Probability density imputation of missing data with Gaussian Mixture Models

Adrià Garriga Alonso
Hertford College

Supervised by
Prof. Mihaela van der Schaar

Co-supervised by
Prof. Edith Elkind

Trinity Term 2017

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Acknowledgements

First of all, I thank my supervisor, Prof. Mihaela van der Schaar, for her invaluable guidance and her keen eye for interesting problems. I also thank my supervising student, Jinsung Yoon, for his many correct intuitions that saved me hours of exploration, and good explanations so that I could learn.

I also thank my co-supervisor, Prof. Edith Elkind, for her excellent actionable advice, given in the direst of moments. On the same note, I thank Sarah Retz-Jones, the coordinator of the MSc in CS, for easing my biggest worries, and ensuring everything runs smoothly so that I don't have more to worry about on top.

I thank my co-students Alex, Eleonora, Michael and Qixuan, for being excellent people, and solidary in our common struggle to do excellent work. I thank my friend Kholood, who is as diligent as no other, and always had a solution to my writing troubles; and my friend Paco, for his effective encouragement and motivation. I thank my friends from Hertford College, Alex, Cécile, David, Devesh, Maalik and Rayner, for listening to my anxious rants, and preventing a breakdown or two.

Finally, I thank my partner Tanja for always being there, and my parents Miquel, Isabel, and sister Núria for their continuous support. Thank you all!

Abstract

Multiple imputation (Rubin, 2004) is a procedure for conducting unbiased data analysis of an incomplete data set $\mathbf{X}$, using complete-data analysis techniques. It consists in generating m imputed data sets in which the missing values $\mathbf{X}^{\text{mis}}$ have been replaced by plausible values, analysing the imputed data sets, and combining the results. To assert the soundness of this approach, the plausible values are assumed to be drawn from the posterior distribution of the missing values given the data, $p(\mathbf{X}^{\text{mis}} \mid \mathbf{X})$. Typically, $m \leq 10$ samples are used.

In this thesis we go one step further, and set $m \to \infty$. That is, we attempt to estimate the distribution $p(\mathbf{X}^{\text{mis}} \mid \mathbf{X})$ with a closed-form probability distribution, performing *probability density imputation*. To this end, we provide the Partially Observed Infinite Gaussian Mixture Model (POIGMM), an algorithm for (1) density estimation from incomplete data sets, and (2) density imputation; based on the Infinite GMM by Blei and Jordan (2006).

A density-imputed data set can easily be reduced to a multiple-imputed data set, by taking m samples from the estimated posterior distribution. Accordingly, we compare the POIGMM with the state of the art in multiple imputation: MissForest (Stekhoven and Bühlmann, 2011) and MICE (Van Buuren and Oudshoorn, 1999).

Contents

Acknowledgements **v**

Abstract **vii**

Contents **ix**

Abbreviations **xiii**

1 Introduction **1**

1.1 Goals and contributions 4

1.2 Intuition behind the POIGMM for imputation 4

1.3 Document structure 5

2 Background **7**

2.1 Preliminaries 7

2.1.1 Notational conventions 7

2.1.2 Basic probability theory 8

2.1.3 Supervised machine learning 8

2.2 Probabilistic Models 10

2.2.1 Maximum likelihood estimation 10

2.2.2 Bayesian inference 11

2.2.3 Bayesian networks 13

2.2.4 Mixture model 14

2.2.5 Variational inference 14

2.2.6 Exponential and conjugate families of distributions 16

2.3 Important probability distributions 17

2.3.1 Multivariate Gaussian 17

2.3.2 Wishart 18

2.4 Missing data 19

2.4.1 Types of missing data: MCAR, MAR, NMAR 20

2.4.2 Handling missing data 21

3 Partially Observed Infinite GMM **25**
3.1 The Dirichlet Process Mixture Model 26
3.1.1 The Dirichlet Process Prior 26
3.1.2 Joint likelihood of a DP mixture 27
3.1.3 The variational distribution 29
3.2 Inference in the partially observed infinite GMM 30
3.2.1 Optimal cluster assignment ($\boldsymbol{\tau}$) update 30
3.2.2 Optimising the weight distribution parameters ($\boldsymbol{\gamma}$) 32
3.2.3 Optimising the component parameters ($\mathbf{m}, \beta, \mathbf{W}, \nu$) 32
3.3 Imputation 35
3.3.1 Semi-Bayesian imputation 36
3.4 Summary 37

4 Experiments **39**
4.1 Data sets 39
4.2 Methodology 40
4.2.1 Creating missing data 40
4.2.2 Algorithms 41
4.2.3 Metrics 41
4.3 Results and discussion 42
4.3.1 How to read the graphs 43
4.3.2 Discussion 43

5 Related Work **45**
5.1 Multiple Imputation by Chained Equations (MICE) 45
5.2 MissForest: Random Forests for imputation 46

6 Conclusions **47**
6.1 Lessons Learned 47
6.2 Future Directions 48

A Experiments **49**

B Mathematical definitions **54**
B.1 Probability distributions 54
B.1.1 Gaussian-Wishart 54
B.1.2 The Beta distribution 55

B.1.3 Multinomial Distribution 55
B.2 Functions 56
B.3 Linear Algebra 56

C Extra derivations **57**
C.1 Expectation of the logarithm of the normal 57

Bibliography **61**

Abbreviations

CI Conditional Independence

DP Dirichlet Process

ELBO Evidence Lower Bound

GMM Gaussian Mixture Model

i.i.d. independently identically distributed

KL Kullback Leibler

MAP Maximum *a posteriori*

MAR Missing at random

MCAR Missing completely at random

MICE Multiple Imputation by Chained Equations

ML Machine Learning

MLE Maximum Likelihood Estimation

MSE Mean Squared Error

NMAR Not missing at random

POIGMM Partially Observed Infinite Gaussian Mixture Model

PSD positive semi-definite

RF Random Forest

RV Random Variable

SGD Stochastic Gradient Descent

Chapter 1

Introduction

Data collected from the real world is often messy. For example:

- A hospital has 10 possible tests that can be administered to patients. Each of them is designed to identify one specific disease, and is given when the patient shows possible symptoms of the disease. None of them, however, is perfectly accurate: each fails to correctly diagnose patients as sick, or healthy, 5% of the time. The hospital hired an analyst to improve the accuracy of the tests, by pooling the results of several of them together, using the records of past patients. However, understandably, not all the patients received all the tests.

- The UK Met Office has more than 200 weather measurement stations, as well as accepting voluntary weather data submissions (*Met Office Weather Stations*). Invariably, some of the sensors in the stations eventually fail until they are replaced. Additionally, the specific types of measurements collected in private submissions may be different for each volunteer, and for the same volunteer on each subsequent submission.

- A municipal library is trying to understand who uses their facilities and how. They make a very detailed survey asking questions about level of satisfaction, frequency of facility usage, and demographics. Of the users that take the survey, many do not finish the survey or skip questions.

Each of these cases can be seen as a set of multi-dimensional data points. In the hospital's case, each point represents a patient, each dimension containing information about one test, or whether or not they had a certain disease. In the weather stations' case, each point is a time instant and contains temperature, humidity, pressure, *et cetera*. In the library survey, each point is a respondent, the dimensions their answers.

These examples have another trait in common. They contain *missing data*: in some points, the values of some of the dimensions are unknown.

This poses a problem to most machine learning methods and statistical tests, since they require fully observed data. One option is to discard all points that have at least one missing value. However, there is still much useful information in the observed values of those points. How can we use this information?

The traditional answer lies in *imputation* (Little and Rubin, 2002). Imputation consists in replacing each of the missing values in the data set by values that are concordant with the other values of the point, and the rest of the data set. Then, the data set is analysed with a standard algorithm that needs fully-observed data.

For example, consider a data set of pine trees, listing their ages and heights. If the height of one pine tree is missing, a plausible replacement is the average height of pine trees that are the same age.

However, most likely the imputed tree does not have *exactly* the mean of the other trees' heights. If further analysis on the imputed data takes the pine's height as exact, the conclusions extracted from the data set will be biased and likely incorrect. A solution, proposed by Rubin (2004), is *multiple imputation*. Multiple imputation consists in creating $m > 1$ imputed data sets. For each data set, the missing values are replaced by a sample from the probability distribution of that missing value, given the original data set. Larger values of m imply less biased results. Commonly $m \leq 10$ is used.

In this thesis we go one step further, and set $m \to \infty$. That is: we aim to estimate a closed-form probability distribution, that approximates as well as possible the true distribution of the missing values given the data set. We call this kind of imputation *probability density imputation*, or just *density imputation*. The closed-form family of distributions we consider is that of mixtures of Gaussians.

Density imputation has several potential advantages:

- In some machine learning methods, it is possible to analytically include the estimated closed-form distribution in computations that ordinarily use fully observed variables. In essence these methods can learn directly from the $m \to \infty$ imputations.
  - Linear models, in which the output is a linear transformation of the input (at least for the popular MSE cost function, see section 2.1.3).
  - Probabilistic models (section 2.2). There, the distribution of the missing data can be seen as one or several latent (that is, unobserved) variables with a known distribution.

  - Kernel machine learning methods use a *kernel function* to determine the similarity between two points. For partially observed points, we can compute the expected value of the kernel instead (Nebot-Troyano and Belanche-Muñoz, 2010).

- Many machine learning algorithms define a parametrised family of functions. Then, they optimise those parameters to minimise the loss function (section 2.1.3). Some of those (notably, nearly all modern neural networks) use *stochastic gradient descent* (SGD) to perform optimisation. SGD is based on incrementally changing the values of the parameters towards stochastic samples of the negative gradient of the loss.

  Here, the estimated distribution can be used to sample new values for the missing variables at each gradient descent step. The effective number of imputations in this case would be finite, but growing linearly with the computation time spent in SGD.

  Furthermore, neural networks are famous for needing a large amount of data to be useful (otherwise they over-fit, section 2.1.3). Dropout, a popular technique that alleviates the over-fitting problem, also consists in adding stochastic noise to the network (Srivastava et al., 2014) . Thus, density imputation may address the over-fitting and the biased imputation problems simultaneously.

For situations that do not benefit from density imputation, we can recover the multiple imputation case simply by drawing m samples from the estimated distribution for each data point.

The idea of using a closed-form distribution family to approximate the true distribution of missing values is not strictly new. The aforementioned Nebot-Troyano and Belanche-Muñoz (2010) use univariate Gaussians centred on each observed value, ignoring correlations between variables in the same point, to compute the expected kernel and perform Support Vector Machine classification. Damianou, Titsias, and Lawrence (2016) estimate a univariate Gaussian for each *missing* value instead, based on the observed values in the same point. However, their method needs to be bootstrapped with a part of the data set that has no missing values, thus becoming biased. Furthermore, they do not propose propagating the closed-form distribution to further analysis.

From another angle, Gaussian Mixture Models have also been used for multiple imputation. Zio, Guarnera, and Luzi (2007) provided a maximum likelihood estimation algorithm for finite mixtures of Gaussians from an incomplete data set, and used it to perform multiple imputation.

1.1 Goals and contributions

The goals of this project are:

- To create a density imputation algorithm that takes into account correlations between variables in the same point. We restrict our attention to data sets where each data point is assumed to be *independently identically distributed* (i.i.d.). This implies, for example, no temporal correlation between the data points, so our method would not be suited to analyse the Met Office's data.
- To improve upon the state of the art of single and multiple imputation methods.

Our contributions are:

- A Bayesian inference algorithm for estimating the probability density in the presence of missing values, that accounts for relationships between variables. This is the Partially Observed Infinite Gaussian Mixture Model (POIGMM).
- An imputation algorithm based on expected conditional distributions of the POIGMM.

1.2 Intuition behind the POIGMM for imputation

Suppose we have an i.i.d. dataset $\mathbf{X}$, where each point $\mathbf{x}_n$ belongs to $\mathbb{R}^2$. Suppose we know the probability distribution that the points are drawn from, for example the one on the left-hand-side of Figure 1.1. Then, we obtain a point $\mathbf{x}_*$ with a missing value: in this case, the y coordinate is unknown. The vertical red line represents the possible values of the point. Then the imputation probability distribution, $p(\mathbf{x}_*^{\text{mis}} \mid \mathbf{x}_*^{\text{obs}}, \mathbf{X})$, is the "slice" of the original probability distribution that falls on the line.

Thus, a way to obtain a density imputation algorithm is to:

1. Estimate the probability distribution of the data, a *density model*, from a data set with missing values.
2. Compute conditional distributions on that model, given a partially observed data point.

GMMs are widely held to be excellent for density estimation. They are also used for clustering, which is finding separated subsets of the data set that have similar values. In recent years, significantly better probabilistic clustering models have emerged (Iwata,

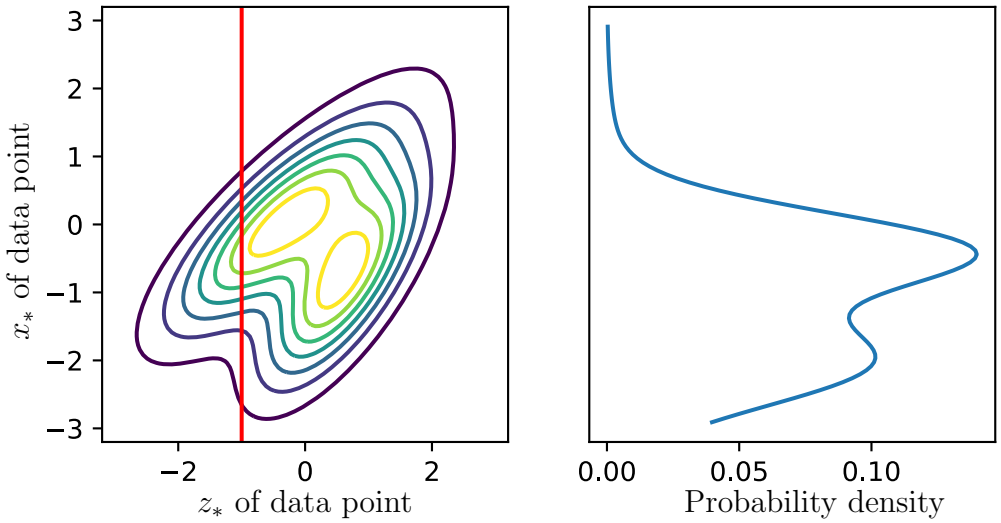

Figure 1.1: Left: the data density, and a point with missing y value in red. Right: the correct $p(\mathbf{x}_*^{\text{mis}} \mid \mathbf{x}_*^{\text{obs}}, \mathbf{X})$.

Duvenaud, and Ghahramani, 2012; Dilokthanakul et al., 2016; Johnson et al., 2016). However, for none of these models the conditional distribution of a partially observed point is tractable to compute. Furthermore, their performance at density estimation is not much better than that of GMMs.

1.3 Document structure

Chapter 2 explains concepts that the main contributions of the thesis rely on. Chapter 3 contains the derivation of the density estimation and imputation algorithms of the POIGMM. Chapter 4 is an account of the performed experiments and their results. Chapter 5 reviews the related state of the art in imputation. Finally, Chapter 6 summarises the contributions, explains things that could have been done better and shows directions for future work.

Chapter 2

Background

This chapter contains concepts necessary for understanding, and following the derivation of, the contributions of this thesis. Section 2.1 briefly reviews notational conventions, basic probability theory, and supervised machine learning. Section 2.2 reviews general probabilistic models, inference, and Bayesian networks. Section 2.3 contains some probability distributions and theorems that are required to develop the imputation algorithm in Chapter 3. Finally, section 2.4 explains the mechanisms by which missing data appears, the reason it is a problem, and several classes of strategies for handling it.

2.1 Preliminaries

This section gives informal definitions of basic probability concepts, to ensure the reader uses the same terminology. The section also establishes notational conventions, and gives a short account of supervised machine learning (ML). This work is not strictly about supervised learning. However it uses terms from that area, and some of the related work uses ML algorithms.

2.1.1 Notational conventions

Boldface lowercase ($\mathbf{x}, \mathbf{z}$) and uppercase ($\mathbf{X}, \mathbf{Z}$) letters denote vectors and matrices, respectively. The ith row of $\mathbf{X}$, as a column vector, is $\mathbf{x}_i$, and the jth element of $\mathbf{x}_i$ is $x_{i,j}$. The jth column of $\mathbf{X}$ is $\mathbf{X}_{:,j}$. Indices may also be sets rather than scalars, for example if mis_n is the set of indices of the missing dimensions of $\mathbf{x}_n$; then $\mathbf{x}_n^{\text{mis}_n}$, abbreviated as $\mathbf{x}_n^{\text{mis}}$, is the vector containing the missing dimensions of $\mathbf{x}$. When applying two subsets of indices to a

matrix dimensions, such as $\mathbf{M}^{\text{mis}_n,\text{obs}_n}$, the resulting matrix contains elements $m_{i,j}$ such that $i \in \text{mis}_n$ and $j \in \text{obs}_n$. Figure 2.3 is a visualisation of this.

Some special functions, such as the Gamma function (Γ) or the Dirac delta (δ), are used throughout the document. Their definitions are compiled in section B.2. They are also named in the text whenever they appear in an expression.

2.1.2 Basic probability theory

Let x be a random variable (RV). We write $x \sim p(x \mid \theta)$ or $x \sim p(\theta)$ to denote that, given parameters θ, x is distributed according to probability (or density) function $p(x \mid \theta)$. Probability densities (for continuous RVs) and probabilities (for discrete RVs, or regions of continuous RVs) are not distinguished throughout this work, we write both as $p(\cdot)$ or $q(\cdot)$.

Let $\mathcal{X}$ and $\mathcal{Y}$ be sets, and $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ be random variables. Then $p(x, y)$ is said to be a *joint probability distribution* if it is a probability distribution over $\mathcal{X} \times \mathcal{Y}$.

We write $p(x \mid y)$ to denote the *conditional* probability distribution, of x given y.

The *fundamental rules of probability* are (Bishop, 2006):

$$\begin{aligned} \text{Sum rule: } p(x) &= \sum_{y \in \mathcal{Y}} p(x, y) \qquad \text{or} \qquad p(x) = \int_{\mathcal{Y}} p(x, y) dy \\ \text{Product rule: } p(x, y) &= p(x \mid y) p(y) \end{aligned} \tag{2.1}$$

The probability distribution $p(x) = \int_{\mathcal{Y}} p(x, y) dy$ is the *marginal* distribution of x, with respect to $p(x, y)$.

From these rules, *Bayes' rule* can be derived (several forms shown):

$$p(y \mid x) = \frac{p(x \mid y) p(y)}{p(x)} = \frac{p(x \mid y) p(y)}{\int_{\mathcal{Y}} p(x, y) dy} = \frac{p(x, y)}{p(x)} \tag{2.2}$$

2.1.3 Supervised machine learning

Let $\mathbf{X}$ be a set of N points, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. Each $\mathbf{x}_n$ belongs to the *input space* $\mathcal{X}$, and has an associated label $y_n \in \mathcal{Y}$.

Assume the data points $\mathbf{X}$ are drawn independently from a probability distribution $\mathcal{D}$ (i.i.d. assumption). Assume there exists a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$, such that for all $n = 1, \ldots, N$, $f(\mathbf{x}_i) = y_i$. Then, the *supervised learning problem* is to estimate a model. That is: from a

given family of functions $\mathcal{G}$, specific to the algorithm, find $g \in \mathcal{G}$ such that $g(\mathbf{x})$ is "close" to $f(\mathbf{x})$ under the distribution $\mathcal{D}$.

Usually, $\mathcal{X}$ is the Cartesian product of several *feature* sets, $\mathcal{F}_1, \ldots, \mathcal{F}_D$. Each of the features $\mathcal{F}_i$ and $\mathcal{Y}$ are often one of the following sets:

- The set of categories $\mathcal{C} = \{0, 1, \ldots, M\}$, for some M. When $\mathcal{Y} = \mathcal{C}$, the supervised learning problem is said to be *multiclass classification* or just classification. When $\mathcal{F}_i = \mathcal{C}$, it is a *categorical feature*.
- The set of real numbers $\mathbb{R}$. When $\mathcal{Y} = \mathbb{R}$, supervised learning is called *regression*. When $\mathcal{F}_i = \mathbb{R}$, it is a *numerical feature*.

Alternative names for features are *variables* or *dimensions*; the latter being especially common when talking about numerical features.

The notion of "closeness" is formalised with a *loss* or cost function $\mathcal{L} : \mathcal{X}^N \times \mathcal{Y}^N \times \mathcal{G} \mapsto \mathbb{R}$. Intuitively, the loss function takes a data set (points and their labels) and the model, and outputs how well the model fits the data set. Popular loss functions are:

- *MSE*: suitable for regression. Given a data set of points $x_1, \ldots, x_N$ and labels $y_1, \ldots, y_N \in \mathbb{R}$, it is defined:

$$\mathcal{L}_{\text{MSE}}(\mathbf{X}, \mathbf{y}; g) = \frac{1}{N} \sum_{n=1}^{N} (y_n - g(x_n))^2 \tag{2.3}$$

- *Cross-entropy*: suitable for classification. When $y_n \in \{1, \ldots, M\}$, for all m set: $y_{n,m} = 1$ if $y_n = m$ and 0 otherwise. Then:

$$\mathcal{L}_{\text{H}}(\mathbf{X}, \mathbf{y}; g) = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} y_{n,m} \log y_{n,m} \tag{2.4}$$

Machine learning algorithms always restrict their search of a model g to a family of functions. Most commonly this family is characterised by *parameters* θ, and the model search problem reduces to a parameter search, or *optimisation*, problem.

Often, an algorithm requires the specification of *hyperparameters* λ. In this case, the family of functions taken into account is characterised by *both* θ and λ. However, the hyperparameters are not included in the "main" optimisation.

Most commonly, a data set is partitioned in three: the *training* set for optimising the parameters, the *validation* set for performing hyperparameter search, and the *test* set to

measure the algorithm's performance. The reason mutually exclusive sets are used for these tasks is that some algorithms will *over-fit*: they will find a model g that minimises the loss function very well in the training data points; so much so, that they assume idiosyncratic patterns of the *sample* $\mathbf{X}$ hold for the entire distribution $\mathcal{D}$.

Usually, the larger the family of possible models $\mathcal{G}$ is (that is, the larger the number of parameters θ to optimise), the more likely the algorithm is to over-fit.

2.2 Probabilistic Models

Consider a data set $\mathbf{X}$ of n points. As in the supervised learning setting, we assume they all come from the same distribution $\mathcal{D}$, which is unknown (i.i.d. assumption). In this case, however, we wish to estimate the distribution $\mathcal{D}$.

One approach to this is to define a *probabilistic model*, that is, a parametrised family of probability distributions. We denote our parameters as θ, and the family is defined by $\mathcal{P} = \{p(\mathbf{x} \mid \theta) : \theta \in \Theta\}$. We shall leave the functional form of $p(x \mid \theta)$ unspecified for now.

We now want to estimate the probability distribution from $\mathcal{P}$ that is *closest* to $\mathcal{D}$, for some metric of closeness. As in supervised learning section 2.1.3, this reduces to optimising the parameters θ.

2.2.1 Maximum likelihood estimation

Because of the i.i.d. assumption, the probability of the data set $\mathbf{X}$ being drawn from a distribution with parameters θ can be factored into points. This quantity is known as the *likelihood*:

$$p(\mathbf{X} \mid \theta) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \theta) \tag{2.5}$$

From here, the first inference method, known as *Maximum Likelihood Estimation (MLE)*, is straightforward. We wish to find the parameters θ under which the probability of obtaining the data set $\mathbf{X}$ is the highest. This is the optimisation problem:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta \in \Theta} p(\mathbf{X} \mid \theta) \tag{2.6}$$

In actual computer hardware, if the data set is not small, the calculation of the above expression is likely to *underflow* (Bishop, 2006). The minimum positive value that 64-bit

IEEE 754 floating point can represent is about $2.225 \cdot 10^{-308}$.[1] Thus: if the points have a geometric mean likelihood of 0.1 (very likely, especially at the beginning of optimisation), and there are more than 308 points (overwhelmingly likely), calculating the likelihood will output 0.

Thus it is more convenient to maximise the logarithm of the likelihood, known as the "log-likelihood". This increases possible representable values in the interval $[0, 1]$, extending the minimum positive floating-point value to $e^{-10^{308}}$:

$$\mathcal{L}(\theta) = \log p(\mathbf{X} \mid \theta) = \sum_{i=1}^{N} \log p(\mathbf{x}_i \mid \theta) \tag{2.7}$$

Since the logarithm is a monotonically increasing function, maximising the log-likelihood is equivalent to maximising the likelihood. The most popular losses (MSE, cross-entropy; see section 2.1.3) used in supervised learning algorithms are (up to sign negation) log-likelihood functions.

Once the parameters have been estimated, we can compute the probability of a test point $p(\mathbf{x}_*)$ by evaluating $p(\mathbf{x}_* \mid \theta)$.

The MLE approach is simple, but it is unfortunately also prone to over-fitting. Suppose we are performing inference for a model of throws of a six-sided die, to measure how biased it is. The parameters are 6 real numbers, corresponding to the unnormalised probability of numbers 1–6. The data set consists of 12 samples, of which none is the number 4. This is not an unlikely occurrence: assuming a fair die, it happens $(5/6)^{12} \simeq 0.11$ of the time. However, a MLE inference algorithm would give probability 0 to the number 4 coming up in the future. How can we prevent our algorithms from making such a rash judgement?

2.2.2 Bayesian inference

One answer lies in Bayesian inference. This class of algorithms expresses its knowledge about the parameters as a probability distribution, rather than a single value. This can prevent making unjustifiably drastic judgements from little data. Because the parameters θ always remain a random variable rather than a value, these methods are often called *Bayesian nonparametrics*.

Bayesian inference starts by giving a probability distribution to the parameters, $p(\theta)$, known as the *prior distribution* or just prior. Then, it calculates the *posterior distribution*

[1] Value obtained by running command: `python -c "import sys; print(sys.float_info.min)"`.

of the parameters given the data, $p(\theta \mid \mathbf{X})$. Using Bayes' rule (equation 2.2):

$$p(\theta \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \theta)p(\theta)}{p(\mathbf{X})} = \frac{p(\mathbf{X} \mid \theta)p(\theta)}{\int_\Theta p(\mathbf{X} \mid \theta)p(\theta)d\theta} \tag{2.8}$$

The denominator $p(\mathbf{X})$ is independent of θ, and is often referred to as a normalisation constant.

Once the posterior is calculated, the algorithm can answer queries about the probability (or probability density) of a test point $\mathbf{x}_*$, taking into account the uncertainty about the parameters:

$$p(\mathbf{x}_* \mid \mathbf{X}) = \int_\Theta p(\mathbf{x}_* \mid \theta)p(\theta \mid \mathbf{X})d\theta \tag{2.9}$$

These inference and test procedures are jointly known as *fully Bayesian inference*.

The prior distribution is a way for an expert analyst to incorporate their knowledge about the application domain into the algorithm. This is sometimes criticised, because ideally machine learning methods would learn everything from the data. The critique is addressed somewhat by using *noninformative priors*, that encode little assumptions. In our dice example, this would be for instance a uniform distribution over the 5-dimensional parameter space. These help *regularise* the resulting model preventing overly rash judgements, and make it possible to have uncertainty about the parameters, while minimising required user intervention.

Another criticism of Bayesian inference is that the prior is often chosen not because it matches the analyst's beliefs, but rather because it is mathematically convenient (Bishop, 2006). See section 2.2.6 about conjugate families.

Maximum *a posteriori* inference

In some cases, it is intractable to calculate the integrals in Equations (2.8) or (2.9). However the regularisation or domain knowlege that a prior can provide is still desirable. In that case, a Maximum *a posteriori* (MAP) estimate of the parameters can be performed:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta \in \Theta} \frac{p(\mathbf{X} \mid \theta)p(\theta)}{p(\mathbf{X})} = \arg\max_{\theta \in \Theta} p(\mathbf{X} \mid \theta)p(\theta)$$

since the denominator does not depend on θ.

2.2.3 Bayesian networks

Consider a joint distribution over variables and its decomposition into conditional probabilities:

$$p(w, x, y, z) = p(w \mid x, y, z)p(x \mid y, z)p(y \mid z)p(z)$$

By the product rule, this holds for any joint distribution.

We say that a random variable a is *conditionally independent* (CI) from RVs $B = b_1, \ldots, b_N$ and RVs $C = c_1, \ldots, c_N$, written $(a \perp B \mid C)$, if $p(a \mid B, C) = p(a \mid C)$.

If we make some conditional independence assumptions, we can simplify the above distribution. For example, if we assume $(w \perp x, y \mid z)$, it follows that $p(w \mid x, y, z) = p(w \mid z)$ and the distribution above can be written as:

$$p(w, x, y, z) = p(w \mid z)p(x \mid y, z)p(y \mid z)p(z)$$

A Bayesian Network (BN) is a joint distribution over several variables, that imposes some CI assumptions. The CI assumptions are characterized by a directed acyclic graph. Each variable corresponds to a node, and needs only be conditioned on its preceding nodes. For example, the distribution above corresponds to the left-hand side on Figure 2.1.

We introduce two more concepts.

- *Plate notation*: the rectangles with quantities on the right-hand side of Figure 2.1 are an example. They represent several repetitions of the same structure, conditionally independent between them. Graph edges between elements of the same plate represent a 1-to-1 relationship between each of them, whereas edges that cross the plate boundary are many-to-1 or 1-to-many relationships.
- *Latent and observed variables*: variables that are grey represent observed variables, that we know the value of. White variables are latent: we do not know their value, only their probability distribution. Conditioning on the value of an observed variable changes the distribution of latent variables in the graph.
- *Parameters*: quantities without a circle around them are parameters of the overall joint distribution.

$$p(v, \mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z} \mid \theta) = p(v, \mathbf{y}) \prod_{n=1}^{N} p(z_n)p(y_n \mid z_n, \theta) \prod_{m=1}^{M} p(w_m \mid \mathbf{z})p(x_m \mid \mathbf{z}, \mathbf{y}) \tag{2.10}$$

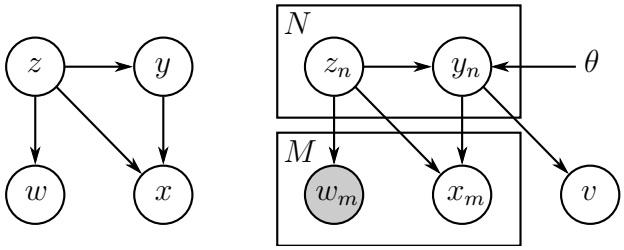

Figure 2.1: Left: Graphical representation of $p(w \mid z)p(x \mid y, z)p(y \mid z)p(z)$. Right: Representation of equation (2.10).

2.2.4 Mixture model

Let $\mathbf{z}$ be a discrete random variable, which is a K-dimensional *one-hot* vector. A one-hot vector has value 1 in one dimension and 0 in all the others. The RV $\mathbf{z}$ follows a multinomial distribution (equation B.7), with probabilities $\mathbf{w} = w_1, \ldots, w_K$.

Given K probability distributions $p_1(\mathbf{x} \mid \theta_1), \ldots, p_K(\mathbf{x} \mid \theta_K)$, a mixture model is defined as:

$$p(\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} p(\mathbf{z}_k = 1) p_k(\mathbf{x} \mid \theta_k) = \sum_{k=1}^{K} w_k p_k(\mathbf{x} \mid \theta_k) \tag{2.11}$$

which corresponds to the graphical model in Figure 2.2.

Each individual p_k is known as a *component*. The latent variable $\mathbf{z}$ controls which component a sample $\mathbf{x}$ is drawn from.

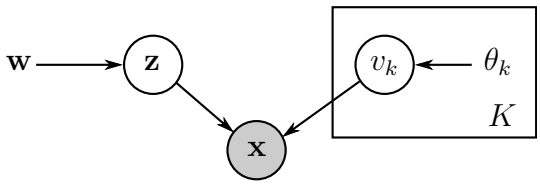

Figure 2.2: Graphical representation of a mixture model.

A commonly used mixture model is the *Gaussian mixture model* (GMM). It consists in having all the components p_k be a Gaussian distribution. The probability density induced by a GMM is also known as a Mixture of Gaussians (MoG).

2.2.5 Variational inference

Suppose we have a probabilistic model p, with latent variables $\mathbf{Z} = \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M$, and observed variables $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N$. In many interesting cases, computing the exact

posterior $p(\mathbf{Z} \mid \mathbf{X})$ is intractable.

A solution to this is *variational inference* (VI). It consists in defining a family of distributions $q_{\boldsymbol{\phi}}(\mathbf{Z})$, parametrised by $\boldsymbol{\phi}$. The family q is often called the *variational distribution*.

The Kullback Leibler (KL) divergence is a measure of the difference between two probability distributions, $q(y)$ and $p(y)$. Its definition is:

$$D_{\mathrm{KL}}(q||p) = \mathop{\mathbb{E}}_{y\sim q}[\log q(y) - \log p(y)] \tag{2.12}$$

In variational inference, we seek to minimise the KL divergence between $q(\mathbf{Z})$ and $p(\mathbf{Z} \mid \mathbf{X})$. Thus Bayesian inference has been reduced to an optimisation problem, that can be solved with standard methods like block coordinate descent or gradient descent. Gradient descent consists in computing the negative gradient of the objective and then adjusting the parameters towards it. Block coordinate descent is explained in Algorithm 1, and is always guaranteed to converge to a local minimum (Grippo and Sciandrone, 2000).

Algorithm 1 Block coordinate descent

Input: a function to minimise, $f(\boldsymbol{\phi})$. A tolerance parameter ϵ.
Initialise $\boldsymbol{\phi}$. Partition $\boldsymbol{\phi}$ into blocks $\phi_1, \phi_1, \ldots, \phi_M$.
Initialise the value of the last iteration, $v \leftarrow \infty$.
while $f(\boldsymbol{\phi}) - v > \epsilon$, that is, $f(\boldsymbol{\phi})$ hasn't converged to a minimum **do**
  $v \leftarrow f(\boldsymbol{\phi})$.
  for each block $i = 1, \ldots, B$ **do**
    Analytically set $\phi_i \leftarrow \arg\max_{\phi_i} f(\phi_1, \ldots, \phi_i, \ldots, \phi_B)$.
  end for
end while
Output $\boldsymbol{\phi}$.

The KL divergence between $q_{\boldsymbol{\phi}}$ and the true posterior is:

$$D_{\mathrm{KL}}(q||p) = \mathop{\mathbb{E}}_{\mathbf{Z}\sim q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}(\mathbf{Z}) - \log \frac{p(\mathbf{X}, \mathbf{Z})}{p(\mathbf{X})}\right] = \mathop{\mathbb{E}}_{\mathbf{Z}\sim q_{\boldsymbol{\phi}}}[\log q_{\boldsymbol{\phi}}(\mathbf{Z}) - \log p(\mathbf{X}, \mathbf{Z})] + \log p(\mathbf{X})$$

Since $\log p(\mathbf{X})$ does not depend on $\boldsymbol{\phi}$, we can ignore it during our optimisation. In fact, since the KL divergence is never negative, we have:

$$\log p(\mathbf{X}) \geq \mathop{\mathbb{E}}_{q_{\boldsymbol{\phi}}}[\log p(\mathbf{X}, \mathbf{Z})] - \mathop{\mathbb{E}}_{q_{\boldsymbol{\phi}}}[\log q_{\boldsymbol{\phi}}(\mathbf{Z})] \tag{2.13}$$

This is known as the *evidence lower bound* (ELBO), and maximising it is equivalent to minimising the KL divergence. The gap between the ELBO and $\log p(\mathbf{X})$ is the KL divergence between $q_{\boldsymbol{\phi}}$ and p.

Usually, q is chosen such that it *factorises*. That is, the estimated distribution of each latent variable is independent of all the others:

$$q(\mathbf{Z}) = \prod_{i=1}^{M} q_{\phi_i}(\mathbf{z}_i)$$

This makes it possible to compute optimal updates for each ϕ_i analytically, and run block coordinate descent. According to Bishop (2006), at each step, the optimal q_{ϕ_i} is:

$$\log q^*_{\phi_i}(\mathbf{z}_i) = \mathop{\mathbb{E}}_{j \neq i}[\log p(\mathbf{X}, \mathbf{Z})] + \mathcal{C} \tag{2.14}$$

where $\mathcal{C}$ is the normalising constant, which we can recover by inspection when deriving updates. If the probability distribution of the right-hand side is not in the family of q_{ϕ_i}, it is sufficient to minimise the KL-divergence between q_{ϕ_i} and $\mathbb{E}_{j \neq i}[\log p(\mathbf{X}, \mathbf{Z})]$.

2.2.6 Exponential and conjugate families of distributions

Let $\mathbf{x}$ be a random variable, and $\boldsymbol{\theta}$ be some parameters. An *exponential family* is a family of probability distributions over $\mathbf{x}$ that can be written as (Bishop, 2006):

$$p(\mathbf{x} \,|\, \boldsymbol{\theta}) = a(\mathbf{x}) \exp(\mathbf{f}(\boldsymbol{\theta})^\top \mathbf{g}(\mathbf{x}) - h(\boldsymbol{\theta})) \tag{2.15}$$

for some fixed vector-valued functions $\mathbf{f}$ and $\mathbf{g}$, and scalar-valued functions a and h. The function $\mathbf{f}$ must have the same number of outputs as $\boldsymbol{\theta}$.

Essentially, in an exponential family the parameters and the RV over which the density is defined can only interact via the dot product $\mathbf{f}(\boldsymbol{\theta})^\top \mathbf{g}(\mathbf{x})$. The transformed parameters $\mathbf{f}(\boldsymbol{\theta})$ are often called the *natural parameters*, and the transformed variables $\mathbf{g}(\mathbf{x})$ the *sufficient statistics*.

Exponential families have several interesting properties. An important one is that they have a *conjugate family*. Let $\mathcal{P}$, and $\mathcal{Q}$ be parametrised families of distributions over domain $\mathbf{x}$. Then $\mathcal{Q}$ is conjugate to $\mathcal{P}$ if, for any $p \in \mathcal{P}$ and $q \in \mathcal{Q}$, $p(\mathbf{x})q(\mathbf{x}) \in \mathcal{Q}$.

That is, if we multiply a distribution in $\mathcal{P}$ with one belonging to its conjugate family $\mathcal{Q}$, the result is also in the conjugate family. This simplifies Bayesian analysis: if the prior is

conjugate to the likelihood, then the posterior belongs to the same family as the prior.

2.3 Important probability distributions

This section contains the definitions and some less well-known observations about probability distributions, which will be used in Chapter 3. Definitions of other used distributions can be found in section B.1.

Each of the two distributions presented in this section is an exponential family.

2.3.1 Multivariate Gaussian

Let $\boldsymbol{\mu} \in \mathbb{R}^D$ be a D-dimensional vector representing the mean, and $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$ the *covariance* matrix, which must be positive semi-definite (section B.3). Let $\mathbf{x}$ be a RV belonging to $\mathbb{R}^D$. Then, $\mathbf{x}$ is Gaussian distributed if (Bishop, 2006):

$$\mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.16}$$

The quantity $\Delta^2 = (\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ is known as the squared *Mahalanobis distance*. As an intuition, consider an uncountably infinite series of D-dimensional ellipsoids with the same shape, and increasing "radius", all centered on the mean $\boldsymbol{\mu}$. The Mahalanobis distance is of a point from the centre is the "number" of ellipsoids between them. The Gaussian density decreases exponentially with the Mahalanobis distance.

An important property of the multivariate Gaussian is that any subset of its variables is also Gaussian distributed, taking the appropriate elements of the covariance matrix. Let obs $\subseteq \{1, \ldots, D\}$ denote a subset of the elements of $\mathbf{x}$, a RV. If $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then $\mathbf{x}^{\text{obs}} \sim \mathcal{N}(\boldsymbol{\mu}^{\text{obs}}, \boldsymbol{\Sigma}^{\text{obs,obs}})$. (Bishop, 2006)

Sometimes it is easier to manipulate the *precision* matrix ($\boldsymbol{\Lambda}$) of a Gaussian, $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$. The marginal distribution property above can be restated:

$$\text{If } \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}), \text{ then } \mathbf{x}^{\text{obs}} \sim \mathcal{N}\left(\boldsymbol{\mu}^{\text{obs}}, \left(\left(\boldsymbol{\Lambda}^{-1}\right)^{\text{obs,obs}}\right)^{-1}\right) \tag{2.17}$$

Finally, an intuitive explanation of the $\cdot^{\text{obs,obs}}$ style indexing in the covariance matrix, and why it can be seen as a partition. Permuting the elements of a Gaussian RV yields a gaussian RV, where the same permutation has been applied to the elements of $\boldsymbol{\mu}$, and

to the rows *and* columns of $\mathbf{\Sigma}$. Given a set of indices of observed variables (obs), we can apply a permutation such that the observed variables are contiguous. The resulting covariance matrix can then be partitioned into blocks, as in Figure 2.3.

The same indexing-as-partition view holds for the precision $\mathbf{\Lambda}$, since it is also symmetric. However, in that case the square matrices resulting from the partition do *not* describe the marginal distributions of the original Gaussian.

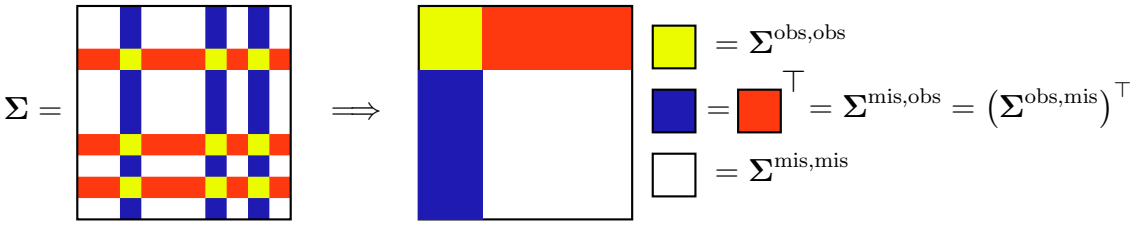

Figure 2.3: Partitioning a symmetric matrix into missing and observed variables.

2.3.2 Wishart

The Wishart distribution is a distribution over positive semi-definite matrices. In Bayesian analysis, it is usually the distribution of the precision parameter of a Gaussian distribution. Let $\mathbf{\Lambda}$ be a $D \times D$ random matrix, $\mathbf{W} \in \mathbb{R}^{D \times D}$ be the covariance parameter and $\nu \in \mathbb{R}$ the degrees of fredom. Then $\mathbf{\Lambda}$ is Wishart distributed if (Bishop, 2006):

$$p(\mathbf{\Lambda} \,|\, \mathbf{W}, \nu) = K_{\mathbf{W},\nu} |\mathbf{\Lambda}|^{(\nu - D - 1)/2} \exp\left(-\frac{1}{2}\text{tr}\left(\mathbf{W}^{-1}\mathbf{\Lambda}\right)\right) \tag{2.18}$$

With $K_{\mathbf{W},\nu}$ the normalisation constant:

$$K_{\mathbf{W},\nu} = |\mathbf{W}|^{-\nu/2} \left(2^{\nu D/2} \pi^{D(D-1)/4} \prod_{i=1}^{D} \Gamma\left(\frac{\nu + 1 - i}{2}\right)\right)^{-1} \tag{2.19}$$

The following expectations of the Wishart distribution are well-known and frequently used (Bishop, 2006, Appendix B):

$$\mathop{\mathbb{E}}_{\mathcal{W}(\mathbf{\Lambda} \,|\, \mathbf{W}, \nu)} [\log |\mathbf{\Lambda}|] = \log |\mathbf{W}| + \sum_{i=1}^{D} \Psi\left(\frac{\nu + 1 - i}{2}\right) + D \log 2 \tag{2.20}$$

$$\mathop{\mathbb{E}}_{\mathcal{W}(\mathbf{\Lambda} \,|\, \mathbf{W}, \nu)} [\mathbf{\Lambda}] = \nu \mathbf{W} \tag{2.21}$$

where Ψ is the Digamma function (equation (B.9)).

Suppose the precision of a Gaussian is Wishart-distributed. Then, what is the distribution of the precision of a marginal distribution of the Gaussian? That is, Equation (2.17). The following theorem will be useful in Chapter 3.

Theorem 2.3.1 (Precision of a marginal Gaussian is Wishart-distributed)**.** *Let W be a $D \times D$ positive semi-definite covariance matrix, and ν a real number, $\nu \geq D$. Let $\mathbf{\Lambda} \sim \mathcal{W}(\mathbf{\Sigma}, \nu)$ be a Wishart-distributed square matrix. Let* obs $\subseteq \{1, \ldots, D\}$ *be a set of indices of the columns/rows of $\mathbf{\Lambda}$. Let* mis $= \{1, \ldots, D\} \setminus$ obs*, that is, the complement of* obs*. Then:*

$$\left(\left(\mathbf{\Lambda}^{-1}\right)^{\text{obs,obs}}\right)^{-1} \sim \mathcal{W}(\mathbf{W}^{\text{obs,obs}}, \nu - D + |\text{obs}|)$$

.

Proof. Recall the view of double indexing as a partition (Figure 2.3). Using a well-known expression about the inverse of a partitioned matrix (Horn and Johnson, 2012, Section 0.7.1) we get:

$$\left(\mathbf{\Lambda}^{-1}\right)^{\text{obs,obs}} = \left(\mathbf{\Lambda}^{\text{obs,obs}} - \mathbf{\Lambda}^{\text{obs,mis}} \left(\mathbf{\Lambda}^{\text{mis,mis}}\right)^{-1} \mathbf{\Lambda}^{\text{mis,obs}}\right)^{-1}$$

Inverting both sides we obtain our expression of interest on the left:

$$\left(\left(\mathbf{\Lambda}^{-1}\right)^{\text{obs,obs}}\right)^{-1} = \mathbf{\Lambda}^{\text{obs,obs}} - \mathbf{\Lambda}^{\text{obs,mis}} \left(\mathbf{\Lambda}^{\text{mis,mis}}\right)^{-1} \mathbf{\Lambda}^{\text{mis,obs}}$$

According to (Muirhead, 1982, Theorem 3.2.10), if $\mathbf{\Lambda} \sim \mathcal{W}(\mathbf{\Sigma}, \nu)$, then an expression like the right-hand-side of the expression above is distributed according to $\mathcal{W}(\mathbf{\Sigma}, \nu - D + |\text{obs}|)$. This implies the theorem. □

2.4 Missing data

Suppose a data analyst has collected a set of points $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, of D features each. The points may or may not have associated labels $\mathbf{y} = y_1, \ldots, y_N$. The analyst then wishes to perform some task with the data, for example:

- Supervised learning (section 2.1.3).
- Clustering. Find a number K of clusters $c_1, \ldots, c_N$, $K \ll N$; and an assignment of points to those clusters, such that the points assigned to the same cluster are "similar" in some way. An example of a similarity measure is the Euclidean distance.
- Statistical tests. For example, a Student's t-test to estimate whether the mean of the

variable $x_i = \{\mathbf{x}_{n,i} : n = 1, \ldots, N\}$ is *significantly* different for points where $y_n = 1$ than for points where $y_n = 0$.

However, it could be that the analyst did not record a value for some points and variables of the data set $\mathbf{X}$, those values are *missing*. The input points $\mathbf{x}_n$ now belong to the set $\mathcal{X} \cup \{\bullet\}$. Many analysis procedures require inputs and outputs in $\{0, \ldots, M\}$ or $\mathbb{R}$. What can our analyst do?

2.4.1 Types of missing data: MCAR, MAR, NMAR

Little and Rubin (2002) distinguish three mechanisms that generate missing data. The distinction is important because it dictates which procedures can be used to handle the missing data: different methods are shown to work based on different mechanism assumptions. Let $m_{i,j}$ be a random variable that is 1 if $x_{i,j}$ is missing and 0 if it is observed. $\mathbf{M}$ is the matrix of all such variables, of the same size as $\mathbf{X}$. We denote as $\mathbf{M}_{:,\text{obs}}$ the set of all entries of $\mathbf{M}$ that have value 1, and $\mathbf{M}_{:,\text{mis}}$ the ones that have value 0. The same indices apply to $\mathbf{X}$.

- Missing completely at random (MCAR). Whether a variable is missing or not is *independent* of the value of any variable. That is, $p(\mathbf{M} \mid \mathbf{X}) = p(\mathbf{M})$. This assumption is almost always unrealistic, however, some methods that assume it (such as MissForest; Stekhoven and Bühlmann, 2011) perform well in other settings in practice.

- Missing at random (MAR). The probability distribution of missing values depends on the observed values only. That is, $p(\mathbf{M} \mid \mathbf{X}) = p(\mathbf{M} \mid \mathbf{X}_{\text{obs}})$. This assumption is often falsified as well, but it can be made closer to truth by measuring other variables that are correlated with $\mathbf{M}$.

- Not missing at random (NMAR). The missingness indicator $\mathbf{M}$ depends on the values that are missing themselves. In theory, it is impossible to reconstruct data that is NMAR from the data alone: for all an algorithm knows, the distribution of missing values is completely different from the one of observed values.

Note that MCAR implies MAR, and MAR implies NMAR. This is because the conditioning that each method allows to exist can be vacuous.

2.4.2 Handling missing data

What follows describes three overarching strategies for dealing with missing data (Little and Rubin, 1989), and some variations of them.

Weighting, complete-case analysis

The first and simplest way to handle data sets with missing data is to ignore the rows that have missing variables. This is known as *complete-case analysis*, and it easily yields a complete data set that the analyser can use standard methods on. This is only appropriate if the data is MCAR, otherwise the newly created complete data set will be biased. A more sophisticated version of this is *weighting*, which assigns numerical weights to the complete rows that remain, to make the removal as unbiased as possible.

The strength of this approach is its simplicity, but it has a number of problems. First, it is only possible to perform if there are some rows with complete data. Even when that is the case, it will reduce the data set size considerably, discarding a lot of useful information, thus reducing the significance of tests and the accuracy of supervised learning. Second, calculating correct statistical tests while taking into account the weights is difficult.

Imputation: single, multiple and "density"

Another approach to analysis with missing data is *imputation*. Imputation consists in filling the missing values in a data point, $\mathbf{x}_n^{\text{mis}}$, with *plausible* values, given the observed values of the data point $\mathbf{x}_n^{\text{obs}}$. Once all the missing values have been filled, the analyst can use standard methods.

A simple approach to imputation is to fill each variable's missing values with the mean (or mode, if it is categorical) of all observed values in the same variable. For a variable j, we replace values $\{x_{n,j} : j \in \text{mis}_n, \forall n\}$ with $\hat{x}_j = \text{mean}(\{x_{n,j} : j \in \text{obs}_n, \forall n\})$. However, assuming that all missing $x_{n,j} = \hat{x}_j$ is very unrealistic. Most obviously, $\hat{x}_j$ is a constant and thus uncorrelated with the observed variables $\mathbf{x}_n^{\text{obs}}$, which will bias statistical correlation tests downwards.

To impute correctly, a procedure needs to estimate the correlations between each of the variables. One popular way to do this is *regression imputation* (Little and Rubin, 2002; Raghunathan et al., 2001). It consists in estimating the missing values for a dimension j using a supervised learning algorithm, with the observed values of $\mathbf{X}_{:,j}$ as labels and the values of $\mathbf{X}_{:,i}$, $i \neq j$ as inputs. The current state of the art in imputation algorithms

(Van Buuren and Oudshoorn, 1999; Stekhoven and Bühlmann, 2011) are all variations of regression imputation that use off-the-shelf supervised learning algorithms. Algorithm 2 describes regression imputation in detail.

The most prominent drawback of imputation is that any analysis performed on imputed data will be unduly overconfident. Imputation algorithms cannot in general recover the exact value that would be observed, they will make stochastic errors; but the subsequent analysis will be unaware of this and take the data as if it was real. Rubin (2004) proposed Multiple Imputation as a solution to this problem. It consists in creating $m > 1$ imputed data sets. For each data set, and each missing value $x_{n,i}$ is replaced by a sample from the estimated distribution $p(x_{n,i} \mid \mathbf{X})$. The data sets are then analysed using complete-data techniques and the results combined.

In standard Multiple Imputation, the m samples that are used to impute $x_{i,j}$ are drawn from the conditional distribution $p(x_{n,i}|\mathbf{x}_{i,\text{obs}})$. This is the assumption Rubin (2004) uses to show that the inferences from the multiple imputed data set are valid. Rubin also argues that modest imputation sizes ($m \in [2, 10]$) are sufficient for analysing surveys.

This dissertation proposes *probability density imputation*, which attempts to estimate the conditional distribution $p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{X})$ in closed form, rather than just draw a few samples from it. We argued in the introduction that some supervised learning methods can benefit from that.

Algorithm 2 Regression imputation

Input: data set with missing data $\mathbf{X}$ of size $N \times D$, and its missingness indicator $\mathbf{M}$.
Optionally: impute $\mathbf{X}$ with a simple procedure, such as mean imputation.
for each $j \in 1, \ldots, D$ **do**
  $\text{mis}_{:,j} \leftarrow \{i \in 1, \ldots, N \mid m_{i,j} = 1\}$. ▷ The row indices where $\mathbf{X}_{:,j}$ is missing.
  $\text{obs}_{:,j} \leftarrow \{1, \ldots, D\} \setminus \text{mis}_{:,j}$. ▷ The row indices where $\mathbf{X}_{:,j}$ is observed.
  $\mathbf{y}'_{\text{obs}} \leftarrow \{x_{i,j} \mid i \in \text{obs}_{:,j}\}$. ▷ These labels are all observed.
  $\mathbf{X}'_{\text{obs}} \leftarrow \{x_{i,k} \mid i \in \text{obs}_{:,j} \wedge k \neq j\}$. ▷ Some of the values here may be unobserved.
  "Regression" step: Using a supervised learning algorithm,
    estimate the mapping g from $\mathbf{X}'_{\text{obs}}$ to $\mathbf{y}'_{\text{obs}}$.
  $\mathbf{X}'_{\text{mis}} \leftarrow \{x_{i,k} \mid i \in \text{mis}_{:,j} \wedge k \neq j\}$.
  Estimate the values $\mathbf{y}'_{\text{mis}} = g(\mathbf{X}'_{\text{mis}})$.
  Update $\mathbf{X}_{\text{mis},j}$ with the values of $\mathbf{y}'_{\text{mis}}$.
end for
Optionally: repeat the previous loop until $\mathbf{X}$ converges, or starts to diverge.
Output $\mathbf{X}$.

Model estimation from incomplete data

It is also possible to design a model that can be inferred and used in the presence of missing data. Arguably this is the most desirable approach, as it directly uses all the available information for the task the analyst cares about, which is modelling or prediction. However, in practice most algorithms that do this use an "internal" imputation step: the aforementioned work by Damianou, Titsias, and Lawrence (2016) first estimates an uncorrelated Gaussian distribution each missing value by using the fully observed part of the data set.

If a supervised learning model is robust enough to noise, it may be able to learn with the missing values replaced by special value (for example the mean), especially if $\mathbf{M}$ is also given as input to the method. This is the approach taken by Che et al. (2016) and Cinar et al. (2017) in the context of predicting the label of a time series with missing data; and by MissForest (Stekhoven and Bühlmann, 2011) and MICE (Van Buuren and Oudshoorn, 1999) when performing supervised learning for imputation (Chapter 5).

Chapter 3

Partially Observed Infinite GMM

This chapter presents an algorithm for fully Bayesian inference of an infinite Gaussian Mixture Model (GMM) from data that includes missing values. The algorithm defines a stochastic process prior on the number and density of clusters. Then, given data, it approximates the posterior distribution using a finite variational distribution.

The inferred variational distribution is chosen such that a reasonable approximation to the posterior distribution $p(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}}, \mathbf{X})$ is available analytically. Thus, this algorithm can perform Bayesian imputation of a MAR data set.

The algorithm extends the work by Blei and Jordan (2006), who give a variational inference algorithm for completely observed data, for an infinite mixture of exponential family distributions. In fact, the prior and variational distributions of our algorithm are the same as theirs, only the *likelihood* function is different. However, this makes one of their coordinate descent updates analytically intractable. To sidestep this, we approximate the marginal likelihood with the *maximum* likelihood of a partially observed point. This is not the same as performing maximum likelihood estimation (Section 2.2.1).

This chapter is structured as follows. Section 3.1 is an exposition of the probabilistic model by Blei and Jordan (2006), followed by some modifications made necessary by partially observed points. Section 3.2 contains the derivation of our inference algorithm, which heavily borrows from the finite Bayesian GMM derivation by Bishop (2006, Section 10.2). However, necessarily, the steps that relate to partially observed data are original. Finally, Section 3.3 describes the algorithm for imputation using an inferred variational distribution, and Algorithm 3 summarises the POIGMM.

3.1 The Dirichlet Process Mixture Model

3.1.1 The Dirichlet Process Prior

Let $\alpha \in \mathbb{R}_{\geq 0}$ be a nonnegative scaling parameter. Let G_0 be a probability distribution over continuous space Θ. Let $\mathbf{v} = v_1, v_2, \ldots$ be an infinite collection of random variables drawn independently from a Beta distribution, $v_k \sim \text{Beta}(1, \alpha)$. Similarly, let $\boldsymbol{\theta} = \theta_1, \theta_2, \ldots$ be an infinite collection of random variables drawn from the distribution G_0, $\theta_k \sim G_0$.

Definition 3.1.1 (Dirichlet Process)**.** *A random variable θ_* is distributed according to a Dirichlet Process (DP) if:*

$$p(\theta_* \,|\, \boldsymbol{\theta}, \mathbf{v}) = \sum_{k=1}^{\infty} \pi_k(\mathbf{v}) \delta(\theta_* - \theta_k) \tag{3.1}$$

$$\pi_k(\mathbf{v}) = v_k \prod_{j=1}^{k-1} (1 - v_j) \tag{3.2}$$

where $\delta(\cdot)$ is a Dirac delta (equation B.10). Note that π_k is not a probability distribution, only a function that outputs probabilities: it doesn't output the probability of its argument.

From (3.1) it is clear that the DP is a *discrete* stochastic process over the continuous space Θ, with countably infinite possible values. The possible values ($\boldsymbol{\theta}$) are drawn independently from G_0. The probability of drawing each θ_k from the DP is determined by the *stick-breaking process* $\pi_k(\mathbf{v})$. Intuitively, π_k starts with a unit length "stick". At each step it draws v_k from a Beta distribution, and discards a v_k fraction of what is left in the stick. The absolute length of the discarded fraction at step k is the probability of drawing θ_k. This stick-breaking representation of the DP was originally formulated by Sethuraman (1994).

Intuition behind the DP prior

The DP is ideally suited as a prior distribution for infinite mixture models. Let $p(\mathbf{x}_n \,|\, \theta_*)$ be a probability distribution over observations, parametrised by θ_*. Combining it with the DP prior:

$$p(\mathbf{x}_n \,|\, \boldsymbol{\theta}, \mathbf{v}) = \int_{\Theta} p(\mathbf{x}_n \,|\, \theta_*) p(\theta_* \,|\, \boldsymbol{\theta}, \mathbf{v}) d\theta_* = \int_{\Theta} p(\mathbf{x}_n \,|\, \theta_*) \sum_{k=1}^{\infty} \pi_k(\mathbf{v}) \delta(\theta_* - \theta_k) d\theta_*$$

Since all the quantities involved are nonnegative, and $\pi_k(\mathbf{v})$ does not depend on θ:

$$p(\mathbf{x}_n \mid \boldsymbol{\theta}, \mathbf{v}) = \sum_{k=1}^{\infty} \pi_k(\mathbf{v}) \int_{\Theta} p(\mathbf{x}_n \mid \theta_*)\delta(\theta_* - \theta_k)d\theta_* = \sum_{k=1}^{\infty} \pi_k(\mathbf{v}) p(\mathbf{x}_n \mid \theta_k) \tag{3.3}$$

It is clear that equation (3.3) is analogous to the expression of a mixture model (equation 2.11), with an infinite amount of components.

3.1.2 Joint likelihood of a DP mixture

Let $\mathbf{X} = N \times D$ be a data set of N points. For each point, we introduce a latent variable $\mathbf{z}_n$ that represents a mixture component assignment. If point $\mathbf{x}_n$ was drawn from component k, which has parameters θ_k, then $z_{n,k} = 1$ and $z_{n,j} = 0$ for $j \neq k$. The distribution of $\mathbf{z}_n$ is an (infinite) multinomial distribution (equation B.7) with parameters $\pi_k(\mathbf{v})$.

The joint likelihood over all observations, parameters and latent variables is illustrated on the left-hand-side of Figure 3.1, and is:

$$p(\mathbf{X}, \mathbf{Z}, \mathbf{v}, \boldsymbol{\theta} \mid \alpha, G_0) = \left[\prod_{n=1}^{N} p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\theta}) p(\mathbf{z}_n \mid \mathbf{v})\right] \prod_{k=1}^{\infty} p(v_k \mid \alpha) \prod_{k=1}^{\infty} p(\theta_k) \tag{3.4}$$

Which corresponds to the following process for generating the data $\mathbf{X}$:

1. For each component $k = 1, 2, \ldots$:
   (a) Draw its stick-breaking fraction: $v_k \mid \alpha \sim \text{Beta}(1, \alpha)$
   (b) Draw its parameters: $\theta_k \mid G_0 \sim G_0$
2. For each data point $n = 1, 2, \ldots, N$:
   (a) Draw its component assignment: $\mathbf{z}_n \mid \mathbf{v} \sim \text{Mult}(\{\pi_k(\mathbf{v}) : k = 1, 2, \ldots\})$.
   (b) Draw the point from the component: $\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\theta} \sim p(\mathbf{x}_n \mid \theta_{\mathbf{z}_n})$.

The scaling parameter α and component parameter distribution G_0 are hyperparameters of the model. Note that the likelihood function $p(\mathbf{x}_n \mid \theta_{\mathbf{z}_n})$ only depends on the $\mathbf{z}_n$th cluster, so we can write:

$$p(\mathbf{x}_n \mid \theta_{\mathbf{z}_n}) = \prod_{k=1}^{\infty} p(\mathbf{x}_n \mid \theta_k)^{\mathbf{z}_n} \tag{3.5}$$

We have assumed the data points are fully observed, which is why they are shaded in the

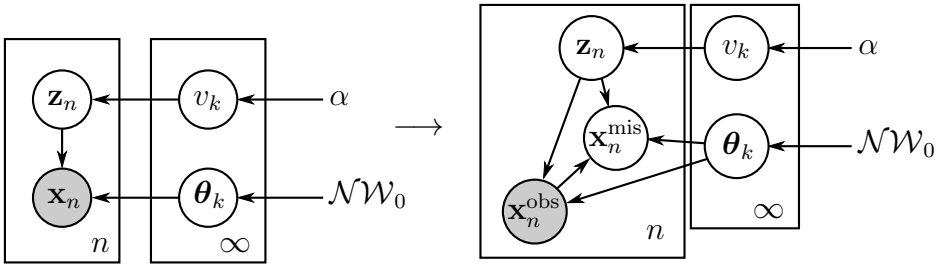

Figure 3.1: Left: the original DP mixture. Right: our DP mixture with marginal Gaussian likelihood

left side of Figure 3.1. However, this is not true: for some data points $\mathbf{x}_n$, some of the values will not be observed. Accordingly, we decompose the joint likelihood of a point into missing and observed values. By the product rule, we write:

$$p(\mathbf{x}_n \mid \theta_k) = p(\mathbf{x}_n^{\text{mis}}, \mathbf{x}_n^{\text{obs}} \mid \theta_k) = p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{x}_n^{\text{obs}}, \theta_k) p(\mathbf{x}_n^{\text{obs}} \mid \theta_k)$$

and update the graphical model accordingly (Figure 3.1, right-hand-side).

The derivations above hold for any form of p. From now on, we fix the form of p. Recall from section 2.3.1 that the marginal and conditional distributions of a Gaussian are easy to compute. Because of that, we define the $\mathbf{x}_n$ as a Gaussian. Then for the observed values of the kth component:

$$p(\mathbf{x}_n^{\text{obs}} \mid \theta_k) = \mathcal{N}\left(\mathbf{x}_n^{\text{obs}} \mid \boldsymbol{\mu}_k^{\text{obs}_n}, \left(\left(\boldsymbol{\Lambda}_k^{-1}\right)^{\text{obs}_n, \text{obs}_n}\right)^{-1}\right) \tag{3.6}$$

The missing values also follow a Gaussian. We write it explicitly in section 3.3.1.

We would like to compute the posterior distribution over our latent variables: $\mathbf{X}^{\text{mis}}$, $\mathbf{Z}$, $\mathbf{v}$ and $\boldsymbol{\theta}$, after seeing some data $\mathbf{X}^{\text{obs}}$. Following equation (2.2):

$$p(\mathbf{X}^{\text{mis}}, \mathbf{Z}, \mathbf{v}, \boldsymbol{\theta} \mid \mathbf{X}^{\text{obs}}) = \frac{p(\mathbf{X}^{\text{mis}}, \mathbf{X}^{\text{obs}}, \mathbf{Z}, \mathbf{v}, \boldsymbol{\theta})}{p(\mathbf{X}^{\text{obs}})}$$

We can separate the distribution over $\mathbf{X}^{\text{mis}}$ and its conditional dependencies, following the modified graphical model in Figure 3.1:

$$p(\mathbf{X}^{\text{mis}}, \mathbf{Z}, \mathbf{v}, \boldsymbol{\theta} \mid \mathbf{X}^{\text{obs}}) = \frac{p(\mathbf{X}^{\text{mis}} \mid \mathbf{Z}, \boldsymbol{\theta}, \mathbf{X}^{\text{obs}}) p(\mathbf{X}^{\text{obs}}, \mathbf{Z}, \mathbf{v}, \boldsymbol{\theta})}{p(\mathbf{X}^{\text{obs}})} \tag{3.7}$$

Thus, to compute $p(\mathbf{X}^{\text{mis}} \mid \mathbf{X}^{\text{obs}})$, it suffices to compute the partial posterior $p(\mathbf{Z}, \mathbf{v}, \boldsymbol{\theta} \mid \mathbf{X}^{\text{obs}})$, then multiply it by $p(\mathbf{X}^{\text{mis}} \mid \mathbf{Z}, \boldsymbol{\theta}, \mathbf{X}^{\text{obs}})$ and integrate the latent variables out.

This partial true posterior is intractable to compute, since the joint distribution has infinite terms. Thus, we approximate it using a *variational distribution*, $q_{\boldsymbol{\phi}}$ (section 2.2.5).

3.1.3 The variational distribution

Blei and Jordan (2006) choose the following variational distribution family, for a Gaussian likelihood:

$$q_{\boldsymbol{\phi}}(\mathbf{Z}, \mathbf{v}, \boldsymbol{\theta}) = \prod_{k=1}^{K-1} q_{\boldsymbol{\gamma}_k}(v_k) \prod_{k=1}^{K} q_{\boldsymbol{\eta}_k}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \prod_{n=1}^{N} q_{\boldsymbol{\tau}_n}(\mathbf{z}_n) \tag{3.8}$$

where:

$$\begin{aligned} q_{\boldsymbol{\gamma}_k}(v_k) &= \text{Beta}(v_k \,|\, \gamma_{k,1}, \gamma_{k,2}) \\ q_{\boldsymbol{\eta}_k}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) &= \mathcal{NW}\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \,|\, \mathbf{m}_k, \beta_k, \mathbf{W}_k, \nu_k\right) \\ q_{\boldsymbol{\tau}_n}(z_n) &= \text{Multinomial}(\mathbf{z}_n \,|\, \boldsymbol{\tau}_n) \end{aligned} \tag{3.9}$$

We have set $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k : k \in 1, \ldots, K\}$, and $\boldsymbol{\phi} = \{\gamma_k, \boldsymbol{\eta}_k\}_k \cup \{\tau_n\}_n$ represents all the parameters each of the distributions are conditioned on. $\mathcal{NW}$ is the Gaussian-Wishart distribution, defined in equation (B.1).

Since we separated $\mathbf{x}^{\text{mis}}$ from the rest of the model in equation (3.7), we can use this variational distribution to approximate the posterior of the other latent variables.

The first two terms of equation (3.8) represent a truncated Dirichlet Process. In effect we have set to 1 the probability of the Kth stick-breaking fraction being 1, that is, $q(v_K = 1) = 1$. This implies that, under the variational distribution, the probability $\pi_{k'}(\mathbf{v})$ of drawing a cluster $k' > K$ is zero. It follows that the distributions for $\theta_{k'}$ will never be drawn from, so they can be removed from $q_{\boldsymbol{\phi}}$. We emphasise that *only* the variational distribution $q_{\boldsymbol{\phi}}$ is finite. The model's prior and true posterior are still infinite.

We also set the component parameter prior G_0 to a Gaussian-Wishart:

$$G_0(\mu_k, \Lambda_k) = \mathcal{NW}\left(\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0^{-1} \,|\, \mathbf{m}_0, \beta_0, \mathbf{W}_0, \nu_0\right) \tag{3.10}$$

The parameters of G_0 are also hyperparameters. Usually we set a noninformative prior, such as $\mathbf{m}_0 = \mathbf{0}$, $\beta_0 = 1$, $\mathbf{W}_0 = \mathbf{I}$, $\nu_0 = D$.

Recall that the Gaussian-Wishart distribution is the conjugate prior (section 2.2.6) of the Gaussian distribution, which is our likelihood function (equation 3.6). In the fully observed case, this makes the optimal partial coordinate descent update also be in the Gaussian-Wishart. Thus, the KL-divergence of the update can be minimised in closed

form. However, this is not true in the partially observed case, as we show in section 3.2.3.

3.2 Inference in the partially observed infinite GMM

To perform variational inference of the posterior, we split the parameters to be optimised in 3 blocks ($\boldsymbol{\gamma}$, $\boldsymbol{\eta}$ and $\boldsymbol{\tau}$ in equation 3.4), and follow the block coordinate descent scheme exposed in Algorithm 1. We give closed form optimum updates for $\boldsymbol{\tau}$ and $\boldsymbol{\gamma}$. We give a closed-form approximation to the minimum for $\boldsymbol{\eta}$, which we argue eventually leads to a local maximum as well, if perhaps slower.

3.2.1 Optimal cluster assignment ($\boldsymbol{\tau}$) update

First, we can optimise the factor relating to the latent cluster assignments $\mathbf{Z}$. Using equation (2.14):

$$\begin{aligned}\log q^*_\tau(\mathbf{Z}) &= \mathop{\mathbb{E}}_{\boldsymbol{\phi}\setminus\tau}\left[\log p(\mathbf{X}, \mathbf{Z}, \mathbf{v}, \boldsymbol{\theta})\right] \\ &= \sum_{n=1}^{N}\left[\mathop{\mathbb{E}}_{\boldsymbol{\eta}}[\log p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\mu}_{\mathbf{z}_n}, \boldsymbol{\Lambda}_{\mathbf{z}_n})] + \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log p(\mathbf{z}_n \mid \mathbf{v})]\right] + \mathcal{C}\end{aligned} \tag{3.11}$$

where we wrote the parameters $\boldsymbol{\eta}$, $\boldsymbol{\gamma}$, as a shorthand for $q_{\boldsymbol{\eta}}$, $q_{\boldsymbol{\gamma}}$, which are the distributions the expectations are calculated over.

We also grouped all the terms of the joint distribution that do not depend on $\mathbf{Z}$ into the catch-all constant $\mathcal{C}$. Henceforth the value of $\mathcal{C}$ varies from expression to expression, and includes quantities irrelevant to our purpose, which is usually minimisation.

Let us focus on the second term, $\mathbb{E}_{\boldsymbol{\gamma}}[\log p(\mathbf{z}_n \mid \mathbf{v})]$. Inspecting the probability of the stick-breaking process (equation 3.2), we see that we can rewrite the multinomial distribution $p(\mathbf{z}_n \mid \mathbf{v})$ as follows:

$$p(\mathbf{z}_n \mid \mathbf{v}) = \prod_{k=1}^{\infty}(1 - v_k)^{z_{n,>k}}\, v_k^{z_{n,k}} \tag{3.12}$$

We slightly abused the notation by writing $z_{n,>k} = \sum_{j=k+1}^{\infty} z_{n,k}$, which is 1 if the one-hot vector $\mathbf{z}_n$ has the one in a position indexed by a number $> k$.

Thus the second term inside the summation of equation (3.11) is:

$$\mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log p(\mathbf{z}_n \mid \mathbf{v})] = \sum_{k=1}^{\infty}\mathop{\mathbb{E}}_{\boldsymbol{\gamma}_k}[z_{n,>k}\log(1 - v_k)] + \mathop{\mathbb{E}}_{\boldsymbol{\gamma}_k}[z_{n,k}\log v_k] \tag{3.13}$$

Recall from 3.8 that we have set $q(v_K = 1) = 1$. This does not depend on any parameters, so it is part of the expectation. In particular, this implies that, for all $j > K$, $z_{n,>j} = z_{n,j} = 0$ with probability 1. Thus they are independent of their v_j, and for all $j > K$:

$$\mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[z_{n,j} \log v_j] = \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[z_{n,j}] \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log v_j] = 0 \cdot \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log v_j] = 0 \tag{3.14}$$

Same process for the other term of the sum. This implies we can truncate the sum in equation (3.13) at K.[1] For the other terms of the sum, $z_{n,k}$ and $z_{n,>k}$ do not depend on $q_{\boldsymbol{\gamma}}$ so they can be extracted from the expectations.

Thus we expand equation (3.11), using equation (3.6) and a truncated equation (3.13):

$$\begin{aligned}
\log q^*_\tau(\mathbf{Z}) &= \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \mathop{\mathbb{E}}_{\eta_k} \left[\log (\mathcal{N}_{n,k})^{z_{n,k}}\right] + \sum_{k=1}^{K} z_{n,>k} \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log(1 - v_k)] + z_{n,k} \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log v_k] \right) + \mathcal{C} \\
\log q^*_\tau(\mathbf{Z}) &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \log \rho_{n,k} + \mathcal{C}
\end{aligned} \tag{3.15}$$

where $\mathcal{N}_{n,k}$ is shorthand for $\mathcal{N}\left(\mathbf{x}_n^{\text{obs}} \mid \boldsymbol{\mu}_k^{\text{obs}_n}, \left(\left(\boldsymbol{\Lambda}_k^{-1}\right)^{\text{obs}_n,\text{obs}_n}\right)^{-1}\right)$ and:

$$\log \rho_{n,k} = \mathop{\mathbb{E}}_{\eta_k} \left[\log \mathcal{N}_{n,k}\right] + \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log v_k] + \sum_{j=1}^{k-1} \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log(1 - v_j)] \tag{3.16}$$

Using Theorem 2.3.1, and the fact that $q_{\boldsymbol{\eta}_k}$ is a Gaussian-Wishart distribution (equation 3.9), we know that the precision matrix of $\mathcal{N}_{n,k}$ follows a Wishart distribution. From there, we can compute $\mathbb{E}_{\eta_k}\left[\log \mathcal{N}_{n,k}\right]$ using Theorem 2.3.1 and some standard manipulations (section C.1), to give:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\eta_k} \left[\log \mathcal{N}_n\right] = \frac{1}{2} \log |\mathbf{W}_k^{\text{obs}_n,\text{obs}_n}| + \frac{1}{2} \sum_{i=1}^{|\text{obs}_n|} \Psi\left(\frac{\nu_k - D + |\text{obs}| + 1 - i}{2}\right) \\
- \frac{\beta_k(\nu_k - D + |\text{obs}|)}{2} (\mathbf{x}_n^{\text{obs}} - \mathbf{m}_k^{\text{obs}_n})^\top \mathbf{W}_k^{\text{obs}_n,\text{obs}_n} (\mathbf{x}_n^{\text{obs}} - \mathbf{m}_k^{\text{obs}_n}) + \mathcal{C}
\end{aligned} \tag{3.17}$$

[1] The original derivation by Blei and Jordan (2006) of the soundness of truncating the sum depends on on $p(\mathbf{z}_n \mid \mathbf{X}, \mathbf{v}, \boldsymbol{\theta})$ being an exponential family. When the data set has missing data, this is not necessarily the case. We eventually prove it is, by deriving a closed form update. However, to do that, we needed to prove the soundness of truncating the sum first; so we give an alternative proof.

The other expectations have value:

$$\mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log v_k] = \Psi(\gamma_{k,1}) - \Psi(\gamma_{k,1} + \gamma_{k,2}) \qquad \mathop{\mathbb{E}}_{\boldsymbol{\gamma}}[\log(1 - v_k)] = \Psi(\gamma_{k,2}) - \Psi(\gamma_{k,1} + \gamma_{k,2})$$

where Ψ is the Digamma function (equation B.9).

Finally we exponentiate equation (3.15):

$$q^*_\tau(\mathbf{Z}) = \exp\left(\sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k} \log \rho_{n,k} + \mathcal{C}\right) \propto \prod_{n=1}^{N}\prod_{k=1}^{K} \rho_{n,k}^{z_{n,k}} \tag{3.18}$$

which is a multinomial distribution. Thus each $\tau_{n,k} \propto \rho_{n,k}$. They can be computed by using this formula, and then normalised such that $\sum_{k=1}^{K} \tau_{n,k} = 1$.

3.2.2 Optimising the weight distribution parameters ($\boldsymbol{\gamma}$)

We proceed in the same way as for the cluster assignments, computing the expectation of the prior over all variational factors except the ones involving $\boldsymbol{\gamma}$.

$$\log q^*_{\boldsymbol{\gamma}}(\mathbf{v}) = \left[\sum_{n=1}^{N} \mathop{\mathbb{E}}_{\boldsymbol{\tau}_n} \left[\log p(\mathbf{z}_n \mid \mathbf{v})\right]\right] + \mathop{\mathbb{E}}_{\boldsymbol{\tau}_n}\left[\sum_{k=1}^{\infty} \log p(v_k \mid \alpha)\right] + \mathcal{C}$$

The data set $\mathbf{X}$ is not involved in this calculation. Thus, no modifications are made necessary because of partial observability. After some manipulations, we arrive at the updates given by Blei and Jordan (2006):

$$\gamma_{k,1} = 1 + \sum_{n=1}^{N} \tau_{n,k} \qquad \gamma_{k,2} = \alpha + \sum_{n=1}^{N}\sum_{j=k+1}^{K} \tau_{n,j} \tag{3.19}$$

3.2.3 Optimising the component parameters ($\mathbf{m}, \beta, \mathbf{W}, \nu$)

Again we use equation (2.14) to compute the optimal partial update:

$$\begin{aligned} \log q^*_{\boldsymbol{\eta}}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) &= \mathop{\mathbb{E}}_{\boldsymbol{\gamma},\boldsymbol{\tau}} \left[\log p(\boldsymbol{\theta}) + \log p(\mathbf{x}_n^{\text{obs}} \mid \mathbf{z}_n, \boldsymbol{\theta})\right] + \mathcal{C} \\ &= \sum_{k=1}^{K} \log p(\boldsymbol{\theta}) + \sum_{n=1}^{N} \mathop{\mathbb{E}}_{\tau_n}[\log p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\theta})] + \mathcal{C} \end{aligned} \tag{3.20}$$

Recall that the probability distribution over $\mathbf{z}_n$, parametrised by $\boldsymbol{\tau}$, is multinomial.

Additionally substituting the component prior by equation (3.10), we obtain:

$$\log q^*_{\boldsymbol{\eta}}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \sum_{k=1}^{K} \log \mathcal{NW}\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1} \,|\, \mathbf{m}_0, \beta_0, \mathbf{W}_0, \nu_0\right) + \sum_{n=1}^{N}\sum_{k=1}^{K} \boldsymbol{\tau}_n \log p(\mathbf{x}_n^{\text{obs}} \,|\, \boldsymbol{\theta}_k) + \mathcal{C} \tag{3.21}$$

This expression factors across components k. Henceforth we write the expression for one component only. Expanding the Gaussian-Wishart and $p(\mathbf{x}^{\text{obs}} \,|\, \boldsymbol{\theta}_k)$:

$$\begin{aligned}\log q^*_{\boldsymbol{\eta}_k}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \log \mathcal{N}_0\left(\boldsymbol{\mu}_k \,|\, \mathbf{m}_0, (\beta_0\boldsymbol{\Lambda}_k)^{-1}\right) + \log \mathcal{W}_0(\boldsymbol{\Lambda}_k \,|\, W_0^{-1}, \nu_0) \\ + \sum_{n=1}^{N} \tau_{n,k} \log \mathcal{N}_{n,k}\left(\mathbf{x}_n^{\text{obs}} \,|\, \mu_k^{\text{obs}}, \left(\left(\boldsymbol{\Lambda}_k^{-1}\right)^{\text{obs,obs}}\right)^{-1}\right) + \mathcal{C}\end{aligned} \tag{3.22}$$

Let us check if this is an exponential family, by looking at the sufficient statistics. The sum of each logarithm of the Gaussian distributions over the observed points is:

$$\sum_{n=1}^{N} \tau_{n,k} \log \mathcal{N}_{n,k}(\cdot) = \sum_{n=1}^{N} -\frac{1}{2}\tau_{n,k} \log \left|(\boldsymbol{\Lambda}_k^{-1})^{\text{obs,obs}}\right| + \mathcal{C}$$

(note: $\mathcal{C}$ contains some terms with $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ in this expression)

For each point $\mathbf{x}_n$, there is the log-determinant of a different sub-matrix of $\boldsymbol{\Lambda}_k^{-1}$, that is interacting with parameter $\tau_{n,k}$. The quantities $\tau_{n,k}$ are not parameters of the Gaussian. However, if they are not included as parameters, then we will obtain a *different* exponential family at each step of the coordinate descent, which is also infeasible. Therefore, this cannot be an exponential family, unless we allow the number of parameters to be as large as the data.[2] Instead, we assume that all the log-determinants are $1/2 \log |\boldsymbol{\Lambda}_k|$.

The next obstacle to viewing equation (3.22) as an exponential family is the fact that each Mahalanobis distance is over a space with a different number of dimensions. To overcome that, the following theorem will be useful.

Theorem 3.2.1. *Let* $\mathbf{x}$ *be a data point, let* $\mathbf{m}$ *be a* D*-dimensional vector. Its element* m_i *is 1 if* x_i *is observed (that is,* $i \in \text{obs}$*), and 0 if* x_i *is missing (that is,* $i \notin \text{obs}$*). Then, for any* $D \times D$ *matrix* $\mathbf{A}$ *and* $D \times 1$ *vectors* $\mathbf{b}$*,* $\mathbf{c}$ *we have:*

$$(\mathbf{b} \odot \mathbf{m})^\top \mathbf{A} (\mathbf{c} \odot \mathbf{m}) = (\mathbf{b}^{\text{obs}})^\top \mathbf{A}^{\text{obs,obs}} \mathbf{c}^{\text{obs}}$$

where $\odot$ *is the element-wise or Hadamard product.*

[2] The number of parameters is also limited by 2^D, where D is the dimensionality of the data, but in practice we can assume the number of dimensions of a data set will increase with the number of points fast enough for this series to grow larger than N.

Proof. Expanding the left-hand-side:

$$(\mathbf{b} \odot \mathbf{m})^\top \mathbf{A} (\mathbf{c} \odot \mathbf{m}) = \sum_{i=1}^{D} (b_i \cdot m_i) \sum_{j=1}^{D} a_{i,j} \cdot (m_j \cdot c_j) = \sum_{i=1}^{D} b_i m_i \sum_{j=1}^{D} m_j a_{i,j} c_j$$
$$= \sum_{i \in \text{obs}} b_i \sum_{j \in obs} a_{i,j}\, c_j = (\mathbf{b}^{\text{obs}})^\top \mathbf{A}^{\text{obs,obs}} \mathbf{c}^{\text{obs}}$$

□

Thus we can write the Mahalanobis distance term of the Gaussian in equation (3.22) as:

$$(\mathbf{x}_n^{\text{obs}} - \boldsymbol{\mu}_k^{\text{obs}})^\top \left(\left(\boldsymbol{\Lambda}_k^{-1} \right)^{\text{obs,obs}} \right)^{-1} (\mathbf{x}_n^{\text{obs}} - \boldsymbol{\mu}_k^{\text{obs}}) = ((\mathbf{x}_n - \boldsymbol{\mu}_k) \odot \mathbf{m}_n)^\top \boldsymbol{\Lambda}_k ((\mathbf{x}_n - \boldsymbol{\mu}_k) \odot \mathbf{m}_n)$$

Note that this expression, in which the missing dimensions contribute zero to the Mahalanobis distance, is the point of the Gaussian likelihood with a highest value, subject to being compatible with the partially-observed data point.

We can remove the Hadamard product with $\mathbf{m}_n$ by setting $\mathbf{x}_n^{\text{mis}} := \boldsymbol{\mu}_k^{\text{mis}}$, so that the relevant terms are zero in the multiplication with the full precision matrix. However, this brings us back to where we started, with the density not being an exponential family.

What *is* possible to do, however, is to fill in the missing values of $\mathbf{x}_n$ with $\mathbb{E}[\boldsymbol{\mu}_k^{\text{mis}}] = \mathbf{m}_k^{\text{mis}}$ *at the previous iteration of the coordinate descent.* We write this previous value as $\mathbf{m}_k^{\text{mis},(t-1)}$.

Intuitively, this implies that, if we use the approximation just developed to compute the optimal parameters in closed form, the distance between this optimum and the true optimum will become smaller as $\mathbf{m}_k^{\text{mis},(t-1)}$ becomes closer to $\mathbf{m}_k^{\text{mis}}$. Essentially, the convergence of the other two coordinate-descent blocks will make the third block converge towards the optimum as well, even when the value it gets at the tth step is only an approximation to the optimum.

The form in equation (3.22) then reduces to a Gaussian-Wishart distribution, like equation (3.9). The update equations are then analogous to the ones given by Bishop (2006).

Defining $N_k = \sum_{n=1}^{N} \tau_{n,k}$ and $\bar{\mathbf{x}}_k = N_k^{-1} \sum_{n=1}^{N} \tau_{n,k} [\mathbf{x}_n^{\text{obs}}; \mathbf{m}_k^{\text{mis}_n,(t-1)}]$:

$$\begin{aligned}
\beta_k &= \beta_0 + N_k \\
\nu_k &= \nu_0 + N_k \\
\mathbf{m}_k &= \frac{1}{\beta_k}(\beta_0 \mathbf{m}_0 + N_k \bar{\mathbf{x}}_k) \\
\mathbf{W}_k^{-1} &= \mathbf{W}_0^{-1} + \frac{1}{N_k} \sum_{n=1}^{N} \tau_{n,k} ([\mathbf{x}_n^{\text{obs}}; \mathbf{m}_k^{\text{mis}_n,(t-1)}] - \bar{\mathbf{x}}_k)([\mathbf{x}_n^{\text{obs}}; \mathbf{m}_k^{\text{mis}_n,(t-1)}] - \bar{\mathbf{x}}_k)^\top \\
&\quad + \frac{\beta_0 N_k}{\beta_0 + N_k} (\bar{\mathbf{x}}_k - \mathbf{m}_0)(\bar{\mathbf{x}}_k - \mathbf{m}_0)^\top
\end{aligned} \tag{3.23}$$

3.3 Imputation

Recall the variational distribution from equation (3.8):

$$q_{\boldsymbol{\phi}}(\mathbf{Z}, \mathbf{v}, \boldsymbol{\theta}) = \prod_{k=1}^{K-1} q_{\boldsymbol{\gamma}_k}(v_k) \prod_{k=1}^{K} q_{\boldsymbol{\eta}_k}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \prod_{n=1}^{N} q_{\boldsymbol{\tau}_n}(\mathbf{z}_n) \tag{3.24}$$

And the likelihood from equation (3.6):

$$p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\theta}) = \prod_{k=1}^{\infty} \mathcal{N}\left(\mathbf{x}_n^{\text{obs}} \mid \boldsymbol{\mu}_k^{\text{obs}_n}, \left(\boldsymbol{\Lambda}_k^{\text{obs}_n, \text{obs}_n}\right)^{-1}\right)^{z_{n,k}} \tag{3.25}$$

We describe a semi-Bayesian procedure for performing imputation. In this case, we calculate the *expected* posterior of the parameters $\boldsymbol{\theta}$, and we use the resulting graphical model to perform imputation.

We are interested in the posterior probability distribution $p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{x}_n^{\text{obs}}, \mathbf{X})$. Only the likelihood $p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{Z}, \boldsymbol{\theta}, \mathbf{x}_n^{\text{obs}}, \mathbf{X})$ and the probability distributions of its parameters are relevant to compute the posterior.

Observe that the likelihood does not depend directly on $\mathbf{v}$. In the Dirichlet Process prior, $\mathbf{x}_n$ depends *indirectly* on $\mathbf{v}$, because $\mathbf{x}_n$ depends on $\mathbf{z}_n$ which depends on $\mathbf{v}$. However, in the variational distribution, $\mathbf{Z}$ is independent of $\mathbf{v}$.

Thus, $\mathbf{v}$ has no bearing on the likelihood function, and it is possible to *discard* the parameters $\boldsymbol{\gamma}$. They make the coordinate descent optimisation possible, but are in fact useless for answering imputation queries once the model is optimised. We write $q_{\boldsymbol{\phi}}(\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})$ for the marginal distribution of the remaining RVs, which is illustrated in Figure 3.2.

Contrast this with an ordinary setting, where we receive test points $\mathbf{x}_*$ that are not the same as the ones the model was trained with. In that case, it is the variational distribution over the per-point cluster assignments $\mathbf{Z}$ that becomes useless: the new point has to get a new cluster assignment, $\mathbf{z}_*$, the distribution of which depends on $\mathbf{v}$.

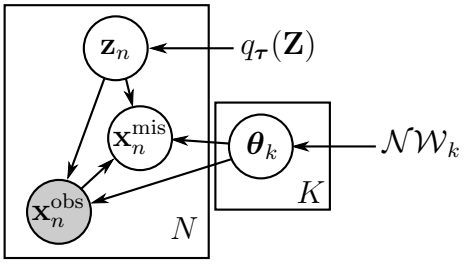

Figure 3.2: The posterior distribution of a POIGMM for imputation.

3.3.1 Semi-Bayesian imputation

.

Given the variational posterior, we can compute the expected component parameters:

$$(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Lambda}}) = \mathop{\mathbb{E}}_{q_{\boldsymbol{\phi}}}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \tag{3.26}$$

- The component means μ_k follow a Gaussian distribution with mean $\mathbf{m}_k$. Thus $\hat{\boldsymbol{\mu}}_k = \mathbf{m}_k$.
- The component precisions $\boldsymbol{\Lambda}_k$ follow a Wishart distribution with covariance $\mathbf{W}_k$. Thus $\hat{\boldsymbol{\Lambda}}_k = \nu_k \mathbf{W}_k$.

Using these point estimates of the cluster parameters, we can compute the desired conditional distribution. For each point to impute $\mathbf{x}_n^{\text{mis}}$:

$$\begin{aligned} p(\mathbf{x}_n^{\text{mis}} \,|\, \mathbf{x}_n^{\text{obs}}, \mathbf{X}_n) &= \sum_{\mathbf{z}_n} p(\mathbf{x}_n^{\text{mis}}, \mathbf{z}_n \,|\, \hat{\boldsymbol{\Lambda}}^{-1}, \hat{\boldsymbol{\mu}}, \mathbf{x}_n^{\text{obs}}, \mathbf{X}) \\ &= \sum_{\mathbf{z}_n} p(\mathbf{x}_n^{\text{mis}} \,|\, \mathbf{z}_n, \mathbf{x}_n^{\text{obs}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Lambda}}^{-1}) p(\mathbf{z}_n \,|\, \mathbf{x}_n^{\text{obs}}, \mathbf{X}) \end{aligned}$$

Now we substitute the true $p(\mathbf{z}_n \,|\, \mathbf{x}_n^{\text{obs}}, \mathbf{X})$ for the variational distribution $q_{\boldsymbol{\tau}_n}(\mathbf{z}_n)$. This is possible because, during optimisation, we already took into account the point $\mathbf{x}^{\text{obs}}$. Using

also the likelihood from equation (3.25), we obtain:

$$p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{x}_n^{\text{obs}}, \mathbf{X}_n) = \sum_k \left[\prod_{j=1}^{\infty} \mathcal{N}(\mathbf{x}_n^{\text{mis}} \mid \hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Lambda}}_j^{-1}, \mathbf{x}_n^{\text{obs}}, \mathbf{X})^{z_{n,j}} \right] q(\mathbf{z}_n = k)$$
$$= \sum_k \tau_{n,k} \mathcal{N}(\mathbf{x}_n^{\text{mis}} \mid \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Lambda}}_k^{-1}, \mathbf{x}_n^{\text{obs}})$$

where we implicitly removed the dependency on $\mathbf{X}$, since the conditional expression of the normal distributions do not use it.

The result is a mixture of Gaussians (Section 2.2.4). Each of the Gaussians is conditioned on $\mathbf{x}^{\text{obs}}$. Using the standard formula for conditional Gaussians, we get the final expression for the posterior probability of $\mathbf{x}^{\text{mis}}$:

$$p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{x}_n^{\text{obs}}, \mathbf{X}) = \sum_k \tau_{n,k} \mathcal{N}\left(\mathbf{x}_n^{\text{mis}} \mid \mathbf{m}_{n,k}, \left(\hat{\boldsymbol{\Lambda}}_k^{\text{mis,mis}}\right)^{-1}\right) \tag{3.27}$$

where:

$$\mathbf{m}_{n,k} = \hat{\boldsymbol{\mu}}_k^{\text{mis}} - \left(\hat{\boldsymbol{\Lambda}}_k^{\text{mis,mis}}\right)^{-1} \hat{\boldsymbol{\Lambda}}^{\text{mis,obs}} (\mathbf{x}_n^{\text{obs}} - \boldsymbol{\mu}_k^{\text{obs}})$$

3.4 Summary

First we perform coordinate descent to optimise the variational parameters, and then we compute the density imputation.

Algorithm 3 Inference and imputation with the POIGMM

Input: data set with missing data $\mathbf{X}$ of size $N \times D$, a tolerance ε.
Randomly initialise the parameters of the variational distribution: $\boldsymbol{\gamma}^{(0)}$, $\boldsymbol{\eta}^{(0)}$, $\boldsymbol{\tau}^{(0)}$ (equation 3.8).
$t \leftarrow 1$
while $||\boldsymbol{\tau}^{(t)} - \boldsymbol{\tau}^{(t-1)}|| + ||\boldsymbol{\gamma}^{(t)} - \boldsymbol{\gamma}^{(t-1)}|| + ||\boldsymbol{\eta}^{(t)} - \boldsymbol{\eta}^{(t-1)}|| > \varepsilon$ **do**
  Update $\boldsymbol{\tau}^{(t)}$ (proportional to equation 3.18)
  Update $\boldsymbol{\gamma}^{(t)}$ (equation 3.19)
  Update $\boldsymbol{\eta}^{(t)}$ (equation 3.23)
  $t \leftarrow t + 1$
end while
Compute $\{p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{x}_n^{\text{obs}}, \mathbf{X}) : \forall n\}$ using equation (3.27).
Output $\{p(\mathbf{x}_n^{\text{mis}} \mid \mathbf{x}_n^{\text{obs}}, \mathbf{X}) : \forall n\}$.

Chapter 4

Experiments

4.1 Data sets

We employed two data sets, obtained from the `mlbench` (Leisch and Dimitriadou, 2010) package for the R language. Both sets contain labelling information for each data point, but this is not used in our experiments.

Our method expects continuous variables as inputs, however, each of the data sets contains one boolean variable. Accordingly, we encode the boolean as a floating-point 0 or 1.

BostonHousing (Harrison and Rubinfeld, 1978). Contains information about 506 tracts of land near Boston, Massachussets, USA. The data was collected during the 1970 census. Examples of features in it include the crime rate, the concentration of nitric oxides in the atmosphere, and the property-tax rate. The boolean variable in it indicates whether the tract is adjacent to the River Charles.

Ionosphere (Lichman, 2013). The data in this set represents the returning signal of radars pointed at the ionosphere. One of the features in this data set has the same value for all points, so it was discarded prior to the analysis.

Name	Ionosphere	BostonHousing
n. data points	351	506
n. numerical features	32	12
n. boolean features	1	1

Table 4.1: Summary of the features of each used data set.

4.2 Methodology

None of the data sets have missing values. We created missing values according to different mechanisms and parameters. Then, we computed each algorithm's performance when reconstructing the data, for several measures of performance.

Concretely, the experiments were conducted according to the following procedure. Starting from a full data set $\mathbf{X}$ of N points with D dimensions:

1. Generate a data set with missing data $\mathbf{X}^{\text{mis}}$, using one of the mechanisms described in section 4.2.1. Each value that is deemed to be missing now has value $x_{i,j} = \bullet$.
2. Using *only* the remaining observed entries in $\mathbf{X}^{\text{mis}}$, we normalise the data set to have mean 0 and standard deviation 1. That is:
   (a) For each feature j, let $S_j = \{x_{i,j} : x_{i,j}^{\text{mis}} \neq \bullet, i = 1, \ldots, N\}$
   (b) Compute $\mu_j = \text{mean}(S_j)$ and $\sigma_j = \text{std\_deviation}(S_j)$.
   (c) Normalise: set $\hat{\mathbf{X}}^{\text{mis}} = [(\mathbf{X}_{:,j}^{\text{mis}} - \mu_j)/\sigma_j : j = 1, \ldots, D]$.
3. Using an imputation algorithm (from section 4.2.2), create $\hat{\mathbf{X}}^{\text{imp}}$ by imputing $\hat{\mathbf{X}}^{\text{mis}}$.
4. Recover the imputed data set by reversing the affine transform introduced to normalise:
$$\mathbf{X}^{\text{imp}} = [\hat{\mathbf{X}}^{\text{imp}} \cdot \sigma_j + \mu_j : j = 1, \ldots, D]$$
5. For each reconstruction metric F in section 4.2.3, compute $F(\mathbf{X}^{\text{true}}, \mathbf{X}^{\text{imp}})$. Depending on the metric being called, $\mathbf{X}^{\text{imp}}$ may contain continuous values, boolean values, a list of points or probability distributions.

Each experiment was repeated several times, to calculate sample standard deviations of the results.

4.2.1 Creating missing data

We created 6 patterns of missing data, following different mechanisms (Section 2.4.1) and patterns. The function to create each pattern takes one argument, r, which is the desired overall proportion of missing data.

- `MCAR_total`: for each entry of $\mathbf{X}$, set it to missing with probability r. Recall that, for a binomial distribution with N trials, the expected value is rN; so in expectation the proportion of missing data is indeed r.

- `MCAR_rows`: mark each row of $\mathbf{X}$ with probability $\sqrt{r}$. Then, for each marked row, set a randomly selected $\sqrt{r}$ fraction of the elements to missing.
- `MAR_rows`:
  1. For 1/5 of the features (named "controlling"), draw random coefficients w_j from a uniform distribution over $[-1/2, 1/2]$.
  2. Next, for each point i, compute $\alpha_i = \sum_{j \in C} w_j x_{i,j}$, for the set C of columns that are controlling. Order the rows by increasing α_i, and mark the first $5/4\, N\sqrt{r}$.
  3. For each marked row, drop each non-controlling value with probability $\sqrt{r}$.

  The overall expected proportion of dropped values is r: 4/5 are non-controlling, of these $5/4\sqrt{r}$ are marked, then out of these $\sqrt{r}$ on average are dropped.
- `NMAR`: randomly shuffle the columns, and mark the first $\lfloor D\sqrt{r} \rfloor$ of them. From each of these columns, set the $N\sqrt{r}$ lowest values to missing.
- `NMAR_random`: same procedure as `NMAR`, except: for each column, merely *mark* the $7/5N\sqrt{r}$ lowest values. Then, randomly drop 5/7 of those.

4.2.2 Algorithms

The algorithms participating in the experiments are:

- `mean`. Impute the normalised data set with a Gaussian with mean 0 and standard deviation 1. This one is merely a baseline, to check if using any of the algorithms is better than doing nothing.
- `MICE`. The MICE algorithm, which is a kind of regression imputation (Algorithm 2) that predicts a value using linear regression and then chooses at random from the k closest values to the prediction. See Section 5.1.
- `MissForest`. The MissForest algorithm, which is also a kind of regression imputation (Algorithm 2) that uses Random Forests. See Section 5.2.
- `GMM`. The Partially Observed Infinite Gaussian Mixture Model, see Chapter 3. The prior used for the POIGMM is the non-informative one described in section 3.1.3.

4.2.3 Metrics

- Normalised Root Mean Squared Error (NRMSE; Oba et al., 2003). Based on MSE

(section 2.1.3), NRMSE ranges from 0 to 1 in most cases Its definition is:

$$\text{NRMSE}(\mathbf{v}^{\text{true}}, \mathbf{v}^{\text{imp}}) = \sqrt{\frac{\text{mean}\left((\mathbf{v}^{\text{true}} - \mathbf{v}^{\text{imp}})^2\right)}{\text{variance}(\mathbf{v}^{\text{true}})}} \tag{4.1}$$

- Proportion of Falsely Classified values (PFC). The relative amount of boolean values that are wrongly classified.

$$\text{PFC}(\mathbf{v}^{\text{true}}, \mathbf{v}^{\text{imp}}) = \text{mean}_i(\mathbf{1}(v_i^{\text{imp}} \neq v_i^{\text{true}})) \tag{4.2}$$

  with $\mathbf{1}$ the indicator function (equation B.11).

Both NRMSE and PFC were used only with single imputation, since they are biased against multiple and density imputation. The squared error grows fast with distance, so if a multiple imputation algorithm makes several similar guesses, and some of them are slightly more off-track, it will be severely penalised.

In the case of MICE and MissForest, we took the average of the multiple imputations as their answer. For POIGMMs, we computed the analytic mean: the weighted average of the means of the Gaussians. This was found to improve the performance of all the methods, except of course `mean`.

- Negative Log-likelihood (equation 2.7). To compute this metric with the POIGMM, knowing $\mathbf{v}^{\text{imp}}$ and $\mathbf{v}^{\text{true}}$ is not sufficient: the original $\mathbf{X}^{\text{mis}}$ is needed, since the log-likelihood of each point depends on how the values are correlated with each other.

MICE and MissForest give only samples, so the log-likelihood cannot be directly computed. Instead, the mean and variance for the imputations each missing measurement is computed. Based on that mean and variance, a Gaussian distribution with diagonal covariance matrix is fit. Then, the log-likelihood of this Gaussian is computed.

4.3 Results and discussion

The results are all in the form of whole-page graphs. Thus, they have all been put into Appendix A.

4.3.1 How to read the graphs

Each of the Figures A.3, A.2, A.1, A.4, A.5 has the same structure. Each figure represents a missing data generation mechanism. Each column of plots represents a data set, and each row a reconstruction metric. Finally, within each plot, there are several clusters of vertical bars. The clusters indicate the proportion of missing data requested to the missing-data-creation routine (Section 4.2.1). Each bar then represents an algorithm, the height is their performance and the black bars are $\pm 2\sigma$, two standard deviations up, and two standard deviations down. In all metrics, *lower is better*.

4.3.2 Discussion

We can clearly see that, most of the time, MissForest is the dominating algorithm, at least on the NRMSE metric. Next usually comes our method, the POIGMM (here written just GMM), then MICE, and finally the `mean` baseline.

The exception are the data sets with `NMAR_rows` missing data mechanism. This data mechanism involves removing the $N\sqrt{r}$ lowest values. Thus, frequentist methods (that behave like MLE, Section 2.2.1), will always overestimate the mean of the missing values. The GMM, instead, has a *prior* that specifies that values closer to 0 are more likely, and thus does not incur so much error in this case. It is however an empty victory, all methods work roughly as badly as mean imputation here.

Where we see some improvement with respect to the other methods is, as expected, in the negative log-likelihood of the missing data. Even when using the logarithmic representation of numbers (Section 2.2.1), which greatly increases range between 0 and 1, many algorithms exceeded the range. This is perhaps because the Gaussian distribution assigns very, very low likelihood to elements that are several standard deviations away. For future experiments, a more tolerant distribution like the Student t-distribution should be used.

In any case, all the methods are using Gaussians. What is most surprising about the negative log-likelihood results is that all methods (even the POIGMM) do not work very well compared to mean imputation.

In the case of PFC, the differences between algorithms seem to be mostly noise. There is only one boolean feature per data set and, in both data sets, the number of ones in the feature is much larger than the number of zeroes. In fact, there is no PFC plot for any of the NMAR data because in many experiments all the zeroes disappeared, the problem thus becoming single-class classification.

Chapter 5

Related Work

5.1 Multiple Imputation by Chained Equations (MICE)

Multiple Imputation by Chained Equations (Van Buuren and Oudshoorn, 1999) is a multiple imputation algorithm that is designed to work in the MAR setting. In the original report, the algorithm is presented as a *Gibbs sampling* method. Gibbs sampling consists in iteratively drawing samples the conditional distributions $p(x_i \mid \mathbf{X}_{-i})$, for each i. The first sample is drawn uniformly randomly from the values of the other samples in the data set.

In the limit, this is guaranteed to converge to a sample from the true posterior. However, it is famously difficult to judge *when* the sequence of samples is close to converging. The solution MICE uses is to run several Gibbs sampling procedures, round-robin style, and stop when the variance between different sequences (which starts large) is not greater than the variance across the past samples of the same sequence.

MICE resembles coordinate descent (algorithm 1) in the sense that it does partial conditional updates that eventually converge to a solution. However, MICE is stochastic while coordinate descent is deterministic. Furthermore, MICE does not have an optimisation objective.

MICE is also an instance of regression imputation (Algorithm 2). The most common method used to draw from $p(x_i \mid \mathbf{X}_{-i})$ at each step consists in fitting a linear model to $x_i \mid \mathbf{X}_{-i}$, using that linear model to predict a value from $\hat{x}_i$, and then drawing a value from the k values of x_i present in the data set that are closest to $\hat{x}_i$. This is known as *predictive mean matching*.

Arguably, MICE could be called "stochastic regression imputation", since the authors do not specify that predictive mean matching is the partial sampling method to be used. Instead, they postulate it as a general framework, with which any conditional sampling procedure can be used.

5.2 MissForest: Random Forests for imputation

MissForest (Stekhoven and Bühlmann, 2011) is the current state of the art in imputation. The algorithm is incredibly simple: regression imputation (Algorithm 2), but using *Random Forests*.

Random Forests (Breiman, 2001) are a regression and classification algorithm that consists in stochastically growing many decision trees (for example: "if $x > 3$ then $y = 1$ else if ...") and taking a majority vote of their predictions. RFs are very popular in machine learning contests, because of their high robustness to noise, and their ability to learn with a moderate amount of data.

Random forests, compared to our work, are best explained through the following analogy. Gaussian mixture models (and thus the POIGMM) attempt to "envelope" the data with one or more elliptical, Gaussian "domes", and then derive the imputations based on the shape and location of those domes.

In contrast, Random Forests (and thus MissForest) create many layers of partitions in the observation space, assigning a label or correlated output to each partition. Then, hundreds of these layers are combined via majority (in classification) or mean (in regression) votes. Perhaps the extra flexibility of these hundreds of layers of space partitions is required to bridge the performance gap between the POIGMM and MissForest.

MissForest does not have any kind of convergence guarantees. Indeed, the stopping criterion for regression imputation with RFs is to stop when the estimate starts to diverge. That is, MissForest stops the regression imputation procedure when the difference between the current imputed matrix and the previous one is larger than the difference between the previous imputed matrix and the one that came before it.

Additionally, Stekhoven and Bühlmann (2011) only show empirically that MissForest works with data that is MCAR. However, in this thesis we learned that it also works better for data that is MAR and NMAR.

Chapter 6

Conclusions

The goal of this project was, nearly from the beginning, to create a *density* imputation algorithm that would improve the state of the art of standard *multiple* imputation methods. In a few ways, this goal has been achieved, but not in many others. Indeed, our approach seems to be better than others at representing uncertainty. However, it is likely that, in many cases, this does not compensate for worse prediction accuracy. As an example, in our experiments, often the second best method for the log-likelihood metric was simply mean imputation. On the other hand, we *did* improve on most metrics compared to the MICE algorithm, which still sees wide use.

6.1 Lessons Learned

One of the main mistakes to learn from is not starting to write earlier: not because of lack of time, but because committing ideas to paper, in a more involved way than quickly scribbled research notes, forces you to think about their shortcomings and implications in a detailed way. Thus by writing, many good ideas come to mind, and it is a waste to not be able to test and use them.

Another mistake is that of not committing to a topic at the beginning of the project and then going deeply into it. This project included several algorithm designs and implementations that did not appear in this report. This is partly to them being less effective, and partly to being restricted in number of pages and time. At the beginning, the goal was to study time-series data using neural networks, exactly like the Met Office example from the introduction. Then, the goal became imputation in i.i.d. settings; we attempted auto-encoders, multi-layer perceptrons (neural networks), mixtures of Gaussians (the final subject of this thesis) and, in the end, Gaussian processes.

6.2 Future Directions

There are many ways to extend the research that is written in this document. The first and most obvious one, and actually one that we have partially done already, is to derive a correct variational inference algorithm that provably minimises the KL-divergence. This is likely possible to do with gradient descent, or perhaps hybrid coordinate and gradient descent. Gradient descent does not require analytically available partial globally optimal updates. As a potential positive side effect, much larger data sets could be handled, by using *stochastic* gradient descent and only accounting for a small subset of the points at every update.

Components of different types, not just Gaussians, could be added. With every component being a Gaussian multiplied by a Multinomial, for example, it would become possible to perform imputation of mixed categorical and numerical data sets. If it proves possible to have several types of components, a combination of Gaussians and Dirichlet distributions would likely make it easier to model densities accurately.

Another line of extension would be to include the conditional distribution of the missing values into variational inference. This would allow us to take into account all the uncertainty we sidestepped in Section 3.3. It is possible that this has been done already, since it would be the Bayesian counterpart to the Expectation Maximisation type of imputation algorithms (Gold and Bentler, 2000).

Finally, and perhaps most importantly, it would be interesting to verify by extensive experiments the potential benefits of density imputation described in the introduction. The presented POIGMM algorithm, combined with the machine learning methods mentioned there, would be enough to do that.

Appendix A

Experiments

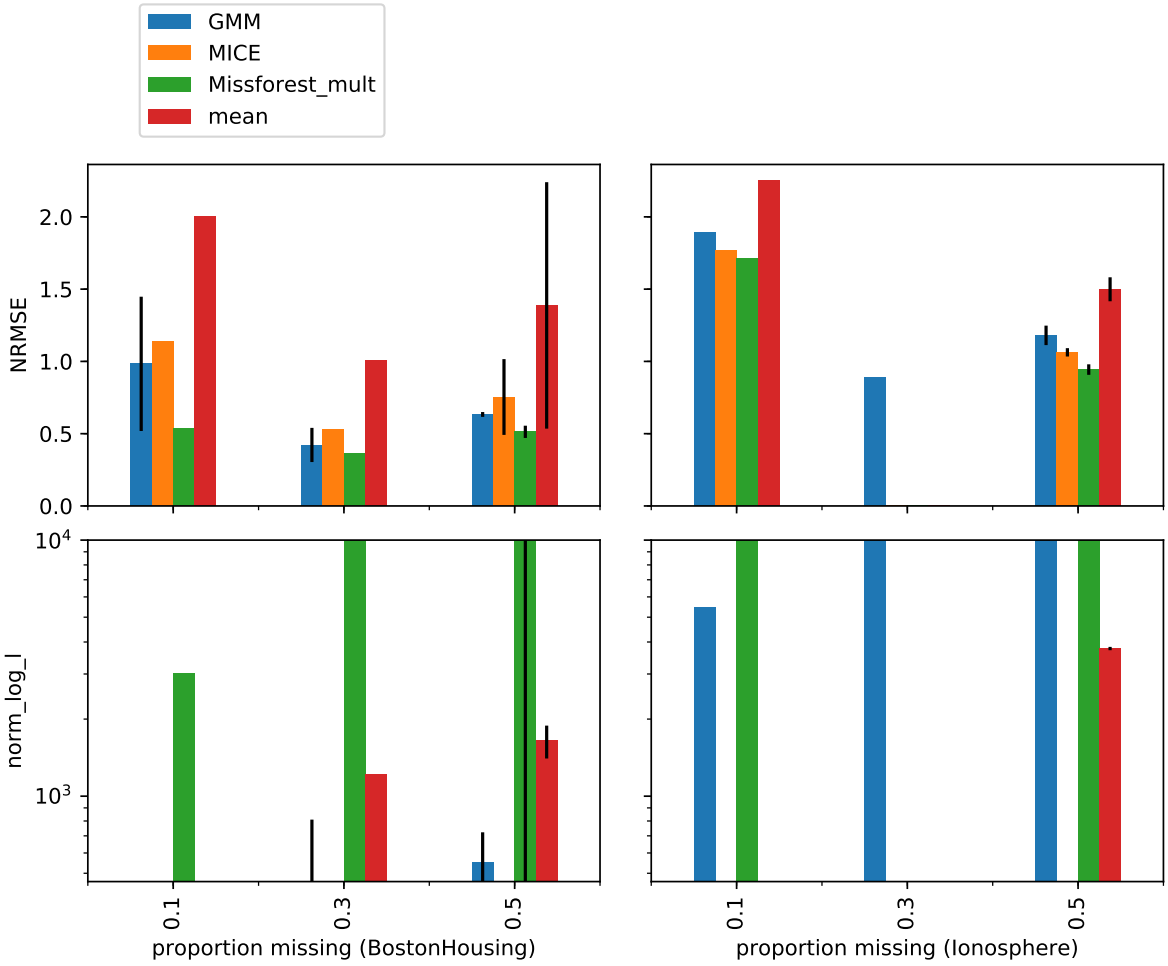

Figure A.1: `NMAR_random` missingness (Section 4.2.1). In all metrics, lower is better.

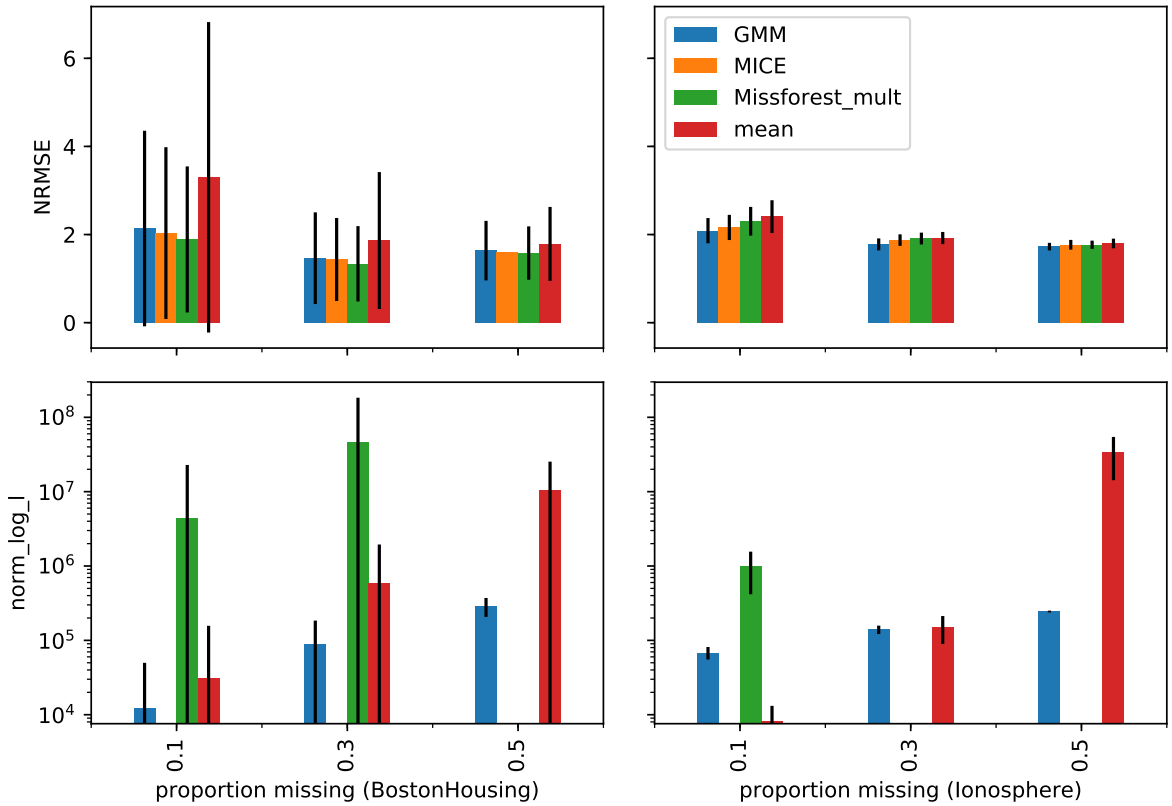

Figure A.2: `NMAR` missingness (Section 4.2.1). In all metrics, lower is better.

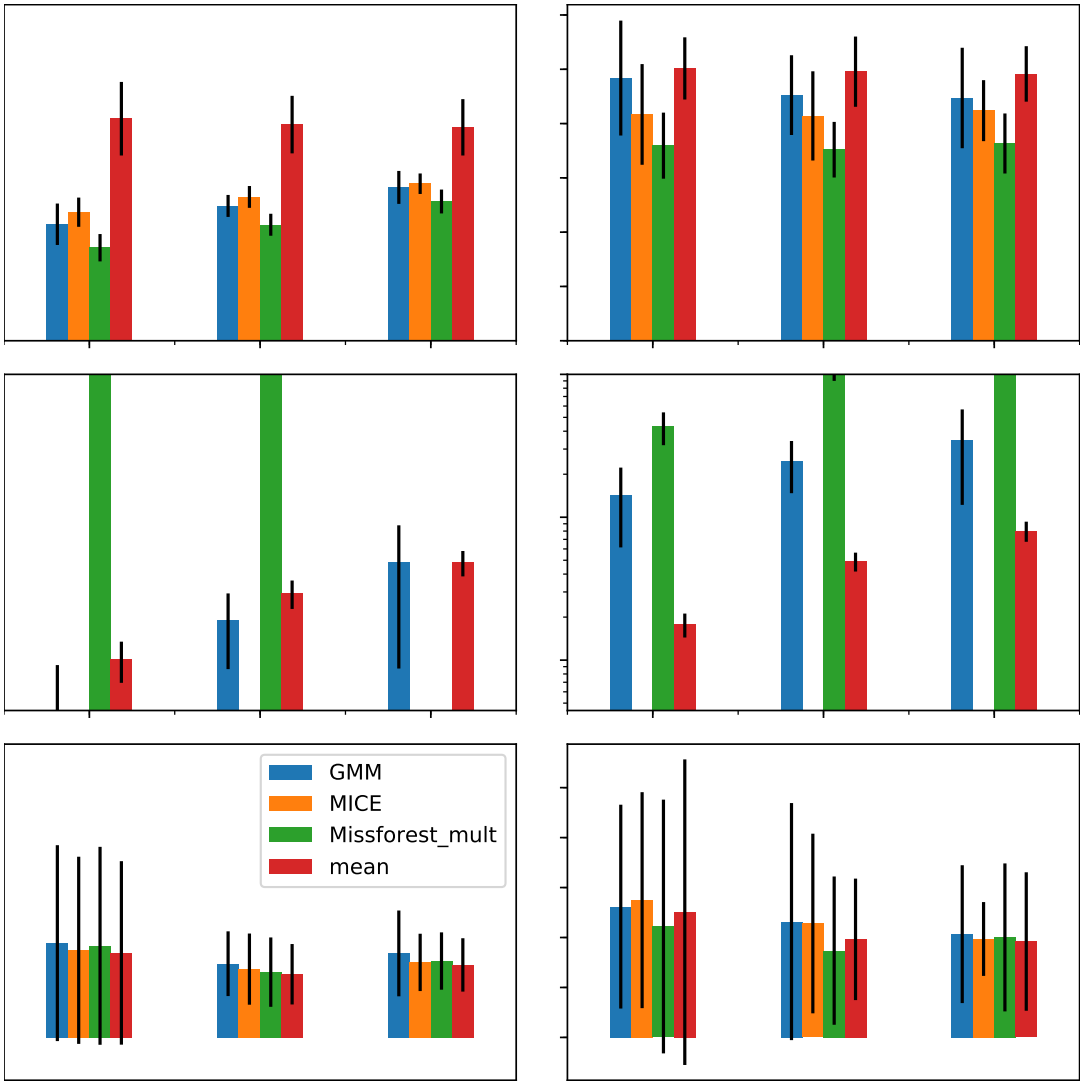

Figure A.3: `MAR_rows` missingness (Section 4.2.1). In all metrics, lower is better.

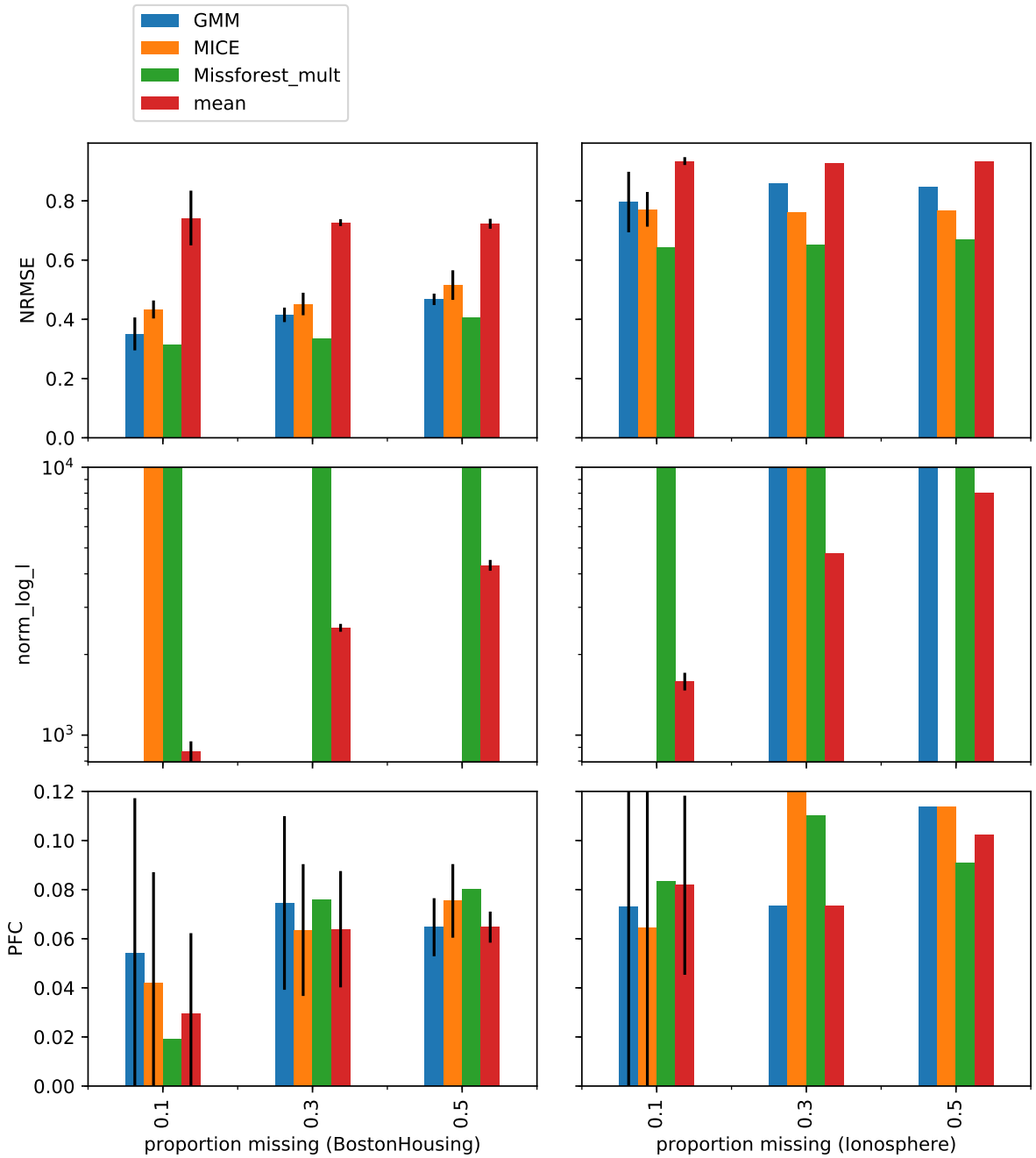

Figure A.4: `MCAR_total` missingness (Section 4.2.1). In all metrics, lower is better.

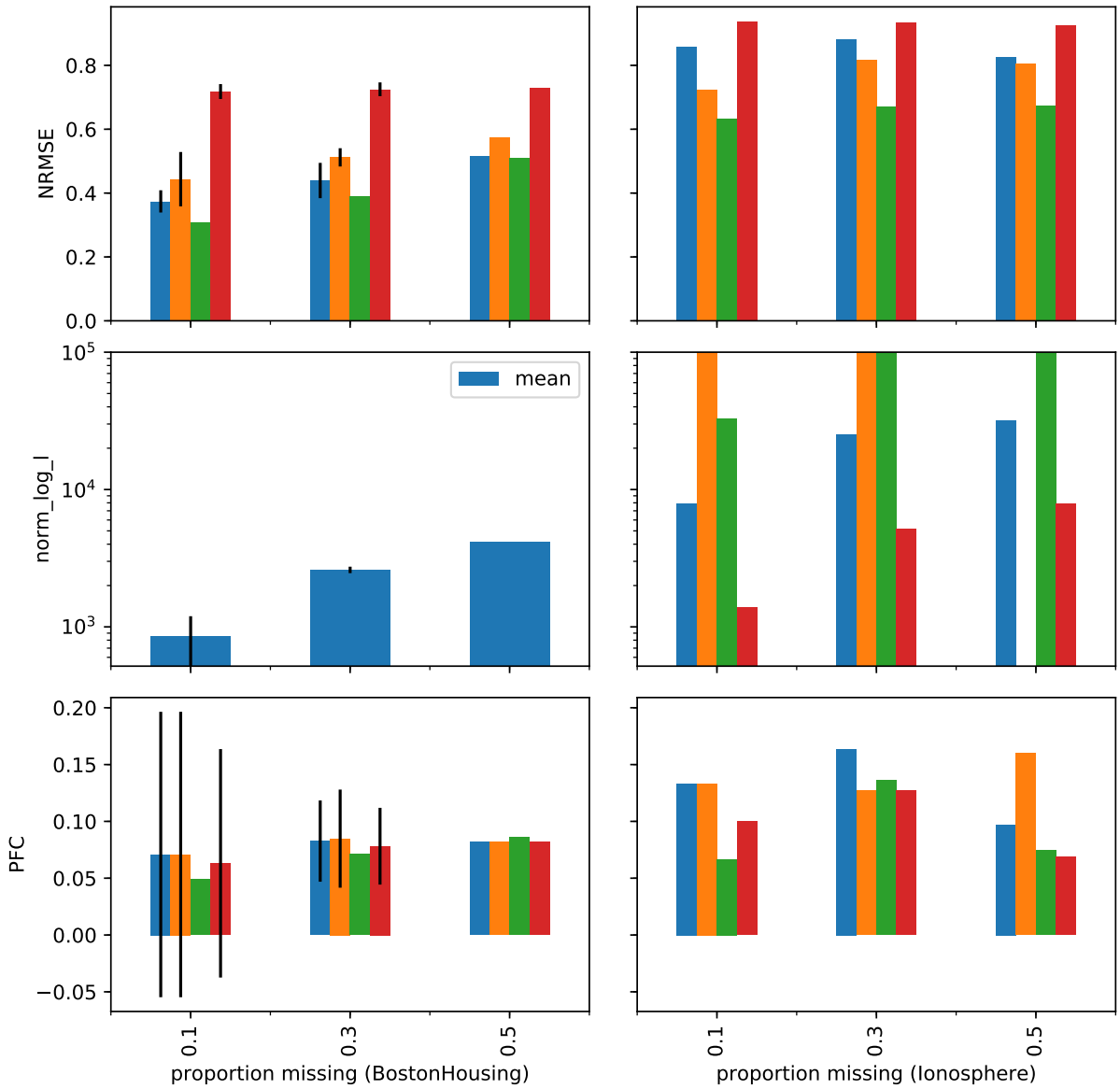

Figure A.5: `MCAR_rows` missingness (Section 4.2.1). In all metrics, lower is better.

Appendix B

Mathematical definitions

B.1 Probability distributions

B.1.1 Gaussian-Wishart

It is a distribution over a D-dimensional vector and a $D \times D$ PSD matrix. Its expression is (Bishop, 2006):

$$\mathcal{NW}(\boldsymbol{\mu}, \boldsymbol{\Lambda} \,|\, \mathbf{m}, \beta, \mathbf{W}, \nu) = \mathcal{N}(\boldsymbol{\mu} \,|\, \mathbf{m}, (\beta\boldsymbol{\Lambda})^{-1})\mathcal{W}(\boldsymbol{\Lambda} \,|\, \mathbf{W}, \nu) \tag{B.1}$$

with $\mathcal{N}$ as in equation (2.16) and Wishart distribution $\mathcal{W}$ as in equation (2.18). Note that the Gaussian's precision depends on the output of the Wishart.

The Gaussian-Wishart is the conjugate distributionsection 2.2.6 of the Gaussian. It is also an exponential family (section 2.2.6):

$$\mathcal{NW}(\boldsymbol{\mu}, \boldsymbol{\Lambda} \,|\, \mathbf{m}, \beta, \mathbf{W}, \nu) = \exp(\mathbf{g}(\mathbf{m}, \beta, \mathbf{W}, \nu) \cdot \mathbf{f}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) - h(\mathbf{m}, \beta, \mathbf{W}, \nu))$$

Where $\mathbf{f}$ and $\mathbf{g}$ are vectors, $\cdot$ is their dot product, and:

$$\mathbf{f}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \Big[\log|\boldsymbol{\Lambda}|; \quad \boldsymbol{\mu}^\top\boldsymbol{\Lambda}\boldsymbol{\mu}; \quad \boldsymbol{\Lambda}\boldsymbol{\mu}; \quad \mathrm{vec}(\boldsymbol{\Lambda})\Big] \tag{B.2}$$

$$\mathbf{g}(\mathbf{m}, \beta, \mathbf{W}, \nu) = \Big[\tfrac{\nu-D}{2}; \quad \beta; \quad -2\beta\mathbf{m}; \quad \mathrm{vec}(\beta\mathbf{m}\mathbf{m}^\top + \mathbf{W}^{-1})\Big] \tag{B.3}$$

$$h(\mathbf{m}, \beta, \mathbf{W}, \nu) = \log\left[2^{(\nu+1)D/2}\pi^{D(D+1)/4}\beta^{-D/2}|W|^{\nu/2}\prod_{i=1}^{D}\Gamma\left(\frac{\nu+1-i}{2}\right)\right] \tag{B.4}$$

Derivation of the exponential family form

According to Teh (2007), the Gaussian-Wishart distribution can be written as follows:

$$\mathcal{NW}(\boldsymbol{\mu}, \boldsymbol{\Lambda} \,|\, \mathbf{m}, \beta, \mathbf{W}, \nu) = h(\mathbf{m}, \beta, \mathbf{W}, \nu) \exp(f(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \mathbf{m}, \beta, \mathbf{W}, \nu))$$

With h given in section B.1.1 and:

$$f(\cdot) = \frac{\nu - d}{2} \log |\boldsymbol{\Lambda}| - \frac{1}{2} \mathrm{tr} \left(\boldsymbol{\Lambda} \left(\beta \mu \mu^\top - 2\mu(\beta \mathbf{m})^\top + \beta \mathbf{m}\mathbf{m}^\top + \mathbf{W}^{-1} \right) \right)$$

We take $(\log |\boldsymbol{\Lambda}|)/2$ as the first sufficient statistic. Observe that the trace on the right can be written as:

$$\mathrm{tr}\,(\cdot) = \beta \mu^\top \boldsymbol{\Lambda} \mu - (2\beta \mathbf{m})^\top \boldsymbol{\Lambda} \mu + \mathrm{tr} \left(\boldsymbol{\Lambda} \left(\beta \mathbf{m}\mathbf{m}^\top + \mathbf{W}^{-1} \right) \right)$$

By inspection, we can separate the natural parameters from the sufficient statistics. The result is in section B.1.1

B.1.2 The Beta distribution

The Beta distribution has support $0 \leq x \leq 1$, for $x \in \mathbb{R}$. A variable x is Beta distributed with shape parameters α and $\beta \in \mathbb{R}$ if:

$$p(x \,|\, \alpha, \beta) \propto x^{\alpha - 1} (1 - x)^{\beta - 1} \tag{B.5}$$

with normalisation constant, known as the Beta function:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \tag{B.6}$$

B.1.3 Multinomial Distribution

Let $\mathbf{x}$ be a random K-dimensional one-hot vector. Then the multinomial distribution assigns probability w_k to the vector having the kth dimension equal to 1:

$$p(\mathbf{x}) = \prod_{k=1}^{K} w_k^{x_k} \tag{B.7}$$

B.2 Functions

Gamma function

The Γ function is defined as the integral:

$$\Gamma(x) = \int_0^\infty z^{x-1} e^{-z} dz \tag{B.8}$$

It can be thought of as a generalisation of factorials to all the real numbers.

Digamma function

The Digamma function is the first derivative of the logarithm of the Gamma function. It is usually calculated using a series approximation.

$$\Psi(x) = \frac{d}{dx} \log \Gamma(x) \tag{B.9}$$

Dirac delta

The Dirac delta is an infinite impulse function at the coordinate origin. It has the following properties:

$$\int_{\mathbb{R}^D} \delta(\mathbf{x}) dx = 1 \qquad \int_{\mathbb{R}^D} f(\mathbf{x}) \delta(\mathbf{x} - \mathbf{x}_0) dx = f(\mathbf{x}_0) \tag{B.10}$$

Indicator function

Let p be a predicate. The indicator function $\mathbf{1}[p]$ is:

$$\mathbf{1}[p] = \begin{cases} 0 \text{ if } p \text{ is false} \\ 1 \text{ if } p \text{ is true} \end{cases} \tag{B.11}$$

B.3 Linear Algebra

Positive semidefinite (PSD) matrix

A matrix $\mathbf{M}$ is PSD if and only if, for all vectors of the appropriate dimensionality $\mathbf{x} \neq 0$, $\mathbf{x}^\top \mathbf{M} \mathbf{x} \geq 0$. PSD matrices are always Hermitian, that is, symmetric and with real values.

Appendix C

Extra derivations

This appendix contains the derivation of some results that are long, or not relevant for the experiments carried out.

C.1 Expectation of the logarithm of the normal

We wish to compute the value of $\mathbb{E}_{\mathcal{W}_k}\mathbb{E}_{\mathcal{N}_k}[\mathcal{N}_{n,k}]$. That is, the expected value under a Gaussian-Wishart distribution, as defined in equation (3.9), of the normal distribution $\mathcal{N}_{n,k}$, which is a distribution over a |obs|-dimensional subset of the mean of $\mathcal{NW}_k$, such as the one above Equation (3.16).

We set $\boldsymbol{\Sigma}_k = \boldsymbol{\Lambda}_k^{-1}$ and apply the logarithm:

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathcal{W}_k}\mathop{\mathbb{E}}_{\mathcal{N}_k}\left[-\log\mathcal{N}_n(\mathbf{x}_n^{\text{obs}} \mid \boldsymbol{\mu}_k^{\text{obs}_n}, \boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n})\right] &= \mathop{\mathbb{E}}_{\mathcal{W}_k(\boldsymbol{\Sigma}_k \mid \mathbf{W}_k,\nu_k)}\left[\frac{1}{2}\log|\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n}|\right. \\ &\left. + \mathop{\mathbb{E}}_{\mathcal{N}_k(\boldsymbol{\mu}_k^{\text{obs}_n} \mid \mathbf{m}_k^{\text{obs}_n}, \beta_k^{-1}\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n})}\left[-\frac{1}{2}(\mathbf{x}_n^{\text{obs}} - \boldsymbol{\mu}_k^{\text{obs}_n})^\top \beta_k \left(\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n}\right)^{-1}(\mathbf{x}_n^{\text{obs}} - \boldsymbol{\mu}_k^{\text{obs}_n})\right]\right]\end{aligned} \tag{C.1}$$

Focusing on the second expectation term, we change the random variable to $\boldsymbol{\mu}_k^{\text{obs}_n} = y + \mathbf{m}_k^{\text{obs}_n}$:

$$\begin{aligned}&\mathop{\mathbb{E}}_{\mathcal{N}_k(y \mid 0, \beta_k^{-1}\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n})}\left[-\frac{\beta_k}{2}(\mathbf{x}_n^{\text{obs}} - y - \mathbf{m}_k^{\text{obs}_n})^\top \boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n}(\mathbf{x}_n^{\text{obs}} - y - \mathbf{m}_k^{\text{obs}_n})\right] \\ &= -\frac{\beta_k}{2}(\mathbf{x}_n^{\text{obs}} - \mathbf{m}_k^{\text{obs}_n})^\top \left(\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n}\right)^{-1}(\mathbf{x}_n^{\text{obs}} - \mathbf{m}_k^{\text{obs}_n}) \\ &\quad + \mathop{\mathbb{E}}_{\mathcal{N}_k(\mathbf{y} \mid 0, \beta_k^{-1}\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n})}\left[-\frac{\beta_k}{2}\mathbf{y}^\top \left(\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n}\right)^{-1}\mathbf{y} + \beta_k \mathbf{y}^\top \left(\boldsymbol{\Sigma}_k^{\text{obs}_n,\text{obs}_n}\right)^{-1}(\mathbf{x}_n^{\text{obs}} + \mathbf{m}_k^{\text{obs}_n})\right]\end{aligned}$$

By linearity of expectation and since $\mathbb{E}_{\mathcal{N}_k}[\mathbf{y}] = 0$, the expectation term is:

$$\mathop{\mathbb{E}}_{\mathcal{N}_k(\mathbf{y}\,|\,0,\beta_k^{-1}\boldsymbol{\Sigma}_k^{\mathrm{obs}_n,\mathrm{obs}_n})}\left[-\frac{\beta_k}{2}\mathbf{y}^\top\left(\boldsymbol{\Sigma}_k^{\mathrm{obs}_n,\mathrm{obs}_n}\right)^{-1}\mathbf{y}\right]$$

Let $\mathbf{K} = (\beta_k^{-1/2}\mathbf{L})(\beta_k^{-1/2}\mathbf{L})^\top = \beta_k^{-1}\boldsymbol{\Sigma}_k^{\mathrm{obs}_n,\mathrm{obs}_n}$. $\mathbf{K}$ is the covariance matrix of the distribution $\mathbf{y}$ follows, so $\mathbf{y} = \beta_k^{-1/2}\mathbf{L}\mathbf{z}$ for $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ a normally distributed vector. Also $\left(\boldsymbol{\Sigma}_k^{\mathrm{obs}_n,\mathrm{obs}_n}\right)^{-1} = (\mathbf{L}^{-1})^\top\mathbf{L}^{-1}$. We rewrite the above expectation:

$$\mathop{\mathbb{E}}_{\mathcal{N}_k(\mathbf{z}\,|\,0,\mathbf{I})}\left[-\frac{\beta_k}{2}(\beta_k^{-1/2}\mathbf{L}\mathbf{z})^\top(\mathbf{L}^{-1})^\top\mathbf{L}^{-1}(\beta_k^{-1/2}\mathbf{L}\mathbf{z})\right] = \mathop{\mathbb{E}}_{\mathcal{N}_k(\mathbf{z}\,|\,0,\mathbf{I})}\left[-\frac{1}{2}\mathbf{z}^\top\mathbf{z}\right] = -\frac{1}{2}|\mathrm{obs}|$$

since $\mathbf{z}^\top\mathbf{z}$ follows a χ^2 distribution with $|\mathrm{obs}|$ degrees of freedom.

We substitute the terms we just derived into equation (C.1). Recall that we are deriving an expression for optimising a KL-divergence, so we can ignore the terms that do not depend on the variational parameters.

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathcal{W}_k}\mathop{\mathbb{E}}_{\mathcal{N}_k}\left[\log\mathcal{N}_n\right] = \mathop{\mathbb{E}}_{\mathcal{W}_k(\boldsymbol{\Sigma}_k\,|\,\mathbf{W}_k,\nu_k)}\Bigg[&-\frac{1}{2}\log|\boldsymbol{\Sigma}_k^{\mathrm{obs}_n,\mathrm{obs}_n}| \\ &-\frac{\beta_k}{2}(\mathbf{x}_n^{\mathrm{obs}}-\mathbf{m}_k^{\mathrm{obs}_n})^\top\left(\boldsymbol{\Sigma}_k^{\mathrm{obs}_n,\mathrm{obs}_n}\right)^{-1}(\mathbf{x}_n^{\mathrm{obs}}-\mathbf{m}_k^{\mathrm{obs}_n})\Bigg] + \mathcal{C}\end{aligned}$$

By Theorem 2.3.1, $\boldsymbol{\Sigma}_k^{\mathrm{obs}_n,\mathrm{obs}_n}$ follows a Wishart distribution with covariance $\mathbf{W}_k^{\mathrm{obs}_n,\mathrm{obs}_n}$ and degrees of freedom $\nu_k - D + |\mathrm{obs}|$. Using equations (2.20 and 2.21) the resulting expectation is:

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathcal{W}_k}\mathop{\mathbb{E}}_{\mathcal{N}_k}\left[\log\mathcal{N}_n\right] = &\frac{1}{2}\log|\mathbf{W}_k^{\mathrm{obs}_n,\mathrm{obs}_n}| + \frac{1}{2}\sum_{i=1}^{|\mathrm{obs}_n|}\Psi\left(\frac{\nu_k - D + |\mathrm{obs}| + 1 - i}{2}\right) \\ &-\frac{\beta_k(\nu_k - D + |\mathrm{obs}|)}{2}(\mathbf{x}_n^{\mathrm{obs}}-\mathbf{m}_k^{\mathrm{obs}_n})^\top\mathbf{W}_k^{\mathrm{obs}_n,\mathrm{obs}_n}(\mathbf{x}_n^{\mathrm{obs}}-\mathbf{m}_k^{\mathrm{obs}_n}) + \mathcal{C}\end{aligned}$$

Bibliography

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer. ISBN: 9780387310732.

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

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Che, Zhengping et al. (2016). "Recurrent neural networks for multivariate time series with missing values". In: *arXiv preprint arXiv:1606.01865*.

Cinar, Yagmur G et al. (2017). "Time Series Forecasting using RNNs: an Extended Attention Mechanism to Model Periods and Handle Missing Values". In: *arXiv preprint arXiv:1703.10089*.

Damianou, Andreas C, Michalis K Titsias, and Neil D Lawrence (2016). "Variational inference for latent variables and uncertain inputs in Gaussian processes". In: *The Journal of Machine Learning Research* 17.1, pp. 1425–1486.

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

Gold, Michael Steven and Peter M Bentler (2000). "Treatments of missing data: A Monte Carlo comparison of RBHDI, iterative stochastic regression imputation, and expectation-maximization". In: *Structural Equation Modeling* 7.3, pp. 319–355.

Grippo, L. and M. Sciandrone (2000). "On the convergence of the block nonlinear Gauss–Seidel method under convex constraints". In: *Operations Research Letters* 26.3, pp. 127 –136. ISSN: 0167-6377. DOI: http://dx.doi.org/10.1016/S0167-6377(99)00074-7.

Harrison, David and Daniel L Rubinfeld (1978). "Hedonic housing prices and the demand for clean air". In: *Journal of environmental economics and management* 5.1, pp. 81–102.

Horn, Roger A. and Charles R. Johnson (2012). *Matrix Analysis*. 2nd ed. Cambridge University Press. DOI: 10.1017/9781139020411.

Iwata, Tomoharu, David Duvenaud, and Zoubin Ghahramani (2012). "Warped mixtures for nonparametric cluster shapes". In: *arXiv preprint arXiv:1206.1846*.

Johnson, Matthew et al. (2016). "Composing graphical models with neural networks for structured representations and fast inference". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 2946–2954.

Leisch, Friedrich and Evgenia Dimitriadou (2010). *mlbench: Machine Learning Benchmark Problems*. R package version 2.1-1.

Lichman, M. (2013). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Little, Roderick J. A. and Donald B. Rubin (1989). "The Analysis of Social Science Data with Missing Values". In: *Sociological Methods & Research* 18.2-3, pp. 292–326. DOI: 10.1177/0049124189018002004.

Little, Roderick J.A. and Donald B Rubin (2002). *Statistical analysis with missing data*. 2nd ed. John Wiley & Sons.

Met Office Weather Stations. http://www.metoffice.gov.uk/learning/science/first-steps/observations/weather-stations. Last retrieved: 26 August 2017. Link to Internet Archive.

Muirhead, R.J. (1982). *Aspects of Multivariate Statistical Theory*. Wiley Series in Probability and Statistics. Wiley. ISBN: 9780471094425.

Nebot-Troyano, Guillermo and Lluís A Belanche-Muñoz (2010). "A kernel extension to handle missing data". In: *Research and Development in Intelligent Systems XXVI*. Springer, pp. 165–178.

Oba, Shigeyuki et al. (2003). "A Bayesian missing value estimation method for gene expression profile data". In: *Bioinformatics* 19.16, pp. 2088–2096. DOI: 10.1093/bioinformatics/btg287.

Raghunathan, Trivellore E et al. (2001). "A multivariate technique for multiply imputing missing values using a sequence of regression models". In: *Survey methodology* 27.1, pp. 85–96.

Rubin, Donald B (2004). *Multiple imputation for nonresponse in surveys*. Vol. 81. John Wiley & Sons.

Sethuraman, Jayaram (1994). "A constructive definition of Dirichlet priors". In: *Statistica sinica*, pp. 639–650.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of machine learning research* 15.1, pp. 1929–1958.

Stekhoven, Daniel J and Peter Bühlmann (2011). "MissForest—non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1, pp. 112–118.

Teh, Yee Whye (2007). *Exponential Families: Gaussian, Gaussian-Gamma, Gaussian-Wishart, Multinomial*.

Van Buuren, Stef and Karin Oudshoorn (1999). *Flexible multivariate imputation by MICE*. Leiden: TNO.

Zio, Marco Di, Ugo Guarnera, and Orietta Luzi (2007). "Imputation through finite Gaussian mixture models". In: *Computational Statistics & Data Analysis* 51.11. Advances in Mixture Models, pp. 5305 –5316. ISSN: 0167-9473. DOI: http://dx.doi.org/10.1016/j.csda.2006.10.002.