Probabilistic graphical models : theory and applications to network diagnosis

Amine Echraibi

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Amine ECHRAIBI

Probabilistic Graphical Models : Theory and Application to Network Diagnosis

Thèse présentée et soutenue à Brest le 06/12/2021
Unité de recherche : Lab-STICC
Thèse N° : 2021IMTA0283

Rapporteurs avant soutenance :

Audrey GIREMUS	Professeure Université de Bordeaux
Marc LELARGE	Directeur de Recherches INRIA - École Normale Supérieure

Composition du Jury :

Président :	Eric FABRE	Directeur de Recherches IRISA/INRIA Rennes
Examinateurs :	Frédéric PENNERATH	Maître de Conférences CentraleSupélec
	Pierre BORGNAT	Directeur de Recherches CNRS - École Normale Supérieure Lyon
Dir. de thèse :	Sandrine VATON	Professeure IMT Atlantique
Encadrants :	Joachim FLOCON-CHOLET	Ingénieur de recherche Orange Labs
	Stéphane GOSSELIN	Ingénieur de recherche Orange Labs
Rapporteurs :	Audrey GIREMUS	Professeure Université de Bordeaux
	Marc LELARGE	Directeur de Recherches INRIA - École Normale Supérieure

Dedicated to my beloved mother for all the sacrifices she made for me, and the support she has shown me, without which I would not be where I am today.

SUMMARY

For any Internet service provider or network operator, it is crucial to quickly and efficiently diagnose the problems that occur on the network. The benefits of a good fault diagnosis system are mainly to minimize the costs of network and service operations and to enhance the customer's quality of experience. One major challenge for any diagnosis system concerns the discovery of new faults, that are unknown to the current version of the diagnosis system. The exploratory process for finding new faults can prove to be expensive and time consuming for internet service providers.

In this thesis, we explore an alternative approach based on learning methods, in order to build learning-based diagnosis systems. Our study explores Probabilistic Graphical Models that are capable of clustering patterns of faults in an unsupervised and a semi-supervised manner. We demonstrate the efficiency of our models on real use-cases of large scale data, extracted from Fiber-to-the Home (FTTH) services based on Gigabit-capable Passive Optical Networks.

Our contributions are as follows:

- We propose the ***Infinite Categorical Mixture Model*** (Echraibi et al., 2020b) for unsupervised clustering of faults from categorical network data, with simultaneous learning of the number of clusters. We show that our model achieves competitive results, and allows for the identification and interpretation of clusters of faults.
- We propose the ***Infinite Mixed-type semi-supervised Mixture Model***, extending the categorical model to treat continuous and categorical data. We also introduce semi-supervision in the form of labels provided by domain experts to guide the clustering process towards relevant fault clusters.
- We propose the ***Deep Infinite Mixture Model*** for semi-supervised discovery of faults from noisy large scale data of GPON-FTTH networks. The model leverages the power of deep neural networks to learn hidden representations relevant to the fault diagnosis task, and the exploration and clustering capability of infinite mixtures to identify clusters of faults.
- We introduce a novel approach to train deep probabilistic graphical models, namely ***Generalized Stochastic Backpropagation*** (Echraibi et

al., 2020a). Our approach gives a theoretical framework for computing low variance estimates of gradients often arising in probabilistic graphical models.

- We introduce a general end-to-end infinite deep probabilistic mixture model, entitled the ***Dirichlet Process Deep Latent Mixture Model*** (Echraibi et al., 2020c). We demonstrate that the model is capable of learning representative hidden features of clusters of faults, while simultaneously clustering them without a prior knowledge on the number of clusters.
- Finally, we demonstrate that shallow infinite mixture models can prove interpretable, in the sense where identified clusters can be traced to network variables (or root-causes). We also give a general method of interpreting clustering results of the deep models, using standard approaches such as decision trees.

PUBLICATIONS

The ideas and content of this thesis have been published in the following publications:

- **Amine Echraibi**, Joachim Flocon-Cholet, Stéphane Gosselin and Sandrine Vaton (2019). "Bayesian Mixture Models For Semi-Supervised Clustering." In: *Preprint*.
- **Amine Echraibi**, Joachim Flocon-Cholet, Stéphane Gosselin and Sandrine Vaton (2020). "An Infinite Multivariate Categorical Mixture Model for Self-Diagnosis of Telecommunication Networks." In: *23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*.
- **Amine Echraibi**, Joachim Flocon-Cholet, Stéphane Gosselin and Sandrine Vaton (2020). "On the Variational Posterior of Dirichlet Process Deep Latent Gaussian Mixture Models." In: *International Conference on Machine Learning 2020 INNF workshop (ICML)*.
- **Amine Echraibi**, Joachim Flocon-Cholet, Stéphane Gosselin and Sandrine Vaton (2020). "Generalized Stochastic Backpropagation." In: *Neural Information Processing 2020 Beyond Backprop workshop (NeurIPs)*.
- **Amine Echraibi**, Joachim Flocon-Cholet, Stéphane Gosselin and Sandrine Vaton (2021). "Stochastic Backpropagation through Fourier Transforms." In: *2021 29th European Signal Processing Conference (EUSIPCO)*.
- **Amine Echraibi**, Joachim Flocon-Cholet, Stéphane Gosselin and Sandrine Vaton (2021). "Deep Infinite Mixture Models for Fault Discovery in GPON-FTTH Networks." In: *IEEE Access 9, pp. 90488–90499. DOI: 10.1109/ACCESS.2021.3091328*.
- **Amine Echraibi**, Joachim Flocon-Cholet, Stéphane Gosselin (2021). "Classification de pannes inconnues dans un système de communications électroniques." In: *Patent INPI FR2102748 filed 19th of March 2021.*

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1.1 NETWORK FAILURE DIAGNOSIS

Identifying faults in networks is of crucial importance to any Internet service provider. An accurate diagnosis system capable of identifying faults, can improve the customer's experience, decrease the down time of network operations, and more importantly reduce the costs of an Internet service provider in terms of interventions to repair the network.

A considerable amount of research has been done to construct and maintain operational fault diagnosis systems in all domains, ranging from network operations to industrial plants and applications (Mosterman and Biswas, 1999; Tembo et al., 2017). However, the research scope for solving the diagnosis task is often specific to the domain, and can even vary in definition.

In the network management community, the diagnosis task is divided into three sub-tasks. The first is the detection step, which can be done proactively, it aims at deciding if a customer experiences a problem and if further investigations are required. The second is the isolation step, where the goal is to identify the root cause of the problem given the current technical status or available data. Third, the mitigation step covers all the actions required to fix the problem (Ayoubi et al., 2018; Boutaba et al., 2018; Cherrared et al., 2019; Steinder and Sethi, 2004).

The first step, consisting in identifying if a fault occurred on the network, is often associated with detecting anomalous behavior. The methods often adopted for this task are *Anomaly Detection* approaches (Chandola, Banerjee, and Kumar, 2009a). For the specific application that we study across the thesis, i.e. fault diagnosis of broadband access services, these methods are not considered, since fault detection is a direct outcome of a customer call to the Internet service provider's hotline.

Thus, we are rather interested in the isolation step, where we know that a fault has occurred on the network, and we must identify the cause or

the nature of the fault. The scope of our application is fault diagnosis of Fiber-to-the Home (FTTH) services based on Gigabit-capable Passive Optical Networks (GPON) (Chanclou et al., 2006; *Gigabit-capable passive optical networks (G-PON): ONT management and control interface specification* 2008; *Gigabit-capable passive optical networks (GPON): General characteristics* 2008).

With the various modern applications of machine learning, the objective is to build models capable of classifying faults from GPON-FTTH network data derived from operations of ***Orange*** networks. Furthermore, the challenge is to build adaptable systems. In the sense that, if a new fault appears on the network (due to changes or other phenomena) the model is capable of clustering the new fault, given that a signature of the fault exists in the data encountered.

1.2 NETWORK DIAGNOSIS AS A MACHINE LEARNING PROBLEM

In order to introduce the network diagnosis task as a machine learning problem, we need to first discuss the current status of diagnosis for an Internet service provider such as ***Orange***, and for the research community in general. Currently, in most real-world operational diagnosis systems, the preferred method is *Expert Systems*. Specifically for ***Orange***, the expert system is called *DELC (Diagnostic Expert de la Ligne Client) or expert diagnostic for the customer's line*.

An *Expert System* is a rule-based system, where given a certain deterministic condition on the network variables an alarm is triggered to alert if a fault happened. This has its advantages and its drawbacks. The first advantage is that these rules can be constructed directly given enough expert knowledge, and they are interpretable and clear to other network or domain experts. However, the drawbacks are numerous. The first if the network is large the expertise and time required to build the expert system are enormous, maintaining it is even more challenging and requires tremendous resources. Second, if a new fault occurs due to changes in the network, the system has to be revisited, which can prove even more daunting. Third, since the rules are deterministic, the expert system cannot always conclude when some variables are missing.

The first approach that has been considerably researched for the diagnosis task are *Bayesian Networks* (Koller and Friedman, 2009; Tembo et al., 2017). A *Bayesian Network* is a probabilistic model of the network variables with dependencies. The model can be constructed by an expert, the dependencies represent causal relations between variables of the network. The objective of

a bayesian network is to model the propagation of faults through variables of the network, thus by reversing the process, we can identify the root cause. The advantage of these models is that they can be learned from data (conditional probabilities estimated from data). However, as expert systems, bayesian networks are highly specific to the application: if a new fault or a change occurs on the real network, the bayesian network has to be revisited which is often an expensive task.

Classical machine learning methods have also been explored for the diagnosis task. One approach is to consider the traditional supervised learning scheme, where a classifier is trained using data labeled according to the different known types of failures (Adda, Qader, and Al-Kasassbeh, 2017; Baras et al., 1997; Chen et al., 2002; Chen et al., 2004). These classifiers are numerous: they can range from simple models such as Naive-Bayes classifiers (Zhang, 2005), to Support-Vector Machines (SVMs) (Cortes and Vapnik, 1995), and even Neural Networks (Rumelhart, Hinton, and Williams, 1985).

Although these methods are appealing, they require a retraining of the classification model with new features and new labeled data for new types of faults, which generally are not available, especially when the fault is very recent. With the Bayesian network framework, taking into account new types of failures, new data or new equipment would mean updating the Bayesian graph by adding and/or removing nodes and edges as well as changing conditional probabilities of the edges. This would require a high network expertise and may be unrealistic if the network is too complex. Though it is possible to derive the Bayesian graph from data (Koller and Friedman, 2009), it becomes unfeasible in practice when dealing with large datasets composed of hundreds or thousands of features.

Furthermore, in order to discover new faults, one needs to perform some data exploration and to find clusters corresponding to new types of faults. The current standard approach is an expert investigation of the data "by hand" in order to identify these clusters. Unsupervised clustering methods can be used to reach good performance. However, using such approaches is often dependent on model assumptions that could hurt the results, such as distance metrics in high dimensional spaces (Aggarwal, Hinneburg, and Keim, 2001; Kriegel, Kröger, and Zimek, 2009). In addition, the clusters may be relevant from a technical point of view (e.g. grouping individuals because they use the same type of equipment), but this type of clustering does not provide any new information that could be used to highlight a new fault. The obvious way to bypass this issue is to resort to cumbersome feature selection and preprocessing of the data by hand, which requires deep expert knowledge of the environment that is constantly changing, and can be expensive time-wise.

In this thesis, we present a general framework based on *Probabilistic Graphical Models* to learn and identify clusters of faults. We argue for a specific range of probabilistic graphical models namely, Dirichlet Process Mixtures or Dirichlet Process Deep Mixtures. These models we propose are capable of clustering faults without knowing the number of clusters (or faults), they can identify hidden features relevant to the diagnosis task, and more importantly, they include expert knowledge in terms of supervision (or semi-supervision) in the form of labels in order to guide and improve fault identification from data.

1.3 STRUCTURE AND RESEARCH QUESTIONS

This thesis is divided into two parts: in Part i, we introduce shallow probabilistic graphical models. Specifically, Infinite Mixture Models that are capable of identifying clusters of faults from network data, in an unsupervised and a semi-supervised way. In Part ii, we evolve these models to include deep hidden representations. The goal of the hidden representations is to construct features relevant for the diagnosis task. These features are constructed to deal with noisy and complex large scale data. Combined with the infinite mixtures, we can use these representations to reach better results on such datasets.

In order to give a guide of the manuscript, the contributions made in Part i are an attempt to answer the following research questions:

Question 1: *Can we develop models capable of identifying clusters adapted to network data, specifically, categorical data, without knowing a priori the number of clusters?*

Our contribution to answer this question is the *Infinite Categorical Mixture Model* (Echraibi et al., 2020b). The model is introduced in chapter 3. We extend the definition of Dirichlet Process mixtures to the categorical variable case and we show how to use mean-field variational inference to learn the model efficiently. We show that our model is capable of identifying clusters of faults without knowing a priori the number of clusters and we demonstrate how the model assumptions lead to better performance in terms of accuracy.

Question 2: *Can we extend the Dirichlet process mixtures to deal with continuous and categorical variables simultaneously, and also take into account expert knowledge in terms of supervision?*

In this contribution, we extend the infinite categorical mixture model to a *Mixed-type model*, that can deal with continuous and categorical variables. The model also takes into account semi-supervision from experts in the form of labels. We evaluate the model on network data and we show its

competitiveness with state-of-the art baselines. This work is presented in chapter 4 and is based partially on work from (Echraibi et al., 2019).

We have introduced shallow probabilistic graphical models, in the form of infinite mixture models, that could be applied to mixed-type data. A problem remains, these models cannot create complex combinations of data in order to discover clusters. In other words, feature creation is not possible. However, in most real-world applications, data is noisy and of large dimensionality. Thus, feature creation is necessary, in Part ii, we develop models and methods in order to accomplish this goal, and the main research question addressed are the following:

Question 3: *Can we construct highly relevant representations or features for the diagnosis task, in order to improve performance?*

In order to answer this question, in chapter 5, we combine deep learning representations and infinite mixture models to perform fault discovery in a partially labeled large scale dataset. Our model is called *Deep Infinite Mixtures* (Echraibi et al., 2021b). We train a deep neural network to recognize labeled faults and build low dimensional hidden representations relevant to the diagnosis task. Then, an infinite Gaussian mixture model is applied on the unlabeled part in order to recognize clusters of faults based on the hidden representation. We evaluate the model on a large scale real-world GPON-FTTH dataset, we show that the model can identify classes of known faults with high accuracy, and that the hidden representation on the unlabeled faults lead to recognizable clusters.

Deep learning and infinite mixtures live in two different universes of methods. Their combination is not natural in the sense we do not have an end-to-end global model, capable of learning features and clustering. Thus, the following question naturally emerges:

Question 4: *Can we construct an end-to-end deep probabilistic model capable of learning hidden representations and of performing clustering?*

It turns out this is possible through deep generative models, which are a class of probabilistic graphical models. The problem of deep generative models is the training process that involves computing estimates of gradients that are often of large variance. In our contributions *Generalized Stochastic Backpropagation*, and *Stochastic Backpropagation through Fourier Transforms* (Echraibi et al., 2020a, 2021a), we present a method to derive low-variance estimates of these gradients. This work is presented in chapter 6.

In chapter 7, we use generalized stochastic backpropagation to train our proposed end-to-end *Dirichlet Process Deep Latent Mixture Model* (Echraibi et al., 2020c). We evaluate our model on open datasets and on real-world large scale GPON-FTTH data for the fault diagnosis and discovery task.

Finally, in chapter 8, we discuss the interpretability of our models. We show that shallow models are naturally interpretable. Deep models, however, are more challenging in terms of interpretability. We discuss how to use decision trees to retrace the decision leading to a fault cluster, making it understandable for a domain expert.

The previous chapters discussed constitute the body of work done during the thesis. In chapter 2, we additionally introduce some background notions necessary for reading this manuscript. Following the body of the manuscript, we give a conclusion and directions of interesting future research in chapter 9. In the following figure we give a guide for reading this thesis, specifically the dependencies between the chapters. The background chapter is necessary for the comprehension of practically all chapters. The variational inference appendix is used in all chapters except chapter 6. Chapter 6 is a theoretical contribution of the thesis, that stands on its own, and is used in chapter 7.

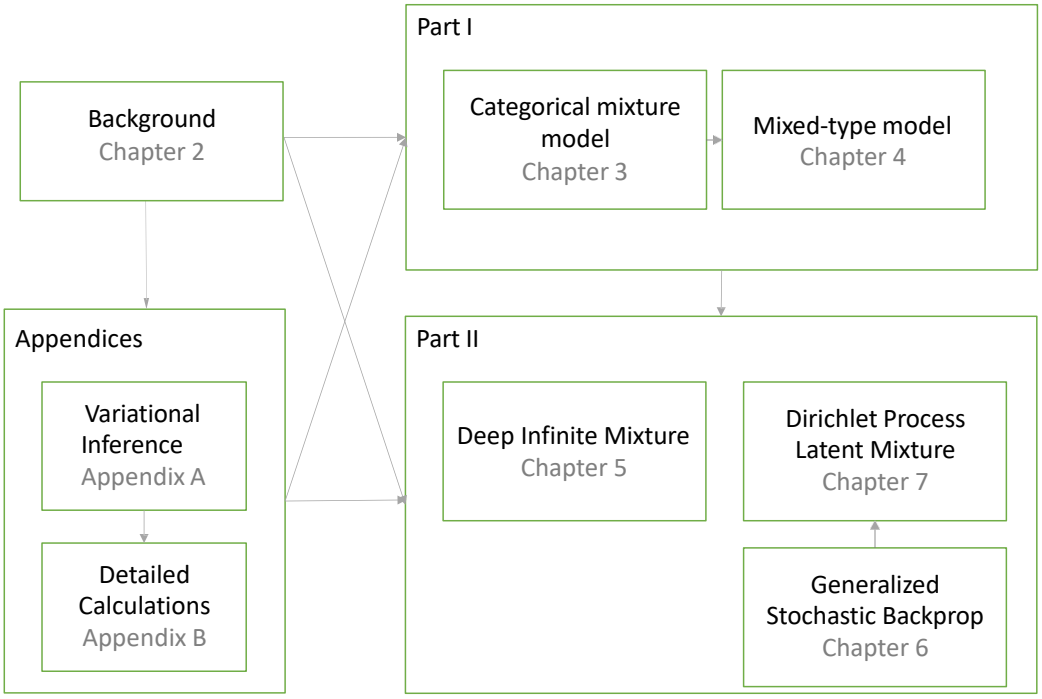

Figure 1: Summary of the structure of the document

2

BACKGROUND

In this chapter we present background subjects relevant to the overall manuscript, as well as the mathematical notations used throughout the manuscript. In the case where some additional materiel is relevant, it will be introduced in the relevant chapter. The current chapter is organized as follows: in section 2.1 we introduce important notations. In section 2.2 we present Probabilistic Graphical Models on which the whole manuscript is based. In sections 2.3 and 2.4, we present Markov Chain Monte Carlo and Variational Inference methods, required for inference and learning in probabilistic graphical models. In section 2.5, we give an introduction of deep learning methods required for the second part of the manuscript.

2.1 NOTATIONS

The following table gathers a non exhaustive list of mathematical notations and conventions which will be used throughout the manuscript chapters:

Notation	**Definition**
x	We denote by a lowercase letter a scalar or a scalar valued random variable.
$\mathbf{x}$	We denote by a bold lowercase letter a vector or a vector valued random variable.
$\mathbf{X}$	We denote by an uppercase bold letter a tensor or a tensor valued random variable.
$\mathbf{x}_i$	denotes the i^{th} scalar of vector $\mathbf{x}$.
$\mathbf{x}_{-i}$	denotes the vector $\mathbf{x}$ without the i^{th} entry $\mathbf{x}_i$.
$f_{\boldsymbol{\theta}}(\cdot)$ or $f(\cdot;\boldsymbol{\theta})$	denotes a function f dependent on parameters $\boldsymbol{\theta}$.
$f(\mathbf{x})$	denotes element-wise application of a function f on $\mathbf{x}$.

Notation	Definition
$\mathbb{1}[\text{condition}]$	represents the indicator function, that is equal to 1 if condition is true, 0 otherwise.
$p(\cdot)$	denotes a probability distribution over a continuous random variable $\mathbf{x}$, or a probability mass function if $\mathbf{x}$ is discrete.
$p(\cdot\|\mathbf{x})$	denotes a probability distribution conditionally to the value of the random variable $\mathbf{x}$.
$\mathbf{x} \sim p(\cdot)$	indicates that the random variable $\mathbf{x}$ follows the probability distribution $p(\cdot)$.
$\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	denotes the multivariate normal distribution of mean $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$, defined by: $$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})},$$ where d represents the dimension of $\mathbf{x}$ and $\det(\cdot)$ denotes the determinant operator.
$\text{Beta}(\cdot; \gamma_1, \gamma_2)$	denotes the beta distribution of parameters γ_1 and γ_2, defined by: $$\text{Beta}(\beta; \gamma_1, \gamma_2) = \frac{\Gamma(\gamma_1 + \gamma_2)}{\Gamma(\gamma_1)\Gamma(\gamma_2)} \beta^{\gamma_1 - 1}(1-\beta)^{\gamma_2 - 1},$$ where Γ represents the gamma function.
$\text{Cat}(\cdot; \boldsymbol{\pi})$	denotes the categorical distribution of parameters $\boldsymbol{\pi}$, defined by: $$\text{Cat}(z; \boldsymbol{\pi}) = \prod_{k=1}^{K} \boldsymbol{\pi}_k^{\mathbb{1}[z=k]} \quad \text{s.t.} \quad \sum_{k=1}^{K} \boldsymbol{\pi}_k = 1,$$ where z takes values in the discrete set {1, ..., K}.
$\text{Dir}(\cdot; \boldsymbol{\alpha}, K)$	represents the Dirichlet distribution of parameters $\boldsymbol{\alpha}$ and K, defined by: $$\text{Dir}(\mathbf{x}; \boldsymbol{\alpha}, K) = \frac{1}{\text{B}(\boldsymbol{\alpha})} \prod_{k=1}^{K} \mathbf{x}_k^{\boldsymbol{\alpha}_k - 1} \quad \text{s.t.} \quad \sum_{k=1}^{K} \mathbf{x}_k = 1$$ where $\text{B}(\cdot)$ represents the multivariate beta function.
$\nabla_{\mathbf{x}} f$	represents the gradient of a scalar function f w.r.t. to the variable $\mathbf{x}$, defined by: $$[\nabla_{\mathbf{x}} f]_i = \frac{\partial f}{\partial x_i}.$$

Notation	Definition
$\mathbb{E}_{\mathbf{x}\sim p}$	Expectation taken w.r.t. the random variable $\mathbf{x}$ following the distribution $p(\cdot)$.
$\mathbb{H}[p]$ or $\mathbb{H}[p(\mathbf{x})]$	represents the entropy of probability distribution $p(\cdot)$ defined by: $$\mathbb{H}[p] = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}$$
$\mathbb{D}_{KL}[q\|\|p]$ or $\mathbb{D}_{KL}[q(\mathbf{x})\|\|p(\mathbf{x})]$	represents the Kullback-Leibler divergence between two probability distributions $q(\cdot)$ and $p(\cdot)$ defined by: $$\mathbb{D}_{KL}[q\|\|p] = \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x}$$.

The previous notations are not exhaustive, if necessary additional notations will be introduced in the relevant chapters.

2.2 PROBABILISTIC GRAPHICAL MODELS

Probabilistic Graphical Models (PGMs) are a class of models aimed at specifying the dependency structure over random variables of the problem treated. The advantages of such representations are numerous, the most important of which are:

- The ability to visualize the implications and properties of the model (dependency between interacting random variables).
- PGMs motivate the design and discovery of new models, generalizing previous models.
- They provide an approach to manipulate complex probabilistic computations through graphs.

PGMs are a central topic throughout the thesis. For a brief introduction, consider two vector random variables $\mathbf{h}$ and $\mathbf{x}$, where the $\mathbf{h}$ random variable is hidden or unobserved (i.e. no samples of data exist for such variables), and the $\mathbf{x}$ random variable is observed, thus samples of this random variable are available. Let us further assume that the random variable $\mathbf{x}$ depends on the random variable $\mathbf{h}$. This provides us with a probabilistic graphical model specified by the previous assumptions. One can visualize this model by

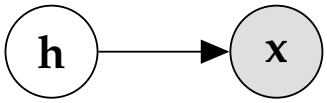

Figure 2: An example of a PGM.

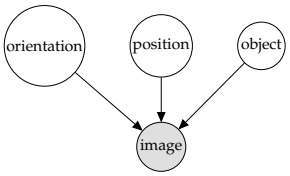

Figure 3: A generative model of an image.

sketching the dependencies as presented in figure 2. Observable variables are denoted by gray circles, hidden variables are denoted by clear circles. Arrows or directed edges between the circles represent causality or conditional dependency, for example, in our case $\mathbf{x}$ depends on $\mathbf{h}$.

Probabilistic graphical models provide us with a clear modeling paradigm for interacting parts in an environment, thus making them highly interpretable and subject to inspection. In addition, PGMs can be seen as generative models, where the observable quantities are generated based on hidden or unobserved factors (hidden variables). This approach is very useful for modeling causality in real-world applications. A classical example is given in figure 3, where an image is considered as an observable random variable. One can represent the process by which an image is generated based on the orientation and position of objects in it, by supposing that the object, position, and orientation are hidden factors on which the image depends.

Maximum Likelihood (ML) based learning

Let us reconsider the previous toy model of figure 2. Creating a model in the sense of PGMs is the task of specifying a joint probability distribution over the random variables $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{h})$ often parameterized by some learnable parameters $\boldsymbol{\theta}$. Given N samples of the observed random variable $\mathbf{x}$, independent and identically distributed (i.i.d.): $\mathbf{x}^{(1:N)} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$, the objective is to maximize the likelihood w.r.t. the parameters $\boldsymbol{\theta}$ defined by:

$$l(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(n)}). \tag{2.1}$$

Maximizing the function l is done mainly through the computation of the gradient $\nabla_{\boldsymbol{\theta}} l$, either by setting it to zero and deriving a fixed point equation, or through gradient ascent algorithms. However, computing the gradient requires marginalization over the hidden variables $\mathbf{h}$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{h}) d\mathbf{h}. \tag{2.2}$$

Exact marginalization is often intractable, except for simple models. We will revisit different approaches to bypass this issue in later sections.

Inference of hidden variables

In our modeling assumptions, hidden variables are generally what interests us, they often represent an interesting feature of the problem that has to be inferred from the data, for example, a class, or the position of an object as shown in figure 3. The inference process requires the computation of the posterior of the latent or hidden variables knowing the observed random variables. Namely, it requires the computation of the quantity:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\mathbf{h}|\mathbf{x}) &= \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{h})}{p_{\boldsymbol{\theta}}(\mathbf{x})} \qquad \text{(Bayes rule)} \\ &= \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{h})}{\int p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{h}) d\mathbf{h}} \end{aligned} \tag{2.3}$$

thus facing the same issue of marginalizing over the hidden variables.

For further details on probabilistic graphical models, we suggest the following books by Bishop (2006), Koller and Friedman (2009), and Murphy (2012).

2.3 MARKOV CHAIN MONTE CARLO METHODS

As discussed in section 2.2, the main issue with the learning and inference of probabilistic graphical models, is the exact marginalization of equation (2.2). Fortunately, Markov Chain Monte Carlo (MCMC) methods provide us with a class of approaches to bypass the issue of exact marginalization through sampling based approximations (Hastings, 1970). The main two problems solved by such approaches are the following:

- **Problem 1**: generating S samples $\{\mathbf{x}^{(s)}\}_{s=1}^{S}$ from a given probability distribution $p(\mathbf{x}) = p(x_1, ..., x_d)$.
- **Problem 2**: The estimation of quantities of the form:

$$\mathbb{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] = \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \tag{2.4}$$

Often what interests us is problem 2, however, by solving problem 1, problem 2 becomes trivial. Indeed, for the time being let us suppose that problem 1 is solved, i.e. we have obtained S samples $\{\mathbf{x}^{(s)}\}_{s=1}^{S}$ drawn from the probability distribution $p(\cdot)$. An estimator of the expectation is given by:

$$\hat{I} = \frac{1}{S} \sum_{s=1}^{S} f(\mathbf{x}^{(s)}). \tag{2.5}$$

Notice that if $\{\mathbf{x}^{(s)}\}_{s=1}^{S}$ are generated from the probability distribution $p(\cdot)$ the expectation of $\hat{I}$ is equal to $\mathbb{E}_{\mathbf{x}\sim p}[f(\mathbf{x})]$. Furthermore, the variance of the

estimator is proportional to S^{-1}, thus the more samples drawn the more accurate the estimator becomes.

The theory of Markov chains provides us with a general formalism in order to evaluate sampling algorithms, in terms of consistency of the samples w.r.t. the desired sampled distribution (Neal, 1993). Starting from an initial probability distribution denoted by $\pi^{(0)}(\mathbf{x})$ the Markov chain simulator constructs a probability distribution at time t: $\pi^{(t)}(\mathbf{x})$. The objective is that at convergence: $t \to \infty$, $\pi^{(t)}(\mathbf{x})$ converges to the target distribution $p(\mathbf{x})$. The Markov chain simulator is defined by the initial probability distribution $\pi^{(0)}(\mathbf{x})$, and the transition kernel $T(\mathbf{x}', \mathbf{x})$ such that:

$$\pi^{(t+1)}(\mathbf{x}') = \int T(\mathbf{x}', \mathbf{x})\pi^{(t)}(\mathbf{x})d\mathbf{x}. \tag{2.6}$$

In order to guarantee uniqueness of convergence to the target distribution, some required properties have to be respected. Namely, the invariance property, i.e. the target distribution has to be invariant under the kernel:

$$p(\mathbf{x}') = \int T(\mathbf{x}', \mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{2.7}$$

and the ergodicity property, i.e. the Markov chain converges to the target distribution irrespective of the choice of the initial distribution $\pi^{(0)}(\mathbf{x})$.

Throughout the manuscript we will focus on the Gibbs sampling algorithm (Geman and Geman, 1984), which is an instance of Markov Chain Monte Carlo algorithms, where the transition kernel is given by:

$$T(\mathbf{x}', \mathbf{x}) = \prod_{j=1}^{d} p(x'_j | \mathbf{x}_{-j}). \tag{2.8}$$

The key advantage of Gibbs sampling is in the fact that it utilizes the conditional distributions $p(x_j|\mathbf{x}_{-j})$, which are often easy to derive in closed form. Algorithm 1 summarizes the procedure of Gibbs sampling.

A complete review of Markov Chain Monte Carlo methods is beyond the scope of the thesis, however for more details on MCMC methods we highly recommend the book by MacKay, 2003 and the seminal papers by Hastings (1970) and Neal (1993).

2.4 VARIATIONAL INFERENCE

Variational inference is a class of approaches that provide us with an alternative way of computing or approximating the posterior of equation 2.3, without resorting to exact marginalization. Variational methods have their origin in the early work of Euler on the calculus of variations, where the

Algorithm 1 Gibbs sampling

Initialize $\mathbf{x}^{(0)}$
for $t = 1...\infty$ **do**
  sample $x_1^{(t)} \sim p(x_1|\mathbf{x}_{-1}^{(t-1)})$
  sample $x_j^{(t)} \sim p(x_j|x_1^{(t)},...,x_{j-1}^{(t)},x_{j+1}^{(t-1)},...,x_d^{(t-1)})$
  sample $x_d^{(t)} \sim p(x_d|\mathbf{x}_{-d}^{(t)})$
end for
return $\mathbf{x}^{(\infty)}$

idea is to minimize a functional (a function of a function) with respect to its argument, which is also a function. The calculus of variations is closely related to standard calculus, often we only require the computation of a functional derivative, i.e. elementary changes of the functional subject to small changes in the argument function (Feynman, Leighton, and Sands, 1964).

In the case of probabilistic graphical models, we transform the inference problem into an optimization problem of a certain functional. The functional is derived from the likelihood of equation (2.1) and often referred to as the evidence lower bound or (ELBO), a term that we will use throughout the manuscript. Let us reconsider our toy probabilistic graphical model of figure 2, defined by the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{h})$ over the $\mathbf{x}$ and $\mathbf{h}$ random variables. The evidence lower bound is derived through Jensen's inequality as follows:

$$\begin{aligned} l(\boldsymbol{\theta}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1:N)}) \\ &= \log \int p_{\boldsymbol{\theta}}(\mathbf{x}^{(1:N)},\mathbf{h})d\mathbf{h} \\ &= \log \int \frac{q(\mathbf{h})p_{\boldsymbol{\theta}}(\mathbf{x}^{(1:N)},\mathbf{h})}{q(\mathbf{h})}d\mathbf{h} \\ &\geq \int q(\mathbf{h}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}^{(1:N)},\mathbf{h})}{q(\mathbf{h})}d\mathbf{h} := \mathcal{L}(\boldsymbol{\theta},q). \end{aligned} \tag{2.9}$$

The main idea of variational inference is to introduce an auxiliary probability distribution $q(\mathbf{h})$ over the hidden random variables, called the variational distribution. **Throughout the thesis we will reserve the letter q to denote variational distributions as opposed to true model distributions denoted by p.** The variational distribution serves as an approximate or surrogate posterior over the hidden random variables, and the main objective is to find the probability distribution q that maximizes the evidence lower bound $\mathcal{L}$ at $\boldsymbol{\theta}$ fixed. In order to show the relation between $q(\mathbf{h})$ and the true posterior $p_{\boldsymbol{\theta}}(\mathbf{h}|\mathbf{x}^{(1:N)})$, we can rewrite the evidence lower bound as:

$$\mathcal{L}(\boldsymbol{\theta},q) = \int q(\mathbf{h}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}^{(1:N)},\mathbf{h})}{q(\mathbf{h})}d\mathbf{h}$$

$$
\begin{aligned}
&= \int q(\mathbf{h}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{h}|\mathbf{x}^{(1:N)}) p_{\boldsymbol{\theta}}(\mathbf{x}^{(1:N)})}{q(\mathbf{h})} d\mathbf{h} \\
&= \int q(\mathbf{h}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{h}|\mathbf{x}^{(1:N)})}{q(\mathbf{h})} d\mathbf{h} + \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1:N)}) \int q(\mathbf{h}) d\mathbf{h} \\
&= l(\boldsymbol{\theta}) - \mathbb{D}_{KL}[q(\cdot)||p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}^{(1:N)})]. \qquad (2.10)
\end{aligned}
$$

Where $\mathbb{D}_{KL}$ represents the Kullback-Leibler divergence, a measure of distance between two probability distributions. Notice that the evidence lower bound is a functional of the probability distribution $q(\cdot)$, hence the variational aspect of the problem.

The objective of the variational distribution is to approximate the true posterior. This can be seen through the Kullback-Leibler term, by maximizing the evidence lower bound w.r.t. q, we minimize the Kullback-Leibler term, thus minimizing the distance between the approximate posterior q and the true posterior $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}^{(1:N)})$. The optimization problem can be written as:

$$
\max_{q} \mathcal{L}(\boldsymbol{\theta}, q) \quad \text{s.t.} \quad \int q(\mathbf{h}) d\mathbf{h} = 1. \qquad (2.11)
$$

The optimization problem of equation (2.11) is often intractable over the entire space of possible probability distributions. Hence, an often used strategy is to restrict the space of search. One of the most used restrictions is what we call the mean-field hypothesis (Jaakkola, 2001; Jordan et al., 1999), in which we suppose that the variational distribution respects a factorization condition:

$$
q(\mathbf{h}) = \prod_{j} q(h_j). \qquad (2.12)
$$

Another recent popular approach is to choose a specific distribution and parameterize it through some set of parameters ϕ: q_{ϕ}. In this case the optimization is transferred to the parameters ϕ, thus returning to classical optimization methods over the function $\mathcal{L}(\boldsymbol{\theta}, \phi)$ with respect to its parameters (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014).

The final piece of the puzzle is to ensure that the optimization of $\mathcal{L}$ w.r.t. the parameters $\boldsymbol{\theta}$ follows the maximum likelihood criterion. This is the case here, and it can easily be proven under the hypothesis that the optimal solution of (2.11) q^* approximates the true posterior. Indeed, if:

$$
q^*(\cdot) \approx p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}^{(1:N)}) \quad \text{then,} \quad \mathbb{D}_{KL}[q^*(\cdot)||p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}^{(1:N)})] \approx 0, \qquad (2.13)
$$

and thus with equation (2.10), we can deduce that:

$$
\mathcal{L}(\boldsymbol{\theta}, q^*) \approx l(\theta). \qquad (2.14)
$$

Hence maximizing $\mathcal{L}(\boldsymbol{\theta}, q^*)$ w.r.t. $\boldsymbol{\theta}$ is equivalent to maximum likelihood estimation.

For further details on Variational Inference methods for probabilistic graphical models we highly recommend the book by Wainwright, Jordan, et al. (2008).

2.5 DEEP LEARNING

Deep learning as a field has retained considerable interest from researchers from all disciplines in recent years. The ability of deep neural networks to solve with high accuracy machine learning tasks ranging from classification and regression to reinforcement learning, is one of the most noteworthy breakthroughs in recent history. Our interest throughout the thesis in neural networks will be mainly in their ability to learn complex features relevant to the task at hand. For this purpose, we will present neural networks as generalized function approximators, defined by the application of successive linear and nonlinear functions (Rosenblatt, 1961). A neural network in our case will be defined by a function often denoted by g, such as:

$$g_{\boldsymbol{\theta}} = f^{(L)} \circ \sigma \circ f^{(L-1)} \circ ... \circ \sigma \circ f^{(2)} \circ \sigma \circ f^{(1)}. \tag{2.15}$$

where each function f^l represents the l^{th} application of the linear or fully connected layer, defined by:

$$f^{(l)}(\mathbf{h}) = \mathbf{h}^T \mathbf{W}_l + \mathbf{b}_l \tag{2.16}$$

$\mathbf{W}_l$, and $\mathbf{b}_l$ represent the weight matrix and bias vector for the l^{th} layer. $\circ$ represents the composition of functions operator. Between each application of two linear layers, a non linear function is applied denoted by σ. Throughout the manuscript this function will denote the Rectified Linear Unit (Nair and Hinton, 2010) defined by:

$$\sigma(\mathbf{h}) = \max(0, \mathbf{h}) \tag{2.17}$$

We will be restricted to the previously defined structure of neural networks as it is sufficient for our applications. The overall parameters of the neural network are assembled in the vector denoted by $\boldsymbol{\theta}$. The parameters are optimized to minimize some cost function relevant to some task, we will often denote this function by $\mathscr{C}$. The optimization is done using a stochastic gradient descent algorithm with momentum (Sutskever et al., 2013) or a close variant such as the ADAM optimizer (Kingma and Ba, 2014).

The computation of the gradient of the loss function $\nabla_{\boldsymbol{\theta}} \mathscr{C}$ is done by the classical backpropagation algorithm (Werbos, 1982). The algorithm is based on the chain rule of calculus, $\mathscr{C}$ can be seen as the composition of two functions u and v, that is $\mathscr{C} = u(\mathbf{y})$ and $\mathbf{y} = v(\mathbf{x})$, then:

$$\frac{\partial \mathscr{C}}{\partial \mathbf{x}_i} = \sum_j \frac{\partial \mathscr{C}}{\partial \mathbf{y}_j} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_i} \tag{2.18}$$

For further details on Deep learning methods and neural network architectures, we highly recommend the following book by Goodfellow et al. (2016).

Part I

SHALLOW PROBABILISTIC GRAPHICAL MODELS FOR FAULT IDENTIFICATION

3

INFINITE CATEGORICAL MIXTURE MODELS FOR FAULT CLUSTERING

3.1 INTRODUCTION

Identifying fault patterns in large telecommunication network and service infrastructures is a difficult task. Most of the considered technical solutions rely either on rule based expert systems or hand crafted expert Bayesian networks (Kogeda, Agbinya, and Omlin, 2006; Kogeda and Agbinya, 2006; Tembo et al., 2017). Although these approaches have had tremendous success, one of the unsung drawbacks is the data processing and the expert knowledge required to build the diagnosis model or rules. For expert systems, for example, the rules created by the expert require knowledge of the existing fault, and the identification of the variables describing the fault. This process requires data processing by hand by an expert of the domain, which is an expensive and time consuming task. Also, the maintenance of the model or rules in the long run can be a significant issue for operational teams.

Recently, machine learning techniques have been tremendously successful in the identification and the extraction of patterns in various domains. In the context of our problem, similar approaches can be used to identify patterns of faults from diagnosis data. This task is thus an unsupervised machine learning task, where labels are not available and obtaining them is as complex as the data processing required to construct expert rules. However, the data gathered from various devices and services in the network is often structured in the form of a table, where each variable takes some range of values.

Clustering such data, gathered from telecommunication networks and services presents many challenges. The first and the main challenge is the unknown number of clusters of faults in the data. The second challenge is the types and multivariate nature of the data. The data is multi-dimensional and can contain categorical and continuous variables. Therefore, classical

clustering algorithms where the number of clusters is to be set a priori require some form of model selection. Furthermore, classical approaches such as KMeans suppose a specific probability distribution for each cluster. These modeling assumptions, including the assumption on the number of clusters, can hurt the performance of the clustering when the data do not comply with such assumptions, which is often the case when dealing with real-world applications.

In this chapter, we present an infinite multivariate categorical mixture model to identify patterns of faults in an unsupervised setting, without any prior expert knowledge, and without the requirement to know a priori the number of different fault patterns. The model is based on the Dirichlet Process (Ferguson, 1973), which allows for learning the number of clusters from the data. However, the Dirichlet Process supposes an infinite number of clusters which translates to an intractable inference problem on the model. Our contributions are the following:

- We propose an infinite multivariate categorical mixture model for clustering categorical data (section 3.3).
- We show how to perform approximate inference on the model, in order to extract the clusters from the data using Variational Inference (section 3.4).
- We demonstrate how the model is able to identify root causes of faults in a synthetic dataset generated from a real-world expert Bayesian Network (section 4).
- We also demonstrate the clustering performance of the model on real operational data acquired from the Fixed Access Network and the Local Area Network (section 5).

3.2 BACKGROUND

3.2.1 Mixture Models as Probabilistic Graphical Models

Mixture models are one of the simplest forms of probabilistic graphical models. The objective of mixture models is to model the distribution of an observable random variable $\mathbf{x}$ as a mixture of base distributions. The main application goal is to identify patterns captured by the base distributions. The assumption is that the observable random variable $\mathbf{x}$ depends on a latent variable

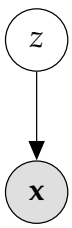

Figure 4: Graphical representation of a mixture model.

denoted by z taking values in a discrete set $\{1, ..., K\}$, with probabilities $\pi_k = p(z = k)$. Depending on the value taken by z, the random variable $\mathbf{x}$ follows some base probability distribution $p_\theta^{(z)}$, where θ and π are parameters of the model to be learned to fit the data through maximum likelihood (2.1). Figure 4 shows a graphical representation of a mixture model.

3.2.2 The Dirichlet Process

In a classical mixture model, we must specify a number of components (or clusters) for the random variable z. Identifying the "correct" number of components is a task full of challenges, that has been tackled through model selection approaches (Bishop, 2006). The Dirichlet process (Ferguson, 1973) allows us to identify directly the number of components much more naturally. The idea behind the Dirichlet process is to suppose an infinite number of components, and specify a prior on the creation of new components. The creation of new components has a prior probability defined by draws from a beta distribution, formally:

$$\begin{aligned} \beta_k &\sim \text{Beta}(\cdot; 1, \eta) \\ \pi_k &= \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) \end{aligned} \tag{3.1}$$

The probability of the k^{th} component being active is governed by draws from the beta distribution of hyperparameter η. One way to think of such construction is through the stick-breaking construction (Sethuraman, 1994). The stick breaking construction is as follows, π_k represents the length of the k^{th} piece broken from a stick of length 1. If the concentration parameter is small, π_k will be large (close to 1) at the first times the stick is broken, therefore the stick will be broken a small number of times (small number of clusters). If η is large, π_k will be small (close to 0) and the stick can be broken a large number of times (large number of clusters). The probability that the $(n+1)^{\text{th}}$ data point belongs to a new cluster k^* or to K existing clusters is (Murphy, 2012):

$$\mathbb{P}[z_{n+1} = z | z_{1:n}, \eta] = \frac{1}{\eta + n} \left[\eta \mathbb{1}[z = k^*] + \sum_{k=1}^{K} n_k \mathbb{1}[z = k] \right] \tag{3.2}$$

where n_k is the number of data points in cluster k. If $\eta \to \infty$ a new cluster would be created for each data point, and if $\eta \to 0$ all data points are concentrated in the first cluster. The intuition behind the Dirichlet process is the following: as the number of data points increases, we allow the number of clusters to grow according to the concentration parameter and the clusters already assigned. New samples are assigned to existing clusters

if they match, otherwise a new cluster is created for them. Therefore the Dirichlet Process allows the mixture model to cluster the data and identify automatically the number of clusters necessary to explain the data.

3.2.3 Notations

In our case, we consider that the random variable $\mathbf{x}$ represents the variables or features of a network such as an equipment status, an alarm, or a physical metric. We only consider categorical random variables. Continuous variables such as optical powers or temperatures are discretized, using standard methods such as equal frequency or equal width discretization. Thus, each random variable takes values in a set$E_i = \{v_{1i}, ..., v_{|\mathbf{x}_i|i}\}$, where $|\mathbf{x}_i|$ is the number of modalities of variable $\mathbf{x}_i$. Let $\mathbf{x}^{(1:N)}$ represent N samples of the vector $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_d]^T$ of dimension d. We denote by $z^{(1:N)}$ N random variables, where z_n represents the fault cluster of sample $\mathbf{x}_n$.

3.3 INFINITE CATEGORICAL MIXTURE MODELS

3.3.1 Structure and Definition

Using the stick-breaking construction introduced in the previous section for the weights π_k of the k^{th} cluster, the generative process for our proposed Infinite Categorical Mixture Model is:

$$z \sim \text{Cat}(\cdot|\boldsymbol{\pi}) = \prod_{k=1}^{\infty} \boldsymbol{\pi}_k^{\mathbb{1}[z=k]} \tag{3.3}$$

$$\mathbf{B}_{ki} \sim \text{Dir}(\cdot; \boldsymbol{\alpha}_i, |\mathbf{x}_i|) \propto \prod_{v \in E_i} \mathbf{B}_{k,i,v}^{\boldsymbol{\alpha}_{iv}-1} \tag{3.4}$$

$$\text{s.t} \quad \sum_{v \in E_i} \mathbf{B}_{k,i,v} = 1 \tag{3.5}$$

$$\mathbf{x}_i | z = k, \mathbf{B} \sim \text{Cat}(\cdot|\mathbf{B}_{ki}) = \prod_{v \in E_i} \mathbf{B}_{k,i,v}^{\mathbb{1}[\mathbf{x}_i=v]} \tag{3.6}$$

where π_k is defined through the Dirichlet process defined in the previous section.

Figure 5 shows a graphical representation of the generative process of the model. The observed variable $\mathbf{x}$ depends on the cluster assignment z, and the parameters $\mathbf{B}$. We adopt a full bayesian approach where we consider the parameters of the model as random variables themselves. Thus, the parameters $\mathbf{B}$ have a prior that we choose as a Dirichlet prior with hyperparameters $\boldsymbol{\alpha}$.

Based on the assignment of cluster z a sample $\mathbf{x}_i$ for dimension i is drawn based on the conditional probability for variable $\mathbf{x}_i$ taking a certain value $v \in E_i$, defined by the $\mathbf{B}$ tensor:

$$\mathbf{B}_{k,i,v} = p[\mathbf{x}_i = v | z = k] \tag{3.7}$$

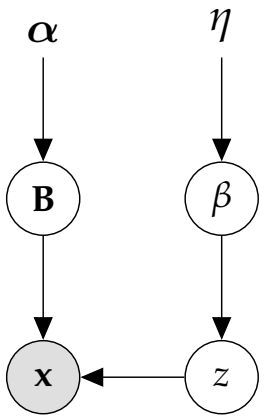

Figure 5: Graphical representation of the model in plate notation.

The Dirichlet prior on the parameters $\mathbf{B}$ is governed by the hyperparameter $\boldsymbol{\alpha}$ called the concentration parameter. It allows us to inject prior knowledge about the modalities of variable $\mathbf{x}_i$. $\boldsymbol{\alpha}_{iv}$ thus represents a weight for modality v in the Dirichlet distribution associated with variable $\mathbf{x}_i$. In the case where no information is available we use an uninformative prior $\boldsymbol{\alpha}_{iv} = \frac{1}{|\mathbf{x}_i|}$.

3.3.2 Learning and Inference

In order to fit the model to the data and identify the clusters representing the different types of network failures, we need to compute or approximate the posterior distribution:

$$p(z^{(1:N)}, \mathbf{B}, \beta | \mathbf{x}^{(1:N)}) = \frac{p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)})}{p(\mathbf{x}^{(1:N)})} \tag{3.8}$$

Markov Chain Monte Carlo (MCMC) approaches are commonly used to fit such models (Neal, 2000). The main idea behind these approaches is to run a Markov chain long enough (until convergence) where the stationary distribution at convergence is the posterior of interest as defined in equation (3.8). The advantage of MCMC methods is that they are guaranteed to converge to the true posterior. However, one major drawback of such approaches arises with large datasets, in this case, MCMC methods are very expensive computationally (Blei, Kucukelbir, and McAuliffe, 2017). Variational inference by comparison can utilize all the optimization tools to bypass this issue, such as Stochastic optimization and distributed optimization (Robbins and Monro, 1951), making the inference run faster. Furthermore, in terms of predictive accuracy it has been shown that variational inference and MCMC methods often have the same performance (Blei, Jordan, et al., 2006; Braun and McAuliffe, 2010).

In the next section we detail the variational inference approach for our proposed model.

3.4 VARIATIONAL INFERENCE FOR THE ICMM

3.4.1 The mean field approximation

In order to introduce the variational inference approach, we denote by $\boldsymbol{\zeta} = \zeta_{1:m} = \{z^{(1:N)}, \mathbf{B}, \beta\}$ the vector grouping all the hidden variables of the model. Solving the inference problem of the model amounts to determining $p(\boldsymbol{\zeta}|\mathbf{x}^{(1:N)})$. As detailed in section 2.4, closed form solutions for this quantity are intractable. Variational inference and the mean-field approximation allow us to approximate this intractable distribution by a set of distributions for which the inference is tractable, namely the mean field family. A distribution q is said to be in the mean field family for variables $\boldsymbol{\zeta}$, if it verifies:

$$q(\boldsymbol{\zeta}) = \prod_{l=1}^{m} q(\zeta_l) \tag{3.9}$$

The main idea of variational inference is to approximate the intractable distribution (3.8), by finding the closest mean field family member in terms of Kullback-Leibler divergence i.e.:

$$q^* = \min_q \mathbb{D}_{KL}\left[q(\cdot)||p(\cdot|\mathbf{x}^{(1:N)})\right] \tag{3.10}$$

By exploiting the factorization in the mean field family, we can show that the solution q^* verifies the following fixed point equations:

$$\log q^*(\zeta_l) = \text{const} + \mathbb{E}_{\boldsymbol{\zeta}_{-l}\sim q^*}\left[\log p(\boldsymbol{\zeta}, \mathbf{x}^{(1:N)})\right] \quad \forall l \tag{3.11}$$

For an explicit derivation of this criterion we refer the reader to appendix A. In the case of our model, the mean-field family is defined as :

$$q(z^{(1:N)}, \mathbf{B}, \beta) = \prod_{n=1}^{N} q(z_n) \prod_{i=1}^{d}\prod_{k=1}^{T} q(\mathbf{B}_{k,i}) \prod_{k=1}^{T} q(\beta_k) \tag{3.12}$$

We also suppose that $q(\beta_T = 1) = 1$ hence $q(z_n > T) = 0$, i.e. the number of clusters is truncated to an upper bound on the true number of clusters (Blei, Jordan, et al., 2006). A noteworthy aspect of this approach is that the true posterior given by (3.8) has an infinite number of factors, however the approximate distribution q is constrained based on the previous conditions. Therefore, the true model is unchanged however the minimization problem is relaxed in order to be solved efficiently.

3.4.2 Optimal Variational Distributions and Update Equations

By applying equation (3.11) to our Infinite Categorical Mixture Model, we obtain:

$$\log q^*(z^{(n)}) \quad = \quad \text{const} + \mathbb{E}_{\{z^{-n},\beta,\mathbf{B}\}\sim q^*}\left[\log p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)})\right]$$

Algorithm 2 Variational Inference for the ICMM

Input: $\mathbf{x}^{(1:N)}, T, \eta$
Initialize $\phi, \epsilon, \gamma_1, \gamma_2$
$\mathcal{L}^{(0)} = -\infty$
for $t = 1...\infty$ **do**
  Compute: $\gamma_{1,k}^{(t)} \quad \forall k \quad (3.16)$
  Compute: $\gamma_{2,k}^{(t)} \quad \forall k \quad (3.17)$
  Compute: $\epsilon_{k,i,v}^{(t)} \quad \forall v, \forall k, \forall i \quad (3.18)$
  Compute: $\phi_{n,k}^{(t)} \quad \forall n, \forall k \quad (3.15)$
  Compute: $\mathcal{L}^{(t)} \quad (3.19)$
  if $\mathcal{L}^{(t)}$ changes by less than 10^{-6} **then**
    break
  end if
end for
$z_n = \arg\max_k \phi_{nk} \quad \forall n$
return $z^{(1:N)}, \phi$

$$\begin{aligned} \log q^*(\mathbf{B}_{k,i}) &= \text{const} + \mathbb{E}_{\{z^{(1:N)},\beta,\mathbf{B}_{-\{k,i\}}\}\sim q^*}\left[\log p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)})\right] \\ \log q^*(\beta_k) &= \text{const} + \mathbb{E}_{\{z^{(1:N)},\beta_{-k},\mathbf{B}\}\sim q^*}\left[\log p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)})\right] \quad (3.13) \end{aligned}$$

And by substituting the expression of $p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)})$ resulting from the graphical representation of the model (Figure 5), we then deduce the following approximating distributions:

$$\begin{aligned} q^*(z^{(n)}) &= \text{Cat}(z_n; \phi_n) \\ q^*(\mathbf{B}_{k,i}) &= \text{Dir}(\mathbf{B}_{k,i}; \epsilon_{k,i}, |\mathbf{x}_i|) \\ q^*(\beta_k) &= \text{Beta}(\beta_k; \gamma_{1,k}, \gamma_{2,k}) \quad (3.14) \end{aligned}$$

And the mean field fixed point equations for the parameters are the following, where ψ is the digamma function (proofs given in appendix B.1):

$$\begin{aligned} \log \phi_{nk} &= \sum_{i=1}^{d} \sum_{v \in E_i} \mathbb{1}[\mathbf{x}_i^{(n)} = v][\psi(\epsilon_{k,i,v}) - \psi(\sum_{v' \in E_i} \epsilon_{k,i,v'})] \\ &+ \psi(\gamma_{1,k}) - \psi(\gamma_{1,k} + \gamma_{2,k}) \quad (3.15) \\ &+ \sum_{l=1}^{k-1} [\psi(\gamma_{2,l}) - \psi(\gamma_{1,l} + \gamma_{2,l})] \\ &\text{s.t} \quad \sum_{k=1}^{T} \phi_{nk} = 1 \\ \gamma_{1,k} &= 1 + \sum_{n=1}^{N} \phi_{nk} \quad (3.16) \end{aligned}$$

$$\gamma_{2,k} = \eta + \sum_{n=1}^{N} \sum_{l=k+1}^{T} \phi_{nl} \tag{3.17}$$

$$\epsilon_{k,i,v} = \alpha_{i,v} + \sum_{n=1}^{N} \phi_{nk} \mathbb{1}[\mathbf{x}_i^{(n)} = v] \tag{3.18}$$

Thus, the identification of the optimal approximating distribution is done by iterating the previous equations. The stopping criterion is defined using the Kullback-Leibler divergence:

$$\mathcal{L}(q) = -\mathbb{D}_{KL}\left[q(\cdot)||p(\cdot, \mathbf{x}^{(1:N)})\right] \tag{3.19}$$

At convergence the value of $\mathcal{L}$ should reach a plateau corresponding to a lower bound on the likelihood. Algorithm 2 summarizes the procedure for learning the approximating posterior and for inferring the clusters.

3.5 EXPERIMENTS

We test our proposed model on the task of fault clustering in two datasets. The first one is a synthetic dataset generated using an expert bayesian network, and the second one a real dataset collected from equipment deployed in a real fiber-to-the-home network. We evaluate our model in comparison with other clustering baselines. Furthermore, we also compare the Variational Inference approach with the Gibbs sampling approach.

3.5.1 Datasets

Synthetic dataset

In order to test our model in a controlled environment, we reuse the work done by Tembo et al. (2017). The authors designed an expert bayesian network that reflects the behaviour of a real GPON-FTTH network. The bayesian network is depicted on figure 6 where the colored top nodes represent the root causes and the other nodes the observations coming from the network. Note here that the bayesian network is only used to generate a synthetic dataset similar to operational data, it is not used in the clustering method.

In order to generate the synthetic data for the experiment, we simulate a fault by activating the root cause for the fault, and we sample the visible variables. Here we sampled 6 uncorrelated faults (for more details on the expert bayesian network and remaining variables, we refer the reader to (Tembo et al., 2017)):

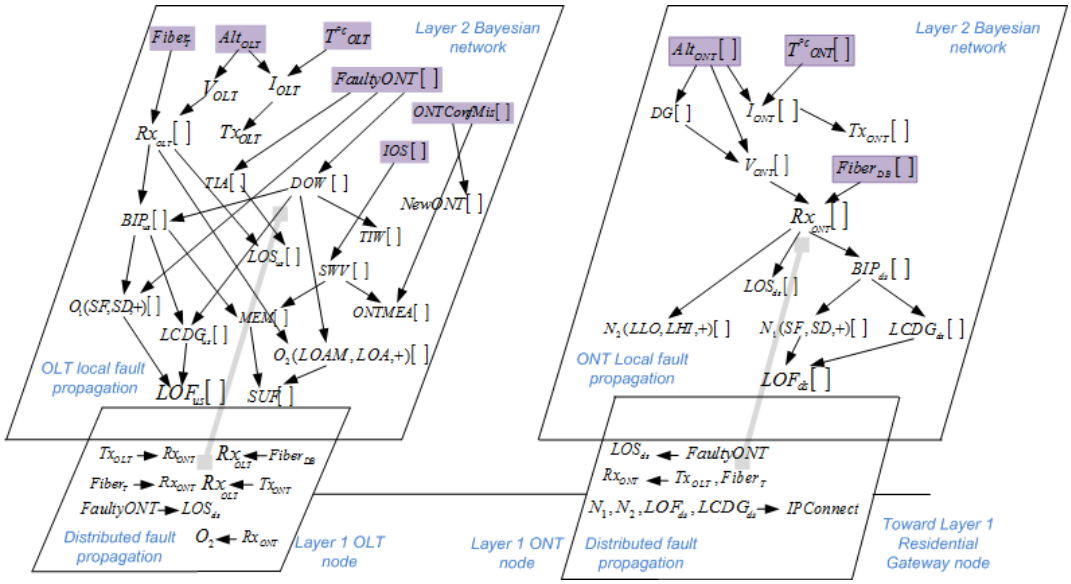

Figure 6: Expert Bayesian Network of the FTTH GPON (Tembo et al., 2017). Colored nodes are possible root causes whereas non-colored nodes are observations.

- **AltONT:** highlights a problem with the power supply of the optical network termination (ONT). This hidden variable controls the current of the ONT I_{ONT}, the Dying Gasp alarm DG, and the electric voltage of the ONT V_{ONT}.
- **AltOLT:** describes a problem with the power supply of the optical line termination (OLT). It controls similar variables for the OLT, V_{OLT} and I_{OLT}.
- **FaultyONT:** denotes whether the ONT is faulty, and symbolizes the global state for an ONT. It controls the alarms TIA (Transmission Interference Alarm), DOW (Drift Of Windows), and SF (Signal Fail) or SD (Signal Degraded).
- **FiberDB:** represents the state of the drop optical fiber and controls Rx_{ONT} the power at which the ONT receives the signal.
- **IOS:** (Image Operating System) refers to an incompatibility between the OS of an OLT and the OS of an ONT. It controls the variable SWV describing the software version alarm.
- **TcOLT:** represents the temperature of the OLT.

We sampled 150 data points for each fault, using the likelihood weighted sampling method, based on the conditional distribution tables of each observation given the root cause (Fung and Chang, 1990). Therefore we generate for each fault a pattern of the visible variables for a specific customer equipment (ONT). The resulting dataset contains 900 samples, where each sample represents a realization of the 29 visible nodes of the Bayesian network.

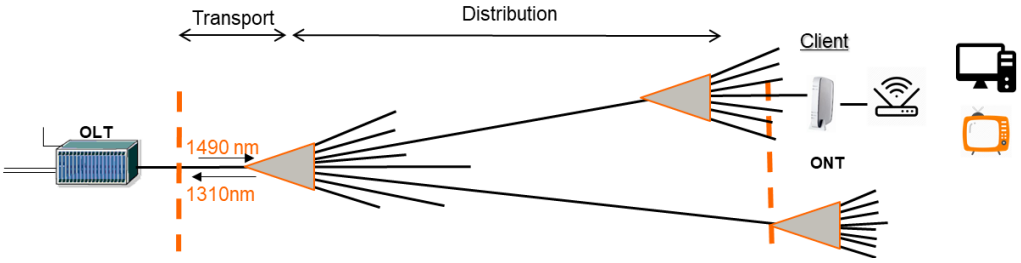

Figure 7: Network scope for fault pattern extraction from real operational data

Dataset	Classes	Features	Instances	Label rate
Synthetic	6	29	900	100 %
Real-world	8	85	41185	29.4 %

Table 1: Description of each dataset.

Real-world dataset

The second dataset used for evaluation is collected from a real world operating GPON-FTTH and Local area network depicted in figure 7. For the fixed access network we collected data describing the status of the OLT, and the status of the ONT (olt_status, ont_status, ont_download_status), in addition to continuous metrics such as the powers of transmission and reception in each equipment. For the local area network, we collected data describing the status of the router and the different services provided such as, IPTV, VOD (Video On Demand) and VoIP. We also collected data describing the account status of the client.

The data collected corresponds to instances where some faults are known (identified) and others where the fault is not known to experts or expert systems. The unidentified fault could be the same as a known fault or entirely novel to the system. In this chapter, we focus on the unidentified faults representing **70.6** % of the dataset. We adopt a full unsupervised method, i.e. we consider that no ground truth is available and we attempt to identify patterns blindly.

Table 1 summarizes each dataset and its characteristics.

3.5.2 Baselines and Metrics

We compare our clustering approach to other state of the art clustering methods, where the number of clusters is not required and is determined automatically. The baselines chosen are the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al., 1996), the affinity propagation clustering method (Frey and Dueck, 2007), the mean shift clustering

	AltONT	AltOLT	FaultyONT	FiberDB	IOS	TcOLT
cluster 11	**150**	0	0	0	0	0
cluster 25	0	**132**	0	0	0	0
cluster 5	0	0	**148**	0	0	0
cluster 3	0	0	1	**150**	0	**2**
cluster 39	0	1	0	0	**147**	3
cluster 17	0	17	1	0	3	**145**
cluster 1	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
cluster 50	0	0	0	0	0	0

Table 2: Confusion Matrix of the Infinite Categorical Mixture Model on the synthetic dataset.

method (Comaniciu and Meer, 2002), and the infinite Gaussian mixture model (IGMM) (Rasmussen et al., 1999).

The evaluation metric is the clustering accuracy, it is similar to the classification accuracy, however, in the clustering task the clusters are not identifiable with the labels, i.e. the clusters can change from one run of the algorithm to another. Therefore we need to test all possible combinations and choose the best one. The clustering accuracy is defined as in (Xie, Girshick, and Farhadi, 2016):

$$\text{ACC} = \max_{m \in \mathcal{M}} \frac{\sum_{n=1}^{N} \mathbb{1}\left[l_n = m(z_n)\right]}{N}$$

where z_n is the cluster assignment, l_n the true label and $\mathcal{M}$ the set of all possible one-to-one mappings.

3.6 RESULTS

3.6.1 Clustering results on the synthetic dataset

Here, we report the clustering results of our model on the synthetic dataset. First, we compare our model with the considered baselines. In table 3, we report the mean and standard deviation of the clustering accuracy for each model over 10 runs. A search over the hyperparameters of each model has been performed and the best performing model is selected for each baseline. As shown our model outperforms all models considerably. This

Table 3: Comparison between our model and the baselines.

Methods	Affinity propagation	Mean shift	DBSCAN	IGMM	**Ours**
Accuracy	0.35 ± 0.0	0.69 ±0.0	0.29 ± 0.0	0.57 ± 0.09	**0.9 ± 0.07**

shows the advantage of explicitly modeling a clustering problem to match the specificity of the data. Using classical approaches blindly often leads to poor results.

Second, we report the evidence lower bound of equation (3.19) across variational inference iterations, for 10 runs of the model with their standard deviation. The evidence lower bound gives us a criterion for evaluating the convergence of the model. As depicted in figure 8, the evidence lower bound increases until reaching a plateau, which indicates that the model has converged.

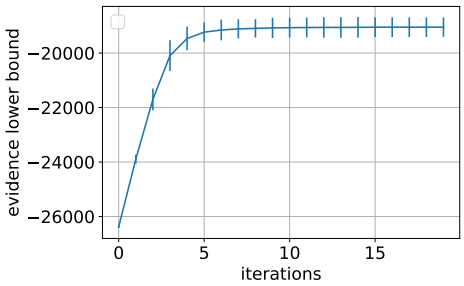

Figure 8: evidence lower bound of the model across variational inference iterations, mean and standard deviation over 10 runs.

Finally, in table 2, we report the confusion matrix for the best model (highest elbo). As shown, the confusion matrix is diagonal with the majority of instances well clustered to their ground-truth classes of faults. Furthermore, and more importantly, the advantage of the Dirichlet process is clearly demonstrated. We start with **50** clusters, however only **6** are filled corresponding to the ground-truth classes, the others are kept empty. Therefore, the model clusters the data and **automatically identifies the correct number of clusters**.

3.6.2 Variational inference vs Gibbs sampling

We report the results of the comparison between the Gibbs sampling fitting of the model and the variational inference method. We compare the two approaches in terms of complexity and accuracy on the synthetic dataset. As seen in figures 9 and 10, both methods arrive at high accuracy, they have the same performance in terms of predictive accuracy around 90 %. However, as the number of instances in the dataset increases, we see that the Gibbs sampling convergence time increases considerably. In contrast with the variational inference approach the convergence time stays stable. This is a foreseeable consequence, that stems from the fact that iterations of Gibbs sampling can only be done sequentially and are of complexity $\mathcal{O}(NKD)$, where N is the number of instances, K the number of classes, and D the

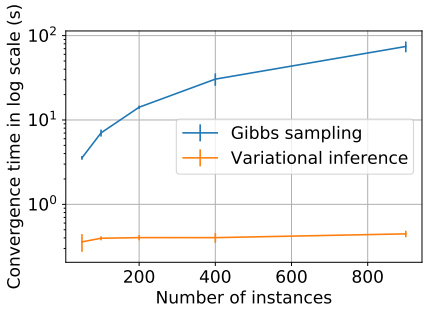

Figure 9: Convergence time as a function of the number of instances for the variational inference and Gibbs sampling approaches.

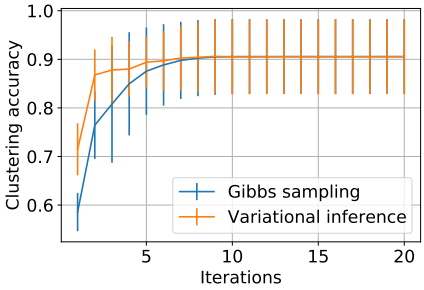

Figure 10: Comparison of the clustering accuracy across iterations for the variational inference and Gibbs sampling approaches.

dimension of the observable random variable $\mathbf{x}$. Variational inference on the other hand, has complexity $\mathcal{O}(NK)$ (updates of ϕ, others are negligible). Furthermore, the updates of variational inference can be written in matrix form that can be run in parallel form.

3.6.3 Pattern recognition on real-world data

In this second experiment, we apply our clustering model on real operational network data. However in this setup we don't have access to a ground truth so the aim of this experiment is to demonstrate the benefit of our clustering model in a data exploration process. We obtained 6 main clusters corresponding to known problems. We report for each cluster the number of customers assigned to the cluster and the most common pattern, i.e. characteristic values that the variables take and the counts for each variable (Figure 11). Furthermore, we give an interpretation based on network expertise to understand what kind of problem the customers are facing in each cluster.

- **Cluster 37**: (600 clients) This cluster gathers customers with a problem on the remote PVR for the IPTV: Remote_PVR = Missing, and all other values correspond to an operational state of the client's line: (client_account_status=1, OLT_status=OK, router_status=Enabled, ONT_status=NODEFECT...).
- **Cluster 30:** (> 2500 clients) In this cluster the ONT is not detected: (ONT_status = Missing), where all other services are operational (router_status=Enabled, voip_status=Ok, tv_profile_status=operational ...).

- **Cluster 43:** ($\approx$ 2500 clients) This cluster corresponds to the optimal behavior of the network and services, all equipment are detected and all services are operational.
- **cluster 44:** ($\approx$ 550 clients) This cluster describes a problem with the router where the router is disabled (router_status=DISABLED, router_ipv6_status=DISABLED), however all other equipments are operational and client account status is activated.
- **Cluster 40:** ($\approx$ 27 clients) This cluster represents the case where an account is suspended and the VOIP service is down (client_account_status = 0, tv_profile_status=SUSPENDED, VOIP_status=KO), however all pieces of equipment are operational.
- **Cluster 12:** ($\approx$ 130 clients) This cluster gathers clients with deactivated WiFi (router_WiFi_status = Down, router_status=Enabled), and all other equipment is operational (ont_status = NODEFECT, OLT_status=OK...).

This experiment shows that our clustering method can be helpful in a data exploration process where the number of clusters is not known a priori. With a simple network expertise we can see that the clusters are relevant and describe a particular behaviour on the network.

3.7 DISCUSSION & PRIOR WORK

Machine learning techniques for fault detection in telecommunication networks are promising. Bayesian networks have been dominant in this domain (Kogeda, Agbinya, and Omlin, 2006; Kogeda and Agbinya, 2006; Tembo et al., 2017). The main advantage of Bayesian network modeling is their easy interpretable nature. Root causes of faults are modeled by variables of the Bayesian network and inference can be performed in order to determine the root cause from the observable variables. This approach however, requires expert knowledge of the specific domain of faults and a time consuming task to build the Bayesian network and all the dependencies between the variables. Recently, researchers have been interested in learning the structure of the Bayesian network from data (Koller and Friedman, 2009), and such approaches have been successfully used for the self-diagnosis problem (Bennacer et al., 2014). However, the main problem, in such approaches, is the loss of the interpretability. The structure learned from the data can be one of a class of equivalence of optimal structures, hence, this often results in structures that are optimal but hard to interpret.

The second group of machine learning approaches, are anomaly detection based approaches (Chandola, Banerjee, and Kumar, 2009b; Hood and Ji, 1997). Although these methods allow for accurate detection of anomalies

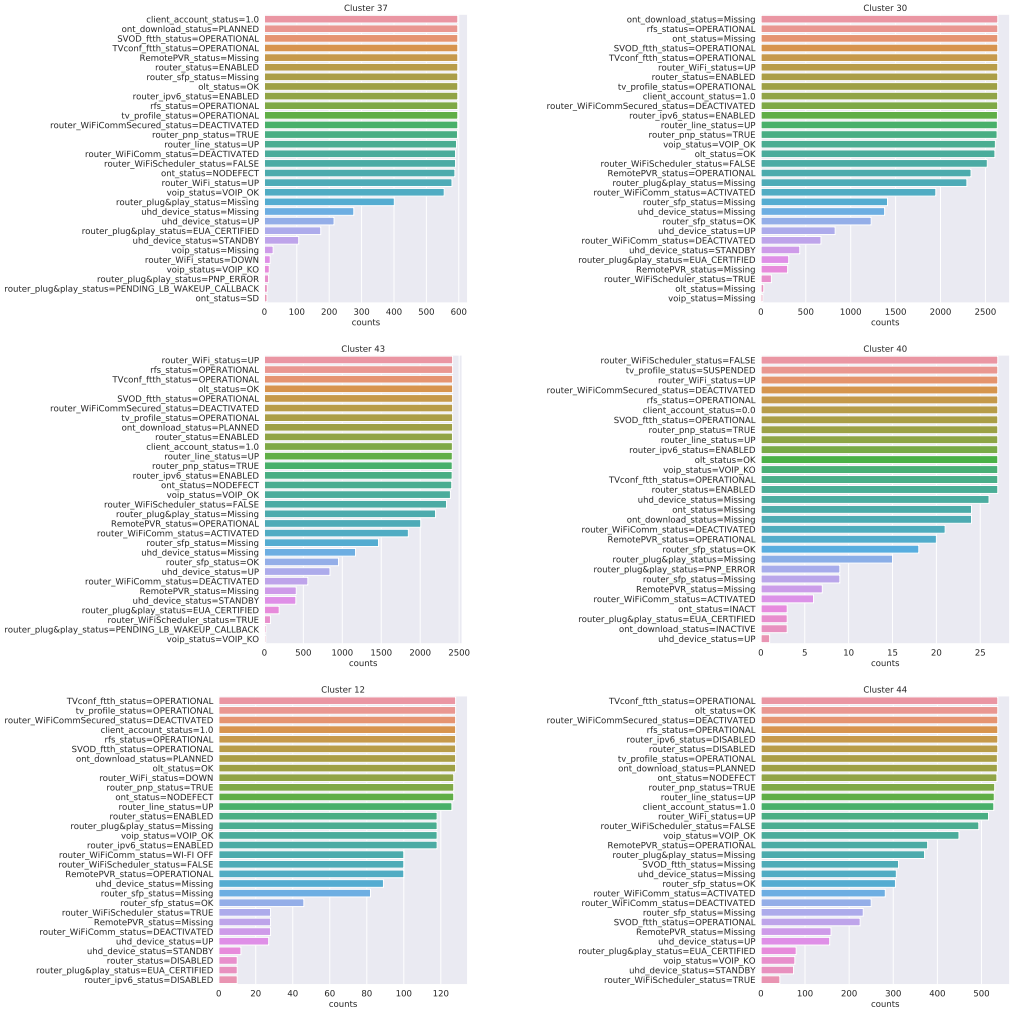

Figure 11: Assignments defining each cluster from most occurring to least occurring

from data, their main drawback is that all anomalies identified are grouped in a single cluster. Classifying each anomaly requires either a relabeling process by hand or a clustering process. Therefore, the natural reformulation of the self-diagnosis problem is as a clustering problem. Clustering types of faults from the data has been previously proposed (Adda, Qader, and Al-Kasassbeh, 2017; Hashmi, Darbandi, and Imran, 2017). However, most of these approaches rely on classical algorithms where the number of clusters is known a priori or estimated by model selection. On the contrary, our approach allows for automatic determination of the number of clusters and requires no intervention from an expert. A post processing of the clusters by an expert is still needed in order to identify the fault present in each cluster and the related root cause. (Chen et al., 2004) proposed a method based on decision trees to accomplish this task.

3.8 LIMITATIONS

Clustering patterns of faults in network data can be achieved using the categorical mixture model approach as proposed in this chapter. However, two main limitations arise and are quite clear at this stage. The first, is the fact that the model can be applied only to categorical random variables. Thus, continuous variables have to be preprocessed in advance. The second limitation stems from the fact that labels are often available as discussed in subsection 3.5.1. However, the proposed infinite mixture model as formulated can only cluster the data in an unsupervised manner, without taking advantage of the existing labels available. In the next chapter, we extend the definition of the mixture models to treat categorical and continuous random variables alike. Furthermore, we show how we can use semi-supervision using the labels available to guide the clustering based on the ground-truth available.

3.9 CONCLUSION

We have introduced the infinite categorical mixture model, for clustering faults in categorical network data. We showed how the model can be used to analyze data gathered from telecommunication networks and services and how to discover fault patterns in an unsupervised manner. The model is capable of identifying the correct number of clusters to analyze the data using the Dirichlet process. We showed how Variational Inference can be used to perform inference on the model. This approach allows inference to scale well with the dimensionality of the data, and the convergence of the model can be determined explicitly.

4

SEMI-SUPERVISION AND MIXED-TYPE INFINITE MIXTURE MODELS

4.1 INTRODUCTION

In the previous chapter, we have explored the problem of clustering faults from network data in an unsupervised manner. As discussed, the previous model, i.e. the infinite categorical mixture model, was capable of identifying patterns of faults and simultaneously inferring the number of faults from the data. However, as its name suggests the model only deals with categorical random variables, meaning that continuous random variables need to be preprocessed in order for the model to be applied. It turns out our model can be generalized to continuous and categorical random variables by expanding the definition of the emission distributions of the visible random variables.

Furthermore, in most real-world applications of diagnostics some expert knowledge is available. The expert knowledge is often represented by partial labels, i.e. an expert looks at a data point and determines which class of fault it belongs too. The infinite categorical mixture model of the previous chapter does not use this extra-information of partial labels. Fortunately, semi-supervised learning methods could be applied to take advantage of both large amounts of unlabeled data and a small fraction of labeled data (Chapelle, Scholkopf, and Zien, 2009).

With slight modifications to our previous infinite categorical mixture model, we could bypass these issues. Indeed, probabilistic graphical models can incorporate labels as observable random variables. This approach has been successfully applied to classical mixture models with the Expectation-Maximization algorithm (Nigam et al., 2000; Seeger, 2006). Furthermore, continuous data can be incorporated in the probabilistic graphical model through continuous distributions such as the Gaussian distribution. The challenge that arises is the non-conjugacy between the categorical distribu-

tions and the continuous ones, making classical inference methods such as Gibbs sampling non viable.

In this chapter, we generalize the previous infinite categorical mixture model to a semi-supervised mixed-type infinite mixture model capable of dealing with categorical and continuous distributions alike, and also capable of incorporating expert knowledge in the form of partial labels. Our contributions are the following:

- We start by giving a formal definition of the semi-supervised mixed-type infinite mixture model as a probabilistic graphical model.
- We show how to solve the inference problem through mean-field variational inference similarly to the previous chapter, and how to use the model to make predictions on future data samples.
- We apply our model on a real-world semi-supervised fault cluster identification problem from GPON-FTTH data, and we compare our model to classical baselines.

4.2 NOTATIONS

We start by defining additional notations in this section in order to rigorously introduce the semi-supervision and the continuous random variables. We keep the same definition for the visible random variables as in the previous chapter, i.e. $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_d]$. However, within the d random variables considered, there exists two subsets, those of the continuous and categorical random variables. We denote $\mathbf{x}_{\mathcal{C}} = \{\mathbf{x}_j, \text{ if } j \text{ is a continuous dimension}\}$ the vector of continuous random variables, and we denote the vector of categorical random variables by $\mathbf{x}_{\mathcal{D}} = \{\mathbf{x}_j, \text{ if } j \text{ is a categorical dimension}\}$.

Second, in order to introduce the semi-supervision, we consider the set of indexes of the whole dataset of N instances denoted by $\{1, ..., N\} = \mathscr{D}_l \cup \mathscr{D}_u$ where $\mathscr{D}_u = \{n \in \{1, ..., N\}, \text{ s.t. } \mathbf{x}^{(n)} \text{ is unlabeled}\}$ denotes the indexes of the observed instances that are unlabeled, and, $\mathscr{D}_l = \{n \in \{1, ..., N\}, \text{ s.t } \mathbf{x}^{(n)} \text{ is labeled}\}$ denotes the indexes of the observed instances that are labeled. We denote the n^{th} label for instance $\mathbf{x}^{(n)}$ by $y^{(n)}$.

Finally, we introduce the Wishart distribution, that will serve as a prior in the continuous case over precision matrices. The Wishart distribution is a distribution defined on the space of $p \times p$ symmetric, non-negative, definite matrices $\mathbf{\Lambda}$. Its density function is defined as:

$$\mathcal{W}(\mathbf{\Lambda}; \mathbf{L}, \nu) = \frac{\det(\mathbf{\Lambda})^{\frac{\nu-p-1}{2}} \exp\{-\frac{1}{2}\text{tr}(\mathbf{L}^{-1}\mathbf{\Lambda})\}}{2^{\frac{\nu p}{2}} \det(\mathbf{L})^{\frac{\nu}{2}} \Gamma_p(\frac{\nu}{2})}, \tag{4.1}$$

where $\mathbf{L}$ is called the scale matrix, ν is a scalar called the degrees of freedom, tr() represents the trace function and Γ_p represents the multivariate gamma function. The Wishart distribution serves as a conjugate prior for the precision matrix of a multivariate normal distribution.

4.3 MODEL DESCRIPTION

4.3.1 Structure and definition

The proposed model in this chapter is an extension of the infinite categorical mixture model to the continuous case, with the advantage of introducing semi-supervision. Certain aspects of the previous model remain unchanged, namely, the Dirichlet process and the cluster hidden variables, defined by the following generative process, which is the same as the previous chapter:

$$\beta_k \sim \text{Beta}(\cdot; 1, \eta) \quad \forall k \in \mathbb{N}^* \tag{4.2}$$

$$\boldsymbol{\pi}_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) \tag{4.3}$$

$$z \sim \text{Cat}(\cdot|\boldsymbol{\pi}). \tag{4.4}$$

Following the generation of the cluster hidden random variable z, we need to specify the emission distributions of the hidden random variables $\mathbf{x} = [\mathbf{x}_\mathcal{D}, \mathbf{x}_\mathcal{C}]$. We can follow the same procedure as the last chapter to model the categorical random variables, that is:

$$\forall i \in \mathcal{D} \quad \mathbf{B}_{ki} \sim \text{Dir}(\cdot; \boldsymbol{\alpha}_i, |\mathbf{x}_i|) \tag{4.5}$$

$$\mathbf{x}_i | z = k, \mathbf{B} \sim \text{Cat}(\cdot|\mathbf{B}_{k,i}), \tag{4.6}$$

where the parameters $\mathbf{B}$ and the sets E_i are defined in the same way as in subsection 3.3.1.

The last part of the puzzle is to specify emission distributions over the continuous observable random variables given the cluster assignment. In this model, we opt for the simplest of modeling and we will explore its limitations in later chapters. We will suppose that the continuous random vector is Gaussian with conjugate priors on the mean and precision matrix (inverse of the covariance matrix):

$$\mathbf{x}_\mathcal{C} | z = k, \mathbf{M}, \boldsymbol{\Lambda} \sim \mathcal{N}(\cdot; \mathbf{M}_k, \boldsymbol{\Lambda}_k^{-1}), \tag{4.7}$$

where the $\mathbf{M} = [\mathbf{M}_{k,j}]$ and $\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_{k,i,j}]$ are respectively the mean and precision matrices for each cluster.

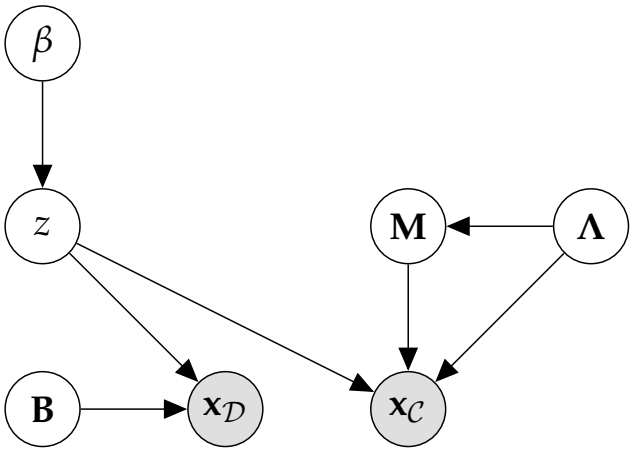

Figure 12: Graphical representation of the mixed-type infinite mixture model in plate notation.

In this approach, we also adopt a full bayesian setting where the mean and precision matrices are considered random variables with Gaussian and Wishart conjugate priors respectively:

$$\forall k \quad \mathbf{M}_k|\mathbf{\Lambda}_k \sim \mathcal{N}(\cdot; \boldsymbol{\mu}_0, (\kappa_0 \mathbf{\Lambda}_k)^{-1}) \tag{4.8}$$

$$\mathbf{\Lambda}_k \sim \mathcal{W}(\cdot; \mathbf{L}_0, \nu_0) \tag{4.9}$$

The previous bayesian priors are standard in classical Gaussian mixture models (Murphy, 2012). $\{\boldsymbol{\mu}_0, \kappa_0, \mathbf{L}_0, \nu_0\}$ are hyperparameters to be set empirically.

In Figure 12 we report a graphical representation of the mixed-type infinite mixture model and the dependencies between the random variables.

4.3.2 Semi-supervision as partially observable cluster variables

One way to incorporate the partial labels in a probabilistic graphical model is to consider the values as observations of the corresponding hidden variables, i.e. the cluster hidden variable. This approach was introduced by (Seeger, 2006). The approach consists in supposing that if $\mathbf{x}^{(n)}$ is labeled, in its generative process its cluster variable $z^{(n)}$ has to correspond to its label $y^{(n)}$. This approach generalizes quite nicely to our probabilistic model.

In our case, the condition would be:

$$p(\mathbf{x}^{(n)}, z^{(n)} \neq y^{(n)}) = 0 \quad \forall n \in \mathscr{D}_l. \tag{4.10}$$

The subtlety of this condition remains in the inference process. The previous criterion of equation (3.19) no longer holds as a variational lower bound in this case. In the next section, we show how to generalize it to the semi-supervised case, and how to derive the corresponding variational update equations.

4.4 VARIATIONAL INFERENCE FOR THE MIXED-TYPE MODEL

4.4.1 The semi-supervised variational lower bound

The semi-supervision condition of equation (4.10) requires us to revisit the likelihood of our model. Indeed, with this condition the likelihood can be decomposed into two terms:

$$\begin{aligned}\log p(\mathbf{x}^{(1:N)}) &= \sum_{n\in\mathscr{D}_l} \log p(\mathbf{x}^{(n)}) + \sum_{n\in\mathscr{D}_u} \log p(\mathbf{x}^{(n)}) \\ &= \sum_{n\in\mathscr{D}_l} \log p(\mathbf{x}^{(n)}, z^{(n)} = y^{(n)}) + \sum_{n\in\mathscr{D}_u} \log p(\mathbf{x}^{(n)})\end{aligned}$$

The last equality follows from the fact that $\forall n \in \mathscr{D}_l : p(\mathbf{x}^{(n)}, z^{(n)} \neq y^{(n)}) = 0$. Given the decomposition of the log likelihood, the corresponding variational lower bound also involves two terms for the labeled and unlabeled terms.

Let us denote by $\boldsymbol{\zeta} = \{\beta, z^{(1:N)}, \mathbf{M}, \boldsymbol{\Lambda}, \mathbf{B}\}$ the set of all hidden random variables, and $\boldsymbol{\zeta}_{-z^{(1:N)}}$ the set without the z random variables. Similarly to chapter 2, using the Jensen inequality as in equation (2.9), and introducing the variational distribution $q(\cdot)$ over the hidden variables, we have:

$$\begin{aligned}\log p(\mathbf{x}^{(1:N)}) \geq &\sum_{n\in\mathscr{D}_l} \int q(\boldsymbol{\zeta}_{-z^{(n)}}) \log \frac{p(\mathbf{x}^{(n)}, \boldsymbol{\zeta}_{-z^{(n)}}, z^{(n)} = y^{(n)})}{q(\boldsymbol{\zeta}_{-z^{(n)}})} d\boldsymbol{\zeta}_{-z^{(n)}} \\ &+ \sum_{n\in\mathscr{D}_u} \int q(\boldsymbol{\zeta}) \log \frac{p(\mathbf{x}^{(n)}, \boldsymbol{\zeta})}{q(\boldsymbol{\zeta})} d\boldsymbol{\zeta} = \mathcal{L}_{\text{ss}}(q) \qquad (4.11)\end{aligned}$$

The new semi-supervised evidence lower bound $\mathcal{L}_{\text{ss}}$ can be written in the form of a Kullback-Leibler divergence, by simply noticing that the semi-supervision condition can be transferred to the variational distribution, that is: $\forall n \in \mathscr{D}_l : q(z^{(n)} = y^{(n)}) = 1$. We force the labeled instances to also have a variational posterior that respects the labelisation. The semi-supervised evidence lower bound in this case can be re-written as:

$$\begin{aligned}\mathcal{L}_{\text{ss}}(q) &= -\mathbb{D}_{KL}\left[q(\cdot)||p(\cdot, \mathbf{x}^{(1:N)})\right] \\ &\text{s.t} \quad \forall n \in \mathscr{D}_l : q(z^{(n)} = y^{(n)}) = 1 \qquad (4.12)\end{aligned}$$

Since the semi-supervised evidence lower bound can be written as a Kullback-Leibler divergence, we can apply mean-field variational inference in order to find the optimal mean-field distribution maximizing $\mathcal{L}_{\text{ss}}$, and thus fitting the model to the observed data. In the next subsection, we apply the mean-field variational inference approach to derive the update equations of the optimal q.

4.4.2 Mean-field variational inference

As discussed in the previous subsection, optimizing the model to fit the data amounts to finding the optimal variational distribution maximizing the semi-supervised evidence lower bound of equation (4.12). i.e.:

$$q^* = \arg\max_q \mathcal{L}_{\text{ss}}(q) \quad \text{s.t} \quad \forall n \in \mathscr{D}_l : q(z^{(n)} = y^{(n)}) = 1 \tag{4.13}$$

In order to apply the mean-field approach of Appendix A, we need to specify the variational family. For the discrete dimensions, we keep the same variational factorization as the previous chapter. As for the continuous dimensions, we use a standard variational family for Gaussian mixtures, where the mean and precision random variables are not independent (Murphy, 2012). The variational family of the distribution q has the following form over the set of hidden random variables $\{\beta, z^{(1:N)}, \mathbf{M}, \mathbf{\Lambda}, \mathbf{B}\}$:

$$\begin{aligned} q(\{\beta, z^{(1:N)}, \mathbf{M}, \mathbf{\Lambda}, \mathbf{B}\}) = &\prod_{n=1}^{N} q(z_n) \prod_{k=1}^{T} q(\beta_k) \\ &\times \prod_{i \in \mathcal{D}} q(\mathbf{B}_{k,i}) q(\mathbf{M}_k | \mathbf{\Lambda}_k) q(\mathbf{\Lambda}_k) \end{aligned} \tag{4.14}$$

Given the previous variational family, solving equation (4.13) leads to the following parametric variational distributions:

$$\begin{aligned} q^*(z_n) &= \text{Cat}(z_n; \boldsymbol{\phi}_n) \\ q^*(\mathbf{B}_{k,i}) &= \text{Dir}(\mathbf{B}_{k,i}; \boldsymbol{\epsilon}_{k,i,} |\mathbf{x}_i|) \quad \forall i \in \mathcal{D} \\ q^*(\beta_k) &= \text{Beta}(\beta_k; \gamma_{1,k}, \gamma_{2,k}) \\ q^*(\mathbf{M}_k | \mathbf{\Lambda}_k) &= \mathcal{N}(\mathbf{M}_k; \boldsymbol{\mu}_k, (\kappa_k \mathbf{\Lambda}_k)^{-1}) \\ q^*(\mathbf{\Lambda}_k) &= \mathcal{W}(\mathbf{\Lambda}_k; \mathbf{L}_k, \nu_k) \end{aligned} \tag{4.15}$$

Note that the hidden variables corresponding to the discrete dimensions remain unchanged. The variational distribution of the continuous part are simply Gaussian and Wishart distributions of variational parameters $\{\mathbf{L}_k, \nu_k, \boldsymbol{\mu}_k, \kappa_k\}$.

The Variational update equations of the discrete part and beta random variables remain familiar as well:

$$\gamma_{1,k} = 1 + \sum_{n=1}^{N} \phi_{n,k} \quad \gamma_{2,k} = \eta + \sum_{n=1}^{N} \sum_{l=k+1}^{T} \phi_{n,l} \tag{4.16}$$

$$\boldsymbol{\epsilon}_{k,i,v} = \boldsymbol{\alpha}_{i,v} + \sum_{n=1}^{N} \phi_{n,k} \mathbb{1}[\mathbf{x}_i^{(n)} = v] \quad \forall i \in \mathcal{D} \tag{4.17}$$

Algorithm 3 Variational Inference for the mixed type infinite mixture model

Input: $\mathbf{x}^{(1:N)}, T, \eta, \mu_0, \kappa_0, \mathbf{L}_0, \nu_0, \boldsymbol{\alpha}$
Initialize ϕ
$\mathcal{L}_{ss}^{(0)} = -\infty$
for $t = 1...\infty$ **do**
  Compute: $\gamma_{1,k}^{(t)}, \gamma_{2,k}^{(t)}, \kappa_k^{(t)}, \nu_k^{(t)} \quad \forall k \quad (4.16)(4.19)$
  Compute: $\boldsymbol{\epsilon}_{k,i,v}^{(t)} \quad \forall v, \forall k, \forall i \quad (4.17)$
  Compute: $\boldsymbol{\mu}_k, \mathbf{L}_k \quad \forall k \quad (4.18)(4.20)$
  Compute: $\boldsymbol{\phi}_{n,k}^{(t)} \quad \forall n, \forall k \quad (4.21)$
  if $\mathcal{L}_{ss}^{(t)}$ changes by less than 10^{-6} **then**
    break
  end if
end for
return $\boldsymbol{\phi}, \mathbf{L}, \nu, \boldsymbol{\mu}, \kappa, \boldsymbol{\gamma}, \boldsymbol{\epsilon}$

For the variational parameters corresponding to the continuous dimensions, i.e. $\{\mathbf{L}_k, \nu_k, \boldsymbol{\mu}_k, \kappa_k\}$, the update equations are the following:

$$\boldsymbol{\mu}_k = \frac{\kappa_0 \boldsymbol{\mu}_0 + \sum_{n=1}^{N} \boldsymbol{\phi}_{n,k} \mathbf{x}_{\mathcal{C}}^{(n)}}{\kappa_k} \tag{4.18}$$

$$\kappa_k = \kappa_0 + \sum_{n=1}^{N} \boldsymbol{\phi}_{n,k} \quad \nu_k = \nu_0 + \sum_{n=1}^{N} \boldsymbol{\phi}_{n,k} + 1 \tag{4.19}$$

$$\begin{aligned} \mathbf{L}_k^{-1} &= \mathbf{L}_0^{-1} + \kappa_0 (\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)^T \\ &\quad + \sum_{n=1}^{N} \boldsymbol{\phi}_{n,k} (\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k)(\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k)^T \end{aligned} \tag{4.20}$$

Finally, the variational parameters representing the cluster probabilities $\boldsymbol{\phi}$ have a different form, they depend naturally on the continuous and discrete dimensions simultaneously:

$$\begin{aligned} \log \boldsymbol{\phi}_{n,k} = &-\frac{1}{2} \left[\frac{|\mathcal{C}|}{\kappa_k} + \nu_k (\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k)^T \mathbf{L}_k (\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k) \right] \\ &+ \sum_{i \in \mathcal{D}} \sum_{v \in E_i} \mathbb{1}[\mathbf{x}_i^{(n)} = v] [\psi(\boldsymbol{\epsilon}_{k,i,v}) - \psi(\sum_{v' \in E_i} \boldsymbol{\epsilon}_{k,i,v'})] \\ &+ \frac{1}{2} \left[\sum_{i=1}^{|\mathcal{C}|} \psi\left(\frac{\nu_k + 1 - i}{2} \right) + \log \det(\mathbf{L}_k) \right] \\ &+ \psi(\gamma_{1,k}) - \psi(\gamma_{1,k} + \gamma_{2,k}) \\ &+ \sum_{l=1}^{k-1} [\psi(\gamma_{2,l}) - \psi(\gamma_{1,l} + \gamma_{2,l})] + \text{const} \\ &\text{s.t} \quad \sum_{k=1}^{T} \boldsymbol{\phi}_{n,k} = 1 \quad \text{and, } \boldsymbol{\phi}_{n,y_n} = 1 \quad \forall n \in \mathscr{D}_l \end{aligned} \tag{4.21}$$

A proof of the previous equations is given in appendices B.1 and B.2. Given all the variational fixed point equations for the parameters, we need only to iterate them until convergence of $\mathcal{L}_{ss}$ in order to fit the model. Algorithm 3 summarizes the process.

4.4.3 The predictive distribution

Given a trained model through the variational inference method, one needs to make predictions of classes using such model. Formally, given a new instance $\mathbf{x}^{(N+1)}$, one needs to determine the probability that the hidden cluster variable $z^{(N+1)}$ is equal to a certain value k, given all the observations $x^{(1:N+1)}$. Meaning, we need to compute the following probability:

$$p(z^{(N+1)} = k|\mathbf{x}^{(1:N+1)}) \propto p(z^{(N+1)} = k, \mathbf{x}^{(N+1)}|\mathbf{x}^{(1:N)}) \tag{4.22}$$

Computing the previous quantity requires a marginalization over all the hidden variables except $z^{(N+1)}$. Which leads to the follwing computation:

$$\begin{aligned} p(z^{(N+1)} = k|\mathbf{x}^{(1:N+1)}) \propto & \\ & \int p(z^{(N+1)} = k, \mathbf{x}^{(N+1)}, \mathbf{B}, \mathbf{M}, \mathbf{\Lambda}, \beta|\mathbf{x}^{(1:N)}) d\mathbf{M} d\mathbf{\Lambda} d\beta d\mathbf{B} \\ & \propto \int p(\mathbf{x}^{(N+1)}|z^{(N+1)} = k, \mathbf{B}, \mathbf{M}, \mathbf{\Lambda}) \\ & \quad \times p(z^{(N+1)} = k|\beta) p(\mathbf{B}, \mathbf{M}, \mathbf{\Lambda}, \beta|\mathbf{x}^{(1:N)}) d\mathbf{M} d\mathbf{\Lambda} d\beta d\mathbf{B} \end{aligned} \tag{4.23}$$

In order to compute this integral, we need to compute the posterior of the parameters $p(\mathbf{B}, \mathbf{M}, \mathbf{\Lambda}, \beta|\mathbf{x}^{(1:N)})$. Unfortunately, as discussed in previous chapters a closed form of this posterior is intractable. However, variational inference gives us an approximation of this posterior: q^*. Therefore, we can use this approximation to compute the integral (Blei, Jordan, et al., 2006):

$$\begin{aligned} p(z^{(N+1)} = k|\mathbf{x}^{(1:N+1)}) \propto & \int p(\mathbf{x}^{(N+1)}|z^{(N+1)} = k, \mathbf{B}, \mathbf{M}, \mathbf{\Lambda}) \\ & \quad \times p(z^{(N+1)} = k|\beta) q^*(\mathbf{B}, \mathbf{M}, \mathbf{\Lambda}, \beta) d\mathbf{M} d\mathbf{\Lambda} d\beta d\mathbf{B} \\ & \propto \mathbb{E}_{\beta \sim q^*}[\boldsymbol{\pi}_k(\beta)] \prod_{j \in \mathcal{D}} \mathbb{E}_{\mathbf{B}_{k,j} \sim q^*}[\text{Cat}(\mathbf{x}_j^{(N+1)}|\mathbf{B}_{k,j})] \\ & \quad \times \mathbb{E}_{\mathbf{M}_k, \mathbf{\Lambda}_k \sim q^*}[\mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(N+1)}; \mathbf{M}_k, \mathbf{\Lambda}_k)] \end{aligned} \tag{4.24}$$

Furthermore, all the expectations of the previous equation can be computed in closed form (see Appendix B), and lead to the following final form of the predictive distribution:

Algorithm 4 Cluster prediction for future samples

Input: $\mathbf{x}^{(N+1)}, \phi, \mathbf{L}, \nu, \boldsymbol{\mu}, \kappa, \boldsymbol{\gamma}, \boldsymbol{\epsilon}$
Compute: $\boldsymbol{\Theta}_k \quad \forall k \quad (4.26)$
Approximate: $p(z^{(N+1)} = k|\mathbf{x}^{(1:N+1)}) \quad \forall k \quad (4.25)$
$\hat{z}^{(N+1)} = \arg\max_k p(z^{(N+1)} = k|\mathbf{x}^{(1:N+1)})$
return $\hat{z}^{(N+1)}$

$$p(z^{(N+1)} = k|\mathbf{x}^{(1:N+1)}) \propto \frac{\gamma_{1,k}}{\gamma_{1,k}+\gamma_{2,k}} \prod_{l<k} \frac{\gamma_{2,l}}{\gamma_{1,l}+\gamma_{2,l}} \prod_{j\in\mathcal{D}} \left[\frac{\epsilon_{k,j,\mathbf{x}_j^{(N+1)}}}{\sum_{v\in E_j} \epsilon_{k,j,v}} \right] \times \sqrt{\left(\frac{\kappa_k}{\pi(\kappa_k+1)}\right)^p} \frac{\det\left((\mathbf{L}_k^{-1}+\boldsymbol{\Theta}_k^{-1})^{-1}\right)^{\frac{\nu_k+1}{2}} \Gamma_p(\frac{\nu_k+1}{2})}{\det(\mathbf{L}_k)^{\frac{\nu_k}{2}} \Gamma_p(\frac{\nu_k}{2})}, \quad (4.25)$$

where p represents the cardinal of the set $\mathcal{C}$, and $\boldsymbol{\Theta}_k$ is $p \times p$ symmetric non-negative definite matrix defined by:

$$\boldsymbol{\Theta}_k^{-1} = \frac{1}{\kappa_k+1}(\kappa_k\boldsymbol{\mu}_k + \mathbf{x}_{\mathcal{C}}^{(N+1)})(\kappa_k\boldsymbol{\mu}_k + \mathbf{x}_{\mathcal{C}}^{(N+1)})^T + \kappa_k\boldsymbol{\mu}_k\boldsymbol{\mu}_k^T + \mathbf{x}_{\mathcal{C}}^{(N+1)}\mathbf{x}_{\mathcal{C}}^{(N+1)^T} \quad (4.26)$$

A detailed derivation of the previous equation is given in Appendix B. Following from the previous computation, predicting the cluster of the $N+1$ sample is simply done by taking the value of k representing the maximum of the probability. Algorithm 4 summarizes the process.

4.5 EXPERIMENTS AND RESULTS

In this section, we demonstrate the ability of the semi-supervised mixed-type mixture model to learn to classify faults in a semi-supervised fashion.

4.5.1 Experimental setup and baselines

Our experimental setup in this chapter is the following, we consider the real-world dataset presented in chapter 3. This dataset is comprised of categorical variables (Alarms, status of equipment,...), and continuous variables (transmission and reception powers, bit error rates, ...). The categorical variables are treated in the same way as the previous chapter. As for the continuous variables, we scale their values between 0 and 1 using min-max scaling. We use the labeled part of the dataset. The evaluation metrics

reported are computed over a held-out 10 % of the overall dataset. We use five cross-validation runs, and we report the mean and standard deviation of each metric.

In the previous chapter we showed how we can cluster patterns of faults in an unsupervised manner. In this chapter, we evaluate the semi-supervised classification capability of our model compared to some state-of-the-art baselines. The baselines considered are some of the best performing semi-supervised learning methods, namely, Label Propagation (Zhur and Ghahramani, 2002) and Transductive Support Vector Machines (TSVM) (Joachims, 1999).

4.5.2 Evaluating convergence of the model

Same as the previous chapter the first task is to ensure that the update equations of the variational inference are correct and lead to model convergence, we report the semi-supervised evidence lower bound of equation (4.12) across updates of the variational parameters. As shown in Figure 13, the evidence lower bound increases monotonically and converges to a plateau, which indicates that the equations derived are correct and that the model is well fitted.

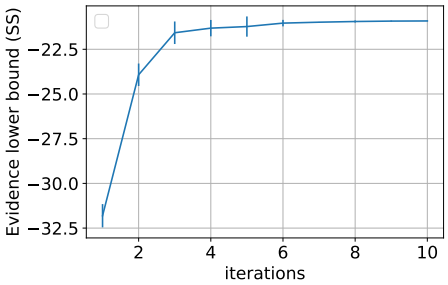

Figure 13: semi-supervised evidence lower bound across variational inference iterations.

4.5.3 Evaluating the classification accuracy

Second, we evaluate the classification accuracy of our model on the held-out datasets. We report the evolution of the classification accuracy across iterations. Same as the evidence lower bound the classification accuracy should increase and converge to a stable value. As shown in Figure 14, it is the case, the accuracy converges steadily over all runs to a value of about **94** % corresponding to **6** % error rate on the test set.

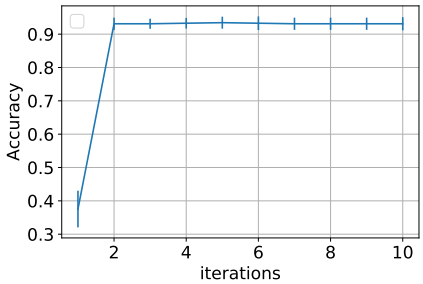

Figure 14: Accuracy of the model across variational inference iterations.

Finally, we evaluate our model in comparison with the considered baselines, Label Propagation and Transductive Support Vector Machines (TSVM)

Table 4: Comparison between our model and the baselines.

Methods	Label Propagation	TSVMs (Linear)	TSVMs (RBF)	**Ours**
Accuracy	0.95 ± 0.01	0.59 ± 0.02	0.62 ± 0.04	$\mathbf{0.94 \pm 0.01}$

with Linear and Gaussian kernels. In Table 4 we report the results of the comparison. The model is competitive with label propagation, with it scoring slightly better (one percent better) than our model on the dataset. Linear and Gaussian TSVMs perform much poorly compared to our model and label propagation. This could be a result of the assumptions behind the Transductive SVM. There is no guarantee that the classes in the data considered are separable linearly, or in the space generated by the kernel.

4.6 DISCUSSION & PRIOR WORK

In the semi-supervised learning literature, two approaches to assign classes to samples with prior knowledge have been extensively studied. The first approach supposes that all the different classes are present in the labeled part of the dataset, thus the number of classes is known. The model then attempts to propagate the labels to the unlabeled set, by respecting some notion of similarity. The previously introduced methods Transductive Support Vector Machines (Joachims, 1999) and label propagation (Bengio, Delalleau, and Le Roux, 2006) and (Zhur and Ghahramani, 2002) belong to this first approach. The Transductive Support Vector Machine model leverages the unlabeled data in the learning of the decision boundary between classes unlike classical support vector machines where only the labeled data is used. Label propagation, on the other hand, is a graph based approach where the labeled and unlabeled data points represent nodes of the graph, and edges represent similarities between the nodes. The known labels are then propagated through the graph to label the unlabeled nodes. However, in some real world applications of data mining and information retrieval, the number of possible clusters is unknown and not all clusters are partially labeled, making the utility of such methods limited.

The second approach is based on the classical KMeans clustering algorithm, where constraints are added in order to guide the clustering, and improve the performance (Wagstaff et al., 2001). The constraints are constructed from the partial labels, must-link constraints assure that two samples must have the same label, and cannot-link constraints assure that two samples shouldn't be labeled the same. (Basu, Banerjee, and Mooney, 2004) proposed a Hidden Markov Random Field (HMRF) formulation of this model and later introduced distance learning to identify relevant features

to the clustering (Basu, Bilenko, and Mooney, 2004). In this approach it is possible to find a hidden cluster not present in the partial labels. However, because it is based on the KMeans algorithm, the number of clusters must be known or some sort of model selection needs to be applied to estimate it. For a detail review of these approaches, we refer the reader to (Chapelle, Scholkopf, and Zien, 2009).

By comparison, our approach does not suppose a fixed number of clusters but rather learns it directly from the data. More importantly, the modeling assumption are in accordance with the data types and structure, categorical variables remain as such and continuous variables are modeled as such. Thus, no distance learning is required to evaluate relevant features. In addition, the model in its semi-supervised form can adjust the parameters to the known labels of faults, however, if a new fault pattern emerges the Dirichlet Process mixture part automatically creates a new cluster for it.

4.7 LIMITATIONS

Although the infinite mixed-type semi-supervised mixture model can successively model the known labels of faults, and through its Dirichlet process mixture cluster new patterns, it remains that the model only functions correctly if the necessary information is directly correlated to the cluster hidden variable. Under this assumption, there is no need to create new features from the data in order to classify or cluster the faults correctly. The previous setting leads to good results, in highly specific applications with low dimensional datasets, that can be heavily preprocessed. In most real-world applications, we can collect hundreds if not thousands of variables and even with thorough preprocessing, noise remains, making shallow models perform poorly. In order to reach good performance in this case, one needs to use models capable of creating latent features relevant to each class, which naturally leads us to deep learning based methods.

4.8 CONCLUSION

In this chapter, we extended the Infinite Mixture Model to the semi-supervised case with mixed type variables (continuous and categorical). We showed how to learn the model using variational inference, and how to make predictions using the model. In the next part of the manuscript, we discuss how to enlarge the scope of infinite mixture models to include feature learning using deep learning methods. These methods as we will show are necessary to deal with high dimensional noisy datasets, that require good feature creation to classify and identify patterns of faults.

Part II

DEEP PROBABILISTIC GRAPHICAL MODELS FOR FAULT IDENTIFICATION

5

DEEP INFINITE MIXTURE MODELS FOR FAULT DISCOVERY

5.1 INTRODUCTION

In chapters 3 and 4, we have introduced infinite mixture models for fault clustering and classification from network data. We generalized the approach to deal with categorical and continuous hidden variables in the presence of expert knowledge in the form of labels. A major limitation however of the previous approach is its inability to create new, and possibly more informative features for the classification task. In this chapter, we discuss how to combine the infinite mixture model approach with deep learning methods in order to solve this problem. We call this approach Deep Infinite Mixture Models.

Deep learning methods have demonstrated tremendous capability of learning representative features for classification tasks, especially for unstructured data such as images (Krizhevsky, Sutskever, and Hinton, 2012). Thanks to the ever increasing size of datasets that are labeled, deep learning can use this massive scale of data to reach good performance on a number of tasks ranging from classical supervised learning to reinforcement learning. Extending the paradigm to network data is very appealing, network data is often labeled through expert systems, is massive in terms of dimensionality and number of instances, thus deep learning methods can be applied to extract features in this domain as well.

The challenge of the diagnosis task in telecommunication networks is the discovery or clustering task. It is not sufficient to classify correctly known faults, but we need to also discover new patterns of faults. The literature of deep learning is still lacking in this task, and the subject is still widely unexplored and an active area of research. The closest deep learning paradigm is Zero-Shot Learning (ZSL), where some classes are known and others are hidden (Xian, Schiele, and Akata, 2017). However, in most applications of zero-shot learning extra semantic information is

available (for example, a description of an image in words accompanying the image). This makes these approaches inapplicable for the diagnosis task, due to the fact that such information is often not available (Larochelle, Erhan, and Bengio, 2008; Socher et al., 2013).

In this chapter, we propose an approach based on deep neural networks and infinite Gaussian Mixture Models (GMM). Our approach combines the power of neural networks, and the exploration potential of infinite mixture models. The neural network identifies useful features for the fault discovery process (based on the labeled faults), then the infinite mixture model clusters the data based on the neural network representation. Thus, the fault discovery process is done on a more representative space of the faults.

We demonstrate our approach on data derived from GPON-FTTH networks, we show that the neural network is capable of identifying relevant features for the diagnosis task. Furthermore, we show that the clustering approach automatically identifies clusters of relevant faults. The remainder of the chapter is organized as follows:

- In section 5.2, we frame the problem of discovering new types of faults as a deep learning problem and we give an overview of the overall model.
- In section 5.3, we detail the feature extraction task as a deep learning classification task.
- In section 5.4 we introduce the infinite Gaussian mixture model for clustering new faults based on the extracted features.
- Finally, in section 5.5, we show how our approach can explore an unlabeled part of a more complex GPON-FTTH dataset in order to identify new types of faults that were previously considered as unidentified faults by an expert system.

5.2 PROBLEM DESCRIPTION & MODEL OVERVIEW

5.2.1 Problem statement

Discovering and diagnosing faults from network data is extremely challenging due to three key properties of large telecommunication networks:

- Scale: when dealing with large telecommunication networks, the scale of the data in terms of dimensionality and the number of instances adds to the complexity of the diagnosis system, and restricts further the approaches that one can adopt to solve the diagnosis problem.

- Novelty: a second issue is the adaptability of the diagnosis system. Given a dynamic corpus of data gathered from the network, the system needs to identify previously known faults if they occur, in addition to clustering and identifying new patterns of faults.
- Noise: when dealing with network data, not all features and data dimensions will be relevant for a specific fault, hence adding noise to the relevant information. Therefore, an efficient diagnosis system needs to filter out the noise in some way in order to identify relevant features for the fault in question.

A learning-based efficient diagnosis system thus needs to identify and learn previously known patterns of faults, discover new patterns unknown to the experts or the expert system, and finally identify patterns relevant to actual faults on the network.

5.2.2 Data specification

The first challenge encountered when dealing with network data is the multi-typed nature of variables. For example, when treating data gathered from the optical network, one finds continuous variables such as power levels, and variables taking discrete, categorical, or binary values such as alarms of some specific device. In order to create the data corpus, we start by normalizing the continuous variables by subtracting the minimum and then scaling the values (dividing by the maximum minus the minimum value). For the categorical variables, we use the one-hot encoding i.e. if a variable X_i takes values in a set $\{c_1, ..., c_K\}$, then we create a representation in $\{0,1\}^K$, for which the k^{th} column is equal to 1 if the variable X_i is equal to c_k. The resulting training data $\mathbf{X} \in \mathbb{R}^{N \times D}$ contains N lines, each representing a customer's data vector, and D columns, each representing a variable of the network. Given the large amount of data gathered D is often very high ($D >> 1$, with often hundreds of variables). In addition to the data vector, for some customers, expert systems have identified the corresponding fault, thus providing us with a label. We denote these labels $\mathbf{y} \in \mathbb{R}^N$, each line represents the label given to a customer. A label $\mathbf{y}_n$ takes values in $\{-1, 1, ..., K\}$, where $\{1, ..., K\}$ represent the known classes, and the value -1 represents an unknown class, as a convention. The key problem is to identify the labels of the class -1. This label could belong to the set of existing known faults (i.e. to the set of known classes $\{1, ..., K\}$) or a completely new set of faults occurring in the network. For what follows, we will denote $\mathbf{x}_n \in \mathbb{R}^D$ the data vector corresponding to the n^{th} customer, and suppose that the samples are independent and identically distributed.

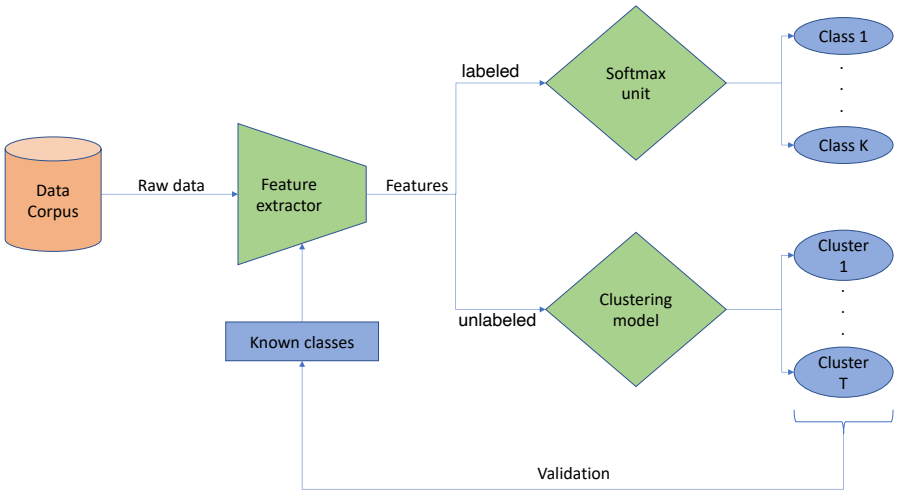

Figure 15: Overview of the diagnosis system.

5.2.3 Overview of the model

As shown in Figure 15, the diagnosis model is composed of three main blocks. The first one is the feature extractor, which will be a multi-layer perceptron or a neural network, responsible for compressing the data into a low dimensional feature vector containing the relevant information. The second one is the softmax unit responsible for classifying the known faults, its output is fed to the loss function in order to train the feature extractor. The final block is the clustering model responsible for identifying new patterns of faults based on the feature vector extracted.

The feature extractor represents the backbone of the system. Learning a good feature extractor is crucial for the performance of the diagnosis system. The feature extractor allows us to solve the scale and noise challenges simultaneously. The scale problem is solved by compressing the high dimensional raw data vector into a compact low dimensional manifold. The learned low dimensional manifold holds the information to correctly classify the known classes, thus creating noiseless features containing only the important information for each class.

The novelty challenge is tackled by the clustering model, which is in our case an infinite Gaussian mixture model. Based on learned features, the clustering model tries to identify new patterns or clusters of customer features, each cluster corresponding to a new unknown class. Then a final process of validation is done by an expert to identify relevant or correct new fault clusters. These correct clusters are then treated as labeled classes, the -1 labels being changed into positive values representing the new known classes. These new known classes are then re-fed to the feature extractor. In the following sections, we detail the model architectures of each block.

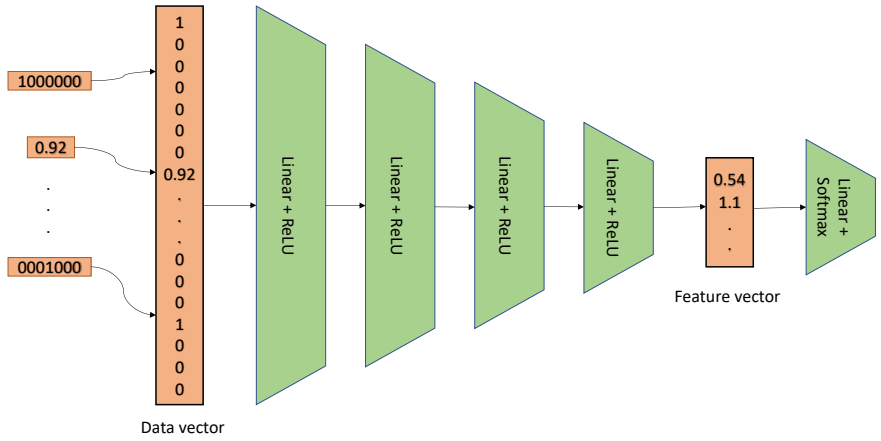

Figure 16: Architecture of the feature extractor followed by the softmax unit.

5.3 FEATURE EXTRACTION & CLASSIFICATION

5.3.1 Feature extractor architecture

The feature extractor in our case is a deep feed forward neural network, that takes as input a data vector $\mathbf{x}_n \in \mathbb{R}^D$ and outputs a low dimensional feature vector denoted $\mathbf{f}_n \in \mathbb{R}^p$, where $p << D$ represents the dimension of the feature manifold. As a remainder, we can think of a neural network as a nonlinear function, built by successive applications of linear layers and nonlinear activation functions. Formally, we denote the overall parameters of the neural network θ, and:

$$\begin{aligned} \mathbf{f}_n &= g_\theta(\mathbf{x}_n) \\ &= \sigma \circ f^{(L-1)} \circ \ldots \circ \sigma \circ f^{(2)} \circ \sigma \circ f^{(1)}(\mathbf{x}_n) \end{aligned} \tag{5.1}$$

where, each function $f^{(l)}$ represents the l^{th} application of a linear or fully connected layer, defined by:

$$f^{(l)}(\mathbf{h}) = \mathbf{h}^T \mathbf{W}_l + \mathbf{b}_l \tag{5.2}$$

$\mathbf{W}_l$, and $\mathbf{b}_l$ represent the weight matrix and bias vector for the l^{th} layer. $\circ$ represents the composition of functions operator. Between each application of two linear layers, a non linear function is applied, denoted by σ, representing the ReLU activation function defined by

$$\sigma(\mathbf{h}) = \max(0, \mathbf{h}) \tag{5.3}$$

where the max is taken term by term.

For our application, we consider a neural network with 4 hidden layers ($L-1=4$), where the dimension of each layer is about half of that of the previous layer. The previous choices are based on common practices in the deep learning community. The choice of the number of layers is done by cross-validation, where we choose the lowest number of hidden layers for which the classification error rate does not change. An overview of the architecture of the feature extractor is given in Figure 16. We detail the loss function and the learning process in the next subsection.

5.3.2 Learning features by classification

In order to learn the parameters of the feature extractor, we define a classification task based on the K known classes. Given the feature vector of a customer denoted by $\mathbf{f}_n$, the softmax unit outputs the probability of each customer being in class k as:

$$\mathbb{P}[\mathbf{y}_n = k|\mathbf{f}_n] = \frac{e^{\mathbf{f}_n^T \mathbf{W}_{L,k} + \mathbf{b}_{L,k}}}{\sum_{j=1}^{K} e^{\mathbf{f}_n^T \mathbf{W}_{L,j} + \mathbf{b}_{L,j}}}. \tag{5.4}$$

where $\mathbf{W}_L, \mathbf{b}_L$ are the weights and biases of the last layer of the whole neural network (feature extractor followed by the softmax unit). The optimization of the parameters of the neural network is done by minimising the cross-entropy loss function defined as:

$$\mathscr{C}(\boldsymbol{\theta}) \;=\; -\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{1}[\mathbf{y}_n = k] \log \mathbb{P}_\theta[\mathbf{y}_n = k|\mathbf{f}_n], \tag{5.5}$$

where θ represents the set of weights of the neural network. Learning θ is done by gradient descent and the standard backpropagation algorithm (Goodfellow et al., 2016). Following the previous learning process, we learn a feature extractor and a classifier of the known classes $\{1, ..., K\}$. In the next section, we show how to identify outlier clusters representing new patterns of faults.

5.4 CLUSTERING MODEL

In the clustering process, the goal is to identify new patterns or clusters of unseen faults in unlabeled data (i.e. data with -1 label). The key trick in our method consists in applying the clustering process to the feature manifold. The assumption is that the feature manifold is a more representative space in terms of filtered noise and relevant features than the original data space. Furthermore, applying the clustering process (step 3 of Algorithm 5) in the

Algorithm 5 Deep Infinite Gaussian Mixture Model

Input: X, y
Initialize parameters
 # Step 1: Training of the feature extractor on labeled data
compute $\boldsymbol{\theta}^*$ by minimizing (5.5)
 # Step 2: Feature computation of unlabeled data
compute $\mathbf{f}_n$ using the last hidden layer
 # Step 3: Clustering process on unlabeled data
compute q^* by minimizing (5.7)
return $\hat{\mathbf{z}}_n = \arg\max_k q^*(\mathbf{z}_n = k) \quad \forall n$

low dimensional feature manifold reduces considerably the complexity and execution time of the approach.

Given that the neural network learned highly expressive features for the known classes, it is highly likely that all information not relevant to the diagnosis task has been filtered. Furthermore, the low dimensional space of the last hidden layer is separated linearly with respect to each known class. Thus, outliers and new clusters should exist in a region outside the class regions.

The previous assumptions and deductions give guidance for the choice of the clustering model. The linearly separable nature of the feature space suggests that a simple Gaussian mixture model should be sufficient in order to identify clusters in it. Coupled with a Dirichlet process the Infinite Gaussian Mixture model can also learn the number of clusters directly without specifying it in advance, in the same manner as in the previous chapters.

5.4.1 Dirichlet Process Gaussian Mixture Model

We adopt the same formulation of the infinite Gaussian mixture as in chapter 4. The only exception is that the observable random variables are no longer the data samples, but the feature vector representation given by the neural network. The generative process is thus the following:

$$
\begin{aligned}
\forall k \quad \beta_k &\sim \text{Beta}(1, \eta) \\
\pi_k &= \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) \\
\boldsymbol{\Lambda}_k &\sim \mathcal{W}(\cdot; \mathbf{L}_0, \nu_0) \\
\boldsymbol{\mu}_k | \boldsymbol{\Lambda}_k &\sim \mathcal{N}(\cdot | \mathbf{m}_0, (\kappa_0 \boldsymbol{\Lambda}_k)^{-1}) \\
\mathbf{z} | \boldsymbol{\pi} &\sim \text{Cat}(\cdot | \boldsymbol{\pi}) \text{ i.e. } \mathbb{P}[\mathbf{z} = k] = \pi_k \\
\mathbf{f}_n | z_n = k, \boldsymbol{\mu}, \Lambda &\sim \mathcal{N}(\cdot | \boldsymbol{\mu}_k, \Lambda_k^{-1})
\end{aligned}
\tag{5.6}
$$

Where, $\{z_{1:N}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \beta\}$ represent the hidden random variables to infer, and $\mathbf{f}_{1:N}$ represent the feature vector of the observable random variables.

5.4.2 Variational Inference for the DPGMM

In the same manner as in the previous chapters, we can apply mean-field variational inference to perform inference of the hidden random variables given the observations, using the following minimization criterion:

$$q^* = \arg\min_q \mathbb{D}_{KL}\left[q||p(\cdot, \mathbf{f}_{1:N})\right] \tag{5.7}$$

The optimal variational distributions have a familiar form, same as the last chapter given by:

$$\begin{aligned} q^*(\beta_k) &= \text{Beta}(\beta_k; \gamma_{1,k}, \gamma_{2,k}) \\ q^*(\boldsymbol{\mu}_k|\boldsymbol{\Lambda}_k) &= \mathcal{N}(\boldsymbol{\mu}_k; \mathbf{m}_k, (\kappa_k \boldsymbol{\Lambda}_k)^{-1}) \\ q^*(\boldsymbol{\Lambda}_k) &= \mathcal{W}(\boldsymbol{\Lambda}_k; \mathbf{L}_k, \nu_k) \\ q^*(z_n) &= \text{Cat}(z_n; \phi_n) \end{aligned} \tag{5.8}$$

Where the variational parameters are updated iteratively using the following equations:

$$\gamma_{1,k} = 1 + \sum_{n=1}^{N} \phi_{nk} \quad \gamma_{2,k} = \eta + \sum_{n=1}^{N} \sum_{l=k+1}^{T} \phi_{nl} \tag{5.9}$$

$$\kappa_k = \kappa_0 + \sum_{n=1}^{N} \phi_{nk} \quad \nu_k = \nu_0 + \sum_{n=1}^{N} \phi_{nk} + 1 \tag{5.10}$$

$$\mathbf{m}_k = \frac{\kappa_0 \mathbf{m}_0 + \sum_{n=1}^{N} \phi_{nk} \mathbf{f}_n}{\kappa_k} \tag{5.11}$$

$$\begin{aligned} \mathbf{L}_k^{-1} &= \mathbf{L}_0^{-1} + \kappa_0 (\mathbf{m}_k - \mathbf{m}_0)(\mathbf{m}_k - \mathbf{m}_0)^T \\ &+ \sum_{n=1}^{N} \phi_{nk} (\mathbf{f}_n - \mathbf{m}_k)(\mathbf{f}_n - \mathbf{m}_k)^T \end{aligned} \tag{5.12}$$

$$\begin{aligned} \log \phi_{nk} &= -\frac{1}{2}\left[\frac{d}{\kappa_k} + \nu_k (\mathbf{f}_n - \mathbf{m}_k)^T \mathbf{L}_k (\mathbf{f}_n - \mathbf{m}_k)\right] \\ &+ \frac{1}{2}\left[\sum_{i=1}^{d} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + \log\det(\mathbf{L}_k)\right] \\ &+ \psi(\gamma_{1,k}) - \psi(\gamma_{1,k} + \gamma_{2,k}) \\ &+ \sum_{l=1}^{k-1} \left[\psi(\gamma_{2,l}) - \psi(\gamma_{1,l} + \gamma_{2,l})\right] \end{aligned} \tag{5.13}$$

The details of the computation of q^* are given in appendix A. A complete overview of the learning process is given in Algorithm 5.

5.5 EXPERIMENTAL EVALUATIONS

5.5.1 Dataset description

We evaluate our approach on real network data. The dataset is built from the technical data for 86241 fiber customers. In one part of the dataset, thanks to a legacy diagnosis system (in our case an expert system based on deterministic rules), the customers have been classified into 8 known types of faults (including a normal behavior) as described in Table 5. This labelled sub-dataset corresponds to 64279 customers. Another part of the dataset (21962 customers) cannot be classified by the legacy diagnosis system, and the corresponding customers thus fall into the "unknown faults" category.

Fault type	# instances (N)
FTTH access OK	19604
Fiber cut	18282
Degraded fiber	6906
ONT-home gateway problem	6782
Gateway update problem	6050
Bad gateway configuration	3732
Account suspended	1527
TV problem	1396
Unknown Faults	21962
TOTAL	86241

Table 5: Description of the different classes in the dataset.

For each customer, we collected 1824 features coming from different parts of the network that describe the state of the customer's line. These variables mainly characterize the properties of the FTTH GPON optical system, the Home Gateway (HG), some services used by the customer (e.g. TV, VoIP) and the Internet session (see Table 6). For instance, we have a set of variables describing the GPON properties such as the optical powers received (Rx) or transmitted (Tx) by the Optical Line Termination (OLT) at the central office, and by the Optical Network Termination (ONT) at the customer side, in addition to the properties of the ONT and OLT (temperature, voltage, version etc.), the alarms encountered by the OLT, and so on.

Network scope	# variables	Description
Wide Area Network	652	GPON (OLT, ONT...)
Home LAN	446	Gateway, home devices
Internet Session	256	DHCP
Services	204	VoIP, TV
Offer	41	Customer offer profile
Miscellaneous	225	-
TOTAL	1824	
	8297	(after One-Hot Encoding)

Table 6: Overview of the network variables in the dataset.

The dataset is thus composed of a mix of categorical variables and numerical variables. The numerical features are normalized in $[0, 1]$ using min-max scaling. The categorical variables are encoded using the one-hot encoding, as explained in section 5.2, in order to obtain a numerical dataset. Thus the final dataset is composed of $N = 86241$ instances and $D = 8297$ variables.

In order to evaluate the classification task on the known classes, the labeled dataset is split into a training set and a test set, where each represent 80% and 20% respectively. The metrics for evaluations are reported over 5

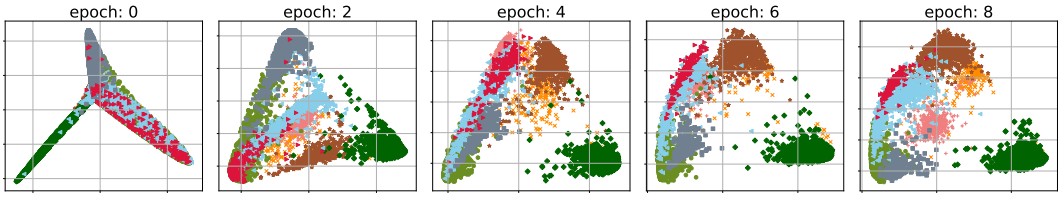

Figure 17: 2-D representation of the feature manifold across epochs of training.

cross validation runs, we report the mean and the standard deviation for each metric. The clustering evaluation is done on the entire unknown fault dataset in a single run.

5.5.2 Evaluation of the Feature Extractor

The first evaluation is that of the neural network representing the feature extractor. Extracting relevant features for the known classes is crucial for a good classification score, and for efficiently filtering noisy features for the subsequent clustering task. The key desirable feature of the hidden data manifold represented by the last hidden layer of the feature extractor is to be separable for each known class. In order to represent this data manifold, we compute the hidden features on the test set after each epoch (full pass of stochastic gradient descent on the training set), we then transform the features into a 2D representation using kernel principal component analysis. This procedure shows plots of the hidden feature points in a 2D representation based on a Gaussian proximity measure.

As shown in Figure 17, the hidden representation is disorganised at the first epochs of training the neural network, the classes are not well separated based on the similarity measure. However, as we converge to the optimum parameters, the hidden feature manifold becomes more separable with respect to each class (epoch 8). This suggests that the features extracted are representative of each class modeled. In addition, the structure is in the form of "lumps" which is coherent with Gaussian model assumptions.

In order to demonstrate the effectiveness of the feature extractor further, we compare the raw two dimensional representation on the unknown faults dataset, and the two dimensional representation of the features extracted using the same dataset. As shown in Figure 18, the 2D representation on the extracted features is much more organized than the 2D representation on the raw data. Furthermore, the 2D representation of the features shows clear patterns of clusters

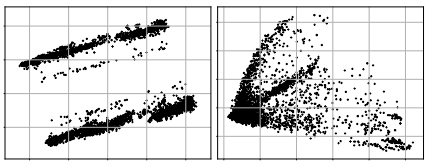

Figure 18: (Left) 2-D representation in the feature space. (Right) 2-D representation of the raw data.

Table 7: The confusion matrix on the known classes.

True classes \ Predicted classes	1	2	3	4	5	6	7	8
1	**1288**	96	10	39	63	13	14	4
2	47	**3619**	0	0	0	45	14	7
3	0	0	**6024**	3	12	5	2	4
4	0	1	5	**18590**	0	304	621	83
5	1	1	8	1	**18258**	7	6	0
6	1	5	9	226	0	**6580**	74	11
7	2	0	0	56	1	10	**6710**	3
8	1	0	3	22	0	2	4	**1364**

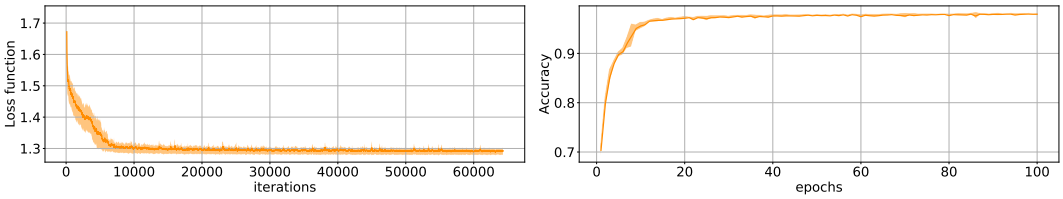

Figure 19: Loss function on the test set across iterations of gradient descent. Accuracy on the test set across epochs.

unlike the raw data representations. This suggests that the feature extractor is capable of identifying features that permit us to separate and identify different clusters in the unknown fault dataset.

5.5.3 Evaluation of the Softmax Unit

The softmax unit classifies the known faults. As a standard evaluation process we report the loss function (5.5) on the test set across training iterations. We report the mean and standard deviation over 5 different runs. As shown in Figure 19 (left) the loss function converges to the set of parameters representing of the minimum of (5.5).

In addition, we report the classification accuracy on the test set. As shown in Figure 19 (right) the classification accuracy increases across training iterations and reaches a high value, showing that the classifier is well trained and robust to errors. This is demonstrated also by the confusion matrix between the known classes, reported in table 7. The standard deviation across the different validation runs is very small, suggesting that the classifier generalizes very well on unseen test set samples.

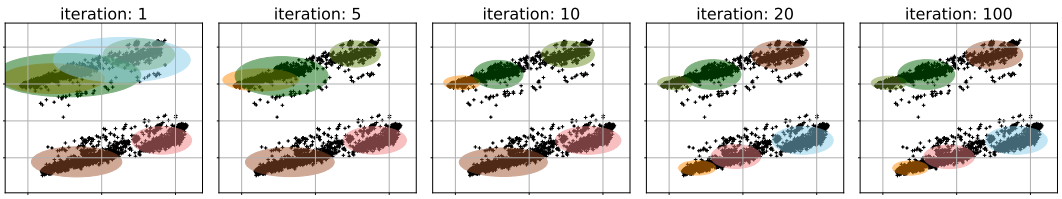

Figure 21: The clustering process of the infinite Gaussian mixture model on the latent feature manifold across iterations of variational inference.

5.5.4 Evaluation of the Clustering Model

In order to evaluate the clustering model, we begin by evaluating the fitting of the model to the unknown fault data. The latent infinite Gaussian mixture model is a probabilistic generative model, a standard approach of evaluating such models is to compute the log likelihood or the log probability of the data under the model across iterations. As shown in Figure 20, the log probability increases across iterations of updates of variational inference, and converges smoothly to a plateau, which suggests the convergence of the model.

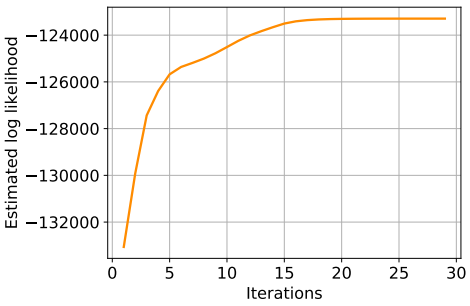

Figure 20: Log probability of mixture across iterations.

The objective of the infinite latent Gaussian mixture model is to cluster patterns in the latent representation of unknown faults. In order to evaluate this clustering process, we report the clusters identified by the model in the 2D representation, across iterations of variational inference updates. As shown in Figure 21, the model starts with random clusters that do not fit the patterns in the unknown faults dataset. Across training iterations the ellipsoids representing the clusters fit with more accuracy the patterns seen in the 2D representation.

Furthermore, using the Dirichlet Process which learns the number of clusters, we see that across iterations of training of the clustering model, the number of clusters changes, in order to evaluate which number of clusters better fits the data. Thus, the model automatically identifies the number of clusters needed to converge to the optimal fit.

5.6 LIMITATIONS

We have discussed how to combine the power of deep learning representations and the exploration capacity of Infinite Mixture Models, in order to

discover faults in large noisy datasets. There are obvious limitations of the proposed combination, first, in the case where new faults are discovered a retraining process has to be re-implemented on the neural network side, which could prove expensive. Second, the unlabeled part only serves in the clustering part not the training of the deep model, thus wasting this unlabeled information that could prove useful for better performance and generalization. And last, the model is naturally sequential thus even the clustering process has to be repeated each time new labels are introduced. A natural answer to all these concerns, would be to consider deep generative models, where the feature learning and clustering are simultaneously done, making the model trainable end-to-end.

5.7 CONCLUSION

In this chapter, we have introduced the Deep Infinite Mixture Model combining deep learning representations and Infinite mixture models in order to explore and identify faults in noisy datasets. The proposed model is sequential and composed of multiple bricks, making training and inference expensive. A natural extension can be proposed using deep generative models, where intermediate hidden random variables would represent the high level features necessary for the clustering and classification task. However, a major challenge in training deep generative models using variational inference, lies in the difficulty of backpropagating gradients through multiple layers of hidden random variables, with nonlinear activations. Specifically, the estimation of gradients of functions of the form $\mathcal{L}(\theta) := \mathbb{E}_{\mathbf{z}\sim q_\theta}[\log p(\mathbf{z})]$ is very difficult in deep generative models. Classical estimators often lead to high variance and make the learning of these models unstable. In the next chapter, we present an original approach developed in the thesis, to compute low variance estimates of these gradients.

6

GENERALIZED STOCHASTIC BACKPROPAGATION

6.1 INTRODUCTION

In the previous chapter, we discussed how to combine neural networks and mixture models to learn and cluster faults based on highly representative features of neural networks. A natural extension is to consider an end-to-end deep generative model that models deep hidden features and cluster assignments simultaneously. The general formalism for these models are what is called deep belief networks (Mnih and Gregor, 2014; Neal, 1992), which are mainly deep neural networks with stochastic neurons instead of deterministic neurons. These models have become crucial in multiple domains, such as generative modeling (Kingma and Welling, 2013; Mnih and Gregor, 2014; Rezende, Mohamed, and Wierstra, 2014), deep reinforcement learning (Sutton et al., 2000), and attention mechanisms (Mnih, Heess, Graves, et al., 2014).

Training these models amounts to performing gradient descent on functions of the form $\mathcal{L}(\theta) := \mathbb{E}_{\mathbf{z}\sim q_\theta}[f(\mathbf{z})]$ with respect to the parameters θ. The difficulty encountered in training these models arises in the computation of gradients of $\mathcal{L}(\theta)$, which requires to backpropagate the gradient through the random variable $\mathbf{z}$. One of the first and most commonly used methods is the score function or reinforce method (Glynn, 1989; Williams, 1992), that requires the computation and estimation of the derivative of the log probability function. For applications with high dimensionality however, it has been noted that reinforce gradients have high variance, making the training process unstable (Rezende, Mohamed, and Wierstra, 2014).

Recently, significant progress has been made in tackling the variance problem. The first class of approaches that deals with continuous random variables are reparameterization tricks. In that case a standardization function is introduced, that separates the stochasticity from the dependency on the parameters θ. This allows to transport the derivative inside the

expectation and to sample from a fixed distribution, resulting in low variance gradient (Figurnov, Mohamed, and Mnih, 2018; Kingma and Welling, 2013; Naesseth et al., 2017; Rezende, Mohamed, and Wierstra, 2014; Ruiz, AUEB, and Blei, 2016; Titsias and Lázaro-Gredilla, 2014). The second class of approaches concerns discrete random variables, for which a direct reparameterization is not known. The first solution uses the score function gradient with control variate methods to reduce its variance (Gu et al., 2016; Mnih and Gregor, 2014). The second consists in introducing a continuous relaxation admitting a reparameterization trick of the discrete random variable, thus being able to backpropagate low-variance reparameterized gradients by sampling from the concrete distribution (Grathwohl et al., 2018; Jang, Gu, and Poole, 2016; Maddison, Mnih, and Teh, 2016; Tucker et al., 2017).

Although recent developments have advanced the state-of-the-art in terms of variance reduction and performance, stochastic backpropagation (i.e. computing gradients through random variables) still lacks theoretical foundation. In particular, the following questions remain open: How to develop stochastic backpropagation rules, where the derivative is transferred explicitly to the function f for a broader range of distributions? And can the discrete and deterministic cases be interpreted in the sense of stochastic backpropagation? In this chapter, we provide a new method to address these questions, and our main contributions are the following:

- We present a theoretical framework based on the link between the multivariate Fourier transform and the characteristic function, that provides a standard method for deriving stochastic backpropagation rules, for **any** distribution of practical interest discrete or continuous.
- We show that deterministic backpropagation can be interpreted as a special case of stochastic backpropagation, where the probability distribution q_θ is a Dirac delta distribution, and that the discrete case can also be interpreted as backpropagating a discrete derivative.
- We generalize previously known estimators, and provide new stochastic backpropagation rules for the special cases of the Laplace, gamma, beta, and Dirichlet distributions.
- We demonstrate experimentally that the resulting new estimators are competitive with state-of-the art methods on simple tasks.

6.2 BACKGROUND & PRELIMINARIES

Let (E, λ) be a d-dimensional measure space equipped with the standard inner product, and f be a square summable positive real valued function on E, that is, $f\colon E \to \mathbb{R}_+$, with $\int_E |f(z)|^2 \lambda(dz) < \infty$. Let q_θ be an arbitrary parameterized probability density on the space E. We denote by φ_θ its

characteristic function, defined as: $\varphi_\theta(\omega) := \mathbb{E}_{\mathbf{z}\sim q_\theta}[e^{i\omega^T \mathbf{z}}]$. We denote by $\hat{f}$ the Fourier transform of the function f defined as:

$$\hat{f}(\omega) := \mathcal{F}\{f\}(\omega) = \int_E f(z) e^{-i\omega^T z} \lambda(dz). \tag{6.1}$$

The inverse Fourier transform is given in this case by:

$$f(z) := \mathcal{F}^{-1}\{\hat{f}\}(z) = \int_{\mathbb{R}^d} \hat{f}(\omega) e^{i\omega^T z} \mu(d\omega), \tag{6.2}$$

where $\mu(d\omega)$ represents the measure in the Fourier domain. In this chapter we treat the cases where $E = \mathbb{R}^d$ for which $\mu(d\omega) = \frac{d\omega}{(2\pi)^d}$, and the case where E is a discrete set, for which the measure μ is defined as: $\mu(d\omega) = \mathbb{1}[\omega \in [-\pi, \pi]^d] \frac{d\omega}{(2\pi)^d}$. Throughout the chapter, we reserve the letter i to denote the imaginary unit: $i^2 = -1$. To denote higher order derivatives of the function f, we use the multi-index notation (Saint Raymond, 2018). For a multi-index $n = (n_1, ..., n_d) \in \mathbb{N}^d$, we define:

$$\partial_z^n := \frac{\partial^{|n|}}{\partial z_1^{n_1} ... \partial z_d^{n_d}} \quad \text{where} \quad |n| = \sum_{j=1}^d n_j \quad \text{and} \quad \omega^n := \prod_{j=1}^d \omega_j^{n_j}.$$

To clarify the multi-index notation, let us consider the example where $d = 3$, and $n = (1, 0, 2)$, in this case:

$$\partial_z^n = \frac{\partial^3}{\partial z_1 \partial z_3^2} \quad \text{and,} \quad \omega^n = \omega_1 \omega_3^2.$$

The objective is to derive stochastic backpropagation rules, similar to that of (Rezende, Mohamed, and Wierstra, 2014), for functions of the form: $\mathcal{L}(\theta) := \mathbb{E}_{\mathbf{z}\sim q_\theta}[f(\mathbf{z})]$, for any arbitrary distribution q_θ, discrete or continuous.

6.3 GENERALIZED STOCHASTIC BACKPROPAGATION

Stochastic backpropagation rules similar to that of (Rezende, Mohamed, and Wierstra, 2014) can in fact be derived for any continuous distribution, under certain conditions on the characteristic function. In the following theorem we present the main result of this chapter concerning the derivation of Generalized stochastic backpropagation rules.

Theorem 1. *(Continuous Stochastic Backpropagation) Let $f \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R}_+)$, under the condition that $\nabla_\theta \log \varphi_\theta$ is a holomorphic function of $i\omega$, then there exists a unique set of real numbers $\{a_n(\theta)\}_{n\in\mathbb{N}^d}$ such that:*

$$\nabla_\theta \mathcal{L} = \sum_{|n|\geq 0} a_n(\theta) \mathbb{E}_{\mathbf{z}\sim q_\theta} \left[\partial_z^n f(\mathbf{z})\right]. \tag{6.3}$$

Where $\{a_n(\theta)\}_{n\in\mathbb{N}^d}$ *are the Taylor expansion coefficients of* $\nabla_\theta \log \varphi_\theta(\omega)$*:*

$$\nabla_\theta \log \varphi_\theta(\omega) = \sum_{|n|\geq 0} a_n(\theta)(i\omega)^n. \tag{6.4}$$

Proof. Let us rewrite $\mathcal{L}$ in terms of $\hat{f}$:

$$\begin{aligned}
\mathcal{L}(\theta) &= \int q_\theta(z) f(z) \lambda(dz) \\
&= \int q_\theta(z) \mathcal{F}^{-1}[\hat{f}](z) \lambda(dz) \\
&= \int_{\mathbb{R}^d} \hat{f}(\omega) \int_E q_\theta(z) e^{i\omega^T z} \lambda(dz)\mu(d\omega) \quad \text{Fubini's theorem} \\
&= \int_{\mathbb{R}^d} \hat{f}(\omega)\varphi_\theta(\omega)\mu(d\omega).
\end{aligned} \tag{6.5}$$

By introducing the derivative under the integral sign, and using the reinforce trick (Williams, 1992) applied to φ_θ, where $\nabla_\theta \varphi_\theta(\omega) = \varphi_\theta(\omega)\nabla_\theta \log \varphi_\theta(\omega)$, (6.5) becomes:

$$\nabla_\theta \mathcal{L} = \int_{\mathbb{R}^d} \hat{f}(\omega)\varphi_\theta(\omega)\nabla_\theta \log \varphi_\theta(\omega)\mu(d\omega). \tag{6.6}$$

Under analyticity conditions of the gradient of the log characteristic function, we can expand the gradient term $\nabla_\theta \log \varphi_\theta(\omega)$, in terms of Taylor series around zero as:

$$\nabla_\theta \log \varphi_\theta(\omega) = \sum_{|n|\geq 0} a_n(\theta)(i\omega)^n. \tag{6.7}$$

Putting everything together, and replacing the characteristic function by its expression, the gradient of $\mathcal{L}$ becomes:

$$\nabla_\theta \mathcal{L} = \int_{\mathbb{R}^d} \hat{f}(\omega) \int_E q_\theta(z) e^{i\omega^T z} \sum_{|n|\geq 0} a_n(\theta)(i\omega)^n \mu(d\omega)\lambda(dz). \tag{6.8}$$

By rearranging the sums using Fubini's theorem a second time, we obtain the following expression for the gradient:

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= \mathbb{E}_{\mathbf{z}\sim q_\theta}\left[\mathcal{F}^{-1}\left\{\omega \mapsto \sum_{|n|\geq 0} a_n(\theta)(i\omega)^n \hat{f}(\omega)\right\}(\mathbf{z})\right] \\
&= \sum_{|n|\geq 0} a_n(\theta)\mathbb{E}_{\mathbf{z}\sim q_\theta}\left[\mathcal{F}^{-1}\left\{\omega \mapsto (i\omega)^n \hat{f}(\omega)\right\}(\mathbf{z})\right] \\
&= \sum_{|n|\geq 0} a_n(\theta)\mathbb{E}_{\mathbf{z}\sim q_\theta}\left[\partial_z^n f(\mathbf{z})\right].
\end{aligned} \tag{6.9}$$

□

Identically, we can follow the same procedure for discrete random variables. We suppose that q_θ factorizes over disjoint cliques of the dependency graph, where each dimension $\mathbf{z}_j$ takes values in a discrete space $\mathcal{V}al(z_j)$. In theorem 2 we derive the result concerning the discrete case.

Theorem 2. *(Discrete Stochastic Backpropagation) Let E be a discrete space: $E = \prod_{j=1}^{d} \mathcal{V}al(z_j)$, and $\mathcal{C}$ the set of disjoint cliques of the dependency graph over z, that is,*

$$q_\theta(z) = \prod_{c \in \mathcal{C}} q_\theta(z_c)$$

then,

$$\nabla_\theta \mathcal{L} = \sum_{c \in \mathcal{C}} \sum_{z_c \neq z_c^*} \nabla_\theta q_\theta(z_c) \mathbb{E}_{\boldsymbol{z}_{-c} \sim q_\theta} \left[\mathbf{D} f(\boldsymbol{z}_{-c}, z_c) \right]. \tag{6.10}$$

Where:

- z_c^**: represents the normalizing assignment* $q_\theta(z_c^*) = 1 - \sum_{z_c \neq z_c^*} q_\theta(z_c)$.
- $\mathbf{D} f(\boldsymbol{z}_{-c}, z_c) := f(\boldsymbol{z}_{-c}, z_c) - f(\boldsymbol{z}_{-c}, z_c^*)$. (6.11)

Proof. The characteristic function for the factored distribution is given by:

$$\varphi_\theta(\omega) = \prod_{c \in \mathcal{C}} \varphi_\theta^{(c)}(\omega_c), \quad \text{where,}$$

$$\varphi_\theta^{(c)}(\omega_c) = \sum_{z_c \neq z_c^*} q_\theta(z_c) e^{i\omega_c^T z_c} + \left(1 - \sum_{z_c \neq z_c^*} q_\theta(z_c) \right) e^{i\omega_c^T z_c^*}. \tag{6.12}$$

Thus the gradient of the log characteristic function becomes:

$$\nabla_\theta \log \varphi_\theta^{(c)}(\omega_c) = \sum_{z_c \neq z_c^*} \nabla_\theta q_\theta(z_c) \left[\frac{e^{i\omega_c^T z_c} - e^{i\omega_c^T z_c^*}}{\varphi_\theta^{(c)}(\omega_c)} \right]. \tag{6.13}$$

By plugging this expression to (6.6), we obtain:

$$\begin{aligned} \nabla_\theta \mathcal{L} &= \sum_{c \in \mathcal{C}} \sum_{z_c \neq z_c^*} \nabla_\theta q_\theta(z_c) \int \prod_{c' \neq c} \varphi_\theta^{(c')}(\omega_{c'}) \left[e^{i\omega_c^T z_c} - e^{i\omega_c^T z_c^*} \right] \hat{f}(\omega) \mu(d\omega) \\ &= \sum_{c \in \mathcal{C}} \sum_{z_c \neq z_c^*} \nabla_\theta q_\theta(z_c) \mathbb{E}_{\mathbf{z}_{-c} \sim q_\theta} \left[\mathbf{D} f(\mathbf{z}_{-c}, z_c) \right]. \end{aligned} \tag{6.14}$$

□

The estimator of (6.10) has been derived in the literature through Rao Blackwellization of the score function gradient, and it has been known under different names (Asadi et al., 2017; Cong et al., 2019; Titsias and Lázaro-Gredilla, 2015). Theorem 2 shows that the discrete case can also be seen as backpropagating a derivative of the function f, in this case a discrete derivative given by (6.11).

6.4 APPLICATIONS OF GENERALIZED STOCHASTIC BACKPROPAGATION

Following from the previous section, we derive the stochastic backpropagation estimators for certain commonly used distributions.

The multivariate Gaussian distribution: In this case $q_\theta(z) = \mathcal{N}(z; \mu_\theta, \Sigma_\theta)$. The log characteristic function is given by: $\log \varphi_\theta(\omega) = i\mu_\theta^T \omega + \frac{1}{2}\text{Tr}\left[\Sigma_\theta i^2 \omega\omega^T\right]$. Thus by applying theorem 1, we recover the stochastic backpropagation rule of (Rezende, Mohamed, and Wierstra, 2014):

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\mathbf{z}\sim q_\theta}\left\{\left(\frac{\partial \mu_\theta}{\partial \theta}\right)^T \nabla_z f(\mathbf{z}) + \frac{1}{2}\text{Tr}\left[\left(\frac{\partial \Sigma_\theta}{\partial \theta}\right)\nabla_z^2 f(\mathbf{z})\right]\right\}, \tag{6.15}$$

where, ∇_z and ∇_z^2, represent the gradient and hessian operators.

The multivariate Dirac distribution: $q_\theta(z) = \delta_{a_\theta}(z)$, the log characteristic function of the Dirac distribution is given by: $\log \varphi_\theta(\omega) = i\omega^T a_\theta$. Thus the stochastic backpropagation rule of the Dirac is given by:

$$\nabla_\theta \mathcal{L} = \left(\frac{\partial a_\theta}{\partial \theta}\right)^T \mathbb{E}_{\mathbf{z}\sim\delta_{a_\theta}}\left[\nabla_z f(\mathbf{z})\right] = \left(\frac{\partial a_\theta}{\partial \theta}\right)^T \nabla_z f(a_\theta), \tag{6.16}$$

resulting in the classical backpropagation rule. In other words, the deterministic backpropagation rule is a special case of stochastic backpropagation where the distribution is a Dirac delta distribution. This result provides a link between probabilistic graphical models and classical neural networks.

The multivariate Bernoulli: $q_\theta(z) = \prod_{j=1}^d \mathcal{B}(z_j; \pi_\theta^{(j)})$, where $\pi_\theta^{(j)} = \mathbb{P}[z_j = 1]$. By applying theorem 2, we obtain the local expectation gradient of (Titsias and Lázaro-Gredilla, 2015):

$$\nabla_\theta \mathcal{L} = \sum_{j=1}^d \frac{\partial \pi_\theta^{(j)}}{\partial \theta}\mathbb{E}_{\mathbf{z}_{-j}\sim q_\theta}\left[f(\mathbf{z}_{-j}, 1) - f(\mathbf{z}_{-j}, 0)\right]. \tag{6.17}$$

The multivariate categorical: $q_\theta(z) = \prod_{j=1}^d \text{cat}(z_j; \pi_\theta^{(j)})$, where the dimensions are independent and take values in the set $\{1, ..., K\}$. Similarly to the Bernoulli case, we obtain the following stochastic backpropagation rule:

$$\nabla_\theta \mathcal{L} = \sum_{j=1}^d \sum_{k=1}^{K-1} \frac{\partial \pi_{\theta_k}^{(j)}}{\partial \theta}\mathbb{E}_{\mathbf{z}_{-j}\sim q_\theta}\left[\mathbf{D}f(\mathbf{z}_{-j}, k)\right]. \tag{6.18}$$

The Laplace distribution: $q_\theta(z) = L(z; \mu_\theta, b_\theta)$, in this case the log characteristic function is the following: $\log \varphi_\theta(\omega) = i\mu_\theta\omega - \log(1 + b_\theta^2\omega^2)$, using the Taylor series expansion for the function $x \mapsto \frac{1}{1-x}$, we get the following stochastic backpropagation rule for the Laplace distribution:

$$\nabla_\theta \mathcal{L} = \frac{\partial \mu_\theta}{\partial \theta}\mathbb{E}_{\mathbf{z}}\left[\frac{df}{dz}(\mathbf{z})\right] + \frac{1}{b_\theta^2}\frac{\partial b_\theta^2}{\partial \theta}\sum_{n=1}^\infty b_\theta^{2n}\mathbb{E}_{\mathbf{z}}\left[\frac{d^{2n}f}{dz^{2n}}(\mathbf{z})\right]. \tag{6.19}$$

The gamma distribution: $q_\theta(z) = \Gamma(z; k_\theta, \mu_\theta)$, the log characteristic function of the Gamma distribution is given by: $\log \varphi_\theta(\omega) = -k_\theta \log(1 - i\mu_\theta\omega)$. By expanding it using Taylor series of the logarithm function, we obtain the following stochastic backpropagation rule:

$$\nabla_\theta \mathcal{L} = \sum_{n=1}^{\infty} \left[\frac{1}{n} \frac{\partial k_\theta}{\partial \theta} + \frac{k_\theta}{\mu_\theta} \frac{\partial \mu_\theta}{\partial \theta} \right] \mu_\theta^n \mathbb{E}_{\mathbf{z}\sim q_\theta} \left[\frac{d^n f}{dz^n}(\mathbf{z}) \right]. \tag{6.20}$$

The estimator of (6.20) gives a stochastic backpropagation rule for the gamma distribution and, hence also applies by extension to the special cases of the exponential, Erlang, and chi-squared distributions.

The beta distribution: $q_\theta(z) = \text{Beta}(z; \alpha_\theta, \beta_\theta)$, in this case the characteristic function is the confluent hypergeometric function: $\varphi_\theta(\omega) = {}_1F_1(\alpha_\theta; \alpha_\theta + \beta_\theta; i\omega)$. A series expansion of the gradient of the log of this function is not trivial to derive. However, we can use the parameterization linking the gamma and beta distributions to derive a stochastic backpropagation rule. Indeed, if $\zeta_1 \sim \Gamma(\alpha_\theta, 1)$ and $\zeta_2 \sim \Gamma(\beta_\theta, 1)$, then $\mathbf{z} = g(\zeta_1, \zeta_2) = \frac{\zeta_1}{\zeta_1+\zeta_2} \sim \text{Beta}(\alpha_\theta, \beta_\theta)$. By substituting in the gamma stochastic backpropagation rule, we obtain:

$$\nabla_\theta \mathcal{L} = \sum_{n=1}^{\infty} \frac{1}{n} \left\{ \frac{\partial \alpha_\theta}{\partial \theta} \mathbb{E}_{\zeta_1,\zeta_2} \left[\frac{\partial^n f}{\partial \zeta_1^n} \left(\frac{\zeta_1}{\zeta_1 + \zeta_2} \right) \right] + \frac{\partial \beta_\theta}{\partial \theta} \mathbb{E}_{\zeta_1,\zeta_2} \left[\frac{\partial^n f}{\partial \zeta_2^n} \left(\frac{\zeta_1}{\zeta_1 + \zeta_2} \right) \right] \right\}. \tag{6.21}$$

The Dirichlet distribution: $q_\theta(z) = \text{Dir}(z; K, \alpha_\theta)$, following the same procedure, as for the beta distribution and using the following parameterization: $\mathbf{z}_k = \frac{\zeta_k}{\sum_{j=1}^K \zeta_j}$ with, $\zeta_k \sim \Gamma(\alpha_\theta^{(k)}, 1)$, we obtain:

$$\nabla_\theta \mathcal{L} = \sum_{n=1}^{\infty} \frac{1}{n} \left\{ \sum_{k=1}^{K} \frac{\partial \alpha_\theta^{(k)}}{\partial \theta} \mathbb{E}_{\zeta_j \forall j} \left[\frac{\partial^n f}{\partial \zeta_k^n} \left(\frac{\zeta_1}{\sum_{j=1}^K \zeta_j}, ..., \frac{\zeta_K}{\sum_{j=1}^K \zeta_j} \right) \right] \right\}. \tag{6.22}$$

6.5 APPROXIMATIONS OF GENERALIZED STOCHASTIC BACKPROPAGATION

The Generalized stochastic backpropagation gradient as presented in previous sections presents two major computational bottlenecks for non-trivial distributions. The first is the computation of infinite series, and the second is evaluating higher order derivatives of the function f. Depending on the application, the function f could be chosen in order to bypass the computational bottlenecks. A trivial example, is if the higher order derivatives of the function f vanish at a certain order: $\partial_z^n f = 0$. Another example, is the exponential function $f(z) = \exp(\epsilon^T z)$. From the fact that it obeys the

following partial differential equation $\frac{\partial f}{\partial z_j}(z) = \epsilon_j f(z)$, one can deduce that the stochastic backpropagation rule reduces in this case to:

$$\nabla_\theta \mathcal{L} = \nabla_\theta \log \varphi_\theta \left(-i\epsilon\right) \mathbb{E}_{\mathbf{z}\sim q_\theta}\left[f(\mathbf{z})\right] \tag{6.23}$$

In most real world applications however, the infinite sum will not often reduce to a tractable expression such as that of the exponential. An example of this case is the evidence lower bound of a generative model with Bernoulli observations. In this case, the natural solution is to truncate the sum up to a finite order. The assumption (although it might be wrong), is that the components associated to higher frequencies of the gradient of the log characteristic function, do not contribute as much. In this case the gradient of the log characteristic function of (6.7), is truncated and becomes:

$$\nabla_\theta \log \varphi_\theta(\omega) = \sum_{n\leq N} a_n(\theta)(i\omega)^n + o((i\omega)^N). \tag{6.24}$$

6.6 EXPERIMENTS

In our experimental evaluations, we test the stochastic backpropagation estimators of equations 6.19 and 6.20 for the gamma and Laplace distributions. In the case of the gamma estimator, we use toy examples where we can derive exact stochastic backpropagation rules without truncating the infinite sum. As for the Laplace stochastic backpropagation rule, we test the estimator in the case of Bayesian logistic regression with Laplacian priors and variational posteriors on the weights. We compare our estimators with the pathwise (Jankowiak and Karaletsos, 2019; Jankowiak and Obermeyer, 2018), and score function estimators, in addition to the weak reparameterization estimator in the gamma case (Ruiz, AUEB, and Blei, 2016). We do not use control variates in our setup, the goal is to verify the exactness of the proposed infinite series estimators and how they compare to current state-of-the-art methods in simple settings. In all our experiments, we use the Adam optimizer to update the weights (Kingma and Ba, 2014), with a standard learning rate of 10^{-3}. In all the curves, we report the mean and standard deviation for all the metrics considered over 5 iterations.

6.6.1 Toy problems

In the toy problem setting, we test the gamma stochastic backpropagation rule following the same procedure as (Mohamed et al., 2019). We consider the following cases:
Toy problem 1: $\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{z}\sim q_\theta}\left[||\mathbf{z} - \epsilon||^2\right]$, where $q_\theta(z) = \prod_{j=1}^{d} \Gamma(z_j; k_j, \mu_j)$, $\theta = \{k, \mu\}$, and $\epsilon = .49$. In this case, we only need to compute the first and

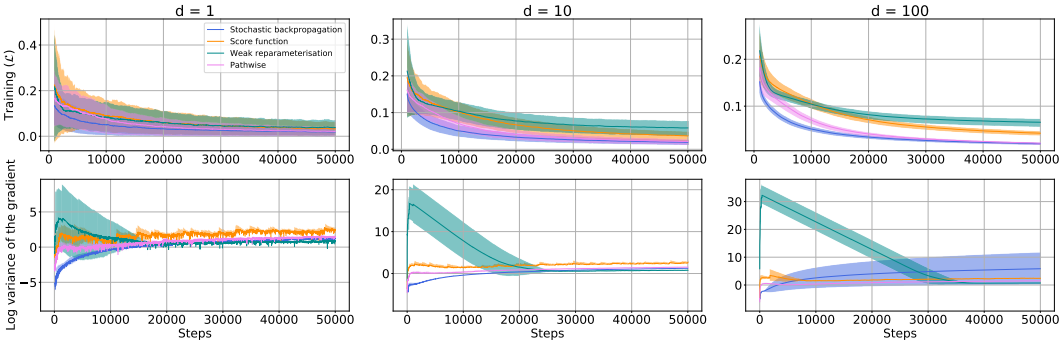

Figure 22: Training loss and log variance of the gradients for the different estimators for $f(z) = \sum_{j=1}^{d}(z_j - \epsilon)^2$ for $d \in \{1, 10, 100\}$.

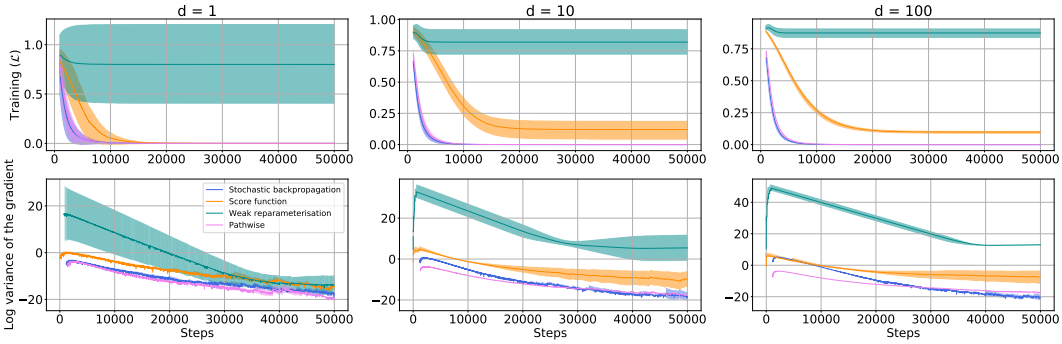

Figure 23: Training loss and log variance of the gradients for the different estimators for $f(z) = \sum_{j=1}^{d} \exp(-\epsilon z_j)$ for $d \in \{1, 10, 100\}$.

second order derivatives of the function f.

Toy problem 2: $\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{z}\sim q_\theta}\left[\sum_{j=1}^{d} \exp(-\epsilon \mathbf{z}_j)\right]$, in this case, the infinite sum transfers to ϵ, which results in the following estimator: $\nabla_\theta \mathcal{L} = \nabla_\theta \log \varphi_\theta(\epsilon i)\, \mathbb{E}_{\mathbf{z}\sim q_\theta}[f(\mathbf{z})]$.

In figures 22 and 23 we report the training loss and log variance of the gradient across iterations of gradient descent for different values of the dimension $d \in \{1, 10, 100\}$. The stochastic backpropagation estimator converges to the minimal value in all cases faster than the other estimators and the variance of the gradient is competitive with the pathwise gradient in the learning regime (number of iterations < 10000).

6.6.2 Bayesian logistic regression with Laplacian Priors

We evaluate the Laplace stochastic backpropagation estimator using a Bayesian logistic regression model (Jaakkola and Jordan, 1997), similarly to (Mohamed et al., 2019). In our case, we substitute the normal prior and posterior on the weights with Laplace priors and posteriors. We adopt the same notations of (Murphy, 2012), where the data, target and weight

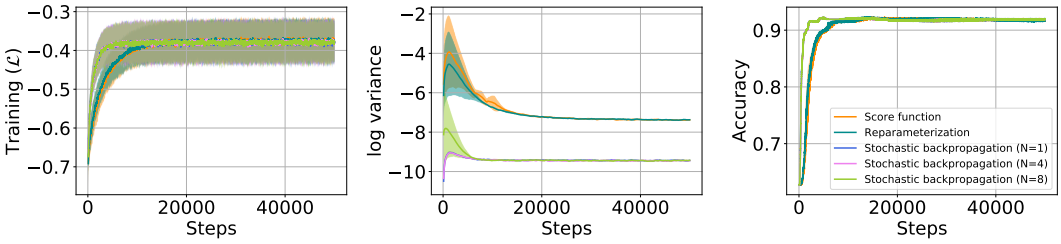

Figure 24: Bayesian Logistic Regression with Laplacian priors

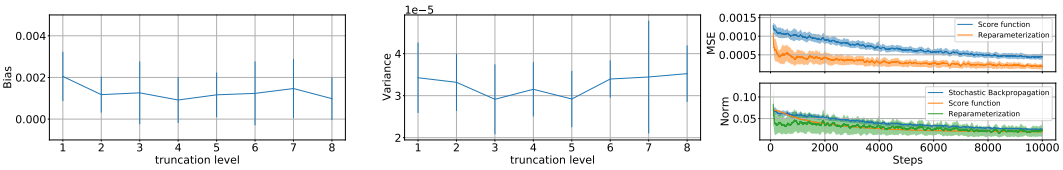

Figure 25: (Left) Bias and variance of the gradient for different values of the truncation level at a fixed parameter value. (Right-top) Mean square error between the Laplace gradient estimator and the score function and reparameterization estimators across iterations. (Right-bottom) Norm of the gradient estimators.

variables are respectively: $x_n \in \mathbb{R}^d$, $y_n \in \{-1, 1\}$, and $\mathbf{w}$. The probabilistic model in our case is the following:

$$p(w) = \prod_{j=1}^{d} L(w_j, 0, 1) \qquad p(y|\mathbf{x}, \mathbf{w}) = \sigma(y\mathbf{x}^T\mathbf{w}), \tag{6.25}$$

where σ represents the sigmoid function. We consider Laplacian variational posteriors of the form $q_\theta(w) = \prod_{j=1}^{d} L(w_j, \mu_j, b_j)$, with $\theta = \{\mu, b\}$. The evidence lower bound of a single sample is given by:

$$\mathcal{L}(x_n, y_n; \theta) = \mathbb{E}_{\mathbf{w}\sim q_\theta}\left[\log \sigma(y_n x_n^T \mathbf{w})\right] - \mathbb{D}_{KL}[q_\theta||p], \tag{6.26}$$

where the Kullback-Leibler divergence between the two Laplace distributions is the following:

$$\mathbb{D}_{KL}[q_\theta||p] = \sum_{j=1}^{d} \left\{ |\mu_j| + b_j e^{-\frac{|\mu_j|}{b_j}} - \log b_j - 1 \right\}. \tag{6.27}$$

We test the model on the UCI women's breast cancer dataset (Dua and Graff, 2017), with a batch size of 64 and 50 samples from the posterior to evaluate the expectation. In the case of the stochastic backpropagation estimator we truncate the infinite series for the scale parameter b of equation 6.19 to $N = 4$ and $N = 8$. In figure 24, we report the training evidence lower bound, the log variance of the gradient, and the accuracy computed on the entire dataset for the different estimators. The stochastic backpropagation estimator converges faster than the considered estimators and the variance

is significantly lower. We also notice that the truncation level of the infinite series for the scale parameter has little effect on the outcome. In figure 25, we report the bias and variance of the estimator at different values of the truncation level, for a fixed parameter value during the training phase (epoch=100). The bias and variance do not vary much, with the truncation level in this case. This result confirms the intuition of neglecting higher frequencies presented in section 6.5. In addition, we compare the mean squared error between the Laplace stochastic backpropagation estimator and the score function and reparameterization estimators. As shown in figure 25 the mean squared error is small, thus the values of the gradients across iterations are close. However, the reparameterization gradient is closer to our estimator than the score function gradient, probably due to the fact that the reparameterization gradient is more stable and has lower variance.

6.7 RELATED WORK & DISCUSSION

Computing gradients through stochastic computation graphs has received considerable attention from the community, due to its application in many fields. The first general approach that provides a closed form solution for any probability distribution is the score function method (Glynn, 1989; Schulman et al., 2015; Sutton et al., 2000; Williams, 1992). The main inconvenience of this approach, is that it results in high variance gradients when the dimension of the random variable becomes high. In order to bypass this issue, the second approach consisted in designing control variates to reduce the variance of the score function estimator (Mnih and Gregor, 2014; Paisley, Blei, and Jordan, 2012; Ranganath, Gerrish, and Blei, 2014; Tokui and Sato, 2017; Weaver and Tao, 2013). In addition to the score function gradient, it was proposed to use an importance weighted estimator instead of the classical score function with a multi-sample objective (Burda, Grosse, and Salakhutdinov, 2015; Mnih and Rezende, 2016).

The second class of approaches is that concerning reparameterization tricks (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014). Through the decoupling of the computation of the gradient from the expectation, reparameterization tricks have shown that they provide low-variance gradients using often a single sample. The issue for these methods is the necessity to find a reparameterization for each probability distribution. Certain distributions such as the Gaussian are easy to reparametrize but others like the gamma are not. In addition, discrete random variables do not admit an easy reparameterization as well. Recently, these issues has been partially solved through implicit reparameterization, the generalized reparameterization gradient, and the pathwise gradient (Figurnov, Mohamed, and Mnih, 2018; Jankowiak and Obermeyer, 2018; Ruiz, AUEB, and Blei, 2016). For the

discrete case, continuous relaxations that are reparameterizable have been proposed and combined with control variate methods (Grathwohl et al., 2018; Gu et al., 2016; Jang, Gu, and Poole, 2016; Maddison, Mnih, and Teh, 2016; Tucker et al., 2017).

Our approach, in contrast provides a new broad family of stochastic backpropagation rules derived using the Fourier transform. One interesting aspect of our approach is the fact that the weighting a_n is separated from the expectation of the higher order derivatives of the function f. Thus the sampled variable does not intervene in the weighting in contrast to other methods such as reparameterization and pathwise gradients. In addition, by applying the derivative to the function f with respect to some variable z_j, terms that do not depend on this variable are eliminated. Thus the other random variables in them do not need to be sampled, which results in lower variance.

It is worth noting that deriving stochastic backpropagation rules using the Fourier transform has been proposed for the Gaussian case (Fellows, Ciosek, and Whiteson, 2018). In our work, we extend it to non Gaussian distributions by way of the characteristic function, and exploiting the invariance of the functional inner product under Fourier transformation (Parseval's theorem).

6.8 CONCLUSION

In this chapter, we presented a new method to compute gradients through random variables for any probability distribution, by explicitly transferring the derivative to the random variable using the Fourier transform. Our approach gives a framework to be applied for any distribution, where the gradient of the log characteristic function is analytic, resulting in a new broad family of stochastic backpropagation rules, that are unique for each distribution.

In the next chapter, we present the Infinite Latent Gaussian Model, an end-to-end model to discover clusters of faults by learning representative deep hidden features. We show how Generalized Stochastic Backpropagation is used to perform variational inference to learn the model parameters.

7

DIRICHLET PROCESS DEEP LATENT MIXTURES FOR FAULT IDENTIFICATION

7.1 INTRODUCTION

In the previous chapter, we have showed how to backpropagate gradients through random variables efficiently. In what follows, we introduce an overall deep generative model capable of learning the hidden features and cluster assignment simultaneously. We show how to use Generalized Stochastic Backpropagation to learn these models, through variational inference. Our model is a generalization of the classical infinite mixture of chapters 3 and 4, with random variables representing hidden features and a Dirichlet process Mixture on the last hidden layer of random variables.

As discussed in previous chapters, nonparametric Bayesian priors, such as the Dirichlet Process (DP), have been widely adopted in the probabilistic graphical community. Their ability to generate an infinite amount of probability distributions using a discrete latent variable makes them ideally suited for automatic model selection. The most famous applications of the DP have been however limited to classical probabilistic graphical models such as Dirichlet Process Mixture Models and Hierarchical Dirichlet Process Hidden Markov Models (Blei, Jordan, et al., 2006; Fox et al., 2008; Zhang, Gultekin, and Paisley, 2016).

Recently, deep generative models such as Deep Latent Gaussian Models (DLGMs) and Variational AutoEncoders (VAEs) (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014) have shown huge success in modeling and generating complex data structures such as images. Various proposals to generalize these models to the mixture and nonparametric mixture cases have been made (Dilokthanakul et al., 2016; Jiang et al., 2016; Nalisnick, Hertel, and Smyth, 2016; Nalisnick and Smyth, 2016). Introducing such priors on top of the deep generative model can improve its generative capabilities, preserve class structure in the latent representation space, and

offer a nonparametric way of performing model selection with respect to the size of the generative model.

The main challenge posed by such models lies in the inference process. Deep generative models with continuous latent variables owe their success mainly to the reparameterization trick (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014). This approach provides an efficient and scalable method for obtaining low variance estimates of the gradient of the variational lower bound with respect to variational posterior parameters. Applying this approach directly to the variational posterior of the DP is not straight-forward, due to the fact that a reparameterization trick for the beta distributions is hard to obtain (Ruiz, AUEB, and Blei, 2016). One approach to bypass this issue have been proposed by (Nalisnick and Smyth, 2016), where the authors used the Kumaraswamy distribution (Kumaraswamy, 1980) as a higher entropy alternative for the beta distribution in the variational posterior. However, by deriving the nature of the variational posterior directly from the variational lower bound, we can show that the appropriate distribution is in fact the beta distribution.

In this chapter we provide an alternative treatment of the variational posterior of Dirichlet Process Deep Latent Mixtures (DP-DLM), where we combine classical variational inference to derive the variational posteriors of the beta distributions and cluster hidden variables, and neural variational inference for the hidden variables of the latent Gaussian model. This leads to gradient ascent updates over the parameters present in nonlinear transformations where Generalized Stochastic Backpropagation can be applied knowing the cluster assignment. As for the remaining parameters, closed-form solutions can be obtained by maximization of the evidence lower bound. The remainder of the chapter is organized as follows:

- In section 7.2, we introduce our proposition of Dirichlet process deep latent mixtures, with nonlinear stochastic hidden layers.
- In section 7.3, we show how to perform variational inference on the model, using Generalized Stochastic Backpropagation of chapter 6.
- In section 7.5, we demonstrate our model on the MNIST (LeCun and Cortes, 2010) open source dataset, and more importantly, in the context of fault identification of our noisy dataset of chapter 5.

7.2 DIRICHLET PROCESS DEEP LATENT MIXTURES

Generalizing deep latent models to the Dirichlet process mixture case can be obtained by adding a Dirichlet process prior on the hidden cluster assignments. We denote these cluster assignments by $\mathbf{z}$. Following the assignment of a cluster hidden variable, a deep latent model is defined for

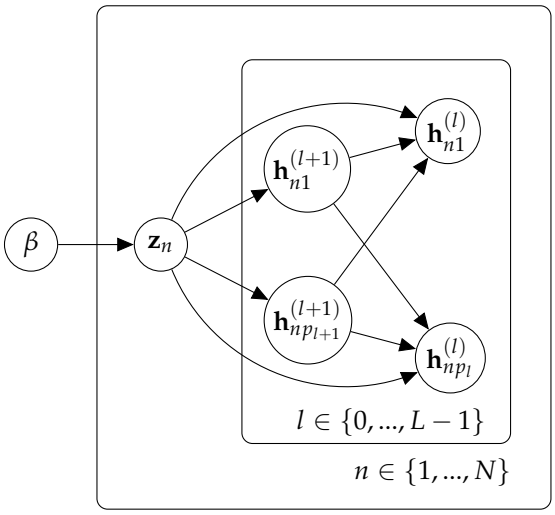

Figure 26: The graphical representation of the generative process of the model, with the convention $\mathbf{x} = \mathbf{h}^{(0)}$.

the assigned cluster similar to (Rezende, Mohamed, and Wierstra, 2014). We adopt the stick-breaking construction of the Dirichlet Process (Sethuraman, 1994). The generative process of the model (Figure 26) is given by:

$$\begin{aligned}
\beta_k &\sim \text{Beta}(\cdot; 1, \eta) \\
\pi_k &= \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) \\
\mathbf{z}_n | \pi &\sim \text{Cat}(\cdot | \pi) \\
\mathbf{h}_n^{(L)} | \mathbf{z}_n = k &\sim p(\cdot | \theta, \mathbf{z}_n = k) \\
\mathbf{h}_n^{(l)} | \mathbf{h}_n^{(l+1)}, \mathbf{z}_n = k &\sim p(\cdot | g_k^{(l)}(\mathbf{h}_n^{(l+1)})) \quad \forall 1 \leq l \leq L-1 \\
\mathbf{x}_n | \mathbf{h}_n^{(1)}, \mathbf{z}_n = k &\sim p(\cdot | g_k^{(0)}(\mathbf{h}_n^{(1)}))
\end{aligned}$$

where $\mathbf{h}_n^{(l)} \in \mathbb{R}^{p_l}$ is the l^{th} layer hidden representation constructed by sampling a base distribution $p(\cdot | g_k^{(l)}(\mathbf{h}_n^{(l+1)}))$ with parameters computed using a nonlinear transformation $g_k^{(l)}(\mathbf{h}_n^{(l+1)}) = \sigma(\mathbf{W}_k^{(l)} \mathbf{h}_n^{(l+1)} + \mathbf{b}_k^{(l)})$. $\mathbf{W}_k^{(l)}$ and $\mathbf{b}_k^{(l)}$ are the weight and bias parameters for the l^{th} hidden layer and k^{th} cluster assignment. The base distribution in this formulation is general, but it can be specified to any classical distribution, for example a Gaussian, with parameters $\{\boldsymbol{m}_k^{(l)}, \boldsymbol{C}_k^{(l)}\} = g_k^{(l)}(\mathbf{h}_n^{(l+1)})$, that is $p(\mathbf{h}_n^{(l)} | g_k^{(l)}(\mathbf{h}_n^{(l+1)})) = \mathcal{N}(\mathbf{h}_n^{(l)}; \boldsymbol{m}_k^{(l)}, \boldsymbol{C}_k^{(l)})$. The last hidden layer is generated using a prior $p(\cdot | \theta, \mathbf{z}_n = k)$ with parameters θ. In the Gaussian example, we have $p(\cdot | \theta, \mathbf{z}_n = k) = \mathcal{N}(\mathbf{h}_n^{(L)}; \boldsymbol{m}_k^{(L)}, \boldsymbol{C}_k^{(L)})$, with $\theta_k = \{\boldsymbol{m}_k^{(L)}, \boldsymbol{C}_k^{(L)}\}$.

We denote by η the concentration parameter of the Dirichlet process, which is a hyperparameter to be tuned manually. The emission distribution of the observable $\mathbf{x}_n$ is usually chosen to be a normal distribution for continuous

variables or the Bernoulli distribution for binary variables. We denote the parameters of the generative model by:

$$\Theta = \{\theta_{1:\infty}, W_{1:\infty}^{(0:L-1)}, b_{1:\infty}^{(1:L-1)}\}$$

The model thus has an infinite number of parameters due to the Dirichlet process prior. Furthermore, the posterior distribution of the hidden variables cannot be computed in closed-form. In order to perform inference on the model we need to use approximate methods such as Markov Chain Monte Carlo (MCMC) or Variational Inference. As discussed in previous chapters, MCMC methods are not suitable for high dimensional models, convergence of the Markov chain to the true posterior can prove to be slow and hard to diagnose (Blei, Kucukelbir, and McAuliffe, 2017).

In the next section, we show how to use variational inference and Generalized Stochastic Backpropagation to learn the model. We show that by choosing a suitable structure for the variational posterior, closed-form solutions can be obtained for the updates of the truncated variational posteriors of the beta distributions, and the variational posteriors of the cluster hidden variables. As for the remaining parameters, gradient ascent coupled with Generalized Stochastic Backpropagation are used to learn the variational posteriors of the hidden layers and model parameters.

7.3 STRUCTURED VARIATIONAL INFERENCE

For a brief review of variational methods, we denote by $\mathbf{x}_{1:N}$ the N samples present in the dataset supposed to be independent and identically distributed. The log-likelihood of the model is intractable due to the required marginalization of all the hidden variables. In order to bypass this marginalization, we introduce an approximate distribution q_Φ and use Jensen's inequality to obtain a lower bound (Jordan et al., 1999):

$$\begin{aligned} l(\Theta) &= \ln p_\Theta(\mathbf{x}_{1:N}) \\ &= \ln \left[\sum_{\mathbf{z}_{1:N}} \int p_\Theta(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta) d\mathbf{h}_{1:N}^{(1:L)} d\beta \right] \\ &\geq \mathbb{E}_{\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta \sim q_\Phi} \left[\ln \frac{p_\Theta(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta)}{q_\Phi(\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta | \mathbf{x}_{1:N})} \right] \\ &= \mathcal{L}(\Theta, \Phi). \end{aligned} \tag{7.1}$$

We have shown that if the distribution q_Φ is a good approximation of the true posterior, maximizing the evidence lower bound (ELBO) with respect to the model parameters Θ is equivalent to maximizing the log-likelihood (cf, chapter 2). For deep generative models, most state-of-the-art methods

use inference networks to construct the posterior distribution (Nalisnick and Smyth, 2016; Rezende, Mohamed, and Wierstra, 2014). For deep mixture models with discrete latent variables, this approach leads to a mixture density variational posterior where the reparameterization trick requires additional investigation (Graves, 2016). Our approach combines standard variational Bayes and neural variational inference. We approximate the true posterior using the following structured variational posterior:

$$q_\Phi(\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta|\mathbf{x}_{1:N}) = \prod_{n=1}^{N}\prod_{l=1}^{L} q_{\psi_{\mathbf{z}_n}^{(l)}}(\mathbf{h}_n^{(l)}|\mathbf{x}_n, \mathbf{z}_n) \times q_{\phi_n}(\mathbf{z}_n|\mathbf{x}_n)\prod_{t=1}^{T} q_{\gamma_t}(\beta_t|\mathbf{x}_{1:N}), \tag{7.2}$$

where T is a truncation level for the variational posterior of the beta distributions obtained by supposing that $q(\beta_T = 1) = 1$ (Blei, Jordan, et al., 2006). We assume a factorized posterior over the hidden layers $\mathbf{h}_n^{(1:L)}$, where the intra-layer dependencies are conserved.

7.3.1 Deriving the variational posteriors q_{ϕ_n} and q_{γ_t}

Deriving the nature of the posterior distributions of the hidden layers $\mathbf{h}_n^{(1:L)}$ using the variational approach is intractable due to the nonlinearities present in the model. Thus, we take a similar approach to (Rezende, Mohamed, and Wierstra, 2014), and we assume that the variational posterior is specified by an inference network, where the parameters of the distribution are the outputs of auxiliary nonlinear transformations of parameters $\psi_t^{(l)}$. For example, in the Gaussian case:

$$q_{\psi_t^{(l)}}(\mathbf{h}_n^{(l)}|\mathbf{x}_n, \mathbf{z}_n = t) = \mathcal{N}\left(\mathbf{h}_n^{(l)}; \mu_{\psi_t^{(l)}}(\mathbf{x}_n), \Sigma_{\psi_t^{(l)}}(\mathbf{x}_n)\right), \tag{7.3}$$

where $\mu_{\psi_t^{(l)}}(\mathbf{x}_n), \Sigma_{\psi_t^{(l)}}(\mathbf{x}_n)$ are outputs of a one layer neural network of parameters $\psi_t^{(l)}$ (Further details provided in Appendix A.2).

In contrast to the hidden layers, we can use the proposed variational posterior of equation (7.2) to derive closed-form solutions for q_{ϕ_n} and q_{γ_t}. Let us consider the Kullback–Leibler definition of the ELBO $\mathcal{L}$:

$$\mathcal{L}(\Theta, \Phi) = -\mathbb{D}_{KL}\left[q_\Phi(\cdot|\mathbf{x}_{1:N})||p_\Theta(\cdot, \mathbf{x}_{1:N})\right].$$

By plugging the variational posterior and isolating β_t terms and $\mathbf{z}_n$ terms, we can analytically derive the optimal distributions q_{γ_t} and q_{ϕ_n} maximizing $\mathcal{L}$ (see Appendix B):

$$q_{\gamma_t}^*(\beta_t|\mathbf{x}_{1:N}) \;=\; \text{Beta}(\beta_t; \gamma_{1,t}, \gamma_{2,t})$$

$$q^*_{\phi_n}(\mathbf{z}_n|\mathbf{x}_n) \;=\; \text{Cat}(\mathbf{z}_n;\phi_n),$$

where the fixed point equations for the variational parameters ϕ_n and γ_t are:

$$\begin{aligned}
\gamma_{1,t} = 1 \;+\; \sum_{n=1}^{N}\phi_{n,t} \quad \gamma_{2,t} = \eta + \sum_{n=1}^{N}\sum_{r=t+1}^{T}\phi_{n,r} \qquad (7.4)\\
\log\phi_{n,t} \;=\; \text{const} + \mathbb{E}_{\beta\sim q}[\log\pi_t]\\
+\; \mathbb{E}_{\mathbf{h}_n^{(1:L)}\sim q_{\psi_t^{(1:L)}}}\Big[\log p(\mathbf{x}_n,\mathbf{h}_n^{(1:L)}|\mathbf{z}_n=t)\Big]\\
+\; \sum_l \mathbb{H}\Big[q_{\psi_t^{(l)}}(\cdot|\mathbf{z}_n=t,\mathbf{x}_n)\Big] \quad \text{s.t.} \quad \sum_{t=1}^{T}\phi_{n,t}=1
\end{aligned}$$

The fixed point equation of $\phi_{n,t}$, requires the evaluation of the expectation over the hidden layers, this can be performed by sampling from the variational posterior of each hidden layer and then forwarding the sample using the generative model:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{h}_n^{(1:L)}\sim q_{\psi_t^{(1:L)}}}\Big[\log p(\mathbf{x}_n,\mathbf{h}_n^{(1:L)}|\mathbf{z}_n=t)\Big]\\
&\qquad\approx \frac{1}{S}\sum_{s=1}^{S}\log p\left(\mathbf{x}_n,\mathbf{h}_{n,t}^{(1:L)(s)}|\mathbf{z}_n=t\right) \qquad (7.5)\\
&\qquad\text{where:}\quad \mathbf{h}_{n,t}^{(l)(s)}\sim q_{\psi_t^{(l)}}(\cdot|\mathbf{x}_n,\mathbf{z}_n=t).
\end{aligned}$$

A key insight here is the following: if a cluster t is incapable of reconstructing a sample $\mathbf{x}_n$ from the variational posterior, this will reinforce the belief that $\mathbf{x}_n$ should not be assigned to that cluster.

7.3.2 Generalized Stochastic Backpropagation

We next show how to perform stochastic backpropagation in order to maximize $\mathcal{L}$ with respect to the parameters $\Theta = \{W_{1:T}^{(0:L-1)}, b_{1:T}^{(0:L-1)}, \theta_{1:T}\}$ and $\Psi = \psi_{1:T}^{(1:L)}$. Similarly to the previous section, we isolate the terms in the evidence lower bound dependent on Ψ and Θ. We have:

$$\begin{aligned}
\mathcal{L}(\Psi,\Theta) = \text{const} + \sum_{n,t}\phi_{n,t}\Big\{\sum_l \mathbb{H}\Big[q_{\psi_t^{(l)}}(\mathbf{h}_n^{(l)}|\mathbf{z}_n=t,\mathbf{x}_n)\Big]\\
+\,\mathbb{E}_{\mathbf{h}_n^{(1:L)}\sim q_{\psi_t^{(1:L)}}}\Big[\log p_\Theta(\mathbf{x}_n,\mathbf{h}_n^{(1:L)}|\mathbf{z}_n=t)\Big]\Big\}. \qquad (7.6)
\end{aligned}$$

By taking the expectation over the hidden cluster variables $\mathbf{z}_n$, we obtain conditional expectations over the hidden layers $\mathbf{h}_n^{(1:L)}$ knowing the cluster

Algorithm 6 Variational Inference for the DP-DLM

Input: $\mathbf{x}_{1:N}, T, \eta, \alpha$
Initialize ϕ, Θ, Ψ
while not converged **do**
  update: $\gamma_t \quad \forall t$ (7.4)
  for each epoch **do**
    $\Theta \leftarrow \Theta + \alpha \nabla_\Theta \mathcal{L}$
    $\Psi \leftarrow \Psi + \alpha \nabla_\Psi \mathcal{L}$
  end for
  update: $\phi_{n,t} \quad \forall n, \forall t$ (7.5)
end while

assignment. In order to apply Generalized Stochastic Backpropagation of chapter 6, let $f(\mathbf{h}_n^{(1:L)}) = \log p(\mathbf{x}_n, \mathbf{h}_n^{(1:L)} | \mathbf{z}_n = t)$. Using Theorem 1, we have:

$$\nabla_\Psi \mathcal{L} = \sum_{|m| \geq 0} a_m(\Psi) \mathbb{E}_{\mathbf{h}_n^{(1:L)} \sim q_\Psi} \left[\partial^m_{\mathbf{h}_n^{(1:L)}} f(\mathbf{h}_n^{(1:L)}) \right] \tag{7.7}$$

In the Gaussian case, for example, that is:

$$q_{\psi_t^{(l)}}(\mathbf{h}_n^{(l)} | \mathbf{x}_n, \mathbf{z}_n = t) = \mathcal{N}\left(\mathbf{h}_n^{(l)}; \mu_{\psi_t^{(l)}}(\mathbf{x}_n), \Sigma_{\psi_t^{(l)}}(\mathbf{x}_n) \right),$$

we have:

$$\nabla_{\psi_t^{(l)}} \mathcal{L} = \mathbb{E}_{\mathbf{h}_n^{(1:L)} \sim q_\Psi} \left\{ \left(\frac{\partial \mu_{\psi_t^{(l)}}(\mathbf{x}_n)}{\partial \psi_t^{(l)}} \right)^T \nabla_{\mathbf{h}_n^{(l)}} f(\mathbf{h}_n^{(1:L)}) + \frac{1}{2} \mathrm{Tr} \left[\left(\frac{\partial \Sigma_{\psi_t^{(l)}}(\mathbf{x}_n)}{\partial \psi_t^{(l)}} \right) \nabla^2_{\mathbf{h}_n^{(l)}} f(\mathbf{h}_n^{(1:L)}) \right] \right\}. \tag{7.8}$$

The gradient and hessian of the function f can be computed using standard automatic differentiation packages such as PyTorch (Baydin et al., 2018; Paszke et al., 2019). The expectation can be estimated using a standard Monte Carlo estimator, the same as equation (7.5). The entropy term of equation (7.6) can be computed analytically, thus gradients can be obtained using standard backpropagation (2.18). As for the Θ parameters, the expectation is independent of these parameters thus:

$$\nabla_\Theta \mathcal{L} = \sum_{n,t} \phi_{n,t} \left\{ \mathbb{E}_{\mathbf{h}_n^{(1:L)} \sim q_{\psi_t^{(1:L)}}} \left[\nabla_\Theta \log p_\Theta(\mathbf{x}_n, \mathbf{h}_n^{(1:L)} | \mathbf{z}_n = t) \right] \right\}, \tag{7.9}$$

which could be also estimated using standard backpropagation and a Monte Carlo estimator. Given the estimated values of these gradients, we can use standard gradient ascent to fit the parameters of the model in parallel with the cluster assignments. Algorithm 6 summarizes the process.

7.4 SEMI-SUPERVISED LEARNING (SSL)

7.4.1 SSL using the DP-DLM

In this section, similarly to chapter 4 we consider a partially labeled dataset $\mathbf{x}_{1:N} = D_l \cup D_u$, where $D_l = \{\mathbf{x}_n, \mathbf{y}_n\}_n$ is the labeled part, $\mathbf{y}_n$ represents the label of the sample $\mathbf{x}_n$, and D_u represents the unlabeled part. The log likelihood can be divided for the labeled and unlabeled parts as:

$$\begin{aligned} l(\Theta) &= \log p_\Theta(\mathbf{x}_{1:N}) \\ &= \sum_{\mathbf{x}_n \in D_l} \log p_\Theta(\mathbf{x_n}) + \sum_{\mathbf{x}_n \in D_u} \log p_\Theta(\mathbf{x}_n) \\ &= \sum_{\mathbf{x}_n \in D_l} \log p_\Theta(\mathbf{x}_n, \mathbf{z}_n = \mathbf{y}_n) + \sum_{\mathbf{x}_n \in D_u} \log p_\Theta(\mathbf{x}_n). \end{aligned}$$

The last equation follows from the fact that $p_\Theta(\mathbf{x}_n|\mathbf{z}_n \neq \mathbf{y}_n) = 0$. By dividing the labeled and unlabeled parts of the dataset, we can follow the same approach presented in section 7.3 in order to derive a variational inference algorithm. In this case, the fixed point updates and the gradient ascent steps remain unchanged if we set $\phi_{n,\mathbf{y}_n} = 1$ for a labeled $\mathbf{x}_n$ sample.

7.4.2 The predictive distribution

In order to make predictions using the model, we need to evaluate the predictive distribution. Given a new sample $\mathbf{x}_{N+1}$, the objective is to evaluate the following quantity $p(\mathbf{z}_{N+1} = k|\mathbf{x}_{1:N+1})$. In the same way as in chapter 4, we can use the variational posterior to approximate the true posterior, which in turn leads to simpler expectation terms:

$$\begin{aligned} p(\mathbf{z}_{N+1} = k|\mathbf{x}_{1:N+1}) &\propto p(\mathbf{z}_{N+1} = k, \mathbf{x}_{N+1}|\mathbf{x}_{1:N}) \\ &\propto\!\!\propto \mathbb{E}_{\beta\sim q}\left[\pi_k(\beta)\right] \\ &\times \mathbb{E}_{\mathbf{h}_{N+1}^{(1:L)} \sim q_{\psi_k^{(1:L)}}} \left[p\left(\mathbf{x}_{N+1}|g_\Theta(\mathbf{h}_{N+1}^{(1:L)}), \mathbf{z}_n = k\right)\right] \end{aligned} \tag{7.10}$$

where $g_\Theta(\cdot)$ represents the forward pass over the generative model. The expectation with respect to the beta terms can be computed in closed-form as a product of expectations over the beta posteriors. The second expectation can be evaluated using a Monte-Carlo estimator same as equation (7.5).

kNN (k=5)	DGM	SB-DGM	**Ours**
$6.13 \pm .13$	$4.86 \pm .14$	$3.95 \pm .15$	$\mathbf{2.90} \pm \mathbf{.17}$

Table 8: Semi-supervised classification error (%) on the MNIST test set with 10 % labelisation. Comparison with (Nalisnick and Smyth, 2016)

.

7.5 EXPERIMENTS ON OPEN-SOURCE DATA

In the previous section, we proposed a novel deep generative model. In order to demonstrate our model, we start by evaluating it on public open-source data. Given the general form of our model and its deep structure, we chose to evaluate it on open-source image data, following the standards of the generative modeling community.

7.5.1 Evaluation of the semi-supervised classification

We evaluate the semi-supervised classification capabilities of the model. We train a two hidden layer DP-DLM model ($L = 2$) on the MNIST dataset (LeCun and Cortes, 2010) with train-valid-test splits equal to $\{45000, 5000, 10000\}$ similarly to (Nalisnick and Smyth, 2016), with 10 % labelisation randomly drawn. We train our algorithm until converenge using 5 cross-validation runs, and we evaluate our model on the test set. We report the mean and standard deviation of the classification error in percentages in Table 8. Our method produces a competitive score with existing state-of-the art methods: Deep Generative Models (DGM) (Kingma et al., 2014) and Stick-Breaking Deep Generative Models (SB-DGM) (Nalisnick and Smyth, 2016). Unlike the previous approaches, the loss was not up-weighted for the labeled samples. Figure 27 (left) shows the t-SNE projections (Maaten and Hinton, 2008a) obtained with 10 % of the labels provided. We notice that by introducing a small fraction of labels the class structure was highly preserved in the latent space.

7.5.2 Data generation and visualization

To further test our model, we generate samples for each cluster from the models trained on both the MNIST and SVHN (Netzer et al., 2011) datasets. The MNIST model is trained in an unsupervised manner, and the SVHN model is trained with semi-supervision where we provide 1000 randomly generated labels. The samples obtained are represented in figure 28. For the unsupervised model, we notice that the clusters are representative of

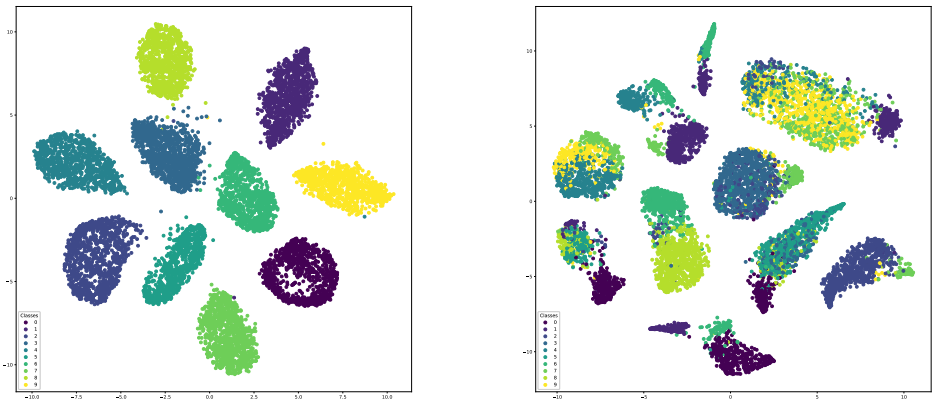

Figure 27: (Left) t-SNE plot of the second stochastic hidden layer on the MNIST test set for the semi-supervised (10% labels) version of the DP-DLM. (Right) t-SNE plot of the second stochastic hidden layer on the MNIST test set for the unsupervised version of the DP-DLM.

Figure 28: Generated samples from the DP-DLM model for the unsupervised version on the MNIST dataset (left) and the semi-supervised version on the SVHN dataset (right).

the shape of each digit. We plot the t-SNE (Maaten and Hinton, 2008b) projections of the MNIST test set of the unsupervised model in Figure 27 (right). We notice that the digits belonging to the same true class tend to group with each other. However, two groups of the same class can be very separated in the embedding space. The interpretation we can draw from this effect is that the DP-DLM tends to separate the latent space in order to distinguish between the variations of hidden representations of the same class. The clusters obtained are not always representative of the true classes which is a common effect with infinite mixture models. In a full unsupervised setting, data can be explained by multiple correct clustering results. This effect can simply be countered by adding a small supervision (figure 27 left).

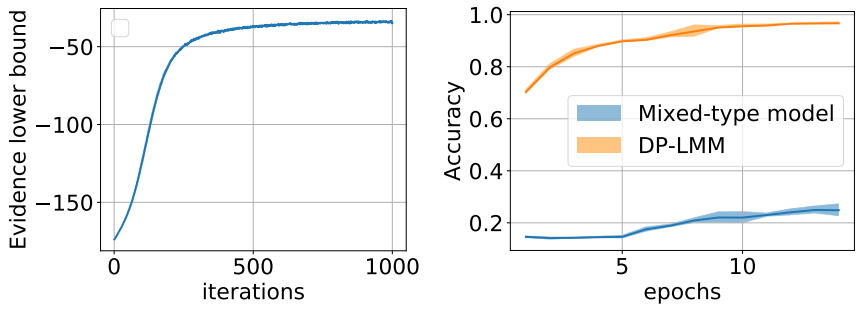

Figure 29: (Left) Evidence lower bound across learning iterations. (Right) Comparison of the accuracy between the Semi-supervised mixed type mixture and the Dirichlet process deep latent mixture.

7.6 EXPERIMENTS ON GPON-FTTH DATA

In this section of experiments, we fit our Dirichlet Process Deep Latent Mixture model in order to model clusters of faults in the noisy GPON-FTTH dataset of chapter 5. Our goal is to demonstrate the representational power in the hidden feature space, as well as, the classification of known faults and exploration of new ones.

7.6.1 Experimental setup

In this setting, we consider all known faults of the noisy dataset as presented in Table 5, in addition to the unknown faults. In training, we up-sample low represented classes to get a balanced dataset for each class. We use our Dirichlet process deep latent mixture to model the known classes and to cluster simultaneously the unknown fault data.

We consider four hidden layers in our probabilistic graphical model ($L = 4$), with Gaussian priors and variational posteriors. The visible variables $\mathbf{x}$ follow a Gaussian distribution if the dimension is continuous and a Bernoulli distribution with one-hot encoding if the dimension is categorical.

We implement Algorithm 6, with the following hyperparameters. The learning rate α is chosen to be equal to .001, decreasing exponentially across iterations. The parameters are learned using stochastic gradient descent with batch size equal to 256, with 100 epochs (passes over the dataset). The truncation level is set to $T = 50$ as an upper bound on the number of clusters. And the concentration parameter is set to $\eta = 1/T$ (uniformative prior).

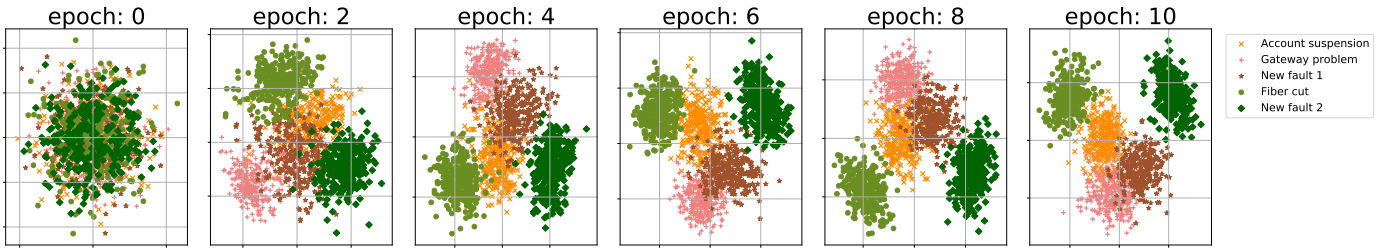

Figure 30: 2D representation of the last hidden layer for each class, and clusters discovered.

7.6.2 Model convergence

Following the same procedure as for previous chapters, in order to assess model convergence, we report the evidence lower bound of equation (7.1). In Figure 29 (left), we can see that the evidence lower bound increases monotonically across iterations reaching a plateau. This indicates that the model is well-fitted and that the optimal variational parameters Ψ and Θ have been identified.

7.6.3 Semi-supervised classification results

In order to demonstrate the benefits of using deep representations, we compare the Dirichlet process deep latent mixture model and the infinite semi-supervised mixed-type mixture model of chapter 4. We evaluate the classification accuracy on the known labels of the noisy dataset for both models. We use a held out dataset unseen during training, containing a .1 fraction of the whole dataset. We run the experiment for five cross-validation runs, and we report the mean and standard deviation for the accuracy.

In Figure 29 (right), we report the classification accuracy across training epochs (passes over the dataset), for the eight known labels. As it can be noticed the DP-DLM outperforms considerably the mixed-type mixture. Furthermore, the shallow semi-supervised mixture model has really poor performance on this dataset. This result is to be expected seeing how the dataset contains many dimensions of noise (not relevant for faults). The DP-DLM has the capacity to compress and filter these dimensions into a more informative representation in the hidden layers. This capability is not available with classical shallow mixtures.

7.6.4 Cluster identification & visualisation

A second useful characteristic of our generative model is its capacity to give us a two dimensional representation of the hidden features. This provides us with a way of visualizing discovered clusters and learned classes. This is done by fixing the dimension of the last hidden layer to two ($p_L = 2$).

In order to demonstrate the usefulness of this 2D representation, we report the evolution of clusters across learning iterations in Figure 30. For visualisation purposes we use three known classes, and two discovered clusters. As it can be seen the clusters are well separated in the feature space, which suggests that the model has learned useful features to separate these clusters.

7.7 CONCLUSION

In this chapter, we introduced a novel model, the Dirichlet Process Deep Latent Mixture Model, for semi-supervised cluster discovery. We have shown how to use variational inference coupled with Generalized Stochastic Backpropagation to learn the model. We demonstrated the model on public open-source data, and GPON-FTTH data for the fault discovery task. We have shown that our model is competitive of state-of-the art methods in terms of semi-supervised classification. We have shown as well, that the model can be used to learn clusters of faults from network data.

The remaining issue after the cluster discovery process is the cluster interpretation problem. Interpretability of machine learning algorithms is still an open research field, with many questions. In the next and last chapter of the thesis, we will discuss the interpretability of the models introduced. We will show which models are interpretable by construction, and for the non-interpretable models, we will give a standard approach to interpret the clusters of faults obtained, so that network experts can fully exploit them.

8

FAULT CLUSTER INTERPRETATION FOR DOMAIN EXPERTS

In the previous chapters, we have presented multiple models for fault diagnosis in GPON-FTTH data. We have tested their accuracy on known faults, and presented their clustering results on unknown and unlabeled instances corresponding to new faults. The clustering results present clear patterns in the data either using the shallow models or the deep models.

However, a problem remains, in order to exploit and make use of these clusters, we need to present them in a comprehensible way to network experts. In other words, either we need to specify directly a pattern of reoccurring variables for each fault, or retrace the fault using deterministic decisions of the sort "if-else" on the network variables.

Interpretability in machine learning is still considered as an open question, with many sub-areas of interesting research avenues. The multitude of models that perform predictions in a black-box way has moved this question to the center of the field, especially if we want to ensure reliability of our algorithms. The majority of the research focuses on making decisions or predictions of a model interpretable (Fong and Vedaldi, 2017; Lundberg and Lee, 2017; Ribeiro, Singh, and Guestrin, 2016). Others, focus on specific models and mainly deep neural networks given their multiple applications (Agarwal et al., 2020; Fong, Patrick, and Vedaldi, 2019; Selvaraju et al., 2017; Yosinski et al., 2015).

Although these methods are widely popular in the machine learning community, their application for the network diagnosis task could prove limited. This is due to the fact, that in the machine learning community we would like to investigate **how** the model reached a decision, but in the network community we would like to determine **why** the model reached a decision. Investigating gradients and visualising hidden layers is useful for the machine learners but not for the network experts.

Therefore, in this chapter, we will go through each of the models presented and their results, and we will detail which are naturally interpretable, and

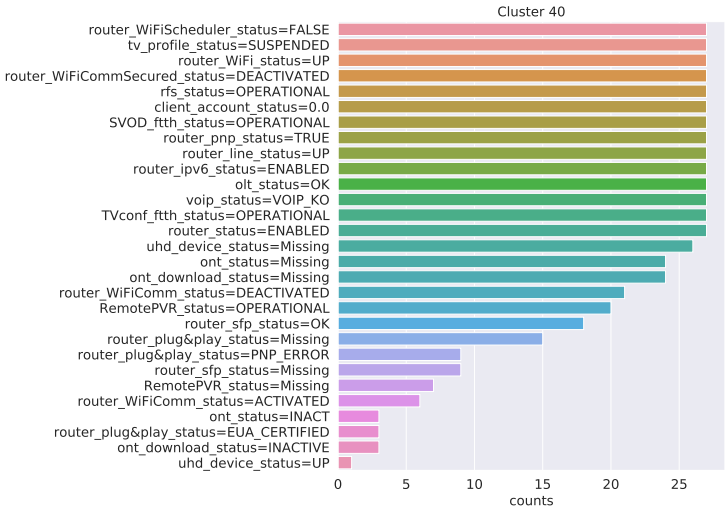

Figure 31: Interpretation of fault cluster 40, corresponding to an account suspension of the customer on the network.

which are not. For the deep models that are not easily interpretable, we will give a method based on decision trees to investigate the clustering results of these models.

8.1 INTERPRETABILITY OF SHALLOW MIXTURE MODELS

The infinite mixture models introduced in chapters 3 and 4 directly model clusters from the data available. Thus in order to interpret the cluster, it suffices to inspect the emission distributions of the observable variables for each cluster on the fitted model. In what follows, we detail how to perform this task for each model.

8.1.1 The Infinite Categorical Mixture Model

Recall that the emission distribution of the Infinite Categorical Mixture model is defined as:

$$\mathbf{x}_i | z = k, \mathbf{B} \sim \text{Cat}(\cdot | \mathbf{B}_{k,i}) = \prod_{v \in E_i} \mathbf{B}_{k,i,v}^{\mathbb{1}[\mathbf{x}_i = v]} \tag{8.1}$$

Meaning, for a certain value $v \in E_i$, $\mathbf{B}_{k,i,v}$ represents the probability of dimension i taking value v, supposing the cluster assignment is k:

$$\mathbf{B}_{k,i,v} = p[\mathbf{x}_i = v | z = k] \tag{8.2}$$

Thus, in order to interpret cluster $z = k$, one only needs to use the posterior of $\mathbf{B}$ (in this case, the variational posterior q^*) to sample from the

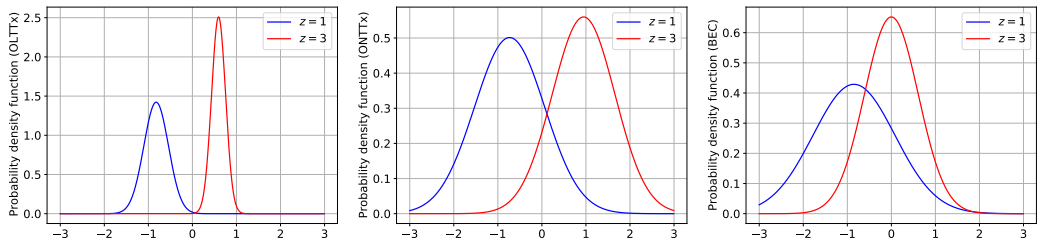

Figure 32: Posterior distributions of the visible variables given the cluster assignment.

generative model by fixing $z = k$. Then, by counting the values occurring for each dimension, we get which occurrences contribute the most.

As an example on the dataset considered for experiments in chapter 3, we report the analysis for a clearly identified cluster in Figure 31. The cluster represents customers for which the account is suspended, this is clear by noticing that the values taken by the variables client account status equal to 0, and tv profile status being suspended. For more examples of explained clusters, we refer the reader back to chapter 3, subsection 3.6.3.

8.1.2 Continuous Dimensions and the Mixed-type Model

The Mixed-type Mixture model adds continuous dimensions to the categorical ones. The categorical ones thus can be treated in the same manner as the previous subsection, for the continuous dimensions however, we recall that:

$$\mathbf{x}_{\mathcal{C}} | z = k, \mathbf{M}, \mathbf{\Lambda} \sim \mathcal{N}(\cdot; \mathbf{M}_k, \mathbf{\Lambda}_k^{-1}) \tag{8.3}$$

where $\mathbf{x}_{\mathcal{C}}$ is the observable vector of continuous variables, following a Gaussian distribution of mean and covariance matrix $\mathbf{M}_k$, and $\mathbf{\Lambda}_k^{-1}$ respectively, under the assumption that the cluster assignment $z = k$. In this case, if in the model assumptions we suppose that the dimensions are uncorrelated, that is, $\mathbf{\Lambda}_k^{-1} = \sigma_k^2 \mathbf{I}$, we can plot the mixture posterior distributions directly for each cluster and dimension. Thus, we have an interpretation of the mean values taken by each fault cluster.

As demonstrated in Figure 32, we plot the posterior distributions for three variables: Optical line termination transmission powers (OLTTx), Optical network termination transmission powers (ONTTx), and The down bit error count (BEC). All values are normalized between 0 and 1, -1 represents the case where an equipment is not responding thus providing no value. We plot the distributions for two recognizable clusters $z = 1$, and $z = 3$. As it can be seen, the cluster $z = 1$ represents the case where there is a problem with the OLT, where values are centered around -1. As for cluster $z = 3$, it represents normal behavior with (BEC equal to 0, and ONT equal to 1), however the OLT transmission power is slightly low (approximately 0.6).

```
|--- delc.res.ftth.client.ols.mesure.ontrx <= 0.76
|   |--- olt.24.res.ont.data.vendor_1397572179 <= 0.50
|   |   |--- livebox.14.res.LANHosts <= 0.04
|   |   |   |--- olt.45.res.olt.temp <= 0.51
|   |   |   |   |--- delc.res.ftth.client.ols.mesure.onttx <= 0.77
|   |   |   |   |   |--- delc.res.ftth.histoalarms.sample.duration <= 0.70
|   |   |   |   |   |   |--- olt.24.res.ont.ols.tx <= 0.52
|   |   |   |   |   |   |   |--- class: 4
```

Figure 33: An example of a decision rule leading to cluster 4.

```
|--- delc.res.ftth.client.ont.lb.link_NOK >  0.50
|   |--- class: 3
```

Figure 34: Clear decision leading to cluster 3.

8.2 INTERPRETABILITY OF DEEP MODELS

Deep models presented in chapters 5 and 7 are much more challenging to interpret. The introduction of non-linear hidden layers, either stochastic or deterministic, creates new features that help performance but heart interpretability. This is due to the fact that there is no way to reverse the process back to the original data. However, these models do provide us with cluster representing new patterns as demonstrated in previous chapters. Furthermore, the data is structured and can be read by experts of the domain. The challenge is to be able to link the clusters identified to patterns in the data.

In order to accomplish this task, we propose to train a decision tree classifier (Breiman et al., 1984) on the unknown fault data with the cluster labels as the true labels of the classifier. Thus, the resulting decision tree classifier gives us a path of logical statements on the features leading to each cluster discovered. This provides a readable interpretable rule that can be exploited by experts of the network domain to identify the fault.

As an example, we revisit the clusters identified using the deep models in previous chapters, and we apply our method to them. In Figure 33, we give an example of such a decision path for the forth cluster discovered in the unknown fault dataset of chapter5. As it can be seen the decision rule mainly involves transmission and reception powers of the optical network termination of the GPON-FTTH network. This suggests a problem with Optical Network Termination (ONT). Another example where the decision is quite clear is given in Figure 34, where the issue is the link between the livebox (lb) or home router of the client and the Optical Network Termination (ONT).

9

CONCLUSION

9.1 THESIS WORK SUMMARY

In this thesis, we have presented a class of probabilistic graphical models, based on Dirichlet process mixtures and deep generative models, in order to cluster faults from network data. Our models are capable of identifying clusters of faults without knowing a priori the number of clusters using the Dirichlet process. In addition, we presented how to extend the Dirichlet process mixtures to include latent hidden representations. These representations can learn features relevant to the fault identification task.

Furthermore, we introduced a novel method called *Generalized Stochastic Backpropagation* to compute low-variance gradients of the evidence lower bound with respect to the variational distribution. Our approach was used to train the proposed end-to-end *Dirichlet Process Deep Latent Mixture Model* for fault classification and discovery. Our approach also generalizes previous estimators and proposes new ones to other distributions.

We validated the presented models in the context of the fault diagnosis and discovery task. However, the models can be used in larger sense to other tasks in various domains. The theoretical results presented can be applied to other fields of machine learning ranging from reinforcement learning (Sutton and Barto, 2018) to attention mechanisms (Mnih, Heess, Graves, et al., 2014; Vaswani et al., 2017). The experimental results show that our models are able to classify with high accuracy known faults as well as clustering new patterns of faults.

The interpretation of new faults is straight-forward for shallow models. In this case, we can compute statistics relevant to each fault class, and investigate the variables occurring for each fault using these models. We can thus deduce the root cause of a fault cluster. As for deep models, where the interpretation is not straight-forward, we can use the clusters found as labels, to train highly interpretable models, such as decision trees, in

order to retrace the path of decision leading to the cluster. This leads to an interpretation even for the deep models.

9.2 PERSPECTIVES

In this thesis we have discussed Dirichlet process mixtures and Dirichlet process deep mixtures for the fault classification and clustering problem. The extensions of these models and their application to different domains provide us with many interesting avenues for future research perspectives.

The first item treated is the clustering of faults using Dirichlet process mixtures. As discussed in chapters 3 and 4, we can see two main benefits with these models. The first is the learning of the number of clusters simultaneously with the cluster assignments. The second is their ability to treat continuous and categorical variables at the same time. However, the Dirichlet process is not the only non-parametric prior that can perform clustering and identify clusters. Other priors such as the mixture of finite mixtures prior (Miller and Harrison, 2015) can identify the number of clusters as well. Furthermore, the assumption behind the Dirichlet Process is that the size of clusters decreases as the number of clusters grows. This assumption follows intuition when dealing with clusters of faults, yet further experimental evaluations need to be performed to validate it. In the case where new faults keep occurring with homogeneous sizes of clusters, other priors might be better suited for the infinite models.

Furthermore, we have assumed in the shallow infinite mixtures that the visible variables are independent. This assumption is clearly violated when dealing with network variables. These variables are often correlated and can be described by a Bayesian network with dependencies. A natural extension of the models would be to include these dependencies in the definition of the mixture models, and to re-derive the mean-field equations to include them. This could improve performance of the models in terms of accuracy and generalization. This conjecture could be verified in future work.

In part ii of the manuscript, we have discussed the extension of the infinite mixture models to include neural networks as feature extractors. Although their efficacy has been proven in chapters 5 and 7, we opted for simple fully connected architectures with a fixed number of hidden layers. The effect of these hyperparameters can be significant in terms of the effectiveness of the feature extractor. Furthermore, given that network data can easily be reconstructed into data originating from graphs, we can explore new architectures that are specifically invented to handle such data, such as, Graph Neural Networks (Gori, Monfardini, and Scarselli, 2005; Kipf and Welling, 2016; Scarselli et al., 2008). Such models can prove effective in

learning features specific to graph data, that are not accessible to classic fully connected layers, where the relationship between the elements of the layers is not taken into account.

Moreover we have introduced a new method of computing low-variance gradients through random variables in chapter 6. These computations arise in multiple domains ranging from deep learning, probabilistic models to reinforcement learning. However, we have not extensively tested our approach in different domains as is usually done in the machine learning community. This aspect still needs some investigation and could be another avenue of future research. This work may include building a generic library for generalized stochastic backpropagation that could be used in multiple applications. Also, for nontrivial distributions the approach requires computations of infinite sums of higher derivatives, which is often of great complexity. Thus further experimental and theoretical work has to be performed to deal with these aspects.

Finally, in this work we have focused mainly on the isolation step of the diagnosis problem, i.e. identifying a fault cluster and its cause. However, the detection step consisting of identifying anomalies can also benefit from the work done in this thesis. Indeed, the models introduced can be adapted to be applied to anomaly detection problems, for example, we can construct a probabilistic model that assigns normal data a high probability, and thus anomolous data is naturally assigned a low probability. In addition, the mitigation step can be addressed using approaches such as Markov Decision Processes and Reinforcement learning (Sutton and Barto, 2018), where the theoretical methods introduced in this thesis could be of use. And last, all that is discussed can be applied to other areas of network management applications, such as Intrusion Detection Systems (IDS), repair agents, and more. All these are interesting areas of future research, that could widen the application of probabilistic graphical models to solve network management problems.

Part III

APPENDICES

A

VARIATIONAL INFERENCE

A.1 MEAN FIELD VARIATIONAL INFERENCE

In this appendix, we provide a review of mean field variational inference specifically the problem of solving equation (3.10):

$$q^* = \arg\min_q \mathbb{D}_{KL}\left[q||p(\cdot|\mathbf{x}^{(1:N)})\right]. \tag{A.1}$$

Let $\boldsymbol{\zeta} = \zeta_{1:M}$ represent the hidden random variables to infer, and $\mathbf{x}^{(1:N)}$ represent the data vector of the observable random variable $\mathbf{x}$. We have:

$$\begin{aligned}
\mathbb{D}_{KL}\left[q||p(\cdot|\mathbf{x}^{(1:N)})\right] &= \sum_{\boldsymbol{\zeta}} q(\boldsymbol{\zeta}) \log \frac{q(\boldsymbol{\zeta})}{p(\boldsymbol{\zeta}|\mathbf{x}^{(1:N)})} && \text{(A.2)}\\
&= \sum_{\boldsymbol{\zeta}} q(\boldsymbol{\zeta}) \log \frac{q(\boldsymbol{\zeta})}{p(\boldsymbol{\zeta}, \mathbf{x}^{(1:N)})} + \log p(\mathbf{x}^{(1:N)}) && \text{(A.3)}\\
&= \mathbb{D}_{KL}\left[q||p(\cdot, \mathbf{x}^{(1:N)})\right] + \underbrace{\log p(\mathbf{x}^{(1:N)})}_{\text{independent of } q} && \text{(A.4)}
\end{aligned}$$

Thus the minimization criterion of equation A.1 is equivalent to the following:

$$q^* = \arg\min_q \mathbb{D}_{KL}\left[q||p(\cdot, \mathbf{x}^{(1:N)})\right]. \tag{A.5}$$

Notice that the last minimizing criterion uses the joint distribution that is known, in contrast to the posterior distribution that is intractable. Let us denote by $\mathcal{L}$ the function to be minimized, we have

$$\begin{aligned}
\mathcal{L}(q) &= \mathbb{D}_{KL}\left[q(\boldsymbol{\zeta})||p(\boldsymbol{\zeta}, \mathbf{x}^{(1:N)})\right]\\
&= \sum_{\boldsymbol{\zeta}} q(\boldsymbol{\zeta}) \log \frac{q(\boldsymbol{\zeta})}{p(\boldsymbol{\zeta}, \mathbf{x}^{(1:N)})}\\
&= \sum_{\zeta_m} \sum_{\boldsymbol{\zeta}_{-m}} \underbrace{q(\zeta_m) q(\boldsymbol{\zeta}_{-m})}_{\text{eq (3.9) Factorisation}} \log \frac{q(\zeta_m) q(\boldsymbol{\zeta}_{-m})}{p(\boldsymbol{\zeta}, \mathbf{x}^{(1:N)})}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\zeta_m} q(\zeta_m)[-\sum_{\zeta_{-m}} q(\zeta_{-m}) \log p(\zeta, \mathbf{x}^{(1:N)}) + \log q(\zeta_m)] \\
&- \underbrace{\sum_{\zeta_m} q(\zeta_m)}_{=1} \mathbb{H}[q(\zeta_{-m})] \\
&= \sum_{\zeta_m} q(\zeta_m) \left[-\mathbb{E}_{\zeta_{-m} \sim q} \left[\log p(\zeta, \mathbf{x}^{(1:N)}) \right] + \log q(\zeta_m) \right] - \mathbb{H}[q(\zeta_{-m})] \\
&= \mathbb{D}_{KL} \left[q(\zeta_m) || f(\zeta_m) \right] - \mathbb{H}[q(\zeta_{-m}] \qquad \text{(A.6)}
\end{aligned}
$$

where :

$$
f(\zeta_m) \propto \exp \left(\mathbb{E}_{\zeta_{-m} \sim q} \left[\log p(\zeta, \mathbf{x}^{(1:N)}) \right] \right) \qquad \text{(A.7)}
$$

Given the decomposition of $\mathcal{L}(q)$ to a term depending on $q(\zeta_m)$ and a term depending on $q(\zeta_{-m})$, we can minimize with respect to each $q(\zeta_m)$, $\forall m$:

$$
\begin{aligned}
q^*(\zeta_m) &= \arg\min_{q(\zeta_m)} \mathcal{L}(q(\zeta_m) q^*(\zeta_{-m})) \\
&= \arg\min_{q(\zeta_m)} \left[\mathbb{D}_{KL} \left[q(\zeta_m) || f^*(\zeta_m) \right] - \mathbb{H}[q^*(\zeta_{-m}]] \right] \\
&= \arg\min_{q(\zeta_m)} \mathbb{D}_{KL} \left[q(\zeta_m) || f^*(\zeta_m) \right] \\
&\quad \text{where } f^*(\zeta_m) \propto \exp \left(\mathbb{E}_{\zeta_{-m} \sim q^*} \left[\log p(\zeta, \mathbf{x}^{(1:N)}) \right] \right) \\
&= f^*(\zeta_m)
\end{aligned}
$$

Therefore, the solution to equation (A.1) is in the form of fixed point equations often referred to as the mean field update equations of the form:

$$
\log q^*(\zeta_m) = \text{const} + \mathbb{E}_{\zeta_{-m} \sim q^*} \left[\log p(\zeta, \mathbf{x}^{(1:N)}) \right] \quad \forall m \qquad \text{(A.8)}
$$

A.2 NEURAL VARIATIONAL INFERENCE

In the previous section we discussed mean field variational inference. In this section, we provide a review of neural variational inference for deep generative models, such as the model presented in chapter 7. As a remainder the evidence lower bound in this case is that of equation 7.1. The neural variational posterior is that of the hidden layers $\mathbf{h}_n^{(1:L)}$, for the $l+1$ layer in the Gaussian case we have:

$$
q_{\psi_t^{(l+1)}}(\mathbf{h}_n^{(l+1)} | \mathbf{x}_n, \mathbf{z}_n = t) = \mathcal{N} \left(\mathbf{h}_n^{(l+1)}; \mu_{\psi_t^{(l+1)}}(\mathbf{x}_n), \Sigma_{\psi_t^{(l+1)}}(\mathbf{x}_n) \right). \qquad \text{(A.9)}
$$

Where, $\mu_{\psi_t^{(l+1)}}(\mathbf{x}_n)$, and $\Sigma_{\psi_t^{(l+1)}}(\mathbf{x}_n)$ are outputs of single hidden layer neural networks often called the encoder, that is:

$$
\{\mu_{\psi_t^{(l+1)}}(\mathbf{x}_n), \Sigma_{\psi_t^{(l+1)}}(\mathbf{x}_n)\} = \sigma \left([1, \mathbf{x}_n^T] \psi_t^{(l+1)} \right) \qquad \text{(A.10)}
$$

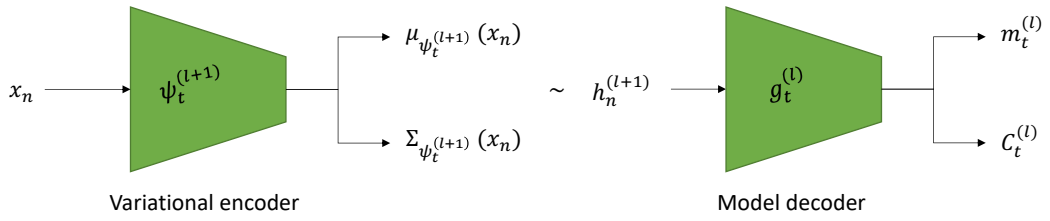

Figure 35: Graphical representation of the different processes in the neural variational model.

In order to optimize for the variational distribution, it suffices to optimize the parameters $\psi_t^{(l+1)}$.

Given the outputs of the decoder defining the variational distribution, we sample $\mathbf{h}_n^{(l+1)}$ from q:

$$\mathbf{h}_n^{(l+1)} \sim q_{\psi_t^{(l+1)}}(\cdot|\mathbf{x}_n, \mathbf{z}_n = t), \tag{A.11}$$

Then we forward it through the probabilistic model to construct the mean and covariance for the next layer as outputs of another neural network called the decoder:

$$\{\boldsymbol{m}_t^{(l)}, \boldsymbol{C}_t^{(l)}\} = \sigma(\mathbf{W}_t^{(l)}\mathbf{h}_n^{(l+1)} + \mathbf{b}_t^{(l)}). \tag{A.12}$$

Then the loss function minimization calibrates the parameters in order to accurately reconstruct the data $\mathbf{x}_n$. A summary of this process is given in Figure 35.

B

FURTHER DETAILS OF CALCULATIONS

B.1 VARIATIONAL INFERENCE FOR THE INFINITE CATEGORICAL MIXTURE MODEL

In order to derive the mean-field equations of the Infinite Categorical Mixture model, we must compute the conditional expectations of equation (3.13), we start by specifying the joint distribution of the model as described in chapter 3:

$$\begin{aligned}\log p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)}) &= \sum_{n=1}^{N} \log p(z^{(n)}, \mathbf{B}, \beta, \mathbf{x}^{(n)}) \\ &= \sum_{n} \Big[\log p(\mathbf{x}^{(n)}|z^{(n)}, \mathbf{B}) + \log p(z^{(n)}|\beta) \\ &\quad + \log p(\mathbf{B}) + \log p(\beta) \Big]\end{aligned}$$

In order to determine the optimal variational distributions and the update equations, we need to compute the conditional expectations of the mean-field equations for each hidden random variable.

B.1.1 Computing $q^*(z_n)$:

$$\begin{aligned}\log q^*(z^{(n)}) &= \text{const} + \mathbb{E}_{\{z^{-n}, \beta, \mathbf{B}\} \sim q^*} \Big[\log p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)}) \Big] \\ &= \text{const} + \mathbb{E}_{\{z^{-n}, \beta, \mathbf{B}\} \sim q^*} \Big[\sum_{m} \Big[\log p(\mathbf{x}^{(m)}|z^{(m)}, \mathbf{B}) \\ &\quad + \log p(z^{(m)}|\beta) + \log p(\mathbf{B}) + \log p(\beta) \Big] \Big]\end{aligned}$$

The main trick of the mean-field equation calculation is to recognize that all expectations and terms not dependent on $z^{(n)}$ are constant values, hence can be integrated in the constant term. Thus, we get:

$$\begin{aligned}\log q^*(z^{(n)}) &= \text{const} + \mathbb{E}_{\mathbf{B}\sim q^*}\left[\log p(\mathbf{x}^{(n)}|z^{(n)}, \mathbf{B})\right] + \mathbb{E}_{\beta\sim q^*}\left[\log p(z^{(n)}|\beta)\right] \\ &= \text{const} + \mathbb{E}_{\mathbf{B}\sim q^*}\left[\sum_i \log \text{Cat}(\mathbf{x}_i^{(n)}|\mathbf{B}_{z^{(n)},i,:})\right] + \mathbb{E}_{\beta\sim q^*}\left[\log \pi_{z^{(n)}}(\beta)\right]\end{aligned}$$

Given that $z^{(n)}$ is a categorical random variable, we deduce that:

$$q^*(z^{(n)}) = \text{Cat}(z_n; \phi_n)$$

with:

$$\begin{aligned}\log \phi_{n,k} &= \text{const} + \mathbb{E}_{\mathbf{B}\sim q^*}\left[\sum_i \log \text{Cat}(\mathbf{x}_i^{(n)}|\mathbf{B}_{k,i,:})\right] + \mathbb{E}_{\beta\sim q^*}\left[\log \pi_k(\beta)\right] \\ &\text{s.t} \quad \sum_{k=1}^{T} \phi_{nk} = 1\end{aligned}$$

B.1.2 Computing $q^*(\mathbf{B}_{k,i})$:

Using the same principle, we have:

$$\begin{aligned}\log q^*(\mathbf{B}_{k,i}) &= \text{const} + \mathbb{E}_{\{z^{(1:N)},\beta,\mathbf{B}_{-k,i}\}\sim q^*}\left[\log p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)})\right] \\ &= \text{const} + \log p(\mathbf{B}_{k,i}) + \sum_n \left[\mathbb{E}_{\{z^{(n)},\mathbf{B}_{-k,i}\}\sim q^*} \log p(\mathbf{x}^{(n)}|z^{(n)}, \mathbf{B})\right] \\ &= \text{const} + \log \text{Dir}(\mathbf{B}_{k,i}; |\mathbf{x}_i|; \boldsymbol{\alpha}_i) + \sum_n \phi_{n,k} \log \text{Cat}(\mathbf{x}_i^{(n)}|\mathbf{B}_{k,i}) \\ &= \text{const} + \sum_{v\in E_i} \left(\alpha_{iv} + \sum_n \phi_{n,k} \mathbb{1}[\mathbf{x}_i^{(n)} = v] - 1\right) \log \mathbf{B}_{k,i} \\ &= \log \text{Dir}(\mathbf{B}_{k,i}; |\mathbf{x}_i|; \boldsymbol{\epsilon}_i)\end{aligned}$$

Where:

$$\boldsymbol{\epsilon}_{i,v} = \alpha_{iv} + \sum_n \phi_{n,k} \mathbb{1}[\mathbf{x}_i^{(n)} = v] \quad \forall v \in E_i$$

B.1.3 computing $q^*(\beta_k)$:

Following the same procedure for the variables β_k, we get:

$$\log q^*(\beta_k) = \text{const} + \mathbb{E}_{\{z^{(1:N)},\beta_{-k},\mathbf{B}\}\sim q^*}\left[\log p(z^{(1:N)}, \mathbf{B}, \beta, \mathbf{x}^{(1:N)})\right]$$

$$
\begin{aligned}
&= \text{const} + \log p(\beta_k) + \sum_n \mathbb{E}_{\{z^{(n)},\beta_{-k}\}\sim q^*}\left[\log p(z^{(n)}|\beta)\right] \\
&= \text{const} + \log \text{Beta}(\beta_k; 1, \eta) + \sum_n \sum_{l=1}^{T} \phi_{n,l} \mathbb{E}_{\beta_{-k}\sim q^*}\left[\log \pi_l(\beta)\right] \\
&= \text{const} + (\eta - 1)\log(1-\beta_k) \\
&+ \sum_n \sum_{l=1}^{T} \phi_{n,l} \mathbb{E}_{\beta_{-k}\sim q^*}\left[\log \beta_l + \sum_{t<l} \log(1-\beta_t)\right] \\
&= \text{const} + (\eta - 1)\log(1-\beta_k) + \sum_n \phi_{n,k} \log \beta_k \\
&+ \sum_n \sum_{l=1}^{T} \sum_{t<l} \phi_{n,l} \mathbb{E}_{\beta_{-k}\sim q^*}[\log(1-\beta_t)] \\
&= \text{const} + (\eta - 1)\log(1-\beta_k) + \sum_n \phi_{n,k} \log \beta_k \\
&+ \sum_n \sum_{t=k+1}^{T} \phi_{n,t} \log(1-\beta_k) \\
&= \log \text{Beta}(\beta_k; \gamma_{1,k}, \gamma_{2,k})
\end{aligned}
$$

With:

$$\gamma_{1,k} = 1 + \sum_{n=1}^{N} \phi_{nk} \quad \gamma_{2,k} = \eta + \sum_{n=1}^{N} \sum_{l=k+1}^{T} \phi_{nl}$$

Finally, given all the variational distributions we need to compute the expectations for the ϕ equation, using standard results of expectations of Dirichlet and beta distributions we have:

$$\mathbb{E}_{\mathbf{B}\sim q^*}\left[\sum_i \log \text{Cat}(\mathbf{x}_i^{(n)}|\mathbf{B}_{z^{(n)},i,:})\right] = \sum_{i=1}^{d} \sum_{v\in E_i} \mathbb{1}[\mathbf{x}_i^{(n)} = v][\psi(\epsilon_{k,i,v}) - \psi(\sum_{v'\in E_i} \epsilon_{k,i,v'})]$$

and,

$$\mathbb{E}_{\beta\sim q^*}\left[\log \pi_k(\beta)\right] = \psi(\gamma_{1,k}) - \psi(\gamma_{1,k} + \gamma_{2,k}) + \sum_{l=1}^{k-1} [\psi(\gamma_{2,l}) - \psi(\gamma_{1,l} + \gamma_{2,l})]$$

Where ψ represents the digamma function.

B.2 VARIATIONAL INFERENCE FOR THE MIXED-TYPE MODEL

The Mixed-Type model is an extension of the Infinite Categorical Mixture Model to incorporate continuous random variables and semi-supervision. The variational inference updates for the random variables $\mathbf{B}, \beta$ remain the

same, new variational updates need to be computed for $\mathbf{M}, \mathbf{\Lambda}$, and the variational equation for the random variables z needs to be updated to incorporate the continuous part of the data.

B.2.1 Computing $q^*(z_n)$:

Using the same procedure as before, but for the new model, we have:

$$\begin{aligned}\log q^*(z^{(n)}) &= \text{const} + \sum_{m=1}^{N} \mathbb{E}_{z^{-n},\mathbf{B},\beta,\mathbf{M},\mathbf{\Lambda}\sim q^*}\left[\log p(z^{(m)}, \mathbf{B}, \beta, \mathbf{M}, \mathbf{\Lambda}, \mathbf{x}^{(m)})\right] \\ &= \text{const} + \mathbb{E}_{\mathbf{B}\sim q^*}\left[\sum_{i\in\mathcal{D}} \log \text{Cat}(\mathbf{x}_i^{(n)} | \mathbf{B}_{z^{(n)},i,:})\right] + \mathbb{E}_{\beta\sim q^*}\left[\log \pi_{z^{(n)}}(\beta)\right] \\ &+ \mathbb{E}_{\mathbf{M},\mathbf{\Lambda}\sim q^*}[\log \mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(n)}; \mathbf{M}_{z^{(n)}}, \mathbf{\Lambda}_{z^{(n)}})]\end{aligned}$$

B.2.2 Computing the variational posterior of the mean and precision matrix:

In order to compute $q^*(\mathbf{M}_k, \mathbf{\Lambda}_k)$, we apply the same principle except in this case to the couple $\{\mathbf{M}_k, \mathbf{\Lambda}_k\}$, we have:

$$\begin{aligned}\log q^*(\mathbf{M}_k, \mathbf{\Lambda}_k) &= \text{const} + \sum_{m} \mathbb{E}_{z^{(1:N)},\mathbf{B},\beta,\mathbf{M}_{-k},\mathbf{\Lambda}_{-k}\sim q^*}\left[\log p(z^{(m)}, \mathbf{B}, \beta, \mathbf{M}, \mathbf{\Lambda}, \mathbf{x}^{(m)})\right] \\ &= \text{const} + \sum_{n,k} \phi_{n,k} \log \mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(n)}; \mathbf{M}_k, \mathbf{\Lambda}_k) \\ &+ \log \mathcal{N}(\mathbf{M}_k, \mu_0, \kappa_0 \mathbf{\Lambda}_k) + \log \mathcal{W}(\mathbf{\Lambda}_k; \mathbf{L}_0; \nu_0)\end{aligned}$$

Using results of conjugate Gaussian and Wishart distributions, we can rewrite the last equation as:

$$\log q^*(\mathbf{M}_k, \mathbf{\Lambda}_k) = \log \mathcal{N}(\mathbf{M}_k; \boldsymbol{\mu}_k, \kappa_k \mathbf{\Lambda}_k) + \log \mathcal{W}(\mathbf{\Lambda}_k; \mathbf{L}_k; \nu_k)$$

By correspondence between the terms, we can deduce, the following equations for the parameters:

$$\boldsymbol{\mu}_k = \frac{\kappa_0 \boldsymbol{\mu}_0 + \sum_{n=1}^{N} \phi_{n,k} \mathbf{x}_{\mathcal{C}}^{(n)}}{\kappa_k} \tag{B.1}$$

$$\kappa_k = \kappa_0 + \sum_{n=1}^{N} \phi_{n,k} \quad \nu_k = \nu_0 + \sum_{n=1}^{N} \phi_{n,k} + 1 \tag{B.2}$$

$$\begin{aligned}\mathbf{L}_k^{-1} &= \mathbf{L}_0^{-1} + \kappa_0 (\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)^T \\ &+ \sum_{n=1}^{N} \phi_{n,k} (\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k)(\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k)^T\end{aligned} \tag{B.3}$$

Finally, in order to give a closed-form for the parameters ϕ, we must use the last distribution to compute the $\{\mathbf{M}_k, \mathbf{\Lambda}_k\}$ expectation:

$$\begin{aligned}\mathbb{E}_{\mathbf{M},\mathbf{\Lambda}\sim q^*}[\log \mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(n)}; \mathbf{M}_k, \mathbf{\Lambda}_k)] = -\frac{1}{2}\left[\frac{|\mathcal{C}|}{\kappa_k} + \nu_k(\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k)^T \mathbf{L}_k(\mathbf{x}_{\mathcal{C}}^{(n)} - \boldsymbol{\mu}_k)\right] \\ + \frac{1}{2}\left[\sum_{i=1}^{|\mathcal{C}|} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + \log\det(\mathbf{L}_k)\right]\end{aligned}$$

B.3 PREDICTIVE DISTRIBUTION OF THE MIXED-TYPE MODEL

As a remainder, the predictive distribution of the mixed-type model is approximately proportional to :

$$\begin{aligned}p(z^{(N+1)} = k|\mathbf{x}^{(1:N+1)}) \propto \mathbb{E}_{\beta\sim q^*}[\boldsymbol{\pi}_k(\beta)] \prod_{j\in\mathcal{D}} \mathbb{E}_{\mathbf{B}_{k,j}\sim q^*}[\text{Cat}(\mathbf{x}_j^{(N+1)}|\mathbf{B}_{k,j})] \\ \times \mathbb{E}_{\mathbf{M}_k,\mathbf{\Lambda}_k\sim q^*}[\mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(N+1)}; \mathbf{M}_k, \mathbf{\Lambda}_k)]\end{aligned}$$

In order to derive this quantity, we need to compute the different expectations, this is possible using results of conjugacy.

B.3.1 Computing the beta expectation:

$$\begin{aligned}\mathbb{E}_{\beta\sim q^*}[\boldsymbol{\pi}_k(\beta)] &= \mathbb{E}_{\beta_t\sim \text{Beta}(\cdot;\gamma_{1,t},\gamma_{2,t})\forall t}[\beta_k \prod_{l<k}(1-\beta_l)] \\ &= \frac{\gamma_{1,k}}{\gamma_{1,k}+\gamma_{2,k}} \prod_{l<k} \frac{\gamma_{2,l}}{\gamma_{1,l}+\gamma_{2,l}}\end{aligned}$$

B.3.2 Computing the $\mathbf{B}$ expectation:

$$\begin{aligned}\mathbb{E}_{\mathbf{B}_{k,j}\sim q^*}[\text{Cat}(\mathbf{x}_i^{(N+1)}|\mathbf{B}_{k,i})] &= \int \text{Dir}(\mathbf{B}_{k,i}; \boldsymbol{\epsilon}_{k,i,}, |\mathbf{x}_i|)\text{Cat}(\mathbf{x}_i^{(N+1)}|\mathbf{B}_{k,i}) d\mathbf{B}_{k,i} \\ &= \frac{\Gamma(\sum_v \boldsymbol{\epsilon}_{k,i,v})}{\prod_v \Gamma(\boldsymbol{\epsilon}_{k,i,v})} \prod_v \int \mathbf{B}_{k,i,v}^{\boldsymbol{\epsilon}_{k,i,v}-1+\mathbb{1}[\mathbf{x}_i^{(N+1)}=v]} d\mathbf{B}_{k,i,v} \\ &= \frac{\Gamma(\sum_v \boldsymbol{\epsilon}_{k,i,v})}{\prod_v \Gamma(\boldsymbol{\epsilon}_{k,i,v})} \frac{\prod_v \Gamma(\boldsymbol{\epsilon}_{k,i,v} + \mathbb{1}[\mathbf{x}_i^{(N+1)} = v])}{\Gamma(\sum_v(\boldsymbol{\epsilon}_{k,i,v} + \mathbb{1}[\mathbf{x}_i^{(N+1)} = v]))} \\ &= \frac{\boldsymbol{\epsilon}_{k,i,\mathbf{x}_j^{(N+1)}}}{\sum_{v\in E_j} \boldsymbol{\epsilon}_{k,i,v}}\end{aligned}$$

B.3.3 Computing the normal expectation:

$$\begin{aligned}\mathbb{E}_{\mathbf{M}_k,\mathbf{\Lambda}_k\sim q^*}[\mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(N+1)};\mathbf{M}_k,\mathbf{\Lambda}_k)] &= \int \mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(N+1)};\mathbf{M}_k,\mathbf{\Lambda}_k) \\ &\quad \mathcal{N}(\mathbf{M}_k;\boldsymbol{\mu}_k,(\kappa_k\mathbf{\Lambda}_k)^{-1})\mathcal{W}(\mathbf{\Lambda}_k;\mathbf{L}_k,\nu_k)d\mathbf{M}_k d\mathbf{\Lambda}_k \\ &= \int I(\mathbf{\Lambda}_k)\mathcal{W}(\mathbf{\Lambda}_k;\mathbf{L}_k,\nu_k)d\mathbf{\Lambda}_k\end{aligned}$$

Where,

$$\begin{aligned}I(\mathbf{\Lambda}_k) &= \int \mathcal{N}(\mathbf{M}_k;\boldsymbol{\mu}_k,(\kappa_k\mathbf{\Lambda}_k)^{-1})\mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(N+1)};\mathbf{M}_k,\mathbf{\Lambda}_k)d\mathbf{M}_k \\ &= \sqrt{\left(\frac{\kappa_k}{2\pi(\kappa_k+1)}\right)^p}\sqrt{\det(\mathbf{\Lambda}_k)}\exp\left\{-\frac{1}{2}\mathrm{tr}\left[\mathbf{\Lambda}_k\mathbf{\Theta}_k^{-1}\right]\right\}\end{aligned}$$

with,

$$\mathbf{\Theta}_k^{-1} = \frac{1}{\kappa_k+1}(\kappa_k\boldsymbol{\mu}_k+\mathbf{x}_{\mathcal{C}}^{(N+1)})(\kappa_k\boldsymbol{\mu}_k+\mathbf{x}_{\mathcal{C}}^{(N+1)})^T+\kappa_k\boldsymbol{\mu}_k\boldsymbol{\mu}_k^T+\mathbf{x}_{\mathcal{C}}^{(N+1)}\mathbf{x}_{\mathcal{C}}^{(N+1)^T}$$

Thus we have:

$$\begin{aligned}&\mathbb{E}_{\mathbf{M}_k,\mathbf{\Lambda}_k\sim q^*}[\mathcal{N}(\mathbf{x}_{\mathcal{C}}^{(N+1)};\mathbf{M}_k,\mathbf{\Lambda}_k)] \\ &\quad= \sqrt{\left(\frac{\kappa_k}{2\pi(\kappa_k+1)}\right)^p}\frac{\int\det(\mathbf{\Lambda}_k)^{\frac{\nu_k+1-p-1}{2}}\exp\left\{-\frac{1}{2}\mathrm{tr}[(\mathbf{L}_k^{-1}+\mathbf{\Theta}_k^{-1})\mathbf{\Lambda}_k]\right\}d\mathbf{\Lambda}_k}{2^{\frac{\nu_k p}{2}}\det(\mathbf{L}_k)^{\frac{\nu_k}{2}}\Gamma_p(\frac{\nu_k}{2})} \\ &\quad= \sqrt{\left(\frac{\kappa_k}{2\pi(\kappa_k+1)}\right)^p}\frac{2^{\frac{(\nu_k+1)p}{2}}\det\left((\mathbf{L}_k^{-1}+\mathbf{\Theta}_k^{-1})^{-1}\right)^{\frac{\nu_k+1}{2}}\Gamma_p(\frac{\nu_k+1}{2})}{2^{\frac{\nu_k p}{2}}\det(\mathbf{L}_k)^{\frac{\nu_k}{2}}\Gamma_p(\frac{\nu_k}{2})} \\ &\quad= \sqrt{\left(\frac{\kappa_k}{\pi(\kappa_k+1)}\right)^p}\frac{\det\left((\mathbf{L}_k^{-1}+\mathbf{\Theta}_k^{-1})^{-1}\right)^{\frac{\nu_k+1}{2}}\Gamma_p(\frac{\nu_k+1}{2})}{\det(\mathbf{L}_k)^{\frac{\nu_k}{2}}\Gamma_p(\frac{\nu_k}{2})}\end{aligned}$$

B.4 VARIATIONAL INFERENCE FOR THE DIRICHLET PROCESS DEEP LATENT MIXTURE

The evidence lower bound of equation 7.1 can be written as :

$$\mathcal{L}(\Theta,\Phi) = -\mathbb{D}_{KL}\left[q_\Phi(\cdot|\mathbf{x}_{1:N})||p_\Theta(\cdot,\mathbf{x}_{1:N})\right],$$

in the following we develop this equation in order to derive the nature of the variational posteriors and their fixed point updates.

B.4.1 Computing $q_{\gamma_t}(\beta_t|\mathbf{x}_{1:N})$:

$$
\begin{aligned}
\mathcal{L} &= \sum_{\mathbf{z}_{1:N}} \int q_\Phi(\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta|\mathbf{x}_{1:N}) \log \frac{p_\Theta(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta)}{q_\Phi(\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta|\mathbf{x}_{1:N})} d\mathbf{h}_{1:N}^{(1:L)} d\beta \\
&= \sum_{\mathbf{z}_{1:N}} \int q_\Phi(\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, |\mathbf{x}_{1:N}) q_\Phi(\beta|\mathbf{x}_{1:N}) \Bigg[\log p(\mathbf{x}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}|\mathbf{z}_{1:N}) \\
&\quad + \log p(\mathbf{z}_{1:N}, \beta) \Bigg] d\mathbf{h}_{1:N}^{(1:L)} d\beta - \mathbb{H}[q_\Phi] \\
&= \underbrace{\text{const}}_{\text{independent of } \beta} + \prod_{t=1}^{T} \int_0^1 q(\beta_t) \Bigg[\sum_n \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log p(\mathbf{z}_n|\beta) \\
&\quad + \sum_{t'} (\log p(\beta_{t'}) - \log q(\beta_{t'})) \Bigg] d\beta_t \\
&= \text{const} + \int_0^1 q(\beta_t) \left[\sum_n \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \log p(\mathbf{z}_n|\beta) + \log p(\beta_t) - \log q(\beta_t) \right] d\beta_t \\
&= \text{const} + \int_0^1 q(\beta_t) \Bigg[\sum_n \sum_k q(\mathbf{z}_n = k) (\log \beta_k \\
&+ \sum_{l<k} \log(1-\beta_l)) + (\eta - 1) \log(1-\beta_t) - \log q(\beta_t) \Bigg] d\beta_t \\
&= \text{const} + \int_0^1 q(\beta_t) \Bigg[(\sum_n q(\mathbf{z}_n = t)) \log \beta_t \\
&+ (\eta + \sum_n \sum_{r=t+1}^{T} q(\mathbf{z}_n = r) - 1) \log(1-\beta_t) - \log q(\beta_t) \Bigg] d\beta_t \\
&= \text{const} - \mathbb{D}_{KL} \left[q(\beta_t) || \text{Beta}(\beta_t; \gamma_{1,t}, \gamma_{2,t}) \right]
\end{aligned}
$$

The distribution $q^*(\beta_t)$ maximizing $\mathcal{L}$, minimizes the kullback-leibler term. Hence:

$$
q^*_{\gamma_t}(\beta_t|\mathbf{x}_{1:N}) = \text{Beta}(\beta_t; \gamma_{1,t}, \gamma_{2,t})
$$

$$
\gamma_{1,t} = 1 + \sum_{n=1}^{N} q(\mathbf{z}_n = t) \quad \gamma_{2,t} = \eta + \sum_{n=1}^{N} \sum_{r=t+1}^{T} q(\mathbf{z}_n = r)
$$

B.4.2 Computing $q_{\phi_n}(\mathbf{z}_n|\mathbf{x}_n)$:

By isolating the terms dependent on $\mathbf{z}_n$ in $\mathcal{L}$, we obtain:

$$
\mathcal{L} = \sum_{\mathbf{z}_{1:N}} \int q_\Phi(\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta|\mathbf{x}_{1:N}) \log \frac{p_\Theta(\mathbf{x}_{1:N}, \mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta)}{q_\Phi(\mathbf{z}_{1:N}, \mathbf{h}_{1:N}^{(1:L)}, \beta|\mathbf{x}_{1:N})} d\mathbf{h}_{1:N}^{(1:L)} d\beta
$$

$$= \text{const} + \sum_{\mathbf{z}_n} q(\mathbf{z}_n) \left\{ \int q(\mathbf{h}_n^{(1:L)}|\mathbf{x}_n, \mathbf{z}_n) \log \left[\frac{p(\mathbf{x}_n|\mathbf{h}_n^{(1:L)}) p(\mathbf{h}_n^{(1:L)}|\mathbf{z}_n)}{q(\mathbf{h}_n^{(1:L)}|\mathbf{x}_n, \mathbf{z}_n)} \right] d\mathbf{h}_n^{(1:L)} \right.$$

$$\left. + \mathbb{E}_{\beta \sim q}[\log p(\mathbf{z}_n|\beta)] - \log q(\mathbf{z}_n) \right\}$$

$$= \text{const} - \mathbb{D}_{KL}\left[q(\mathbf{z}_n)||\text{Cat}(\mathbf{z}_n; \phi_n)\right]$$

Hence, the optimal distribution $q^*_{\phi_n}(\mathbf{z}_n|\mathbf{x}_n)$ maximizing $\mathcal{L}$ satisfies:

$$q^*_{\phi_n}(\mathbf{z}_n|\mathbf{x}_n) = \text{Cat}(\mathbf{z}_n; \phi_n)$$

where,

$$\log \phi_{n,k} = \text{const} + \int q(\mathbf{h}_n^{(1:L)}|\mathbf{x}_n, \mathbf{z}_n = k) \log \left[\frac{p(\mathbf{x}_n, \mathbf{h}_n^{(1:L)}|\mathbf{z}_n = k)}{q(\mathbf{h}_n^{(1:L)}|\mathbf{x}_n, \mathbf{z}_n = k)} \right] d\mathbf{h}_n^{(1:L)} + \mathbb{E}_{\beta \sim q}[\log \pi_k]$$

$$= \text{const} + \mathbb{E}_{\mathbf{h}_n^{(1:L)} \sim q_{\psi_k^{(1:L)}}} \left[\log p(\mathbf{x}_n, \mathbf{h}_n^{(1:L)}|\mathbf{z}_n = k) \right]$$

$$+ \mathbb{E}_{\beta \sim q}[\log \pi_k] + \sum_l \mathbb{H}\left[q_{\psi_k^{(l)}}(\cdot|\mathbf{z}_n = k, \mathbf{x}_n) \right]$$

C

ADDITIONAL THEORETICAL CONTRIBUTIONS

C.1 STOCHASTIC BACKPROPAGATION FOR REINFORCEMENT LEARNING

In the reinforcement learning setting, we would like to optimize the long term expected reward. We follow the notation of (Sutton et al., 2000) and we denote this expected reward $\rho(\pi)$, where the policy π depends on parameters ϕ. The policy gradient theorem states:

$$\nabla_\phi \rho = \sum_s d^\pi(s) \nabla_\phi \mathbb{E}_{a\sim\pi(\cdot|s;\phi)} \left[Q^\pi(s,a) \right] \tag{C.1}$$

where Q^π represents the action-value function, and d^π represents the stationary distribution, or the discounted weighting of probabilities under policy π, depending on the problem formulation. In both cases, we can apply the results of generalized stochastic backpropagation to the expectation inside the sum. For discrete RL, where the action space is $\mathcal{A} = \left\{ a^{(1)}, ..., a^{(K)} \right\}$, we obtain the following expression for the gradient:

$$\nabla_\phi \rho = \sum_s d^\pi(s) \sum_{k=1}^{K-1} \frac{\partial \pi(a^{(k)}|s;\phi)}{\partial \phi} \left[Q^\pi(s, a^{(k)}) - Q^\pi(s, a^{(K)}) \right] \tag{C.2}$$

In the special case of episodic tasks, we would like to optimize the expected sum of rewards $\rho(\pi) = \mathbb{E}_{\tau\sim q_\phi} \left[R(\tau) \right]$. The function R maps a trajectory $\tau = (s_{1:T}, a_{1:T})$ to the sum of rewards obtained from the environment: $R(\tau) = \sum_{t=1}^{T} r_t(s_t, a_t)$. The dynamics are encoded via the probability distribution q_ϕ, where for a set of parameters ϕ, we have:

$$q_\phi(s_{1:T}, a_{1:T}) = q(s_1) \prod_{t=1}^{T} \pi(a_t|s_t;\phi) q(s_{t+1}|s_t, a_t) \tag{C.3}$$

In this case, by exploiting the dependency structure of the distribution q_ϕ, on which the expectation is taken, equation (C.2) transforms into:

$$\begin{aligned}\nabla_\phi \rho &= \sum_{t=1}^{T}\sum_{k=1}^{K-1} \mathbb{E}_{s_{1:t},a_{1:t-1}} \left\{ \frac{\partial \pi(a^{(k)}|s_t;\phi)}{\partial \phi} \left[\mathbb{E}_{s_{t+1:T},a_{t+1:T}} \left[\sum_{t'=t}^{T} r_{t'}(s_{t'},a_{t'}) | a_t = a^{(k)} \right] \right.\right. \\ &- \left.\left. \mathbb{E}_{s_{t+1:T},a_{t+1:T}} \left[\sum_{t'=t}^{T} r_{t'}(s_{t'},a_{t'}) | a_t = a^{(K)} \right] \right] \right\} \qquad \text{(C.4)}\end{aligned}$$

In contrast to the Actor-Critic estimator (Sutton et al., 2000), the stochastic backpropagation estimator only needs the actor neural network of parameters ϕ to output the values of the policy. However, it does require looking ahead to get the expected cumulative reward of each action at time step t, which is an expensive task. In the supplementary materiel we provide a detailed derivation of equation (C.4).

C.2 STOCHASTIC BACKPROPAGATION FOR MIXTURE DENSITY DISTRIBUTIONS

Stochastic backpropagation through mixture density posteriors have been studied recently (Graves, 2016; Roeder, Wu, and Duvenaud, 2017). Following from the last section, we provide an alternative way of deriving stochastic backpropagation rules for mixture density distributions, as long as the base distributions of the mixture admits a stochastic backpropagtion rule. Let us consider a mixture density distribution of the form $q_\phi(z) = \sum_{k=1}^{K} \pi_{\phi_k} q_\phi^{(k)}(z)$, where the component base distributions are $q_\phi^{(k)}$ for all k. In this case, we can rewrite the function $\mathcal{L}$ as :

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{h}\sim \text{cat}(\pi_\phi)} \left\{ \mathbb{E}_{\mathbf{z}\sim q_\phi^{(\mathbf{h})}} \left[f(\mathbf{z}) \right] \right\} \qquad \text{(C.5)}$$

Following from equation (C.5), we can apply the categorical stochastic backpropagation rule on the first expectation, thus obtaining the following expression for the gradient:

$$\nabla_\phi \mathcal{L} = \sum_{k=1}^{K-1} \frac{\partial \pi_{\phi_k}}{\partial \phi} \left[g(k) - g(K) \right] + \mathbb{E}_{\mathbf{h}\sim \text{cat}(\pi_\phi)} \left\{ \nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi^{(\mathbf{h})}} \left[f(\mathbf{z}) \right] \right\} \qquad \text{(C.6)}$$

where $g(k) := \mathbb{E}_{\mathbf{z}\sim q_\phi^{(k)}} \left[f(\mathbf{z}) \right]$. For the second term, we can re-apply stochastic backpropagation to derive a stochastic gradient for each base component and simply take the expectation over the class random variable.

C.3 THE DIRAC DISTRIBUTION: THE LINK BETWEEN NEURAL NETWORKS AND PGMS

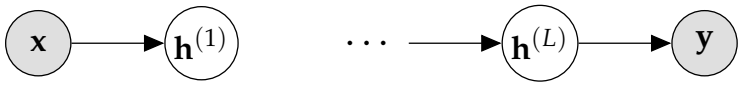

Figure 36: A hidden variable probabilistic model, where the observed variables are the data $\mathbf{x}$ and target $\mathbf{y}$, with L hidden stochastic layers $\mathbf{h}^{(1:L)}$.

In this section, we explore the connection between neural networks and probabilistic graphical models following from the stochastic backpropagation rule of the Dirac delta distribution. To this end, let us consider the probabilistic graphical model of figure 36. The observed random variables in this model are denoted $\mathbf{x}$ and $\mathbf{y}$ representing the data and target variables. We place the analysis in a supervised learning context, but the argument is valid for unsupervised models as well. As usual the goal is to maximize the log likelihood for the data samples (x, y), which is intractable, given that we need to integrate over the hidden variables. However using variational inference, we can maximize an evidence lower bound of the form:

$$\mathcal{L}(\theta; x, y) = \mathbb{E}_{\mathbf{h}^{(1:L)} \sim q_\theta(\cdot|x)} \left[\log p(y, \mathbf{h}^{(1:L)}, x) \right] + \mathbb{H}[q_\theta(\cdot|x)] \tag{C.7}$$

As suggested in the Dirac stochastic backpropagation rule, let us assume that the variational posteriors and priors are Dirac delta distribution of the form:

$$\forall l : q_\theta(\mathbf{h}^{(l+1)}|\mathbf{h}^{(l)}) = p(\mathbf{h}^{(l+1)}|\mathbf{h}^{(l)}) = \delta_{a^{(l+1)}(W^{(l+1)^T}\mathbf{h}^{(l)} + b^{(l+1)})}(\mathbf{h}^{(l+1)}) \tag{C.8}$$

where, the $a^{(l)}$, $W^{(l)}$, and $b^{(l)}$ represent respectively the activation functions, the weights and biases for layer l, with the convention $x := \mathbf{h}^{(0)}$. Under these assumptions, the Kullback-Leibler divergence term is equal to zero, and the evidence lower bound reduces to the the log-likelihood of a classic neural network:

$$\mathcal{L}(\theta; x, y) = \log p(y|g_\theta(x)),$$

with,

$$g_\theta(x) = a^{(L)}(W^{(L)}(....a^{(1)}(W^{(1)}x + b^{(1)})...).$$

Thus, when using neural networks we are indirectly using a probabilistic graphical model and making the strong assumption that the hidden layers follow a parameterized Dirac distribution knowing the previous layer.

D

ADDITIONAL EXPERIMENTAL VALIDATIONS

D.1 EXPERIMENTS USING DISCRETE STOCHASTIC BACKPROPAGATION

We evaluate the Bernoulli and Categorical Stochastic Backpropagation estimators (BSB and CSB) of equations 6.17 and 6.18 on standard generative modeling benchmark tasks, using the MNIST and Omniglot datasets (Lake, Salakhutdinov, and Tenenbaum, 2013; LeCun and Cortes, 2010). We use the REBAR, RELAX, and Gumbel-softmax (or Concrete) estimators as baselines for our comparison (Grathwohl et al., 2018; Jang, Gu, and Poole, 2016; Maddison, Mnih, and Teh, 2016; Tucker et al., 2017). The Bernoulli stochastic backpropagation is compared to the REBAR and RELAX estimators for three models: the sigmoid belief network of one and two stochastic hidden layers (Neal, 1992) and the variational autoencoder. In this case, we adopt the same architectures as (Grathwohl et al., 2018). The categorical stochastic backpropagation estimator is compared to the Gumbel-softmax estimator (Jang, Gu, and Poole, 2016; Maddison, Mnih, and Teh, 2016) using two models: a variational autoencoder and a single layer belief network with categorical priors. In this case, we set the dimension of the hidden layer to $d = 20$ and the number of modalities for each dimension to $K = 10$.

All models are trained using the ADAM optimizer (Kingma and Ba, 2014) using a standard learning rate $\alpha = 10^{-4}$ and batch size of 100. We train the models for 500 epochs on the MNIST dataset and 100 epochs on the Omniglot dataset, longer learning epochs leads to overfitting and lower performance on the test sets for all estimators and models. We perform 5 iterations of training in all experiments and we report the mean and standard deviation of each performance metric considered.

For all models and estimators, we report the mean marginal test likelihood in tables 9 and 10 for both datasets. The test likelihood is estimated via importance sampling using 200 samples from the variational posterior. In

Dataset	Model	REBAR	RELAX	BSB (S=1)	BSB (S=5)	BSB (S=10)
MNIST	one layer SBN	-114.14 ± 0.44	-114.55 ± 0.48	-110.87 ± 0.2	-110.70 ± 0.11	**-110.59 ± 0.08**
	two layer SBN	-101.33 ± 0.04	-101.09 ± 0.07	**-99.74 ± 0.3**	-100.44 ± 0.28	-100.66 ± 0.21
	Bern. VAE	-127.76 ± 0.84	-128.06 ± 2.66	**-107.4 ± 1.47**	-108.46 ± 0.37	-109.19 ± 1.31
Omniglot	one layer SBN	-123.66 ± 0.05	-123.82 ± 0.17	**-113.53 ± 0.21**	-114.34 ± 0.16	-114.37 ± 0.19
	two layer SBN	-117.81 ± 0.17	-117.89 ± 0.04	**-102.05 ± 0.19**	-102.16 ± 0.09	-102.29 ± 0.14
	Bern. VAE	-136.83 ± 0.31	-136.53 ± 0.32	**-126.94 ± 0.81**	-128.69 ± 0.34	-129.48 ± 0.38

Table 9: Test likelihood for the Bernoulli stochastic backpropagation (BSB) estimator compared to the REBAR and RELAX estimators. We report the mean and standard deviation over 5 runs.

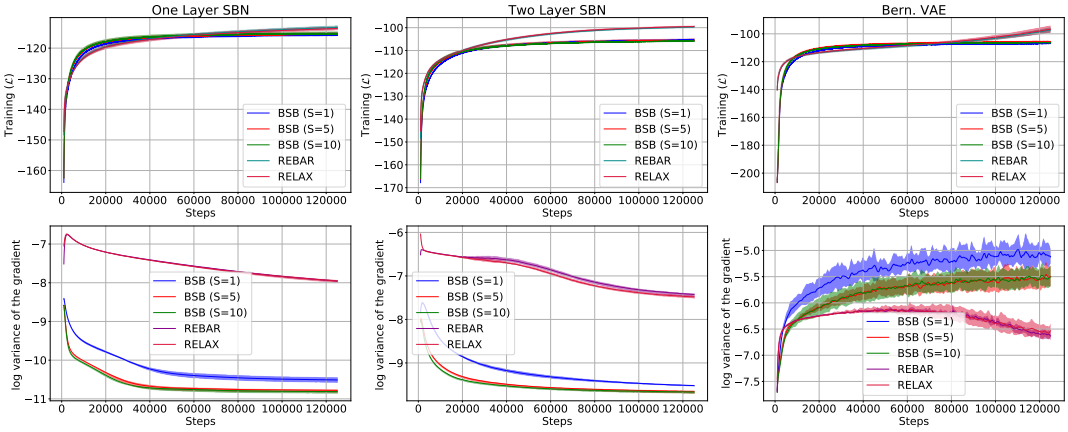

Figure 37: The training evidence lower bound on the MNIST training set (top) and the log variance of the gradient (bottom) over 5 runs. Comparison with the REBAR and RELAX estimators.

all cases the stochastic backpropagation estimator, a control variate free method outperforms the baselines. In the case of the one layer sigmoid belief network the BSB estimator exhibits an increase of performance of about 4 nats in the case of the MNIST dataset and 10 nats in the case of the Omniglot dataset. We also vary the number of samples used to estimate the expectation in the stochastic backpropagation rule $S \in \{1, 5, 10\}$. We notice that using a single sample estimate does not hurt performance and leads to a faster training process.

We estimate the mean variance of the gradients w.r.t the parameters of the models using exponential moving averages of the first and second moments computed by the ADAM optimizer. The BSB estimator significantly outperforms the REBAR, RELAX estimators in terms of variance reduction with a difference of about 2 nats in the case of sigmoid belief networks, and 1 nat in the case of the categorical variational autoencoder on the mnist dataset, leading to a more stable training process as shown in figures 37 and 38.

Dataset	Model	Gumbel-softmax	CSB (S=1)	CSB (S=5)	CSB (S=10)
MNIST	one layer	-113.46 ± 0.59	-107.48 ± 0.37	-107.24 ± 0.31	**-107.33 ± 0.13**
	Cat. VAE	-122.97 ± 5.68	-103.49 ± 0.73	-102.68 ± 0.63	**-101.78 ± 0.88**
Omniglot	one layer	-125.76 ± 0.24	-122.49 ± 0.80	-122.98 ± 0.30	**-122.98 ± 0.21**
	Cat. VAE	-140.25 ± 1.99	**-130.20 ± 0.74**	-131.66 ± 0.84	-131.63 ± 1.05

Table 10: Test likelihood for the categorical stochastic backpropagation (CSB) estimator, compared to the Gumbel-softmax estimator. We report the mean and standard deviation over 5 runs.

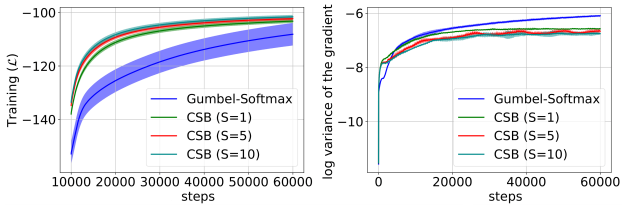

Figure 38: Training evidence lower bound and the log variance of the gradient for the categorical VAE on the MNIST dataset.

Model	GS	CSB (S=1)	CSB (S=5)	CSB (S=10)
one Linear layer	4.11 (s)	6.32 (s)	10.93 (s)	16.83 (s)
Cat. VAE	4.13 (s)	7.32 (s)	13.29 (s)	21.94 (s)

Table 11: Execution time of one epoch of training on the mnist dataset per estimator, per model.

Finally, we evaluate the computational overhead of the categorical stochastic backpropagation estimator compared the Gumbel-softmax estimator. We compare the two estimators in terms of execution time of one epoch of training. The comparison is done using GPU implementations on a NVIDIA GeForce RTX 2080 Ti GPU, where the stochastic backpropagation rule of equation (6.18) is vectorized, thus leveraging the parallel batch treatment of the GPU. As shown in table 11, the Gumbel-softmax method is faster than stochastic backpropagation ($S = 1$) by a difference of about 3 seconds per training epoch. This is due to the forward passes performed to compute each of the terms in equation (6.18). The variance reduction and the increase in performance outweigh however the computational cost.

D.2 REINFORCEMENT LEARNING APPLICATION

In the reinforcement learning setting, we compare the stochastic backpropagation estimator of equation (C.4) to the famous actor-critic baseline (Sutton et al., 2000). We compare the two methods on the classical cart-pole benchmark of openAI gym (Brockman et al., 2016). The Actor network in both cases is a two layer neural network with dimensions equal to [64, 32] and hyperbolic tangent activation functions. The A2C method requires the critic network which serves as the baseline in the A2C gradient estimator. The

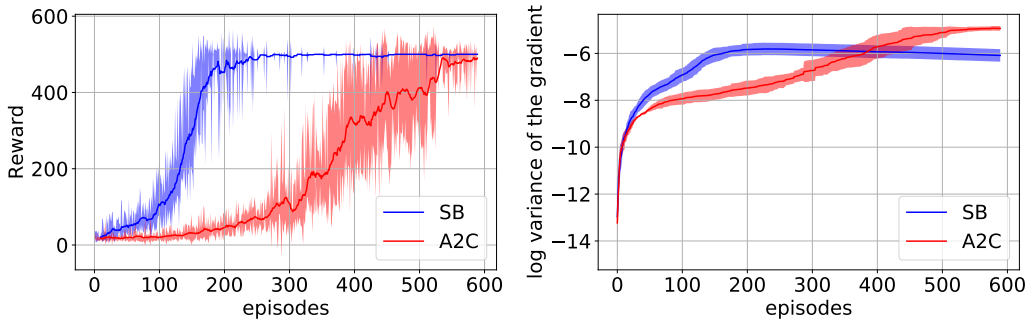

Figure 39: Mean and standard deviation of the cumulative reward (left) and the log variance of the gradient for the actor network (right) over 5 iterations of the training process for the cart-pole task.

critic neural network considered also has two hidden layers of the same dimensionality with ReLU activation functions.

The actor and policy networks are trained using the ADAM optimizer, with a learning rate $\alpha = 10^{-3}$. We compute the cumulative reward at the completion of each episode of the environment, then we update the weights using a single step of gradient descent.

As shown in figure 39, the discrete stochastic backpropagation estimator outperforms the A2C estimator in the sense that it converges faster to the optimal policy. We also estimate the mean variance of the gradient of the policy network. As demonstrated at convergence the stochastic backpropagation estimator has a lower variance of about 1 nat compared to the A2C estimator, leading to more stable training process.

E

RÉSUMÉ EN FRANÇAIS

L'identification des pannes dans les réseaux est d'une importance cruciale pour tout fournisseur de services Internet. Un système de diagnostic précis, capable d'identifier les pannes, peut améliorer l'expérience client, diminuer l'indisponibilité du réseau, et surtout réduire les coûts d'interventions sur le terrain pour le fournisseur d'accès Internet.

Compte tenu des enjeux économiques, de nombreux travaux de recherche ont été menés pour construire et maintenir des systèmes de diagnostic de pannes opérationnels dans tous les domaines, allant des opérations de réseau aux installations et applications industrielles. Cependant, le champ de recherche pour résoudre la tâche de diagnostic est souvent spécifique au domaine, et peut même varier dans sa définition.

Afin de donner une définition formelle de la tâche de diagnostic telle qu'elle est couramment admise dans la communauté de gestion de réseau, nous devons la diviser en trois étapes. La première étape est l'étape de détection, qui peut être effectuée de manière proactive, elle vise à décider si un client rencontre un problème et si des investigations supplémentaires sont nécessaires. La seconde est l'étape d'isolement, où l'objectif est d'identifier la cause racine du problème compte tenu de l'état technique actuel ou des données disponibles. Troisièmement, l'étape de réparation couvre toutes les actions nécessaires pour résoudre le problème.

La première étape, consistant à identifier si une panne s'est produite sur le réseau, est souvent associée à la détection d'un comportement anormal. Les méthodes souvent associées à cette tâche sont des approches de détection d'anomalies. Pour l'application spécifique que nous étudions tout au long de la thèse, à savoir le diagnostic de pannes des services d'accès à large bande, ces méthodes ne sont pas considérées : en effet, une panne est déclarée détectée à la suite d'un appel client.

Ainsi, nous nous intéressons plutôt à l'étape d'isolement, où l'on sait qu'une panne s'est produite sur le réseau, et il faut identifier la cause ou la nature de la panne. Nous nous intéressons plus spécifiquement au diagnostic des pannes des services FTTH (Fiber to the home). Considérant les

différentes applications modernes de l'apprentissage automatique, l'objectif est de construire des modèles capables de classifier les pannes à partir des données du réseau GPON-FTTH issues du fonctionnement des réseaux Orange. De plus, l'enjeu est de construire des systèmes adaptables, dans le sens où, si une nouvelle panne apparaît sur le réseau (due à changements logiciels ou techniques) le modèle est capable de regrouper ensemble les instances de la nouvelle panne, étant donné qu'une signature de la panne existe dans les données rencontrées.

L'APPRENTISSAGE AUTOMATIQUE APPLIQUÉ AU DIAGNOSTIC RÉSEAU

Afin d'introduire la tâche de diagnostic de réseau comme un problème d'apprentissage automatique, nous devons d'abord discuter de l'état actuel du diagnostic pour Orange en particulier, et pour la communauté de recherche en général. Actuellement, dans la plupart des systèmes de diagnostic opérationnels du monde réel, la méthode préférée est les systèmes experts. Spécifiquement pour Orange, le système expert est appelé DELC (Diagnostic Expert de la Ligne Client).

Un système expert est un système basé sur des règles, où, étant donné une certaine condition sur les variables du réseau, une alarme est déclenchée pour alerter si une panne s'est produite et indiquer le cas échéant la cause racine de la panne. Cela a ses avantages et ses inconvénients. Le premier avantage est que ces règles peuvent être construites directement avec suffisamment de connaissances spécialisées, et elles sont interprétables et claires pour les autres experts du réseau ou du domaine. Cependant, les inconvénients sont nombreux. Le premier si le réseau est suffisamment grand, l'expertise et le temps requis pour construire le système expert sont énormes, le maintenir est encore plus difficile. Deuxièmement, si une nouvelle panne se produit en raison de changements dans le réseau, le système doit être revisité, ce qui peut s'avérer encore plus complexe. Troisièmement, puisque les règles sont déterministes, le système expert ne peut pas toujours conclure lorsque certaines variables sont manquantes.

La première approche qui a fait l'objet de nombreuses recherches pour la tâche de diagnostic fait appel aux réseaux bayésiens. Un réseau bayésien est un modèle probabiliste des variables de réseau avec dépendances. Le modèle peut être construit par un expert, les dépendances représentent des relations causales entre les variables du réseau. L'objectif d'un réseau bayésien est de modéliser la propagation des pannes à travers les variables du réseau bayésien, ainsi en inversant le processus, on peut identifier la cause racine. L'avantage de ces modèles est qu'ils peuvent être appris à

partir de données (probabilités conditionnelles estimées à partir de données). Cependant, l'inconvénient reste le même que les systèmes experts : ils sont très spécifiques à l'application, si une nouvelle panne ou un changement survient sur le réseau réel, le réseau bayésien doit être revisité ce qui est souvent une tâche coûteuse.

Des méthodes classiques d'apprentissage automatique ont également été explorées pour la tâche de diagnostic. Une approche consiste à considérer le schéma d'apprentissage supervisé traditionnel, où un classificateur est appris à l'aide de données étiquetées selon les différents types de pannes connus. Ces classificateurs sont nombreux et peuvent aller de modèles simples tels que les classificateurs Naive-Bayes, aux Support-Vector Machines (SVM), et même aux réseaux de neurones.

Bien que ces méthodes soient attrayantes, en utilisant ces approches, nous aurions besoin de réapprendre notre modèle de classification avec de nouvelles données étiquetées pour de nouveaux types de pannes, qui ne sont généralement pas disponibles, surtout lorsque la panne est très récente. En utilisant les réseaux bayésiens, la prise en compte de nouveaux types de pannes, de nouvelles données ou de nouveaux équipements reviendrait à mettre à jour le graphe bayésien en ajoutant et/ou en supprimant des nœuds et des arêtes ainsi qu'en modifiant les probabilités conditionnelles des arêtes. Cela nécessiterait une grande expertise du réseau et peut être irréaliste si le réseau est trop complexe. Bien qu'il soit possible de dériver le graphe bayésien à partir de données, cela devient impossible en pratique lorsqu'il s'agit de grands ensembles de données composés de centaines ou de milliers d'entités.

D'autre part, afin de découvrir de nouvelles pannes, il faut effectuer une exploration des données et trouver des clusters correspondant à de nouveaux types de pannes. L'approche standard actuellement, consiste pour un expert à étudier les données "à la main" afin d'identifier ces clusters. Les méthodes de clustering non supervisées peuvent être utilisées pour obtenir de bonnes performances. Cependant, l'utilisation de telles méthodes dépend souvent d'hypothèses de modèle qui pourraient nuire aux performances, telles que les métriques de distance dans les espaces de grande dimension. De plus, dans les méthodes non supervisées, les clusters peuvent être pertinents d'un point de vue technique (par exemple, regrouper des individus parce qu'ils utilisent le même type d'équipement), mais ce type de clustering n'apporte aucune information nouvelle qui pourrait être utilisée pour mettre en évidence une nouvelle panne. Le moyen évident de contourner ce problème est de recourir à une sélection fastidieuse de variables caractéristiques et à un prétraitement manuel des données, ce qui nécessite une connaissance approfondie de l'environnement en constante évolution et peut être coûteux en temps.

Dans cette thèse, nous présentons un cadre général basé sur des modèles probabilistes graphiques pour apprendre et identifier des clusters de pannes. Nous plaidons en faveur d'un type spécifique de modèles probabilistes graphiques, à savoir les mélanges à base de processus de Dirichlet ou les mélanges profonds à base de processus de Dirichlet. Ces modèles que nous proposons sont capables de clusteriser les pannes sans connaître le nombre de clusters (ou de pannes), ils peuvent identifier des signatures cachées pertinentes pour la tâche de diagnostic, et plus important encore, ils incluent des connaissances expertes en termes de supervision (ou semi-supervision) sous forme d'étiquettes pour guider et améliorer l'identification des pannes à partir des données.

STRUCTURE DE LA THÈSE ET QUESTIONS DE RECHERCHES

Cette thèse est divisée en deux parties: dans la partie I, nous introduisons des modèles probabilistes graphiques superficiels. Plus précisément, des modèles de mélange infini capables d'identifier des clusters de pannes à partir de données de réseau, de manière non supervisée et semi-supervisée. Dans la partie II, nous faisons évoluer ces modèles pour inclure des représentations cachées profondes. Le but des représentations cachées est de construire des signatures pertinentes pour la tâche de diagnostic. Ces fonctionnalités sont conçues pour traiter des données à grande échelle bruitées et complexes. Combiné avec les mélanges infinis, nous pouvons utiliser ces représentations pour obtenir de meilleurs résultats sur de tels ensembles de données. Afin de guider la lecture du manuscrit, les contributions apportées dans la partie I tentent de répondre aux questions de recherche suivantes:

Question 1 : Peut-on développer des modèles capables d'identifier des clusters adaptés aux données du réseau, notamment des données catégorielles, sans connaître a priori le nombre de clusters ?

Notre contribution pour répondre à cette question est le modèle de mélange infini catégoriel. Le modèle est présenté au chapitre 3. Nous étendons la définition des mélanges à base de processus de Dirichlet au cas des variables catégorielles multidimensionnelles et nous montrons comment utiliser l'inférence variationnelle de type champ-moyen pour apprendre le modèle efficacement. Nous montrons que notre modèle est capable d'identifier des clusters de pannes sans connaître a priori le nombre de clusters et nous montrons comment les hypothèses du modèle conduisent à de meilleures performances en termes de précision.

Question 2 : Peut-on étendre les mélanges à base de processus de Dirichlet pour traiter simultanément des variables continues et catégorielles, et

prendre également en compte les connaissances expertes en termes de supervision ?

Dans cette contribution, nous étendons le modèle de mélange catégoriel infini à un modèle de type mixte, qui peut traiter des variables continues et catégorielles. Le modèle prend également en compte la semi-supervision d'experts sous forme de labels. Nous évaluons le modèle sur les données du réseau et nous montrons ses très bonnes performances comparées à des modèles références de l'état de l'art. Ce travail est présenté dans le chapitre 4.

Nous avons introduit des modèles probabilistes graphiques superficiels, sous la forme de modèles de mélanges infinis, qui pourraient être appliqués à des données de type mixte. Un problème demeure, ces modèles ne peuvent pas créer de combinaisons complexes de données qui seraient nécessaires pour découvrir des clusters. En d'autres termes, aucune création de signatures n'est possible. Cependant, dans la plupart des applications du monde réel, les données sont bruitées et de grande dimensionnalité. Ainsi, la création de signatures ou "features" est nécessaire. Dans la partie II, nous développons des modèles et des méthodes afin d'atteindre cet objectif, et la principale question de recherche abordée est la suivante:

Question 3: Pouvons-nous construire des représentations ou des signatures très pertinentes pour la tâche de diagnostic, afin d'améliorer les performances ?

Afin de répondre à cette question, dans le chapitre 5, nous combinons des représentations de l'apprentissage profond et des modèles de mélange infini pour effectuer la découverte de pannes dans un ensemble de données à grande échelle partiellement étiqueté. Nous parlons pour cette approche de modèles de mélanges infinis profonds. Nous créons un réseau de neurones profond pour reconnaître les pannes étiquetées et construire des représentations cachées de faible dimension pertinentes pour la tâche de diagnostic. Ensuite, un modèle de mélange gaussien infini est appliqué sur la partie non étiquetée afin de reconnaître les clusters de pannes sur la base de la représentation cachée. Nous évaluons le modèle sur un ensemble de données GPON-FTTH du monde réel à grande échelle, nous montrons que le modèle peut identifier des classes de pannes connues avec une grande précision, et que la représentation cachée sur les pannes non étiquetées conduit à des clusters reconnaissables.

Le deep learning et les mélanges infinis vivent dans deux univers de méthodes différents. Leur combinaison n'est pas naturelle, dans le sens où nous n'avons pas de modèle global de bout en bout, capable d'apprendre des signatures et de faire du clustering. Ainsi se pose naturellement la question suivante:

Question 4: Peut-on construire un modèle probabiliste profond de bout en bout capable d'apprendre des représentations cachées et d'effectuer du clustering ?

Il s'avère que cela est possible grâce aux modèles génératifs profonds, qui sont une classe de modèles probabilistes graphiques. Le problème des modèles génératifs profonds est le processus d'apprentissage qui implique le calcul des estimateurs de gradients qui sont souvent de grande variance. Dans nos contributions Rétropropagation stochastique généralisée et Rétropropagation stochastique par transformées de Fourier, nous présentons une méthode pour dériver des estimateurs à faible variance de ces gradients. Ce travail est présenté au chapitre 6.

Dans le chapitre 7, nous utilisons la rétropropagation stochastique généralisée pour former notre modèle de mélange latent profond à base de processus de Dirichlet de bout en bout. Nous évaluons notre modèle sur des ensembles de données ouverts et sur des données GPON-FTTH à grande échelle du monde réel pour la tâche de diagnostic et de découverte des pannes.

Enfin, au chapitre 8, nous discutons de l'interprétabilité de nos modèles. Nous montrons que les modèles superficiels sont naturellement interprétables. Les modèles profonds, cependant, sont plus difficiles en termes d'interprétabilité. Nous discutons de la façon d'utiliser les arbres de décision pour retracer la décision menant à un cluster de pannes, la rendant compréhensible pour un expert du domaine.

BIBLIOGRAPHY

Adda, Mo, Karwan Qader, and Mouhammd Al-Kasassbeh (2017). "Comparative analysis of clustering techniques in network traffic faults classification." In: *International Journal of Innovative Research in Computer and Communication Engineering* 5.4, pp. 6551–6563.

Agarwal, Rishabh, Nicholas Frosst, Xuezhou Zhang, Rich Caruana, and Geoffrey E Hinton (2020). "Neural additive models: Interpretable machine learning with neural nets." In: *arXiv preprint arXiv:2004.13912*.

Aggarwal, Charu C, Alexander Hinneburg, and Daniel A Keim (2001). "On the surprising behavior of distance metrics in high dimensional space." In: *International conference on database theory*. Springer, pp. 420–434.

Asadi, Kavosh et al. (2017). "Mean actor critic." In: *stat* 1050, p. 1.

Ayoubi, S. et al. (2018). "Machine Learning for Cognitive Network Management." In: *IEEE Communications Magazine* 56.1, pp. 158–165.

Baras, John S, M Ball, S Gupta, P Viswanathan, and P Shah (1997). "Automated network fault management." In: *MILCOM 97 MILCOM 97 Proceedings*. Vol. 3. IEEE, pp. 1244–1250.

Basu, Sugato, Arindam Banerjee, and Raymond J Mooney (2004). "Active semi-supervision for pairwise constrained clustering." In: *Proceedings of the 2004 SIAM international conference on data mining*. SIAM, pp. 333–344.

Basu, Sugato, Mikhail Bilenko, and Raymond J Mooney (2004). "A probabilistic framework for semi-supervised clustering." In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 59–68.

Baydin, Atilim Gunes, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind (2018). "Automatic differentiation in machine learning: a survey." In: *Journal of machine learning research* 18.

Bengio, Yoshua, Olivier Delalleau, and Nicolas Le Roux (2006). "11 label propagation and quadratic criterion." In:

Bennacer, Leila, Yacine Amirat, Abdelghani Chibani, Abdelhamid Mellouk, and Laurent Ciavaglia (2014). "Self-diagnosis technique for virtual private networks combining Bayesian networks and case-based reasoning." In: *IEEE Transactions on Automation Science and Engineering* 12.1, pp. 354–366.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. springer.

Blei, David M, Michael I Jordan, et al. (2006). "Variational inference for Dirichlet process mixtures." In: *Bayesian analysis* 1.1, pp. 121–143.

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). "Variational inference: A review for statisticians." In: *Journal of the American statistical Association* 112.518, pp. 859–877.

Boutaba, Raouf et al. (2018). "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities." In: *Journal of Internet Services and Applications* 9.1, p. 16.

Braun, Michael and Jon McAuliffe (2010). "Variational inference for large-scale models of discrete choice." In: *Journal of the American Statistical Association* 105.489, pp. 324–335.

Breiman, Leo, Jerome Friedman, Charles J Stone, and Richard A Olshen (1984). *Classification and regression trees*. CRC press.

Brockman, Greg et al. (2016). *OpenAI Gym*. eprint: arXiv:1606.01540.

Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov (2015). "Importance weighted autoencoders." In: *arXiv preprint arXiv:1509.00519*.

Chanclou, P., S. Gosselin, J. F. Palacios, V. L. Alvarez, and E. Zouganeli (2006). "Overview of the optical broadband access evolution: a joint article by operators in the IST network of excellence e-Photon/One." In: *IEEE Communications Magazine* 44.8, pp. 29–35.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (July 2009a). "Anomaly Detection: A Survey." In: *ACM Comput. Surv.* 41.3. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (July 2009b). "Anomaly Detection: A Survey." In: *ACM Comput. Surv.* 41.3. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

Chapelle, Olivier, Bernhard Scholkopf, and Alexander Zien (2009). "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]." In: *IEEE Transactions on Neural Networks* 20.3, pp. 542–542.

Chen, Mike Y, Emre Kiciman, Eugene Fratkin, Armando Fox, and Eric Brewer (2002). "Pinpoint: Problem determination in large, dynamic internet services." In: *Proceedings International Conference on Dependable Systems and Networks*. IEEE, pp. 595–604.

Chen, Mike, Alice X Zheng, Jim Lloyd, Michael I Jordan, and Eric Brewer (2004). "Failure diagnosis using decision trees." In: *International Conference on Autonomic Computing, 2004. Proceedings.* IEEE, pp. 36–43.

Cherrared, S., S. Imadali, E. Fabre, G. Gössler, and I. G. B. Yahia (2019). "A Survey of Fault Management in Network Virtualization Environments: Challenges and Solutions." In: *IEEE Transactions on Network and Service Management* 16.4, pp. 1537–1551.

Comaniciu, Dorin and Peter Meer (2002). "Mean shift: A robust approach toward feature space analysis." In: *IEEE Transactions on pattern analysis and machine intelligence* 24.5, pp. 603–619.

Cong, Yulai, Miaoyun Zhao, Ke Bai, and Lawrence Carin (2019). "GO gradient for expectation-based objectives." In: *arXiv preprint arXiv:1901.06020*.

Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks." In: *Machine learning* 20.3, pp. 273–297.

Dilokthanakul, Nat et al. (2016). "Deep unsupervised clustering with gaussian mixture variational autoencoders." In: *arXiv preprint arXiv:1611.02648*.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Echraibi, Amine, Joachim Cholet, Stéphane Gosselin, and Sandrine Vaton (2020a). "Generalized Stochastic Backpropagation." In: *Neural Information Processing 2020 Beyond Backprop workshop*.

Echraibi, Amine, Joachim Flocon-Cholet, Stéphane Gosselin, and Sandrine Vaton (Nov. 2019). "Bayesian Mixture Models For Semi-Supervised Clustering." working paper or preprint. URL: https://hal.archives-ouvertes.fr/hal-02372337.

Echraibi, Amine, Joachim Flocon-Cholet, Stéphane Gosselin, and Sandrine Vaton (2020b). "An Infinite Multivariate Categorical Mixture Model for Self-Diagnosis of Telecommunication Networks." In: *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, pp. 258–265.

Echraibi, Amine, Joachim Flocon-Cholet, Stéphane Gosselin, and Sandrine Vaton (2020c). "On the Variational Posterior of Dirichlet Process Deep Latent Gaussian Mixture Models." In: *International Conference on Machine Learning 2020 INNF workshop*.

Echraibi, Amine, Joachim Flocon-Cholet, Stéphane Gosselin, and Sandrine Vaton (2021a). "Stochastic Backpropagation through Fourier Transforms." In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE.

Echraibi, Amine, Joachim Flocon-Cholet, Stéphane Gosselin, and Sandrine Vaton (2021b). "Deep Infinite Mixture Models for Fault Discovery in GPON-FTTH Networks." In: *IEEE Access* 9, pp. 90488–90499. DOI: 10.1109/ACCESS.2021.3091328.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34, pp. 226–231.

Fellows, Matthew, Kamil Ciosek, and Shimon Whiteson (2018). "Fourier policy gradients." In: *arXiv preprint arXiv:1802.06891*.

Ferguson, Thomas S (1973). "A Bayesian analysis of some nonparametric problems." In: *The annals of statistics*, pp. 209–230.

Feynman, RP, RB Leighton, and M Sands (1964). *The Feynman Lectures on Physics, Vol. II, Sec. 5-8*.

Figurnov, Mikhail, Shakir Mohamed, and Andriy Mnih (2018). "Implicit reparameterization gradients." In: *Advances in Neural Information Processing Systems*, pp. 441–452.

Fong, Ruth C and Andrea Vedaldi (2017). "Interpretable explanations of black boxes by meaningful perturbation." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437.

Fong, Ruth, Mandela Patrick, and Andrea Vedaldi (2019). "Understanding deep networks via extremal perturbations and smooth masks." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2950–2958.

Fox, Emily B, Erik B Sudderth, Michael I Jordan, and Alan S Willsky (2008). "An HDP-HMM for systems with state persistence." In: *Proceedings of the 25th international conference on Machine learning*, pp. 312–319.

Frey, Brendan J and Delbert Dueck (2007). "Clustering by passing messages between data points." In: *science* 315.5814, pp. 972–976.

Fung, Robert and Kuo-Chu Chang (1990). "Weighing and integrating evidence for stochastic simulation in Bayesian networks." In: *Machine Intelligence and Pattern Recognition*. Vol. 10. Elsevier, pp. 209–219.

Geman, Stuart and Donald Geman (1984). "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." In: *IEEE Transactions on pattern analysis and machine intelligence* 6, pp. 721–741.

Gigabit-capable passive optical networks (G-PON): ONT management and control interface specification (2008). ITU-T.

Gigabit-capable passive optical networks (GPON): General characteristics (2008). ITU-T.

Glynn, Peter W (1989). "Optimization of stochastic systems via simulation." In: *Proceedings of the 21st conference on Winter simulation*, pp. 90–105.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio (2016). *Deep learning*. Vol. 1. 2. MIT press Cambridge.

Gori, Marco, Gabriele Monfardini, and Franco Scarselli (2005). "A new model for learning in graph domains." In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. IEEE, pp. 729–734.

Grathwohl, Will, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud (2018). "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation." In:

Graves, Alex (2016). "Stochastic backpropagation through mixture density distributions." In: *arXiv preprint arXiv:1607.05690*.

Gu, Shixiang, Sergey Levine, Ilya Sutskever, and Andriy Mnih (2016). "MuProp: Unbiased Backpropagation for Stochastic Neural Networks." In:

Hashmi, Umair Sajid, Arsalan Darbandi, and Ali Imran (2017). "Enabling proactive self-healing by data mining network failure logs." In: *2017 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, pp. 511–517.

Hastings, W Keith (1970). "Monte Carlo sampling methods using Markov chains and their applications." In:

Hood, Cynthia S and Chuanyi Ji (1997). "Proactive network-fault detection [telecommunications]." In: *IEEE Transactions on reliability* 46.3, pp. 333–341.

Jaakkola, T (2001). "tutorial on variational approximation methods." In: *Advanced mean field methods: theory and practice*, p. 129.

Jaakkola, Tommi and Michael Jordan (1997). "A variational approach to Bayesian logistic regression models and their extensions." In: *Sixth International Workshop on Artificial Intelligence and Statistics*. Vol. 82. 4.

Jang, Eric, Shixiang Gu, and Ben Poole (2016). "Categorical reparameterization with gumbel-softmax." In: *arXiv preprint arXiv:1611.01144*.

Jankowiak, Martin and Theofanis Karaletsos (2019). "Pathwise Derivatives for Multivariate Distributions." In: *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 333–342.

Jankowiak, Martin and Fritz Obermeyer (2018). "Pathwise Derivatives Beyond the Reparameterization Trick." In: *International Conference on Machine Learning*, pp. 2235–2244.

Jiang, Zhuxi, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou (2016). "Variational deep embedding: An unsupervised and generative approach to clustering." In: *arXiv preprint arXiv:1611.05148*.

Joachims, Thorsten (1999). "Transductive inference for text classification using support vector machines." In: *Icml*. Vol. 99, pp. 200–209.

Jordan, Michael I, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul (1999). "An introduction to variational methods for graphical models." In: *Machine learning* 37.2, pp. 183–233.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114*.

Kingma, Durk P, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling (2014). "Semi-supervised learning with deep generative models." In: *Advances in neural information processing systems*, pp. 3581–3589.

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks." In: *arXiv preprint arXiv:1609.02907*.

Kogeda, Okuthe P, Johnson I Agbinya, and Christian W Omlin (2006). "A probabilistic approach to faults prediction in cellular networks." In: *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*. IEEE, pp. 130–130.

Kogeda, P and Johnson I Agbinya (2006). "Prediction of faults in cellular networks using bayesian network model." In: *International conference on Wireless Broadband and Ultra Wideband Communication*. UTS ePress.

Koller, Daphne and Nir Friedman (2009). *Probabilistic graphical models: principles and techniques*.

Kriegel, Hans-Peter, Peer Kröger, and Arthur Zimek (2009). "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.1, pp. 1–58.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25, pp. 1097–1105.

Kumaraswamy, Ponnambalam (1980). "A generalized probability density function for double-bounded random processes." In: *Journal of Hydrology* 46.1-2, pp. 79–88.

Lake, Brenden M, Russ R Salakhutdinov, and Josh Tenenbaum (2013). "One-shot learning by inverting a compositional causal process." In: *Advances in neural information processing systems*, pp. 2526–2534.

Larochelle, Hugo, Dumitru Erhan, and Yoshua Bengio (2008). "Zero-data learning of new tasks." In: *AAAI*. Vol. 1. 2, p. 3.

LeCun, Yann and Corinna Cortes (2010). "MNIST handwritten digit database." In: URL: `http://yann.lecun.com/exdb/mnist/`.

Lundberg, Scott and Su-In Lee (2017). "A unified approach to interpreting model predictions." In: *arXiv preprint arXiv:1705.07874*.

Maaten, Laurens van der and Geoffrey Hinton (2008a). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Maaten, Laurens Van der and Geoffrey Hinton (2008b). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11.

MacKay, David JC (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Maddison, Chris J, Andriy Mnih, and Yee Whye Teh (2016). "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables." In:

Miller, Jeffrey W. and Matthew T. Harrison (2015). *Mixture models with a prior on the number of components*. arXiv: `1502.06241 [stat.ME]`.

Mnih, Andriy and Karol Gregor (2014). "Neural variational inference and learning in belief networks." In: *Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32*, pp. II–1791.

Mnih, Andriy and Danilo J Rezende (2016). "Variational inference for monte carlo objectives." In: *arXiv preprint arXiv:1602.06725*.

Mnih, Volodymyr, Nicolas Heess, Alex Graves, et al. (2014). "Recurrent models of visual attention." In: *Advances in neural information processing systems*, pp. 2204–2212.

Mohamed, Shakir, Mihaela Rosca, Michael Figurnov, and Andriy Mnih (2019). "Monte carlo gradient estimation in machine learning." In: *arXiv preprint arXiv:1906.10652*.

Mosterman, Pieter J and Gautam Biswas (1999). "Diagnosis of continuous valued systems in transient operating regions." In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 29.6, pp. 554–565.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

Naesseth, CA, FJR Ruiz, SW Linderman, and DM Blei (2017). "Reparameterization gradients through acceptance-rejection sampling algorithms." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*.

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines." In: *Icml*.

Nalisnick, Eric, Lars Hertel, and Padhraic Smyth (2016). "Approximate inference for deep latent gaussian mixtures." In: *NIPS Workshop on Bayesian Deep Learning*. Vol. 2.

Nalisnick, Eric and Padhraic Smyth (2016). "Stick-breaking variational autoencoders." In: *arXiv preprint arXiv:1605.06197*.

Neal, Radford M (1992). "Connectionist learning of belief networks." In: *Artificial intelligence* 56.1, pp. 71–113.

Neal, Radford M (1993). *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science University of Toronto Toronto, Ontario, Canada.

Neal, Radford M (2000). "Markov chain sampling methods for Dirichlet process mixture models." In: *Journal of computational and graphical statistics* 9.2, pp. 249–265.

Netzer, Yuval et al. (2011). "Reading digits in natural images with unsupervised feature learning." In:

Nigam, Kamal, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell (2000). "Text classification from labeled and unlabeled documents using EM." In: *Machine learning* 39.2-3, pp. 103–134.

Paisley, John, David Blei, and Michael Jordan (2012). "Variational Bayesian inference with stochastic search." In: *arXiv preprint arXiv:1206.6430*.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Ranganath, Rajesh, Sean Gerrish, and David Blei (2014). "Black box variational inference." In: *Artificial Intelligence and Statistics*. PMLR, pp. 814–822.

Rasmussen, Carl Edward et al. (1999). "The infinite Gaussian mixture model." In: *NIPS*. Vol. 12, pp. 554–560.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In: *International Conference on Machine Learning*, pp. 1278–1286.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). ""Why should i trust you?" Explaining the predictions of any classifier." In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method." In: *The annals of mathematical statistics*, pp. 400–407.

Roeder, Geoffrey, Yuhuai Wu, and David K Duvenaud (2017). "Sticking the landing: Simple, lower-variance gradient estimators for variational inference." In: *Advances in Neural Information Processing Systems*, pp. 6925–6934.

Rosenblatt, Frank (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY.

Ruiz, Francisco R, Michalis Titsias RC AUEB, and David Blei (2016). "The generalized reparameterization gradient." In: *Advances in neural information processing systems*, pp. 460–468.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Saint Raymond, Xavier (2018). "Elementary introduction to the theory of pseudodifferential operators." In: Routledge. Chap. 1, pp. 2–3.

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2008). "The graph neural network model." In: *IEEE transactions on neural networks* 20.1, pp. 61–80.

Schulman, John, Nicolas Heess, Theophane Weber, and Pieter Abbeel (2015). "Gradient estimation using stochastic computation graphs." In: *Advances in Neural Information Processing Systems*, pp. 3528–3536.

Seeger, Matthias (2006). *A taxonomy for semi-supervised learning methods*. Tech. rep. MIT Press.

Selvaraju, Ramprasaath R et al. (2017). "Grad-cam: Visual explanations from deep networks via gradient-based localization." In: *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.

Sethuraman, Jayaram (1994). "A constructive definition of Dirichlet priors." In: *Statistica sinica*, pp. 639–650.

Socher, Richard, Milind Ganjoo, Christopher D Manning, and Andrew Ng (2013). "Zero-shot learning through cross-modal transfer." In: *Advances in neural information processing systems*, pp. 935–943.

Steinder, Malgorzata and Adarshpal S. Sethi (2004). "A survey of fault localization techniques in computer networks." In: *Sci. Comput. Program.* 53, pp. 165–194.

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton (2013). "On the importance of initialization and momentum in deep learning." In: *International conference on machine learning*. PMLR, pp. 1139–1147.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, Richard S, David A McAllester, Satinder P Singh, and Yishay Mansour (2000). "Policy gradient methods for reinforcement learning with function approximation." In: *Advances in neural information processing systems*, pp. 1057–1063.

Tembo, S. R., S. Vaton, J. L. Courant, S. Gosselin, and M. Beuvelot (2017). "Model-Based Probabilistic Reasoning for Self-Diagnosis of Telecommunication Networks: Application to a GPON-FTTH Access Network." In: *Journal of Network and Systems Management* 25.3, pp. 558–590. ISSN: 1573-7705. DOI: 10.1007/s10922-016-9401-0.

Titsias, Michalis and Miguel Lázaro-Gredilla (2014). "Doubly stochastic variational Bayes for non-conjugate inference." In: *International conference on machine learning*, pp. 1971–1979.

Titsias, Michalis and Miguel Lázaro-Gredilla (2015). "Local expectation gradients for black box variational inference." In: *Advances in neural information processing systems*, pp. 2638–2646.

Tokui, Seiya and Issei Sato (2017). "Evaluating the variance of likelihood-ratio gradient estimators." In: *International Conference on Machine Learning*, pp. 3414–3423.

Tucker, George, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein (2017). "Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models." In: *Advances in Neural Information Processing Systems*, pp. 2627–2636.

Vaswani, Ashish et al. (2017). "Attention is all you need." In: *arXiv preprint arXiv:1706.03762*.

Wagstaff, Kiri, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. (2001). "Constrained k-means clustering with background knowledge." In: *Icml*. Vol. 1, pp. 577–584.

Wainwright, Martin J, Michael I Jordan, et al. (2008). "Graphical models, exponential families, and variational inference." In: *Foundations and Trends® in Machine Learning* 1.1–2, pp. 1–305.

Weaver, Lex and Nigel Tao (2013). "The optimal reward baseline for gradient-based reinforcement learning." In: *arXiv preprint arXiv:1301.2315*.

Werbos, Paul J (1982). "Applications of advances in nonlinear sensitivity analysis." In: *System modeling and optimization*. Springer, pp. 762–770.

Williams, Ronald J (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4, pp. 229–256.

Xian, Yongqin, Bernt Schiele, and Zeynep Akata (2017). "Zero-shot learning-the good, the bad and the ugly." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4582–4591.

Xie, Junyuan, Ross Girshick, and Ali Farhadi (2016). "Unsupervised deep embedding for clustering analysis." In: *International conference on machine learning*, pp. 478–487.

Yosinski, Jason, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson (2015). "Understanding neural networks through deep visualization." In: *arXiv preprint arXiv:1506.06579*.

Zhang, Aonan, San Gultekin, and John Paisley (2016). "Stochastic variational inference for the HDP-HMM." In: *Artificial Intelligence and Statistics*, pp. 800–808.

Zhang, Harry (2005). "Exploring conditions for the optimality of naive Bayes." In: *International Journal of Pattern Recognition and Artificial Intelligence* 19.02, pp. 183–198.

Zhur, Xiaojin and Zoubin Ghahramani (2002). "Learning from labeled and unlabeled data with label propagation." In:

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Sandrine Vaton and my team at Orange Labs for giving me the opportunity to pursue a PhD under their supervision. I am tremendously grateful for their support, and for the continuous encouragement to pursue challenging ideas. Special thanks also to my Orange Labs advisers Joachim Flocon-Cholet and Stéphane Gosselin for their support, encouragement, and valuable discussions and insights through the years. My time at Orange Labs has truly been a wonderful and fulfilling experience thanks to them all.

I would also like to thank my family who has supported me through the years across my academic journey, and especially during the unusual circumstances under which the thesis took place. Without the many welcomed distractions, calls, jokes, the stress of the situation at the time would have surely gotten the better of me. I would like to thank my sister Meriem for her unwavering optimism and support, without her I would be a different person today. Finally, I would like to thank my parents, Omar and Khadija for their support, encouragements, and the sacrifices that they had to endure for me to be where I am today. A debt that I would never be able to repay!

Sincerely,
Amine Echraibi

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Les modèles probabilistes graphiques : théorie et application au diagnostic réseau

Mots clés : modèles probabilistes graphiques, diagnostic réseau, inférence variationnelle.

Résumé : Pour tout fournisseur d'accès Internet ou opérateur de réseau, il est crucial d'identifier rapidement et efficacement les problèmes qui surviennent sur le réseau. Les avantages d'un bon système de diagnostic des pannes sont principalement de minimiser les coûts d'exploitation du réseau et d'améliorer la qualité de l'expérience du client. Un défi majeur pour tout système de diagnostic concerne la découverte de nouvelles pannes, inconnus de la version actuelle du système de diagnostic. Le processus exploratoire pour trouver de nouvelles pannes peut s'avérer coûteux et long pour les fournisseurs de services Internet. Dans cette thèse, nous explorons une approche alternative basée sur des méthodes d'apprentissage, afin de construire des systèmes de diagnostic autonomes. Notre étude explore des modèles graphiques probabilistes capables de regrouper des motifs de pannes de manière non supervisée et semi-supervisée. Nous démontrons l'efficacité de nos modèles sur des cas d'utilisation réels de données à grande échelle, extraites de réseaux et de services 'Fibre-to-the-Home' (FTTH).

Title : Probabilistic Graphical Models: Theory and Application to Network Diagnosis

Keywords : Probabilistic Graphical Models, Network Diagnosis, Variational Inference.

Abstract : For any Internet service provider or network operator, it is crucial to quickly and efficiently diagnose the problems that occur on the network. The benefits of a good fault diagnosis system are mainly to minimize the costs of network and service operations and to enhance the customer's quality of experience. One major challenge for any diagnosis system concerns the discovery of new faults, that are unknown to the current version of the diagnosis system. The exploratory process for finding new faults can prove to be expensive and time consuming for internet service providers. In this thesis, we explore an alternative approach based on learning methods, in order to build learning-based diagnosis systems. Our study explores Probabilistic Graphical Models that are capable of clustering patterns of faults in an unsupervised and a semi-supervised manner. We demonstrate the efficiency of our models on real use-cases of large scale data, extracted from Fiber-to-the Home (FTTH) services based on Gigabit-capable Passive Optical Networks.