice. Furthermore, since the model is based on estimating the model's parameters as the algorithm goes deeper in the hierarchy (the distributions' parameters of the children clusters depend on the parameters of the parents clusters), when changing the hierarchical model, a whole new estimation should take place, which increases the computational cost. The user intervention to build a hierarchical mixture has been introduced in [57] which developed the concept of hierarchical visualization where the construction of the hierarchical tree proceeds top-down, and for each level, the user decides on the suitable number of models to fit at the next level down. Indeed this interaction, may serve to have an optimal number of clusters for each level according to the user, but it does not permit the user to define any semantic meaning to the clusters or group the clusters as he/she needs. Moreover, the user cannot define any ontological model to the data, and there is a new estimation of the parameters to be calculated at each level when the model goes deeper in the tree.

In this chapter, we present a novel way to model data and assign a semantic meaning to clusters according to the user needs which can reduce significantly the gap between the system representation and the user level representation. We tackle the challenging problem of object clustering, and recognition of new unseen data in terms of affectation to the appropriate clusters forming the object class. Naturally, the choice of the distribution forming the mixture model is crucial in terms of clustering efficiency and accuracy of the classification of unseen data. Indeed, many works have focused on Gaussian mixture models (GMM) to build their applications such as in [58–61], but recent researches have shown that it is not appropriate to always assume that data follow a normal distribution. For instance, the works in [62–65] have considered the Dirichlet and generalized Dirichlet mixture models, to model proportional data, which have been shown to outperform the GMM. We have developed in a previous work the inverted Dirichlet (ID) mixture model which has better capabilities than the GMM when modeling positive data that occur naturally in many real

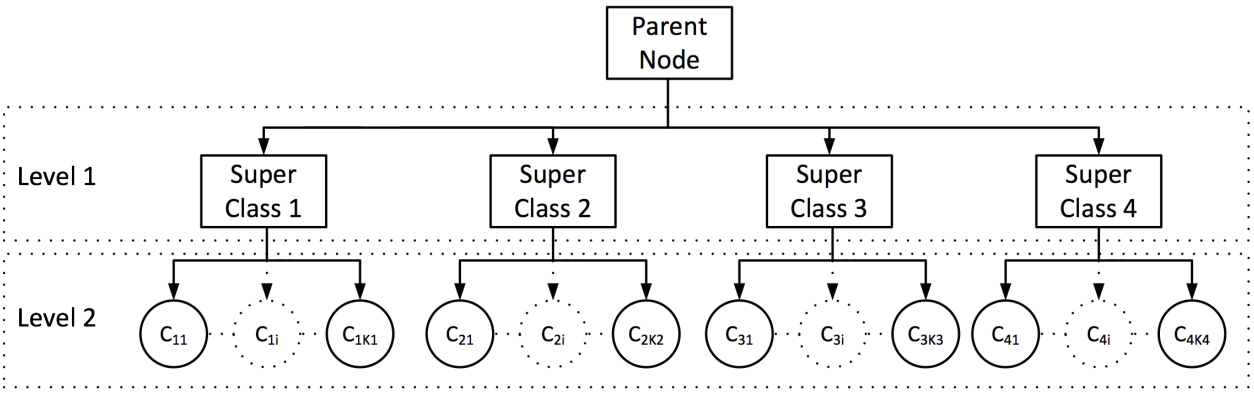


Figure 2.1: Two-levels hierarchical model.

applications [66, 67]. Hence, we propose our new methodology using ID mixture models, although it is noteworthy to bear in mind than any other distribution can be used as the presented framework is general.

This chapter is organized as follows; in section 2.2 we present our statistical framework by considering a two-levels hierarchy for ease of representation and understanding of the general methodology. In section 2.3, we propose a detailed approach to learn the proposed statistical model. In section 2.4, we propose a generalization of our modeling framework to cover many hierarchical levels and finally section 2.5 is devoted to present the experimental results using both synthetic data and a real-life application concerning object recognition. Finally, section 2.6 gives a brief summary of the chapter.

2.2 Statistical Framework : The Model

We propose to develop a statistical framework that can model data in a hierarchical fashion. The attribution of a semantic meaning to the model is discussed in subsection 2.5.2. In this section, for the easiness of presentation, we consider a two-levels hierarchy where we have a set of *super classes*, composed each, of a set of classes. Let us consider a set $\mathscr{X}$ of N D-dimensional vectors, such that $\mathscr{X} = (\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_N)$. Let M denote the number of different *super classes* and K_j the number of classes forming the *super class* j. We assume that $\mathscr{X}$ is controlled by a mixture of mixtures, such that each *super class* j is represented by a mixture of K_j components and the parent mixture is composed of M mixtures representing the *super classes*. Thus, we consider two views or levels for the statistical model. The first view focuses on the *super classes* and the second one zooms on the classes (see Figure 2.1). We suppose that the vectors follow a common but unknown probability density function $p(\vec{X}_n|\Xi)$, where Ξ is the set of its parameters.

2.2.1 First Level

Let $Z = \{\vec{Z}_1, \vec{Z}_2, \ldots, \vec{Z}_N\}$ denote the missing group indicator, where $\vec{Z}_n = (z_{n1}, z_{n2}, \ldots, z_{nM})$ is the label of $\vec{X}_n$, such that $z_{nj} \in \{0,1\}$, $\sum_{j=1}^{M} z_{nj} = 1$ and z_{nj} is equal to one if $\vec{X}_n$ belongs to *super class* j and zero, otherwise. Then, the distribution of $\vec{X}_n$ given the *super class* label $\vec{Z}_n$ is :

$$p(\vec{X}_n|\vec{Z}_n, \Theta) = \prod_{j=1}^{M} p(\vec{X}_n|\theta_j)^{z_{nj}} \tag{2.1}$$

where $\Theta = \{\theta_1, \theta_2, \ldots, \theta_M\}$ and θ_j is the set of parameters of the *super class* j. In practice, $p(\vec{X}_n|\Theta)$ can be obtained by marginalizing the complete likelihood $p(\vec{X}_n, \vec{Z}_n|\Theta)$ over the hidden variables . We define the prior distribution of $\vec{Z}_n$ as follows:

$$p(\vec{Z}_n|\vec{\pi}) = \prod_{j=1}^{M} \pi_j^{z_{nj}} \tag{2.2}$$

where $\vec{\pi} = (\pi_1, \ldots, \pi_M)$, $\pi_j > 0$ and $\sum_{j=1}^{M} \pi_j = 1$, then we have:

$$p(\vec{X}_n, \vec{Z}_n|\Theta, \vec{\pi}) = p(\vec{X}_n|\vec{Z}_n, \Theta)P(\vec{Z}_n|\vec{\pi}) = \prod_{j=1}^{M} (p(\vec{X}_n|\theta_j)\pi_j)^{z_{nj}} \tag{2.3}$$

We proceed by the marginalization of Eq.2.3 over the hidden variable (see Appendix A), so the first level of our mixture for a given vector $\vec{X}_n$ can be written as follows:

$$p(\vec{X}_n|\Theta, \vec{\pi}) = \sum_{j=1}^{M} p(\vec{X}_n|\theta_j)\pi_j \tag{2.4}$$

Thus, the set of parameters Ξ corresponding to the first level is $\Xi = (\Theta, \vec{\pi})$.

2.2.2 Second Level

When we examine the second level which considers the classes, given that $\vec{X}_n$ is generated from the mixture j, we suppose that it is also generated from one of the K_j components of the mixture j. Thus, let $Y_j = \{\vec{Y}_{1j}, \vec{Y}_{2j}, \ldots, \vec{Y}_{Nj}\}$ denote the missing group indicator where the i^{th} element of $\vec{Y}_{nj}$, y_{nji} is equal to one if $\vec{X}_n$ belongs to the class i of the *super class* j and zero, otherwise. Let $\{\vec{Y}_{nj}\} = \{\vec{Y}_{n1}, \vec{Y}_{n2}, ..., \vec{Y}_{nM}\}$ denote the classes label of $\vec{X}_n$. Then, the distribution of $\vec{X}_n$ given the

super class label $\vec{Z}_n$ and the classes label $\{\vec{Y}_{nj}\}$ is

$$p(\vec{X}_n|\vec{Z}_n,\{\vec{Y}_{nj}\},\Theta,\varphi) = p(\vec{X}_n|\vec{Z}_n,\{\vec{Y}_{nj}\},\varphi) = \prod_{j=1}^{M}\left(\prod_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji})^{y_{nji}}\right)^{z_{nj}} \tag{2.5}$$

where $\varphi = \{\vec{\varphi}_{ji}\}$ with $j = 1,\ldots,M$ and $i = 1,\ldots,K_j$, is the set of parameters of the components representing the different classes. It is noteworthy to mention that K_j depends on the number of classes that the *super class* j contains. We define the prior distribution of $\vec{Y}_{nj}$ and $\{\vec{Y}_{nj}\}$ as follows:

$$p(\vec{Y}_{nj}|\vec{Z}_n,\{w_{ji}\},\vec{\pi}) = p(\vec{Y}_{nj}|\vec{Z}_n,\{w_{ji}\}) = \prod_{i=1}^{K_j} w_{ji}^{y_{nji}} \tag{2.6}$$

$$p(\{\vec{Y}_{nj}\}|\vec{Z}_n,\{w_{ji}\},\vec{\pi}) = p(\{\vec{Y}_{nj}\}|\vec{Z}_n,\{w_{ji}\}) = \prod_{j=1}^{M}\left(\prod_{i=1}^{K_j} w_{ji}^{y_{nji}}\right)^{z_{nj}} \tag{2.7}$$

where $w_{ji} > 0$, $\sum_{j=1}^{M}\sum_{i=1}^{K_j} w_{ji} = 1$, and $\{w_{ji}\}$ is the set of the components' mixing weights with $j = 1...M$ and $i = 1...K_j$, then we have :

$$\begin{aligned} p(\vec{X}_n,\vec{Z}_n,\{\vec{Y}_{nj}\}|\varphi,\{w_{ji}\},\vec{\pi}) &= p(\vec{X}_n|\vec{Z}_n,\{\vec{Y}_{nj}\},\varphi,\{w_{ji}\},\vec{\pi})p(\{\vec{Y}_{nj}\}|\vec{Z}_n,\{w_{ji}\})p(\vec{Z}_n|\vec{\pi}) \\ &= \prod_{j=1}^{M}\pi_j^{z_{nj}}\left(\prod_{i=1}^{K_j} w_{ji}^{y_{nji}} p(\vec{X}_n|\vec{\varphi}_{ji})^{y_{nji}}\right)^{z_{nj}} \end{aligned} \tag{2.8}$$

We proceed by the marginalization of Eq.2.8 over the hidden variables (see appendix A), so the second level mixture can be written as :

$$p(\vec{X}_n|\varphi,\{w_{ji}\},\vec{\pi}) = \sum_{j=1}^{M}\pi_j\Big(\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji})w_{ji}\Big) \tag{2.9}$$

Thus, according to the previous equation, the set of parameters Ξ corresponding to the second level is $\Xi = (\varphi,\{w_{ji}\},\vec{\pi})$.

2.3 Model Learning

2.3.1 Log-Likelihood of the Complete Data

The model for the complete data $< \mathscr{X}, Z, \{Y_j\} >$ is given by :

$$p(\mathscr{X}, Z|\Theta, \vec{\pi}) = \underbrace{\prod_{n=1}^{N}\prod_{j=1}^{M} p(\vec{X}_n|\theta_j)^{z_{nj}}\pi_j^{z_{nj}}}_{\text{First level}}$$
$$= \underbrace{\prod_{n=1}^{N}\prod_{j=1}^{M}\left(\prod_{i=1}^{K_j}\left(p(\vec{X}_n|\vec{\varphi}_{ji})w_{ji}\right)^{y_{nji}*z_{nj}}\right)\pi_j^{z_{nj}}}_{\text{Second level}} = p(\mathscr{X}, Z, \{Y_j\}|\varphi, \{w_{ji}\}, \vec{\pi}) \quad (2.10)$$

We maximize the log-likelihood instead of the likelihood. The log-likelihood is given by:

$$\Phi_c(\mathscr{X}, Z, \{Y_j\}|\varphi, \{w_{ji}\}, \vec{\pi}) = \log(p(\mathscr{X}, Z, \{Y_j\}|\varphi, \{w_{ji}\}, \vec{\pi})$$
$$= \sum_{n=1}^{N}\sum_{j=1}^{M}\sum_{i=1}^{K_j} z_{nj}\Big(\log(\pi_j) + y_{nji}\big(\log(w_{ji}) + \log(p(\vec{X}_n|\vec{\varphi}_{ji}))\big)\Big) \quad (2.11)$$

Let us recall that $\varphi = \{\vec{\varphi}_{ji}\}$ with $j = 1, \ldots, M$ and $i = 1, \ldots, K_j$, such that $\vec{\varphi}_{ji}$ is the set of parameters of the inverted Dirichlet distribution, for the *class* i, of the *super class* j. In order to estimate the parameters, we use the expectation maximization (EM) algorithm which proceeds iteratively in two steps; the expectation (E) step and the maximization (M) step. In the E-step, we compute the conditional expectation of $\Phi_c(\mathscr{X}, Z, \{Y_j\}|\varphi, \{w_{ji}\}, \vec{\pi})$ which is reduced to the computation of the posterior probabilities (e.g. the probability that a vector $\vec{X}_n$ is assigned to a mixture j, and the probability that $\vec{X}_n$ is assigned to the component i of j) , such that (see Appendix A):

$$p(j, i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) = \frac{w_{ji}\pi_j p(\vec{X}_n|\vec{\varphi}_{ji})}{\sum_{j=1}^{M}\sum_{i=1}^{K_j} w_{ji}\pi_j p(\vec{X}_n|\vec{\varphi}_{ji})} \quad (2.12)$$

and

$$p(j|\vec{X}_n, \Theta, \vec{\pi}) = \frac{\pi_j p(\vec{X}_n|\theta_j)}{\sum_{j=1}^{M}\pi_j p(\vec{X}_n|\theta_j)} = \frac{\pi_j \sum_{i=1}^{K_j} w_{ji}p(\vec{X}_n|\vec{\varphi}_{ji})}{\sum_{j=1}^{M}\pi_j\sum_{i=1}^{K_j} w_{ji}p(\vec{X}_n|\vec{\varphi}_{ji})} = p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \quad (2.13)$$

Thus, we have :

$$\log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi}) = \sum_{n=1}^{N}\sum_{j=1}^{M}\sum_{i=1}^{K_j} p(j|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi}) \times \left(\log(\pi_j) + p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})\big(\log(w_{ji}) + \log(p(\vec{X}_n|\vec{\varphi}_{ji}))\big)\right) \quad (2.14)$$

Then, the conditional expectation of the complete-data log likelihood, using Lagrange multipliers to introduce the constraints about the mixture proportions $\{\pi_j\}$ and $\{w_{ij}\}$, is given by :

$$Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda) = \log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi}) + \lambda_1(1-\sum_{j=1}^{M}\pi_j) + \lambda_2(1-\sum_{j=1}^{M}\sum_{i=1}^{K_j} w_{ij}) \quad (2.15)$$

where $\Lambda = \{\lambda_1,\lambda_2\}$ is the Lagrange multiplier.

2.3.2 The Inverted Dirichlet Distribution

Although the framework that we propose is general and can therefore adopt any possible probability density function, we consider the inverted Dirichlet distribution as the density model, as we have used positive data. This distribution permits multiple symmetric and asymmetric modes and it may be skewed to the right, skewed to the left or symmetric, which gives it suitable properties to model different forms of positive data. Several interesting properties of the inverted Dirichlet can be found in [68]. If a D-dimensional positive vector $\vec{X} = (X_1,X_2,\ldots,X_D)$ follows an inverted Dirichlet distribution, the joint density function is given by [66–69]:

$$p(\vec{X}|\vec{\varphi}) = \frac{\Gamma(|\vec{\varphi}|)}{\prod_{d=1}^{D+1}\Gamma(\vec{\varphi}(d))}\prod_{d=1}^{D} X_d^{\vec{\varphi}(d)-1}(1+\sum_{d=1}^{D}X_d)^{-|\vec{\varphi}|} \quad (2.16)$$

where $\Gamma(.)$ is the gamma function, $X_d > 0, d = 1,2,\ldots,D$, $\vec{\varphi} = (\vec{\varphi}(1),\ldots,\vec{\varphi}(D+1))$ is the vector of parameters and $|\vec{\varphi}| = \sum_{d=1}^{D+1}\vec{\varphi}(d)$, $\vec{\varphi}(d) > 0, d = 1,2,\ldots,D+1$. The inverted Dirichlet distribution was introduced for the first time in [69] where the authors derived it from the Dirichlet distribution. Another derivation based on the Gamma distribution was proposed also in [70]. The mean and the variance of the inverted Dirichlet distribution are given by [69]

$$E(X_d) = \frac{\vec{\varphi}(d)}{\vec{\varphi}(D+1)-1} \quad (2.17)$$

$$Var(X_d) = \frac{\vec{\varphi}(d)(\vec{\varphi}(d)+\vec{\varphi}(D+1)-1)}{(\vec{\varphi}(D+1)-1)^2(\vec{\varphi}(D+1)-2)} \quad (2.18)$$

The parameters of an inverted Dirichlet distribution can be estimated using the method of moments that relies on low order statistics namely the mean and the variance given by Eqs. 2.17 and 2.18, respectively, from which we can straightforwardly find the following estimates :

$$\vec{\varphi}(D+1) = \frac{E(X_d)^2 + E(X_d)}{Var(X_d)} + 2 \tag{2.19}$$

$$\vec{\varphi}(d) = E(X_d)(\vec{\varphi}(D+1) - 1) \quad d = 1, \ldots, D+1 \tag{2.20}$$

In [70], a method has been proposed to generate inverted Dirichlet data. Let $X_1, X_2, \ldots, X_{D+1}$ be independent variables which follow Gamma distributions having the same scale but with different parameters $\vec{\varphi}(1), \vec{\varphi}(2), \ldots, \vec{\varphi}(D+1)$, respectively. Let $Y_d = \frac{X_d}{X_{D+1}}, \quad d = 1, 2, \ldots, D$, then the vector $\vec{Y} = (Y_1, Y_2, \ldots, Y_D)$ has a D-variate inverted Dirichlet distribution with parameter vector $\vec{\varphi} = (\vec{\varphi}(1), \vec{\varphi}(2), \ldots, \vec{\varphi}(D+1))$.

2.3.3 Parameters Estimation of the Hierarchical Mixture

The parameters estimation is based on the maximization of the log likelihood (Eq.2.15). The maximization gives the following for the mixtures weights (see Appendix A):

$$\pi_j = \frac{\sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})}{N} \tag{2.21}$$

$$w_{ji} = \frac{\sum_{n=1}^{N} p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})}{N} \tag{2.22}$$

Calculating the derivative with respect to $\vec{\varphi}_{ji}(d)$, $d = 1, \ldots, D$, and using the inverted Dirichlet distribution, we obtain:

$$\begin{aligned} \frac{\partial Q(\mathscr{X}, \varphi, \{w_{ji}\}, \vec{\pi}, \Lambda)}{\partial \vec{\varphi}_{ji}(d)} &= \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \frac{\partial \log(p(\vec{X}_n|\vec{\varphi}_{ji})}{\partial \vec{\varphi}_{ji}(d)} \\ &= \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \\ &\times (\Psi(|\vec{\varphi}_{ji}|) - \Psi(\vec{\varphi}_{ji}(d)) + \log(\frac{X_{nd}}{1 + \sum_{d=1}^{D} X_{nd}})) \end{aligned} \tag{2.23}$$

where $\Psi(.)$ is the digamma function. The derivative with respect to $\vec{\varphi}_{ji}(D+1)$ is given by :

$$\frac{\partial Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial\vec{\varphi}_{ji}(D+1)} = \sum_{n=1}^{N} p(j|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})\frac{\partial\log(p(\vec{X}_n|\vec{\varphi}_{ji})}{\partial\vec{\varphi}_{ji}(D+1)}$$
$$= \sum_{n=1}^{N} p(j|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})$$
$$\times(\Psi(|\vec{\varphi}_{ji}|)-\Psi(\vec{\varphi}_{ji}(D+1))+\log\frac{1}{1+\sum_{d=1}^{D}X_{nd}})) \tag{2.24}$$

Looking at the previous two equations, it is clear that an explicit form of the solution to estimate $\vec{\varphi}_{ji}$ does not exist. Thus, we refer to Newton Raphson method expressed as :

$$\vec{\varphi}_{ji}^{new} = \vec{\varphi}_{ji}^{old} - H_{ji}^{-1}G_{ji} \quad , j = 1...M, \quad i = 1...K_j \tag{2.25}$$

where H_{ji} is the Hessian matrix associated with $Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)$ and G_{ji} is the first derivatives vector,
$G_{ji} = (\frac{\partial Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial\vec{\varphi}_{ji}(1)},\ldots,\frac{\partial Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial\vec{\varphi}_{ji}(D+1)})^T$. To calculate the Hessian of $Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)$ we have to compute the second and mixed derivatives:

$$\frac{\partial^2 Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial\vec{\varphi}_{ji}(d)^2} = (\Psi'(|\vec{\varphi}_{ji}|)-\Psi'(\vec{\varphi}_{ji}(d)))(\sum_{n=1}^{N} p(j|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})) \tag{2.26}$$

$$\frac{\partial^2 Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial\vec{\varphi}_{ji}(d_1)\partial\vec{\varphi}_{ji}(d_2)} = (\Psi'(|\vec{\varphi}_{ji}|))(\sum_{n=1}^{N} p(j|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})) \tag{2.27}$$

where $\Psi'(.)$ is the trigamma function.Thus ,

$$H_{ji} = \sum_{n=1}^{N} p(j|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})$$
$$\times\begin{pmatrix}\Psi'(|\vec{\varphi}_{ji}|)-\Psi'(\vec{\varphi}_{ji}(1)) & \Psi'(|\vec{\varphi}_{ji}|) & \cdots & \Psi'(|\vec{\varphi}_{ji}|)\\ \Psi'(|\vec{\varphi}_{ji}|) & \Psi'(|\vec{\varphi}_{ji}|)-\Psi'(\vec{\varphi}_{ji}(2)) & \cdots & \Psi'(|\vec{\varphi}_{ji}|)\\ \vdots & \vdots & \ddots & \vdots\\ \Psi'(|\vec{\varphi}_{ji}|) & \cdots & & \Psi'(|\vec{\varphi}_{ji}|)-\Psi'(\vec{\varphi}_{ji}(D+1))\end{pmatrix} \tag{2.28}$$

we can write :

$$H_{ji} = D_{ji} + \alpha_{ji} A_{ji}^T A_{ji} \tag{2.29}$$

where

$$D_{ji} = diag\big[- \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \Psi'(\vec{\varphi}_{ji}(1)), \ldots$$
$$\ldots, - \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \Psi'(\vec{\varphi}_{ji}(D+1))\big] \tag{2.30}$$

D_{ji} is a diagonal matrix, $\alpha_{ji} = (\sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})) \Psi'(|\vec{\varphi}_{ji}|)$ and $A_{ji}^T = (a_1, ..., a_{D+1})$, $a_d = 1$, $d = 1, ..., D+1$, then using the Theorem 8.3.3 in [71], the inverse matrix is given by :

$$H_{ji}^{-1} = D_{ji}^{-1} + \alpha_{ji}^* A_{ji}^{*T} A_{ji}^* \tag{2.31}$$

$D_{ji}^{-}1$ can be easily computed as following:

$$A_{ji}^* = \frac{1}{\sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})} \left(\frac{1}{\Psi'(\vec{\varphi}_{ji}(1))}, \ldots, \frac{1}{\Psi'(\vec{\varphi}_{ji}(D+1))}\right) \tag{2.32}$$

and

$$\alpha_{ji}^* = [(\Psi'(|\vec{\varphi}_{ji}|) \sum_{d=1}^{D+1} \frac{1}{\Psi'(\vec{\varphi}_{ji}(d))}) - 1]^{-1} \Psi'(|\vec{\varphi}_{ji}|) \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \tag{2.33}$$

2.4 Generalization of the Model

2.4.1 A Flexible Hierarchical Model

In the previous sections, two levels of the hierarchy were modeled with prior knowledge about how to group the classes within the *super classes*. In many real life applications, however, we do not have this prior knowledge. Thus, we propose to overcome this problem by generalizing our model to several hierarchical levels. Indeed, when we change the strategy of how we look at the model, we can overcome the model selection problem at each *super class* level (e.g. prior knowledge about the number of components that each *super class* must have)[1]. Indeed, the second level can

[1]Many applications might find the prior specification of the number of components forming each *super class*, useful. In our case we want the system to form the classes and then group them dynamically according to the user's

be viewed as a mixture of $\sum_{j=1}^{M} K_j$ components with mixture weights w_{ji} such as $\sum_{j=1}^{M}\sum_{i=1}^{K_j} w_{ji} = 1$. According to Eqs. 2.12 and 2.13, we have :

$$p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) = \sum_{i=1}^{K_j} p(j,i|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \tag{2.34}$$

Thus, Eqs.2.21, 2.22 and 2.34 give us:

$$\pi_j = \sum_{i=1}^{K_j} w_{ji} \tag{2.35}$$

When considering a hierarchical model with several levels, we propose to estimate the probability densities parameters and the weight of the bottom level using the algorithm that we proposed in [66], and then estimate the weight of each upper level by summing the weights of their children. This approach can be practical in real life applications when we do not have prior knowledge about the hierarchy as we will show further in the experimental part. For instance, an object class can be modeled by a K_j-components mixture without having prior knowledge about K_j. To define a *super class* of objects classes, we simply sum their corresponding weights and we move to an upper level in the hierarchy. Thus, we mainly develop two approaches, the first one is when we have prior knowledge about the number of clusters/classes forming a *super class* and the second one when we do not have such knowledge, so we can group the clusters/classes according to a given hierarchy that can change on the fly, which is the purpose of this work.

2.4.2 Learning Algorithm

To initialize the EM algorithm we use the approach proposed in [66] based on K-Means algorithm and the method of moments. As we have two approaches for building the hierarchical model, we present two algorithms. The first one when we have the required prior knowledge about the *super classes* and the second one when we do not have such knowledge.

Learning in the Presence of Prior Knowledge

When we know in advance the number of components forming each *super class*, we have two ways to initialize the parameters. The first way is to let the system group the clusters basing on similarities, thus, we use the K-Means algorithm to have M clusters representing the *super classes* and then reuse K-Means to determine the appropriate classes K_j of each *super class*. The second way is to use K-Means to have $\sum_{j=1}^{M} K_j$ clusters and then group them within each *super class* j.

needs. In this chapter, the two approaches are proposed.

- **Initialization Algorithm**

  1 – Grouping by the system: Apply the K-Means on the N D-dimensional vectors to obtain initial M *super classes* and reapply it on each *super class* j to obtain K_j clusters.

  – Grouping by the user : Apply the K-means on the N D-dimensional vectors to obtain initial $\sum_{j=1}^{M} K_j$, and then group the chosen clusters K_j for each *super class* j.

  2 Weights initialization :

  – Calculate $w_{ji} = \frac{Number\ of\ elements\ in\ cluster\ ji}{N}$

  – Calculate $\pi_j = \frac{Number\ of\ elements\ in\ super\ class\ j}{N}$

  3. Apply the moments method for each component ji to obtain a vector of parameters corresponding to a given cluster ji.

  Then, the estimation algorithm can be summarized as follows

- **Estimation Algorithm**

  1. INPUT: D-dimensional data $\vec{X}_n$, $n = 1,\ldots,N$, a specified number of *super classes* M, and the number of classes, K_j, in each *super class* j .

  2. Initialization algorithm

  3. E-Step: Compute the posterior probabilities using Eqs.2.12 and 2.13.

  4. M-Step:

  – Update $\vec{\varphi}_{ji}$ using Eq.2.25, $j = 1,\ldots,M$, $i = 1,\ldots,K_j$.

  – Update w_{ji} using Eq.2.22, and π_j using Eq.2.35, $j = 1,\ldots,M$, $i = 1,\ldots,K_j$

  5. If the convergence test ($\Delta p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi}) < \varepsilon$) is passed terminate, else go to 3.

where $\Delta p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi})$ is the difference between the likelihoods calculated in two consecutive iterations.

Dynamical Hierarchical Grouping Without Prior Knowledge About the *Super Classes*

For many applications, the hierarchical model can be built on the fly according to the user's need or depending on some circumstances. Here, we present the algorithm proposed in this case.

- **Initialization Algorithm**

1.1 Apply K-Means on the N D-dimensional vectors to obtain initial $\sum_{j=1}^{M} K_j$ clusters.

2. Calculate $w_{ji} = \frac{Number\ of\ elements\ in\ cluster\ ji}{N}$

3. Apply the moments method for each component ji to obtain the initial parameters corresponding to each cluster.

Then, the estimation algorithm can be summarized as follows

- **Estimation Algorithm**

1. INPUT: D-dimensional data $\vec{X}_n$, $n = 1,\ldots,N$ and a specified number of total clusters $\sum_{j=1}^{M} K_j$.

2. Initialization algorithm.

3. E-Step: Compute the posterior probabilities for $\sum_{j=1}^{M} K_j$ components of an IDMM using the algorithm in [66].

4. M-Step:

- Update the parameters of the mixture at the lowest level according to [66], considering a finite mixture of IDMM with $\sum_{j=1}^{M} K_j$ components.
- Update $w_t = w_{ji}$ using Eq.2.22 when considering the posterior probabilities calculated in 3, $j = 1,\ldots,M$, $i = 1,\ldots,K_j$, $t = 1,\ldots,\sum_{j=1}^{M} K_j$.

5. If the convergence test ($\Delta p(\mathscr{X}|\varphi,\{w_t\}) < \varepsilon$) is passed go to 6, else go to 3.
$p(\mathscr{X}|\varphi,w_t)$ is the likelihood of the considered IDMM having $\sum_{j=1}^{M} K_j$-components, with a set of parameters $\{\varphi,\{w_t\}\}$, $t = 1,\ldots,\sum_{j=1}^{M} K_j$, $j = 1,\ldots,M$.

6. For each level of the hierarchy: compute $\pi_j = \sum_{i=1}^{K_j} w_{ji}$ according to a given ontological model.

Using this algorithm, we do not have to specify K_j for each *super class* j, but we rather specify the total number $\sum_{j=1}^{M} K_j$ of the modeled classes at the lowest level of the hierarchy. We then specify K_j for each *super class* j basing on a given ontological model, and the obtained results. It is noteworthy to mention that any number of levels in the hierarchy can be built by simply constructing its weights by summing their respective children weights. In our application and experiments we use this algorithm.

2.5 Experimental Results

2.5.1 Synthetic Data

In this section, an evaluation of our proposed algorithm is performed using synthetic data. We report results on one-dimensional and multi-dimensional synthetic datasets. We also analyze the performance of our estimation approach in the presence of outliers and finally we analyze the capabilities of our approach in modeling overlapping classes. It is noteworthy that we suppose here that we have a certain knowledge about the grouping of the different classes into *super classes* based on a given semantic model.

Results

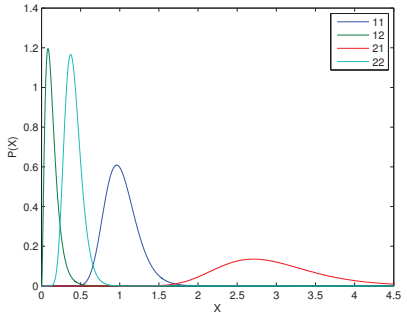
We performed all the following experiments with a Newton Raphson convergence precision equal to $\varepsilon = 10^{-5}$. We generated artificial histograms from artificial inverted Beta mixture distributions[2], and we estimated their related parameters according to the proposed algorithm. For the first histogram we considered two *super classes* with two classes each, as shown in Figure 2.2 which displays different representations of the considered model. Figure 2.2a shows the different classes of the model, whereas Figure 2.2b shows the considered *super classes*. Note that a different hierarchy of *super classes* can be chosen by grouping different classes. Figure 2.2c shows the whole mixture. The obtained results are reported in Table 2.1 where we show the real and estimated parameters. The maximum detected percent error of relative change between real and estimated parameters among all the set of parameters is 1.06% which reflects a good estimation. Thus, the estimated histogram and the real one are indistinguishable. Our algorithm is shown to perform well for one dimensional data.

Table 2.1: Real and estimated parameters in the case of a one-dimensional dataset generated from 4-components mixture model.

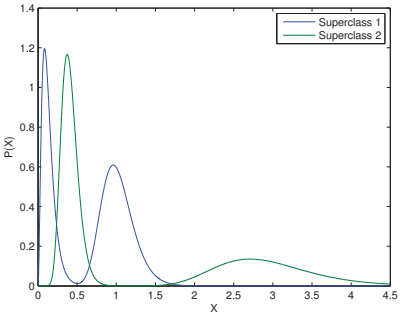
j	i	π_j	w_{ji}	$\vec{\varphi}_{ji}(1)$	$\vec{\varphi}_{ji}(2)$	$\hat{\pi}_j$	$\hat{w}_{ji}$	$\hat{\vec{\varphi}}_{ji}(1)$	$\hat{\vec{\varphi}}_{ji}(2)$
1	1	0.5	0.3	50	50	0.4997	0.2993	50.5311	50.4579
	2		0.2	3	23		0.2004	2.9847	22.8295
2	1	0.5	0.2	85	30	0.5003	0.2000	85.0137	29.9959
	2		0.3	20	50		0.3003	20.0125	49.9427

We also performed a set of experiments on multi-dimensional generated data. We generated two different 2-dimensional datasets (see Figures 2.3 and 2.4). In the first experiment associated with Figure 2.3, we generate three *super classes*, the first one groups two classes, and the others

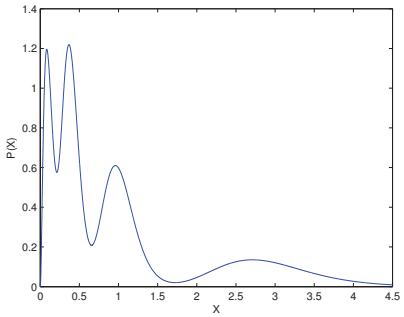
[2]The inverted Beta is the one-dimensional special case of the inverted Dirichlet obtained when $D = 1$.



(a) Classes labeled in ji



(b) *Super classes*



(c) Mixture

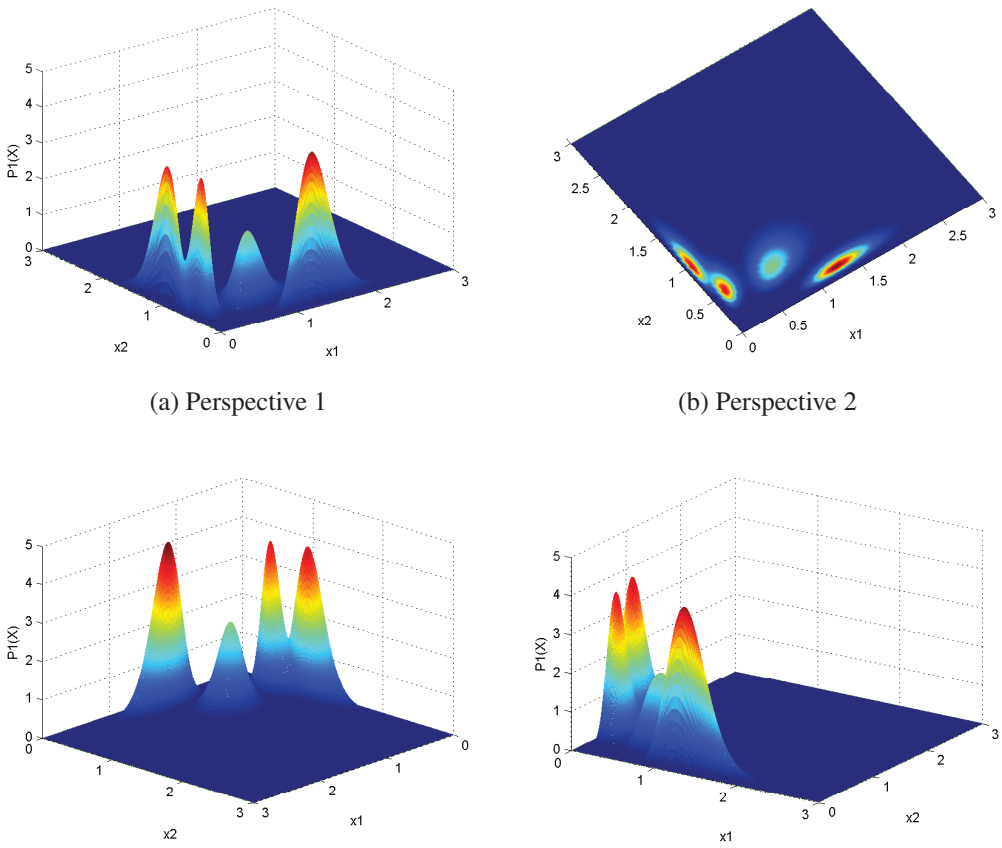
Figure 2.2: Different representations of the model in Table 2.1.

Table 2.2: Real and estimated parameters in the case of a two-dimensional dataset generated from 4-components mixture model.

j	i	π_j	w_{ji}	$\vec{\varphi}_{ji}(1)$	$\vec{\varphi}_{ji}(2)$	$\vec{\varphi}_{ji}(3)$	$\hat{\pi}_j$	$\hat{w}_{ji}$	$\hat{\vec{\varphi}}_{ji}(1)$	$\hat{\vec{\varphi}}_{ji}(2)$	$\hat{\vec{\varphi}}_{ji}(3)$
1	1	0.4	0.2	5	53	50	0.4000	0.2003	4.9778	52.6717	49.7877
	2		0.2	10	30	50		0.1997	9.9563	29.8248	49.7256
2	1	0.3	0.3	32	23	40	0.3000	0.3000	32.2221	23.1393	40.2735
3	1	0.3	0.3	80	10	60	0.3000	0.3000	80.3250	10.0611	60.3193

Table 2.3: Real and estimated parameters in the case of a two-dimensional dataset generated from 6-components mixture model.

j	i	π_j	w_{ji}	$\vec{\varphi}_{ji}(1)$	$\vec{\varphi}_{ji}(2)$	$\vec{\varphi}_{ji}(3)$	$\hat{\pi}_j$	$\hat{w}_{ji}$	$\hat{\vec{\varphi}}_{ji}(1)$	$\hat{\vec{\varphi}}_{ji}(2)$	$\hat{\vec{\varphi}}_{ji}(3)$
1	1	0.3	0.15	18	60	60	0.2997	0.1500	18.1848	60.6012	60.6201
	2		0.15	18	20	34		0.1497	17.8870	19.9435	33.9275
2	1	0.7	0.10	53	9	39	0.7003	0.1001	52.5474	8.9642	38.6597
	2		0.20	50	50	50		0.2007	49.4120	49.4034	49.4537
	3		0.20	5	40	23		0.1997	5.0077	40.1441	23.0689
	4		0.20	20	8	40		0.1998	20.0005	7.9958	39.9639



(a) Perspective 1

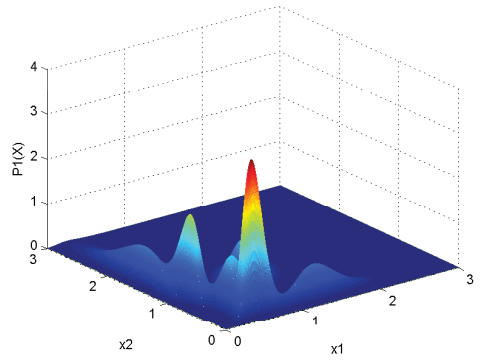
(b) Perspective 2

(c) Perspective 3

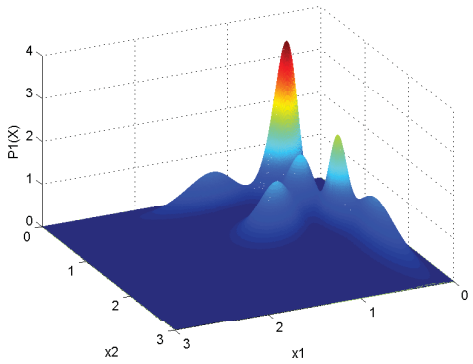
(d) Perspective 4

Figure 2.3: Different perspectives for the representation of the model in Table 2.2.

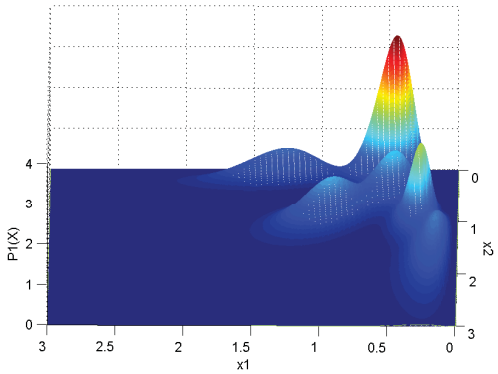
have one class each. The results are reported in Table 2.2. In the second experiment associated with Figure 2.4, we generate two *super classes* where the first one groups two classes and the second one groups four classes. The results are reported in Table 2.3. The maximum detected percent error of relative change between real and estimated parameters among all the set of parameters for the two experiments are respectively 0.69% and 1.19% which reflects a good estimation as well, as we have considered overlapping densities. Naturally the more the densities get overlapped the harder it is to have good estimates of their parameters. It is needless to say that the representation of the estimated probability mixtures and the real probability mixtures are also indistinguishable. Finally, we considered 6-dimensional data where we generated 2 *super classes*, one having 2 classes and the other having one single class. The results are reported in Tables 2.4 and 2.5 where we respectively report on real and estimated parameters. The maximum detected percent error of relative change between real and estimated parameters among all the set of parameters is 0.44% which shows again that our algorithm performs well when dealing with multi-dimensional data.



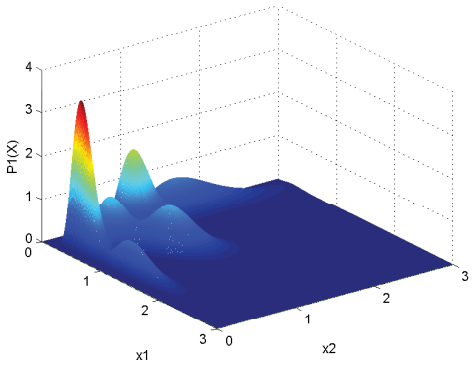
(a) Perspective 1



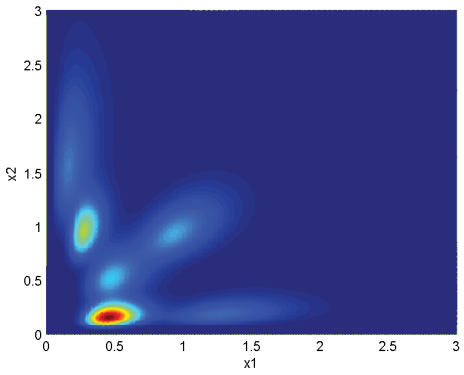
(b) Perspective 2



(c) Perspective 3



(d) Perspective 4



(e) Perspective 5

Figure 2.4: Different perspectives for the representation of the model in Table 2.3.

Table 2.4: Real parameters in the case of a 6-dimensional dataset generated from 3-components mixture model.

j	i	π_j	w_{ji}	$\vec{\varphi}_{ji}(1)$	$\vec{\varphi}_{ji}(2)$	$\vec{\varphi}_{ji}(3)$	$\vec{\varphi}_{ji}(4)$	$\vec{\varphi}_{ji}(5)$	$\vec{\varphi}_{ji}(6)$	$\vec{\varphi}_{ji}(7)$
1	1	0.8	0.4	50	39	34	22	56	3	41
	2		0.4	18	19	29	39	49	32	95
2	1	0.2	0.2	43	56	90	93	94	95	32

Table 2.5: Estimated parameters in the case of a 6-dimensional dataset generated from 3-components mixture model.

j	i	$\hat{\pi}_j$	$\hat{w}_{ji}$	$\hat{\vec{\varphi}}_{ji}(1)$	$\hat{\vec{\varphi}}_{ji}(2)$	$\hat{\vec{\varphi}}_{ji}(3)$	$\hat{\vec{\varphi}}_{ji}(4)$	$\hat{\vec{\varphi}}_{ji}(5)$	$\hat{\vec{\varphi}}_{ji}(6)$	$\hat{\vec{\varphi}}_{ji}(7)$
1	1	0.8000	0.4000	49.9276	38.9625	33.9762	21.9628	55.9363	2.9990	40.9625
	2		0.4000	18.0368	18.9945	29.0011	39.0077	49.0462	32.0068	95.0483
2	1	0.2000	0.2000	43.1571	56.2434	90.3963	93.2768	94.2857	95.2934	32.1198

The Estimator Robustness

The purpose of this experiment is to empirically evaluate a breakdown point which gives an idea about the robustness of our estimator and how our algorithm gets affected by outliers, as in the previous subsections we used perfect data that are generated completely from inverted Dirichlet distributions. The breakdown point is the smallest fraction of data contamination needed to cause an arbitrarily large change in the estimate [72, 73]. We propose to generate two *super classes* one with two classes and the other with one single class as shown in Table 2.6, then we contaminate the data with a certain percentage of outliers, that do not follow any of the previous used distributions. The outliers can be generated from any other distribution (e.g., normal distribution, uniform distribution, etc.). We model the outliers by a normal distribution with mean $\mu = 0$ and a standard deviation $\sigma = 4$, as the histogram is located between 0 and 4. We substitute the perfect generated data by the outliers according to the contamination level using a uniform distribution. Naturally the weight can be significantly affected, but we focus more on the change of the distributions shape. The obtained results are reported in Figure 2.5 where we plot the different histograms including

Table 2.6: Real parameters in the case of a one-dimensional dataset generated from 3-components mixture model.

j	i	π_j	w_{ji}	$\vec{\varphi}_{ji}(1)$	$\vec{\varphi}_{ji}(2)$
1	1	0.8	0.4	8	10
	2		0.4	90	40
2	1	0.2	0.2	10	60

the real one and the estimated histograms of the data contaminated with different percentage of outliers. As shown in the previous subsections when using perfect data with 0% contamination the real and estimated histograms are indistinguishable. As we increase the contamination level the shape of the histograms starts to differ from the real one. The shape does not change significantly

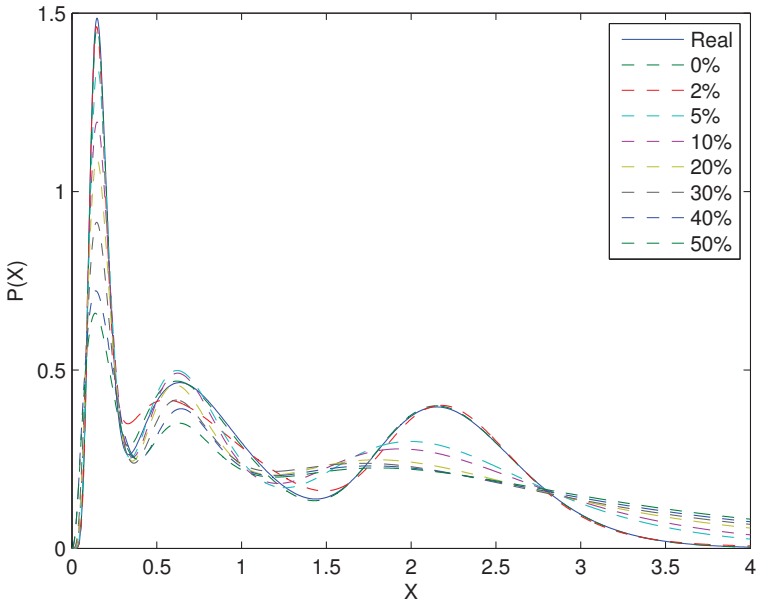


Figure 2.5: Real and estimated histograms using different data contamination levels (%).

until a certain level of contamination, and then it naturally starts to deviate from the original shape. The aspect of the histogram with a very high contamination achieving 50% remains acceptable as we still can distinguish the three clusters with their respective shapes. These results show that our estimator is robust when affected by outliers. Nevertheless, the choice and nature of outliers themselves can affect significantly these results. For instance, if we use outliers whose values overpass certain boundaries (extreme outliers), these results might be altered. An eventual solution to face the outliers problem is to model the outliers themselves within a *super class* grouping eventual outliers as done, for example, in [74].

Classic Model vs Multi-Clusters Model

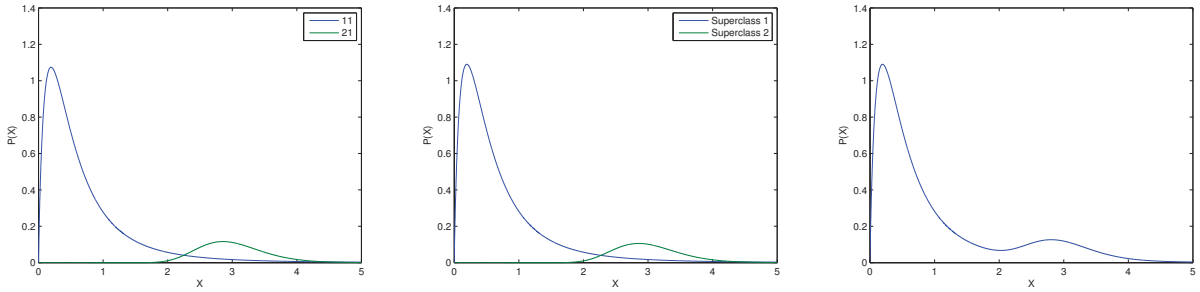
When we encounter a clustering problem, one important aspect is to precise the number of classes representing the data. Nevertheless, in many applications, the number of classes is already known (e.g. binary classification: an image contains an object or not). The classical approach when we have a fixed number of classes, is to model the data by a mixture whose number of components is equal to the number of classes. However, when using the classical approach, the modeling process can face two major problems. The first problem is that one single density component does not necessarily fit the data class. Even though we use flexible probability densities in terms of shape and symmetry or asymmetry, in many cases the data points of certain classes cannot be modeled by a single mixture component. The second major problem is the overlapping between the classes

when using a single distribution to model each class. Consider for instance that the class elements of a given class X are distributed between the class elements of another class Y, an adequate data modeling would be unlikely well done when using a single density component to represent each class. In this subsection, we compare the classic data modeling approach with our proposed one in terms of data fitting and adequate representations. Let us reconsider the model in Table 2.1 displayed in Figure 2.2. We generate a set of points according to these distributions, and we propose to use them to estimate their related distributions' parameters assuming that they are generated from inverted Dirichlet distributions. Knowing that we have two *super classes*, thus two categories, the classical approach consists of representing every *super class*, with one single density. The second approach which is ours, is to represent every *super class* with a set of classes or clusters. We report on the resulted histograms in Figures 2.6 and 2.7.

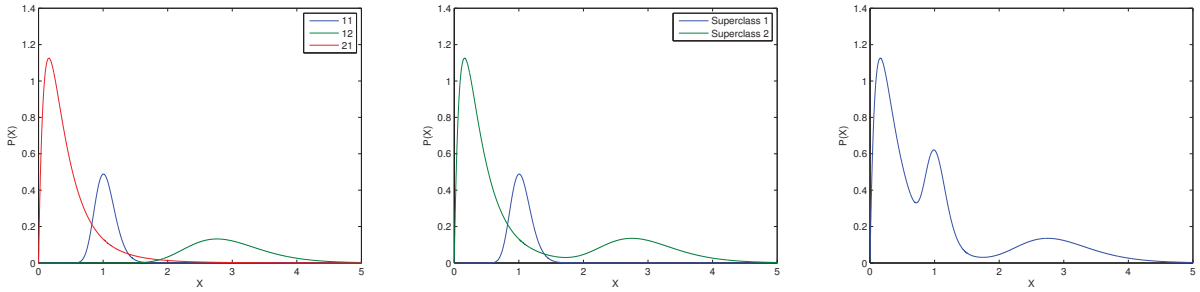
We established several experiments using different numbers of clusters. The exact number of model clusters which is four, is already experimented and it gives very good results as it generates histograms that are indistinguishable from the real ones. When using two clusters only, the results do not reflect the real histograms and the algorithm fails to find the real shape of the mixture, which is expected as we have two *super classes* whose parts are within each others. So, the classical approach fails to find out the real shape of the histograms. When using three clusters, we have better results as we approach the real shape of the *super class* 2. When using four clusters and more, the algorithm succeeds to find the real shape of the *super classes* and the mixture. It is noteworthy that we assume the presence of knowledge about how to group the clusters into *super classes*. When using ten clusters, we distinguished the classes in Figure 2.7d (for easiness of presentation, the classes belonging to the *super class* 2 were plotted in dash lines). Increasing the number of classes within a *super class* does not affect the shape of the *super class*, as we just distinguish more between the different classes at a lower level, but this does not affect their membership to their respective *super classes*. This observation will be discussed in details in subsection 2.5.2. As for now, It is clear that the classic model gives an inadequate representation, as the two *super classes* overlap and the classic approach fails to distinguish them. When considering our algorithm, the results are much closer to reality, since we clearly overcome the overlapping problem.

2.5.2 Real Data

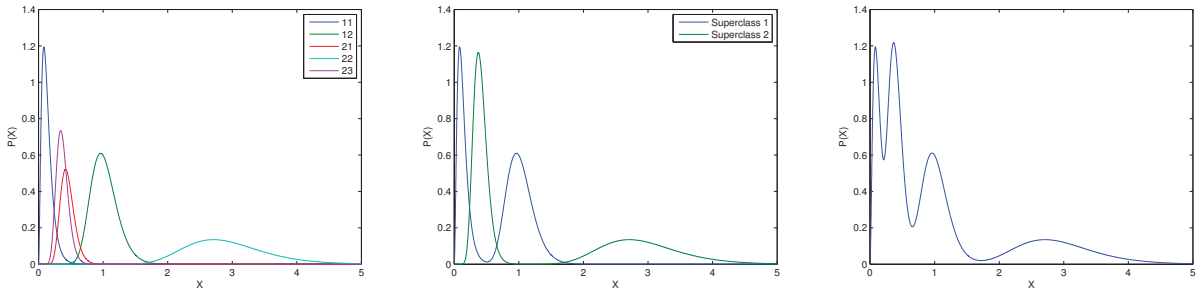
In this section we investigate the performance of our algorithm using real-life data extracted when tackling the challenging problem of object modeling and recognition. The first goal of this application is to analyze the benefits of our approach as compared to the commonly used method where every object class is represented by one single cluster, thus one single probability density. We also



(a) 2 classes labeled in ji (b) 2 *super classes* formed by 2 classes (c) Mixture formed by 2 classes



(d) 3 classes labeled in ji (e) 2 *super classes* formed by 3 classes (f) Mixture formed by 3 classes

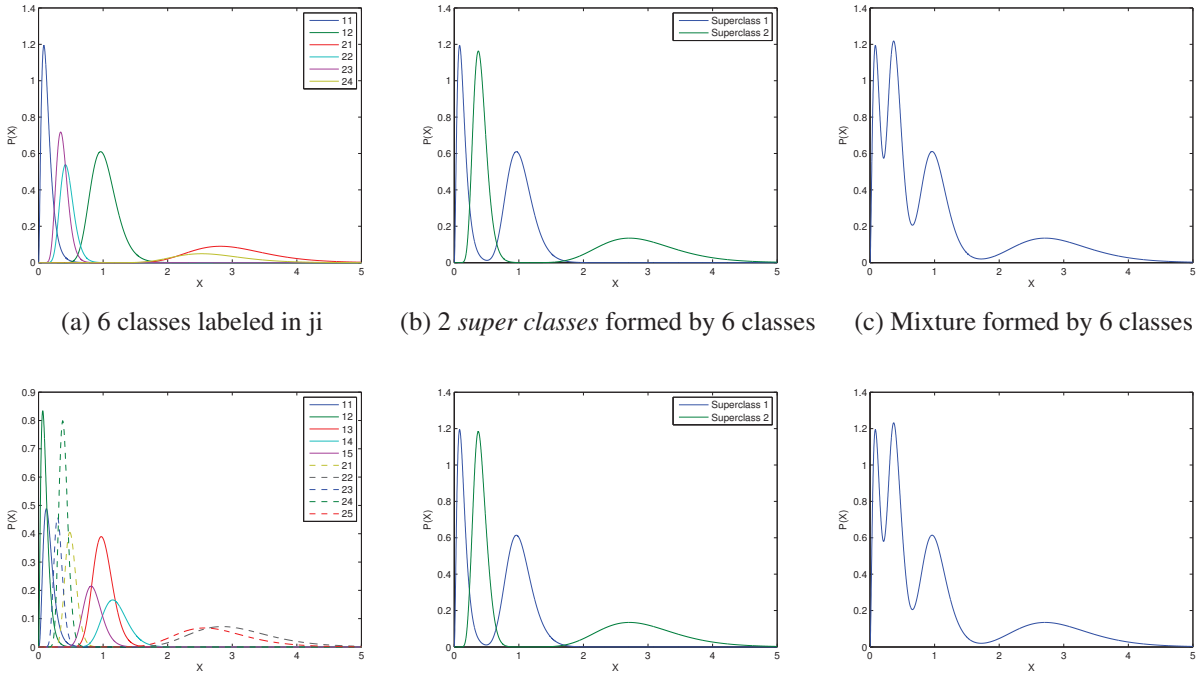


(g) 5 classes labeled in ji (h) 2 *super classes* formed by 5 classes (i) Mixture formed by 5 classes

Figure 2.6: 1. Different generated models and their representations as a function of the number of used classes

discuss how to attribute different semantic meanings to the clusters according to a chosen hierarchy during the training phase. It is noteworthy that the proposed hierarchical grouping approach is simple and do not need extra computational time when changing the hierarchical model. We have considered the publicly available ETH-80 [3] dataset which is composed of 3280 images grouped into eight object classes having ten unique objects each, as it is shown in Figure 2.8. Each object has 41 perspectives. Figure 2.9 shows the different perspectives of the object 'Dog' located on the extreme left of the 'Dog' class in Figure 2.8. We propose to apply our algorithm on this database which is interesting in terms of hierarchical classification as the objects classes can be grouped

[3] http://www.d2.mpi-inf.mpg.de/Datasets/ETH80



(a) 6 classes labeled in ji (b) 2 *super classes* formed by 6 classes (c) Mixture formed by 6 classes

(d) 10 classes labeled in ji (e) 2 *super classes* formed by 10 classes (f) Mixture formed by 10 classes

Figure 2.7: 2. Different generated models and their representations as a function of the number of used classes

into different hierarchical levels. Indeed, different *super classes* can be considered such as objects (Car, Cup) vs non objects, or animals vs fruits vs objects or fruits vs non fruits, etc. The features of each image have been extracted using the local Histogram of Oriented Gradient (HOG) descriptor proposed in [75]. The HOG descriptor is efficient in terms of detecting local characteristics of images and object detection. It generates descriptors having positive values which justifies the use of the inverted Dirichlet distribution. In our experiment each image was represented by 81-dimensional feature vector. We considered a 10-fold cross validation [76] for the evaluation of our algorithm. The training data are used to learn the mixture model at the bottom level, and to build the hierarchy according to the model that we shall discuss in the next subsection. The testing data are used to validate the statistical model and analyze its capability to recognize different testing objects and affect the unseen data to their related clusters properly. The results are illustrated in 'Results' subsection.

Clusters Assignment and Semantic Grouping of Data

Our used approach may generate many clusters that need to be assigned to a *super class* and a label. In order to attribute well a cluster to a specific object class, we label each cluster according

Figure 2.8: 10 objects in 8 classes.

to the images that have been grouped into it. Indeed, we attribute each cluster k to the *super class* j whose elements are the most present in cluster k such that:

$$label_{cluster_k} = \arg\max_j \frac{number\ of\ elements\ of\ super\ \ class\ j\ in\ cluster_k}{number\ of\ elements\ in\ cluster_k} \tag{2.36}$$

The training data labels serve to specify the membership of each element present in a given cluster to the appropriate *super class*. Hence, the labeled training data can be seen as the user's semantic model of object classes serving to reduce the distance between the system representation and the user comprehension e.g. a user can specify a set of training data labeled as 'Tomato' and 'Apple' (binary model), then perform a clustering of the data into $\sum_{j=1}^{2} K_j$ clusters, say 10 clusters. The framework would label each cluster model as being a representation of 'Tomato' or 'Apple' according to the training labels and using Eq. 2.36. Indeed, one major aspect of this work is the semantic grouping of clusters as the proposed approach can fit any hierarchical model using a bottom-top methodology. The proposed method can change the hierarchy model at anytime with negligible

Figure 2.9: 41 different views of the object 'Dog'.

computational costs as it does not need to recompute the different density parameters once we have them (see subsection 2.4.1). Indeed, the used labeled data represent a key component to setup the hierarchy, as we can assign a semantic meaning to system level clusters using Eq. 2.36. We adopt a bottom-top approach, where we estimate the densities parameters and their related weights at the lowest hierarchy level, then we construct the *super classes* by grouping these densities into a sort of sub mixtures representing the object classes or the semantic that we want to consider. In the following, we discuss three possible hierarchical models for our application using ETH-80 dataset. It is important to bear in mind that other models may be considered.

- **Class to object:** As shown in Figure 2.10 the *super class* is considered to be an object class e.g. 'Tomato', 'Pear' ,'Cow', etc. Thus, $M = 8$ and each *super class* is composed of K_j clusters that are assigned according to Eq.2.36. When we consider the model from this perspective, we consider that each object is formed by a set of probability densities or

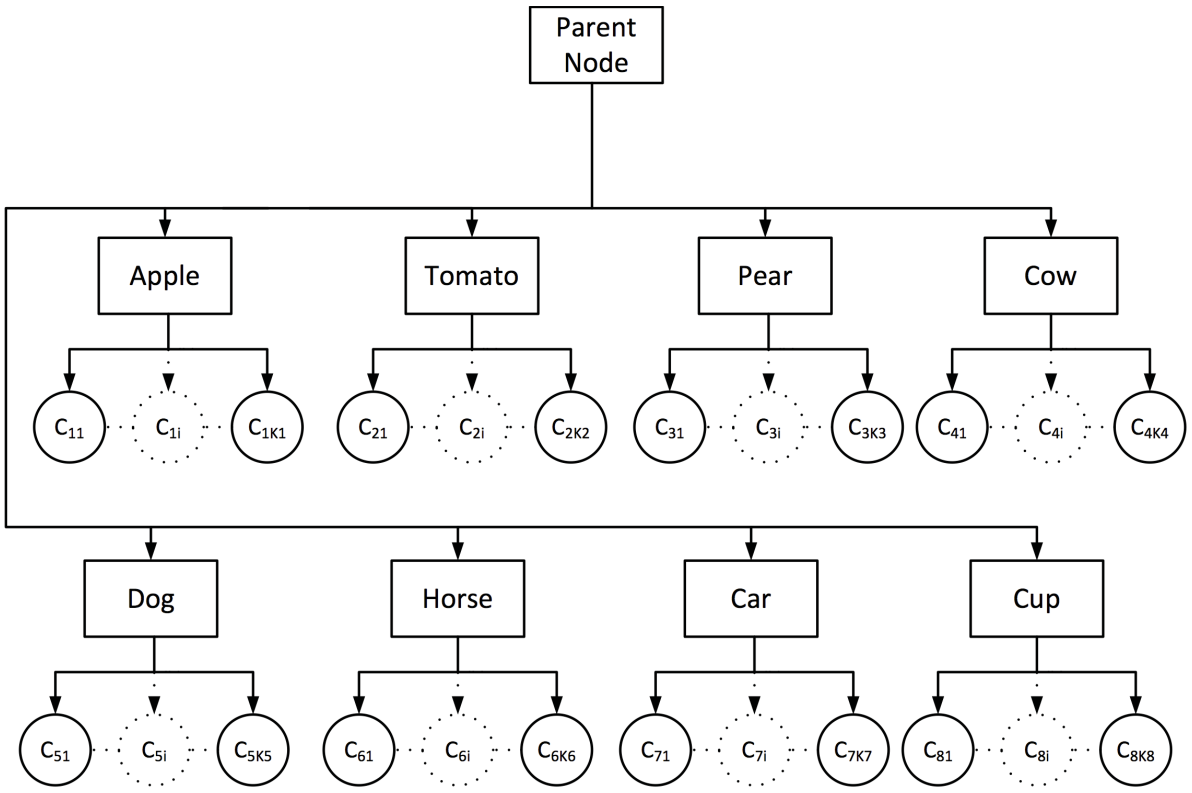


Figure 2.10: 8 objects hierarchical model of ETH-80.

a probabilistic mixture. The fact that in ETH-80 every object class is formed by 10 unique objects, each having 41 views, makes this choice practical as the diversity of visual features within a given object class is considerable. Semantically we can explain such a choice by the fact that an object can be formed by different low level features that can describe that class. e.g. an 'Apple' may have different colors, different shapes, different types, etc., so the diversity might be important in terms of system features, but expresses the same meaning to a human being.

- **Class to fruits vs non fruits:** Figure 2.11 considers a model of a higher level in the ontology where the *super classes* are the fruits and non fruits. In this case $M = 2$ and K_j is also concluded from Eq.2.36. Indeed, we simply change the different clusters grouping, so we keep the same system clustering at the lowest level but change the user semantic perspective to the data. We just group the clusters according to the given training labeled vector, which in this case composed of binary valued elements telling if an image is a fruit or not.

- **Class to object to fruits vs non fruit** When we merge the model of Figure 2.11 with the model of Figure 2.10, we can form a model where three levels are considered (see Figure 2.12). The first level is the system level, where the clusters are generated using the system

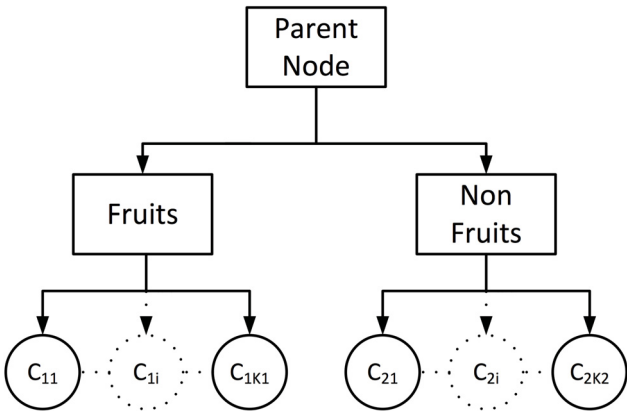


Figure 2.11: Fruits vs non fruits hierarchical model of ETH-80.

lowest features that do not necessarily have a comprehensible human semantic. The second level is a human semantic level where we construct the objects, and the third level is where we group the objects into a final constructed level which is fruits vs non fruits. It is noteworthy that we do not need any extra calculation in order to consider this hierarchy. This is a very interesting way to construct the model, as we can construct two human comprehensible semantic levels, and we can easily move from the system representation of the features to the human representation.

Results

We report on the accuracy of object classes categorization in Figure 2.13, and the classification of the two *super classes* fruits and non fruits in Figure 2.14. Recall that we used a bottom-top methodology to establish the classification using the model in Figure 2.12 mentioned previously. Both figures show clearly that the best result is obtained when considering 90 clusters. The objects classification accuracy starts with a very poor performance that equals 44.27% when using a number of clusters equal to the number of object classes, and ends up to be 71.80% when using 90 clusters. However, the accuracy of the fruits vs non fruits already starts with a respectable value that equals 87.87% when using a number of clusters equal to the number of objects classes (8 clusters) and it ends up to have a close accuracy value equal to 96.19% when using 90 clusters. It is needless to say that using two clusters only to model fruits vs non fruits gives very bad results (e.g. the accuracy goes under 40%), so using 8 clusters at the beginning of the experiment is already considered as a practice to our proposed approach.To better interpret those results and to explain why the difference of accuracy between the different models is huge for objects but not as much important when dealing with the fruits vs non fruits model we report on the detailed classification results using different numbers of total used clusters reported in Tables 2.7, 2.8, 2.9, 2.10, 2.11 and

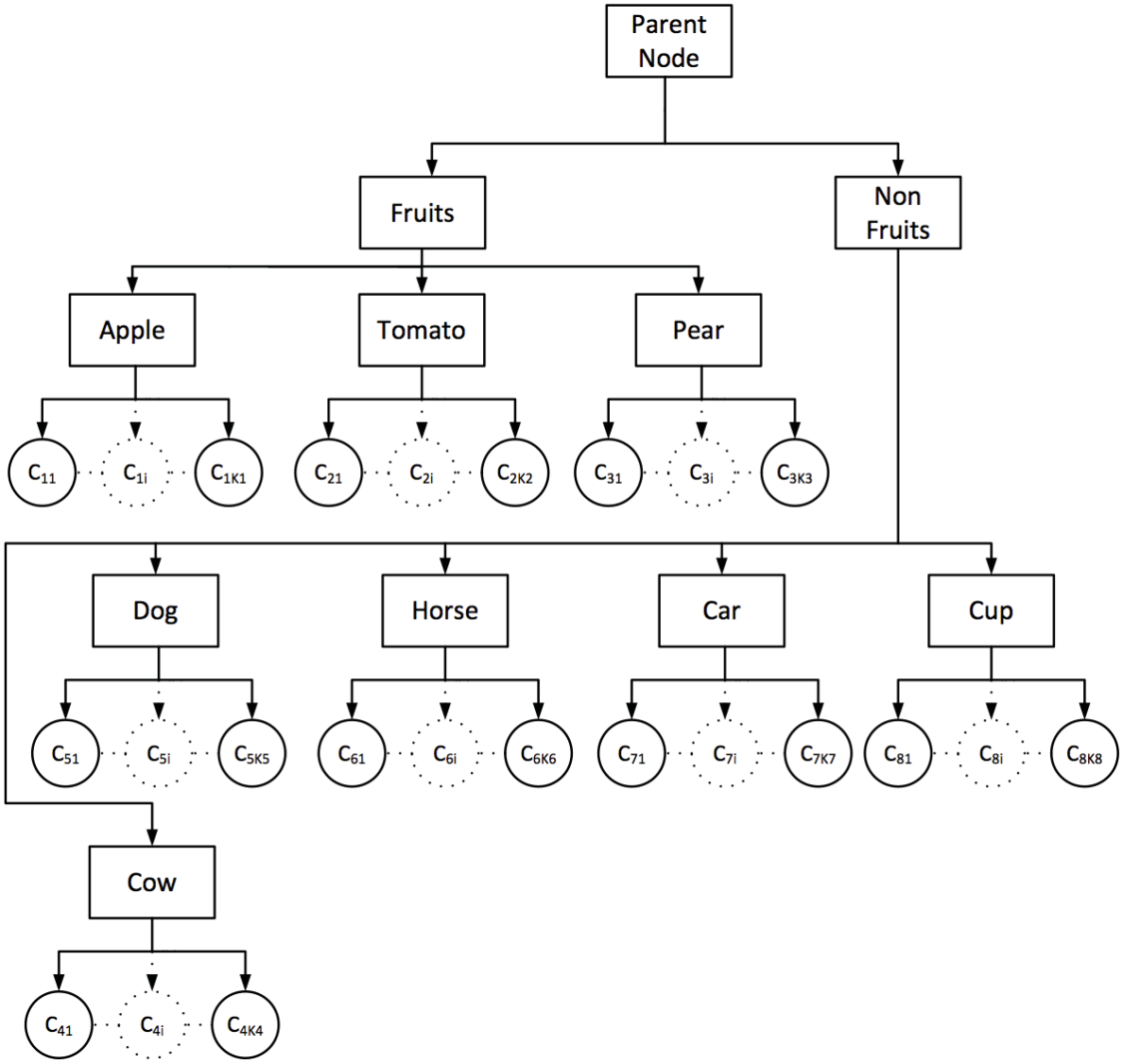


Figure 2.12: 3 levels hierarchical model of ETH-80.

2.12. For the easiness of lecture, we wiped all the misclassified percentage values that are under 10%, we considered two decimal points for the percentages, and colored the good classified cells on the confusion matrices diagonals with a blue color. When we look at the object classification accuracy we notice that using a number of clusters equal to the number of object classes gives poor results as we have 44.27% accuracy. The corresponding confusion matrix in Table 2.7 clearly shows that 8 clusters are unable to adequately model the data as the results are confusing and the model cannot differentiate between different objects whose features are certainly overlapping in the features space when considering that few number of clusters. A small increase in the number of the used clusters to 15 gives better results achieving 55.39 %. The corresponding confusing matrix in Table 2.8 informs us that some objects starts to be distinguished from the other objects and form

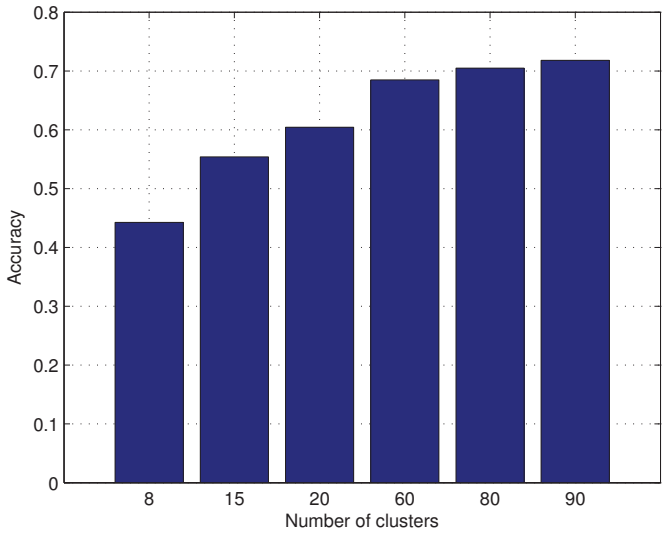


Figure 2.13: Classification Accuracy of objects classes in % as a function of the total used number of clusters.

their own clusters like the 'Cow' class .We notice that the objects that share some common visual characteristics are grouped together. For instance, we notice that most of the 'Cow' class images goes whether to the 'Dog' class or the 'Horse' class. The same behavior happens for the 'Tomato' and 'Apple' classes. This explains why the accuracy of the fruits vs non fruits model is high at the beginning when we use 8 clusters. Indeed, if a 'Tomato' object is considered as an 'Apple', it remains within the fruits true positives and is not considered as a wrong classification for the *super class* fruits. A high increase in the number of clusters at the end does not have much effect on the classification results as the used features are not discriminative anymore. Hence, the difference in accuracy when using 20 clusters and when considering 60 clusters is almost 8% and the difference between using 80 clusters and 90 clusters is almost 1.5% only. This is expected as the algorithm starts to deal with hard cases where the dissimilarities are not detected anymore between objects and it fails to construct more distinguishable clusters. Nevertheless, looking at the confusion matrix in Table 2.12 where we considered 90 clusters, shows clearly that the algorithm distinguishes well between animals, objects and fruits, and considerably outperforms the classical model.

2.5.3 Discussion

It is clear that the obtained results when using multi clusters to represent an object class outperforms the results when modeling each object class by one single cluster. The more we have increased the number of clusters the more we have obtained better results as the overlapping features

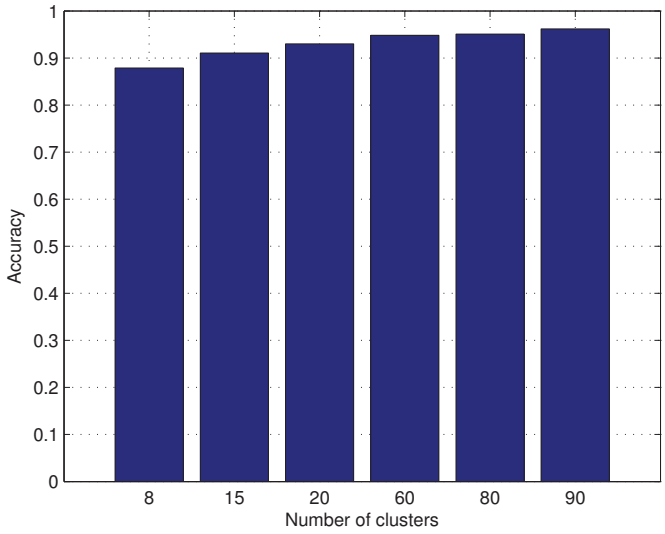


Figure 2.14: Classification accuracy of Fruits vs Non Fruits classes in % as a function of the total used number of clusters.

Table 2.7: 8 clusters confusion matrix.

	Apple	Tomato	Pear	Cow	Dog	Horse	Car	Cup
Apple	49.51%	20.24%	-	-	-	-	-	13.90%
Tomato	42.19%	33.41%	-	-	11.21%	-	-	-
Pear	16.34%	11.95%	30.97%	-	12.68%	27.80%	-	-
Cow	-	-	-	0%	50%	37.56%	11.46%	-
Dog	-	-	-	-	61.70%	31.46%	-	-
Horse	-	-	-	-	40.48%	49.75%	-	-
Car	-	-	-	-	10.73%	-	78.04%	-
Cup	10.97%	-	-	-	-	-	30.48%	50.73%

Table 2.8: 15 clusters confusion matrix.

	Apple	Tomato	Pear	Cow	Dog	Horse	Car	Cup
Apple	57.07%	35.36%	-	-	-	-	-	-
Tomato	10.73%	81.95%	-	-	-	-	-	-
Pear	-	10%	64.87%	-	11.46%	-	-	-
Cow	-	-	-	8.29%	36.58%	30.48%	19.02%	-
Dog	-	-	-	-	57.07%	21.70%	-	-
Horse	-	-	-	10.24%	37.07%	36.09%	-	-
Car	-	-	-	-	-	-	85.85%	-
Cup	13.17%	-	-	-	-	-	26.09%	51.95%

Table 2.9: 20 clusters confusion matrix.

	Apple	Tomato	Pear	Cow	Dog	Horse	Car	Cup
Apple	67.80%	25.36%	-	-	-	-	-	-
Tomato	10.73%	86.34%	-	-	-	-	-	-
Pear	-	-	70%	-	10.97%	-	-	-
Cow	-	-	-	23.65%	34.14%	31.21%	-	-
Dog	-	-	-	16.34%	51.70%	25.60%	-	-
Horse	-	-	-	13.90%	35.12%	38.29%	-	-
Car	-	-	-	-	-	-	78.78%	-
Cup	-	-	-	-	-	-	16.82%	66.82%

Table 2.10: 60 clusters confusion matrix.

	Apple	Tomato	Pear	Cow	Dog	Horse	Car	Cup
Apple	77.56%	19.02%	-	-	-	-	-	-
Tomato	11.70%	83.17%	-	-	-	-	-	-
Pear	-	-	86.34%	-	-	-	-	-
Cow	-	-	-	44.39%	26.09%	19.02%	-	-
Dog	-	-	-	24.63%	47.80%	18.04%	-	-
Horse	-	-	-	27.31%	28.29%	35.60%	-	-
Car	-	-	-	-	-	-	87.56%	-
Cup	-	-	-	-	-	-	-	85.36%

Table 2.11: 80 clusters confusion matrix.

	Apple	Tomato	Pear	Cow	Dog	Horse	Car	Cup
Apple	74.63%	20.73%	-	-	-	-	-	-
Tomato	-	87.31%	-	-	-	-	-	-
Pear	-	-	84.87%	-	-	-	-	-
Cow	-	-	-	41.95%	25.85%	24.87%	-	-
Dog	-	-	-	22.68%	49.51%	21.95%	-	-
Horse	-	-	-	27.31%	25.60%	41.46%	-	-
Car	-	-	-	-	-	-	92.68%	-
Cup	-	-	-	-	-	-	-	91.46%

Table 2.12: 90 clusters confusion matrix.

	Apple	Tomato	Pear	Cow	Dog	Horse	Car	Cup
Apple	78.53%	18.53%	-	-	-	-	-	-
Tomato	-	85.36%	-	-	-	-	-	-
Pear	-	-	84.87%	-	-	-	-	-
Cow	-	-	-	43.90%	24.39%	27.56%	-	-
Dog	-	-	-	24.63%	48.04%	22.92%	-	-
Horse	-	-	-	25.60%	24.14%	46.58%	-	-
Car	-	-	-	-	-	-	93.41%	-
Cup	-	-	-	-	-	-	-	93.65%

vectors start to be assigned to different clusters and the visually similar objects are more distinguished. We notice that at the beginning, when increasing the number of modeled clusters, the classification accuracy is improved tremendously and then it starts to stabilize when the maximum possible number of modeled clusters is achieved as the algorithm finds more difficulties to distinguish the features differentiating certain images, and it ends up by achieving a maximum possible number of clusters that can represent the data. One possible way to improve these results is to perform a feature selection algorithm so it can discard the visual similarities between different objects and focus on the discriminating features at the lowest level of the hierarchy. Still, our proposed approach outperforms the classic one. One interesting point as well is that the hierarchical model can be changed easily once we have the clusters representing the lowest level of the hierarchy. This is useful when we have a dynamic hierarchy based on the user desire. Consider for example search engines, or recommendation systems that have to represent a given hierarchy according to the user's needs who might change his/her mind during time. Also one single computation of the bottom representation of the data can serve many users according to their request in a hierarchical fashion. The user can build up on the fly any model that is computed in a costless effort in terms of resources and time. Note that for any hierarchy we just estimate the parameters of $\sum_{j=1}^{M} K_j$ densities and it is only the weights of the higher levels that change according to the ontology or hierarchical model. This is very practical as this approach helps to reduce the distance between the systems level features and the human semantic and understanding. The system level features similarities are not necessarily the same of what a user might have in mind or comprehend. Thus, the first generation of the model at its bottom level can be made off line, and then the representation of the data is done according the users needs with costless computational time.

2.6 Summary

We proposed a flexible hierarchical model that can be altered on the fly according to a given ontology. We have used the inverted Dirichlet distribution to develop the model, still, it remains general enough to integrate any chosen probability density function. The merit of the approach that we have proposed is shown through the application of our algorithm on synthetic and real data. In the next chapter, we will introduce a building methodology to the mixture model and propose a model update strategy when new data are introduced to the system on line.

Chapter 3

A Statistical Framework for Online Learning Using Adjustable Model Selection Criteria

3.1 Introduction

One of the most important success keys to design a system that is dedicated to serve human beings, is to ensure that the system's responses are approaching the satisfaction of the real needs and intentions of its users. Designing such systems is a challenging task especially when their responses should change according to the users interactions that differ from one user to another, which discards the option of designing predefined/deterministic systems' behaviors [77]. A typical example of those systems are the search engines and object recognition applications where a user is looking for an object of interest among a huge amount of data [78]. Naturally a given property of data/object can be perceived by a user as being "interesting" while it could be "meaningless" for another user, also, an object of interest can be a target for two different users who have different behavior patterns, which should lead the system to close results using different interactions patterns. During the recent years, and with the tremendous evolution of multimedia devices, these users interactions/behaviors can generate a huge amount of data e.g. via internet through recommendation systems and users feedbacks [79, 80]. Many researchers considered model-based approaches to analyze users-systems interactions, such as in [80, 81]. Indeed, this family of approaches has been used extensively in classification and users feedbacks modeling, in order to feed inference engines that are capable of generating knowledge which enables a given system to learn and model the users' needs. Still, a semantic gap between the system representation of data, and the user mental representation of the same data, is considerably affecting the quality of the responses of the systems that are not usually aware of the users intentions when representing the data in the features' space. Adopting a model-based approach to solve this problem, we have concluded in chapter 2 that the

system should not represent a given class via a single mixture mode but rather model it using different components in a flexible hierarchical way. In this chapter we treat the problem of new coming data and how it should be perceived by the system in order to be able to satisfy the users needs. This issue is crucial as discussed in different research works (see, for example, [82, 83]). Consider, for instance, that the system receives new coming data online that should update the model at the system level without loosing the flexibility and the degree of dissimilarity/similarity perceived by the users. Thus, the system should take an important decision whether new classes should be created to represent the new coming data or not. In unsupervised learning, many model selection criteria have been developed such as the minimum description length (MDL) [52] and minimum message length (MML) [24] criteria. Also, the model selection problem was treated through a nonparametric Bayesian technique namely the Dirichlet Process (DP) by assuming that there is an infinite number of mixture components such as in [84]. Yet, the current use of these approaches does not incorporate a semantic meaning to the mixture components in the selected model that does not necessarily fit with a human comprehensible semantic or intentions. We propose in this chapter to develop an approach that integrates a perception parameter that helps the system to take appropriate decisions to model data as intended. Indeed, the system should learn how to model data according to a certain degree of similarity/dissimilarity tolerance. For example, consider that we have a class representing red apples, and the new coming data are representing tomatoes, should the system consider the red apples and tomatoes as being one single class, or differ between them by creating a new class for the tomatoes. Naturally the decision (creating a new class or not) depends on the application itself and the purposes of the users. In this chapter, we propose a new methodology that can control how the system should perceive new coming data over time. We propose a statistical framework based on the generalized inverted Dirichlet (GID) distribution [85] which is the generalization of the inverted Dirichlet (ID) distribution that we have considered in chapter 2. It is noteworthy to mention that the proposed framework can consider any other distribution. However, we consider the GID distribution because of an interesting property that enables the transformation of its representation into a space where the features are independent and follow each an inverted Beta distribution. We propose in this chapter an algorithm to learn a mixture of GID, update it and create new components depending on the users perception. We perform different simulations on synthetic and real data in order to validate our methodology.
The rest of this chapter is organized as follows; in section 3.2 we introduce the GID mixture model. Then, we propose an estimation algorithm for its parameters by considering both batch and online settings in section 3.3. In section 3.4 we define some model selection criteria for choosing the adequate number of components of a GID mixture using an unsupervised approach. Then, in section 3.5 we introduce a new updating scheme for a growing mixture whose number of components can

increase according to users perceptions, we define some dissimilarity/similarity metrics, and we introduce the complete algorithm. We introduce our experimental results on synthetic and real data in section 3.6, and we conclude this chapter in section 3.7.

3.2 Finite Generalized Inverted Dirichlet Mixture Model

Let us consider a set $\mathscr{Y}$ of N D-dimensional vectors, such that $\mathscr{Y} = (\vec{Y}_1, \vec{Y}_2, \ldots, \vec{Y}_N)$. Let M denote the number of different components forming a flat mixture model [17, 86] at the system level. We assume that $\mathscr{Y}$ is controlled by a mixture of GID distributions such that the vectors follow a common probability density function $p(\vec{Y}_i|\Theta)$, where Θ is the set of its parameters. Let $Z = \{\vec{Z}_1, \vec{Z}_2, \ldots, \vec{Z}_N\}$ denote the missing group indicator, where $\vec{Z}_i = (z_{i1}, z_{i2}, \ldots, z_{iM})$ is the label of $\vec{Y}_i$, such that $z_{ij} \in \{0,1\}$, $\sum_{j=1}^{M} z_{ij} = 1$ and z_{ij} is equal to one if $\vec{Y}_i$ belongs to class j and zero, otherwise. Then, the distribution of $\vec{Y}_i$ given the class label $\vec{Z}_i$ is :

$$p(\vec{Y}_i|\vec{Z}_i, \Theta) = \prod_{j=1}^{M} p(\vec{Y}_i|\theta_j)^{z_{ij}} \tag{3.1}$$

with $\Theta = \{\theta_1, \theta_2, \ldots, \theta_M\}$ and θ_j is the set of parameters of the class j. In practice, we define $p(\vec{Y}_i|\Theta)$ which can be obtained by marginalizing the complete likelihood $p(\vec{Y}_i, \vec{Z}_i|\Theta)$ over the hidden variables. We define the prior distribution of $\vec{Z}_i$ as follows :

$$p(\vec{Z}_i|\vec{\pi}) = \prod_{j=1}^{M} \pi_j^{z_{ij}} \tag{3.2}$$

where $\vec{\pi} = (\pi_1, \ldots, \pi_M)$, $\pi_j > 0$ and $\sum_{j=1}^{M} \pi_j = 1$, then we have:

$$p(\vec{Y}_i, \vec{Z}_i|\Theta, \vec{\pi}) = p(\vec{Y}_i|\vec{Z}_i, \Theta)P(\vec{Z}_i|\vec{\pi}) = \prod_{j=1}^{M} (p(\vec{Y}_i|\theta_j)\pi_j)^{z_{ij}} \tag{3.3}$$

We proceed by the marginalization of Eq. 3.3 over the hidden variables, which gives us the mixture for a given vector $\vec{Y}_i$:

$$p(\vec{Y}_i|\Theta, \vec{\pi}) = \sum_{j=1}^{M} p(\vec{Y}_i|\theta_j)\pi_j \tag{3.4}$$

The GID was introduced by Lingappaiah [87] as follows:

$$p(\vec{Y}_i|\theta_j) = \prod_{l=1}^{D} \frac{\Gamma(\alpha_{jl} + \beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} \frac{Y_{il}^{\alpha_{jl}-1}}{(1 + \sum_{k=1}^{l} Y_{ik})^{\gamma_{jl}}} \tag{3.5}$$

By substituting Eq. 3.5 in Eq. 3.4, we obtain :

$$p(\mathscr{Y}|\Theta,\vec{\pi}) = \prod_{i=1}^{N}\Big(\sum_{j=1}^{M}\pi_j \prod_{l=1}^{D}\frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})}\frac{Y_{il}^{\alpha_{jl}-1}}{(1+\sum_{k=1}^{l}Y_{ik})^{\gamma_{jl}}}\Big) \tag{3.6}$$

where $\Theta = (\theta_1,\theta_2,...,\theta_M)$, with $\theta_j = \{\theta_{j1},\theta_{j2},...,\theta_{jD}\}$, where $\theta_{jl} = (\alpha_{jl},\beta_{jl})$, $j = 1,...,M$, $l = 1,...,D$. We define γ_{jl} such that $\gamma_{jl} = \beta_{jl} + \alpha_{jl} - \beta_{j(l+1)}$ for $l = 1,\ldots,D$ with $\beta_{j(D+1)} = 0$.

The purpose of this chapter is to establish a way to 1) estimate the GID mixture's parameters, 2) update these parameters when new data are introduced, 3) select the optimal number of classes to consider when new data are introduced, 4) consider an infinite possible number of classes when we have a stream of new data that can form new classes that were not considered at the beginning. These new classes can be set up by the user or the system.

3.3 Parameters Estimation

The posterior probabilities are a key factor to cluster data and classify it in mixture model-based representations of data. Indeed maximizing the posterior probabilities is considered an intuitive rule to assign data to its appropriate clusters. The GID has an interesting property enabling the factorization of its posterior probability such that : (See Appendix B.1)

$$p(j|\vec{Y}_i,\Theta,\vec{\pi}) \propto p_j \prod_{l=1}^{D} p_{iBeta}(X_{il}|\theta_{jl}) \tag{3.7}$$

where we have set $X_{i1} = Y_{i1}$ and $X_{il} = \frac{Y_{il}}{1+\sum_{k=1}^{l-1}Y_{ik}}$ for $l > 1$. $p_{iBeta}(X_{il}|\theta_{jl})$ is an inverted Beta distribution with parameters $\theta_{jl} = (\alpha_{jl},\beta_{jl})$, $l = 1,\ldots,D$, such that :

$$p_{iBeta}(X_{il}|\theta_{jl}) = \frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} X_{il}^{\alpha_{jl}-1}(1+X_{il})^{-(\alpha_{jl}+\beta_{jl})} \tag{3.8}$$

Thus, the clustering structure underlying $\mathscr{Y}$ is the same as the one underlying $\mathscr{X} = \{\vec{X}_1,\vec{X}_2,...$ $...,\vec{X}_i\}$, where $\vec{X}_i = \{X_{i1},X_{i2},...,X_{iD}\}$, $i = 1,...,N$, governed by the following mixture model with conditionally independent features

$$p(\mathscr{X}|\Theta,\vec{\pi}) = \prod_{i=1}^{N}\Big(\sum_{j=1}^{M}\pi_j \prod_{l=1}^{D} p_{iBeta}(X_{il}|\theta_{jl})\Big) \tag{3.9}$$

The estimation of the parameters in Eq. 3.6 is then equivalent to the estimation of the parameters in Eq. 3.9.

3.3.1 Log-Likelihood of the Complete Data

The model for the complete data $< \mathscr{X}, Z >$ is given by :

$$p_c(\mathscr{X}, Z, |\Theta, \vec{\pi}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \pi_j^{z_{ij}} \prod_{l=1}^{D} p_{iBeta}(X_{il}|\theta_{jl})^{z_{ij}} \tag{3.10}$$

We maximize the log-likelihood instead of the likelihood. The log-likelihood is given by :

$$\Phi_c(\mathscr{X}, Z|\Theta, \vec{\pi}) = \log(p(\mathscr{X}, Z|\Theta, \vec{\pi}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{l=1}^{D} z_{ij}(\log(\pi_j) + \log p_{iBeta}(\vec{X}_i|\theta_{jl})) \tag{3.11}$$

We define θ_j as being the set of parameters describing the component j, such that $\theta_j = \{\theta_{j1}, \theta_{j2}, ...$ $..., \theta_{jD}\}$, $\theta_{jl} = (\alpha_{jl}, \beta_{jl})$, $j = 1, \ldots, M$ and $l = 1, \ldots, D$. In order to estimate the parameters we use the EM algorithm [19] which proceeds iteratively in two steps; the expectation (E) step and the maximization (M) step. In the E-step, we compute the conditional expectation of $\Phi_c(\mathscr{X}, Z, |\Theta, \vec{\pi})$ which is reduced to the computation of the posterior probabilities (i.e. the probability that a vector $\vec{X}_i$ is assigned to a cluster j) :

$$p(j|\vec{X}_i, \Theta, \vec{\pi}) = \frac{\pi_j p(\vec{X}_i|\theta_j)}{\sum_{j=1}^{M} \pi_j p(\vec{X}_i|\theta_j)} = \frac{\pi_j \prod_{l=1}^{D} p_{iBeta}(X_{il}|\theta_{jl})}{\sum_{j=1}^{M} \pi_j \prod_{l=1}^{D} p_{iBeta}(X_{il}|\theta_{jl})} \tag{3.12}$$

Thus, we have :

$$\log p(\mathscr{X}|\Theta, \vec{\pi}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{l=1}^{D} p(j|\vec{X}_i, \Theta, \vec{\pi}) \Big(\log(\pi_j) + \log p_{iBeta}(\vec{X}_i|\theta_{jl}) \Big) \tag{3.13}$$

Then, the conditional expectation of the complete-data log likelihood is given by :

$$Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda) = \log p(\mathscr{X}|\Theta, \vec{\pi}) + \lambda(1 - \sum_{j=1}^{M} \pi_j) \tag{3.14}$$

where λ is the Lagrange multiplier.

3.3.2 The EM Algorithm

As we are trying to maximize Eq. 3.14, we proceed by finding the roots of its derivations with respect to our parameters. The mixtures weights can be estimated straightforwardly as follows :

$$\pi_j = \frac{\sum_{i=1}^{N} p(j|\vec{X}_i, \Theta, \vec{\pi})}{N} \tag{3.15}$$

Calculating the derivative with respect to α_{jl} and β_{jl} , $j = 1, \ldots, M$, $l = 1, \ldots, D$, we obtain :

$$\begin{aligned} \frac{\partial Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \alpha_{jl}} &= \sum_{i=1}^{N} p(j|\vec{X}_i, \Theta, \vec{\pi}) \frac{\partial \log p_{iBeta}(\vec{X}_i|\theta_{jl})}{\partial \alpha_{ji}} \\ &= \sum_{i=1}^{N} p(j|\vec{X}_i, \Theta, \vec{\pi}) \left(\Psi(\alpha_{jl} + \beta_{jl}) - \Psi(\alpha_{jl}) + \log(\frac{X_{il}}{1 + X_{il}}) \right) \end{aligned} \tag{3.16}$$

$$\begin{aligned} \frac{\partial Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \beta_{jl}} &= \sum_{i=1}^{N} p(j|\vec{X}_i, \Theta, \vec{\pi}) \frac{\partial \log p_{iBeta}(\vec{X}_i|\theta_{jl})}{\partial \beta_{jl}} \\ &= \sum_{i=1}^{N} p(j|\vec{X}_i, \Theta, \vec{\pi}) \left(\Psi(\alpha_{jl} + \beta_{jl}) - \Psi(\beta_{jl}) + \log(\frac{1}{1 + X_{il}}) \right) \end{aligned} \tag{3.17}$$

where $\Psi(.)$ is the digamma function. Looking at the previous two equations, it is clear that an explicit form of the solution to estimate θ_{jl} does not exist. Thus, we refer to Newton Raphson method expressed as:

$$\theta_{jl}^{new} = \theta_{jl}^{old} - H_{jl}^{-1} G_{jl} \quad , j = 1, \ldots, M, \quad l = 1, \ldots, D \tag{3.18}$$

where H_{jl} is the Hessian matrix associated with $Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)$:

$$H_{jl} = \begin{pmatrix} \frac{\partial^2 Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \alpha_{jl}^2} & \frac{\partial^2 Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \alpha_{jl} \partial \beta_{jl}} \\ \frac{\partial^2 Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \alpha_{jl} \partial \beta_{jl}} & \frac{\partial^2 Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \beta_{jl}^2} \end{pmatrix} \tag{3.19}$$

and G_{ji} is the first derivatives vector:

$$G_{jl} = \left(\frac{\partial Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \alpha_{jl}}, \frac{\partial Q(\mathscr{X}, \Theta, \vec{\pi}, \Lambda)}{\partial \beta_{jl}} \right) \tag{3.20}$$

To calculate the Hessian of $Q(\mathscr{X},\Theta,\vec{\pi},\Lambda)$ we have to compute the second and mixed derivatives:

$$\frac{\partial^2 Q(\mathscr{X},\Theta,\vec{\pi},\Lambda)}{\partial \alpha_{jl}^2} = \sum_{i=1}^{N} p(j|\vec{X}_i,\Theta,\vec{\pi})\Big(\Psi'(\alpha_{jl}+\beta_{jl}) - \Psi'(\alpha_{jl})\Big) \tag{3.21}$$

$$\frac{\partial^2 Q(\mathscr{X},\Theta,\vec{\pi},\Lambda)}{\partial \beta_{jl}^2} = \sum_{i=1}^{N} p(j|\vec{X}_i,\Theta,\vec{\pi})\Big(\Psi'(\alpha_{jl}+\beta_{jl}) - \Psi'(\beta_{jl})\Big) \tag{3.22}$$

$$\frac{\partial^2 Q(\mathscr{X},\Theta,\vec{\pi},\Lambda)}{\partial \alpha_{jl}\partial \beta_{jl}} = \sum_{i=1}^{N} p(j|\vec{X}_i,\Theta,\vec{\pi})\Big(\Psi'(\alpha_{jl}+\beta_{jl})\Big) \tag{3.23}$$

where $\Psi'(.)$ is the trigamma function.

3.3.3 Online Estimation

Establishing an algorithm for online learning is important in many real life applications [88, 89]. For learning online we consider a factorized GID mixture, as expressed in Eq. 3.24. We assume in this section that the number of components M composing the mixture remains the same during time, which means that the coming vectors cannot form new classes and should be affected to the classes already modeled. Naturally the parameters of those classes should be updated accordingly. We assume that at a given time t we have a set a dataset $\mathscr{X} = \{\vec{X}_1,\vec{X}_2,\ldots,\vec{X}_N\}$ of N feature vectors which is represented by M-components mixture of GID distributions:

$$p(\mathscr{X}|\Theta^{(t)}_{N^{(t)}},\vec{\pi}^{(t)}_{N^{(t)}}) = \prod_{i=1}^{N}\Big(\sum_{j=1}^{M}\pi^{(t)}_{N^{(t)}_j}\prod_{l=1}^{D} p_{iBeta}(X_{il}|\theta^{(t)}_{N^{(t)}_{jl}})\Big) \tag{3.24}$$

with $\Theta^{(t)}_{N^{(t)}} = \{\theta^{(t)}_{N^{(t)}{}_1},\ldots,\theta^{(t)}_{N^{(t)}{}_M}\}$, is the set of parameters such that $\theta^{(t)}_{N^{(t)}{}_j} = \{\theta^{(t)}_{N^{(t)}{}_{j1}},\ldots,\theta^{(t)}_{N^{(t)}{}_{jl}}\}$, $j = 1...M$, $l = 1,\ldots,D$. At time $t+1$, a new data vector is introduced, and the model should be updated accordingly. For the ease of notification, we let $\Theta^{(t)} = \Theta^{(t)}_{N^{(t)}}$, $\theta^{(t)}_1 = \theta^{(t)}_{N^{(t)}{}_1}$ and $\pi^{(t)}_j = \pi^{(t)}_{N^{(t)}{}_j}$. Naturally we have to keep the constraints $0 < \pi^{(t)}_j \leq 1$ and $\sum_{j=1}^{M}\pi^{(t)}_j = 1$. Considering this fact, we propose to update the parameters as follows [88, 90]

$$\begin{aligned}
\pi^{(t+1)}_j &= \frac{\exp(w(j)^{(t+1)})}{1+\sum_{j=1}^{M-1}\exp(w(j)^{(t+1)})}, \quad j = 1,\ldots,M-1 \\
\pi^{(t+1)}_M &= 1 - \sum_{j=1}^{M-1}\pi^{(t+1)}_j
\end{aligned} \tag{3.25}$$

such that,

$$w(j)^{(t+1)} = w(j)^{(t)} + \gamma_N(\hat{Z}_{N+1_j} - \pi_j^{(t)}) \tag{3.26}$$

$w(j)$ is considered to ensure the unity of mixing proportion π_j by introducing the logit transform $w(j) = \log(\frac{\pi_j}{\pi_M})$. $\hat{Z}_{N+1_j}$ is the posterior probability of the new coming vector $\vec{X}_{N+1}$ given the set of parameters $\Theta^{(t)}$:

$$\hat{Z}_{N+1_j} = \frac{\pi_j^{(t)} p(\vec{X}_{N+1}|\theta_j^{(t)})}{\sum_{j=1}^{M} \pi_j^{(t)} p(\vec{X}_{N+1}|\theta_j^{(t)})} \tag{3.27}$$

γ_N is a sequence of positive number that decreases to zero such that $\sum|\gamma_n| = \infty$ and $\sum|\gamma_n|^2 < \infty$ [90]. In our case, we have chosen $\gamma_N = \frac{1}{N+1}$ and the parameters $\{\theta_{jl}\}$ are updated as follows :

$$\theta_{jl}^{(t+1)} = \theta_{jl}^{(t)} + \frac{\hat{Z}_{N+1_j}}{N+1} \frac{\partial \log p_{iBeta}(\vec{X}_{N+1}, \vec{Z}_{N+1}|\theta_{jl})}{\partial \theta_{jl}} \tag{3.28}$$

which gives us :

$$\begin{pmatrix} \alpha_{jl}^{(t+1)} \\ \beta_{jl}^{(t+1)} \end{pmatrix} = \begin{pmatrix} \alpha_{jl}^{(t)} \\ \beta_{jl}^{(t)} \end{pmatrix} + \frac{\hat{Z}_{N+1_j}}{N+1} \begin{pmatrix} \Psi(\alpha_{jl}+\beta_{jl}) - \Psi(\alpha_{jl}) + \log(\frac{X_{(N+1)l}}{1+X_{(N+1)l}}) \\ \Psi(\alpha_{jl}+\beta_{jl}) - \Psi(\beta_{jl}) + \log(\frac{1}{1+X_{(N+1)l}}) \end{pmatrix} \tag{3.29}$$

3.4 Model Selection

When modeling data using a finite mixture, the model selection process is considered as a crucial phase to select the adequate number of components forming the mixture in terms of best representation for data. Model selection in the case of mixtures is mainly based on the evaluation of some criteria that would specify the“best" model that should be selected. As we have discussed before, we aim at integrating a perception parameter that helps the system to better represent the data in terms of reducing the gap between the human being understandable level and the data representation at the system level. To do so, we propose to use system level criteria and users perception criteria (that we shall develop in the next sections), in the same model. As for the system critera, we consider five criteria namely minimum message length (MML) [24], Akaike information criterion (AIC) [25], minimum description length (MDL) [26], mixture MDL (MMDL) [27], and LEC [17]. The "best" model that should be selected would be the model minimizing those criteria. The message length is given by [24] :

$$MML(\mathscr{X}, \Theta, M) = -\log(h(\Theta)) + \frac{1}{2}\log(|F(\Theta)|) + \frac{N_p}{2}(1 + \log(\frac{1}{12})) - \log(p(\mathscr{X}|\Theta, \vec{\pi})) \tag{3.30}$$

where $h(\Theta)$ is the prior probability, $p(\mathscr{X}|\Theta)$ is the likelihood, $F(\Theta)$ is the expected Fisher information matrix which is approximated by the complete-data Fisher information matrix, $|F(\Theta)|$ is its determinant, and N_p is the number of the estimated parameters which is equal to $M(2D+1)-1$ in the case of the GID mixture model. The LEC is given by [17]:

$$LEC(\mathscr{X},\Theta,M) = -\log(h(\Theta)) + \frac{1}{2}\log(|F(\Theta)|) - \frac{N_p}{2}\log(2\pi) - \log(p(\mathscr{X}|\Theta,\vec{\pi})) \tag{3.31}$$

The MMDL is given by [27]:

$$MMDL(\mathscr{X},\Theta,M) = -\log(p(\mathscr{X}|\Theta,\vec{\pi})) + \frac{N_p}{2}\log(N) + \frac{c}{2}\sum_{j=1}^{M}\log((\pi_j)) \tag{3.32}$$

where $c = 2D+1$ is the number of parameters describing each component. The MDL is given by [26]:

$$MDL(\mathscr{X},\Theta,M) = -\log(p(\mathscr{X}|\Theta,\vec{\pi})) + \frac{N_p}{2}\log(N) \tag{3.33}$$

The AIC is given by [25]:

$$AIC(\mathscr{X},\Theta,M) = -\log(p(\mathscr{X}|\Theta,\vec{\pi})) + \frac{N_p}{2} \tag{3.34}$$

To find the explicit expression of the MML and LEC criteria in the case of the GID mixture model, we use the same approach as in [91], so we can show that :

$$\log(h(\Theta)) = \sum_{j=1}^{M-1}\log(j) - 10MD - 2MD\log(2D) + M\sum_{j=1}^{2D}\log(j) \tag{3.35}$$

and

$$\begin{aligned}\log(F|\Theta|) = {} & (M-1)\log(N) - \sum_{j=1}^{M}\log(\pi_j) + 2D\sum_{j=1}^{M}\log(n_j) \\ & + \sum_{j=1}^{M}\sum_{l=1}^{D}\log\left(\left|\Psi'(\alpha_{jl})\Psi'(\beta_{jl}) - \Psi'(\alpha_{jl}+\beta_{jl})\left(\Psi'(\alpha_{jl})+\Psi'(\beta_{jl})\right)\right|\right)\end{aligned} \tag{3.36}$$

3.5 A Novel Online Learning Approach for Growing Mixture Models

In this section we present a novel method to learn a GID mixture whose number of components can increase over time. We propose to build our framework using the EM algorithm, a selection

criterion namely the MML [24], and probabilistic metrics such as the symmetric Kullback-Leibler divergence distance [92–94]. We consider that at time t we have a GID mixture model, and at time $t+1$ a set of new data arrives at once leading to the update of the model accordingly. In this section we consider that we have a set of arriving data and we propose to update the model using the whole set, as it has been shown in [95, 96] that it is better to consider multiple observations per update in order to reduce noise and have better parameters estimates that reflect a good representation of the data. Mathematically speaking, if we suppose that at time t we have a dataset $\mathscr{X}^{(t)}_{N^{(t)}} = \{\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_{N^{(t)}}\}$ following a GID mixture model, with parameters $(\Theta^{(t)}_{N^{(t)}}, \vec{\pi}^{(t)}_{N^{(t)}})$. At time $(t+1)$ a new dataset $\mathscr{X}^{(t+1)}_{L^{(t+1)}} = \{\vec{X}_{N^{(t)}+1}, \vec{X}_{N^{(t)}+2}, \ldots, \vec{X}_{N^{(t)}+L^{(t+1)}}\}$ arrives and the system should update the parameters $\Theta^{(t)}_{N^{(t)}}$ accordingly. The whole dataset at time $(t+1)$ becomes :

$$\mathscr{X}^{(t+1)}_{N^{(t+1)}} = \mathscr{X}^{(t+1)}_{N^{(t)}+L^{(t+1)}} = \mathscr{X}^{(t)}_{N^{(t)}} \cup \mathscr{X}^{(t+1)}_{L^{(t+1)}} \tag{3.37}$$

Here we assume three scenarios when a new set of data arrives:

- The new coming data is totally formed by the already modeled classes, so the problem is reduced to an update problem as discussed in the previous section.
- The new coming data is composed from ancient classes and new non modeled ones, so the parameters should be updated accordingly by updating the ancient classes and forming the new classes.
- The new coming data is totally composed from new classes, so the model should add the new classes to the ancient ones and update the model accordingly.

When designing our framework we should consider and handle these three cases. So, if we reconsider our mathematical model, we suppose that at time t we have a dataset $\mathscr{X}^{(t)}_{N^{(t)}}$ composed of $N^{(t)}$ vectors that follow the factorized distribution of a GID mixture composed of $M^{(t)}_{N^{(t)}}$ components such that:

$$p(\mathscr{X}^{(t)}_{N^{(t)}} | \Theta^{(t)}_{N^{(t)}}, \vec{\pi}^{(t)}_{N^{(t)}}) = \prod_{i=1}^{N^{(t)}} \left(\sum_{j=1}^{M^{(t)}_{N^{(t)}}} \pi^{(t)}_{N^{(t)}{}_j} \prod_{l=1}^{D} p_{iBeta}(X_{il} | \theta^{(t)}_{N^{(t)}{}_{jl}}) \right) \tag{3.38}$$

At time $(t+1)$ a new dataset $\mathscr{X}^{(t+1)}_{L^{(t+1)}}$ composed of $L^{(t+1)}$ vectors is introduced to the system. Note that $L^{(t+1)}$ can change over time. The new size of the whole data is $N^{(t+1)} = N^{(t)} + L^{(t+1)}$. One important assumption in our work is that we suppose that the new data $\mathscr{X}^{(t+1)}_{L^{(t+1)}}$ will contain at minimum one single class, and at maximum $(M^{(t+1)}_{L^{(t+1)}})_{max}$ classes. $(M^{(t+1)}_{L^{(t+1)}})_{max}$ can be defined according to the size of the new dataset $L^{(t+1)}$, and the membership frequency of the coming data.

Thus we have,

$$p(\mathscr{X}_{L^{(t+1)}}^{(t+1)}|\Theta_{L^{(t+1)}}^{(t+1)},\vec{\pi}_{L^{(t+1)}}^{(t+1)}) = \prod_{n=N^{(t)}+1}^{N^{(t)}+L^{(t+1)}} \left(\sum_{j=1}^{M_{L^{(t+1)}}^{(t+1)}} (\pi_{L^{(t+1)}}^{(t+1)})_j \prod_{l=1}^{D} p_{iBeta}(X_{il}|\theta_{L^{(t+1)}\,jl}^{(t+1)}) \right) \tag{3.39}$$

such that :

$$(\Theta_{L^{(t+1)}}^{(t+1)}, M_{L^{(t+1)}}^{(t+1)}) \in \left\{ \left((\Theta_{L^{(t+1)}}^{(t+1)})^1, 1\right), \ldots, \left((\Theta_{L^{(t+1)}}^{(t+1)})^{(M_{L^{(t+1)}}^{(t+1)})_{max}}, (M_{L^{(t+1)}}^{(t+1)})_{max}\right) \right\} \tag{3.40}$$

It is noteworthy to mention that the notation we use enables the model to consider different scenarios and makes it considerably flexible. For instance the maximum number of components $(M_{L^{(t+1)}}^{(t+1)})_{max}$ that can represent a given dataset can change over time for the same size L. This enables our model to be parametrized according to the factors we want to consider (e.g. frequency of arrival of data, learning strategy, previous modeling experience, etc). As we suppose that the new coming data can be composed of a number of classes ranging from 1 to $(M_{L^{(t+1)}}^{(t+1)})_{max}$, we run the algorithm $(M_{L^{(t+1)}}^{(t+1)})_{max}$ times, incrementing the number of components at each time, and we end up by having $(M_{L^{(t+1)}}^{(t+1)})_{max}$ models for the new data. In order to consider the three cases that we mentioned previously, we set a rule that determines how the system should perceive the similarity/dissimilarity between the obtained $M_{L^{(t+1)}}^{(t+1)}$ components of the new data and the $M_{N^{(t)}}^{(t)}$ components representing the old data. Let d denote a metric in the distributions space measuring the dissimilarity. If we denote $S\left((\theta_{L^{(t+1)}}^{(t+1)})_i, (\theta_{N^{(t)}}^{(t)})_j\right)$ the degree of similarity between the component i of the new data and the component j of the old data, it can be expressed as follows:

$$S\left((\theta_{L^{(t+1)}}^{(t+1)})_i, (\theta_{N^{(t)}}^{(t)})_j\right) = \frac{\Phi_s\left(d\left((\theta_{L^{(t+1)}}^{(t+1)})_i, (\theta_{N^{(t)}}^{(t)})_j\right)\right)}{\sum_{j=1}^{M_{N^{(t)}}^{(t)}} \Phi_s\left(d\left((\theta_{L^{(t+1)}}^{(t+1)})_i, (\theta_{N^{(t)}}^{(t)})_j\right)\right)} \tag{3.41}$$

where Φ_s is the perceived similarity between two different components that should be inversely proportional to their distance considering the metric d which means that Φ_s is monotonically decreasing in d.

We consider that when two sets of vectors are represented by “similar" distributions, they can be considered to be generated from the same distribution. The decision that the system takes to consider that two distributions are similar defines the way how the data are perceived. As we mentioned before, the purpose / nature / context of the application may have an important impact on the system responses. Leaning on the feature space only to take decision about objects may lead

to inappropriate responses that do not satisfy the user needs. In some applications, the differentiation between very similar objects should take place, even if they are classified in the same class in the feature space, and in some other applications, very dissimilar objects should be considered as belonging to the same class of objects for a user. For each component modeled by $(\theta^{(t)}_{N^{(t)}})_j$ we consider a saturation threshold $(T^{(t)}_{N^{(t)}})_j$ beyond which the similarity is not significant anymore. $(T^{(t)}_{N^{(t)}})_j$ can be viewed as the maximum distance beyond which we discard the probability that a given component can be similar to the component modeled by $(\theta^{(t)}_{N^{(t)}})_j$. We propose to consider the following estimation of $T^{(t)}_{N_j}$ at a given time t :

$$(T^{(t)}_{N^{(t)}})_j = d\left(\left((\theta^{(t)}_{N^{(t)}})_j + \sigma^{(t)}(\theta^{(t)}_{N^{(t)}})_j\right), (\theta^{(t)}_{N^{(t)}})_j\right); \tag{3.42}$$

$\sigma^{(t)}$ can be seen as the parameter controlling how the system perceives the data. If $\sigma^{(t)}$ leads to a high dissimilarity, the system will tend to perceive the new data as being similar to the old one. On the other hand if $\sigma^{(t)}$ leads to a low dissimilarity, the system will consider the new data as a candidate to form new classes. We propose to use the following function for Φ_s

$$\Phi_s\left(d\left((\theta^{(t+1)}_{L^{(t+1)}})_i, (\theta^{(t)}_{N^{(t)}})_j\right)\right) = \begin{cases} e^{-d\left((\theta^{(t+1)}_{L^{(t+1)}})_i, (\theta^{(t)}_{N^{(t)}})_j\right)} & \text{if } d\left((\theta^{(t+1)}_{L^{(t+1)}})_i, (\theta^{(t)}_{N^{(t)}})_j\right) < (T^{(t)}_{N^{(t)}})_j \\ 0 & \text{else.} \end{cases} \tag{3.43}$$

and we define S as follows

$$S\left((\theta^{(t+1)}_{L^{(t+1)}})_i, (\theta^{(t)}_{N^{(t)}})_j\right) = \begin{cases} Eq.3.41 & \text{if } \left\{\exists j | 1 \leq j \leq M^{(t)}_{N^{(t)}}, \Phi_s\left(d\left((\theta^{(t+1)}_{L^{(t+1)}})_i, (\theta^{(t)}_{N^{(t)}})_j\right)\right) \neq 0\right\} \\ 0, \ \forall j & \text{else.} \end{cases} \tag{3.44}$$

To decide which of the components forming the new data are already representing the old data by the set of parameters $(\theta^{(t)}_{N^{(t)}})_j$ we use the following rule :

$$R(\theta^{(t+1)}_{L^{(t+1)}})_i = \left\{ j | 1 \leq j \leq M^{(t)}_{N^{(t)}}, S\left((\theta^{(t+1)}_{L^{(t+1)}})_i, (\theta^{(t)}_{N^{(t)}})_j\right) > 0, \arg\max_j S\left((\theta^{(t+1)}_{L^{(t+1)}})_i, (\theta^{(t)}_{N^{(t)}})_j\right)\right\} \tag{3.45}$$

If $R(\theta^{(t+1)}_{L^{(t+1)}})_i \neq \{\emptyset\}$, this means that the component i representing the new data already exists, otherwise it is considered as a new component. For instance, if $R(\theta^{(t+1)}_{L^{(t+1)}})_i = \{j\}$ we update the parameters of the component j of the old data as it shall be detailed in the following subsection.

3.5.1 Update the $\theta_{jl}^{(t+1)}$

A recursive method has been proposed in [90] to update a mixture model using EM. Using the same approach we propose to update the parameters as follows :

$$\Theta_{N^{(t+1)}}^{(t+1)} = \Theta_{N^{(t)}}^{(t)} + \gamma_{N^{(t+1)}} \frac{\log(p(\mathscr{X}_{L^{(t+1)}}^{(t+1)} | \Theta_{N^{(t)}}^{(t)})}{\partial \Theta} \tag{3.46}$$

$\gamma_{N^{(t+1)}}$ is a sequence of positive number that decreases to zero. We choose $\gamma_{N^{(t+1)}} = \frac{1}{N^{(t)}+L^{(t+1)}}$, which gives us :

$$\begin{aligned}\begin{pmatrix}(\alpha_{N^{(t+1)}}^{(t+1)})_{jl}\\(\beta_{N^{(t+1)}}^{(t+1)})_{jl}\end{pmatrix} &= \begin{pmatrix}(\alpha_{N^{(t)}}^{(t)})_{jl}\\(\beta_{N^{(t)}}^{(t)})_{jl}\end{pmatrix} \\ &+ \frac{\sum_{n=1}^{L^{(t+1)}} \hat{Z}_{(N^{(t)}+n)_j}}{N^{(t)}+L^{(t+1)}} \begin{pmatrix}\Psi((\alpha_{N^{(t)}}^{(t)})_{jl} + (\beta_{N^{(t)}}^{(t)})_{jl}) - \Psi((\alpha_{N^{(t)}}^{(t)})_{jl}) + \log(\frac{X_{(N^{(t)}+n)l}}{1+X_{(N^{(t)}+n)l}})\\ \Psi((\alpha_{N^{(t)}}^{(t)})_{jl} + (\beta_{N^{(t)}}^{(t)})_{jl}) - \Psi((\beta_{N^{(t)}}^{(t)})_{jl}) + \log(\frac{1}{1+X_{(N^{(t)}+n)l}})\end{pmatrix}\end{aligned} \tag{3.47}$$

where $\hat{Z}_{(N^{(t)}+n)_j}$ is the posterior probability of the new coming vector $\vec{X}_{N^{(t)}+n}$ given the set of parameters $(\Theta_{L^{(t+1)}}^{(t+1)})$ and considering that $R(\theta_{L^{(t+1)}}^{(t+1)})_i = \{j\}$, which means that the component i of the new coming data is equivalent to the component j of the old data. As we suppose that $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ should only be represented by $M_{L^{(t+1)}}^{(t+1)}$ components, we also assume that the membership probability for of the new coming vector $\vec{X}_{N^{(t)}+n}$ remains the same for the component i of the new data, and the component j of the old data. Thus, we have :

$$\hat{Z}_{(N^{(t)}+n)_j} = \frac{(\pi_{L^{(t+1)}}^{(t+1)})_i p(\vec{X}_{N^{(t)}+n} | (\theta_{L^{(t+1)}}^{(t+1)})_i)}{\sum_{i=1}^{M_{L^{(t+1)}}^{(t+1)}} (\pi_{L^{(t+1)}}^{(t+1)})_i p(\vec{X}_{N^{(t)}+n} | (\theta_{L^{(t+1)}}^{(t+1)})_i)} \tag{3.48}$$

Note that when $R(\theta_{L^{(t+1)}}^{(t+1)})_i = \{\emptyset\}$, we create the new class i with parameters $(\theta_{L^{(t+1)}}^{(t+1)})_i)$, and we add it to the set of parameters $\Theta_{N^{(t+1)}}^{(t+1)}$.

3.5.2 Update the Mixing Weights

We update the mixing weights as follows :

$$\pi_{N^{(t+1)}_j}^{(t+1)} = \frac{N^{(t)} \pi_{N^{(t)}_j}^{(t)} + L^{(t+1)} \pi_{L^{(t+1)}_i}^{(t+1)}}{N^{(t)} + L^{(t+1)}} \tag{3.49}$$

Note that Eq.3.49 can be used for $j = 1, \ldots, M^{(t+1)}_{N^{(t+1)}}$, we consider that $\pi^{(t)}_{N^{(t)}_j} = 0$ for $j > M^{(t)}_{N^{(t)}}$, and $\pi^{(t+1)}_{L^{(t+1)}_i} = 0$ when $j \leq M^{(t)}_{N^{(t)}}$ and the new data are not represented by the class j.

3.5.3 Considered Metrics

In this section we develop a set of metrics d for the GID distribution. The calculation details can be found in Appendix B.2. The first distance is the symmetric Kullback-Leibler (SKL) divergence between two GID distributions with parameters θ_j and θ_i :

$$SKL\Big((p(X|\theta_j), p(X|\theta_i)\Big) = \sum_{l=1}^{D}\Bigg[(\alpha_{jl}-\alpha_{il})\Big(\Psi(\alpha_{jl}) - (\Psi(\alpha_{il}) - \Psi(\alpha_{jl}+\beta_{jl}) + \Psi(\alpha_{il}+\beta_{il})\Big) + (\beta_{jl}-\beta_{il})\Big(\Psi(\beta_{jl}) - (\Psi(\beta_{il}) - \Psi(\alpha_{jl}+\beta_{jl}) + \Psi(\alpha_{il}+\beta_{il})\Big)\Bigg] \tag{3.50}$$

The second developed distance is the symmetric Renyi divergence proposed in [97]. We calculate it for GID distribution as follows :

$$R_a\Big(p(\vec{X}|\theta_i), p(\vec{X}|\theta_j)\Big) = \sum_{l=1}^{D}\Bigg[\frac{1}{a-1}\log\Big(\int_0^{+\infty} p(X_i|\theta_{il})^a p(X_i|\theta_{jl})^{1-a} dX\Big) + \frac{1}{a-1}\log\Big(\int_0^{+\infty} p(X_i|\theta_{jl})^a p(X_i|\theta_{il})^{1-a} dX\Big)\Bigg] \tag{3.51}$$

such that

$$\int_0^{+\infty} p(X_i|\theta_{il})^a p(X_i|\theta_{jl})^{1-a} = \left[\frac{\Gamma(\alpha_{il}+\beta_{il})}{\Gamma(\alpha_{il})\Gamma(\beta_{il})}\right]^a \times \left[\frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})}\right]^{1-a} \times \frac{\Gamma(a\alpha_{il}+\alpha_{jl}-a\alpha_{jl})\Gamma(a\beta_{il}+\beta_{jl}-a\beta_{jl})}{\Gamma\big((a\alpha_{il}+\alpha_{jl}-a\alpha_{jl})+(a\beta_{il}+\beta_{jl}-a\beta_{jl})\big)} \tag{3.52}$$

The third one is the Bhattacharyya distance which measures the similarity between two probability densities. We calculate it for the GID distributions as follows :

$$B_{\frac{1}{2}}\Big(p(\vec{X}|\theta_i), p(\vec{X}|\theta_j)\Big) = \prod_{l=1}^{D}\left[\frac{\Gamma\Big(\frac{1}{2}(\alpha_{jl}+\alpha_{il})\Big)\Gamma\Big(\frac{1}{2}(\beta_{jl}+\beta_{il})\Big)\sqrt{\Gamma(\alpha_{jl}+\beta_{jl})\Gamma(\alpha_{il}+\beta_{il})}}{\Gamma\Big(\frac{1}{2}(\alpha_{jl}+\beta_{jl}+\alpha_{il}+\beta_{il})\Big)\sqrt{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})\Gamma(\alpha_{il})\Gamma(\beta_{il})}}\right] \tag{3.53}$$

in this case we consider $d = 1 - B_{\frac{1}{2}}$.

3.5.4 Detailed Algorithm

Algorithm 1 details how our framework processes the update of the old model when we have new data arriving at the system level. The main idea is to consider that we have an old data model that is already represented by a finite GID mixture, and when the new data come, we also model it by a GID finite mixture whose number of components varies from 1 to a fixed maximum number of allowed components. Then, we measure the distances between the components of the new data, and the components of the old data. If the measured distance between two given components is under the threshold, they are considered as being the same component, so the ancient component gets updated and then substituted in the models of the old and new data. If the new components are considered as being totally dissimilar with the old ones, the updated model of the whole data will consider them as new components that should be created and add them to the model. In each iteration, we calculate the model selection criterion for the new data and the whole data considering the new parameters in order to select the best model at the end. In Algorithm 1 we use the MML as a model selection criterion.

Algorithm 1 GID online learning using EM-MML and probabilistic distance

At t, **Input** : D-dimensional dataset $\mathscr{X}^{(t)}_{N^{(t)}} = \{\vec{X}_1, \ldots, \vec{X}_{N^{(t)}}\}$, $\Theta^{(t)}_{N^{(t)}}$,
At $t+1$ **Input** : $\mathscr{X}^{(t+1)}_{L^{(t+1)}} = \{\vec{X}_{N^{(t)}+1}, \vec{X}_{N^{(t)}+2}, \ldots, \vec{X}_{N^{(t)}+L^{(t+1)}}\}$, $(M^{(t+1)}_{L^{(t+1)}})_{max}$, $\sigma^{(t)}$, d: SKL, B, R.
for all $1 \leq k \leq (M^{(t+1)}_{L^{(t+1)}})_{max}$ **do**
    Model $\mathscr{X}^{(t+1)}_{L^{(t+1)}}$ by a finite GID mixture composed of k-components using EM on Eq.3.39 using the same approach in chapter 2.
    for all $1 \leq i \leq k$
        Calculate $R(\theta^{(t+1)}_{L^{(t+1)}})_i$ using Eq.3.45 considering one of the developed metrics in Eqs. 3.50, 3.51, and 3.53.
        if $R(\theta^{(t+1)}_{L^{(t+1)}})_i \neq \{\emptyset\}$
            Update $(\theta^{(t)}_{N^{(t)}})_{R(\theta^{(t+1)}_{L^{(t+1)}})_i}$ using Eq.3.47
            $(\theta^{(t+1)}_{L^{(t+1)}})_i = (\theta^{(t)}_{N^{(t)}})_{R(\theta^{(t+1)}_{L^{(t+1)}})_i}$
        end if
    end for
    Calculate $(\vec{\pi}^{(t+1)}_{N^{(t+1)}})_k$ using equation Eq.3.49
    Calculate $(M^{(t+1)}_{N^{(t+1)}})_k$, $(\Theta^{(t+1)}_{N^{(t+1)}})_k$
    Calculate $MML^{newData}_k(\mathscr{X}^{(t+1)}_{L^{(t+1)}}, (\Theta^{(t+1)}_{L^{(t+1)}})_k, k)$ using Eq.3.30
    Calculate $MML^{wholeData}_k(\mathscr{X}^{(t+1)}_{N^{(t+1)}}, (\Theta^{(t+1)}_{N^{(t+1)}})_k, (M^{(t+1)}_{N^{(t+1)}})_k)$ using Eq.3.30
end for
Decide on the model basing on the MML and the chosen distance d, return : $(M^{(t+1)}_{N^{(t+1)}})$, $(\Theta^{(t+1)}_{N^{(t+1)}})$, $(\vec{\pi}^{(t+1)}_{N^{(t+1)}})$

3.6 Experimental Results

3.6.1 Synthetic Data

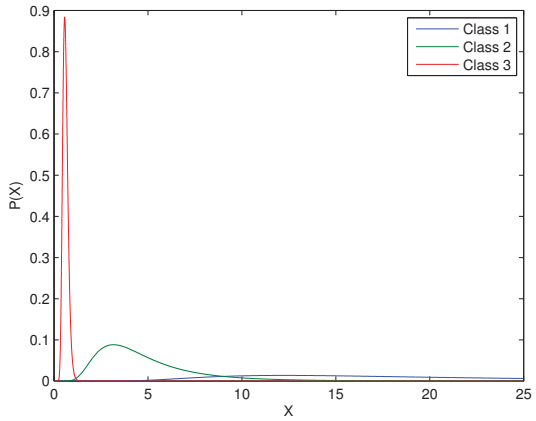
We propose to validate our framework using synthetic data. First we build $\mathscr{X}_{N^{(t)}}^{(t)}$, then we estimate its model parameters, and validate the model selection criteria that we have considered. Then we build $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$, and we propose to update the model accordingly using Algorithm 1.

Parameters Estimation

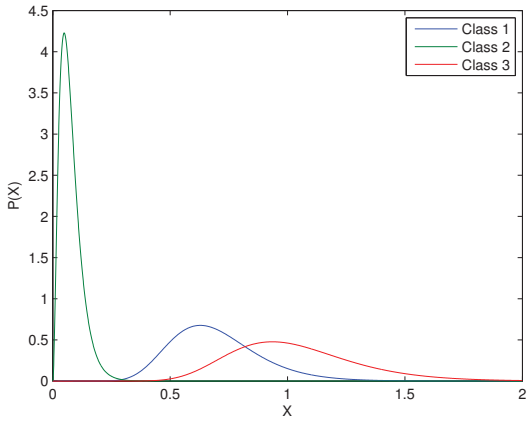
To construct $\mathscr{X}_{N^{(t)}}^{(t)}$, we generate a dataset in the transformed space from different inverted Beta distributions. We mainly consider three classes composed of four-dimensional vectors, following the parameters in Table 3.1. Thus, $\mathscr{X}_{N^{(t)}}^{(t)}$ is modeled by a 4-dimensional mixture model composed of 3 components. The histograms of the inverted Beta distributions forming each dimension are illustrated in Figure 3.1. Note that the different distributions are overlapping which makes the estimation harder. We estimate the parameters of the mixture using the EM algorithm, and report on the results in Table 3.1, where we show the real and estimated parameters. The maximum detected percent error of relative change between real and estimated parameters among all the set of parameters is 3.89% which reflects a good estimation. Thus, the estimated histogram and the real one are indistinguishable, and our algorithm performs well when estimating the parameters using EM.

Table 3.1: Real and estimated parameters in the case of a 4-dimensional dataset generated from 3-components mixture model.

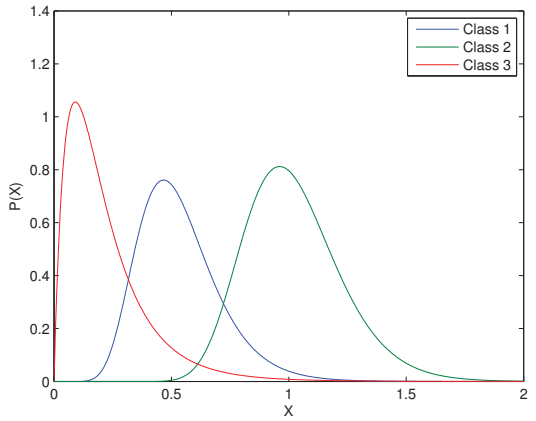
j	l	π_j	α_{ji}	β_{jl}	$\hat{\pi}_j$	$\hat{\alpha_{jl}}$	$\hat{\beta_{jl}}$
1	1	0.3	50	3	0.3	51.8900	3.1216
	2		23	34		23.7812	35.0890
	3		15	29		14.7051	28.3307
	4		20	49		20.5157	50.2743
2	1	0.4	20	5	0.4	19.8558	4.8733
	2		3	40		3.1068	41.2450
	3		50	50		50.1681	50.1832
	4		34	18		34.5822	18.1180
3	1	0.3	30	50	0.3	29.8472	49.6698
	2		30	30		29.8738	29.8352
	3		2	10		2.0625	10.3134
	4		19	23		19.1818	23.2054



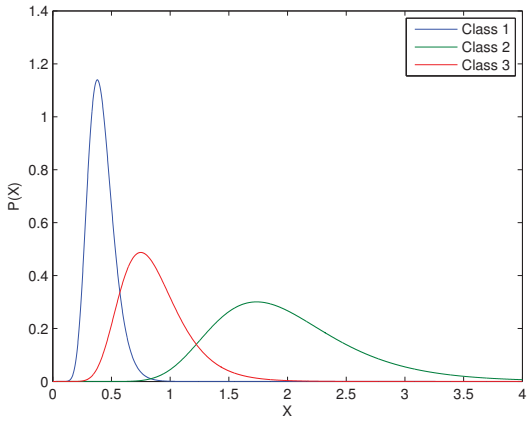
(a) dimension 1



(b) dimension 2



(c) dimension 3



(d) dimension 4

Figure 3.1: Different generated inverted Beta distributions representing dimensions 1, 2 , 3 and 4 of the model in Table 3.1.

Model Selection

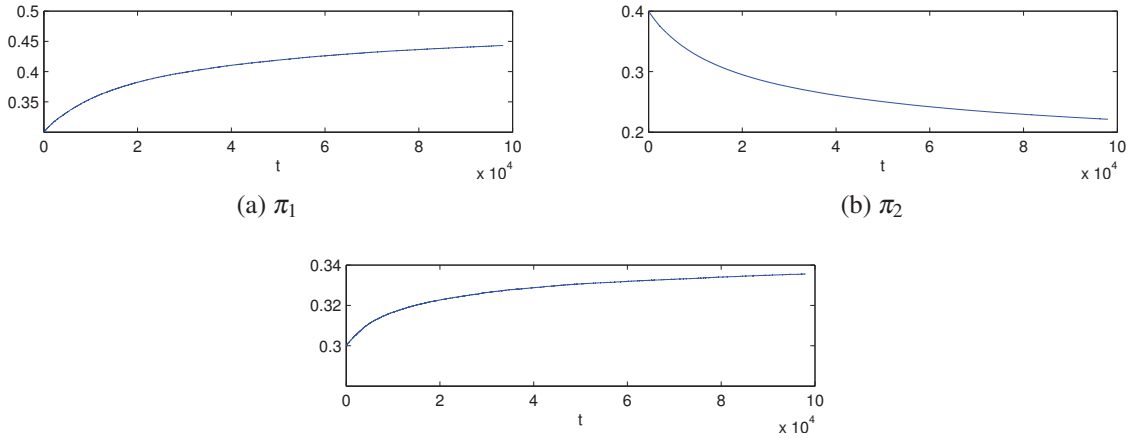
In this subsection we propose to perform the model selection algorithm on $\mathscr{X}_{N^{(t)}}^{(t)}$, using the MML, AIC, LEC, MDL, and MMDL criteria. The results are reported in Table 3.2. The algorithm succeeds to find the appropriate number of classes which is equal to three using MML, MDL, LEC and MMDL, whereas it fails to find the exact number of classes using AIC. As the MML criterion takes into consideration a prior information about the parameters, we will mainly consider it for the rest of simulations to illustrate the results.

Table 3.2: Selection Criteria Values of the mixture 4-dimensional dataset generated from 3-components mixture model

M	MML	AIC	LEC	MDL	MMDL
1	51885	51810	51883	51847	51847
2	37525	37365	37522	37439	37432
3	19844	19597	19840	19707	19692
4	19922	19596	19916	19744	19717
5	20002	19597	19994	19781	19744
6	20077	19596	20068	19818	19766
7	20153	19598	20142	19857	19788
8	20221	19599	20208	19895	19800
9	20297	19600	20282	19932	19822
10	20378	19604	20362	19973	19851

Static Online Learning

During this experiment, we aim at validating the updating approach developed in Section 3.3.3. We suppose that the data are received over time, and does not affect the number of classes as the vectors are supposed to belong to the existing classes, but rather affect the parameters and weight of each class. At a given time t_0, we assume that we have the estimated parameters in Table 3.1. Also, we assume that we know that the first and third classes have 3000 vectors, each, and the second class has 4000 vectors. Now we suppose that we are receiving new vectors over time and at each time t one single vector is received and used to update the model as explained in section 3.3.3. We set our experiment in such a way that the model receives 49000 vectors from class 1, also 49000 from class 3, and 22000 vectors from class 2. We randomly permute all the vectors, so the system receives a vector with a random class membership at each time t. At the end of the experiment, we know that we will have 52000 vectors from classes 1 and 3, each, and 26000 vectors from class 2. We report on the changes of the mixture weights during time, in Fig. 3.2. As we can see the evolution of the weight π_2 of class 2 in time, goes from 0.400 to reach 0.2213, and the weights π_1 and π_3 of the classes 1 and 3 change respectively from 0.3 and 0.3, to 0.4431 and 0.3356. Naturally when updating the parameters online, we are not expecting to exactly find the weights of classes 1, 2 and 3 to be equal to 0.4, 0.2 and 0.4, respectively, but we expect that their estimation will tend to those values. Indeed, the membership's frequency of the arriving vectors have an influence on the weights updating. Suppose for instance, that at the beginning we receive 49000 vectors only from the first class, sequentially over time, the probability of receiving a vector whose membership belongs to class 1, will increase significantly, which means that the weight π_1 will increase. If we stop the experiment at this point, our algorithm would give a high weight to the first class which is expected. π_1 will start to decrease when receiving vectors that belong to classes 2 or 3. During our experiment, we tried to randomize the membership of the arriving vectors, but still, we did not make it uniform, therefore we obtained the reported results. The evolutions of the



(a) π_1 (b) π_2

(c) π_3

Figure 3.2: Evolution of the parameters π_1, π_2 and π_3 in time.

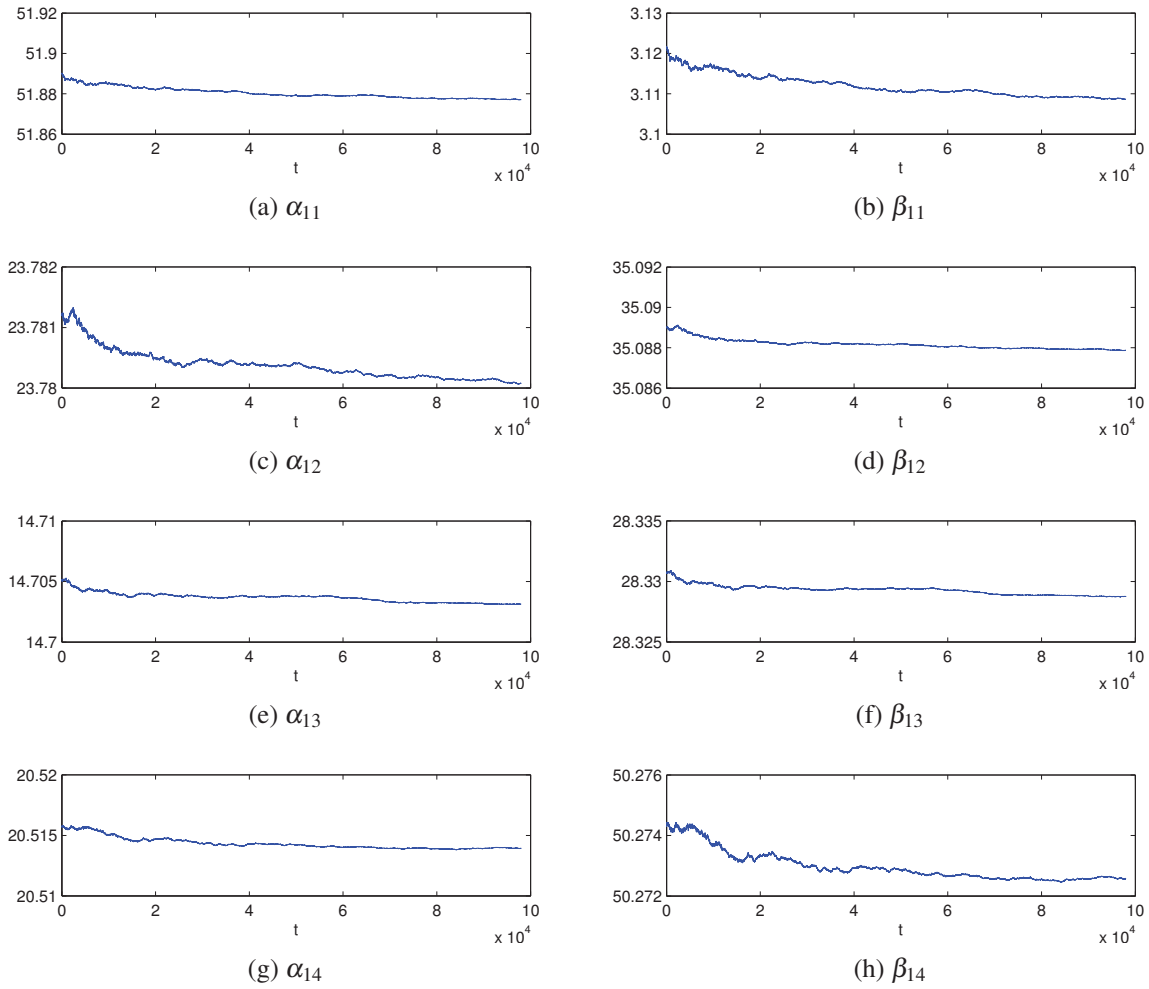
parameters representing the classes 1, 2, and 3 are respectively reported in Figs. 3.3, 3.4, and 3.5. According to the different graphs, we can conclude that our updating model for the finite mixture gives very acceptable estimated values, and behaves as expected.

Dynamic Online Learning

During this experiment, we generate a new dataset $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ in such a way that the whole model should be updated according to the parameters in Table 3.12. Figure 3.6 illustrates the histograms of the inverted Beta distributions forming each dimension when considering the whole data $\mathscr{X}_{N^{(t+1)}}^{(t+1)}$. These distributions are overlapping in the features space which makes the experiment a good case to validate the performance of our algorithm. $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ is mainly composed of four datasets. The first and second datasets are generated respectively from the third and first components in Table 3.1, whereas the third and fourth datasets are generated respectively from the fourth and fifth components in Table 3.12. Thus, we consider the case where the new dataset is composed of new and old classes. We fix the maximum number of classes $(M_{L^{(t+1)}}^{(t+1)})_{max}$ that can represent $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ to be 10. Apart from the choice of the distance d , the choice of $\sigma^{(t)}$ in Eq. 3.42 controls how the data should be perceived by the system. We propose to adopt the following choice for $\sigma^{(t)}$:

$$\sigma^{(t)} = |scale^{(t)} \times \mathscr{N}(0,1)| \tag{3.54}$$

where $\mathscr{N}(0,1)$ is the standard normal distribution. We propose to study the effect of $scale^{(t)}$ on the behavior/perception of the system toward the new data. In the following we will consider three cases, the first one when $scale^{(t)}$ is equal to 50%, then 10000% and finally 1%. $scale^{(t)}$ can



(a) α_{11} (b) β_{11}

(c) α_{12} (d) β_{12}

(e) α_{13} (f) β_{13}

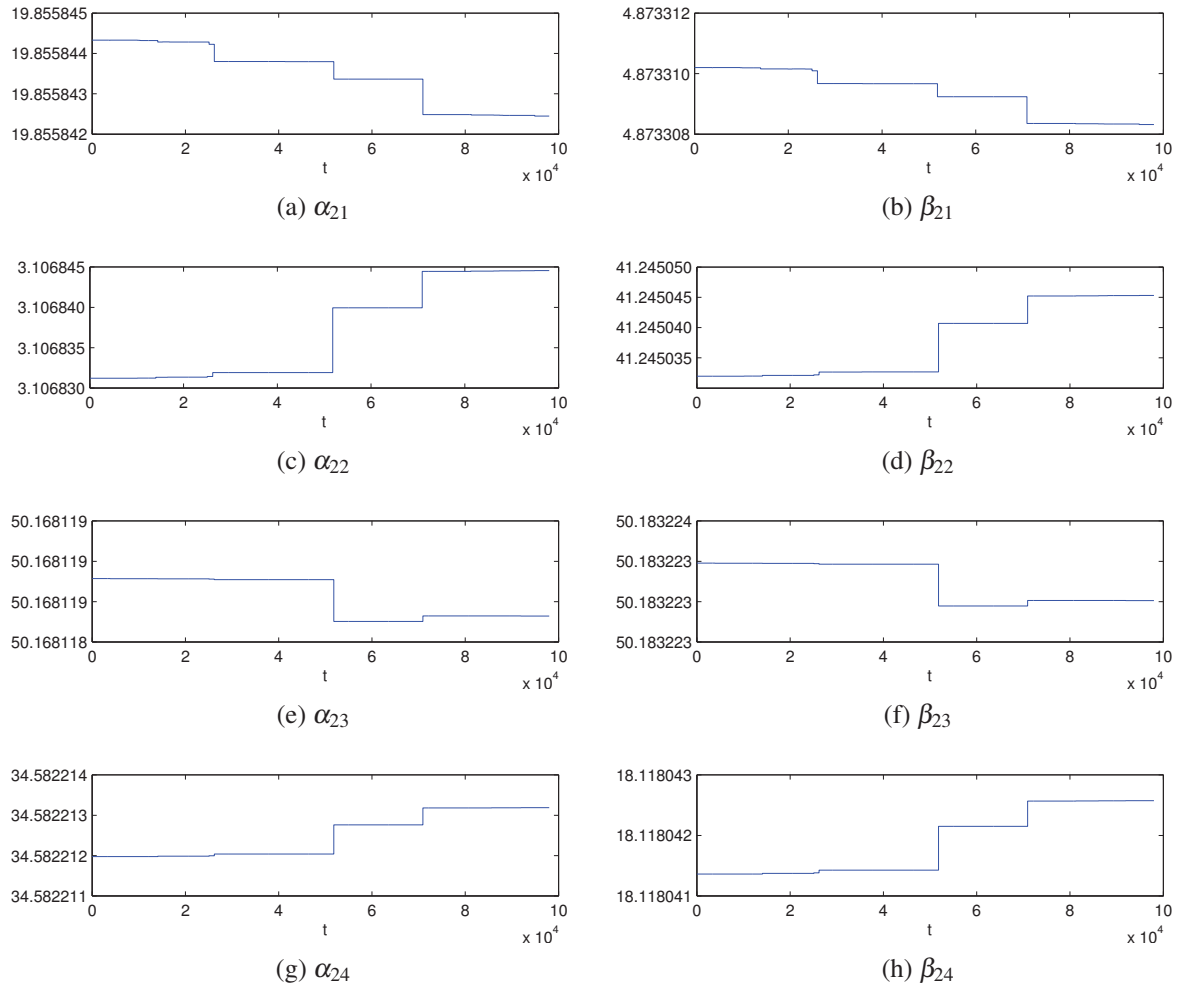
(g) α_{14} (h) β_{14}

Figure 3.3: Evolution of the parameters of class 1 in time t.

be considered as the percentage of tolerated dissimilarity that we want to consider. During each experiment, we build a “mapping vector" for the different components of the model representing $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$, telling whether the component is considered as new, or as an already represented one, using the set $R(\theta_{L^{(t+1)}}^{(t+1)})_i$. For easiness of presentation, we report on the selected class only.

Tolerance of 50%**:**

In this experiment we run the algorithm using $scale^{(t)} = 0.5$. We consider that 50% of tolerated dissimilarity should lead to find the exact model that considers that $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ is composed of two old components, and two new ones. We report on the results in Tables 3.3 and 3.4. We note that all the selected metrics in the features probabilistic space lead to the same results reflecting the targeted model. Table 3.3 informs us that the algorithm is capable to find that $M_{L^{(t+1)}}^{(t+1)} = 4$ classes is the best model for $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ using 50% of tolerated dissimilarity. Table 3.4 also shows that the



(a) α_{21} (b) β_{21}

(c) α_{22} (d) β_{22}

(e) α_{23} (f) β_{23}

(g) α_{24} (h) β_{24}

Figure 3.4: Evolution of the parameters of class 2 in function in time t.

algorithm ends up by selecting the model composed of $M_{N^{(t+1)}}^{(t+1)} = 5$ classes, which is the intended model to select when considering $scale^{(t)} = 0.5$. The mapping vector in Table 3.5 shows that the components are successfully substituted as we generated the data such that the first and second new components are substituted by the third and first ancient classes respectively, whereas the third and fourth new components are considered as the components that should be constructed and added to the ancient model. Thus, the algorithm succeeds to model the data as expected.

Tolerance of 10000%:

In this subsection we take $scale^{(t)} = 100$. We run our algorithm, and we report on the results on Tables 3.6 and 3.7. We notice that the choice of the distance when considering a high value of dissimilarity rate influences the model selection of the new coming data in terms of the choice of $M_{L^{(t+1)}}^{(t+1)}$, but the three metrics lead to the selection of the same number $M_{N^{(t+1)}}^{(t+1)} = 3$ of classes that finally represent the whole data $\mathscr{X}_{N^{(t+1)}}^{(t+1)}$. According to the generated mapping vectors shown in

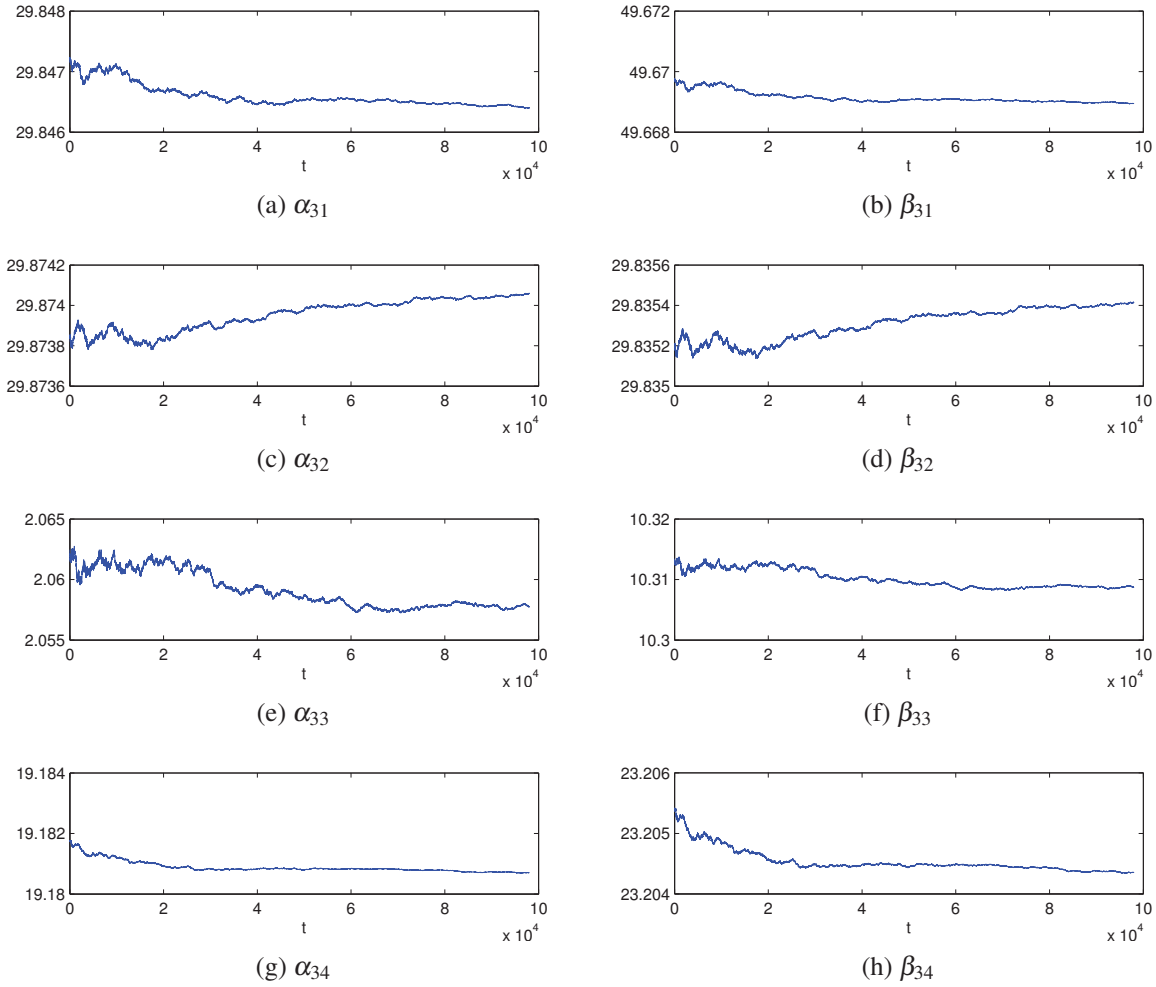


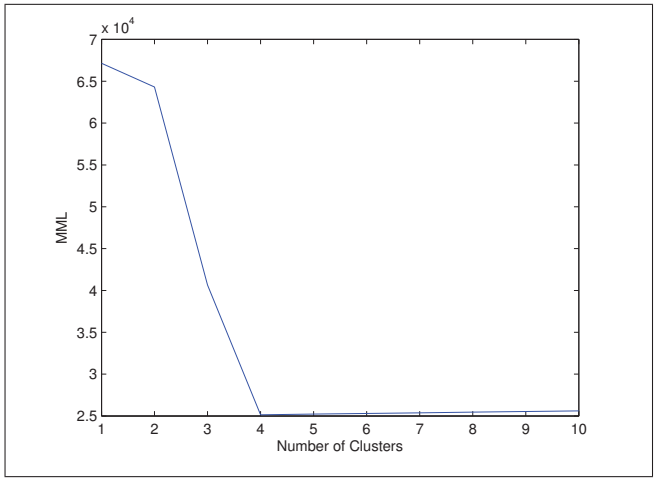
Figure 3.5: Evolution of the parameters of class 3 in time t.

Table 3.8 when using the Kullback-Leibler divergence, the algorithm tends to represent the whole new data by one single class that can be substituted by the third component of ancient model. According to our approach, only the third component of the already built model is updated. The Renyi divergence leads to the representation of the new data by three classes that are all substituted by the third, second, and first components of the ancient model which get updated accordingly. Finally, the use of the Bhattacharyya similarity leads to the selection of four classes for the new data, that are substituted by the first and second classes of the ancient model. We consider that the fact of representing the whole data by three classes, as being a success, as we expected the algorithm not to create new classes when we consider a high rate of dissimilarity.

Tolerance of 1%:

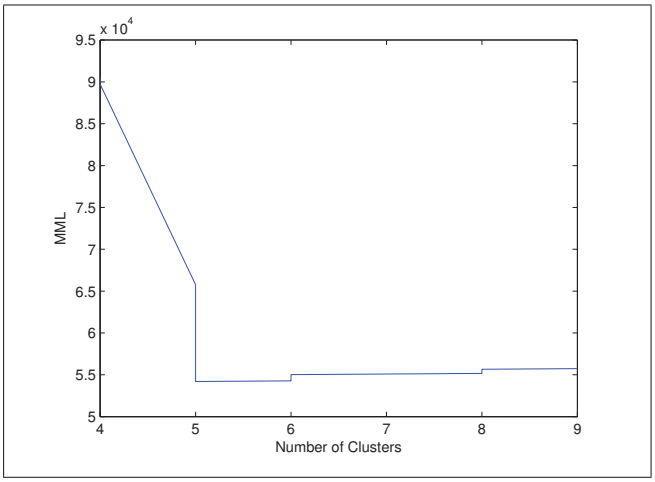
In this experiment, we want to make our system sensitive to the dissimilarity and considers any new data that are not represented by components that are considerably close to the old ones, as

Table 3.3: Message length values as a function of the number of clusters for the new coming data $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ when considering 0.5.



$M_L^{(t+1)}$	MML
1	67144
2	64312
3	40647
4	25144
5	25223
6	25301
7	25379
8	25458
9	25536
10	25613

Table 3.4: Message length values as a function of the number of clusters for the whole $\mathscr{X}_{N^{(t+1)}}^{(t+1)}$ when considering 0.5.



$M_{L^{(t+1)}}^{(t+1)}$	$M_{N^{(t+1)}}^{(t+1)}$	MML
1	4	90564
2	4	89730
3	5	65760
4	5	54193
5	6	54272
6	6	55014
7	7	55092
8	8	55164
9	8	55656
10	9	55734

Table 3.5: Mapping vector for Table 3.4.

$\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ component i	1	2	3	4
$R(\theta_{L^{(t+1)}}^{(t+1)})_i$	3	1	$\emptyset$	$\emptyset$

being a data that should serve to the construction of new classes for the whole data. We report on the results in Tables 3.9 and 3.10. All the considered metrics lead to the same results when considering a very low dissimilarity tolerance. The algorithm succeeds to select the appropriate model that represents the new data by four classes, and none of these classes is substituted with the

Table 3.6: Message length values as a function of the number of clusters for the coming data $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ when considering 100.

$M_{L^{(t+1)}}^{(t+1)}$	MML_{SKL}	MML_R	MML_B
1	586000	586000	1022300
2	587600	424100	453900
3	592700	387100	454500
4	593300	394000	435700
5	596000	394100	436100
6	596100	394300	436200
7	598200	394400	445000
8	598800	394500	445200
9	598900	394500	445300
10	599500	390000	441800

Table 3.7: Message length values as a function of the number of clusters for the whole data $\mathscr{X}_{N^{(t+1)}}^{(t+1)}$ when considering 100.

$M_{L^{(t+1)}}^{(t+1)}$	$M_{N^{(t+1)}}^{(t+1)}$	MML_{SKL}	MML_R	MML_B
1	3	405920	405920	411940
2	3	415510	403730	409590
3	3	411910	402850	410990
4	3	409650	409550	400510
5	3	415430	409550	402680
6	3	416630	411560	415180
7	3	412640	412590	414220
8	3	414050	412710	414340
9	3	416160	413430	415680
10	3	415360	414280	409230

Table 3.8: Mapping vectors for Table 3.7.

$\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ mode i	1
$R(\theta_{L^{(t+1)}}^{(t+1)})_i$	3

Symmetric Kullback-Leibler

$\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ mode i	1	2	3
$R(\theta_{L^{(t+1)}}^{(t+1)})_i$	3	2	1

Renyi

$\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ component i	1	2	3	4
$R(\theta_{L^{(t+1)}}^{(t+1)})_i$	1	2	1	2

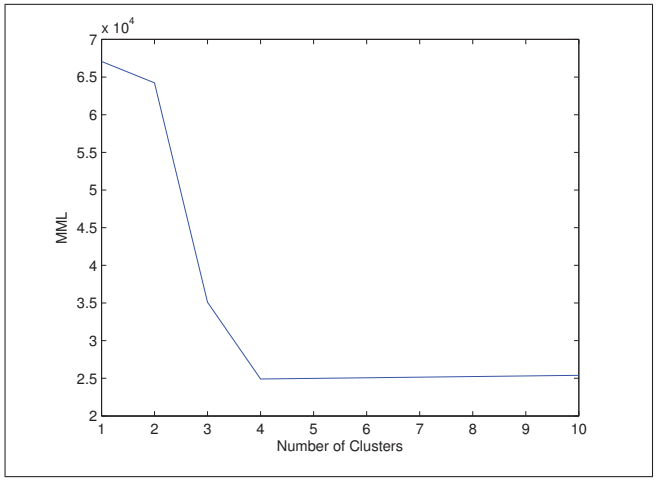
Bhattacharyya

already created components of the old data as reported in the mapping vector in Table 3.11. Thus, the system behaves as expected, which validates our algorithm.

Estimates of Parameters for the Real Model:

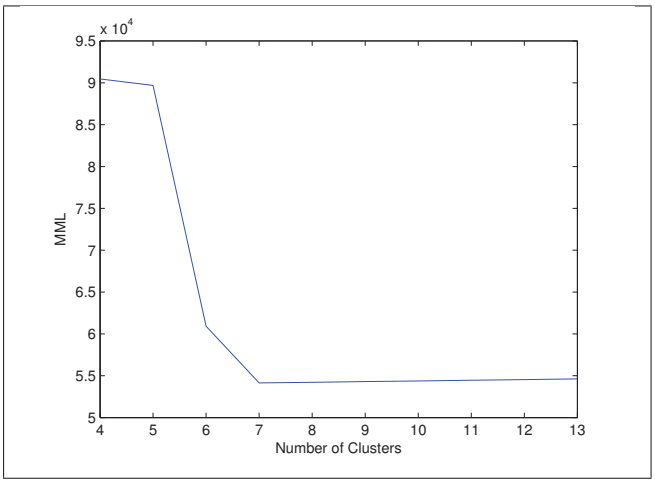
As we mentioned above, we could find the generated model when we used $scale^{(t)} = 0.5$, that is a

Table 3.9: Message length values as a function of the number of clusters for the new coming data $\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ when considering 0.01.



$M_L^{(t+1)}$	MML
1	67066
2	64230
3	35093
4	24916
5	24997
6	25078
7	25158
8	25242
9	25316
10	25389

Table 3.10: Message length values as a function of the number of clusters for the whole $\mathscr{X}_{N^{(t+1)}}^{(t+1)}$ when considering 0.01.

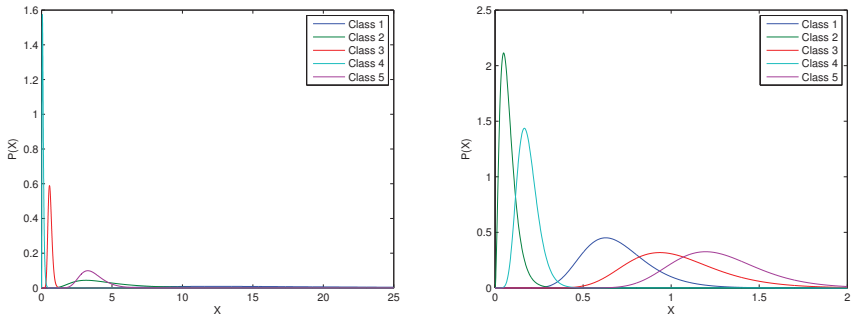


$M_{L^{(t+1)}}^{(t+1)}$	$M_{N^{(t+1)}}^{(t+1)}$	MML
1	4	90460
2	5	89698
3	6	60920
4	7	54137
5	8	54219
6	9	54300
7	10	54380
8	11	54464
9	12	54540
10	13	54619

Table 3.11: Mapping Vector for Table 3.10.

$\mathscr{X}_{L^{(t+1)}}^{(t+1)}$ component i	1	2	3	4
$R(\theta_{L^{(t+1)}}^{(t+1)})_i$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

rate of dissimilarity equal to 50% basing on our definition. In this subsection, we report on the real and estimated parameters shown in Table 3.12 of the selected model. The maximum detected percent error of relative change between real and estimated parameters among all the set of parameters is 3.86%, which reflects a good result.



(a) dimension 1 (b) dimension 2

(c) dimension 3 (d) dimension 4

Figure 3.6: Different generated inverted Beta distributions representing dimensions 1, 2, 3 and 4 of the model in Table 3.12.

Discussion

The model we have built shows that it is capable to change how the systems perceives the new coming data over time. This means that we can build customizable system behaviors depending on the application, and the degree of dissimilarity that we want to establish between different components. The distance metrics that we have used lead to the same results when we tolerate a moderate or low level of dissimilarity. When we tolerate a considerable rate of dissimilarity, depending on the distance metric, the system selects different models to represent the coming data, but still decide not to create new classes for the whole data. Thus, we consider that the distances that we have adopted in order to measure the similarity/dissimilarity between different modes have given good results with the adopted forms of the tolerance thresholds. It is noteworthy to mention that the model can be more flexible by making $\sigma^{(t)}$ and $scale^{(t)}$ depend on a given component such that they become $\sigma_j^{(t)}$ and $scale_j^{(t)}$, so we can select similarity/dissimilarity tolerance by component. Also this model can be interpreted as an infinite mixture, as it is not limited by a fixed number of new classes when we have an unlimited/huge number of coming data.

Table 3.12: Real and estimated parameters in the case of a 4-dimensional dataset generated from 3-components mixture model when considering $scale^{(t)} = 0.5$ and the selected model.

j	l	π_j	α_{ji}	β_{jl}	$\hat{\pi}_j$	$\hat{\alpha_{jl}}$	$\hat{\beta_{jl}}$
1	1	0.2	50	3	0.2	51.8901	3.1205
	2		23	34		23.7813	35.0890
	3		15	29		14.7049	28.3308
	4		20	49		20.5156	50.2744
2	1	0.2	20	5	0.2	19.8558	4.8733
	2		3	40		3.1068	41.2450
	3		50	50		50.1681	50.1832
	4		34	18		34.5822	18.1180
3	1	0.2	30	50	0.2	29.8472	49.6698
	2		30	30		29.8739	29.8351
	3		2	10		2.0611	10.3133
	4		19	23		19.1814	23.2057
4	1	0.2	3	30	0.2	2.9950	29.7101
	2		12	65		12.0874	65.3781
	3		19	10		18.7560	9.9524
	4		23	30		22.2941	29.0916
5	1	0.2	80	23	0.2	78.5230	22.4339
	2		56	45		57.3767	46.0903
	3		87	65		88.1882	65.7813
	4		33	3		32.9821	3.0035

3.6.2 Real Application

Figure 3.7: Inria Horses dataset : sample images.

In this section, we validate our algorithm using real computer vision datasets for object recognition. We mainly investigate the performance of our algorithm within two applications, the first one is recognizing objects through a binary classification (i.e. tell wether an object exists in an image or not), and the second one is to build a hierarchical model according to a given ontology representing users intentions as discussed in chapter 2. For the first application, we propose to compare our algorithm with recent developed mixture models in the literature in terms of objects recognition, specifically those reported in [85]. The results reported in [85] have been obtained

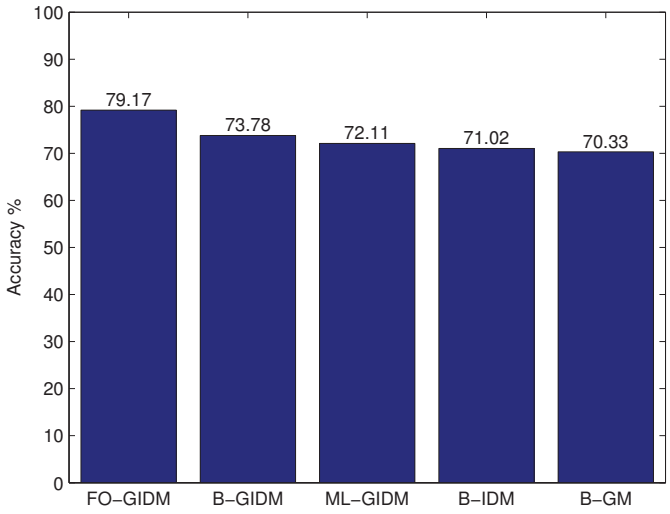


Figure 3.8: Classification accuracies of the Inria Horses dataset.

Figure 3.9: WS Horses dataset : sample images.

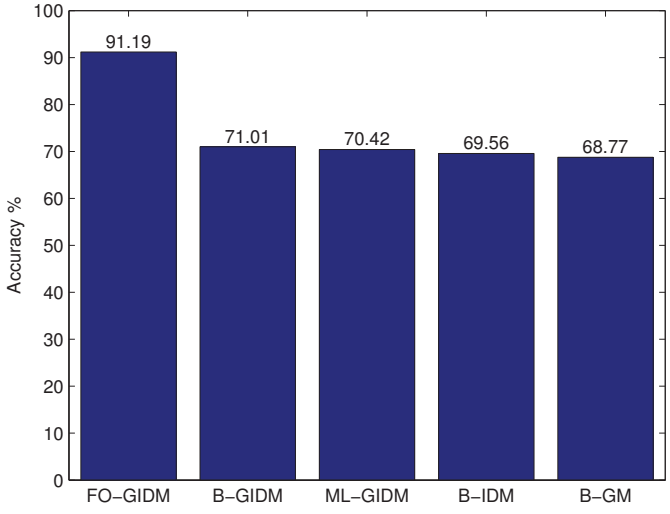


Figure 3.10: Classification accuracies of the WS Horses dataset.

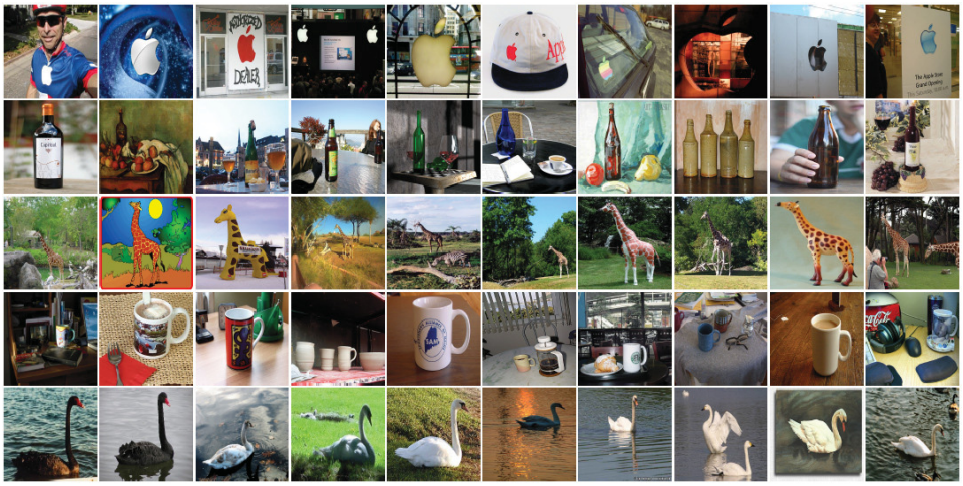


Figure 3.11: ETHZ dataset : sample images

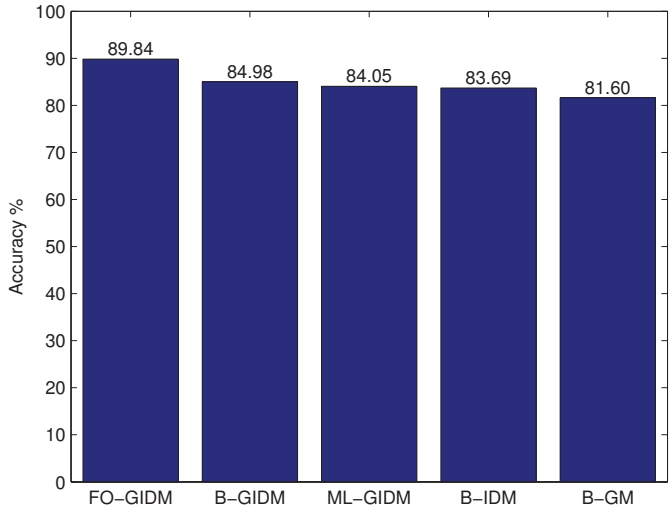


Figure 3.12: Classification accuracies of the Apple Logo dataset.

using different mixture models, namely the GID mixture learned in a fully Bayesian way, (B-GIDM), the GID mixture learned using maximum likelihood (ML-GIDM), the inverted Dirichlet mixture learned via Bayesian inference (B-IDM), and the Bayesian Gaussian mixture (B-GM). We name our algorithm the flexible online generalized inverted Dirichlet mixture (FO-GIDM), where we mainly use the MML as a selection criterion, and the Kullback-Leibler distance for the similarity perception in the proposed algorithm. During our experiment we empirically adopt a tolerance threshold of similarity equal to 20%. The main goal of this experiment is to detect the presence of specific objects in a given image, which can be reduced to the binary classification of images telling the presence or the absence of a specific object. We use three datasets which

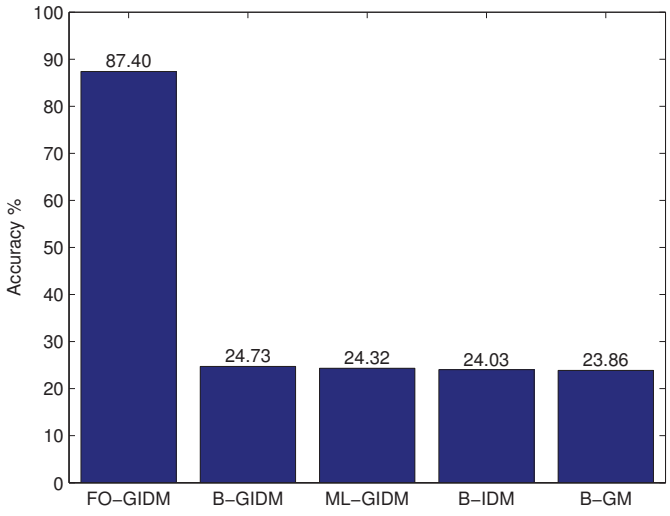


Figure 3.13: Classification accuracies of the Bottles dataset.

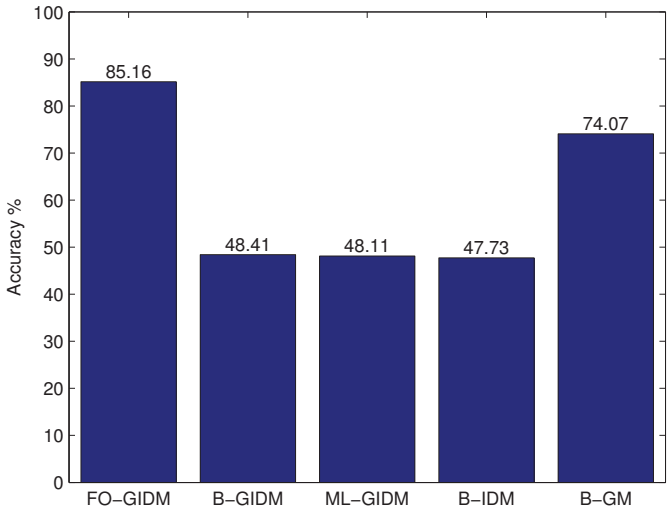


Figure 3.14: Classification accuracies of the Giraffes dataset.

are the “Inria Horses" [98, 99] , “Wiseman Horses" [100, 101] and “ETHZ" [98]. The features of each image have been extracted using the local Histogram of Oriented Gradient (HOG) descriptor proposed in [75]. The HOG algorithm is efficient in terms of detecting local characteristics of images and object detection and during our experiments each image was represented by a 81-dimensional feature vector, as used in [85]. The experiment consists of a training phase and a testing phase. The training phase is the phase during which we build the mixture model that represents the data, and the testing phase is a classification process where we assign each given

test image to a specific class according to the Baye's rule.

During the training phase we proceed as follows; if we have N classes, we suppose that they come sequentially at time t. At $t = 1$ we inject the training images of the first class, and build the model according to our algorithm, using the MML to estimate the adequate number of classes. At time $t = 2$ we inject the second class vectors and we update the model according to our proposed algorithm. One can argue that the use of MML is not always adequate to represent a given data especially when a class is composed of different sub classes that might be overlapping in the features space as shown in chapter 2, but in this case of supervised experiment, we assume that the elementary class we want to model is known in advance, which means that we do not need to zoom into a given class to know its subclasses at a further stage. In other words if the purpose is to detect a horse in a given image, the type of the horse is not needed to be known, so if the MML tells that the class should be represented by three components for example, while we have five types of horses, that would not cause a problem, as the final goal is to tell whether an image contains a horse or not and not to distinguish the horse type into some sub classes. Building the model incrementally using the approach that we mentioned, enables us to model well separated components representing specifically the given classes. The "Inria Horses" dataset is composed of 170 images containing horses (positive images) and 170 images composed of diverse animals and visual scenes without horses (negative images). Figure 3.7 illustrates some examples of the positive and negative images. As in [85], we use 100 training images, 50 positive images and 50 negative images. We compare the performance in terms of accuracy and we report on the results in Fig. 3.8 that shows that we obtain an accuracy of 79.17% which outperforms the reported accuracies of the other algorithms. The learning phase was established by injecting the 50 positive images, and then building the model incrementally by injecting the 50 negative images. The used similarity tolerance threshold ensures that the modes of different classes are well separated.

The second dataset is the "Weizmann-Shotton Horses" dataset which is composed of 327 positive images with horses and 327 negative images without horses. Following the experiment in [85], we consider 50 positive images and 50 negative images for the training phase and the rest of images are used for testing. As we did for the "Inria Horses" dataset, we build the model by considering that we have at the beginning 50 positive images, and then we inject the 50 negative images. Figure 3.9 shows a sample of images of the considered dataset. The obtained results in terms of accuracy are reported in Fig. 3.10. Our algorithm clearly outperforms the other algorithms reaching an accuracy of 91.19%. The third dataset "ETHZ shape classes" is composed of five objects classes (bottles, swans, mugs, giraffes, and apple logos) with a total of 255 images collected from the Web. As illustrated in the dataset sample in Fig. 3.11, this dataset is particularly challenging as the objects are placed in different parts of the images, with different shapes and scales. Following [85, 98], we

build a model for each object separately as a binary classification.The whole dataset is composed of 255 images (40 apple logos, 48 bottles, 87 giraffes, 48 mugs, and 32 swans). For each object, we use the half of positive images as positive training images, and we construct a set of training negative images from the other objects in such a way that the images of every other remaining object constitute $\frac{1}{4}$ of the negative training images.The rest of all the dataset is used as a testing set. For instance, when building the model for bottles, we consider 24 images from the bottles images set, and 6 images from each of the other classes in order to build the set of training images. The obtained results for each class are shown in Figs. 3.12, 3.13, 3.14, 3.15, and 3.16. For each object class, the FO-GIDM significantly outperforms the reported results of the other models. It's noteworthy that our algorithm combines the positive and negative images in the same model, instead of building separately two models for the positive and negative images and maximizing the likelihood in order to attribute a testing image to its specific class.

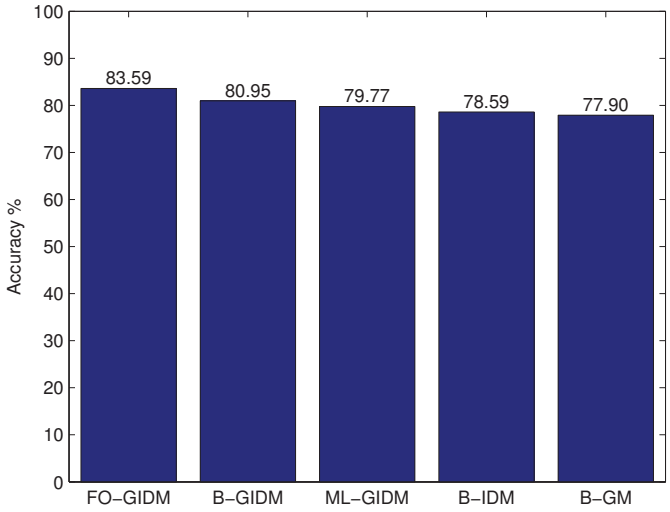


Figure 3.15: Classification accuracies of the Mugs dataset.

Table 3.13: Classification accuracies of the Hierarchical Model in %

	FO-GIDM	FO-IDM
Artificial - Natural	90.22	83.70
Electronic - Furniture - Insect	86.41	77.72
Camera - Cellphone - Chair - Chandelier - Butterfly	86.41	74.46

The last goal is to compare the performance of the generalized inverted Dirichlet distribution (GID) with the inverted Dirichlet distribution (ID) within the same proposed framework (i.e. comparing

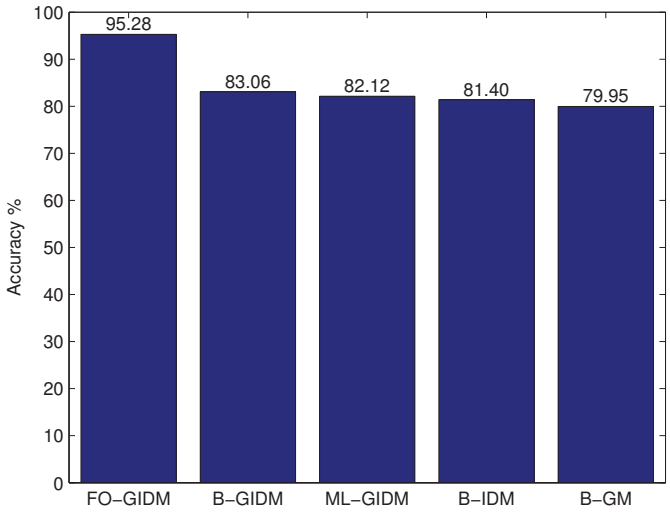


Figure 3.16: Classification accuracies of the Swans dataset.

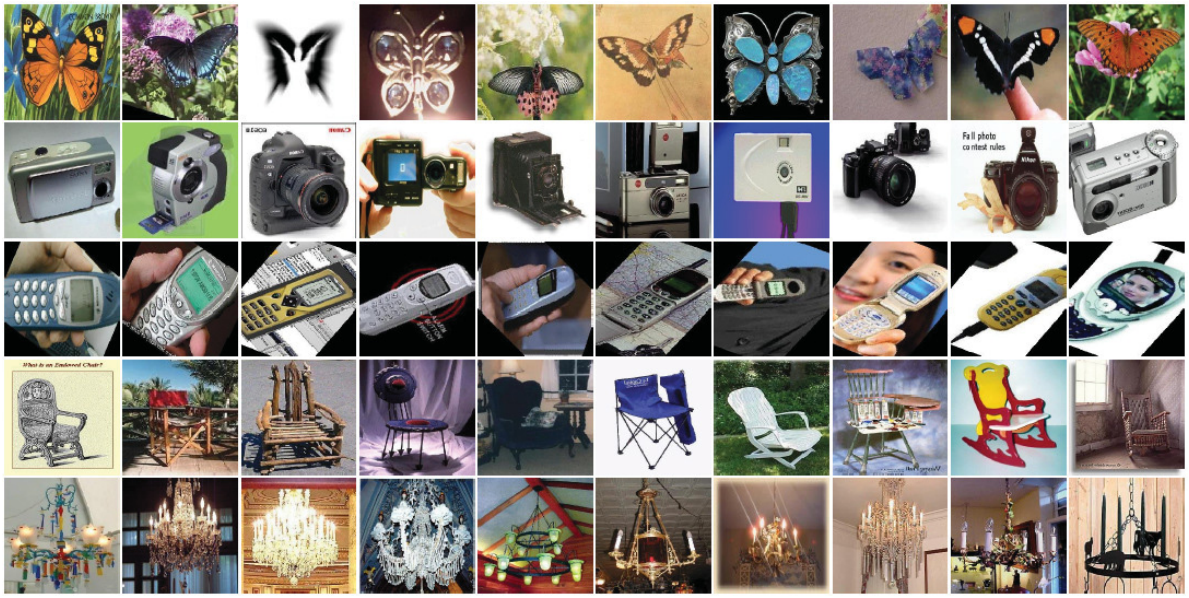


Figure 3.17: Butterfly, Camera, Cellphone, Chair, Chandelier : sample images.

FO-GIDM and FO-IDM). We also integrate the whole framework within the hierarchical model that we have proposed in chapter 2. We consider a subset of the Calltech 101 dataset, which is composed of 5 objects that are Camera, Butterfly, Cellphone, Chair, Chandelier. As we did for the previous experiment, we propose to use the half of images for training, and the other half for testing. We propose to model the hierarchy illustrated by the model shown in Fig. 3.18. We train our model by constructing it online, starting by injecting different classes at each time t. During our experiment, we started by modeling the Camera class, then we sequentially injected the Butterfly,

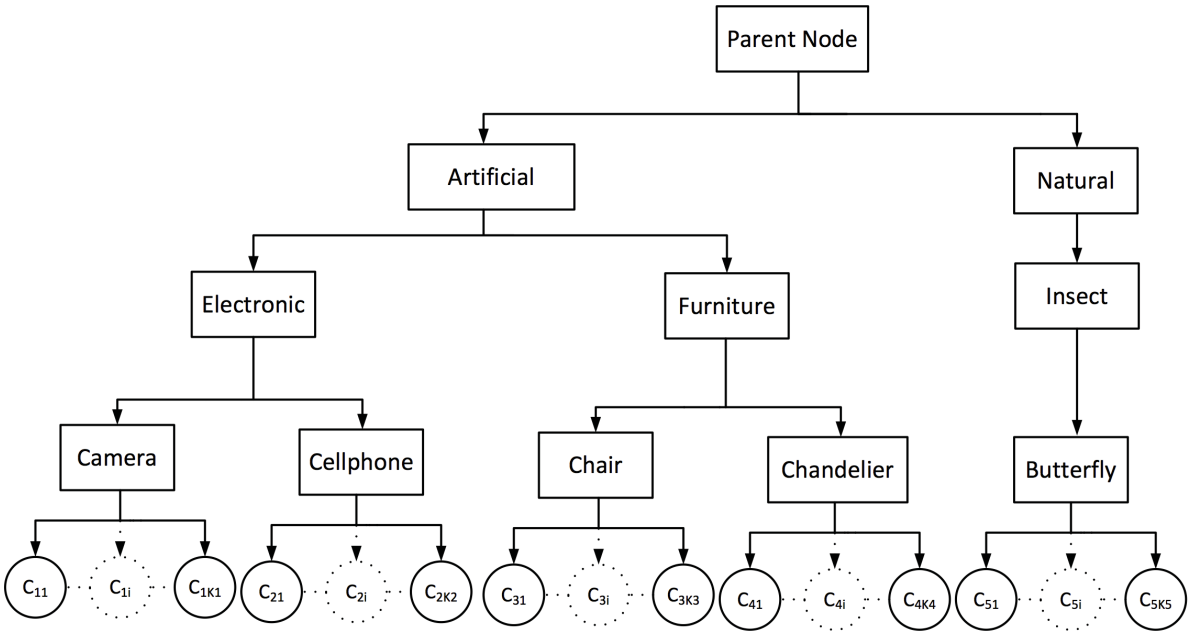


Figure 3.18: Butterfly, Camera, Cellphone, Chair, Chandelier : proposed hierarchical model.

Cellphone, Chair and Chandelier classes. For the different levels of the hierarchy, we report on the accuracy in Table 3.13. We show that the model performs better when using the GID distribution. This can be explained by the fact the GID has a more flexible covariance structure, and that the inverted Dirichlet is a specific case of the GID which makes the latter more flexible in terms of modeling different data shapes in the feature space. Naturally the distance threshold that we use can significantly affect the results especially when dealing with unsupervised clustering, where we do not have an idea about the nature of the coming data. In this experiment we assume that we already have a training phase where we perfectly know the labels of the images. Therefore, we can systematically built our model by considering that the coming data are totally generated by new components. The obtained results are reported in Table 3.13, we notice that the GID mixture outperforms the ID mixture with 86.41% accuracy at the first level against 74.46%, and 86.41% at the second level against 77.72% and finally 90.22% against 83.70% at the third level. This clearly shows that the GID hierarchical mixture outperforms the inverted Dirichlet hierarchical mixture model in the proposed framework. Notice that the accuracy does not change from the first level to the second level when using the FO-GID. This indicates that at the first level there was not any Camera that was misclassified as a Cellphone or vis-versa. Also there was not any Chandelier that was misclassified as a Chair or vis-versa. That is why the accuracy remains the same at the first and second level. The idea behind this work is to form a mental model capable of representing a specific hierarchy proposed by a user via his/her interaction with the system. Suppose for instance that a user interaction with the system records that he/she has chosen some images to be in different

super classes at a specific level of the hierarchy. The labeled training images simulate the user feedback to the system that uses that information to rank non classified images that are represented by the testing images. The system shall propose some images according to the users target classes. Modeling the user interaction using our algorithm, can enable the system to propose some images according to the users needs using, for instance, the ranking of their posteriors.

3.7 Summary

Apart from developing an algorithm to estimate the parameters of a finite GID mixture model and establish a model selection at the system level. We have proposed a novel update strategy for mixture models where a perception parameter is introduced to help the system decide weather it should create new components or update the already existing components when new data are introduced on line. The strategy is based on some probabilistic metrics that we have developed specifically for the GID distribution. The merit of our approach is validated through synthetic data and real-world applications on images datasets. In the next Chapter, we will develop a statistical framework to which we integrate the algorithms that we have designed, and which consists of a search engine for images without an explicit query from the users but rather based on a search process using visual features only.

Chapter 4

A Statistical Framework for Mental Targets Search Using Mixture Models

4.1 Introduction

New technological achievements during the recent years caused the appearance of large data collections that are complex to represent, analyze and search. The amount of information that could be derived from such collections can vary depending on the purpose, context and intention of the users dealing with them. Therefore, many systems adopt query-based structures in order to satisfy context-aware needs from a given data collection [102]. In many real life situations, a search problem occurs naturally when a human being is interested in one or several concepts that exist within a certain amount of data. Such concepts could be visual, textual, or within any other information container that is usually represented by the system in a well defined feature space. A common application of such search processes comes with the recommendation frameworks where the system uses different feed backs from users in order to suggest them what could be categorized as “interesting items" [103, 104]. Some other systems collect data about the users behaviors, then search and suggest advertising items for commercial purposes [105, 106]. In many cases, a user is intentionally searching an item but has no clue about the features that a system is using to represent a concept, or lacks the needed tools and semantics to describe his/her needs and express them. Thus, he/she cannot provide a precise query to the system in order to tune the search process. We refer to this problem as the semantic gap between the user and the system. In order to solve such a problem, some works have proposed a mental matching which consists of a search process during which the system tries to identify the mental target of a given user without having an explicit query. An overview about mental matching can be found in [107] where the authors track the early works on this field and propose a new bayesian framework to search an image category based on

a mental image. They mainly design a system where several images are displayed to a user, and he/she has to select the image that appears to be the closest to the image he/she has in mind, until the system shows a target image. In this chapter, we propose to extend the work of [107] which has the advantage of being query-free, in order to 1) cover multi targets category search, 2) include the possibility of a multiple images selection and a no preference choice when looking for a target category during the search process, and 3) use a new model that serves to model the data and to measure similarities between different images using the GID distribution.

This chapter is structured as follows. In section 4.2 we present the problem and the techniques that have been proposed to solve it, as well as the main contributions of this work. The proposed framework is introduced in section 4.3. In sections 4.4, 4.5, 4.6 and 4.7, we detail the Data Model, Update Model, Answer Model and Display Model, respectively. We present our experimental results in section 4.8. In section 4.9 we interpret the results and we conclude in section 4.10.

4.2 Related Work

When they first appeared, images databases were carefully annotated with keywords by experts, in order to easily browse them and retrieve images by query. With the tremendous increase of the volume of available images that are published on a daily bases by millions of internet users, manual annotation turns to be impossible to realize. To solve this issue, researchers started by proposing an automated tagging based on annotation propagation such as in [108, 109]. Although those methods were successful, it is not always possible to formulate an explicit query in order to retrieve images. Many users lack the necessary vocabulary and are not able to formulate a system understandable query that could lead them to their target images. Thus, many researchers based their work on the query by image content (QBIC) framework that was initially published in [110], where a query is an image that can serve to find similar images within a database. Still, the QBIC system needs example images to serve as a graphical query. Those example images that serve to start the search process are not always available which represents a problem known as the page zero problem. On the other hand, to help the system making suggestion that may interest users, many researchers have proposed the relevance feedback such as in [111–113]. Relevance feedback uses high level information provided by the user about a given concept, using an iterative fashion, in order to adjust the data representation and converge toward the discovery of new elements that represent a value to that user. In order to construct a solid model founded on a strong mathematical background, many model-based framework have been developed for content-based images retrieval. The work in [114] proposes to minimize the probability of retrieval error by combining feature selection

and similarity measures into a Bayesian formulation, while the work in [79] uses the context-aware formulation to identify several cases where the context serves to generate more accurate recommendations. Some other works propose hybrid models that combine both generative and discriminative learning, such as in [53] where the authors propose a hybrid model-based framework for efficient image retrieval . While those works were not dedicated to the page zero problem, they prove that statistical frameworks can give accurate results. The first model-based work that was trying to solve the page zero problem is the mental matching which tries to find a “mental image" that resides in the mind of a user without having an explicit query. It was first pioneered by the work in [115]. The search process was done iteratively, and at every round the user was asked to choose the closest image to the target image that resides in his/her mind, among two displayed images by the search engine. The work in [115] served as a basis for the framework proposed by [107] which extended their work to cover semantic target category search using a bayesian model that includes a pair of positive and negative answer models. The statistical framework proposed in [107] has been adopted and extended for large-scale image collections of millions of images in the HEAT retrieval system proposed in [116] , and extended in [117]. Still, the framework proposed by [107] is limited to one single target category within the same search process. The user has to repeat the process N times if he/she is targeting N categories. Also, a user is always forced to choose an image that is closest to the image in his/her mind even if there is not anyone displayed that matches the targeted mental images. Moreover, a user is allowed to have one single selection, while he/she could be interested in several images that could lead to the target category. In this chapter we extend the work proposed in [107], to cover many target images search within a single mental search process, including the non selection and multi selection preferences. We also use a new data model constructed by the generalized inverted Dirichlet (GID) mixture that has proven its capability to robustly model and cluster multi-dimensional data [85, 118]. This work focuses on visual concepts and mainly deals with images but it is noteworthy to mention that a concept can be any other information representation that a user can use to interact with the system.

4.3 The Framework Structure

Let $\Omega = \{X_1, X_2, ..., X_N\}$ be a set of N images. The main purpose of this work is to determine a subset $S \subset \Omega$ that represents a set of target images that matches the visual interests in the mind of a given user. S could be formed by diverse images' categories $\{S_j\}$ that do not necessarily have visual similarities at the system level, or have different semantic meanings for the user. Thus, S

can be written under the following form :

$$S = \bigcup_{j=1}^{M} S_j \tag{4.1}$$

where $S_j \subset \Omega$, $j = 1,...,M$, are the different target classes forming S and M represents their number. We assume that the user knows exactly how many classes M he/she is targeting. We also assume that if an image that belongs to a target class is displayed, the user will be able to identify it. If the user is not interested anymore in a given target class S_k the search problem is reduced to find :

$$S = \bigcup_{\substack{j=1 \\ j \neq k}}^{M} S_j \tag{4.2}$$

The mental search consists of several iterations during which a set of different images $D \subset \Omega$ is displayed to the user. If $D \cap S_j = \emptyset$ the user specifies the set of images $\{X_i\}_j \subset D$ that are the closest to what he/she might have in mind in order to approach S_j. We associate a random variable Y_{X_k} with each image $X_k \in \Omega$ such that $Y_{X_k} = j$ if $X_k \in S_j$ and $Y_{X_k} = 0$ if $X_k \notin S$. In other terms we have :

$$\begin{aligned} S &= \{X_k \in \Omega \ / \ Y_{X_k} \neq 0\} \\ S_j &= \{X_k \in \Omega \ / \ Y_{X_k} = j\} \end{aligned} \tag{4.3}$$

During the search process Y_{X_k} is updated according to the user responses to the different displays $\{D_t\}$. Let B_t denote the responses of the user for the first t displays. The distribution of $Y_{X_k} = j$ knowing B_t is represented by :

$$p_t(X_k)_j = P(Y_{X_k} = j | B_t), \ \ j = 1,...,M \tag{4.4}$$

As S is seen by the system as a random subset, we have no prior knowledge about it. Therefore we initialize the values of $p_0(X_k)_j$ as follows :

$$P_0(X_k)_j = \frac{1}{M+1}, \ \ j = 1,...,M, \ \ k = 1...N \tag{4.5}$$

which is a uniform distribution over all the $M+1$ classes, the $(M+1)^{th}$ component is an eventual additional class which represents the negative set of images that represent no interest to the user. Note that if we consider the specific case $M = 1$ we find the same framework structure proposed in [107]. Our work treats a generalized case where a user can choose not to select any image, or to select multiple images. We also suggest a different model describing the data that is discussed in section 4.4. The framework we propose is as follows : we design a graphical interface through

which the user will interact with the system in order to identify his/her target classes within a dataset using the visual features only. This interaction is realized through different iterations, during which the system would display at iteration t a set of images D_t such that :

$$\begin{aligned} |D_t| &= N_t \\ N_t &= \sum_{j=1}^{M} N_t(j) \end{aligned} \tag{4.6}$$

N_t is the total number of displayed images at time t, and $N_t(j)$ is the number of images proposed for a specific target class S_j, $j = 1,...,M$. The user would select the images that he/she might find interesting and specify their corresponding classes among the M available classes defined at the beginning. e.g. say we have a scrolling list that can be used for each displayed image in order to specify its class j (see Fig.4.1). The user responses enable the system to update its parameters and come up with a new display D_{t+1}. The user repeats the same selection process until he/she is satisfied of the search process concerning a target class S_j. In order to develop a such search

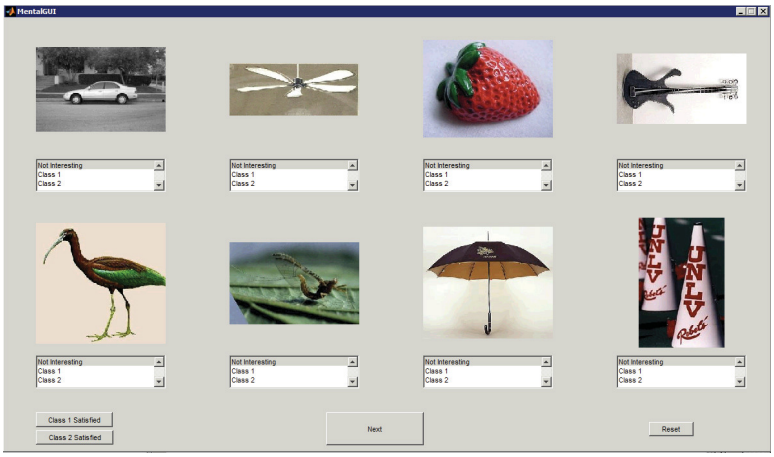


Figure 4.1: Example of the first display of 8 images

process, we mainly design four models :

- **Data Model**: This component covers the images representation at the system level, and how the system perceives the data. It can be constructed in a supervised or unsupervised way.
- **Update Model**: This component computes for each image category, $p_{t+1}(X_k)_j$ in terms of $p_t(X_k)_j$ and the user's answers at step t.
- **Answer Model**: This component specifies for each image $X_k \in \Omega$ the probability that a user chooses an image $X_i \in D$ given $Y_{X_k} = j$.

- **Display Model**: This component specifies which images to display at step t, basing on the search history.

In the following we develop each of these components.

4.4 Data Model

We propose to model the data using the statistical framework developed in the previous chapter using the GID mixture model. We use the local Histogram of Oriented Gradient (HOG) descriptor [75], in order to extract the images features. The GID mixture model supposes that the data are positive, which makes the HOG suitable for consideration as it generates descriptors having positive values. In our experiment each image was represented by a feature vector whose dimension is $D = 81$. Let us consider a set Υ of N D-dimensional vectors, that represents the features extracted from the set of images $\Omega = \{X_1, X_2, ..., X_N\}$, such that $\Upsilon = \{\vec{F}_1, \vec{F}_2, ..., \vec{F}_N\}$, where $\vec{F}_n$ is the feature vector of the image X_n. Let C denotes the number of different components forming a flat mixture model at the system level. We assume that Υ is controlled by a mixture of GID distributions such that the vectors follow a common probability density function $p(\vec{F}_n|\Theta)$, where Θ is the set of its parameters such that :

$$p(\Upsilon|\Theta, \vec{\pi}) = \prod_{i=1}^{N} \Big(\sum_{j=1}^{C} \pi_j \prod_{l=1}^{D} \frac{\Gamma(\alpha_{jl} + \beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} \frac{F_{il}^{\alpha_{jl}-1}}{(1 + \sum_{k=1}^{l} F_{nl})^{\gamma_{jl}}} \Big) \tag{4.7}$$

where $\Theta = \{\vec{\theta}_1, \vec{\theta}_2, ..., \vec{\theta}_C\}$, with $\vec{\theta}_j = (\alpha_{j1}, \beta_{j1}, ..., \alpha_{jD}, \beta_{jD})$, $\gamma_{jl} = \beta_{jl} + \alpha_{jl} - \beta_{j\{l+1\}}$ for $d = 1, ..., D$ with $\beta_{j\{D+1\}} = 0$. $\vec{\pi}$ is the vector of mixing weights such that $\vec{\pi} = (\pi_1, \ldots, \pi_C)$, $\pi_j > 0$ and $\sum_{j=1}^{C} \pi_j = 1$.

We use the same methodology used in chapter 3 in order to cluster the data into classes at the system level.

4.5 Update Model

The update model updates the posterior probabilities $p_t(X_k)_j$ according to the responses of the users. Let us denote X_{D_t} the user response to the display D_t, then we have :

$$X_{D_t} = \{X_{D_{t0}}, ..., X_{D_{tM}}\} \tag{4.8}$$

X_{D_t} is the whole user response set for a display D_t, and $X_{D_{tj}}$ is the user selection for a specific class j at time t for the same display D_t. As we consider $(M+1)$ selection sets in Eq.(4.8), we consider

the non selected images as a “user selection" for the negative images class, so when a user decides not to select specific images, this is considered as being a choice for the class containing the “not interesting" images set. Naturally, the system has no prior knowledge about the user mental targets, therefore, in order to constitute D_1, the most trivial method is to select a number N_1 of images with a selection probability equal to $\frac{1}{|\Omega|}$. A more adequate method is to use our data model in order to suggest the initial display. Recall that we have a mixture model composed of C clusters that group similar images together. We propose to randomly select M clusters with the probability $\frac{1}{C}$, and then select from each selected cluster j, $N_1(j)$ images to display in D_1, with the probability $\frac{1}{|j|}$, with $|j|$ the cardinal of the cluster j and $j = 1...M$. Thus, we can construct D_1, composed of N_1 images such that $N_1 = \sum_{j=1}^{M} N_1(j)$, with $N_1(j) \geq 0$, $j = 1...M$. At time t, the search history is expressed as follows :

$$B_t = \{X_{D_1}, X_{D_2}, ..., X_{D_t}\} \tag{4.9}$$

we construct B_{t+1} such that

$$B_{t+1} = B_t \cup \{X_{D_{t+1}}\} \tag{4.10}$$

Note that our proposed model is not limited to a binary model for each target class, where we have a positive and a negative model as proposed in [107]. We rather have $M+1$ classes, with the $(M+1)^{th}$ representing the negative class. The consideration of M binary models, where we have for each class a positive and negative model, increases the complexity and time cost when targeting many classes. Moreover when considering $M+1$ classes, we take into consideration that when an image $X_k \in \Omega$ does not belong to a target class S_j, it should not be systemically considered as a negative image, as it could belong to another target class S_k, with $k \neq j$. The selection probability of a displayed image X_D when $D = D_{t+1}$ and given $Y_{X_k} = j$, in order to approach the target class S_j, can be written under the form:

$$p(X_D = i | Y_{X_k} = j, D_{t+1} = D) = p_+(i, X_k, D)_j \tag{4.11}$$

We consider Eq.(4.11) as being the “positive answer model" for the class j, and basing on the work of [107], we update the distribution of $Y_{X_k} = j$ knowing B_{t+1} as follows :

$$\begin{aligned} p_{t+1}(X_k)_j &= p(Y_{X_k} = j | B_{t+1}) \\ &= \frac{p(X_D = i | Y_{X_k} = j, D_{t+1} = D) p_t(X_k)_j}{C_{t+1}} \end{aligned} \tag{4.12}$$

$$= \frac{p_+(i|X_k,D)_j p_t(X_k)_j}{C_{t+1}}$$

where C_{t+1} is a normalizing factor such that :

$$C_{t+1} = \sum_{j=0}^{M} p_+(i|X_k,D)_j p_t(X_k)_j \tag{4.13}$$

note that we have :

$$\sum_{j=0}^{M} p_t(X_k)_j = 1, \quad \forall t, \ \forall X_k \in \Omega \tag{4.14}$$

4.6 Answer Model

The answer model is based on the data model. The assumption that the user is not satisfied yet by the displayed images still hold. When considering the mixture model, we assume that the more an image is close to a mental target image belonging to a class j, the more likely their posterior probabilities will be close to each others. Let $(X_i)_j$ be the image selected by the user to be the closest image to the class j and let $(X_k)_j \in S_j$. We define the pseudo metric $d((X_i)_j,(X_k)_j)$ that we denote $d_j(X_i,X_k)$, between a selected image $(X_i)_j$ and a target image $(X_k)_j$, where $Y_{(X_k)_j} = j$, as follows :

$$d_j(X_i,X_k) = \sum_{c=1}^{C} \left(p(c|\vec{F}_i,\Theta,\vec{\pi}) - p(c|\vec{F}_k,\Theta,\vec{\pi})\right)^2 \tag{4.15}$$

Notice that $d_j(X_i,X_k)$ can be turned into a full metric distance, if we consider an equivalence classes such that : if $d_j(X_i,X_k) = 0$ it implies that $X_i \sim X_k$. We adopt the answer model form proposed by [107] for each target class S_j such that :

$$p_+(i|X_k,D)_j = \frac{\phi_+(d_j(X_i,X_k))}{\sum_{X_d \in D} \phi_+(d_j(X_d,X_k))} \tag{4.16}$$

where $\phi_+(x)$ is a monotonically decreasing function, such that if $X_i,X_j \in D$ and $d_j(X_i,X_k) < d_j(X_j,X_k)$ we expect $p_+(i|X_k,D)_j > p_+(j|X_k,D)_j$. We propose to consider $\phi_+(x)$ as a Gaussian distribution with mean $\mu = 0$ and standard deviation σ. Thus, we have :

$$p_+(i|X_k,D)_j = \frac{e^{-\frac{1}{2}\left(\frac{d_j(X_i,X_k)}{\sigma}\right)^2}}{\sum_{X_d \in D} e^{-\frac{1}{2}\left(\frac{d_j(X_d,X_k)}{\sigma}\right)^2}} \tag{4.17}$$

σ can be seen as a precision parameter, to specify how the distance metric should be perceived.

4.6.1 The No Preference Selection Case

As we mentioned before, the user has the choice not to select any image, therefore, all the images that are not selected, are considered to be part of the “not interesting" class. When a user does not select any image that might be semantically and visually close to what he/she has in mind concerning a target class j, we propose to base the next suggestions on the last selection belonging to that class and reestablish all the display process basing on it such that :

$$X_{D_t j} = X_{D_{t-1} j} \tag{4.18}$$

The special case of not having a previous selection comes to surface when a user decides that the earliest displays are not interesting and considers that there is not any interesting image targeting the class j during all the previous iterations such that :

$$X_{D_i j} = \emptyset, \quad i = 1...t \tag{4.19}$$

In that case we use the negative answer to have an implicit selection of the target class j in order to be able to update our model. We assume that in terms of the distance metric, the further an image is from the images of the negative class, the more probable it will belong to a target class. Thus, we consider a set $\{X_h\}$ that represents those furthest images from the selected images of the negative class and consider it as a selection for the target class j. This assumption enables the model to conserve its integrity in the early displays when no selection of target classes takes place. In this case we consider that D implicitly contains $\{X_h\}$ and the equations using D are used with a new $D = D_{implicit}$ such that :

$$D_{implicit} = D \cup \{X_h\} \tag{4.20}$$

still, when a target class has no selections, the system will not consider the implicit selections as a real user selection and will display a random selection from different non-displayed clusters when it suggests images for that class, as described in section 4.5.

4.6.2 The Multi-Selection Case

The multi selection case takes places when a user wants to select more than one single image in order to approach a target class j. We assume that the user selections are independent, so if we consider that the user selects J images to target the class j, we calculate the probability of the multi

selection as follows :

$$p_+(i_1,....,i_J|X_k,D)_j = \prod_{l=1}^{J} p_+(i_l|X_k,D)_j \ , \quad J \leq |D| \tag{4.21}$$

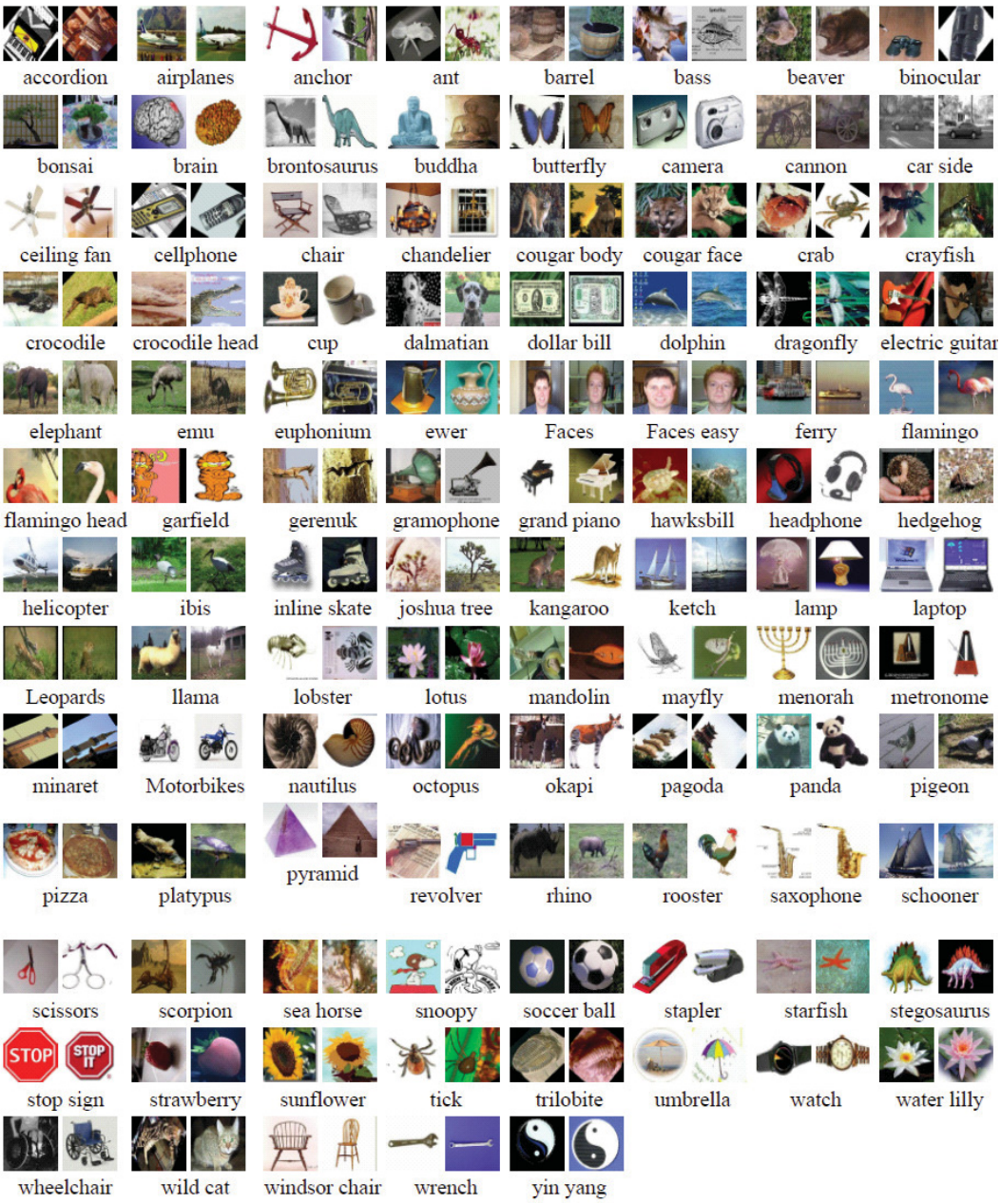


Figure 4.2: CalTech 101: two randomly chosen samples for each category

4.7 Display Model

According to our framework, at time t, we should display N_t images including $N_t(j)$ suggestions for the target class j. Naturally, an image should not be displayed twice, and should not have been already displayed in the past. In order to suggest images for a target class j, we rank the images, according to their $p_t(X_k)_j$ and we pick up the first $N_t(j)$ images. We define D_{t+1} as follows :

$$D_{t+1} = \bigcup_{j=1}^{M} D_{t+1_j} \tag{4.22}$$

where

$$D_{t+1_j} = \underset{\substack{D_j \subset \{\Omega \setminus B_t\} \\ |D_j| = N_{t+1}(j) \\ D_j \cap \{D_i\} = \emptyset}}{\arg\max} \sum_{X_k \in D_j}^{N_{t+1}(j)} p_t(X_k)_j \tag{4.23}$$

where $\{D_i\}$ in Eq.4.23 is the set of the already picked images to be displayed in the display D_{t+1} for any other target classes k, such that $k \neq j$. We also propose, at first, to rank only the images that belong to the same cluster of the images selected by the user, that is defined by the mixture model, which leads to suggest images that are semantically alike. If a cluster does not have enough images to display, we extend the application of Eq.4.23 on the whole set of images Ω. If the user selects images from different clusters we have two choices, either we select a defined number from each cluster to form the display of the target class j, or we build a hierarchical model forming a parent cluster containing the sub clusters (such as in chapter 2) and then we select the images from it using Eq.4.23.

4.8 Experimental Results

Figure 4.3: Target images from “buddha" and “sunflower" classes

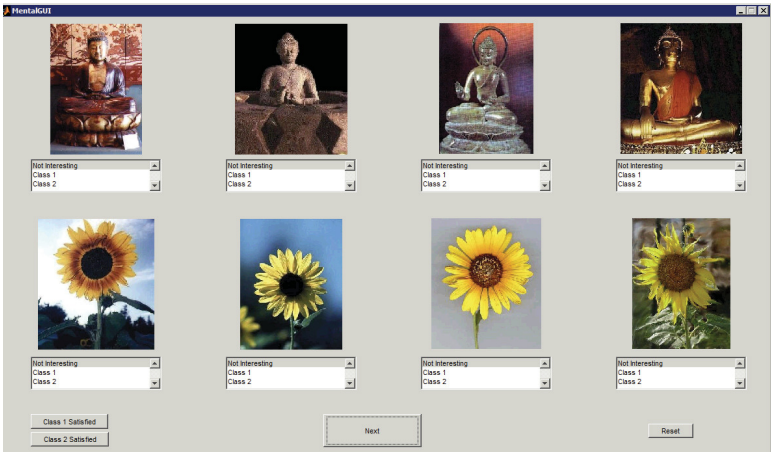
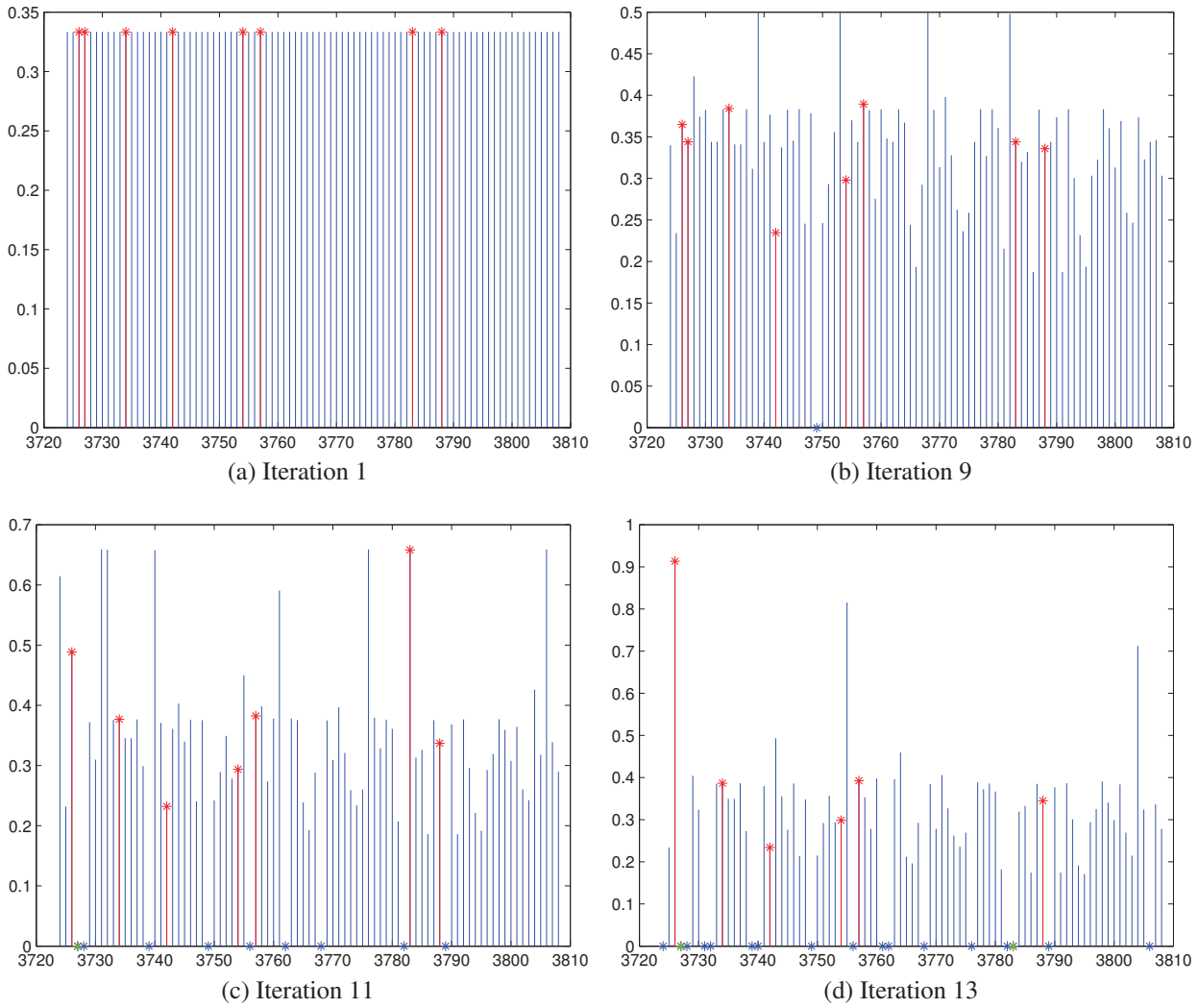


Figure 4.4: Display 18

In order to test our framework we consider the publically avaiable Caltech101 dataset [119]. Caltech101 is composed of 8676 images of different objects representing 101 categories. These objects have different shapes and colors as shown in Fig.4.2 which illustrates two randomly chosen samples from each category of the dataset. In our experiment we used all the 101 categories, and we have considered two target classes such that $M = 2$. The construction of the data model was established using the GID mixture model basing on the construction strategy that is proposed in chapter 3, and using the HOG features. The modeling process resulted 124 components that compose the GID mixture. Those components can be considered as the system perception for the data in the HOG feature space. In order to estimate the number of representing classes of the data, we used the MML criteria, the GID Kullback-Leibler distance and a perception tolerance rate equal to 20% to measure the similarity between the different components as it has been proposed in chapter 3. We designed a Graphical User Interface (GUI) that displays eight images at each iteration and we fixed the display number for each category such that $N_1(1) = N_1(2) = 4$, which means that the system should suggest four images from each target class. We also considered a precision parameter $\sigma = 1$ for the Eq.4.17 to have the standard normal distribution.

The system starts by displaying eight images randomly, such that an image from each constructed component of the mixture is selected as explained in section 4.5. Fig.4.1 shows a screenshot of the developed GUI that shows a first display D_1. The user selection is set, by default, to 'not interesting'. If some interesting images are displayed, the user selects them by specifying their corresponding classes. Basing on those selection, the system tries to find images that could belong to the mental target classes of the user. Using a normal PC, the suggestion of a new display takes an average of 1 second. The performance analysis of such systems remains hard to interpret as it is related to the satisfaction of users, and it involves human psychology and decision making. As a first approach, we have tried to measure users satisfaction by asking 20 users that are not familiar

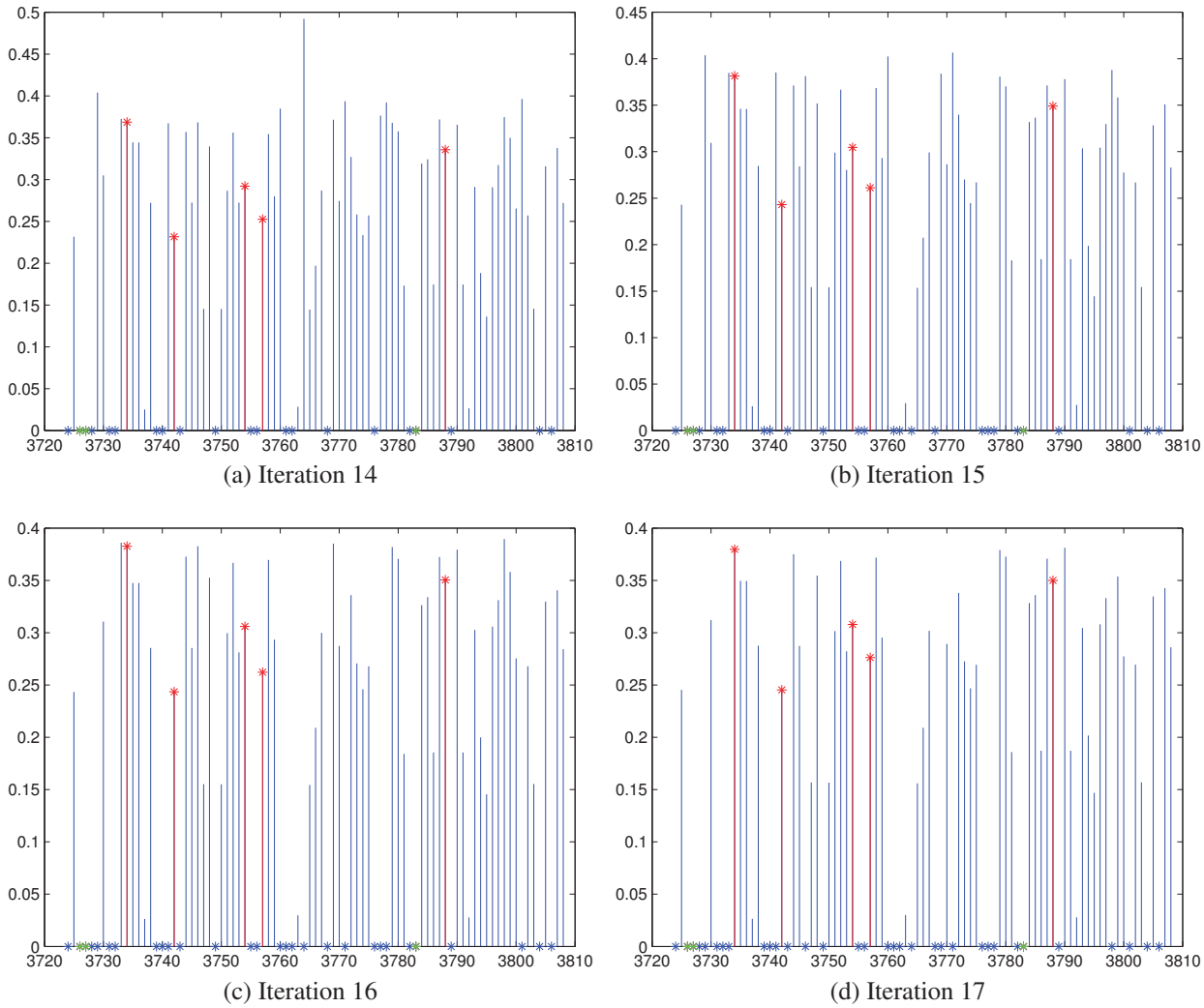


(a) Iteration 1 (b) Iteration 9

(c) Iteration 11 (d) Iteration 13

Figure 4.5: Class “buddha" : $p_t(X_k)_j$ evolution during different iterations : 1,9,11,13

with the system to use the search interface in order to target some images from a target class. We asked them to give a satisfaction grade that goes from 1 to 10, one being the worse grade and 10 represents the full satisfaction grade. We obtained an overage equal to $7.95/10$. We noticed that the dissatisfaction usually occurs with users who have missed an image from their category class, or mislead the system toward other categories. Indeed, it takes 16 iterations, for the system to show at least one image from each component of the mixture, considering 124 components with a screen size of 8 images to display. For those who miss an image that would lead them to the component containing their target class, they may spend a longer time to find their target images which lower their satisfaction of the system. In order to quantify the displaying results, we propose to illustrate how the posterior $p_t(X_k)_j$ changes over time. We choose two target classes that are constructed from the categories “sunflower" and “buddah". Our main purpose during the search



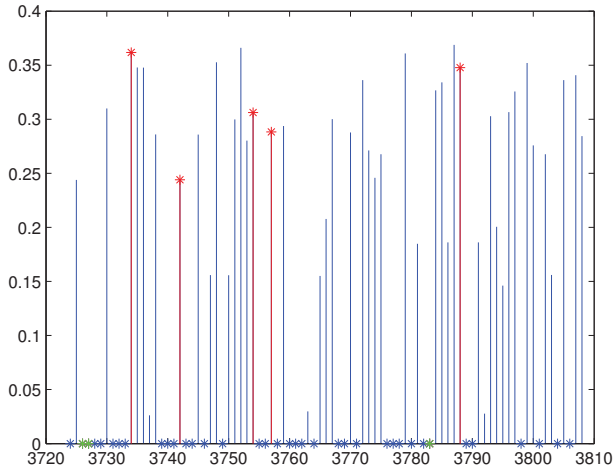
(a) Iteration 14 (b) Iteration 15

(c) Iteration 16 (d) Iteration 17

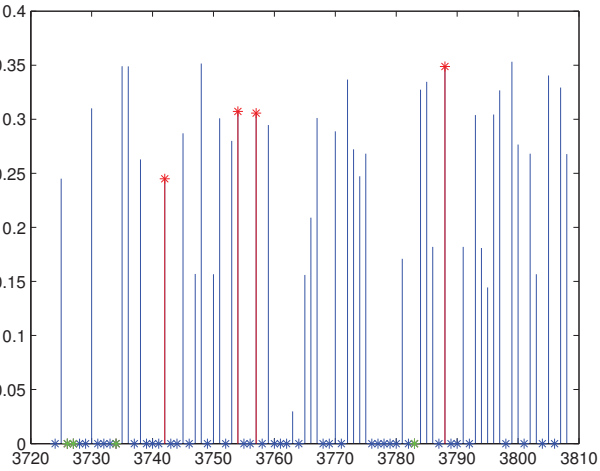
Figure 4.6: Class “buddha" : $p_t(X_k)_j$ evolution during different iterations : 14-17

process is to find eight specific images from each class. For the “sunflower", we propose to find the target class that represent the sunflowers with a tall stem, and for the “buddah" we propose to pick up the buddah that are constructed of gold. Fig.4.3 shows the 16 targeted images.

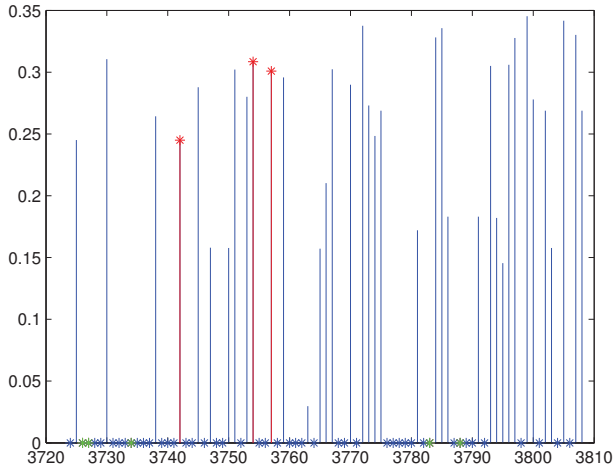
We report on the results after 23 displays, in Figs.4.8 and 4.12, where we illustrate the posterior probability of the “buddha" class and the “sunflower" class, respectively. The X axis represents the image index, and the Y axis represents the posterior probability. The blue stars shows the already displayed images that do not make part of the target images set, while the green ones represent the target images that are already displayed. The lines with red stars show the probabilities of target images. At the first display all the images have the same probability as the system has no input from the user. The first appearance of an image of buddha takes place in iteration 9 as shown in Fig.4.5b, and the first appearance of a sun flower image takes place in



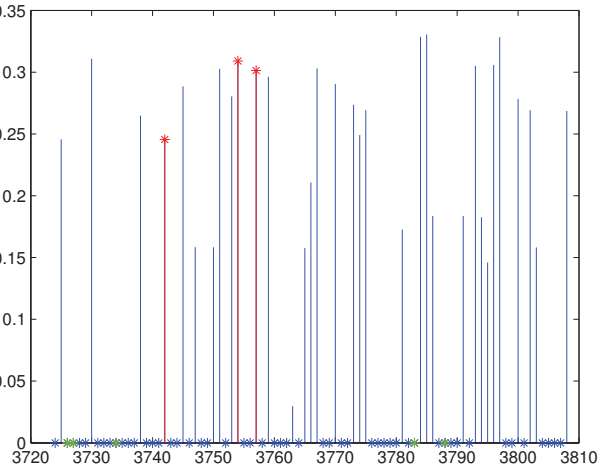
(a) Iteration 18



(b) Iteration 19

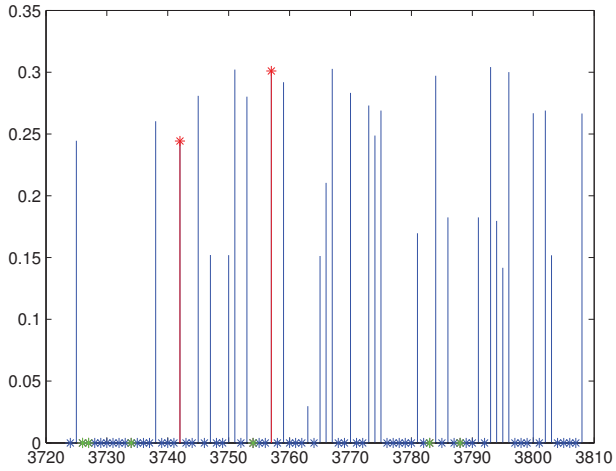


(c) Iteration 20

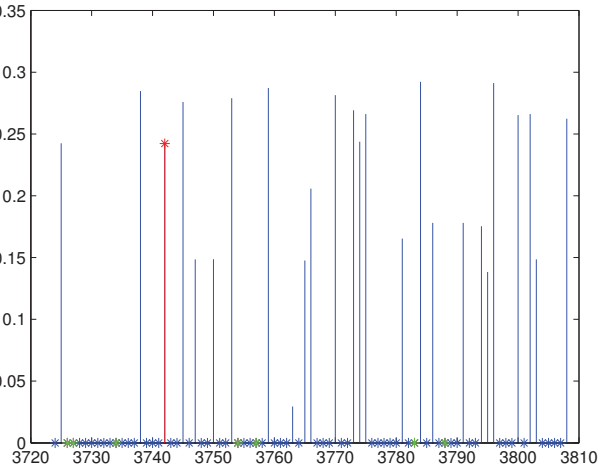


(d) Iteration 21

Figure 4.7: Class “buddha" : $p_t(X_k)_j$ evolution during different iterations : 18-21

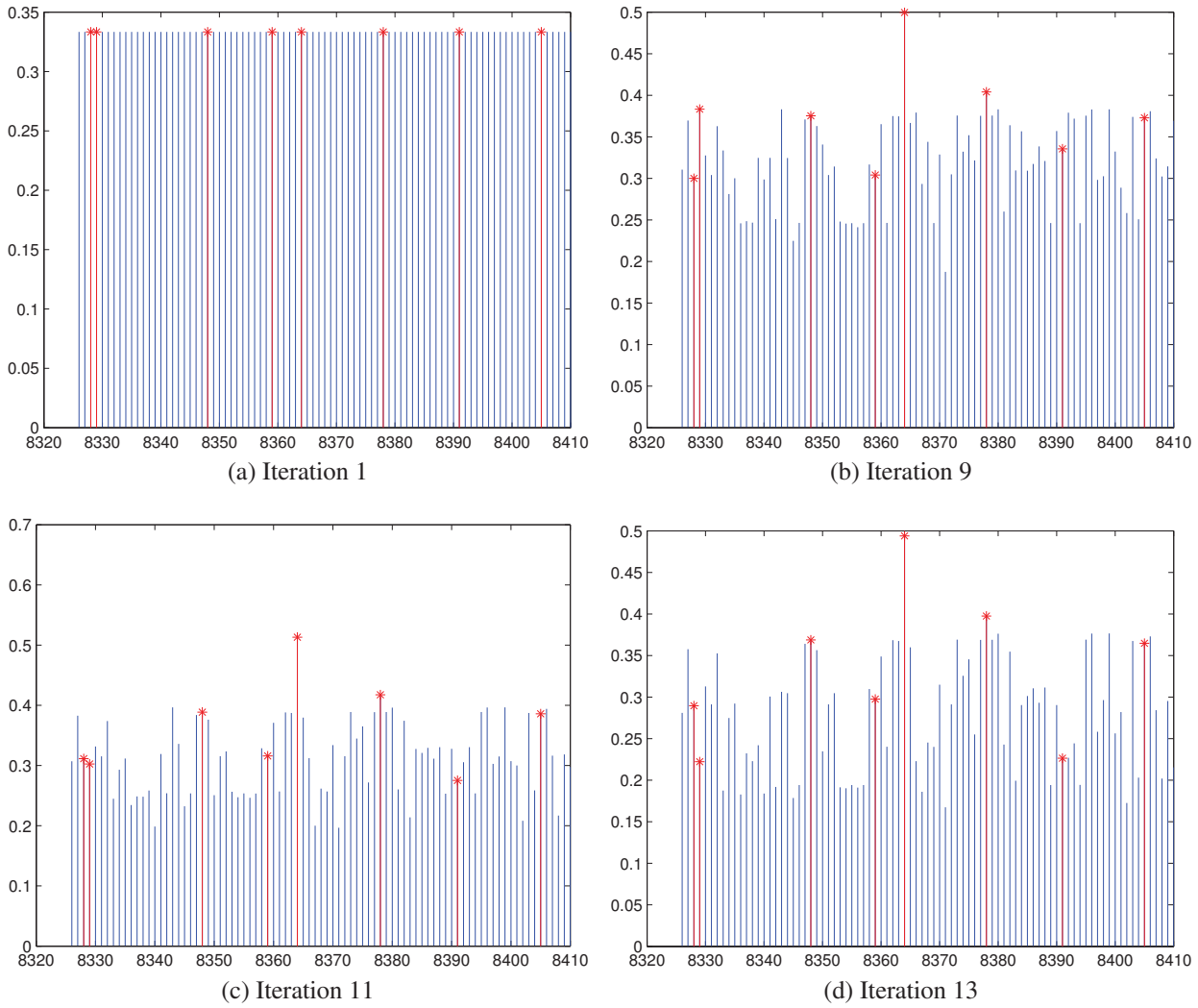


(a) Iteration 22



(b) Iteration 23

Figure 4.8: Class “buddha" : $p_t(X_k)_j$ evolution during different iterations : 22-23



(a) Iteration 1

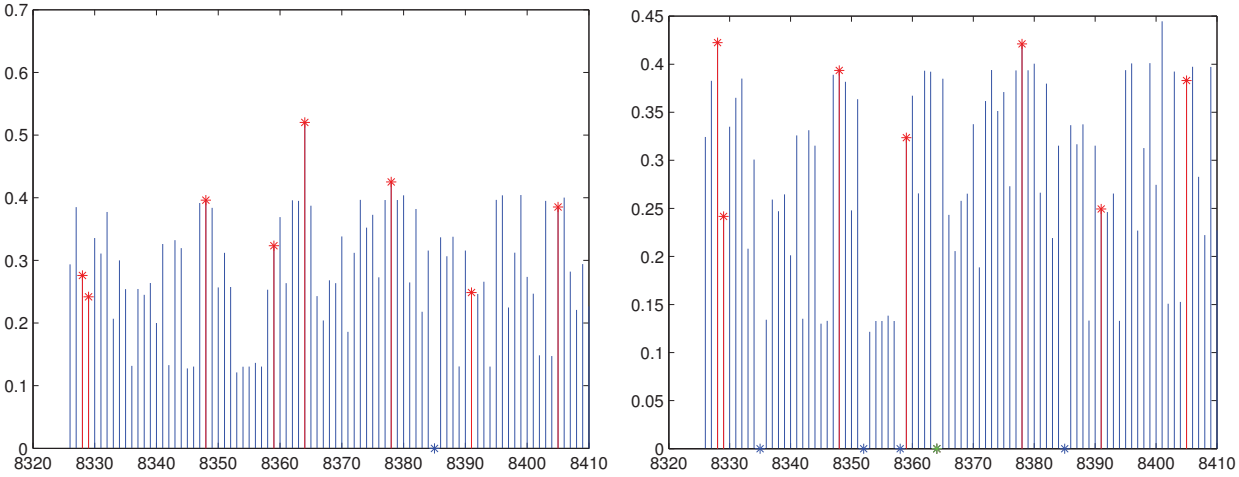
(b) Iteration 9

(c) Iteration 11

(d) Iteration 13

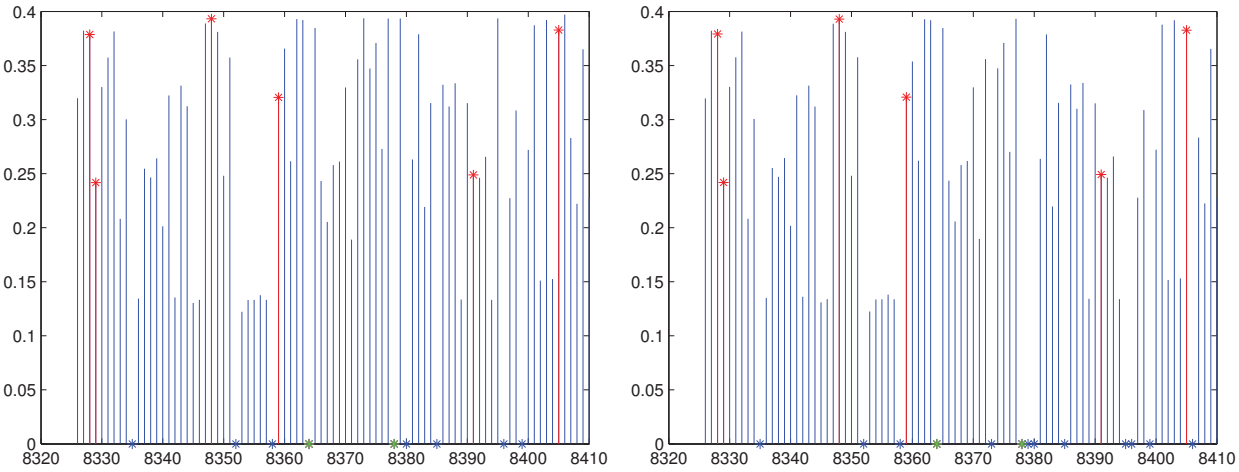
Figure 4.9: Class “sunflowers" : $p_t(X_k)_j$ evolution during different iterations : 1,9,11,13

Iteration 14 as shown in Fig.4.10a. As the images of buddha appear before the appearance of the “sunflower" images, it is expected that the system displays a target image from the buddha set before showing a target image from the “sunflower" set. The first target image of buddha appears in iteration 11 as shows in Fig.4.5c, and the first appearance of a target image of a “sunflower" appears in iteration 15 as shown in Fig.4.10b. Fig.4.4 shows the display at Iteration 18, note that some target images have been already displayed at that stage. After 23 displays, the system was able to suggest 5 target images of “sunflower" and 7 target images of “buddha", which makes a total of 12 images out of 16.



(a) Iteration 14

(b) Iteration 15



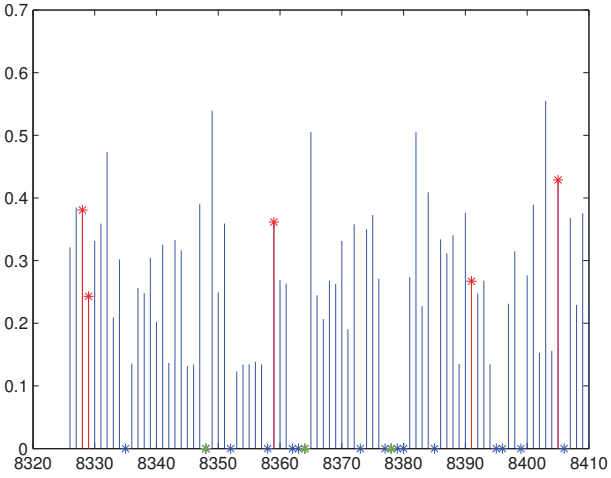
(c) Iteration 16

(d) Iteration 17

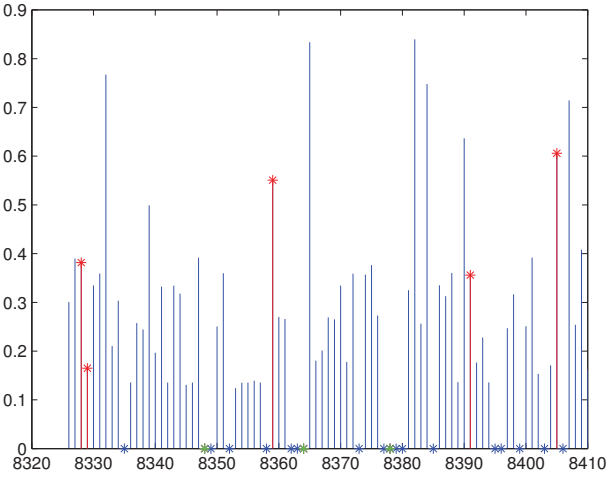
Figure 4.10: Class “sunflowers" : $p_t(X_k)_j$ evolution during different iterations : 14-17

4.9 Discussion

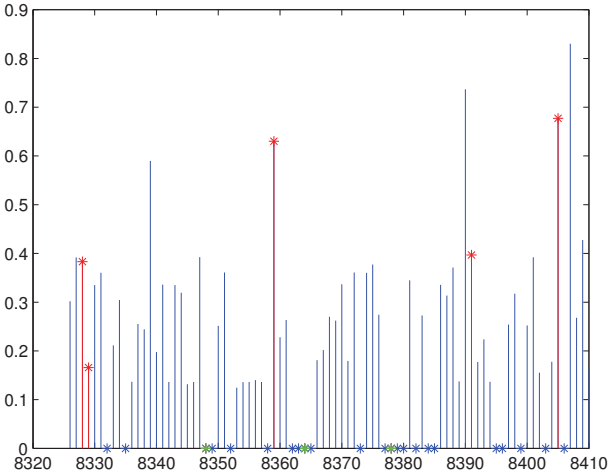
The system was able to converge toward the specified needs of the user, nevertheless it is noteworthy to mention that the framework is still very sensitive to the used features in order to measure the similarity between different concepts. The distance metric that we have used combines the HOG features and the probabilistic data model that we have constructed, further improvement may be done by using new features in order to compare the similarity between images, and new metrics approaches such as in [120]. Also it is clear that the better the clustering of the data, the better are the suggestions of the system. The construction of the model is semi supervised in terms of the class injections, but unsupervised in terms of the representation of each class by a certain number of components basing on the MML criterion. Such framework may be built using a totally unsupervised data model using any clustering technique including the mixture models. The search



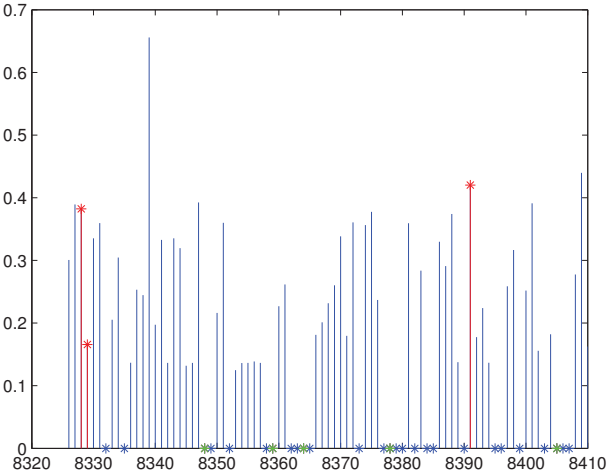
(a) Iteration 18



(b) Iteration 19

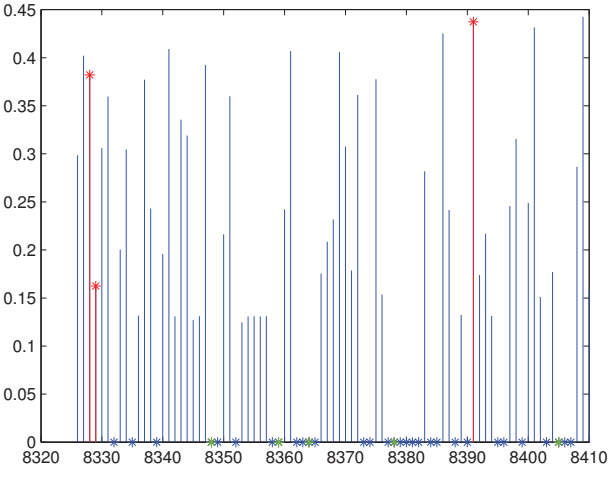


(c) Iteration 20

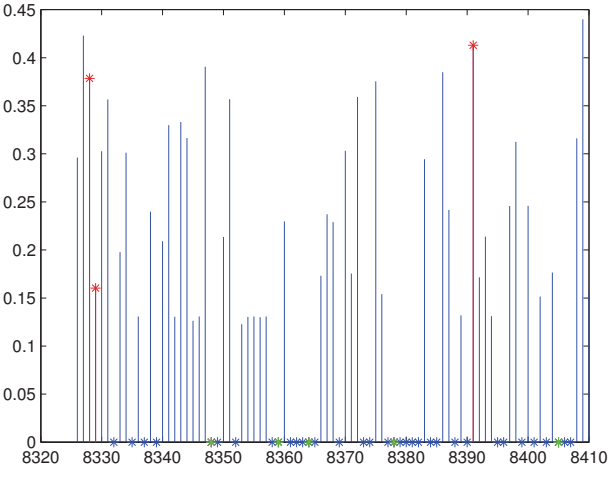


(d) Iteration 21

Figure 4.11: Class “sunflowers" : $p_t(X_k)_j$ evolution during different iterations : 18-21



(a) Iteration 22



(b) Iteration 23

Figure 4.12: Class “sunflowers" : $p_t(X_k)_j$ evolution during different iterations : 22-23

process can be longer when a component contains a large subset of images, another perspective of research can be dedicated to find more appropriate techniques to discard “not interesting" images within a component. The strategy followed by [107] to discard clusters that are not interesting to the user is not appropriate, as a parent cluster that is discarded could contain sub clusters that are interesting to the user. When compared with the work in [107], we introduced the possibility of having multiple selections, and also the no preference option. In many cases the user decides that many images are close to his/her target image and selects them all. Also, having a no preference selection enables the user not to mislead the system which may happen in the framework proposed by [107]. We have introduced the possibility of having many target classes within the same search process. Indeed, in many cases, the target class is composed of many sub classes that can be mapped to several target classes, and the user does not have to redo the search process in order to find them. For example, in the framework proposed by [107], it is not possible for the user to have a suggested display that contains a mix of pictures including “buddha" and “sunflower", as the system would only suggest one target class. The framework we have proposed is rather a generalization to what has been proposed to cover multiple target classes within a single framework. We also do not discard any cluster, but we rather zoom on a cluster to select images from it, and other clusters can reappear according to the choices of the user.

4.10 Summary

We have proposed a statistical framework where a search process without an explicit query but rather basing on the interaction of users with system and using visual features only. We have developed a bayesian formulation that provides the possibility of searching multi target classes within the same search process. The similarity between images is quantified using a GID mixture that also serves to model data. The merit of the model is shown through experiments including real users and we present a case study of a search process that gives promising results in terms of number of iterations needed to find the mental target classes within a given dataset. In the next Chapter we propose to improve the data model that is based on the approache developed in chapter 3. It is clear that a better data clustering leads to a better performance, therefore we will propose a new model to estimate the parameters of GID mixtures basing on variational Bayesian inference, and include a features selection strategy in order to better distinguish objects classes.

Chapter 5

Variational Bayesian Inference for Infinite Generalized Inverted Dirichlet Mixtures with Features Selection

5.1 Introduction

In chapter 3, we have proposed a new methodology in order to update a given model when new data arrive on line. The approach proposes to establish a first model of the arriving data, then compare its components with the existing components in the model, and decide wether 1) to create new components in the final model, or 2) update the existing components of the old model, or 3) do both. We have also introduced a user perception parameter that can help the system to have a data representation, that we can use to form a hierarchical model which groups sub-clusters in order to form a parent cluster that does not necessarily have similar data in the system feature space but have the same meaning in the users semantic as shown in chapter 2. Still the challenging point in the proposed model is to create the new arriving data model. Previously, in chapter 3, we have considered five criteria in order to establish model selection, namely the minimum message length (MML) [24], Akaike information criterion (AIC) [25], minimum description length (MDL) [26], mixture MDL (MMDL) [27], and LEC [17]. Yet, these approaches are demanding in terms of computional cost as we have to establish N complete estimations in order to select a model among N models. Also the use of Newton-Raphson technique within the expectation maximization (EM) framework [19] is not always performant, as 1) it is not guaranteed to converge in general, 2) is prone to finding poor solutions in the case of multi-modal distributions [121] and 3) is dependent on initialization [17, 122]. In order to mitigate these problems many researches

considered pure Bayesian approaches, where the parameters are considered to be random variables and then follow probability distributions called priors that describe our knowledge before using data [123], e.g. Bayesian frameworks using Markov chain Monte Carlo (MCMC) have been proposed in [124, 125], and a non parametric Bayesian approach based on Dirichlet processes is proposed in [126]. The main concern about fully Bayesian approaches is that they are computationally costly and their convergence is difficult to assess [127, 128]. In order to overcome those problems, several variational Bayesian techniques has been proposed [129–132]. Indeed, the variational Bayesian inference consists of estimating a lower bound for the likelihood of observed data with a marginalization performed over unobserved variables and it constitutes a better alternative to MCMC. The other interesting aspect when performing clustering is to establish features selection. Indeed, it has been shown that not all the features have the same contribution in the clustering process such as in [62] where the generalized Dirichlet (GD) mixture has been used and factorized into a set of Beta distributions giving good results for proportional data clustering. Still, the Beta distribution support is defined in $[0\ 1]$, so it is not always an appropriate choice to represent positive data. The GID mixture that can be factorized into a set of Beta prime (inverted Beta) distributions whose support is $]0\ \infty[$, is still a more adequate choice to represent positive data [133]. We propose in this chapter to build a variational Bayesian framework of GID mixture with features selection, and investigate its modeling capabilities using synthetic and real data.
The rest of this chapter is organized as follows; in section 5.2 we introduce the statistical model that is based on an infinite GID mixture model with features selection.Then, we propose a variational Bayesian inference that estimates the parameters of the proposed model in section 5.3. In section 5.4 we investigate the performance of our algorithm on synthetic and real data and we conclude in section 5.5.

5.2 The Statistical Model

5.2.1 Finite Generalized Inverted Dirichlet Mixture Model

In this subsection we recall the definition of the finite GID mixture model. Let us consider a set $\mathscr{Y}$ of N D-dimensional vectors, such that $\mathscr{Y} = (\vec{Y}_1, \vec{Y}_2, \ldots, \vec{Y}_N)$. Let M denote the number of different components forming a flat mixture model at the system level. We assume that $\mathscr{Y}$ is controlled by a mixture of GID distributions and the vectors follow a common probability density function $p(\vec{Y}_n|\vec{\pi}, \vec{\alpha}, \vec{\beta})$ such that

$$p(\vec{Y}_n|\vec{\pi}, \vec{\alpha}, \vec{\beta}) = \sum_{j=1}^{M} \pi_j \prod_{d=1}^{D} \frac{\Gamma(\alpha_{jd} + \beta_{jd})}{\Gamma(\alpha_d)\Gamma(\beta_d)} \frac{Y_{nd}^{\alpha_d - 1}}{(1 + \sum_{l=1}^{d} Y_{nl})^{\gamma_d}} \tag{5.1}$$

The GID posterior probability can be factorized such that (See Appendix B.1)

$$p(j|\vec{X}_i,\vec{\pi},\vec{\alpha},\vec{\beta}) \propto \pi_j \prod_{l=1}^{D} p_{IBeta}(X_{il}|\alpha_{jl},\beta_{jl}) \tag{5.2}$$

where we have set $X_{i1} = Y_{i1}$ and $X_{il} = \frac{Y_{il}}{1+\sum_{k=1}^{l-1} Y_{ik}}$ for $l > 1$. $p_{IBeta}(X_{il}|\alpha_{jl},\beta_{jl})$ is an inverted Beta distribution with parameters α_{jl} and β_{jl}

$$p_{IBeta}(X_{il}|\alpha_j,\beta_j) = \frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} X_{il}^{\alpha_{jl}-1}(1+X_{il})^{-(\alpha_{jl}+\beta_{jl})} \tag{5.3}$$

The estimation of the parameters in Eq.5.1 is then equivalent to the estimation of the parameters of the following mixture

$$p(\mathscr{X}|\vec{\pi},\vec{\alpha},\vec{\beta})) = \prod_{n=1}^{N} \left(\sum_{j=1}^{M} \pi_j \prod_{l=1}^{D} p_{IBeta}(X_{il}|\alpha_{jl},\beta_{jl}) \right) \tag{5.4}$$

5.2.2 Infinite Generalized Inverted Dirichlet Mixture Model

We construct an infinite GID model using a Dirichlet Process with a stick-breaking representation [134] such that

$$p(X_i|\vec{\pi},\vec{\alpha},\vec{\beta})) = \sum_{j=1}^{\infty} \pi_j \prod_{l=1}^{D} p_{IBeta}(X_{il}|\alpha_{jl},\beta_{jl}) \tag{5.5}$$

with $\pi_j = \lambda_j \prod_{s=1}^{j-1}(1-\lambda_s)$ and $\lambda_j \sim Beta(1,\psi)$ where ψ is a real number. Let $\mathscr{Z} = \{\vec{Z}_1,\vec{Z}_2,...$ $...,\vec{Z}_N\}$ denote the missing group indicator, where $\vec{Z}_n = (z_{n1},z_{n2},...)$ is the label of $\vec{X}_n$, such that $z_{nj} \in \{0,1\}$, $\sum_{j=1}^{\infty} z_{nj} = 1$ and z_{nj} is equal to one if $\vec{X}_n$ belongs to class j and zero, otherwise. Then, the distribution of $\mathscr{X}$ given the class label Z is

$$p(\mathscr{X}|Z,\vec{\alpha},\vec{\beta}) = \prod_{i=1}^{N} \prod_{j=1}^{\infty} \Big(\prod_{l=1}^{D} p_{IBeta}(X_{il}|\alpha_{jl},\beta_{jl}) \Big)^{z_{ij}} \tag{5.6}$$

5.2.3 Infinite Generalized Inverted Dirichlet Mixture Model With Features Selection

Features selection is a fundamental aspect of machine learning especially with the growth of data dimensionality [135]. Indeed, when data are multidimensional some of the features could be noisy to the clustering process and deteriorate its performance. A feature is irrelevant when it does not have a discriminatory effect on the clusters. Mathematically speaking, if we consider M clusters,

and $\forall n,m \in \{1,...,M\}$, $KL\Big(p_{IBeta}(.|\alpha_{nl},\beta_{nl}), p_{IBeta}(.|\alpha_{ml},\beta_{ml})\Big) \simeq 0$ where KL is the Kullback-Leibler divergence, then $\{\alpha_{jl},\beta_{jl}\} = \{\sigma_l,\tau_l\}, \forall j$, such that $\{\sigma_l,\tau_l\}$ are the parameters of a common inverted Beta distribution independent from the class labels. The works in [119, 136] have considered a common univariate distribution to model the irrelevant features, while the works in [62] and [137] have respectively considered finite and infinite mixtures of overlapped distributions to be common to all clusters than a single univariate distribution. We adopt the infinite mixture model to represent the irrelevant features therefore we approximate each features as follows

$$p(X_{il}) \simeq \Big(iBeta(X_{il}|\alpha_{jl},\beta_{jl})\Big)^{\phi_{il}} \Big(\prod_{k=1}^{\infty}\big(iBeta(X_{il}|\sigma_{kl},\tau_{kl})\big)^{W_{ikl}}\Big)^{1-\phi_{il}} \tag{5.7}$$

where $W_{ikl} \in \{1,0\}$ such that W_{ikl} is equal to one if X_{il} is generated from the k^{th} components of the infinite Beta mixture representing the irrelevant features. $\phi_{il} \in \{0,1\}$ and is equal to one when l is a relevant feature and follows an inverted Beta distribution $iBeta(X_{il}|\alpha_{jl},\beta_{jl})$, otherwise the feature l follows an infinite mixture of inverted Beta distributions such that

$$p(X_{il}) = \sum_{k=1}^{\infty} \eta_k iBeta(X_{il}|\sigma_{kl},\tau_{kl}) \tag{5.8}$$

where η_k is the mixing probability, and $\{\sigma_{kl},\tau_{kl}\}$ are the parameters of the inverted Beta representing the k^{th} components of the irrelevant feature. The likelihood of $\mathscr{X}$ that follows an infinite GID mixture model with features selection can be written under the form

$$p(\mathscr{X}|\mathscr{Z},\mathscr{W},\vec{\phi},\vec{\alpha},\vec{\beta},\vec{\sigma},\vec{\tau}) = \prod_{i=1}^{N}\prod_{j=1}^{\infty}\left[\prod_{l=1}^{D}\Big(iBeta(X_{il}|\alpha_{jl},\beta_{jl})\Big)^{\phi_{il}}\Big(\prod_{k=1}^{\infty}\big(iBeta(X_{il}|\sigma_{kl},\tau_{kl})\big)^{W_{ikl}}\Big)^{1-\phi_{il}}\right]^{Z_{ij}} \tag{5.9}$$

5.2.4 Prior Distributions for the Infinite GID Mixture With Features Selection

The variational Bayesian approach needs the definition of priors for $\mathscr{Z},\mathscr{W},\vec{\phi},\vec{\alpha},\vec{\beta},\vec{\sigma}$ and $\vec{\tau}$. We consider priors that can give us tractable solutions for updating the variational factors. The priors of $\mathscr{Z}$ and $\mathscr{W}$ given the mixing coefficients $\vec{\pi}$ and $\vec{\eta}$ can be set as

$$p(\mathscr{Z}|\vec{\pi}) = \prod_{i=1}^{N}\prod_{j=1}^{\infty}\pi_j^{Z_{ij}} \qquad p(\mathscr{W}|\vec{\eta}) = \prod_{i=1}^{N}\prod_{k=1}^{\infty}\prod_{l=1}^{D}\eta_k^{W_{ikl}} \tag{5.10}$$

π_j and η_k can be written according the stick-breaking approach under the following form

$$\pi_j = \lambda_j \prod_{s=1}^{j-1}(1-\lambda_s) \qquad \eta_k = \gamma_k \prod_{s=1}^{k-1}(1-\gamma_s) \tag{5.11}$$

Using Eqs.5.10 and 5.11, we can have the following distribution for $\mathscr{Z}$ and $\mathscr{W}$

$$p(\mathscr{Z}|\vec{\lambda}) = \prod_{i=1}^{N}\prod_{j=1}^{\infty}\left[\lambda_j \prod_{s=1}^{j-1}(1-\lambda_s)\right]^{Z_{ij}} \qquad p(\mathscr{W}|\vec{\gamma}) = \prod_{i=1}^{N}\prod_{k=1}^{\infty}\prod_{l=1}^{D}\left[\gamma_k \prod_{s=1}^{k-1}(1-\gamma_s)\right]^{W_{ikl}} \tag{5.12}$$

The prior distributions of $\vec{\lambda}$ and $\vec{\gamma}$ are given by the stick-breaking approach as follows

$$p(\vec{\lambda}|\vec{\psi}) = \prod_{j=1}^{\infty} Beta(1,\psi_j) = \prod_{j=1}^{\infty}\psi_j(1-\lambda_j)^{\psi_j-1} \qquad p(\vec{\gamma}|\vec{\phi}) = \prod_{k=1}^{\infty} Beta(1,\phi_k) = \prod_{j=1}^{\infty}\psi_k(1-\gamma_k)^{\phi_k-1} \tag{5.13}$$

where $\vec{\psi}$ and $\vec{\phi}$ are the hyperparameters to which we can attribute conjugate Gamma priors [138] as follows

$$p(\vec{\psi}) = \mathscr{G}(\vec{\psi}|\vec{a},\vec{b}) = \prod_{j=1}^{\infty}\frac{b_j^{a_j}}{\Gamma(a_j)}\psi^{a_j-1}e^{-b_j\psi_j} \qquad p(\vec{(}\phi)) = \mathscr{G}(\vec{\phi}|\vec{c},\vec{d}) = \prod_{k=1}^{\infty}\frac{d_k^{c_k}}{\Gamma(c_k)}\varphi^{c_k-1}e^{-d_k\varphi_k} \tag{5.14}$$

The elements of hyperparameters vectors $\vec{a},\vec{b},\vec{c}$ and $\vec{d}$ are strictly positive. The prior distribution of the feature indicator variable $\vec{\phi}$ is defined as

$$p(\vec{\phi}|\vec{\varepsilon}) = \prod_{i=1}^{N}\prod_{l=1}^{D}\varepsilon_{l1}^{\phi_{il}}\varepsilon_{l2}^{1-\phi_{il}} \tag{5.15}$$

where $\vec{\varepsilon} = (\vec{\varepsilon}_1,..,\vec{\varepsilon}_D)$ is the features saliencies such that $\vec{\varepsilon}_l = (\varepsilon_{l1},\varepsilon_{l2})$ and $\varepsilon_{l1}+\varepsilon_{l2} = 1$. $\vec{\varepsilon}$ can be seen as a proportional data so it can be modeled by a Dirichlet distribution such that

$$p(\vec{\varepsilon}) = \prod_{l=1}^{D} Dir(\vec{\varepsilon}_l|\vec{\xi}) = \prod_{l=1}^{D}\frac{\Gamma(\xi_1+\xi_2)}{\Gamma(\xi_1)\Gamma(\xi_2)}\varepsilon_{l_1}^{\xi_1-1}\varepsilon_{l_2}^{\xi_2-1} \tag{5.16}$$

with the hyperparameters $\vec{\xi} = (\varepsilon_{l1},\varepsilon_{l2})$ that are strictly positive. As for the parameters $\vec{\alpha},\vec{\beta},\vec{\sigma}$ and $\vec{\tau}$ we consider the Gamma distribution as a prior distribution for them such that

$$p(\vec{\alpha}) \quad = \mathscr{G}(\vec{\alpha}|\vec{u},\vec{v}) = \prod_{j=1}^{\infty}\prod_{l=1}^{D}\frac{v_{jl}^{u_{jl}}}{\Gamma(u_{jl})}\alpha_{jl}^{u_{jl}-1}e^{-v_{jl}\alpha_{jl}}$$

$$
\begin{aligned}
p(\vec{\beta}) &= \mathscr{G}(\vec{\beta}|\vec{p},\vec{q}) = \prod_{j=1}^{\infty}\prod_{l=1}^{D}\frac{q_{jl}^{p_{jl}}}{\Gamma(p_{jl})}\beta_{jl}^{p_{jl}-1}e^{-q_{jl}\beta_{jl}} \\
p(\vec{\sigma}) &= \mathscr{G}(\vec{\sigma}|\vec{g},\vec{h}) = \prod_{k=1}^{\infty}\prod_{l=1}^{D}\frac{h_{kl}^{g_{kl}}}{\Gamma(g_{kl})}\sigma_{kl}^{g_{kl}-1}e^{-h_{kl}\sigma_{kl}} \\
p(\vec{\tau}) &= \mathscr{G}(\vec{\sigma}|\vec{s},\vec{t}) = \prod_{k=1}^{\infty}\prod_{l=1}^{D}\frac{t_{kl}^{s_{kl}}}{\Gamma(s_{kl})}\tau_{kl}^{s_{kl}-1}e^{-t_{kl}\tau_{kl}}
\end{aligned} \tag{5.17}
$$

where all the elements of hyperparameters vectors $\vec{u},\vec{v},\vec{q},\vec{g},\vec{h},\vec{t}$, and $\vec{s}$ are strictly positive. To summarize, the set of parameters of unknown variables is $\Theta = \{\mathscr{Z},\mathscr{W},\vec{\alpha},\vec{\beta},\vec{\sigma},\vec{\tau},\vec{\lambda},\vec{\psi},\vec{\gamma},\vec{\varphi},\vec{\varepsilon}\}$, and the joint distribution of all the random variables is given by

$$
\begin{aligned}
& p(\mathscr{X},\Theta) = p(\mathscr{X}|\mathscr{Z},\mathscr{W},\vec{\alpha},\vec{\beta},\vec{\sigma},\vec{\tau},\vec{\lambda},\vec{\psi},\vec{\gamma},\vec{\varphi},\vec{\varepsilon}) \\
= \;& p(\mathscr{X}|\mathscr{Z},\mathscr{W},\Phi,\vec{\alpha},\vec{\beta},\vec{\sigma},\vec{\tau})p(\mathscr{Z}|\vec{\lambda})p(\vec{\lambda}|\vec{\Psi})p(\vec{\Psi})p(\mathscr{W}|\vec{\gamma})p(\vec{\gamma}|\vec{\varphi})p(\vec{\varphi})p(\vec{\phi}|\vec{\varepsilon})p(\vec{\varepsilon})p(\vec{\alpha})p(\vec{\beta})p(\vec{\sigma})p(\vec{\tau}) \\
= \;& \prod_{i=1}^{N}\prod_{j=1}^{\infty}\left\{\prod_{l=1}^{D}\left[\frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})}\frac{X_{il}^{\alpha_{jl}-1}}{(1+X_{il})^{(\alpha_{jl}+\beta_{jl})}}\right]^{\phi_{il}}\left[\prod_{k=1}^{\infty}\left(\frac{\Gamma(\sigma_{kd}+\tau_{kd})}{\Gamma(\sigma_{kl})\Gamma(\tau_{kl})}\frac{X_{il}^{\sigma_{kl}-1}}{(1+X_{il})^{(\sigma_{kl}+\tau_{kl})}}\right)^{w_{ikl}}\right]^{1-\phi_{il}}\right\}^{Z_{ij}} \\
\times \;& \prod_{i=1}^{N}\prod_{j=1}^{\infty}\left[\lambda_j\prod_{s=1}^{j-1}(1-\lambda_s)\right]^{Z_{ij}} \times \prod_{j=1}^{\infty}\psi_j(1-\lambda_j)^{\psi_j-1} \times \prod_{j=1}^{\infty}\frac{b_j^{a_j}}{\Gamma(a_j)}\psi^{a_j-1}e^{-b_j\psi_j} \\
\times \;& \prod_{i=1}^{N}\prod_{k=1}^{\infty}\prod_{l=1}^{D}\left[\gamma_k\prod_{s=1}^{k-1}(1-\gamma_s)\right]^{W_{ikl}} \times \prod_{k=1}^{\infty}\varphi_k(1-\gamma_k)^{\varphi_k-1} \times \prod_{k=1}^{\infty}\frac{d_k^{c_k}}{\Gamma(c_k)}\varphi^{c_k-1}e^{-d_k\varphi_k} \\
\times \;& \prod_{i=1}^{N}\prod_{l=1}^{D}\varepsilon_{l_1}^{\phi_{li}}\varepsilon_{l_2}^{1-\phi_{li}} \times \prod_{l=1}^{D}\frac{\Gamma(\xi_1+\xi_2)}{\Gamma(\xi_1)\Gamma(\xi_2)}\varepsilon_{l_1}^{\xi_1-1}\varepsilon_{l_2}^{\xi_2-1} \times \prod_{j=1}^{\infty}\prod_{l=1}^{D}\left[\frac{v_{jl}^{u_{jl}}}{\Gamma(u_{jl})}\alpha_{jl}^{u_{jl}-1}e^{-v_{jl}\alpha_{jl}}\frac{q_{jl}^{p_{jl}}}{\Gamma(p_{jl})}\beta_{jl}^{p_{jl}-1}e^{-q_{jl}\beta_{jl}}\right] \\
\times \;& \prod_{k=1}^{\infty}\prod_{l=1}^{D}\left[\frac{h_{kl}^{g_{kl}}}{\Gamma(g_{kl})}\sigma_{kl}^{g_{kl}-1}e^{-h_{kl}\sigma_{kl}}\frac{t_{kl}^{s_{kl}}}{\Gamma(s_{kl})}\tau_{kl}^{s_{kl}-1}e^{-t_{kl}\tau_{kl}}\right]
\end{aligned} \tag{5.18}
$$

Fig.5.1 illustrates the dependencies between all the variables via a graphical model.

5.3 Variational Inference

In this section we develop a variational inference framework for the parameters estimation of the infinite GID mixture with features selection. The aim of variational inference is to determine a distribution $Q(\Theta)$ that approximates the true posterior distribution $p(\Theta|\mathscr{X})$. The Kullback-Leibler (KL) divergence between $Q(\Theta)$ and $p(\Theta|\mathscr{X})$ is given by

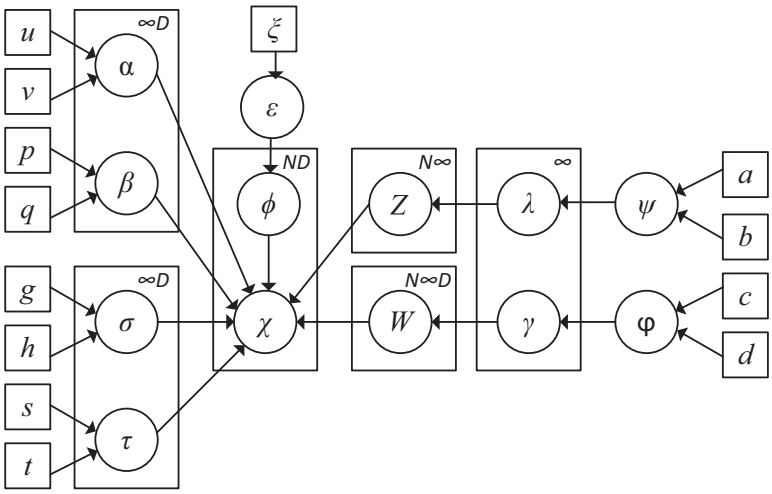


Figure 5.1: Graphical model representation of the infinite GID mixture model with features selection. The random variables are in circles, and the model parameters in squares. The number mentioned in the right upper corner of the plates indicates the number of repetition of the contained random variables. The arcs describe the conditional dependencies between variables.

$$KL(Q||P) \quad = \quad -\int Q(\Theta)\ln\left(\frac{p(\Theta|\mathscr{X})}{Q(\Theta)}\right)d\Theta = \ln p(\mathscr{X}) - \mathscr{L}(Q) \tag{5.19}$$

with

$$\mathscr{L}(Q) = \int Q(\Theta)\ln\left(\frac{p(\Theta,\mathscr{X})}{Q(\Theta)}\right)d\Theta \tag{5.20}$$

since $KL(Q||P) \geq 0$ and equal to zero when $Q(\Theta) = p(\Theta|\mathscr{X})$, we can conclude from Eq.5.19 that $\mathscr{L}(Q) \leq \ln p(\mathscr{X})$, which means that $\mathscr{L}(Q)$ can be considered as a lower bound to $\ln p(\mathscr{X})$. However in practice, the true posterior distribution is computationally intractable and cannot be directly used for variational inference. Thus, we have to consider a restricted family of $Q(\Theta)$ that can be computed [139]. Indeed we factorize $Q(\Theta)$ into disjoint tractable distributions such that $Q(\Theta) = \prod_i Q_i(\Theta_i)$, which is known as the mean field theory [140]. The maximization of $\mathscr{L}(Q)$ is established through a variational optimization with respect to each of the factor distributions $Q_i(\Theta_i)$. If we consider a specific Q_s,the variational approximation consists of keeping $\{\Theta_i\}_{i\neq s}$ fixed and maximize $\mathscr{L}(Q)$ with respect to all possible forms for the distribution $Q_s(\Theta_s)$. The optimal solution for $Q_s(\Theta_s)$ is given by [23]

$$\ln Q_s(\Theta_s) = \langle \ln p(\mathscr{X},\Theta)\rangle_{j\neq s} + const \tag{5.21}$$

where $\langle . \rangle_{j\neq s}$ denotes and expectation with respect to all the distributions $Q_i(\Theta_i)$ except for

$i = s$, such that

$$\langle \ln p(\mathscr{X}, \Theta) \rangle_{j \neq s} = \int \ln p(\mathscr{X}, \Theta) \prod_{i \neq s} Q_i(\Theta_i) d\Theta_i \tag{5.22}$$

and the normalized solution is given by

$$Q_s(\Theta_s) = \frac{\exp \langle \ln p(\mathscr{X}, \Theta) \rangle_{j \neq s}}{\exp \int \langle \ln p(\mathscr{X}, \Theta) \rangle_{j \neq s} d\Theta} \tag{5.23}$$

As proposed by [23], we initialize $Q_s(\Theta_s)$ appropriately, and then we cycle through the factors and replace each in turn with a revised estimate given by Eq.5.21 and evaluated using the current estimates for all the other factors. A convergence is guaranteed because the bound is convex with respect to each of the factors $Q_i(\Theta_i)$ [23, 141]. In order to exploit the bound, we should consider the truncation of the stick-breaking representation as proposed by [138] to establish a variational inference for DP mixtures. The truncation of the stick-breaking representation was also proposed by [142] in the context of sampling-based inference for an approximation to the DP mixture model. This is done by fixing a value M, such that $\lambda_M = 1$, and $\pi_j = 0$ when $j > M$, which leads to $\sum_{j=1}^{M} \pi_j = 1$. We apply a similar truncation to the infinite inverted Beta mixture representing the irrelevant features by fixing a value K, such that $\gamma_K = 1$, and $\eta_k = 0$ when $k > K$, which leads to $\sum_{k=1}^{K} \eta_k = 1$. Note that the model still a full Dirichlet process and is not truncated, only the variational distribution is truncated. The truncation levels M and K are variational parameters that can be freely set, and they are not a part of the prior model specification [23]. Thus M and K can be optimized during the learning process. The factorization of $Q(\Theta)$ can be written under the following form

$$\begin{aligned} Q(\Theta) &= \left[\prod_{i=1}^{N}\prod_{j=1}^{M} Q(Z_{ij})\right]\left[\prod_{j=1}^{M} Q(\lambda_j)Q(\Psi_j)\right]\left[\prod_{i=1}^{N}\prod_{k=1}^{K}\prod_{l=1}^{D} Q(W_{ikl})\right]\left[\prod_{k=1}^{K} Q(\gamma_k)Q(\varphi_k)\right] \\ &\times \left[\prod_{i=1}^{N}\prod_{l=1}^{D} Q(\phi_{il})\right]\left[\prod_{l=1}^{D} Q(\vec{\varepsilon}_l)\right]\left[\prod_{j=1}^{M}\prod_{l=1}^{D} Q(\alpha_{jl})Q(\beta_{jl})\right]\left[\prod_{k=1}^{K}\prod_{l=1}^{D} Q(\sigma_{kl})Q(\tau_{kl})\right] \end{aligned} \tag{5.24}$$

The optimal solutions are the following, (details in C.1)

$$Q(Z) = \prod_{i=1}^{N}\prod_{j=1}^{M} r_{ij}^{Z_{ij}} \qquad Q(\vec{\lambda}) = \prod_{j=1}^{M} Beta(\lambda_j|\theta_j, v_j)$$

$$Q(\vec{\psi}) = \prod_{j=1}^{M} \mathscr{G}(\psi_j|a_j^*, b_j^*) \qquad Q(\vec{\alpha}) = \prod_{j=1}^{M}\prod_{l=1}^{D} \mathscr{G}(\alpha_{jl}|u_{jl}^*, v_{jl}^*)$$

$$
\begin{aligned}
Q(\vec{\beta}) &= \prod_{j=1}^{M}\prod_{l=1}^{D}\mathscr{G}(\beta_{jl}|p^*_{jl},q^*_{jl}) & Q(W) &= \prod_{i=1}^{N}\prod_{k=1}^{K}\prod_{l=1}^{D} m_{ikl}^{W_{ikl}} \\
Q(\vec{\gamma}) &= \prod_{k=1}^{K} Beta(\gamma_k|\rho_k,\varpi_k) & Q(\vec{\varphi}) &= \prod_{k=1}^{K}\mathscr{G}(\varphi_k|c^*_k,d^*_k) \\
Q(\vec{\phi}) &= \prod_{i=1}^{N}\prod_{l=1}^{D} f_{il}^{\phi_{il}}(1-f_{il})^{(1-\phi_{il})} & Q(\vec{\varepsilon}) &= \prod_{l=1}^{D} Dir(\varepsilon_l|\vec{\xi}^*) \\
Q(\vec{\sigma}) &= \prod_{k=1}^{K}\prod_{l=1}^{D}\mathscr{G}(\sigma_{kl}|g^*_{kl},h^*_{kl}) & Q(\vec{\tau}) &= \prod_{k=1}^{K}\prod_{l=1}^{D}\mathscr{G}(\tau_{kl}|s^*_{kl},t^*_{kl})
\end{aligned}
\tag{5.25}
$$

with

$$
r_{ij} = \frac{\tilde{r}_{ij}}{\sum_{j=1}^{M}\tilde{r}_{ij}}, \quad m_{ikl} = \frac{\tilde{m}_{ikl}}{\sum_{k=1}^{K}\tilde{m}_{ikl}}, \quad f_{il} = \frac{f_{il}^{(\phi_{il})}}{f_{il}^{(\phi_{il})}+f_{il}^{(1-\phi_{il})}} \tag{5.26}
$$

$$
\begin{aligned}
\tilde{r}_{ij} &= \exp\left\{\sum_{l=1}^{D}\left(\langle\phi_{il}\rangle\left[\tilde{\mathscr{R}}_{jl}+(\bar{\alpha}_{jl}-1)\ln X_{il}-(\bar{\alpha}_{jl}+\bar{\beta}_{jl})\ln(1+X_{il})\right]\right.\right. \\
&+ \langle 1-\phi_{il}\rangle\sum_{k=1}^{K}\langle w_{ikl}\rangle\left.\left[\tilde{\mathscr{F}}_{kl}+(\bar{\sigma}_{kl}-1)\ln X_{il}-(\bar{\sigma}_{kl}+\bar{\tau}_{kl})\ln(1+X_{il})\right]\right) \\
&+ \langle\ln\lambda_j\rangle+\sum_{s=1}^{j-1}\langle\ln(1-\lambda_s)\rangle\Bigg\}
\end{aligned}
\tag{5.27}
$$

$$
\tilde{m}_{ikl} = \exp\left\{\langle 1-\phi_{il}\rangle\left(\mathscr{F}_{kl}+(\bar{\sigma}_{kl}-1)\ln X_{il}-(\bar{\sigma}_{kl}+\bar{\tau}_{kl})\ln(1+X_{il})\right)+\langle\ln\gamma_k\rangle+\sum_{s=1}^{k-1}\langle\ln(1-\gamma_s)\rangle\right\} \tag{5.28}
$$

$$
f_{il}^{(\phi_{il})} = \exp\left\{\langle\ln\varepsilon_{l1}\rangle+\sum_{j=1}^{M}\langle Z_{ij}\rangle\left[\mathscr{R}_{jl}+(\bar{\alpha}_{jl}-1)\ln X_{il}-(\bar{\alpha}_{jl}+\bar{\beta}_{jl})\ln(1+X_{il})\right]\right\} \tag{5.29}
$$

$$f_{il}^{(1-\phi_{il})} = \exp\left\{ \langle \ln \varepsilon_{l2} \rangle + \left\{ \sum_{k=1}^{K} \langle W_{ikl} \rangle \left[\mathscr{F}_{kl} + (\bar{\sigma}_{kl} - 1) \ln X_{il} - (\bar{\sigma}_{kl} + \bar{\tau}_{kl}) \ln (1 + X_{il}) \right] \right\} \right\} \quad (5.30)$$

$$\begin{aligned} \tilde{\mathscr{R}} &= \ln \frac{\Gamma(\bar{\alpha}+\bar{\beta})}{\Gamma(\bar{\alpha})\Gamma(\bar{\beta})} + \bar{\alpha}[\psi(\bar{\alpha}+\bar{\beta}) - \psi(\bar{\alpha})](\langle \ln \alpha \rangle - \ln \bar{\alpha}) + \bar{\beta}[\psi(\bar{\beta}+\bar{\alpha}) - \psi(\bar{\beta})](\langle \ln \beta \rangle - \ln \bar{\beta}) \\ &+ 0.5\bar{\alpha}^2[\psi'(\bar{\alpha}+\bar{\beta}) - \psi'(\bar{\alpha})]\langle (\ln \alpha - \ln \bar{\alpha})^2 \rangle + 0.5\bar{\beta}^2[\psi'(\bar{\beta}+\bar{\alpha}) - \psi'(\bar{\beta})]\langle (\ln \beta - \ln \bar{\beta})^2 \rangle \\ &+ \bar{\alpha}\bar{\beta}\psi'(\bar{\alpha}+\bar{\beta})(\langle \ln \alpha \rangle - \ln \bar{\alpha})(\langle \ln \beta \rangle - \ln \bar{\beta}) \end{aligned} \quad (5.31)$$

$$\begin{aligned} \tilde{\mathscr{F}} &= \ln \frac{\Gamma(\bar{\sigma}+\bar{\tau})}{\Gamma(\bar{\sigma})\Gamma(\bar{\tau})} + \bar{\sigma}[\psi(\bar{\sigma}+\bar{\tau}) - \psi(\bar{\sigma})](\langle \ln \sigma \rangle - \ln \bar{\sigma}) + \bar{\tau}[\psi(\bar{\tau}+\bar{\sigma}) - \psi(\bar{\tau})](\langle \ln \tau \rangle - \ln \bar{\tau}) \\ &+ 0.5\bar{\sigma}^2[\psi'(\bar{\sigma}+\bar{\tau}) - \psi'(\bar{\sigma})]\langle (\ln \sigma - \ln \bar{\sigma})^2 \rangle + 0.5\bar{\tau}^2[\psi'(\bar{\tau}+\bar{\sigma}) - \psi'(\bar{\tau})]\langle (\ln \tau - \ln \bar{\tau})^2 \rangle \\ &+ \bar{\sigma}\bar{\tau}\psi'(\bar{\sigma}+\bar{\tau})(\langle \ln \sigma \rangle - \ln \bar{\sigma})(\langle \ln \tau \rangle - \ln \bar{\tau}) \end{aligned} \quad (5.32)$$

where $\psi(.)$ is the digamma function that is defined as $\psi(\alpha) = \frac{d \ln \Gamma(\alpha)}{d\alpha}$.

$$\theta_j = 1 + \sum_{i=1}^{N} \langle Z_{ij} \rangle \ , \quad \vartheta_j = \langle \Psi_j \rangle + \sum_{1}^{N} \sum_{s=j+1}^{M} \langle Z_{is} \rangle \quad (5.33)$$

$$a_j^* = a_j + 1 \ , \quad b_j^* = b_j - \langle \ln (1 - \lambda_j) \rangle \quad (5.34)$$

$$\rho_k = 1 + \sum_{i=1}^{N} \sum_{l=1}^{D} \langle W_{ikl} \rangle \ , \quad \varpi_k = \langle \varphi_k \rangle + \sum_{i=1}^{N} \sum_{s=k+1}^{K} \sum_{l=1}^{D} \langle W_{isl} \rangle \quad (5.35)$$

$$c_k^* = c_k + 1 \ , \quad d_k^* = d_k - \langle \ln (1 - \gamma_k) \rangle \quad (5.36)$$

$$\xi_1^* = \xi_1 + \sum_{i=1}^{N} \langle \phi_{il} \rangle \ , \qquad \xi_2^* = \xi_2 + \sum_{i=1}^{N} \langle 1 - \phi_{il} \rangle \quad (5.37)$$

$$u_{jl}^* = u_{jl} + \sum_{i=1}^{N} \langle Z_{ij} \rangle \langle \phi_{il} \rangle \left[\psi(\bar{\alpha}_{jl} + \bar{\beta}_{jl}) - \psi(\bar{\alpha}_{jl}) + \bar{\beta}_{jl}\psi'(\bar{\alpha}_{jl} + \bar{\beta}_{jl})) (\langle \ln \beta_{jl} \rangle - \ln \bar{\beta}_{jl}) \right] \bar{\alpha}_{jl} \quad (5.38)$$

$$v^*_{jl} = v_{jl} - \sum_{i=1}^{N} \langle Z_{ij} \rangle \langle \phi_{il} \rangle \ln \frac{X_{il}}{1+X_{il}} \tag{5.39}$$

$$p^*_{jl} = p_{jl} + \sum_{i=1}^{N} \langle Z_{ij} \rangle \langle \phi_{il} \rangle \left[\psi(\bar{\beta}_{jl} + \bar{\alpha}_{jl}) - \psi(\bar{\beta}_{jl}) + \bar{\alpha}_{jl} \psi'(\bar{\beta}_{jl} + \bar{\alpha}_{jl})) (\langle \ln \alpha_{jl} \rangle - \ln \bar{\alpha}_{jl}) \right] \bar{\beta}_{jl} \tag{5.40}$$

$$q^*_{jl} = q_{jl} - \sum_{i=1}^{N} \langle Z_{ij} \rangle \langle \phi_{il} \rangle \ln \frac{1}{1+X_{il}} \tag{5.41}$$

$$g^*_{kl} = g_{kl} + \sum_{i=1}^{N} \langle 1-\phi_{il} \rangle \langle W_{ikl} \rangle \left[\psi(\bar{\sigma}_{kl} + \bar{\tau}_{kl}) - \psi(\bar{\sigma}_{kl}) + \bar{\tau}_{kl} \psi'(\bar{\sigma}_{kl} + \bar{\tau}_{kl})) (\langle \ln \tau_{kl} \rangle - \ln \bar{\tau}_{kl}) \right] \bar{\sigma}_{kl} \tag{5.42}$$

$$h^*_{kl} = h_{kl} - \sum_{i=1}^{N} \langle 1-\phi_{il} \rangle \langle W_{jkl} \rangle \ln \frac{X_{il}}{1+X_{il}} \tag{5.43}$$

$$s^*_{kl} = s_{kl} + \sum_{i=1}^{N} \langle 1-\phi_{il} \rangle \langle W_{ikl} \rangle \left[\psi(\bar{\tau}_{kl} + \bar{\sigma}_{kl}) - \psi(\bar{\tau}_{kl}) + \bar{\sigma}_{kl} \psi'(\bar{\tau}_{kl} + \bar{\sigma}_{kl})) (\langle \ln \sigma_{kl} \rangle - \ln \bar{\sigma}_{kl}) \right] \bar{\tau}_{kl} \tag{5.44}$$

$$t^*_{kl} = t_{kl} - \sum_{i=1}^{N} \langle 1-\phi_{il} \rangle \langle W_{jkl} \rangle \ln \frac{1}{1+X_{il}} \tag{5.45}$$

The expected values are given by

$$\bar{\alpha}_{jl} = \frac{u^*_{jl}}{v^*_{jl}}, \quad \bar{\beta}_{jl} = \frac{p^*_{jl}}{q^*_{jl}}, \quad \bar{\sigma}_{kl} = \frac{g^*_{kl}}{h^*_{jl}}, \quad \bar{\tau}_{kl} = \frac{s^*_{kl}}{t^*_{kl}} \tag{5.46}$$

$$\langle \psi_j \rangle = \frac{a^*_j}{b^*_j}, \quad \langle \varphi_k \rangle = \frac{c^*_k}{d^*_k}, \quad \langle Z_{ij} \rangle = r_{ij}, \quad \langle W_{ikl} \rangle = m_{ikl} \tag{5.47}$$

$$\langle \phi_{il} \rangle = f_{il}, \quad \langle 1-\phi_{il} \rangle = 1-f_{il}, \quad \langle \ln \alpha \rangle = \psi(u^*) - \ln v^*, \quad \langle \ln \beta \rangle = \psi(p^*) - \ln q^* \tag{5.48}$$

$$\langle \ln \sigma \rangle = \psi(g^*) - \ln h^*, \quad \langle \ln \tau \rangle = \psi(s^*) - \ln t^* \tag{5.49}$$

$$\langle \ln \lambda_j \rangle = \psi(\theta) - \psi(\theta + \vartheta), \quad \langle \ln (1 - \lambda_j) \rangle = \psi(\vartheta) - \psi(\theta + \vartheta) \tag{5.50}$$

$$\langle \ln \gamma_k \rangle = \psi(\rho) - \psi(\rho + \varpi), \quad \langle \ln (1 - \gamma_k) \rangle = \psi(\varpi) - \psi(\rho + \varpi), \tag{5.51}$$

$$\langle \ln \varepsilon_{l_1} \rangle = \psi(\xi_1^*) - \psi(\xi_1^* + \xi_2^*), \quad \langle \ln \varepsilon_{l_2} \rangle = \psi(\xi_2^*) - \psi(\xi_1^* + \xi_2^*) \tag{5.52}$$

$$\langle (\ln \alpha - \ln \bar{\alpha})^2 \rangle = [\psi(u^*) - \ln u^*]^2 + \psi'(u^*) \tag{5.53}$$

$$\langle (\ln \beta - \ln \bar{\beta})^2 \rangle = [\psi(p^*) - \ln p^*]^2 + \psi'(p^*) \tag{5.54}$$

$$\langle (\ln \sigma - \ln \bar{\sigma})^2 \rangle = [\psi(g^*) - \ln g^*]^2 + \psi'(g^*) \tag{5.55}$$

$$\langle (\ln \tau - \ln \bar{\tau})^2 \rangle = [\psi(s^*) - \ln s^*]^2 + \psi'(s^*) \tag{5.56}$$

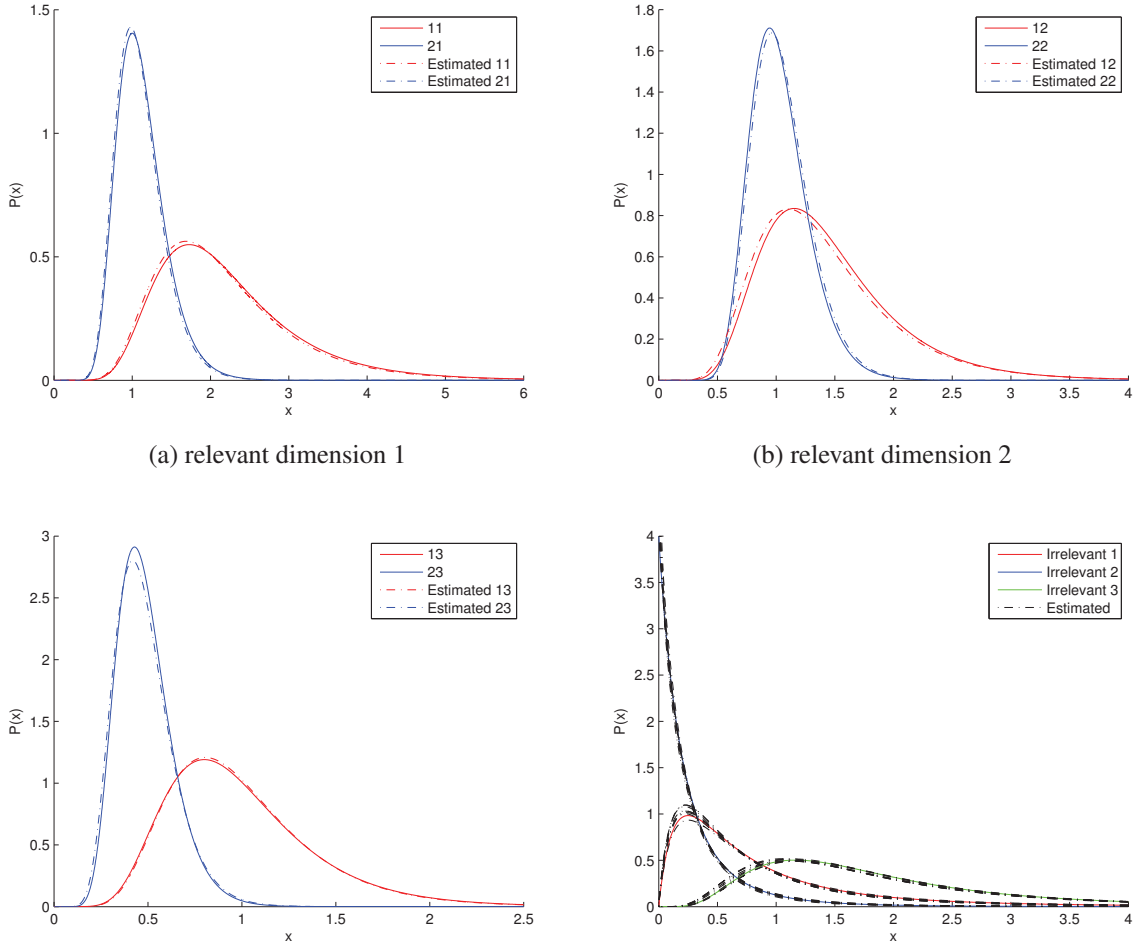
In order to initialize M and K, we adopt the same strategy adopted by [62, 143, 144] which consists of over-initializing the number of clusters M and K to be much larger than the true model, and then infer the structure of the mixture by discarding the components whose mixing probabilities are close to zero. The learning process is summarized in Algorithm.2.

Algorithm 2 Infinite GID learning using variational inference with features selection

1: Choose truncations levels M and K
2: Initialize the values for hyperparameters $u_{ji}, v_{ji}, p_{ji}, q_{ji}, g_{kl}, h_{kl}, s_{kl}, t_{kl}, a_j, b_j, c_k, d_k, \xi_1$, and ξ_2.
3: Initialize the values of r_{ij} and m_{ikl} using the K-means algorithm.
4: Estimate the expected value using Eqs.5.46-5.56.
5: Update the variational solutions of each factor using Eqs.5.25.
6: If converge criteria is reached go to 7. else go to 4.
7: Compute the expected value of λ_j as $\langle \lambda_j \rangle = \frac{\theta_j}{\theta_j + \vartheta_j}$ and substitute it into Eq.5.11 in order to compute the estimated mixing probabilities π_j
8: Compute the expected value of γ_k as $\langle \gamma_k \rangle = \frac{\rho_k}{\rho_k + \varpi_k}$ and substitute it into Eq.5.11 in order to compute the estimated mixing probabilities η_k
9: Calculate the expected values of the features saliencies $\langle \varepsilon_l \rangle = \frac{\xi_1^*}{\xi_1^* + \xi_2^*}$
10: Set the optimal number of components M and K by discarding the components whose mixing probabilities are close to 0.

5.4 Experimental Results

In this section, an evaluation of our proposed algorithm is performed using synthetic and real computer vision datasets. The values of hyper parameters are empirically initialized basing on several runs. u, p, h, t, g and s are set to 1, and v, q are set to 0.05. The hyper parameters a, b, c and d are set to 1, and ξ_1, ξ_2 are set to 0.01. The choice of these parameters have supported our experiments. The hyper parameters can be randomly set, but a good initialization is still crucial because the Kullback-Leibler divergence to the true posterior may contain many local minima [145]. Several methods have been proposed to estimate an initial value for the hyperparameters in [145, 146] that are beyond the scope of this work.



(a) relevant dimension 1

(b) relevant dimension 2

(c) relevant dimension 3

(d) irrelevant dimensions 4-5-6-7-8-9-10-11

Figure 5.2: Different generated inverted Beta distributions labeled in ji representing the 11 dimensions of the model in Table 5.1.

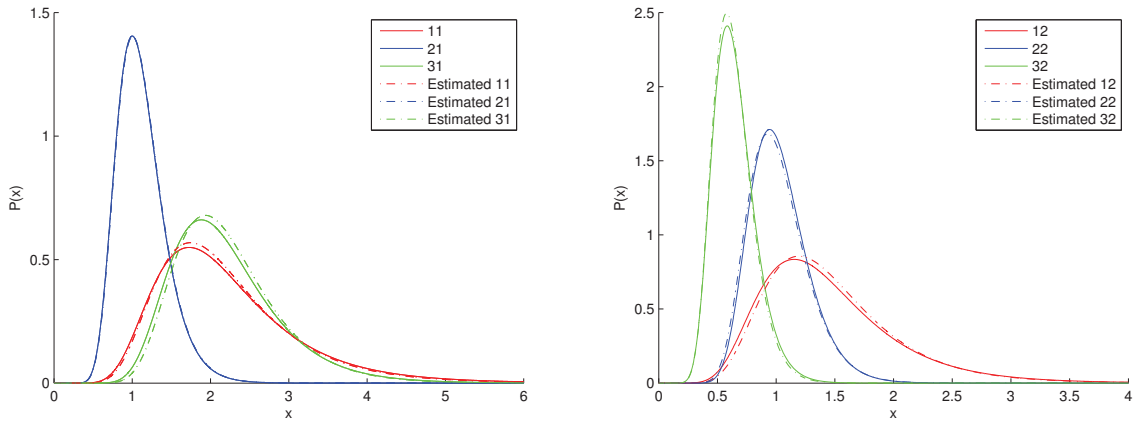
Table 5.1: Real and estimated parameters of relevant components in the case of a 11-dimensional dataset generated from 2-components mixture model.

j	l	n_j	π_j	α_{ji}	β_{jl}	$\hat{\pi}_j$	$\hat{\alpha}_{jl}$	$\hat{\beta}_{jl}$
1	1	600	0.5	20	10	0.5077	19.76	10.11
	2			16	12		14.53	11.24
	3			13	14		13.48	14.50
2	1	600	0.5	28	26	0.4923	27.55	26.09
	2			35	35		35.60	34.97
	3			16	34		14.29	30.71

5.4.1 Synthetic Data

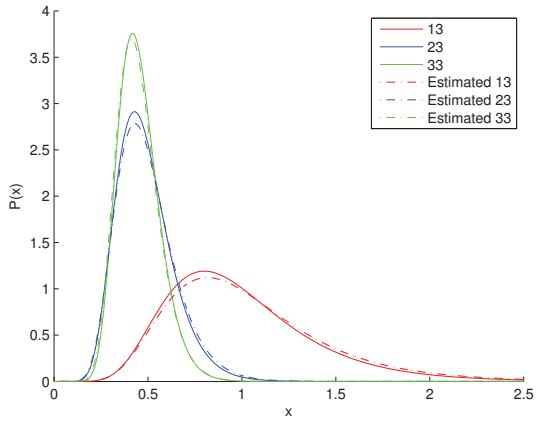
At a first stage we evaluate the performance of our algorithm using synthetic data. We propose to use 11-dimensional datasets, whose first three dimensions are relevant and the remaining eight dimensions are irrelevant. Depending on the dataset, the relevant features are generated using at least 2 distinguishable inverted beta distributions, while the irrelevant features are generated by a mixture whose inverted beta distributions are all overlapping ($IBeta(2,3), IBeta(1,4), IBeta(8,5)$). For all our experiments on synthetic data we use an initial value of $M = 15$ and $K = 10$. The first dataset is constituted of two components, composed of 600 samples each. Fig.5.2 represents the real and estimated histograms, where Fig.5.2a, Fig.5.2b, and Fig.5.2c represent the different inverted beta distributions for the first three relevant dimensions while Fig.5.2d represents the distributions of the 8 irrelevant features. The continuous lines represent the real histograms while the dashed ones represent the estimated histograms. The obtained results are reported in Table.5.1 where we show the real and estimated parameters. For the easiness of presentation we do not illustrate the values of the estimated parameters of the irrelevant features, but as it can be shown in Fig.5.2d, the estimated histograms reflect a good estimation. The algorithm was capable to find the correct number of classes $M = 2$ and the correct number of distributions forming the irrelevant features which is $K = 3$. The feature saliency was also correctly estimated, as $\varepsilon_{l1} = 1$ for the first three features and zero for the rest. The maximum detected error is 10.69%, which remains a good estimate with the presence of irrelevant features and the fact that we have considered slightly overlapping relevant features.

The second dataset is constructed by adding a third component to the first dataset, as shown in Fig.5.3, where we plot the estimated and real histograms of the different dimensions, as for the first dataset in Fig.5.2. The variational inference algorithm was capable to find the correct number of classes $M = 3$ and $K = 3$. We report on the estimated parameters in Table.5.2, where the maximum detected relative error is equal to 12.13%. Still the estimated histograms are almost indistinguishable from the real ones, which means that the estimated parameters leads to the same

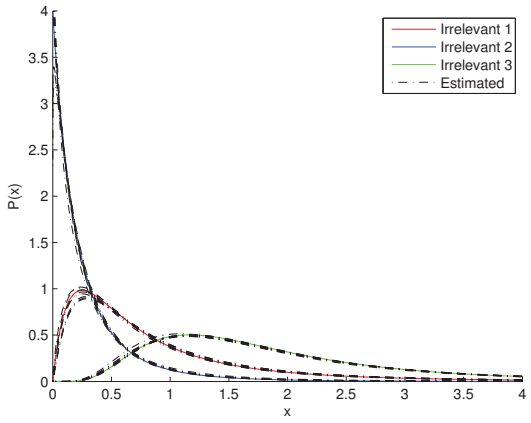


(a) relevant dimension 1

(b) relevant dimension 2



(c) relevant dimension 3



(d) irrelevant dimensions 4-5-6-7-8-9-10-11

Figure 5.3: Different generated inverted Beta distributions labeled in ji representing the 11 dimensions of the model in Table 5.2.

performance of clustering using the real parameters.

The third dataset that we have used in constructed by adding a fourth component to the third dataset as illustrated in Fig.5.4. The algorithm was capable to find the correct number of relevant components $M = 4$, and the number of components composing the mixture of irrelevant features $K = 3$. We report on the estimated parameters of the relevant features in Table.5.3 where the maximum detected relative error is equal to 13.70%. It is noteworthy that the synthetic dataset that we are considering are difficult to estimate even with the absence of irrelevant features as the relevant features are still overlapping. The obtained results show that our algorithm performs well for synthetic data, as it was capable to optimally select M and K , and generate estimated histograms that are almost indistinguishable from the real ones, leading to the same clustering

Table 5.2: Real and estimated parameters of relevant components in the case of a 11-dimensional dataset generated from 3-components mixture model.

j	l	n_j	π_j	α_{ji}	β_{jl}	$\hat{\pi}_j$	$\hat{\alpha_{jl}}$	$\hat{\beta_{jl}}$
1	1	600	0.4	20	10	0.3988	21.61	10.81
	2			16	12		17.94	13.19
	3			13	14		12.34	12.88
2	1	600	0.4	28	26	0.4084	28.22	26.10
	2			35	35		32.68	33.11
	3			16	34		14.92	31.34
3	1	300	0.2	33	16	0.1928	36.89	17.56
	2			22	35		22.99	37.02
	3			24	54		22.64	51.48

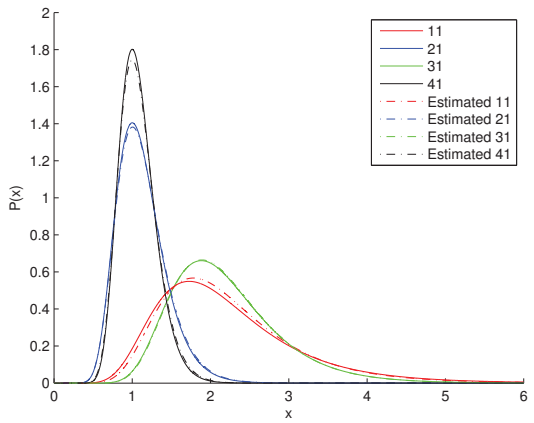
Table 5.3: Real and estimated parameters of relevant components in the case of a 11-dimensional dataset generated from 4-components mixture model.

j	l	n_j	π_j	α_{ji}	β_{jl}	$\hat{\pi}_j$	$\hat{\alpha_{jl}}$	$\hat{\beta_{jl}}$
1	1	600	0.30	20	10	0.3053	22.53	11.09
	2			16	12		14.78	11.00
	3			13	14		12.29	13.45
2	1	600	0.30	28	26	0.2991	27.45	25.34
	2			35	35		38.99	38.92
	3			16	34		15.02	32.16
3	1	300	0.15	33	16	0.1447	33.66	16.28
	2			22	35		20.44	33.51
	3			24	54		25.20	57.62
4	1	500	0.25	44	42	0.2509	41.42	39.44
	2			50	23		43.15	20.26
	3			35	22		35.52	23.16

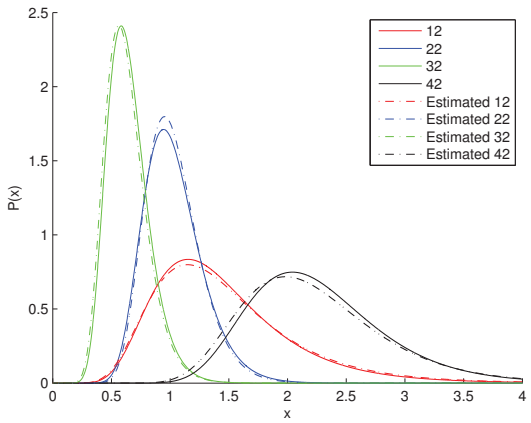
results.

5.4.2 Visual Scenes Categorization

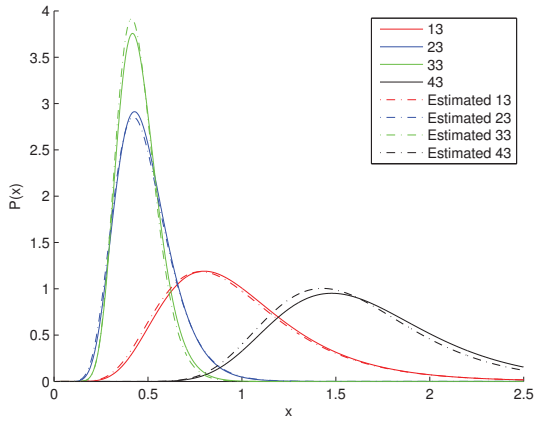
In this section we investigate the performance of our algorithm using real-life data. We consider the publicly available visual scenes dataset that was proposed in [147]. We mainly used four categories which are Highway (260 images), InsideCity (308 images), TallBuilding (356 images), and Forest (328 images). Fig.5.5 illustrates samples of the considered dataset. The images within a given category are diverse and the visual scenes have different objects, colors and shapes. The features of each image have been extracted using the local Histogram of Oriented Gradient (HOG) descriptor proposed in [75]. The HOG algorithm is efficient in terms of detecting local characteristics of



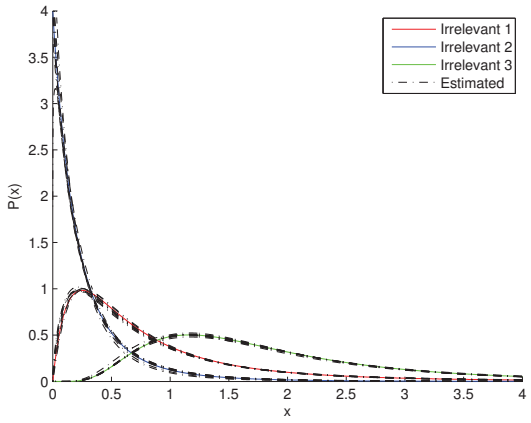
(a) relevant dimension 1



(b) relevant dimension 2



(c) relevant dimension 3



(d) irrelevant dimensions 4-5-6-7-8-9-10-11

Figure 5.4: Different generated inverted Beta distributions labeled in ji representing the 11 dimensions of the model in Table 5.3.

images and object recognition. During our experiments each image was represented by a 81-dimensional feature vector.

The experiments consists of establishing a clustering of the data basing on the hierarchal construction strategy that has been proposed in chapter 2, where a parent cluster is composed of sub children clusters. As proposed in chapter 2, we label each cluster according to the images that have been grouped into it. Indeed, we attribute each cluster k to the super class j whose elements are the most present in cluster k such that

$$label_{cluster_k} = \arg\max_j \frac{elements\ of\ superclass\ j\ in\ cluster_k}{elements\ in\ cluster_k} \tag{5.57}$$

We consider that we have four super classes that are Highway, InsideCity, TallBuiling and Forest.

Figure 5.5: Visual Scenes Dataset : Forest, InsideCity, Highway, TallBuilding.

Indeed, we have shown in in chapter 2 that it is more appropriate to represent a given object class by a mixture than one single distribution, which is convenient with variational inference that is able to set the optimal number of components to model a given dataset without the need of forcing the number of components to be equal to the number of object classes. Thus, the optimal number of components can be largely higher than the number of object classes. The work in [129] particularly shows that the appropriate number of components for the mixture can be determined in a single training run without recourse to cross-validation using variational inference. In this experiment we consider to compare mixture models that mainly use the inverted Dirichlet distributions and the generalized inverted Dirichlet distributions that involve inverted Beta distributions for each dimension. It as been shown in chapter 3 that these mixture models outperform the GMM in terms of clustering accuracy. We propose to consider the variational infinite GID mixture with features selection (Var-InfGID-FS), the variational infinite GID mixture without considering the features selection (Var-InfGID), the GID mixture using EM and MML (MML-EM-GID), and the ID mixture using EM and MML (MML-EM-ID). For the Var-InfGID-FS we set the initial values of $M = 25$ and $K = 30$, following the strategy proposed by [62, 143, 144] which consists of setting a larger number of clusters than four in our case. For the Var-InfGID we set the initial values of $M = 25$, and as for the MML-EM-GID and MML-EM-GID, we consider 25 models going from 1 component to 25 components, and then we use the MML as defined in chapter 3 to select the optimal model. Tables.5.4, 5.5, 5.6 and 5.7 show the confusion matrices of the clustering results using Var-InfGID-FS , Var-InfGID, MML-EM-GID and MML-EM-ID, respectively. Using those confusion matrices we calculate the clustering accuracies reported in Table.5.8, with the relevant number of clusters that constitute the model. The obtained results show that the Var-InfGID-FS outperforms the other algorithms with an accuracy equal to 83.39%, followed by the

Table 5.4: Visual Scenes Var-InfGID-FS confusion matrix.

	Forest	Highway	InsideCity	TallBuilding
Forest	324	0	0	4
Highway	11	238	9	2
InsideCity	22	9	192	85
TallBuilding	28	13	25	290

Table 5.5: Visual Scenes Var-InfGID confusion matrix.

	Forest	Highway	InsideCity	TallBuilding
Forest	322	0	2	4
Highway	1	194	64	1
InsideCity	0	9	184	115
TallBuilding	1	5	46	304

Table 5.6: Visual Scenes MML-EM-GID confusion matrix.

	Forest	Highway	InsideCity	TallBuilding
Forest	319	0	2	7
Highway	0	188	70	2
InsideCity	0	5	246	57
TallBuilding	1	3	116	236

Table 5.7: Visual Scenes MML-EM-ID confusion matrix.

	Forest	Highway	InsideCity	TallBuilding
Forest	320	0	2	6
Highway	1	185	74	0
InsideCity	0	5	281	22
TallBuilding	1	4	152	199

Var-InfGID with accuracy equal to 80.19% which shows the merit of variational inference and features selection.

Table 5.8: Obtained Accuracies and corresponding of clusters for each model : Visual Scenes Dataset

	GID VAR FS	GID VAR	GID EM MML	ID EM MML
Accuracy	83.39%	80.19%	78.99%	78.67%
Number of Clusters	7	5	6	8

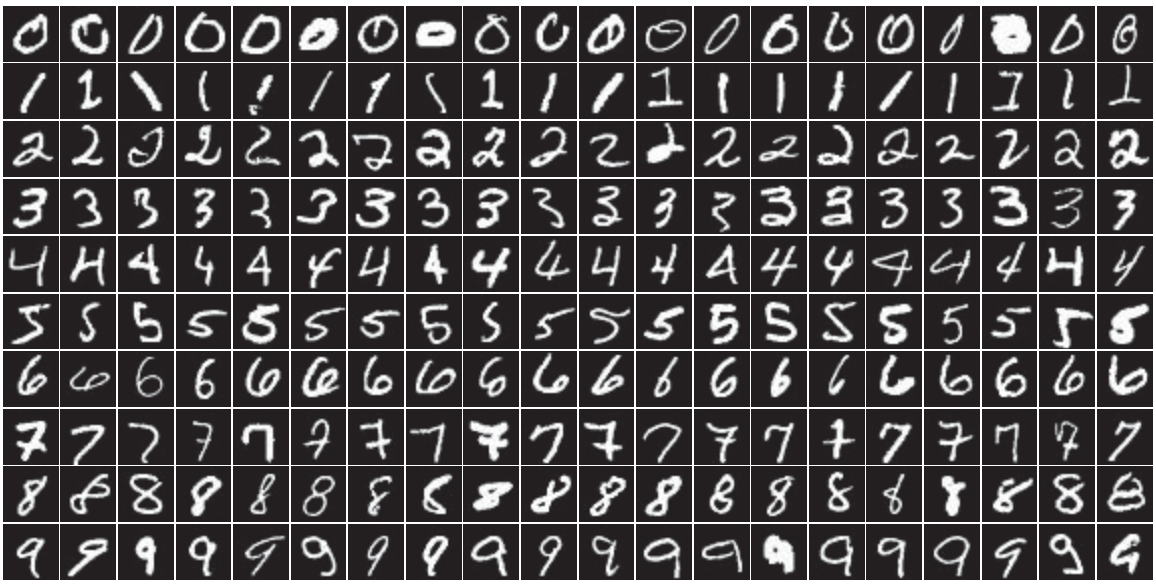


Figure 5.6: Digits dataset : sample images.

5.4.3 Digits Categorization

For the second application on a real computer vision dataset, we consider the public available MNIST digits database which is a large database of handwritten digits [148]. Each MNIST image is a digitized picture of a single handwritten digit character. Each image is 28 x 28 pixels in size. Each pixel value is between 0, which represents white, and 255, which represents black. Intermediate pixel values represent shades of gray. Fig.5.6 shows samples from 1 to 9 of the handwritten digits. The digits recognition may be easy for a human being in most of cases, but it still a challenging application for machine learning approaches. We also consider the HOG features that we extract from 60000 digits images that are distributed as follows ; Zero (5923 images), One (6742 images), Two (5958 images), Three (6131 images), Four (5842 images), Five (5421 images), Six (5918 images), Seven (6265 images), Eight (5851 images), Nine (5949 images). Bear in mind that our experiment consists of clustering without any training phase as data is injected to the algorithm without any prior knowledge about the observations labels. As illustrated in Fig.5.6, the optical digits for a given number take different shapes and orientations, therefore we expect them to form a considerable number of clusters for one single number. We set an initial value of $M = 500$ for the Var-InfGID-FS and Var-InfGID. We set $K = 100$ for the Var-InfGID-FS. The use of MML is not practical in this case, especially when the number of classes increases significantly and we have to go through 500 models that are computationally costly in order to select the best model, in contrast with the variational inference that needs a single run in order to select the optimal value of M. Therefore this experiment is restricted on the Var-InfGID-FS and Var-InfGID, in order to show the merit of features selection on the clustering accuracy. Tables. 5.9 and 5.10 respectively

show the confusion matrices obtained using the Var-InfGID-FS and Var-InfGID. We report on the accuracy of clustering in Table.5.11.

Table 5.9: Digits Var-InfGID-FS confusion matrix.

	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
Zero	5716	39	22	4	2	18	23	11	67	21
One	0	6522	64	1	6	0	4	16	28	101
Two	12	14	5502	198	15	13	4	83	96	21
Three	16	12	436	5036	6	229	2	125	202	67
Four	1	45	23	2	5320	5	88	26	46	286
Five	31	26	20	209	7	4712	65	32	238	81
Six	31	57	9	1	23	40	5655	0	100	2
Seven	6	23	96	48	2	4	0	5753	43	290
Eight	38	59	61	70	18	192	97	43	5128	145
Nine	63	29	24	49	62	27	11	229	103	5352

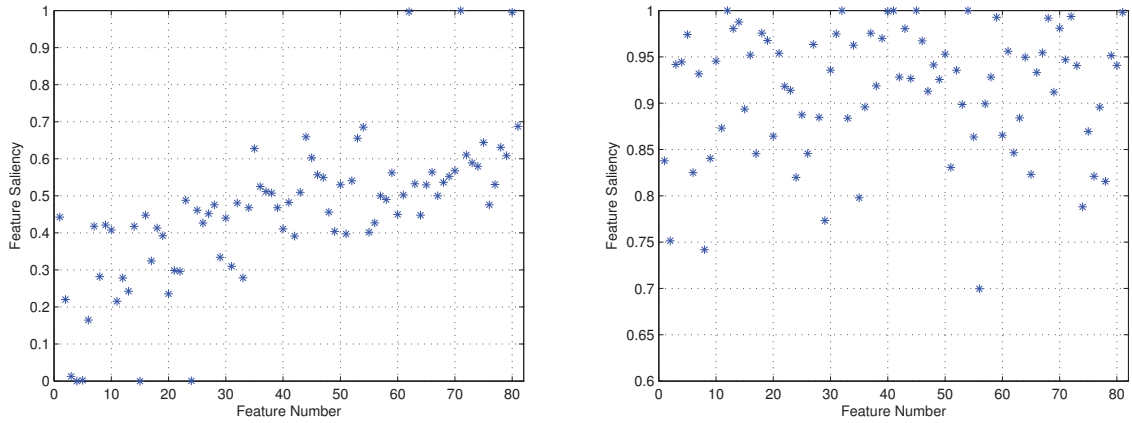
Table 5.10: Digits Var-InfGID confusion matrix.

	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
Zero	5601	47	18	128	0	26	36	13	46	8
One	0	6595	92	10	5	1	3	32	0	4
Two	6	15	5413	374	2	8	7	67	49	17
Three	9	3	373	5431	3	130	4	61	72	45
Four	0	51	25	21	5396	11	121	32	25	160
Five	13	4	24	545	2	4553	76	12	159	33
Six	34	59	12	25	16	25	5678	1	66	2
Seven	0	9	101	151	20	22	1	5868	14	79
Eight	49	59	48	497	29	191	147	44	4640	147
Nine	53	43	20	318	127	16	11	355	64	4942

Table 5.11: Obtained Accuracies and corresponding of clusters for each model : Digits Dataset

	GID VAR FS	GID VAR
Accuracy	91.16%	90.20 %
Number of Clusters	240	179

We notice that the accuracy is improved by 0.96% when using the features selection, which concern 576 images out of 60000. Naturally the impact of this accuracy improvement increases when the number of clustered data is considerable.



(a) Visual Scenes dataset (b) Digits dataset

Figure 5.7: Features saliency of different datasets.

5.4.4 Discussion

The model we have built shows the merit of the variational inference over the EM learning using MML. Using the Var-InfGID-FS improved the clustering accuracy by 4.72% (using MML-EM-ID), 4.40% (using MML-EM-GID), and 3.20% (using Var-InfGID), when applied on the visual scenes dataset. Also, using the Var-InfGID-FS improved the accuracy of clustering by 0.96% (using Var-InfGID) when applied on the 60000 digits dataset, which shows the merit of features selection. Fig.5.7 show the features saliency of the two datasets when using the Var-InfGID-FS. We notice that the algorithm was capable to assign different weights to features as some features are insignificant in the clustering process, and some others have a high discriminative effect. When we compare the number of clusters obtained when using the Var-InfGID-FS with the one obtained when using Var-InfGID, we notice that usually the number of clusters resulted by the Var-InfGID-FS is higher. This is due to the fact that the features selection leads to a denoising process that discard the features that do not have a discriminative effect and focus on the features that contribute the most to construct clusters with a higher cohesion which may lead to an increase of the number of clusters.

5.5 Summary

We have developed a variational Bayesian learning framework for the infinite generalized Dirichlet mixture model that has proven its capability to model multidimensional data. We also integrate a

“features selection" approach to highlight the features that are most informative in order to construct an appropriate model in terms of clustering accuracy. Experiments on synthetic data as well as real data generated from visual scenes and handwritten digits datasets illustrate and validate the proposed approach that can be used as complementary alternative to the Markov chain Monte Carlo for inference in large statistical models. The variational Bayesian learning is shown to be more effective than the EM algorithm and MML used in chapter 3.

Chapter 6

Conclusions

In this thesis we have developed several techniques basing on mixture-models in order to find new approaches to model data and improve clustering accuracy that is dependent on the application context and users perceptions. We have mainly developed our approaches using two distributions, namely the inverted Dirichlet (ID) and the generalized inverted Dirichlet (GID) distributions. The latter is shown to give better results in terms of data modeling capabilities and clustering accuracy. Apart from Chapter 5 which is specifically dedicated to the GID mixture model, the developed techniques in Chapter 2, Chapter 3, and Chapter 4 have enough generalization to integrate any other distribution. The choice of the of the ID and GID distributions is driven by their capabilities to model positive data that naturally occur in real life applications.

In Chapter 2, we have developed a methodology to model data in a flexible hierarchical way that can be altered on the fly according to a given ontology. The proposed model reduces the gap between the system representation of data in terms of clustering, and the human comprehension level that groups these clusters to form semantically understandable object representations. The work addresses the parameter estimation of different inverted Dirichlet distributions within a hierarchical model using a bottom-top strategy. The proposed methodology has been shown to outperform the classical approach where each class is represented by a single mixture component in the training phase. We also proposed a voting strategy to assign semantic meanings to the system level clusters in order to represent a given object.

Chapter 3 was devoted to develop a methodology that enables a learning system to handle new coming data, learn it and update the data model according to it. The proposed approach enables users to control how the data should be perceived by the system. The work addresses the parameter estimation of GID mixtures, the update of their parameters and creation of new classes when needed. Several probabilistic metrics has been developed for the GID distribution such as Kullback-Leibler divergence. We performed a simulation on synthetic data which has shown the

capability of the framework to model the data as intended. We also validated this model using real-life data for the problem of clustering and classification, including the hierarchical model that we proposed before. We have also shown that our model outperforms in terms of classification accuracy several other model-based approaches. The application of our approach can reach several research and industrial domains especially when a system needs a sort of semantic understanding when processing the data which is introduced by the perception parameter that we integrated in our model.

In Chapter 4 we have proposed a statistical framework whose purpose is to help a user find target concepts within a set of data, without expressing specific query to the system but rather basing on a mental process. The system tries to distinguish the user needs basing on his/her selections. Such search processes are justified by the fact that, in many cases, users cannot express what they have in mind, or lack the needed vocabulary to describe a concept. We have mainly included the multi target classes search within one single search process, and also included the multi selection and no preference options compared to some previous research work that tried to solve the "Zero Page" problem. We have also proposed a new data model basing on the GID mixture. A such model, can be used to construct different models of the data according to the needs of the system designer, and basing on the hierarchical model proposed in Chapter 2 and Chapter 3.

Last and not least, we have proposed in Chapter 5 an approach to model and cluster data using an infinite GID mixture with features selection. The model integrates a set of inverted Beta distributions that represent two majors aspects ; relevant features and irrelevant features. The variational Bayesian inference has been used to estimate the parameters of our model, and the obtained results when applied on real data show its merit over the conventional EM algorithm. We also show that the use of features selection leads to better results in terms of clustering accuracy. The proposed model can be used for any positive data and has promising applications in different areas that have huge amount of data to be clustered and analyzed statistically. The developed model represents an alternative choice to the model developed in Chapter 3 for the arriving data that uses MML and EM to estimate the parameters and select the model that serves to establish updates basing on developed metrics.

The application of our approaches can reach several research and industrial domains, and can help to reduce the semantic gap between system levels and human beings understandable level. Future works could be devoted to the learning of the proposed models in the presence of constraints (e.g. two objects should be assigned the same label) (see, for instance, [149]), and the investigation of its performance within the statistical mental frameworks proposed in Chapter 4. Further research perspectives could also cover the construction of a link between two target classes like for example finding an image of buddha with a sunflower. Also research can be conducted to find

suitable ways to update mixtures while performing features selection. Indeed the new coming data could have other relevant features than the ones of the already existing data constructing the same object class. Therefore, hierarchical feature selection could be considered as well. Another interesting point to conduct research on is the features role in the hierarchy. Indeed, the adoption of a bottom-top strategy to form the hierarchy enables the system to have very discriminative features on the lowest level of the hierarchy that become less and less selective when we move to the upper levels. For instance, consider a feature which is discriminative for 'Tomato' and 'Pear'. When we move to the *super class* fruits, that feature is not discriminative anymore as the *super class* fruits contains both 'Tomato' and 'Pear'. On the other hand when we move from an upper level to the bottom in the hierarchy, one feature might be discriminative between fruits and non fruits, but loses its importance, when we want to distinguish between 'Tomato', and 'Apple' once we zoom on the fruits. Thus, furture research can be conducted on hierarchical features importance/contribution models for the classification that defines for each level what are the features that are the most discriminative. This can improve the results of classification when we are interested in one specific level or a specific *super class*. Also, future research can be devoted to investigate automatic learning of the perception parameter introduced in Chapter 3 instead of fixing it by the user basing on the structure of the training datasets. Last and not least, expectation propagation algorithms can be investigated as a possible alternative to the variational Bayesian inference that we have used in Chapter 5, in order to have better data models.

Bibliography

[1] Google. YouTube Statistics. https://www.youtube.com/yt/press/statistics.html, 2015. [Online; accessed 7-March-2015].

[2] X. Rui and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[3] N. Bargary, J. Hinde, and A.F. Garcia. Finite mixture model clustering of snp data. In G. MacKenziet and D. Peng, editors, *Statistical Modelling in Biostatistics and Bioinformatics*, Contributions to Statistics, pages 139–157. Springer International Publishing, 2014.

[4] D. C. Koestler, C. J. Marsit, B. C. Christensen, K. T. Kelsey, and E. A. Houseman. A recursively partitioned mixture model for clustering time-course gene expression data. *Translational Cancer Research*, 3(3), 2014.

[5] S. Prabhakaran, M. Rey, O. Zagordi, N. Beerenwinkel, and V. Roth. Hiv haplotype inference using a propagating dirichlet process mixture model. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 11(1):182–191, 2014.

[6] K.A. Tran, N.Q. Vo, and G. Lee. A novel clustering algorithm based gaussian mixture model for image segmentation. In *Proc. of the 8th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '14, pages 97:1–97:5. ACM, 2014.

[7] I.S. Topkaya, H. Erdogan, and F. Porikli. Counting people by clustering person detector outputs. In *Proc. of the 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 313–318, Aug 2014.

[8] B. Zhou, X.Tang, and X. Wang. Learning collective crowd behaviors with dynamic pedestrian-agents. *International Journal of Computer Vision*, 111(1):50–68, 2015.

[9] S. Boutemedjet and D. Ziou. Predictive approach for user long-term needs in content-based image suggestion. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1242–1253, 2012.

[10] A. Beutel, K. Murray, C. Faloutsos, and A.J. Smola. Cobafi: Collaborative bayesian filtering. In *Proc. of the 23rd International Conference on World Wide Web*, WWW 14, pages 97–108. ACM, 2014.

[11] H. Yin, B. Cui, L. Chen, Z. Hu, and Z. Huang. A temporal context-aware model for user behavior modeling in social media systems. In *Proc. of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1543–1554. ACM, 2014.

[12] M. S. Handcock, A. E. Raftery, and J. M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

[13] T. Couronne, A. Stoica, and J.S. Beuscart. Online social network popularity evolution: An additive mixture model. In *Proc. of International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 346–350, 2010.

[14] Dan Xu and Shiqiang Yang. Location prediction in social media based on contents and graphs. In *Proc. of Fourth International Conference on Communication Systems and Network Technologies (CSNT)*, pages 1177–1181, 2014.

[15] A.P. Dawid. Properties of diagnostic data distributions. *Biometrics*, pages 647–658, 1976.

[16] B.S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Wiley, 2009.

[17] G. J. McLachlan and D. Peel. *Finite Mixture Models*. New York: Wiley, 2000.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society*, 39(1):1–38, 1977.

[19] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. A Wiley interscience publication. Wiley, 1997.

[20] D. A. Binder. Bayesian cluster analysis. *Biometrika*, 65(1):31–38, 1978.

[21] D. Husmeier. The bayesian evidence scheme for regularizing probability-density estimating neural networks. *Neural Comput.*, 12(11):2685–2717, 2000.

[22] C. Andrieuand, N. Freitas, A. Doucet, and M. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[23] C.M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[24] C. S. Wallace. *Statistical and inductive inference by minimum message length*. Springer-Verlag, 2005.

[25] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[26] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[27] M. A. T. Figueiredo, J. M. N Leit ao, and A. Jain. On fitting mixture models. In *In Proc. of the Second International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 54–69. Springer-Verlag, 1999.

[28] T. Bdiri, N. Bouguila, and D. Ziou. Object clustering and recognition using multi-finite mixtures for semantic classes and hierarchy modeling. *Expert Systems with Applications*, 41(4, Part 1):1218 – 1235, 2014.

[29] T. Bdiri, N. Bouguila, and D. Ziou. A statistical framework for online learning using adjustable model selection criteria. *Engineering Applications of Artificial Intelligence*, 2014. Manuscript submitted for publication.

[30] T. Bdiri, N. Bouguila, and D. Ziou. A statistical framework for mental targets search using mixture models. In Y. Laalaoui and N. Bouguila, editors, *Artificial-Intelligence Applications in Information and Communication Technologies*. Springer-Verlag, 2015. To appear.

[31] T. Bdiri, N. Bouguila, and D. Ziou. Variational bayesian inference for infinite generalized inverted dirichlet mixtures with features selection via inverted beta distributions. *Applied Intelligence*, 2015. Manuscript submitted for publication.

[32] A. K. Jain, M. Murty and P. Flynn. Data clustering: a Review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[33] D. H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, 4:147–178, 1996.

[34] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2002.

[35] AW. Edwards and LL. Cavalli-Sforza. A method of cluster analysis. *Biometrics*, 21:362–375, Jun 1965.

[36] N. E. Day. Estimating the components of a mixture of normal distributions. *Biometrika*, 56(3):463–474, 1969.

[37] A. Sefidpour and N. Bouguila. Spatial color image segmentation based on finite non-gaussian mixture models. *Expert Systems With Applications*, 39(10):8993–9001, 2012.

[38] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:747–757, 2000.

[39] M.S. Allili, D. Ziou, N. Bouguila, and S. Boutemedjet. Image and video segmentation by combining unsupervised generalized gaussian mixture modeling and feature selection. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(10):1373–1377, 2010.

[40] I.C. Gormley and T.B. Murphy. A mixture of experts latent position cluster model for social network data. *Statistical Methodology*, 7(3):385 – 405, 2010.

[41] M. Newman, M. EJ, and E.A. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564–9569, 2007.

[42] S. Morinaga and K. Yamanishi. Tracking dynamics of topic trends using a finite mixture model. In *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 811–816, 2004.

[43] D. C. Koestler, C. J. Marsit, B. C. Christensen, M. R. Karagas, R. Bueno, D. J. Sugarbaker, K. T. Kelsey, and E. A. Houseman. Semi-supervised recursively partitioned mixture models for identifying cancer subtypes. *Bioinformatics*, 26(20):2578–2585, October 2010.

[44] W. Tao, I. Cheng, and A. Basu. Fully automatic brain tumor segmentation using a normalized gaussian bayesian classifier and 3d fluid vector flow. In *Proc. of the 17th IEEE International Conference on Image Processing (ICIP)*, pages 2553–2556, 2010.

[45] B. Neelon, G. K. Swamy, L. F. Burgette, and M. L. Miranda. A bayesian growth mixture model to examine maternal hypertension and birth outcomes. *Statistics in Medicine*, 30(22):2721–2735, 2011.

[46] P. Schlattmann. *Medical applications of finite mixture models*. Statistics for Biology and Health. Springer-Verlag Berlin Heidelberg, 2009.

[47] S. Rattanasiri, D. Böhning, P. Rojanavipart, and S. Athipanyakom. A mixture model application in disease mapping of malaria. *The Southeast Asian journal of tropical medicine and public health*, 35(1):38–47, 2004.

[48] M. Kim, S. B. Cho, and J. H. Kim. Mixture-model based estimation of gene expression variance from public database improves identification of differentially expressed genes in small sized microarray data. *Bioinformatics*, 26(4):486–492, 2010.

[49] P. Meinicke, K. P. AÃ§hauer, and T. Lingner. Mixture models for analysis of the taxonomic composition of metagenomes. *Bioinformatics*, 27:1618–1624, 2011.

[50] Y. Ji, C. Wu, P. Liu, J. Wang, and K. R. Coombes. Applications of beta-mixture models in bioinformatics. *Bioinformatics*, 21(9):2118–2122, 2005.

[51] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[52] J. Rissanen. Hypothesis selection and testing by the MDL principle. *The Computer Journal*, 42(4):260–269, 1999.

[53] D. Ziou, T. Hamri, and S. Boutemedjet. A hybrid probabilistic framework for content-based image retrieval with feature weighting. *Pattern Recognition*, 42(7):1511–1519, 2009.

[54] A. S. Bakhtiari and N. Bouguila. A hierarchical statistical model for object classification. In *Proc. of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 493–498, 2010.

[55] A. S. Bakhtiari and N. Bouguila. An expandable hierarchical statistical framework for count data modeling and its application to object classification. In *Proc. of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 817–824, 2011.

[56] A. S. Bakhtiari and N. Bouguila. A novel hierarchical statistical model for count data modeling and its application in image classification. In *Proc. of the 19th International Conference on Neural Information Processing (ICONIP)*, pages 332–340, 2012.

[57] C. M. Bishop and M. E. Tipping. A hierarchical latent variable model for data visualization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):281–293, March 1998.

[58] H. Permuter, J. Francos, and H. Jermyn. Gaussian mixture models of texture and colour for image database retrieval. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 569–572, 2003.

[59] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proc. of the 17th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 28–31, 2004.

[60] M.Yang and N. Ahuja. Gaussian mixture model for human skin color and its applications in image and video databases. In *Proc. of the International Society for Optics and Photonics (SPIE)*, pages 458–466, 1999.

[61] C. Weiling, L. Lei, and M. Yang. A gaussian mixture model-based clustering algorithm for image segmentation using dependable spatial constraints. In *Proc. of the 3rd International Congress on Image and Signal Processing (CISP)*, volume 3, pages 1268–1272, 2010.

[62] S. Boutemedjet, N. Bouguila, and D. Ziou. A hybrid feature extraction selection approach for high-dimensional non-gaussian data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1429–1443, 2009.

[63] N. Bouguila, D. Ziou, and J. Vaillancourt. Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application. *IEEE Transactions on Image Processing*, 13(11):1533–1543, 2004.

[64] N. Bouguila and D. Ziou. A hybrid sem algorithm for high-dimensional unsupervised learning using a finite generalized dirichlet mixture. *IEEE Transactions on Image Processing*, 15(9):2657–2668, 2006.

[65] N. Bouguila, D. Ziou, and R. I. Hammoud. On bayesian analysis of a finite generalized dirichlet mixture via a metropolis-within-gibbs sampling. *Pattern Analysis and Applications*, 12(2):151–166, 2009.

[66] T. Bdiri and N. Bouguila. Positive vectors clustering using inverted dirichlet finite mixture models. *Expert Systems With Applications*, 39(2):1869–1882, 2012.

[67] T. Bdiri and N. Bouguila. Learning inverted dirichlet mixtures for positive data clustering. In *Proc. of the 13th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC)*, pages 265–272, 2011.

[68] M. Ghorbel. On the Inverted Dirichlet Distribution. *Communications in Statistics-Theory and Methods*, 39:21–37, 2010.

[69] G. G. Tiao and I. Cuttman. The inverted dirichlet distribution with applications. *Journal of the American Statistical Association*, 60(311):793–805, 1965.

[70] H. Yassaee. Inverted Dirichlet Distribution and Multivariate Logistic Distribution. *The Canadian Journal of Statistics*, 2(1):99–105, 1974.

[71] F. A. Graybill. *Matrices with applications in statistics*. Wadsworth statistics/probability series. Wadsworth International Group, 1983.

[72] X. He, D. Simpson, and S. Portnoy. Breakdown robustness of tests. *Journal of the American Statistical Association*, 85(410):446–452, 1990.

[73] C. Hennig. Breakdown points for maximum likelihood estimators of location-scale mixtures. *Annals of Statistics*, 32(4):1313–1340, 2004.

[74] N. Bouguila, K. Almakadmeh, and S. Boutemedjet. A finite mixture model for simultaneous high-dimensional clustering, localized feature selection and outlier rejection. *Expert Systems with Applications*, 39(7):6641–6656, 2012.

[75] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE Computer Society, 2005.

[76] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. of the 14th international joint conference on Artificial Intelligence*, volume 2, pages 1137–1143. Morgan Kaufmann Publishers Inc., 1995.

[77] Ye Lu, HongJiang Zhang, Liu Wenyin, and Chunhui Hu. Joint semantics and feature based image retrieval using relevance feedback. *IEEE Transactions on Multimedia*, 5(3):339–347, 2003.

[78] J. Hu and A. Bagga. Categorizing images in web documents. *IEEE MultiMedia*, 11(1):22–30, 2004.

[79] S. Boutemedjet and D. Ziou. Long-term relevance feedback and feature selection for adaptive content based image suggestion. *Pattern Recognition*, 43(12):3925 – 3937, 2010.

[80] S. Boutemedjet and D. Ziou. A graphical model for context-aware visual content recommendation. *IEEE Transactions on Multimedia*, 10(1):52–62, 2008.

[81] I. Mironica, B. Ionescu, J. Uijlings, and N. Sebe. Fisher kernel based relevance feedback for multimodal video retrieval. In *Proc. of the 3rd ACM Conference on International Conference on Multimedia Retrieval (ICMR)*, pages 65–72. ACM, 2013.

[82] J.Y. Chen, C.A. Bouman, and J.C. Dalton. Hierarchical browsing and search of large image databases. *IEEE Transactions on Image Processing*, 9(3):442–455, 2000.

[83] J.M. Kleinberg. Bursty and hierarchical structure in streams. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 91–101. ACM, 2002.

[84] R. M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[85] S. Bourouis, M.A. Mashrgy, and N. Bouguila. Bayesian learning of finite generalized inverted dirichlet mixtures: Application to object classification and forgery detection. *Expert Systems with Applications*, 41(5):2329 – 2336, 2014.

[86] K.Zhang and J.T. Kwok. Simplifying mixture models through function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1577–1584. MIT Press, 2006.

[87] G. S. Lingappaiah. On the generalised inverted dirichlet distribution. *Demonstratio Mathematica*, 9:423–433, 1976.

[88] N. Bouguila and D. Ziou. Online clustering via finite mixtures of dirichlet and minimum message length. *Engineering Applications of Artificial Intelligence*, 19(4):371–379, 2006.

[89] O. Amayri and N. Bouguila. On online high-dimensional spherical data clustering and feature selection. *Engineering Applications of Artificial Intelligence*, 26(4):1386–1398, 2013.

[90] J. F. Yao. On recursive estimation in incomplete data models. *Statistics*, 34(1):27–51, 2000.

[91] N. Bouguila and D. Ziou. High-dimensional unsupervised selection and estimation of a finite generalized dirichlet mixture model based on minimum message length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1716–1731, 2007.

[92] N. Vasconcelos. On the complexity of probabilistic image retrieval. In *Proc. of the Eighth IEEE International Conference on Computer Vision (ICCV)*, pages 400–407 vol.2. IEEE, 2001.

[93] J. Sinkkonen and S. Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14(1):217–239, 2002.

[94] T.F Wu, C.J Lin, and R.C. Weng. Probability estimates for multi-class classification by pairwise coupling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–8. MIT Press, 2003.

[95] P. Liang and D. Klein. Online em for unsupervised models. In *Proc of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619. Association for Computational Linguistics, 2009.

[96] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 161–168. Curran Associates, 2007.

[97] A. Rényi. On measures of entropy and information. In *Proc. of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, pages 547–561. Univ. of Calif. Press, 1960.

[98] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008.

[99] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages II–90–II–96 Vol.2. IEEE, 2004.

[100] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1270–1281, 2008.

[101] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *Proc. of the Eighth IEEE International Conference on Computer Vision (ICCV)*, pages 503–510 Vol. 1. IEEE, 2005.

[102] J.Z. Pan, Y. Ren, H. Wu, and M. Zhu. Query generation for semantic datasets. In *Proc of the Seventh International Conference on Knowledge Capture*, K-CAP '13, pages 113–116. ACM, 2013.

[103] S. ShahabSaquib, S. Jamshed, and A. Rashid. User feedback based evaluation of a product recommendation system using rank aggregation method. In *Advances in Intelligent Informatics*, volume 320 of *Advances in Intelligent Systems and Computing*, pages 349–358. Springer International Publishing, 2015.

[104] Y. Zhimin, Y. Xiangzhan, and Z. Hongli. Commodity recommendation algorithm based on social network. In *Advances in Computer Science and its Applications*, volume 279 of *Lecture Notes in Electrical Engineering*, pages 27–33. Springer Berlin Heidelberg, 2014.

[105] A. Kaasinen and Y. Yong-Ik. Service engagement model for mobile advertising based on user behavior. In *International Conference on Information Networking (ICOIN)*, pages 131–134, 2013.

[106] S. Kim, T. Qin, T.Y. Liu, and H. Yu. Advertiser-centric approach to understand user click behavior in sponsored search. *Information Sciences*, 276(0):242 – 254, 2014.

[107] M. Ferecatu and D. Geman. A statistical framework for image category search from a mental picture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1087–1101, 2009.

[108] L. Jia and J.Z. Wang. Real-time computerized annotation of pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):985–1002, 2008.

[109] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[110] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, H. Qian, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *Computer*, 28(9):23–32, 1995.

[111] M.L. Kherfi and D. Ziou. Relevance feedback for cbir: a new approach based on probabilistic feature weighting with positive and negative examples. *Image Processing, IEEE Transactions on*, 15(4):1017–1030, 2006.

[112] R. Yong, T.S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[113] X.S. Zhou and T.S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8(6):536–544, 2003.

[114] N. Vasconcelos and A. Lippman. A probabilistic architecture for content-based image retrieval. In *Proc of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 216–221, 2000.

[115] I.J. Cox, M.L. Miller, T.P. Minka, T.V. Papathomas, and P.N. Yianilos. The bayesian image retrieval system, pichunter: theory, implementation, and psychophysical experiments. *IEEE Transactions on Image Processing*, 9(1):20–37, 2000.

[116] N. Suditu and F. Fleuret. Heat: Iterative relevance feedback with one million images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2118–2125, 2011.

[117] N. Suditu and F. Fleuret. Iterative relevance feedback with adaptive exploration/exploitation trade-off. In *Proc of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1323–1331. ACM, 2012.

[118] M.A. Mashrgy, T. Bdiri, and N. Bouguila. Robust simultaneous positive data clustering and unsupervised feature selection using generalized inverted dirichlet mixture models. *Knowledge-Based Systems*, 59(0):182 – 195, 2014.

[119] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Proc. of conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 178–178, 2004.

[120] J. Lokoc, T. Grosup, P. Cech, and T. Skopal. Towards efficient multimedia exploration using the metric space approach. In *12th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–4, 2014.

[121] J. Winn and C.M. Bishop. Variational Message Passing. *Journal of Machine Learning Research*, 6:661–694, 2005.

[122] K. Dimitris and X. Evdokia. Choosing initial values for the {EM} algorithm for finite mixtures. *Computational Statistics and Data Analysis*, 41(3âĂŞ4):577 – 590, 2003.

[123] C. P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer, 2nd edition, 2007.

[124] N. Bouguila and T. Elguebaly. A fully bayesian model based on reversible jump {MCMC} and finite beta mixtures for clustering. *Expert Systems with Applications*, 39(5):5946 – 5959, 2012.

[125] M. Pereyra, N. Dobigeon, H. Batatia, and J. Tourneret. Estimating the granularity coefficient of a potts-markov random field within a markov chain monte carlo algorithm. *IEEE Transactions on Image Processing*, 22(6):2385–2397, 2013.

[126] N. Bouguila and D. Ziou. A dirichlet process mixture of dirichlet distributions for classification and prediction. In *IEEE Workshop on Machine Learning for Signal Processing (MLSP)*, pages 297–302, 2008.

[127] M. K. Cowles and B. P. Carlin. Markov chain monte carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[128] N. Bhatnagar, A. Bogdanov, and E. Mossel. The computational complexity of estimating mcmc convergence time. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 6845 of *Lecture Notes in Computer Science*, pages 424–435. Springer Berlin Heidelberg, 2011.

[129] A. Corduneanu and C. M. Bishop. Variational bayesian model selection for mixture distributions. In *Proc. of the Eighth International Conference on Artificial Intelligence and Statistics*, page 27âĂŞ34. Morgan Kaufmann, 2001.

[130] S.L Tan and D.J. Nott. Variational approximation for mixtures of linear mixed models. *Journal of Computational and Graphical Statistics*, 23(2):564–585, 2014.

[131] M. N. Thanh and Q.M.J.Wu. Asymmetric mixture model with variational bayesian learning. In *Proc. of International Joint Conference on Neural Networks (IJCNN)*, pages 285–290, July 2014.

[132] M. Zhanyu and A. Leijon. Bayesian estimation of beta mixture models with variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2160–2173, 2011.

[133] N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions: Vol.: 2*. Wiley series in probability and mathematical statistics. Applied probability and statistics, 1995.

[134] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[135] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[136] C. Constantinopoulos, M.K. Titsias ., and A. Likas. Bayesian feature and model selection for gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1013–1018, 2006.

[137] W. Fan and N. Bouguila. Variational learning of a dirichlet process of generalized dirichlet distributions for simultaneous clustering and feature selection. *Pattern Recognition*, 46(10):2754 – 2769, 2013.

[138] D.M. Blei and M.I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.

[139] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[140] M. Opper and D. Saad. *Advanced mean field methods: theory and practice*. Neural Information Processing. Massachusetts Institute of Technology Press (MIT Press), 2001.

[141] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[142] H. Ishwaran and L.F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453), 2001.

[143] M. A.T Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[144] M.H.C Law, M.A.T. Figueiredo, and A.K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, 2004.

[145] M. T. Salter and T.B. Murphy. Variational bayesian inference for the latent position cluster model for network data. *Computational Statistics & Data Analysis*, 57(1):661–671, 2012.

[146] N. Nasios and A.G. Bors. Variational learning for gaussian mixture models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(4):849–862, 2006.

[147] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001.

[148] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[149] T. Lange, M.H.C. Law, A.K. Jain, and J.M. Buhmann. Learning with constrained and unlabelled data. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 731–738 vol. 1. IEEE, 2005.

[150] S. R. Dalal and W. J. Hall. Approximating priors by mixtures of natural conjugate priors. *Journal of the Royal Statistical Society*, 45(2):278–286, 1983.

[151] C. P. Robert. *The Bayesian Choice*. Springer Texts in Statistics. Springer-Verlag, 2001.

[152] L. D. Brown. Fundamentals of statistical exponential families with applications in statistical decision theory. *Lecture Notes-Monograph Series*, 9, 1986.

Appendix A

A.1 The Marginalization

Let the notation $\vec{Z}_n = z_{nj}$ denotes that $\vec{Z}_n(j) = z_{nj} = 1$, and all the other elements of $\vec{Z}_n$ are equal to zero. Let also the notation $\{\vec{Y}_{nj}\} = y_{nji}$ denotes that all the elements of $\{\vec{Y}_{nj}\}$ are equal to zero, except the element $y_{nji} = 1$.

A.1.1 First Level

The first level of the mixture can be deducted by the marginalization of Eq.2.3 over the hidden variables:

$$p(\vec{X}_n|\Theta,\vec{\pi}) = \sum_{j=1}^{M} p(\vec{X}_n,\vec{Z}_n = z_{nj}|\Theta,\vec{\pi}) = \sum_{j=1}^{M} p(\vec{X}_n|\vec{Z}_n = z_{nj},\Theta,\vec{\pi})p(\vec{Z}_n = z_{nj}|\vec{\pi}) \tag{A.1}$$

from Eq.2.2 we have :

$$p(\vec{Z}_n = z_{nj}|\vec{\pi}) = p(z_{nj} = 1|\vec{\pi}) = \pi_1^0 ... \pi_{j-1}^0 \pi_j^1 \pi_{j+1}^0 \dots \pi_M^0 = \pi_j \tag{A.2}$$

and from Eq.2.1, we have :

$$\begin{aligned} p(\vec{X}_n|\vec{Z}_n = z_{nj},\Theta,\vec{\pi}) &= p(\vec{X}_n|\vec{Z}_n = z_{nj},\Theta) \\ &= p(\vec{X}_n|z_{nj} = 1,\Theta) \\ &= p(\vec{X}_n|\theta_1)^0 ... p(\vec{X}_n|\theta_{j-1})^0 p(\vec{X}_n|\theta_j)^1 p(\vec{X}_n|\theta_{j+1})^0 ... p(\vec{X}_n|\theta_M)^0 \\ &= p(\vec{X}_n|\theta_j) \end{aligned} \tag{A.3}$$

By substituting Eqs.A.2 and A.3 into Eq.A.1, the first level of the mixture for a given vector $\vec{X}_n$ can be obtained:

$$p(\vec{X}_n|\Theta,\vec{\pi}) = \sum_{j=1}^{M} p(\vec{X}_n|\theta_j)\pi_j \tag{A.4}$$

A.1.2 Second Level

We proceed by the marginalization over the hidden variables, so the first level of our mixture can be deducted by the marginalization of Eq. 2.8:

$$\begin{aligned} p(\vec{X}_n|\varphi,\{w_{ji}\},\vec{\pi}) &= \sum_{j=1}^{M}\sum_{i=1}^{K_j} P(\vec{X}_n,\vec{Z}_n = z_{nj},\{\vec{Y}_{nj}\} = y_{nji}|\varphi,\{w_{ji}\},\vec{\pi}) \\ &= \sum_{j=1}^{M}\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{Z}_n = z_{nj},\{\vec{Y}_{nj}\} = y_{nji},\varphi)p(\{\vec{Y}_{nj}\} = y_{nji}|\vec{Z}_n = z_{nj},\{w_{ji}\},\vec{\pi})p(\vec{Z}_n = z_{nj}|\vec{\pi}) \end{aligned} \tag{A.5}$$

Eq.2.7 gives us

$$\begin{aligned} P(\{\vec{Y}_{nj}\} = y_{nji}|\vec{Z}_n = z_{nj},\{w_{ji}\},\vec{\pi}) &= P(y_{nji} = 1|z_{nj} = 1,\{w_{ji}\}) \\ &= w^0_{j1}...w^0_{j(i-1)}w^1_{ji}w^0_{j(i+1)}...w^0_{jK_j} = w_{ji} \end{aligned} \tag{A.6}$$

and we can obtain the following from Eq. 2.5

$$\begin{aligned} &p(\vec{X}_n|\vec{Z}_n = z_{nj},\{\vec{Y}_{nj}\} = y_{nji},\varphi) = p(\vec{X}_n|z_{nj} = 1,y_{nji} = 1,\varphi) \\ &= (p(\vec{X}_n|\vec{\varphi}_{11})^0...(p(\vec{X}_n|\vec{\varphi}_{1K_1})^0)\dots((p(\vec{X}_n|\vec{\varphi}_{(j-1)1})^0\dots(p(\vec{X}_n|\vec{\varphi}_{(j-1)K_{j-1}})^0)) \\ &...((p(\vec{X}_n|\vec{\varphi}_{(j)1})^0...(p(\vec{X}_n|\vec{\varphi}_{(j)(i-1)})^0)(p(\vec{X}_n|\vec{\varphi}_{ji}))^1)(p(\vec{X}_n|\vec{\varphi}_{j(i+1)})^0)(p(\vec{X}_n|\vec{\varphi}_{(j)K_j})^0)) \\ &\dots((p(\vec{X}_n|\vec{\varphi}_{(j+1)1})^0...(p(\vec{X}_n|\vec{\varphi}_{(j+1)K_{j+1}})^0))\dots((p(\vec{X}_n|\vec{\varphi}_{M1})^0\dots(p(\vec{X}_n|\vec{\varphi}_{MK_M})^0)) = p(\vec{X}_n|\vec{\varphi}_{ji}) \end{aligned} \tag{A.7}$$

By substituting Eqs.A.2, A.6 and A.7 into Eq.A.5, the second level mixture can be written as

$$p(\vec{X}_n|\varphi,\{w_{ji}\},\vec{\pi}) = \sum_{j=1}^{M}\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji})w_{ji}\pi_j = \sum_{j=1}^{M}\pi_j(\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji})w_{ji}) \tag{A.8}$$

A.2 w_{ij} and π_j estimation

A.2.1 Estimation of w_{ji}

we have :

$$\begin{aligned}\frac{\partial Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial w_{ji}} &= \frac{\partial}{\partial w_{ji}}(\log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi}) + \lambda_1(1-\sum_{j=1}^{M}\pi_j) + \lambda_2(1-\sum_{j=1}^{M}\sum_{i=1}^{K_j}w_{ij})) \\ &= \frac{\partial \log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi})}{\partial w_{ji}} - \lambda_2 \end{aligned} \tag{A.9}$$

using Eq. 2.9 :

$$\begin{aligned}\frac{\partial \log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi})}{\partial w_{ji}} &= \sum_{i=1}^{N}\frac{\partial}{\partial w_{ji}}\log(\sum_{j=1}^{M}\pi_j(\sum_{i=1}^{K_j}p(\vec{X}_n|\vec{\varphi}_{ji})w_{ji})) \\ &= \sum_{n=1}^{N}\frac{\pi_j p(\vec{X}_n|\vec{\varphi}_{ji})}{\sum\limits_{j=1}^{M}\pi_j(\sum\limits_{i=1}^{K_j}p(\vec{X}_n|\vec{\varphi}_{ji})w_{ji})} \qquad //we\ multiply\ by\ \frac{w_{ji}}{w_{ji}} \\ &= \sum_{n=1}^{N}\frac{w_{ji}\pi_j p(\vec{X}_n|\vec{\varphi}_{ji})}{w_{ji}\sum\limits_{j=1}^{M}\sum\limits_{i=1}^{K_j}w_{ji}\pi_j p(\vec{X}_n|\vec{\varphi}_{ji})} \end{aligned} \tag{A.10}$$

The posterior probability $p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})$ is given by :

$$p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi}) = \frac{w_{ji}\pi_j p(\vec{X}_n|\vec{\varphi}_{ji})}{\sum\limits_{j=1}^{M}\sum\limits_{i=1}^{K_j}w_{ji}\pi_j p(\vec{X}_n|\vec{\varphi}_{ji})} \tag{A.11}$$

Then, Eq. A.10 becomes :

$$\frac{\partial \log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi})}{\partial w_{ji}} = \frac{\sum_{n=1}^{N}p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})}{w_{ji}} \tag{A.12}$$

So we have,

$$\frac{\partial Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial w_{ji}} = \frac{\sum_{n=1}^{N}p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})}{w_{ji}} - \lambda_2 = 0 \tag{A.13}$$

now we derive with respect to λ_2. We find :

$$\frac{\partial Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial \lambda_2} = 1 - \sum_{j=1}^{M}\sum_{i=1}^{K_j} w_{ij} = 0 \tag{A.14}$$

we have :

$$w_{ji} = \frac{1}{\lambda_2}\sum_{n=1}^{N} p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi}) \tag{A.15}$$

$$\sum_{j=1}^{M}\sum_{i=1}^{K_j} w_{ji} = \frac{1}{\lambda_2}\sum_{j=1}^{M}\sum_{i=1}^{K_j}\sum_{n=1}^{N} p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi}) = 1 \tag{A.16}$$

since,

$$\sum_{j=1}^{M}\sum_{i=1}^{K_j} p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi}) = 1 \tag{A.17}$$

Thus,

$$\sum_{j=1}^{M}\sum_{i=1}^{K_j}\sum_{n=1}^{N} p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi}) = N \tag{A.18}$$

So,

$$\lambda_2 = N \tag{A.19}$$

$$w_{ji} = \frac{\sum_{n=1}^{N} p(j,i|\vec{X}_n,\varphi,\{w_{ji}\},\vec{\pi})}{N} \tag{A.20}$$

A.2.2 Estimation of π_j

we have :

$$\begin{aligned}\frac{\partial Q(\mathscr{X},\varphi,\{w_{ji}\},\vec{\pi},\Lambda)}{\partial \pi_j} &= \frac{\partial}{\partial \pi_j}(\log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi}) + \lambda_1(1-\sum_{j=1}^{M}\pi_j) + \lambda_2(1-\sum_{j=1}^{M}\sum_{i=1}^{K_j} w_{ij})) \\ &= \frac{\partial \log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi})}{\partial \pi_j} - \lambda_1\end{aligned}$$

using Eq.2.9 :

$$\frac{\partial \log p(\mathscr{X}|\varphi,\{w_{ji}\},\vec{\pi})}{\partial \pi_j} = \sum_{i=1}^{N}\frac{\partial}{\partial \pi_j}\log(\sum_{j=1}^{M}\pi_j(\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji})w_{ji}))$$

$$= \sum_{n=1}^{N} \frac{\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji}) w_{ji}}{\sum_{j=1}^{M} \pi_j (\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji}) w_{ji})} \qquad //we\ multiply\ by\ \frac{\pi_j}{\pi_j}$$

$$= \sum_{n=1}^{N} \frac{\pi_j \sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji}) w_{ji}}{\pi_j \sum_{j=1}^{M} \pi_j (\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji}) w_{ji})} \tag{A.21}$$

The posterior probability $p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})$ is given by :

$$p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) = \frac{\pi_j \sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji}) w_{ji}}{\sum_{j=1}^{M} \pi_j (\sum_{i=1}^{K_j} p(\vec{X}_n|\vec{\varphi}_{ji}) w_{ji})} \tag{A.22}$$

Then Eq.A.21 becomes :

$$\frac{\partial \log p(\mathscr{X}|\varphi, \{w_{ji}\}, \vec{\pi})}{\partial \pi_j} = \frac{\sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})}{\pi_j} \tag{A.23}$$

So we have

$$\frac{\partial Q(\mathscr{X}, \varphi, \{w_{ji}\}, \vec{\pi}, \Lambda)}{\partial \pi_j} = \frac{\sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})}{\pi_j} - \lambda_1 = 0 \tag{A.24}$$

Thus,

$$\pi_j = \frac{1}{\lambda_1} \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) \tag{A.25}$$

We have

$$\sum_{j=1}^{M} \pi_j = \frac{1}{\lambda_1} \sum_{j=1}^{M} \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) = 1 \tag{A.26}$$

since,

$$\sum_{j=1}^{M} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) = 1 \tag{A.27}$$

then,

$$\sum_{j=1}^{M} \sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi}) = N \tag{A.28}$$

so we have :

$$\lambda_1 = N \tag{A.29}$$

so we have,

$$\pi_j = \frac{\sum_{n=1}^{N} p(j|\vec{X}_n, \varphi, \{w_{ji}\}, \vec{\pi})}{N} \tag{A.30}$$

Appendix B

B.1 Conditional Independence in the Transformed Space

We know that the posterior probability is $p(j|\vec{Y}_i,\Theta,\vec{\pi}) \propto \pi_j p(\vec{Y}_i|\theta_j)$, so every vector $\vec{Y}_i$ is assigned to its cluster j such as $j = \arg\max_j p(j|\vec{Y}_i,\Theta,\vec{\pi}) = \arg\max_j \pi_j p(\vec{Y}_i|\theta_j)$. For the GID, it is possible to compute the posterior probability by examining the form of the product in Eq.3.5 and considering every feature separately, so if we want to consider the feature D, Eq.3.5 becomes for a specific vector $\vec{Y}_i = (Y_{i1}, Y_{i2}, ..., Y_{iD})$:

$$\frac{1}{B(\alpha_{jD},\beta_{jD})} Y_{iD}^{\alpha_{jD}-1} (1+\sum_{l=1}^{D} Y_{il})^{-\beta_{jD}-\alpha_{jD}+\beta_{j(D+1)}} \times \prod_{l=1}^{D-1} \frac{1}{B(\alpha_{jl},\beta_{jl})} Y_{iD}^{\alpha_{jl}-1} (1+\sum_{k=1}^{l} Y_{ik})^{-\beta_{jl}-\alpha_{jl}+\beta_{j(l+1)}} \quad \text{(B.1)}$$

where $B(\alpha_{jl},\beta_{jl})$ is the beta function such that $B(\alpha_{jl},\beta_{jl}) = \frac{\Gamma(\alpha_{jl})\Gamma(\alpha_{jl})}{\Gamma(\alpha_{jl}+\beta_{jl})}$. As $\beta_{j(D+1)} = 0$, Eq.B.1 becomes :

$$\frac{1}{B(\alpha_{jD},\beta_{jD})} Y_{iD}^{\alpha_{jD}-1} (1+\sum_{l=1}^{D} Y_{il})^{-\beta_{jD}-\alpha_{jD}} \prod_{l=1}^{D-1} \frac{1}{B(\alpha_{jl},\beta_{jl})} Y_{iD}^{\alpha_{jl}-1} (1+\sum_{k=1}^{l} Y_{ik})^{-\beta_{jl}-\alpha_{jl}+\beta_{j(l+1)}} \quad \text{(B.2)}$$

by multiplying Eq.B.2 by the constant $(1+\sum_{l=1}^{D-1} Y_{il})^{\beta_{jD}+\alpha_{jD}-\alpha_{jD}+1} = (1+\sum_{l=1}^{D-1} Y_{il})^{\beta_{jD}+1}$, Eq.B.2 becomes proportional to :

$$\frac{1}{B(\alpha_{jD},\beta_{jD})} (\frac{Y_{iD}}{1+\sum_{l=1}^{D-1} Y_{il}})^{\alpha_{jD}-1} (1+\frac{Y_{iD}}{1+\sum_{l=1}^{D-1} Y_{il}})^{-\beta_{jD}-\alpha_{jD}}$$

$$\times \prod_{l=1}^{D-1} \frac{1}{B(\alpha_{jl},\beta_{jl})} Y_{iD}^{\alpha_{jl}-1} (1+\sum_{k=1}^{l} Y_{ik})^{-\beta_{jl}-\alpha_{jl}+\beta_{j(l+1)}} \tag{B.3}$$

We know that :

$$\frac{1}{B(\alpha_{jD},\beta_{jD})} (\frac{Y_{iD}}{1+\sum_{l=1}^{D-1} Y_{il}})^{\alpha_{jD}-1} (1+\frac{Y_{iD}}{1+\sum_{l=1}^{D-1} Y_{il}})^{-\beta_{jD}-\alpha_{jD}} = p_{iBeta}(\frac{Y_{iD}}{1+\sum_{l=1}^{D-1} Y_{il}}|\alpha_{jD},\beta_{jD}) \tag{B.4}$$

so Eq.B.2 becomes :

$$p_{iBeta}(\frac{Y_{iD}}{1+\sum_{l=1}^{D-1} Y_{il}}|\alpha_{jD},\beta_{jD}) \prod_{l=1}^{D-1} \frac{1}{B(\alpha_{jl},\beta_{jl})} Y_{iD}^{\alpha_{jl}-1} (1+\sum_{k=1}^{l} Y_{ik})^{-\beta_{jl}-\alpha_{jl}+\beta_{j(l+1)}} \tag{B.5}$$

For every remaining feature l in the product from 1 to $D-1$ we mutiply Eq.B.5 by the constant $(1+\sum_{k=1}^{l-1} Y_{ik})^{\beta_{jl}+\alpha_{jl}-\alpha_{jl}+1}(1+\sum_{k=1}^{l} Y_{ik})^{-\beta_{j(l+1)}} = (1+\sum_{k=1}^{l-1} Y_{ik})^{\beta_{jl}+1}(1+\sum_{k=1}^{l} Y_{ik})^{-\beta_{j(l+1)}}$ so Eq.B.5 will be proportional to :

$$\prod_{l=1}^{D} p_{iBeta}(\frac{Y_{il}}{1+\sum_{k=1}^{l-1} Y_{ik}}|\alpha_{jl},\beta_{jl}) = \prod_{l=1}^{D} p_{iBeta}(\frac{Y_{il}}{1+\sum_{k=1}^{l-1} Y_{ik}}|\theta_{jl}) \tag{B.6}$$

the first term of the product in Eq.B.6 is : $p_{iBeta}(Y_{i1}|\theta_{jl})$
so we finally have :

$$p(j|\vec{Y}_i) \propto \pi_j p(\vec{Y}_i|\theta_j) \propto \pi_j p_{iBeta}(Y_{i1}|\theta_{j1}) \prod_{l=2}^{D} p_{iBeta}(\frac{Y_{il}}{1+\sum_{k=1}^{l-1} Y_{ik}}|\theta_{jl}) \tag{B.7}$$

B.2 Calculation of Distances

B.2.1 Rényi divergence

The Rényi divergence is defined as follows [97]

$$R_a\Big(p(X_l|\theta_{il}), p(X_l|\theta_{jl})\Big) = \frac{1}{a-1}\log\left(\int_0^{+\infty} p(X_l|\theta_{il})^a p(X_l|\theta_{jl})^{1-a}\right) + \frac{1}{a-1}\log\left(\int_0^{+\infty} p(X_l|\theta_{jl})^a p(X_l|\theta_{il})^{1-a}\right) \tag{B.8}$$

In the case of independent probability densities and using postulate 9 in [97] we have :

$$R_a\Big(p(\vec{X}|\theta_i),p(\vec{X}|\theta_j)\Big)=R_a\Big(\prod_{l=1}^{D}p(X_l|\theta_{il}),\prod_{l=1}^{D}p(X_l|\theta_{jl})\Big)=\sum_{l=1}^{D}R_a\Big(p(X_l|\theta_{il}),p(X_l|\theta_{jl})\Big) \quad \text{(B.9)}$$

We can find an explicit form to the integrals, and we have :

$$\begin{aligned}
&\int_0^{+\infty}p(X_l|\theta_{il})^a p(X_l|\theta_{jl})^{1-a}=\left[\frac{\Gamma(\alpha_{il}+\beta_{il})}{\Gamma(\alpha_{il})\Gamma(\beta_{il})}\right]^a\times\left[\frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})}\right]^{1-a}\\
&\times\int_0^{+\infty}\left[X_l^{\alpha_{il}-1}(1+X_l)^{-(\alpha_{il}+\beta_{il})}\right]^a dX\times\int_0^{+\infty}\left[X_l^{\alpha_{jl}-1}(1+X_l)^{-(\alpha_{jl}+\beta_{jl})}\right]^{1-a}dX\\
&=\left[\frac{\Gamma(\alpha_{il}+\beta_{il})}{\Gamma(\alpha_{il})\Gamma(\beta_{il})}\right]^a\times\left[\frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})}\right]^{1-a}\\
&\times\int_0^{+\infty}\left[X_l^{a\alpha_{il}+\alpha_{jl}-a\alpha_{jl}-1}(1+X_l)^{-\left((a\alpha_{il}+\alpha_{jl}-a\alpha_{jl})+(a\beta_{il}+\beta_{jl}-a\beta_{jl})\right)}\right]dX
\end{aligned} \quad \text{(B.10)}$$

Let $\alpha=a\alpha_{il}+\alpha_{jl}-a\alpha_{jl}$ and $\beta=a\beta_{il}+\beta_{jl}-a\beta_{jl}$. We have the PDF of an inverted beta distribution that integrates to one:

$$\int_0^{+\infty}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}X_l^{\alpha-1}(1+X_l)^{-(\alpha+\beta)}dX=1; \quad \text{(B.11)}$$

which gives us:

$$\int_0^{+\infty}X_l^{\alpha-1}(1+X_l)^{-(\alpha+\beta)}dX=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \quad \text{(B.12)}$$

We substitute Eq. B.12 into Eq. B.10, then:

$$\begin{aligned}
&\int_0^{+\infty}p(X_l|\theta_{il})^a p(X_l|\theta_{jl})^{1-a}=\\
&\left[\frac{\Gamma(\alpha_{il}+\beta_{il})}{\Gamma(\alpha_{il})\Gamma(\beta_{il})}\right]^a\times\left[\frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})}\right]^{1-a}\times\frac{\Gamma(a\alpha_{il}+\alpha_{jl}-a\alpha_{jl})\Gamma(a\beta_{il}+\beta_{jl}-a\beta_{jl})}{\Gamma\big((a\alpha_{il}+\alpha_{jl}-a\alpha_{jl})+(a\beta_{il}+\beta_{jl}-a\beta_{jl})\big)}
\end{aligned} \quad \text{(B.13)}$$

To calculate the Rény divergence we use Eq.B.10, in Eq.B.8, and then we calculate Eq.B.9.

B.2.2 Symmetric Kullback-Leibler Distance (SKL)

The GID belongs to the exponential family so we can write it under the following form [150, 151]:

$$p(\vec{X}|\theta_j)=H(\vec{X})\exp(\sum_{l=1}^{s}G_l(\theta_j)T_l(\vec{X})+\Phi(\theta_j)) \quad \text{(B.14)}$$

We have,

$$p(\vec{X}|\theta_j) = \prod_{l=1}^{D} \frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} \frac{X_l^{\alpha_{jl}-1}}{(1+X_l)^{\alpha_{jl}+\beta_{jl}}}$$
$$= \exp[\sum_{l=1}^{D} \log\Gamma(\alpha_{jl}+\beta_{jl}) - \log\Gamma(\alpha_{jl}) - \log\Gamma(\beta_{jl}) + (\alpha_{jl}-1)\log(X_l) - (\alpha_{jl}+\beta_{jl})\log(1+X_l)] \tag{B.15}$$

We set S=2D

$$T_l(\vec{X}) = \frac{\log(X_l)}{\log(1+X_l)} \quad l = 1..D \tag{B.16}$$
$$T_{D+l}(\vec{X}) = \frac{1}{\log(1+X_l)} \tag{B.17}$$

$$G_l(\theta_j) = \alpha_{jl} \quad l = 1..D \tag{B.18}$$
$$G_{D+l}(\theta_j) = \beta_{jl} \quad l = 1..D \tag{B.19}$$

$$H(\vec{X}) = \exp(-\sum_{l=1}^{D} \log(X_d)) \tag{B.20}$$

$$\Phi(\theta_j) = \sum_{l=1}^{D} \log\Gamma(\alpha_{jl}+\beta_{jl}) - \log\Gamma(\alpha_{jl}) - \log\Gamma(\beta_{jl}) \tag{B.21}$$

The KL divergence can be defined as [152] :

$$KL\Big(p(X|\theta_j), p(X|\theta_i)\Big) = \Phi(\theta_j) - \Phi(\theta_i) + (G(\theta_j) - G(\theta_i))E_{\theta_j}[T(X)] \tag{B.22}$$

we have :

$$E_{\theta_j}[\frac{\log(X_l)}{\log(1+X_l)}] = -\frac{\partial\Phi(\theta_j)}{\alpha_{jl}} = \Psi(\alpha_{jl}) - \Psi(\alpha_{jl}+\beta_{jl}) \quad l = 1...D \tag{B.23}$$

$$E_{\theta_j}[\frac{1}{\log(1+X_l)}] = -\frac{\partial\Phi(\theta_j)}{\beta_{jl}} = \Psi(\beta_{jl}) - \Psi(\alpha_{jl}+\beta_{jl}) \quad l = 1...D \tag{B.24}$$

The KL divergence can be computed as follows

$$KL\Big(p(X|\theta_j), p(X|\theta_i)\Big) =$$

$$\sum_{l=1}^{D} \log\Gamma(\alpha_{jl}+\beta_{jl}) - \log\Gamma(\alpha_{jl}) - \log\Gamma(\beta_{jl}) - \log\Gamma(\alpha_{il}+\beta_{il}) + \log\Gamma(\alpha_{il}) + \log\Gamma(\beta_{il})$$
$$+ \sum_{l=1}^{D} (\alpha_{jl}-\alpha_{il})\Big(\Psi(\alpha_{jl}) - \Psi(\alpha_{jl}+\beta_{jl})\Big) + (\beta_{jl}-\beta_{il})\Big(\Psi(\beta_{jl}) - \Psi(\alpha_{jl}+\beta_{jl})\Big)$$
$$= \log\left[\prod_{l=1}^{D} \frac{\Gamma(\alpha_{jl}+\beta_{jl}))\Gamma(\alpha_{il})\Gamma(\beta_{il})}{\Gamma(\alpha_{il}+\beta_{il})\Gamma(\alpha_{jl})\Gamma(\beta_{jl})}\right]$$
$$+ \sum_{l=1}^{D} (\alpha_{jl}-\alpha_{il})\Big(\Psi(\alpha_{jl}) - \Psi(\alpha_{jl}+\beta_{jl})\Big) + (\beta_{jl}-\beta_{il})\Big(\Psi(\beta_{jl}) - \Psi(\alpha_{jl}+\beta_{jl})\Big) \tag{B.25}$$

so the symmetric Kullback-Leibler distance is given by

$$KL_S\Big((p(X|\theta_j), p(X|\theta_i)\Big) = KL\Big((p(X|\theta_j), p(X|\theta_i)\Big) + KL\Big((p(X|\theta_i), p(X|\theta_j)\Big)$$
$$= \sum_{l=1}^{D} (\alpha_{jl}-\alpha_{il})\Big(\Psi(\alpha_{jl}) - (\Psi(\alpha_{il}) - \Psi(\alpha_{jl}+\beta_{jl}) + \Psi(\alpha_{il}+\beta_{il})\Big)$$
$$+ (\beta_{jl}-\beta_{il})\Big(\Psi(\beta_{jl}) - (\Psi(\beta_{il}) - \Psi(\alpha_{jl}+\beta_{jl}) + \Psi(\alpha_{il}+\beta_{il})\Big) \tag{B.26}$$

B.2.3 Bhattacharyya

The Bhattacharrya distance is defined as follows:

$$B_{\frac{1}{2}}\Big(p(X|\theta_j), p(X|\theta_i)\Big) = \int_0^{+\infty} \sqrt{p(X|\theta_j)p(X|\theta_i))}dX \tag{B.27}$$

It is possible to compute the Bhattacharyya kernel in a closed form for densities that belong to the exponential family of distributions, that is, densities that can be written in the form given by Eq.B.14

$$B_{\frac{1}{2}}\Big(p(X|\theta_j), p(X|\theta_i)\Big) = \exp\left[\frac{1}{2}\Phi(\theta_j) + \frac{1}{2}\Phi(\theta_i) - \Phi\Big(\frac{1}{2}\theta_j + \frac{1}{2}\theta_i\Big)\right] \tag{B.28}$$

$$B_{\frac{1}{2}}\Big(p(X|\theta_j), p(X|\theta_i)\Big)$$
$$= \exp\Big[- \sum_{l=1}^{D} \log\Gamma\Big(\frac{1}{2}(\alpha_{jl}+\beta_{jl}+\alpha_{il}+\beta_{il})\Big) - \log\Gamma\Big(\frac{1}{2}(\alpha_{jl}+\alpha_{il})\Big) - \log\Gamma\Big(\frac{1}{2}(\beta_{jl}+\beta_{il})\Big)$$
$$+ \frac{1}{2}\sum_{l=1}^{D}\Big(\log\Gamma(\alpha_{jl}+\beta_{jl}) - \log\Gamma(\alpha_{jl}) - \log\Gamma(\beta_{jl})\Big)$$

$$
\begin{aligned}
&+\frac{1}{2}\sum_{l=1}^{D}\Big(\log\Gamma(\alpha_{il}+\beta_{il})-\log\Gamma(\alpha_{il})-\log\Gamma(\beta_{il})\Big)\Big] \\
&=\prod_{l=1}^{D}\Big[\frac{\Gamma\Big(\frac{1}{2}(\alpha_{jl}+\alpha_{il})\Big)\Gamma\Big(\frac{1}{2}(\beta_{jl}+\beta_{il})\Big)\sqrt{\Gamma(\alpha_{jl}+\beta_{jl})\Gamma(\alpha_{il}+\beta_{il})}}{\Gamma\Big(\frac{1}{2}(\alpha_{jl}+\beta_{jl}+\alpha_{il}+\beta_{il})\Big)\sqrt{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})\Gamma(\alpha_{il})\Gamma(\beta_{il})}}\Big]
\end{aligned}
\tag{B.29}
$$

Appendix C

C.1 Proof of equations

Eq.5.21 shows that the terms that are independent of $Q_s(\Theta_s)$ are absorbed into an additive constant. In order to make use of Eq.5.21 we need to calculate the logarithm of Eq.5.18 with the truncation of number of components of the GID mixture M, and the number of components of the irrelevant features to K. We also know that $\left\{ \sum_{j=1}^{M} \langle Z_{ij} \rangle \right\} = 1$, so this term will be discarded when it is factorized in the variational factors.

C.1.1 Variational solution to $Q(\vec{\phi})$

We compute the logarithm of the variational factor $Q(\phi_{il})$ as

$$\ln Q(\phi_{il}) = \phi_{il} \left\{ \langle \ln \varepsilon_{l1} \rangle + \sum_{j=1}^{M} \langle Z_{ij} \rangle \left[\mathscr{R}_{jl} + (\bar{\alpha}_{jl} - 1) \ln X_{il} - (\bar{\alpha}_{jl} + \bar{\beta}_{jl}) \ln (1 + X_{il}) \right] \right\} \quad \text{(C.1)}$$

$$+ (1 - \phi_{il}) \left\{ \langle \ln \varepsilon_{l2} \rangle + \left\{ \sum_{k=1}^{K} \langle W_{ikl} \rangle \left[\mathscr{F}_{kl} + (\bar{\sigma}_{kl} - 1) \ln X_{il} - (\bar{\sigma}_{kl} + \bar{\tau}_{kl}) \ln (1 + X_{il}) \right] \right\} \right\} + const$$

where

$$\bar{\alpha} = \langle \alpha \rangle, \quad \bar{\beta} = \langle \beta \rangle, \quad \bar{\tau} = \langle \tau \rangle \quad \text{(C.2)}$$

and

$$\mathscr{R}_{jl} = \langle \ln \frac{\Gamma(\alpha_{jl} + \beta_{jl})}{\Gamma(\alpha_{jl}) \Gamma(\beta_{jl})} \rangle, \quad \mathscr{F}_{kl} = \langle \ln \frac{\Gamma(\sigma_{kl} + \tau_{kl})}{\Gamma(\sigma_{kl}) \Gamma(\tau_{kl})} \rangle \quad \text{(C.3)}$$

The expectations in Eq.C.3 are analytically intractable, thus, we apply the second-order Taylor series expansion in order to obtain a closed-form expression such as in [132]. The approximation of $\mathscr{R}_{jl}$ and $\mathscr{F}_{kl}$ are given by $\tilde{\mathscr{R}}_{jl}$ (Eq.5.31) and $\tilde{\mathscr{F}}_{kl}$ (Eq.5.32), respectively. We substitute the

lower bound of Eqs. 5.31 and 5.32 in Eq.C.1 we obtain

$$\ln Q(\phi_{il}) = \phi_{il}\left\{ \langle \ln \varepsilon_{l1} \rangle + \sum_{j=1}^{M} \langle Z_{ij} \rangle \left[\tilde{\mathscr{R}}_{jl} + (\bar{\alpha}_{jl} - 1) \ln X_{il} - (\bar{\alpha}_{jl} + \bar{\beta}_{jl}) \ln (1 + X_{il}) \right] \right\} \quad \text{(C.4)}$$

$$+ (1 - \phi_{il}) \left\{ \langle \ln \varepsilon_{l2} \rangle + \left\{ \sum_{k=1}^{K} \langle W_{ikl} \rangle \left[\tilde{\mathscr{F}}_{kl} + (\bar{\sigma}_{kl} - 1) \ln X_{il} - (\bar{\sigma}_{kl} + \bar{\tau}_{kl}) \ln (1 + X_{il}) \right] \right\} \right\} + const$$

From Eq.C.4 we can deduce the variational solution of $Q(\vec{\phi})$ as a Bernoulli distribution such that

$$Q(\vec{\phi}) = \prod_{i=1}^{N} \prod_{l=1}^{D} f_{il}^{\phi_{il}} (1 - f_{il})^{(1-\phi_{il})} \quad \text{(C.5)}$$

where f_{il} is defined in Eq.5.26, and from the Bernoulli distribution it is straightforward to have

$$\langle \phi_{ij} \rangle = f_{ij}, \quad \langle 1 - \phi_{ij} \rangle = 1 - f_{ij} \quad \text{(C.6)}$$

C.1.2 Variational solution to Q(Z)

The logarithm of the variational factor of $Q(Z_{ij})$ is calculated as

$$\begin{aligned} \ln Q(Z_{ij}) &= Z_{ij} \left\{ \sum_{l=1}^{D} \left(\langle \phi_{il} \rangle \left[\tilde{\mathscr{R}}_{jl} + (\bar{\alpha}_{jl} - 1) \ln X_{il} - (\bar{\alpha}_{jl} + \bar{\beta}_{jl}) \ln (1 + X_{il}) \right] \right. \right. \quad \text{(C.7)} \\ &+ \left. \langle 1 - \phi_{il} \rangle \sum_{k=1}^{K} \langle w_{ikl} \rangle \left[\tilde{\mathscr{F}}_{kl} + (\bar{\sigma}_{kl} - 1) \ln X_{il} - (\bar{\sigma}_{kl} + \bar{\tau}_{kl}) \ln (1 + X_{il}) \right] \right) \\ &+ \left. \langle \ln \lambda_j \rangle + \sum_{s=1}^{j-1} \langle \ln (1 - \lambda_s) \rangle \right\} + const \end{aligned}$$

By analyzing the form of Eq.C.7 we can write $\ln Q(Z)$ as

$$\ln Q(Z) = \sum_{i=1}^{N} \sum_{j=1}^{M} Z_{ij} \ln \tilde{r}_{ij} + const \quad \text{(C.8)}$$

where $\tilde{r}_{ij}$ is defined in Eq.5.27. Thus we have

$$Q(Z) \propto \prod_{i=1}^{N} \prod_{j=1}^{M} \tilde{r}_{ij}^{Z_{ij}} \quad \text{(C.9)}$$

We know that Z_{ij} are binary and we have $\sum_{j=1}^{M} Z_{ij} = 1$, so we can normalize Eq.C.10 such that

$$Q(Z) = \prod_{i=1}^{N}\prod_{j=1}^{M} r_{ij}^{Z_{ij}} \tag{C.10}$$

where r_{ij} is defined in Eq.5.26. We can obtain $\langle Z_{ij}\rangle$ from the multinomial distribution of $Q(Z)$ such that

$$\langle Z_{ij}\rangle = r_{ij} \tag{C.11}$$

C.1.3 Variational solution to $Q(\vec{\lambda})$

The logarithm for of the variational factor $Q(\vec{\lambda})$ is given by

$$\ln Q(\lambda_j) = \ln \lambda_j \sum_{i=1}^{N} \langle Z_{ij}\rangle + \ln(1-\lambda_j)\left(\sum_{1}^{N}\sum_{s=j+1}^{M} \langle Z_{is}\rangle + \langle \Psi_j\rangle - 1\right) + const \tag{C.12}$$

Eq.C.12 has the logarithm form of the Beta distribution, by taking the exponential we obtain

$$Q(\vec{\lambda}) = \prod_{j=1}^{M} Beta(\lambda_j|\theta_j,\vartheta_j) \tag{C.13}$$

where θ_j and ϑ_j are defined in Eq.5.33. As $\vec{\gamma}$ has the Beta prior distribution, $Q(\vec{\gamma})$ can be derived in a similar way as for $Q(\vec{\lambda})$. Following the same steps we define ρ_k and ϖ_k in Eq.5.35

C.1.4 Variational solution to $Q(\vec{\psi})$

The logarithm form of $Q(\vec{\psi})$ is given by

$$\ln Q(\psi_j) = \ln \psi_j a_j + \psi_j(\langle \ln(1-\lambda_j)\rangle - b_j) + const \tag{C.14}$$

by taking the exponential in Eq.C.1.4 we obtain

$$Q(\vec{\psi}) = \prod_{j=1}^{M} \mathcal{G}(\psi_j|a_j^*,b_j^*) \tag{C.15}$$

where a_j^* and b_j^* are defined in Eq.5.34. As $\vec{\varphi}$ has the Gamma prior distribution, $Q(\vec{\varphi})$ can be derived in a similar way as for $Q(\vec{\psi})$. Following the same steps we define c_k^* and d_k^* in Eq.5.36.

C.1.5 Variational solution to $Q(W)$

The logarithm of the variational factor $Q(W_{ikl})$ is given by

$$\begin{aligned}\ln Q(W_{ikl}) &= W_{ikl}\Bigg\{\langle 1-\phi_{il}\rangle\Bigg(\tilde{\mathscr{F}}_{kl}+(\bar{\sigma}_{kl}-1)\ln X_{il}-(\bar{\sigma}_{kl}+\bar{\tau}_{kl})\ln(1+X_{il})\Bigg) \\ &+ \langle\ln\gamma_k\rangle+\sum_{s=1}^{k-1}\langle\ln(1-\gamma_s)\rangle\Bigg\}+const\end{aligned} \tag{C.16}$$

By taking the exponential in Eq.C.16 we obtain

$$Q(W)=\prod_{i=1}^{N}\prod_{k=1}^{K}\prod_{l=1}^{D}m_{ikl}^{W_{ikl}} \tag{C.17}$$

where m_{ikl} is given by Eq.5.26.

C.1.6 Variational solution to $Q(\vec{\varepsilon})$

The logarithm of the variational factor $Q(\vec{\varepsilon}_l)$ is given by

$$\ln Q(\vec{\varepsilon}_l)=\ln\varepsilon_{l_1}\Bigg(\sum_{i=1}^{N}\langle\phi_{il}\rangle+\xi_1-1\Bigg)+\ln\varepsilon_{l_2}\Bigg(\sum_{i=1}^{N}\langle 1-\phi_{il}\rangle+\xi_2-1\Bigg)+const \tag{C.18}$$

Eq.C.18 has a logarithmic form similar to the logarithm form of a Dirichlet distribution. The variational solution to $Q(\vec{\varepsilon}_l)$ can be obtained by

$$Q(\vec{\varepsilon}_l)=\prod_{l=1}^{D}Dir(\vec{\varepsilon}_l|\vec{\xi}^{\,*}) \tag{C.19}$$

where $\vec{\xi}^{\,*}=(\xi_1^*,\xi_2^*)$ in Eq.5.37.

C.1.7 Variational solution to $Q(\vec{\alpha})$, $Q(\vec{\beta})$, $Q(\vec{\sigma})$, and $Q(\vec{\tau})$

The logarithm of the variational factor $Q(\alpha_{jl})$ can be calculated as

$$\begin{aligned}\ln Q(\alpha_{jl}) &= \langle\ln p(\mathscr{X},\Theta)\rangle_{\Theta\neq\alpha_{jl}} \\ &= \sum_{i=1}^{N}\langle Z_{ij}\rangle\langle\phi_{il}\rangle\left[\mathscr{D}(\alpha_{jl})+\alpha_{jl}\ln\frac{X_{il}}{1+X_{il}}\right]+(u_{jl}-1)\ln\alpha_{jl}-v_{jl}\alpha_{jl}+const\end{aligned} \tag{C.20}$$

where

$$\mathscr{D} = \langle \ln \frac{\Gamma(\alpha_{jl}+\beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} \rangle_{\beta_{jl}} \tag{C.21}$$

using a non-linear approximation as proposed in [132] we have

$$\mathscr{D}(\alpha) \geq \ln \alpha \Big\{ \psi(\bar{\alpha}+\bar{\beta}) - \psi(\bar{\alpha}) + \bar{\beta}\,\psi'(\bar{\alpha}+\bar{\beta}))(\langle \ln \beta \rangle - \ln \bar{\beta}) \Big\} \bar{\alpha} \tag{C.22}$$

We substitute the lower bound in Eq.C.22 into the Eq.C.20 we have

$$\begin{aligned} \ln Q(\alpha_{jl}) &= \ln \alpha_{jl} \Big\{ \sum_{i=1}^{N} \langle Z_{ij} \rangle \langle \phi_{il} \rangle \big[\psi(\bar{\alpha}_{jl}+\bar{\beta}_{jl}) - \psi(\bar{\alpha}_{jl}) \\ &+ \bar{\beta}_{jl}\psi'(\bar{\alpha}_{jl}+\bar{\beta}_{jl}))(\langle \ln \beta_{jl} \rangle - \ln \bar{\beta}_{jl}) \big] \bar{\alpha}_{jl} + u_{jl} - 1 \Big\} \\ &+ \alpha_{jl} \Big\{ \sum_{i=1}^{N} \langle Z_{ij} \rangle \langle \phi_{il} \rangle \ln \frac{X_{il}}{1+X_{il}} - v_{jl} \Big\} + const \end{aligned} \tag{C.23}$$

Eq.C.23 has the form of a Gamma distribution. By taking it to the exponential we obtain

$$Q(\vec{\alpha}) = \prod_{j=1}^{M} \prod_{l=1}^{D} \mathscr{G}(\alpha_{jl} | u^*_{jl}, v^*_{jl}) \tag{C.24}$$

The hyperparameters u^*_{jl} and v^*_{jl} can be estimated by Eq.5.38 and Eq.5.39, respectively. Since $\vec{\beta}$, $\vec{\sigma}$ and $\vec{\tau}$ have the Gamma prior, we obtain the variational solutions to $Q(\vec{\beta})$, $Q(\vec{\sigma})$, and $Q(\vec{\tau})$ in the same way as for $Q(\vec{\alpha})$.