Unsupervised automatic dataset repair

James U. Allingham
St Edmund's College

UNIVERSITY OF
CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Department of Computer Science and Technology
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: jua23@cam.ac.uk

June 12, 2018

Declaration

I James U. Allingham of St Edmund's College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 12842

Signed:

Date:

Acknowledgements

Firstly, I would like to extend my gratitude to my supervisor Prof. Zoubin Ghahramani for taking me on despite his incredibly busy schedule, and for the useful advice throughout the course of this project.

Secondly, I would like to thank Dr. Christian Steinruecken who co-supervised me, for his patience and dedication to helping me learn, as well as his meticulous feedback and support throughout the dissertation.

Thirdly, I would like to thank my course advisor Prof. Alan Blackwell for all the advice he has given me throughout the last 9 months, and for all of the stimulating conversations.

Finally, I would like to thank my girlfriend Sasha, for making the whole process easier by working alongside me.

Unsupervised automatic dataset repair

Abstract

This dissertation describes algorithms for repairing datasets that contain missing or corrupt data. Datasets with missing values are a common occurrence, due to faulty sensor readings, unanswered survey questions, or data loss, for example. Unfortunately, many common algorithms fail on such datasets. One option is not to use algorithms that fail on corrupt datasets, and use more robust algorithms instead. A second option is to repair the dataset first, and then use the original algorithm; this is the approach taken in this dissertation. Using robust algorithms that handle missing data natively and sensibly should always be preferred: the algorithm has the opportunity to make choices that take into account both the defects in the data and the objective of the computation. However, such algorithms are not always available, and might not be able to handle all types of dataset defects equally well.

This dissertation presents AutoImpute, an application for repairing missing data using probabilistic machine learning models. The tool includes implementations of various imputation models, some of which have been implemented with automatic Bayesian model comparison. In developing this tool, various approaches were taken to investigate the challenges encountered when imputing missing data.

Imputation is difficult for several reasons: the data types of variables must be maintained, different types of missing data might need to be treated differently, and it is difficult to evaluate models for missing data. Probabilistic machine learning offers solutions to some of these problems. Bayesian inference allows us to encode prior knowledge about the data into our models, and helps to prevent over-fitting. Bayesian model comparison can be used to choose among several probabilistic models automatically, based on the data. Probabilistic machine learning methods often allow sampling multiple imputations, rather than just one; these samples can be used to get error bounds or uncertainty estimates in later data processing steps. This dissertation hopes to provide a foundation for future work in solving problems involved in imputation of missing data.

Contents

List of Figures

List of Tables

Abbreviations

DP	Dirichlet process
EM	Expectation maximisation
GMM	Gaussian mixture model
i.i.d.	independent and identically distributed
KL	Kullback–Leibler
MAP	Maximum a posteriori
MAR	Missing at random
MCAR	Missing completely at random
MICE	Multiple imputation by chained equations
MLE	Maximum likelihood estimation
NMAR	Not missing at random
RF	Random forest
VB	Variational Bayes

Chapter 1

Introduction

Datasets are often messy – it is common for values to be missing or corrupt. Examples include empty cells in spreadsheets, unanswered survey questions, or readings from faulty sensors. Unfortunately, despite the frequent occurrence of such defects, software engineers tend not to develop algorithms that are robust to missing values. As a result, many common algorithms fail on such datasets. Even simple algorithms, such as calculating the mean of a set of values, fail when applied to missing values.

What should one do in a situation where a machine learning, visualization, or other algorithm cannot handle missing elements in a dataset? One option could be to remove all of the entries that contain missing values. However, in datasets with many variables, this can mean throwing away a large portion of the data, which is undesirable for at least three reasons: firstly, some of the information being thrown away could be useful or important for the algorithm. For example, most machine learning algorithms tend to perform better when trained with more data. Secondly, collecting and curating data can be expensive, so having to throw much of it away is rather dissatisfying. Thirdly, depending on the mechanism that caused the data to be missing, throwing away data can bias further analyses. An alternative approach is to try and repair the dataset by filling in the missing values with guesses for what the original value was. This approach is called data *imputation*. Unfor-

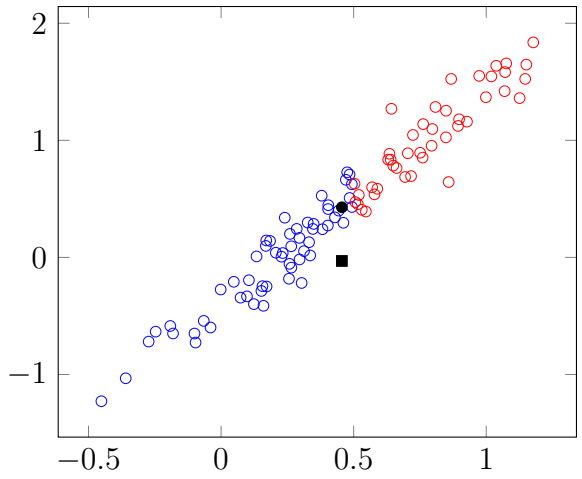


Figure 1.1: An example of why it is difficult to analyse datasets with missing data. The blue circles are non-missing data. Only the y-values for the red circles are missing – the y-value has been deleted for points where the x-value is above 0.5. In the real world this could be due to a sensor which cannot measure voltages when there is a current above 0.5 A. The black square shows the arithmetic mean for the observed data. The black circle shows the true arithmetic mean calculated using full dataset.

tunately, while this process does not throw away as much useful information, it can still introduce bias into further analyses.

For example, consider an algorithm that computes the arithmetic mean of a set of 2D points. If the dataset *does not* contain any missing elements, then this is as simple as calculating the arithmetic mean of the x- and y-values separately. However, if the dataset *does* contain missing elements then this simple approach may not work. For example, suppose that points with $x > 0.5$ have their y-values deleted. In this case, the calculated mean of the y-values could be significantly lower than true mean. This situation is depicted in Figure 1.1. Note that the y-value of the true mean is higher than most of the observed y-values.

There are many factors to consider when imputing missing data. One such consideration is the type of the data that is missing. The method for imputing a categorical value will likely be different to that of a continuous value. Another consideration is the mechanism that caused the data to be missing

in the first place. Imputing data that is missing at random is different from imputing data that was systematically removed or censored.

This dissertation aims to investigate issues such as those discussed above. The main contributions of this dissertation are:

- AutoImpute – a command line application for repairing missing data using a variety of algorithms and probabilistic models,
- a discussion of key challenges with missing data that were encountered during the development of AutoImpute, and
- proposed solutions to these challenges.

Chapter 2

Background

2.1 Probability Theory

Probability theory forms the basis for machine learning as it provides us with the mathematical machinery for understanding and representing uncertainty.

Consider two random variables X and Y, from which can take values x from a set $\mathcal{X}$ and y from a set $\mathcal{Y}$, respectively. The probability that the random variable X takes on the value x is $p(X = x)$, and $p(X)$, is the *probability distribution* of X. We call $p(X, Y)$ the *joint distribution* of X and Y, and $p(X \mid Y)$ the *conditional distribution* of X given Y.

The two *fundamental rules of probability* according to Bishop (2006, ch 1) are the sum rule and the product rule:

$$p(X=x) = \sum_{y \in \mathcal{Y}} p(X=x, Y=y) \tag{2.1}$$

$$p(X=x, Y=y) = p(X=x \mid Y=y)\, p(Y=y)\,. \tag{2.2}$$

For the continuous case, the summation in the sum rule is replaced with an integral over y. In the context of the sum rule, the distribution $p(X)$ is known as the *marginal distribution* of X.

Using the product rule, and the symmetry $p(X, Y) = p(Y, X)$, one can derive the rule of inverse probability, also known as *Bayes' rule*:

$$\begin{aligned} p(Y = y \,|\, X = x) &= \frac{p(X = x \,|\, Y = y)\, p(Y = y)}{p(X = x)} \\ &= \frac{p(X = x \,|\, Y = y)\, p(Y = y)}{\sum_{y \in \mathcal{Y}} p(X = x,\, Y = y)}. \end{aligned} \tag{2.3}$$

If the joint distribution $p(X, Y)$ factorises into the product $p(X) \times p(Y)$ then the random variables X and Y are independent. In this case, the conditional distribution $p(X \,|\, Y)$ is equal to the marginal distribution $p(X)$.

2.1.1 Probability density and mass functions

We often use functions to describe the distributions of random variables. *Continuous* random variables are often described by a *probability density function* (PDF). The PDF describes the relative probabilities of the values that a random variable can take. A function f is the PDF of a random variable X if the probability that the value x of the random variable will fall in the range (a, b) is:

$$p(a \leq X \leq b) = \int_a^b f(x)\, \mathrm{d}x. \tag{2.4}$$

If f is the PDF of X then we say that X is distributed according to f. A shorthand for writing this statement is: $X \sim f(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are parameters controlling the density of X.

A *probability mass function* (PMF) is the equivalent of a PDF for *discrete* random variables. A function f is the PMF of a random variable X if $P(X = c) = f(c)$.

In this dissertation, a number of probability distributions are used frequently. The Gaussian (or normal) distribution is used to model continuous random variables. The categorical distribution is used for discrete random variables.

The Wishart is a distribution over prevision matrices. The Dirichlet distribution is a distribution over finite discrete probability distributions. Appendix B provides the probability mass / density functions, and description of these distributions.

2.2 Machine Learning and Inference

Consider a dataset of N entries each of which describes a house with D features, represented as an $N \times D$ matrix $\mathbf{X}$. Additionally, let us assume that we have the price of each of these houses – we can represent this information as an N-dimensional vector $\mathbf{y}$. We might want to build a model that predicts the price of a house based on the D features in our dataset.

The problem above is a typical example of a *supervised* learning problem. In a supervised learning setting we wish to learn a function from inputs $\mathbf{x}$ to outputs $\hat{y}$, parametrised by $\boldsymbol{\theta}$, such that:

1. the difference between $\hat{y}$ and y is minimised, and
2. $\hat{y}$ is generalisable to new examples outside of $\mathbf{X}$.

In this case, learning is inferring both the parameters $\boldsymbol{\theta}$ and the family of functions for $\hat{y}$ that best achieve these two goals. The families of functions to search for $\hat{y}$, the metric by which we measure the closeness of $\hat{y}$ and y, and the method for updating the parameters $\boldsymbol{\theta}$, are choices that must be made by us.

Alternatively, rather than predicting the price of a house, we might wish to model the distribution of house features. We might wish to know, for example, the distribution of bed rooms and how that relates to the distribution of bathrooms. Density estimation is a typical *unsupervised* learning problem. Other unsupervised learning problems include anomaly detection and clustering. The difference between unsupervised and supervised learning problems is that in the supervised setting we are given labels. However, both problems can be viewed as function estimation.

2.2.1 Learning as inference

There are a number of methods for finding the parameters of a model. One simple (but often problematic) approach to finding the best parameters is to *maximise the likelihood* of the dataset $\mathbf{X}$ given the parameters $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{X} \,|\, \boldsymbol{\theta}) \tag{2.5}$$

where $p(\mathbf{X} \,|\, \boldsymbol{\theta})$ is called the *likelihood*. We wish to find the parameters of the model that maximise the probability of observing the data. In practice, to avoid numerical underflow, the *log-likelihood* is computed instead.

$$\log p(\mathbf{X} \,|\, \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}_n \,|\, \boldsymbol{\theta})\,. \tag{2.6}$$

Maximising the log-likelihood results in the same set of parameters as maximising the standard likelihood because the logarithm is a convex function and therefore

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\mathbf{X} \,|\, \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{X} \,|\, \boldsymbol{\theta})\,. \tag{2.7}$$

While the maximum likelihood estimation (MLE) approach can give good results, it is not perfect. MLE tends to *over-fit* to the training data. To illustrate this issue, let us consider another example. Suppose we wanted to infer the probability of flipping a coin and it landing on tails. We could flip a coin 10 times and observe the results. However, if we were to observe 10 heads, and we used MLE to infer the probability of flipping a tails, we would arrive at 0%. This result does not seem correct – based on our intuitions about how a coin flip works it is surprising that it is impossible to get a tails. Things get more dubious when we consider that we would arrive at the same conclusion if we only flipped a single coin and observed a heads.

One solution to the problem of over-fitting with MLE is to use *Bayesian inference*. To describe Bayesian inference, let us take another look at Bayes'

rule applied to the dataset $\mathbf{X}$ and the parameters $\boldsymbol{\theta}$:

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

where $p(\mathbf{X} \mid \boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta})$ is called the prior, and $p(\mathbf{X})$ is called the *evidence* or *marginal likelihood*. The prior describes our beliefs about what the parameters of the model should be. For example we could use it to encode our belief that a coin should have a non-zero probability of landing on either side. The evidence describes how probable the observed data is according to our model. Here the model refers not to a specific instance of the model with a particular set of parameters, but rather the class of models for which we are trying to determine the parameters. The posterior $p(\boldsymbol{\theta} \mid \mathbf{X})$ it is the probability of the parameters given the data, i.e. what we should believe the parameters of the model to be after observing the data. The key idea of Bayesian inference is that we can use Bayes' rule to update our belief about the parameters of our model having seen the data.

Given the posterior distribution of the parameters, we can determine the posterior predictive distribution of a test point $\mathbf{x}^*$:

$$p(\mathbf{x}^* \mid \mathbf{X}) = \int_{\boldsymbol{\theta}} p(\mathbf{x}^* \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathbf{X})\ \mathrm{d}\boldsymbol{\theta}. \tag{2.9}$$

The prior serves two main purposes in Bayesian inference. Firstly, it is a way for expert knowledge to be incorporated into the inference. Secondly, it serves as a form of regularisation that prevents over-fitting – solving the main issue with MLE.

Unfortunately, the integrals in the calculation of the evidence in equations (2.8) and (2.9) are often intractable to compute. There are a number of approaches that address this problem. For example, the use of *conjugate priors* – which result in a posterior that is from the same family as the prior – can sometimes result in easily computable closed forms for the integral.

Another way of dealing with intractable integrals is to use approximation

techniques. Broadly speaking, there are three main approaches to approximating intractable integrals (Steinruecken and Iwata, 2013).

1. Sample from the distributions by replacing the integrals with summations, for example Monte-Carlo methods.
2. Replace the difficult integrals with integrals that are easier to compute and (hopefully) give similar results, for example Variational Bayes (Section 2.2.6).
3. Use non-Bayesian methods, for example MAP estimation (Section 2.2.4).

Each of these methods has pros and cons.

2.2.2 Graphical models

Some probabilistic models can be represented in a graphical form. One of the benefits of a graphical representation is that the conditional dependency structure of models can be expressed in a simple visual form. For example, consider the following joint distribution over a few variables:

$$p(A,\, B,\, C,\, D,\, E) = p(A \,|\, B,\, C)\, p(B \,|\, C)\, p(C \,|\, D,\, E)\, p(D \,|\, E)\,. \qquad (2.10)$$

This joint distribution can be represented as a graph, where the nodes are random variables and the edges indicate dependence between those random variables, as shown in Figure 2.1. This figure depicts a particular type of probabilistic graphical model – a *Bayesian network*. Bayesian networks are a subset of graphical models and are represented by directed acyclic graphs (DAGs).

For a slightly more complicated example consider the following distribution:

$$p(\{A_n\},\, B \,|\, c,\, d) = p(B \,|\, c) \prod_{n=1}^{N} p(A_n \,|\, B,\, d) \qquad (2.11)$$

and its corresponding graphical model shown in Figure 2.2. This figure in-

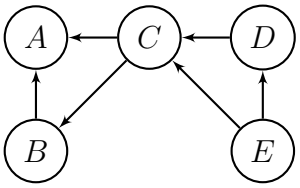


Figure 2.1: Graphical representation of a simple model described in equation (2.10).

troduces a few more devices for describing probabilistic models. There are three kinds of nodes:

1. observed variables, which are depicted using shaded circles,
2. unobserved variables, which are depicted with unfilled circles, and
3. hyper-parameters, which are depicted without circles.

The unobserved variables can be further divided into latent variables and parameters. Some parts of the graph are surrounded by "plates", which indicate that the surrounded subgraph is repeated a certain number of times.

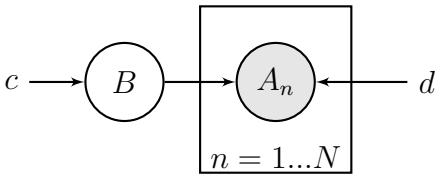


Figure 2.2: Graphical representation of a more complex model described in equation (2.11).

2.2.3 Mixture models

Mixture models are a type of probabilistic model which leverage discrete latent variables to combine simple probability distributions into complex marginal distributions. For example, the *Gaussian mixture model* (GMM) is a combination of Gaussian distributions. To motivate why we might want to have this more expressive distribution consider the example shown in Fig-

ure 2.3. A single Gaussian is unable to capture multi-modal features in data, however, GMMs are able to do so.

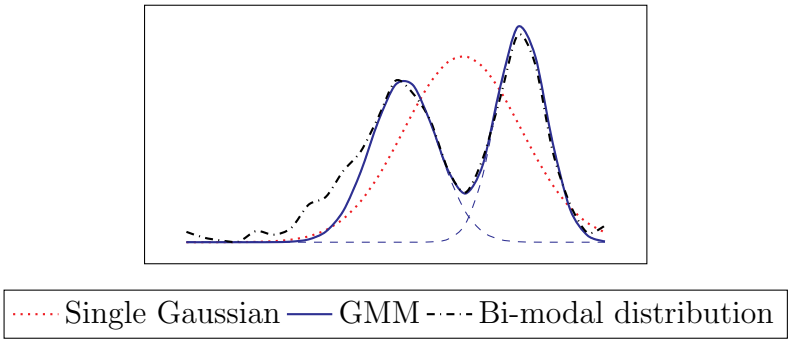


Figure 2.3: Modelling a bimodal distribution. The black dash-dotted curve is a bimodal distribution which we wish to model. The red dotted curve is the fit resulting from a single Gaussian. The solid blue curve is the superposition of two Gaussian components (dashed blue lines) of a GMM.

The general form of a mixture with K components is:

$$p(\mathbf{x} \,|\, \{\boldsymbol{\theta}_k\}) = \sum_{k=1}^{K} \pi_k \, p_k \left(\mathbf{x} \,|\, \boldsymbol{\theta}_k\right) \tag{2.12}$$

where π_k are the weights between 0 and 1 that sum to unity, $p_k(\cdot)$ is the probability distribution for the kth component with parameters $\boldsymbol{\theta}_k$, and $\{\boldsymbol{\theta}_k\}$ is the set containing $\boldsymbol{\theta}_k$ for all k. The mixture weights π_k can be interpreted as the probability that $\mathbf{x}$ is drawn from component k. In a GMM, the p_k are Gaussian distributions, and the parameters $\boldsymbol{\theta}_k$ are the means $\boldsymbol{\mu}_k$ and the covariance matrices $\boldsymbol{\Sigma}_k$. Mixture models can be conveniently represented using Bayesian networks, as show in Figure 2.4.

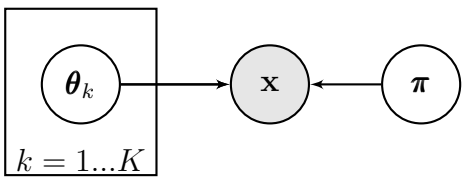


Figure 2.4: Graphical representation of a mixture model as described in equation (2.12).

2.2.4 MAP estimation

Maximum a posteriori (MAP) estimation can be seen as a compromise between maximum likelihood estimation and Bayesian inference. MAP estimation incorporates a prior into the maximum likelihood estimate, however, MAP estimation is not a Bayesian method because it results in a point estimate of the parameters (as opposed to a distribution over parameters as Bayesians would prefer). Note that in Bayes' rule (2.8) the evidence is effectively just a normalising constant. Bayesian inference tells us that the distribution for the parameters after seeing the data is *proportional* to the product of the likelihood, and our prior belief about the distribution of the parameters:

$$p(\boldsymbol{\theta} \,|\, \mathbf{X}) \propto p(\mathbf{X} \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})\,. \tag{2.13}$$

The most probable value of the parameters $\boldsymbol{\theta}$ according to the posterior distribution can be calculated: $\boldsymbol{\theta}_{\mathrm{MAP}} = \operatorname{argmax}_{\boldsymbol{\theta}} p(\mathbf{X} \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$.

Because this calculation does not depend on the evidence, that source of intractable integrals is avoided and because the MAP estimate is a point estimate, the integral in equation (2.9) is not required.

2.2.5 Expectation maximisation

Fitting mixture models, or any class of models with latent variables, is not as straight forward as for models that only have parameters and observed variables. *Expectation maximisation* (EM) is an iterative method for finding MLE and MAP estimates. The EM algorithm starts with guesses for the parameters and then refines those guesses by alternating between two steps:

1. The E-step – which uses the current version of the parameters, $\boldsymbol{\theta}_{\mathrm{old}}$, to find the posterior distribution for the latent variables, and then uses this posterior to find the expectation of the log-likelihood, with respect to the latent variables, given some parameters $\boldsymbol{\theta}$: $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\mathrm{old}})$.
2. The M-step – which maximises $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\mathrm{old}})$ to find the new parameters:

$$\boldsymbol{\theta}_{\text{new}} = \text{argmax}_{\boldsymbol{\theta}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}).$$

These steps are repeated until convergence. Each iteration of the EM algorithm is guaranteed not to decrease the likelihood of the unobserved data (Bishop, 2006, ch 9). In the case of a MAP estimate, the E-step remains the same and the M-step maximises $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}) + \log p(\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ is the prior on the parameters.

A remark

In the discussion above, we have considered the use of the EM algorithm specifically for dealing with unobserved latent variables in the model. However, the EM algorithm is applicable to a much broader set of problems. Of particular interest to us is the use of the EM algorithm for dealing with missing values in a dataset.

2.2.6 Variational Bayes

As mentioned in Section 2.2.1, Variational Bayes (VB) is a method for approximating intractable integrals in Bayesian inference. Broadly speaking, variational methods make use of principles from the *calculus of variations* to perform optimisation over functions. By restricting the classes of functions being optimised, to those classes that are tractable to compute, approximate solutions to intractable integrals can be found.

Like the EM algorithm, VB is an iterative method which successively converges on the optimum parameters. However, unlike EM, VB does not produce point estimates for parameters but rather estimates for the posterior distributions of the parameters.

To better explain VB let use consider a probabilistic model $p(\mathbf{Z}, \mathbf{X})$ with observed variables $\mathbf{X}$, latent variables $\mathbf{Z}$, and a posterior distribution for the latent variables $p(\mathbf{Z} \mid \mathbf{X})$. Let us assume, as is often the case, that the posterior is intractable. To approximate this posterior using VB we define a

family of distributions we wish to optimise over: $q(\mathbf{Z})$.

We can measure the difference between the posterior $p(\mathbf{Z} \mid \mathbf{X})$ and the *variational distribution* $q(\mathbf{Z})$ using the Kullback–Leibler (KL) divergence:

$$D_{\mathrm{KL}}(q \,||\, p) = \int_{\mathbf{X}} q(\mathbf{Z}) \log \frac{q(\mathbf{Z})}{p(\mathbf{Z} \mid \mathbf{X})} \, \mathrm{d}\mathbf{X}. \tag{2.14}$$

$D_{\mathrm{KL}}(q \,||\, p)$ can be used as an optimisation objective for methods such as gradient descent or block coordinate descent, because minimising $D_{\mathrm{KL}}(q \,||\, p)$ reduces the difference between the variational and true posterior distributions (the approximation error).

However, there is a problem with minimising $D_{\mathrm{KL}}(q \,||\, p)$ as defined in equation (2.14) – the reason that we are approximating $p(\mathbf{Z} \mid \mathbf{X})$ is that it is intractable to compute. How can we minimise a quantity that we cannot compute? We can rewrite the RHS of equation (2.14) as follows:

$$D_{\mathrm{KL}}(q \,||\, p) = \mathbb{E}_{\mathbf{Z}}\big[\, \log q(\mathbf{Z}) - \log p(\mathbf{X},\, \mathbf{Z}) \,\big] + \log p(\mathbf{X}) \,. \tag{2.15}$$

Noting that the KL divergence is non-negative ($D_{\mathrm{KL}} \geq 0$):

$$\log p(\mathbf{X}) \geq \mathbb{E}_{\mathbf{Z}}\big[\, \log q(\mathbf{Z}) - \log p(\mathbf{X},\, \mathbf{Z}) \,\big] \tag{2.16}$$

where the expectation is known as the *evidence lower bound* (ELBO). Maximising the ELBO is equivalent to minimising the KL divergence, and as the ELBO does not contain any intractable integrals we are able to perform this optimisation.

Variational Bayes' requires choosing a variational distribution $q(\mathbf{Z})$. We want to choose a family of distributions that expressive enough that it can be close to $p(\mathbf{Z} \mid \mathbf{X})$, and that is also computationally tractable. A common approach is to choose a *factorised distribution*. With this approach we assume that the

distribution of each of the latent variables is independent of the others:

$$q(\mathbf{Z}) = \prod_{n=1}^{N} q_n(\mathbf{z}_n) \tag{2.17}$$

where N is the number of latent variables. Under the factorisation assumption, the optimal updates are analytically computable and can be updated with block coordinate descent. The optimal updates are:

$$\log q^*(\mathbf{z}_n) = \mathbb{E}_{m \neq n}\big[\log p(\mathbf{X}, \mathbf{Z})\big] + c \tag{2.18}$$

where c is a constant that can be recovered by inspection.

Note that both Variational Bayes and the EM-algorithm can be used in cases where there are missing values in the dataset.

2.3 Missing Data

Consider a dataset with N entries of D-dimensional data, represented as an $N \times D$ matrix $\mathbf{X}$. Any of the elements x_{nd} could be missing. In order to describe which elements are missing we use a mask matrix $\mathbf{M}$, with the same dimensions as $\mathbf{X}$. If a value x_{ij} is missing then $m_{ij} = 1$, otherwise $m_{ij} = 0$. For any row $\mathbf{x}_n$ the set of indices for missing values is denoted as $\mathbf{m}$, and set of indices for values that are non-missing is denoted as $\mathbf{o}$. Using this notation, $\mathbf{x}_{n\mathbf{o}}$ and $\mathbf{x}_{n\mathbf{m}}$ are vectors containing the non-missing and missing elements, respectively. $\mathbf{X}_{:\mathbf{o}}$ and $\mathbf{X}_{:\mathbf{m}}$ denote all of the non-missing and missing values in the dataset, respectively.

2.3.1 Missing data taxonomy

The nature of the missingness is an important consideration when handling missing data. Little and Rubin (2002, ch 1) classify missing data based on

the conditional distribution $p(\mathbf{M} \mid \mathbf{X})$. They define three types of missing data:

1. *Missing completely at random* (MCAR): $p(\mathbf{M} \mid \mathbf{X}) = p(\mathbf{M})$, the pattern of missing does not depend at all on the data.
2. *Missing at random* (MAR): $p(\mathbf{M} \mid \mathbf{X}) = p(\mathbf{M} \mid \mathbf{X}_{:\mathbf{o}})$, the pattern of missingness may depend on the observed values but not the missing values. Figure 1.1 shows an example of this case.
3. *Not missing at random* (NMAR): the pattern of missingness may depend on any values including those that are missing. This is the case e.g. when values have been censored, or when a sensor doesn't report values in a certain range.

Note that NMAR is a stronger condition than MAR, and MAR is a stronger condition than MCAR.

The MCAR and MAR assumptions are almost always unrealistic. However, using a method that assumes MAR can still give reasonable results. Furthermore, when handling missing data, it is often convenient to make the MAR assumption since there are no general approaches for handling NMAR data (Ghahramani and Jordan, 1996).

2.3.2 Handling missing data

There are a number of approaches for dealing with missing data. Possibly the most simple approach, called *listwise deletion*, is to discard any entries with missing values. However, as discussed in Chapter 1, this is not usually a good idea because useful data is thrown away. For example, training deep neural network models is a data-intensive task that requires as much training data as possible (Jordan and Mitchell, 2015).

A second approach is to use an algorithm that can automatically handle missing values. Many algorithms are either indifferent to the presence of missing values or are specifically designed to account for the missingness in some

way. This approach has the advantage that users of the algorithm need not concern themselves with the details of *how* to repair the dataset. Algorithms that handle missing values directly can make use of the information about *which* values are missing. This information is not available to algorithms that operate on datasets that have been repaired by imputation or deletion. However, not all algorithms have this capability.

A third approach is to use an imputation algorithm to repair the dataset before proceeding with the task at hand. This approach has the advantage that *any* algorithm can then be used on the repaired dataset, even one that fails on incomplete datasets.

Simple imputation methods

To give a better intuition of how imputation might be performed, let us consider a few simple methods for imputing missing values, i.e. methods for predicting $\mathbf{X}_{:\mathbf{m}}$ given $\mathbf{X}_{:\mathbf{o}}$:

- *Hot-decking*: replacing all missing values in an entry with the corresponding value in a similar entry. It is also possible to use a random entry rather than a similar one.
- *Mean or mode substitution*: replacing all missing values with the mean (or mode for categorical variables) of the corresponding variable. The assumption that all of the missing values are equal to the mean of variable is probably unrealistic. Note that mean imputation will not change the mean of which could be a useful property in some cases.
- *Regression imputation*: train a machine learning model to predict missing values in an entry given the values that are not missing. This approach addresses the a problem with the above two methods: ignoring correlation between variables. However, this method can incorrectly increase the correlation between two variables. This issue can be somewhat corrected for by adding noise to the imputed value, based on the variance of the regression.

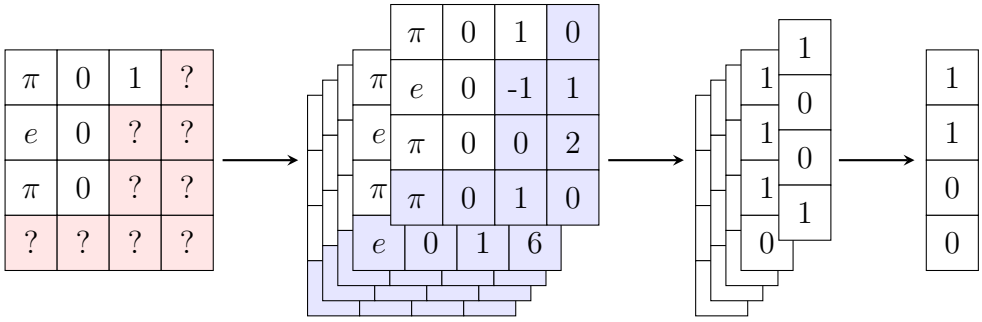


Figure 2.5: A pictorial description of the multiple imputation pipeline. An incomplete dataset is first imputed multiple times. Each of the imputed datasets might have different values in the places where data is missing. Analysis is then performed on each of the imputed datasets individually. The results of the analyses are then pooled to give a final result for which we can now have an estimate of the variance.

Multiple imputation

A problem with the methods described above, and in fact any imputation method, is that any analyses performed on the imputed dataset can be overconfident. This overconfidence is the result of treating imputed values the same as values which are not missing. But there should be be a difference: we should be less confident about the imputed values than the known values. Unfortunately, after a dataset has been repaired, subsequent analyses are unable to take this distinction into account.

Rubin (1987, 1996) proposed *multiple imputation* to address this problem. The idea is to stochastically impute a dataset multiple times, using different random numbers during each imputation, and then perform an analysis on each of the imputed datasets, aggregating the results. This way the uncertainty in each individual imputation is included in the final analysis. An example of this process is shown in Figure 2.5.

Multiple imputation is flexible in that many imputation methods can be used to repair each dataset, and the results can be aggregated in a number of ways. For example, if the analysis being performed in Figure 2.5 is *classification*, then the results could be aggregated by taking the mode. However,

if the analysis being performed is *regression* then the mean could be taken instead. The main constraint on the imputation method is that it must be non-deterministic so that multiple samples can be drawn from it.

Chapter 3

Related Work

The problem of handling missing data began to see a lot of attention in the 1970s (Little and Rubin, 2002). The EM algorithm was published by Dempster et al. (1977) based on the theory of Baum et al. (1970), Sundberg (1974), and Orchard and Woodbury (1976). Since then the problem of how to handle missing data has continued to garner attention. A more recent example of an approach to handling missing data is collaborative filtering for recommender systems (Ricci et al., 2011).

3.1 Methods for learning despite missing data

Ghahramani and Jordan (1996) use the EM algorithm to estimate the density (or mass) of missing data given the non-missing data. They give E and M-steps for Gaussian and Binomial mixture models for learning the density and mass of real-valued and binary discrete valued variables. In their method, the EM algorithm is used to both learn the mixture components as well as to handle missing data. Their method is able to perform both classification and regression tasks in the supervised setting as well as other tasks in the unsupervised setting. However, because their approach uses the EM algorithm to give maximum-likelihood estimates of the parameters of the model, it is

prone to over-fitting. Another problem with this method is that it cannot correctly handle data that is NMAR, because it does not specifically model the pattern of missingness.

Melchior and Goulding (2016) propose using the EM algorithm for estimating the density of missing data. Their method is specifically aimed at handling "truncated" datasets, in which whole entries are missing, as opposed to datasets where some elements of any given entries can be missing. Delalleau et al. (2018) propose modifications to the approach of Ghahramani and Jordan (1996) to improve the scalability to large data-sets.

Smola et al. (2005) extend Gaussian Processes and Support Vector Machines for making predictions on datasets with missing values. Their work is aimed at solving a supervised learning problem in the presence of missing data. They propose two methods, one for the case in which part of the input dataset $\mathbf{X}$ is missing, and one for the case in which some of the labels $\mathbf{y}$ are missing.

The above methods all assume that the dataset is either MCAR or MAR, because they do not model the mechanism for missingness. One domain where the MAR and MCAR assumptions are invalid is collaborative filtering for recommender systems: in this setting, the entries in the dataset often contain user ratings for products. However, not every user gives a rating for every product. These unrated entries can be interpreted as missing values. However, it is incorrect to assume that the data is missing at random, because the absence of a user's rating for particular product gives information about that user's likes and dislikes, as shown by (Marlin et al., 2012). To deal with NMAR data, Hernández-Lobato et al. (2014) specifically model the missing data and perform approximate inference on this model using expectation propagation and variational inference.

Faes et al. (2010) use variational inference to learn models describing the missing data when performing regression on data in which the labels are non-missing but some of the predictors are missing. As a result of explicitly modelling the missing data mechanism, they did not have to make the MCAR

or MAR assumptions. Their method achieved high accuracy for the regression task, but was not successful in learning the parameters of the missing data model.

Williams and Nash (2018) extend variational auto-encoders to handle missing data. They note that there is a non-trivial dependence on the pattern of missing data.

3.2 Imputation methods

There is a wide range of different approaches to imputing missing data. Some of the most popular methods for imputation are MissPaLasso (Städler et al., 2014), KNNimpute (Troyanskaya et al., 2001), Multiple Imputation by Chained Equations (MICE) (Van Buuren and Groothuis-Oudshoorn, 2011; Van Buuren and Oudshoorn, 1999) and MissForest (Stekhoven and Bühlmann, 2012).

However, many of these methods leave something to be desired. KNNimpute, MissPaLasso, and many other methods, assume that the data is continuous. Various methods, including KNNimpute and MICE, require the user to tune a number of parameters in order to achieve good results. These parameters are typically application-dependent and require domain knowledge to appropriately specify.

MissForest uses random forests (Breiman, 2001) to perform regression imputation with an iterative update scheme. Random forests are used due to their ability to scale to high dimensional datasets, robustness to noise, ability to handle mixed type datasets, and ability to learn non-linear relationships.

Stekhoven and Bühlmann (2012) benchmark MissForest, MICE, KNNImpute, and MissPaLasso on a range of different datasets with varying proportions of artificially created MCAR data. They find that MissForrest outperforms all of the other methods they tested. The strong performance of MissForest, in addition to the fact that it can handle both categorical and

continuous data and does not require tuning parameters, indicate that MissForest could be a good choice for some imputation problems.

Chapter 4

Design and Implementation

This chapter describes the design and implementation details of the AutoImpute, my tool for missing data imputation.

4.1 Overview of AutoImpute

AutoImpute is a command line application for imputing missing data. The implementation is written in Python 3, and uses the NumPy and SciPy scientific computing libraries. All algorithms were written from scratch without using any high-level machine learning libraries such as scikit-learn.

4.1.1 Features

AutoImpute offers a number of features for imputing missing data. For example it includes the following imputation models:

- mean imputation (described in Section 4.2),
- single Gaussian imputation (described in Section 4.3),
- GMM imputation (described in Section 4.4), and

- Dirichlet process imputation (described in Section 4.5).

The parameters of the single Gaussian and GMM models can be inferred using either MLE or MAP estimation. Additionally, Pure Bayesian inference is supported for the single Gaussian. Bayesian inference allows automatic model comparison to be performed between the DP model and the single Gaussian model. This model comparison is described in Section 4.6.

The application allows the user to specify the names of input and output files as well as the format of the input file – whether it has a header, what kind of delimiter it uses, and the indicator for missing values.

The user can choose between different imputation policies: using the *mode* of the missing data distribution to impute the most likely missing data, or *sampling* from the missing data distribution to create multiple repaired files i.e. performing multiple imputation.

The user is also able to control a number of other settings such as:

- a random seed for reproducibility,
- the formatting for floating point numbers, and
- the verbosity of the output.

4.1.2 User interface

The tool can be used by running this command:

```
python3 auto_impute.py [-h] [-v] [-d D] [-hd HEADER]
[-i INDICATOR] [-rs RAND_SEED] [-o FILE_NAME] [-t TEST_FILE]
[-fmt FORMAT] [-e EPSILON] [-n MAX_ITERS] [-k NUM_COMP]
[-a ASSIGNMENTS] [-mle] [-s SAMPLE | -m]
[-mi | -sg | -gmm | -dp | -mix] file
```

The arguments are described as follows.

Positional arguments

`file`	The name of the file to repair.

Optional arguments

`-h`	`--help`	Show a help message and then exit.
`-v`	`--verbose`	Increase the verbosity of the output.
`-d D`	`--delimiter D`	Specify the file delimiter (default: ",").
`-hd`	`--header`	Specify that the first row of the file contains column names.
`-i IND`	`--indicator IND`	Specify the missing value indicator in the file (default: "" = empty string).
`-rs RS`	`--rand_seed RS`	Specify a random seed for reproducibility (default: none).
`-o FNM`	`--file_name FNM`	File to write repaired files to (if unspecified repaired files are written to `stdout`).
`-t FNM`	`--test_file FNM`	File containing the ground truth so that error metrics can be reported.
`-fmt F`	`--format F`	Python format string for printing floating point numbers (default: "%.5g").
`-e EPS`	`--epsilon EPS`	Stopping criterion: if $\mathrm{LL}_{\mathrm{new}} - \mathrm{LL}_{\mathrm{old}} <$ `EPS` then stop training (default: `EPS` $= 0.01$).
`-n MAX`	`--max_iters MAX`	Maximum number of iterations for VB and EM (default: `MAX` $= 10$).
`-k NUM`	`--num_comp NUM`	The number of mixture components for GMM (default: `K` $= 3$).

`-a A`	`--assignments A`	Optional data type assignments for the DP-single Gaussian mixture: "d" for discrete and "c" for continuous. For example use "dddrrr" for 3 discrete columns followed by 3 continuous columns.
`-mle`	`--ml_estimation`	Use MLE rather than the default MAP estimation.
`-s NUM`	`--sample NUM`	Repair the file `SAMPLE` times by sampling from the missing data distribution.
`-m`	`--mode`	Use the *mode* (value with the highest probability) of the missing data distribution to repair file (default option).
`-mi`	`--mean_impute`	Use mean imputation to repair the file.
`-sg`	`--single_gaussian`	Use a single Gaussian distribution to model the missing data and repair the file.
`-gmm`	`--gaussian_mix`	Use a Gaussian mixture model to repair the file.
`-dp`	`--dirichlet_proc`	Use a Dirichlet process to repair the file.
`-mix`	`--sg_dp_mix`	Use a combination of DPs and single Gaussian distributions to repair the file, using automatic model selection.

4.1.3 Software engineering details

The code has been written with extensibility in mind – it should be easy to add additional modules. Furthermore, the code has extensive commentary and documentation throughout; the idea that *code is read more than it is written* has been kept in mind throughout development.

AutoImpute has a small suite of unit-tests for each of the models, summarised in Table 4.3. The unit-tests have been written as black-box tests i.e. they treat the modules as black boxes and are only aware of the inputs and

outputs. This approach has been used to allow the implementation of each module to be easy to change. More information about the unit-tests can be found in Appendix C.

Table 4.3: Summary of unit-tests for AutoImpute.

Module	**Number of tests**
Dirichlet process	6
Mean imputation	9
Single Gaussian	12
GMM	15
Total	42

The testing has been carried out in the testing environment described in Table 4.4.

Table 4.4: Software specifications for the testing environment.

Software	**Version**
Python	3.5.2 [GCC 5.4.0 20160609]
SciPy	1.0.1
NumPy	1.14.2
Ubuntu	Ubuntu 16.04.4 LTS
Linux kernel	4.13.0-43-generic

Finally, `git` was used for version control so that the tracking of changes and future collaboration are easier. Development was carried out on a `development` branch, and commits that passed the unit tests were merged to the `master` branch.

4.2 Mean Imputation

The mean imputation algorithm simply replaces any missing elements in x_{nd} the D-dimensional dataset $\mathbf{X}$ with the mean of the column $\mathbf{x}_{:,d}$:

$$p(x_{nd}) = \begin{cases} 1 & \text{if } x_{nd} = \frac{1}{N}\sum_{i=1}^{N} x_{id} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Note that with this model, each of the dimensions of the dataset are assumed to be independent. As a result, imputing a D-dimensional dataset is equivalent to imputing D one-dimensional datasets.

The benefit of this model is that it is simple to implement and very fast to run. However, it has a number of issues: firstly, this model will produce absurd imputations for discrete data. Consider a one-dimensional dataset containing only 1s and 0s. The mean of this dataset will be some number between 0 and 1. However, in the original distribution for the data there is mass on only 2 points: 1 and 0. This is because this model is only applicable when the underlying data type has support for the mean operation (i.e. it has support for addition and scalar division). Secondly, because we have assumed that each dimension of the dataset is independent, this model throws away information about the correlation in multivariate datasets.

4.3 Single Gaussian

Modelling the distribution of a D-dimensional dataset $\mathbf{X}$ using a single Gaussian distribution addresses the issue of throwing away information about correlations. An entry $\mathbf{x}_n$ sampled from a Gaussian distribution has a likelihood:

$$p(\mathbf{x}_n \,|\, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_n \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma})\,. \tag{4.2}$$

If none of the elements of the dataset $\mathbf{X}$ are missing then the parameters can

be estimated as:

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{4.3}$$

$$\boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T. \tag{4.4}$$

However, if $\mathbf{X}$ contains missing elements, then the estimations above will be biased. The estimate of the mean will be incorrect, as is illustrated in Figure 1.1, and the covariance estimate will be over-confident. See Appendix B.1 for details on the Gaussian distribution.

4.3.1 EM for MLE

To deal with the issues caused by missing values, we can use the EM algorithm (described in Section 2.2.5) to iteratively determine the maximum likelihood estimate of the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of the Gaussian distribution. The E-step is simply replacing the missing values of the dataset with their expectations (Delalleau et al., 2018). For any data point in our dataset $\mathbf{x}_n$, the probability of the missing data $\mathbf{x}_{n\mathbf{m}}$ given the non-missing data $\mathbf{x}_{n\mathbf{o}}$ is:

$$\begin{aligned} p(\mathbf{x}_{n\mathbf{m}} \,|\, \mathbf{x}_{n\mathbf{o}}, \boldsymbol{\theta}) = \mathcal{N}\big(\mathbf{x}_{n\mathbf{m}} \,\big|\, \boldsymbol{\mu}_\mathbf{m} + \boldsymbol{\Sigma}_{\mathbf{mo}}{\boldsymbol{\Sigma}_{\mathbf{oo}}}^{-1}(\mathbf{x}_{n\mathbf{o}} - \boldsymbol{\mu}_\mathbf{o}), \\ \boldsymbol{\Sigma}_{\mathbf{mm}} - \boldsymbol{\Sigma}_{\mathbf{mo}}{\boldsymbol{\Sigma}_{\mathbf{oo}}}^{-1}\boldsymbol{\Sigma}_{\mathbf{om}}\big). \end{aligned} \tag{4.5}$$

The expected value of the missing data is $\boldsymbol{\mu}_\mathbf{m} + \boldsymbol{\Sigma}_{\mathbf{mo}}{\boldsymbol{\Sigma}_{\mathbf{oo}}}^{-1}(\mathbf{x}_{n\mathbf{o}} - \boldsymbol{\mu}_\mathbf{o})$. Let us define a repaired data point $\hat{\mathbf{x}}_n$ as $\hat{\mathbf{x}}_{n\mathbf{o}} = \mathbf{x}_{n\mathbf{o}}$ and $\hat{\mathbf{x}}_{n\mathbf{m}} = \boldsymbol{\mu}_\mathbf{m} + \boldsymbol{\Sigma}_{\mathbf{mo}}{\boldsymbol{\Sigma}_{\mathbf{oo}}}^{-1}(\mathbf{x}_{n\mathbf{o}} - \boldsymbol{\mu}_\mathbf{o})$. In the M-step we calculate (Delalleau et al., 2018):

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\hat{\mathbf{x}}_n \tag{4.6}$$

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}\big[(\hat{\mathbf{x}}_n - \boldsymbol{\mu})^T(\hat{\mathbf{x}}_n - \boldsymbol{\mu})\big] + \mathbf{C} \tag{4.7}$$

where:

$$\mathbf{C_{mm}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{\Sigma_{mm}} - \mathbf{\Sigma_{mo}}\mathbf{\Sigma_{oo}}^{-1}\mathbf{\Sigma_{om}}. \tag{4.8}$$

The notation has been used slightly unconventionally in equation (4.8): $\mathbf{C_{mm}}$ refers to the elements of $\mathbf{C}$ corresponding to the missing elements of $\mathbf{x}_n$ at each summation iteration. Note that $\mathbf{C}$ is calculated using the previous estimate for $\mathbf{\Sigma}$.

4.3.2 EM for MAP estimation

To reduce over-fitting, or to encode prior knowledge about the dataset into the estimate of the parameters, we might wish to use a MAP estimate of the parameters rather than a MLE. We will place a Gaussian-Wishart prior on the parameters $\boldsymbol{\mu}$ and $\mathbf{\Lambda}$:

$$p(\boldsymbol{\mu}, \mathbf{\Lambda}) = \mathcal{N}\big(\boldsymbol{\mu} \,\big|\, \mathbf{m}_0, (\beta_0\mathbf{\Lambda})^{-1}\big)\, \mathcal{W}(\mathbf{\Lambda} \,|\, \mathbf{W}_0, \nu_0) \tag{4.9}$$

where $\mathbf{m}_0$ is the prior mean, β_0 is the mean precision prior, $\mathbf{W}_0^{-1}$ is the covariance prior, ν_0 is the degrees-of-freedom prior, and $\mathcal{W}(\cdot)$ is the Wishart distribution (see Appendix B.3). The Gaussian-Wishart distribution is used because it is a conjugate prior (see Section 2.2.1) for the multivariate Gaussian distribution with unknown mean $\boldsymbol{\mu}$ and precision $\mathbf{\Lambda}$.

Assuming the dataset $\mathbf{X}$ *does not* contain any missing elements, the MAP estimate of the parameters is (Murphy, 2007):

$$\hat{\boldsymbol{\mu}} = \frac{\beta_0\mathbf{m}_0 + \sum_{n=1}^{N}\mathbf{x}_n}{\beta_0 + N} \tag{4.10}$$

$$\hat{\mathbf{\Sigma}} = \frac{\sum_{n=1}^{N}\left[(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T(\mathbf{x}_n - \hat{\boldsymbol{\mu}})\right] + \beta_0(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}})^T(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}) + \mathbf{W}_0^{-1}}{\nu - D}. \tag{4.11}$$

If the dataset *does* contain missing elements, we can again use the EM algorithm to iteratively determine the MAP estimate of the parameters $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$. The E-step is the same as the E-step for MLE. Note, however, that we are

now using the MAP estimates to fill in the missing values. Recall the definition of a repaired entry $\hat{\mathbf{x}}_n$: $\hat{\mathbf{x}}_{n\mathbf{o}} = \mathbf{x}_{n\mathbf{o}}$ and $\hat{\mathbf{x}}_{n\mathbf{m}} = \boldsymbol{\mu}_{\mathbf{m}} + \boldsymbol{\Sigma}_{\mathbf{mo}}\boldsymbol{\Sigma}_{\mathbf{oo}}{}^{-1}(\mathbf{x}_{n\mathbf{o}} - \boldsymbol{\mu}_{\mathbf{o}})$. Now in the M-step we compute:

$$\hat{\boldsymbol{\mu}} = \frac{\beta_0 \mathbf{m}_0 + \sum_{n=1}^{N} \hat{\mathbf{x}}_n}{\beta_0 + N} \tag{4.12}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{\sum_{n=1}^{N} \left[(\hat{\mathbf{x}}_n - \hat{\boldsymbol{\mu}})^T(\hat{\mathbf{x}}_n - \hat{\boldsymbol{\mu}})\right] + \mathbf{C} + \beta_0(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}})^T(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}) + \mathbf{W}_0^{-1}}{\nu - D} \tag{4.13}$$

where $\mathbf{C}$ is defined as in equation (4.8).

4.3.3 Bayesian inference

Going one step further than MAP estimation, we might wish to perform Bayesian inference to determine the *posterior distribution* over the parameters. Assuming the dataset $\mathbf{X}$ has no missing entries, we can update the prior distribution over the parameters using closed form equations (Murphy, 2007):

$$\beta = \beta_0 + N \tag{4.14}$$

$$\nu = \nu_0 + N \tag{4.15}$$

$$\mathbf{m} = \frac{\beta_0 \mathbf{m}_0 + N\bar{\mathbf{x}}}{\beta_0 + N} \tag{4.16}$$

$$\mathbf{W} = \mathbf{W}_0 + \mathbf{S} + \frac{\beta_0 N}{\beta_0 + N}(\mathbf{m}_0 - \bar{\mathbf{x}})^T(\mathbf{m}_0 - \bar{\mathbf{x}}) \tag{4.17}$$

where:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \tag{4.18}$$

$$\mathbf{S} = \sum_{n=1}^{N} (\mathbf{x_i} - \bar{\mathbf{x}})^T(\mathbf{x_i} - \bar{\mathbf{x}}). \tag{4.19}$$

These updates are not valid if our dataset has missing entries. To circumvent this issue we can perform variational inference to iteratively perform

approximate versions of these updates.

4.4 Gaussian Mixture Model

Although the single Gaussian model addresses the independence assumption of the mean imputation model, it too has flaws. For example, we are assuming that the distribution of the data is Gaussian (which is a strong assumption to make). We can partially address this issue by using a more flexible model. One such model is a Gaussian Mixture Model (GMM).

Let us consider a D-dimensional data point $\mathbf{x}$, which we assume has been generated by the kth component of a mixture model with K Gaussian components. Let us also assume, for the time being, that $\mathbf{x}$ contains no missing values. We can calculate the likelihood of $\mathbf{x}$ as:

$$p(\mathbf{x} \,|\, z_k = 1, \boldsymbol{\theta}_k) = \mathcal{N}_k \left(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{4.20}$$

where $\mathbf{z}$ is a categorical variable that describes which component k is responsible for $\mathbf{x}$, and $\mathcal{N}_k(\cdot)$ is the kth component of the mixture model with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. Let us now assume that $\mathbf{x}$ could have been sampled from any of the K components. We let $\boldsymbol{\pi}$ be the vector of mixing proportions for each component: $\pi_k = p(z_k = 1)$, we then have:

$$p(\mathbf{x} \,|\, \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}_k \left(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right). \tag{4.21}$$

We can also calculate the likelihood of a D-dimensional dataset $\mathbf{X}$ with N entries:

$$p(\mathbf{X} \,|\, \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}, \boldsymbol{\pi}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \, \mathcal{N}_k \left(\mathbf{x}_n \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right). \tag{4.22}$$

There are a number of unknown variables that need to be estimated: the component means $\boldsymbol{\mu}_k$ and covariances $\boldsymbol{\Sigma}_k$, and the mixing proportions $\boldsymbol{\pi}$. One

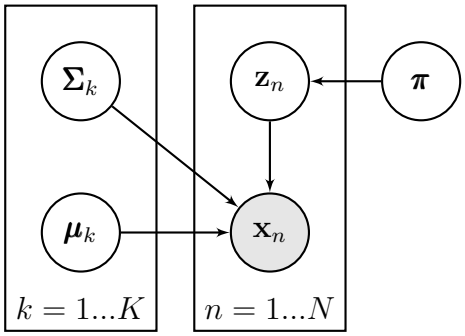


Figure 4.1: Graphical representation of the Gaussian mixture model as described in equation (4.24).

method for determining the parameters is to maximise the log-likelihood:

$$\log p(\mathbf{X} \,|\, \{\boldsymbol{\mu}_k\},\, \{\boldsymbol{\Sigma}_k\},\, \boldsymbol{\pi}) = \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}_k \left(\mathbf{x}_n \,|\, \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\right) \right]. \tag{4.23}$$

However, maximising the sum of a logarithm is non-trivial and there is no closed form solution for maximising the above log-likelihood. If we knew the component assignments $\mathbf{z}_n$ for each entry, then the complete-data log-likelihood would be:

$$\begin{aligned} \log p(\mathbf{X},\, \mathbf{Z} \,|\, \{\boldsymbol{\mu}_k\},\, \{\boldsymbol{\Sigma}_k\},\, \boldsymbol{\pi}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \Bigg[& \log \pi_k \\ & + \sum_{d=1}^{D} \log \mathcal{N}_k \left(\mathbf{x}_n \,|\, \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\right) \Bigg]. \end{aligned} \tag{4.24}$$

Figure 4.1 depicts the model described by this log-likelihood.

Unfortunately, we do not know the latent variables $\mathbf{Z}$ and they depend on $\boldsymbol{\pi}$. There is no closed form solution to maximising this log-likelihood. The EM algorithm provides an iterative method for performing this maximisation. For this model, the E-step simplifies to calculating the posterior distribution of z_{nk}, called the responsibility:

$$r_{nk} = \frac{\pi_k \mathcal{N}_k \left(\mathbf{x}_n \,|\, \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\right)}{\sum_{i=1}^{K} \pi_i \mathcal{N}_i \left(\mathbf{x}_n \,|\, \boldsymbol{\mu}_i,\, \boldsymbol{\Sigma}_i\right)}. \tag{4.25}$$

The M-step re-estimates all of the parameters of the model:

$$\pi_k = \frac{\sum_{n=1}^{N} r_{nk}}{N} \tag{4.26}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}} \tag{4.27}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k)^T(\mathbf{x}_n - \boldsymbol{\mu}_k)}{\sum_{n=1}^{N} r_{nk}}. \tag{4.28}$$

Let us now assume that there *are* missing elements in the dataset. In this case the E-step must be adjusted slightly, r_{nk} must be calculated using only the values of $\mathbf{X}$ that are not missing. The M-step is also mostly unchanged. The calculation for π_k remains the same and the calculations for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ become:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\hat{\mathbf{x}}_{nk}}{\sum_{n=1}^{N} r_{nk}} \tag{4.29}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} r_{nk}(\hat{\mathbf{x}}_{nk} - \boldsymbol{\mu}_k)^T(\hat{\mathbf{x}}_{nk} - \boldsymbol{\mu}_k)}{\sum_{n=1}^{N} r_{nk}} + \mathbf{C}_k \tag{4.30}$$

where $\hat{\mathbf{x}}_{nk}$ is defined as $\hat{\mathbf{x}}_{nk\mathbf{o}} = \mathbf{x}_{n\mathbf{o}}$ and $\hat{\mathbf{x}}_{nk\mathbf{m}} = \boldsymbol{\mu}_{k\mathbf{m}} + \boldsymbol{\Sigma}_{k\mathbf{mo}}\boldsymbol{\Sigma}_{k\mathbf{oo}}{}^{-1}(\mathbf{x}_{n\mathbf{o}} - \boldsymbol{\mu}_{k\mathbf{o}})$, and $\mathbf{C}_k$ is defined as (Delalleau et al., 2018; Ghahramani and Jordan, 1996):

$$\mathbf{C}_{k\mathbf{mm}} = \frac{\sum_{n=1}^{N} r_{nk}(\boldsymbol{\Sigma}_{k\mathbf{mm}} - \boldsymbol{\Sigma}_{k\mathbf{mo}}\boldsymbol{\Sigma}_{k\mathbf{oo}}{}^{-1}\boldsymbol{\Sigma}_{k\mathbf{om}})}{\sum_{n=1}^{N} r_{nk}}. \tag{4.31}$$

Details can be found in (Delalleau et al., 2018; Ghahramani and Jordan, 1996).

4.5 Dirichlet Process

All of the models discussed so far assume that the data is continuous. One model which does not make this assumption is a Dirichlet process (DP). A DP is a stochastic process with two parameters: a concentration parameter α and

a base distribution G_0. Given N data points $X_1, X_2, ..., X_N$ sampled from a DP, the probability that a new entry X_{N+1} takes on the value x is:

$$p(X_{N+1} = x \,|\, X_1,\, X_2,\, ...,\, X_N,\, \alpha,\, G_0) = \frac{n_x}{N+\alpha} + \frac{\alpha}{N+\alpha} G_0(x) \qquad (4.32)$$

where n_x is the number of times x occurs in $X_1, X_2, ..., X_N$. The concentration parameter α controls how many repetitions we expect to see in the data, a higher α will result in fewer repetitions. The base distribution G_0 is responsible for generating previously unseen values. It can be used to encode information about what sort of values we expect to see. Note that G_0 can be a distribution over any kind of values including discrete numbers, continuous numbers, or strings.

With a small value for α and a large dataset, the effect of the base distribution is minimised. As a result, a DP that has seen a large amount of data with only a few unique values (i.e. categorical data) will begin to predict more categorical data.

Remarkably, although the probability of X_{N+1} depends on all of the previous values $X_1, X_2, X_3, ..., X_N$ of X, the joint probability distribution of $p(X_1, X_2, ..., X_{N+1} \,|\, \alpha,\, G_0)$ is independent of order.

4.6 Bayesian Model Comparison

Using a Bayesian approach to machine learning gives us a principled way for choosing the best model for a particular dataset. Perhaps unsurprisingly, this approach is based on Bayes' rule. We noted in Section 2.2.4 that when inferring the parameters of the model, the denominator in Bayes' rule is a normalising constant. However, we will now see that the evidence was ignorable only for the *first level of inference*, in which we assumed the model was correct and inferred the parameters, but will no longer be ignorable in the *second level of inference*, in which we will infer which model is best (MacKay, 2003, ch 28).

Let M be a random variable that represents our choice in model. We can now write Bayes' rule to determine the posterior probability for a model given the data $\mathbf{X}$:

$$p(M = i \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid M = i)\, p(M = i)}{\sum_{i=1}^{I} p(\mathbf{X} \mid M = i)\, p(M = i)} \tag{4.33}$$

where $p(M = i)$ is the prior probability that model i is correct, and I is the total number of models we are considering. The likelihood of the data given the model $p(\mathbf{X} \mid M = i)$ is actually the evidence from the first level of inference.

If we set the prior probabilities for all of the candidate models to be the same then determining which model best describes the data, is equivalent to determining which model has the largest evidence. One of the main benefits of this approach is that it embodies *Occam's Razor*, the idea that the simplest explanation for the data is probably the correct one. This is because the evidence naturally punishes complex models with too many parameters, and rewards models which have just enough parameters to explain the data.

Unfortunately, calculating the evidence is often intractable, as a result the evidence must often be approximated. One common choice for this approximation is the *Bayes'* factor, which approximates the evidence using the product of the best-fit likelihood and the Occam factor (which is derived by approximating the posterior as Gaussian distribution using Laplace's approximation) (MacKay, 2003, ch 28).

However, the Gaussian distribution has a closed form equation for the evidence. In particular, the evidence for a single Gaussian distribution with a Gaussian-Wishart prior is given by Murphy (2007):

$$p(\mathbf{X}) = \frac{1}{\pi^{ND/2}} \frac{\Gamma_d(\nu/2)}{\Gamma_d(\nu_0/2)} \frac{|\mathbf{W}_0^{-1}|^{\nu_0/2}}{|\mathbf{W}^{-1}|^{\nu/2}} \left(\frac{\beta_0}{\beta}\right)^{D/2} \tag{4.34}$$

where $\mathbf{X}$ is the $N \times D$ dataset and $\Gamma_d(\cdot)$ is the multivariate Gamma function. β_0, ν_0, and $\mathbf{W}_0$ are the prior parameters, and β, ν, and $\mathbf{W}$ are the posterior

parameters, defined in equation (4.14), equation (4.15), and equation (4.17).

The evidence for the DP is also straight forward to compute because we are not inferring any parameters. We can think of this as placing a delta prior on the parameters:

$$p(\mathbf{X}) = \int_{\boldsymbol{\theta}} p(\mathbf{X} \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})\ \mathrm{d}\boldsymbol{\theta} \tag{4.35}$$

$$= \int_{\boldsymbol{\theta}} p(\mathbf{X} \,|\, \boldsymbol{\theta})\, \delta(\theta)\, \mathrm{d}\boldsymbol{\theta} \tag{4.36}$$

$$= p(\mathbf{X} \,|\, \boldsymbol{\theta}) \tag{4.37}$$

from which we can see that the evidence is equal to the likelihood.

Using equations (4.34) and (4.37), we can perform model comparison between a single Gaussian distribution and a DP.

Chapter 5

Evaluation

5.1 Simple Examples

To better understand how AutoImpute works – where it fails and where it succeeds – we can consider simple example datasets that highlight different aspects of the behaviour of the tool and the implemented models.

Let us first discuss examples for which the tool fails. Figure 5.1 shows two example datasets that will cause the tool to fail. The first dataset has an invalid entry in the fourth row of each column. In the first column, a string is present in an integer column. If a column is all strings, the tool will attempt to convert them to numeric values, but mixed strings and numeric values, or any strings that cannot be converted to numeric values, are not supported. In the second and third columns, the values ∞ and NaN will cause failures. Although both values are valid floating point numbers, due to their nature they will cause many mathematical operations to produce unsupported results. In the fourth column, no value has been entered. AutoImpute requires missing values to be explicitly indicated with a predefined string. This choice was made to prevent errors caused by incorrectly specified input files. The second dataset in Figure 5.1 has a more subtle problem that will only be observed when using maximum likelihood estimation. The estimated variance

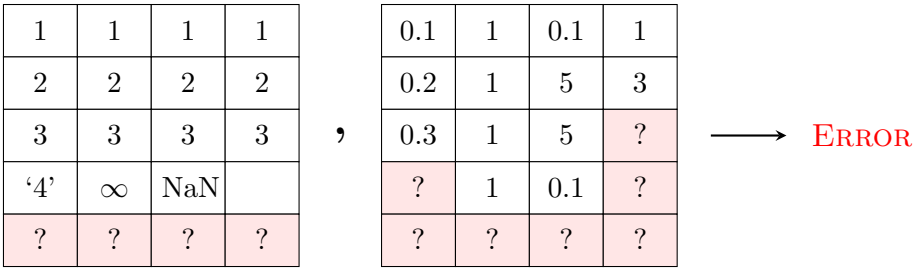


Figure 5.1: Pathological examples. Each column in the 4th row of the left table has an invalid value that will cause AutoImpute to fail. The right table has a subtle issue that is only observed when using MLE. The 2nd column has only 1 unique value, as a result the MLE of the variance will go to 0 and the likelihood for the values in the column will become arbitrarily high. This problem does not occur with MAP estimation or Bayesian Inference.

for the second column will become arbitrarily small causing numerical issues. Note that this is not a defect of AutoImpute but rather of MLE. For this reason, the default behaviour of AutoImpute is to use MAP estimation.

One way to better understand the characteristics of a probabilistic machine learning algorithm is to sample from its prior distribution. We can use AutoImpute to fill in a completely empty file where 100% of the values are missing. This use of AutoImpute lets us look at samples from the prior of whichever algorithm was selected for the imputation. Figure 5.2 shows two samples from a Dirichlet process. The *rich get richer* property of the DP is evident here, as there are a number of columns in which the values are mostly repeated. This gives us an idea of the value of α: small values of α tend to have more repetitions and higher values of α tend to produce more values from the base distribution G_0. Similarly, we can get an idea of G_0 by looking at the entire set of values present.

Figure 5.3 shows two samples from a single Gaussian distribution. Looking at these samples it seems that the variance is small, compared to the base distribution of the DP in Figure 5.2, and that the mean is close to the zero vector.

The specific values of the prior parameters are not what is important here.

?	?	?
?	?	?
?	?	?
?	?	?

1850.4	-10681	5805.2
6388.9	-15977	-9329.4
6388.9	8728.9	5805.2
6388.9	-15977	5805.2

,

5192.8	-3985.2	-11630
9652.7	-11844	-11630
-2121.4	6703.9	-11630
-2121.4	-3985.2	-8802.2

Figure 5.2: DP prior sampling. Two samples from a DP with $\alpha = 0.5$ and $G_0 = \mathcal{N}(0,\, 10000)$.

?	?	?
?	?	?
?	?	?
?	?	?

-0.1084	-0.1230	0.4914
0.1539	0.4858	-0.2336
-0.3940	-0.2160	-0.0261
0.0823	-0.0964	-0.1299

,

-0.5151	0.3033	0.0970
0.1007	-0.5903	-0.4197
-0.0492	0.3037	-0.8623
0.0428	-0.3266	-0.0089

Figure 5.3: Single Gaussian prior sampling. Two samples from a single Gaussian with $\mathbf{W}_0 = \mathbf{I}_3$ and $\mathbf{m}_0 = (0, 0, 0)$.

The key takeaway is that we can develop an intuition for the prior beliefs of our models by imputing files that contain *only* missing values.

Now let us look at samples from the *posterior distributions* of our models. Figure 5.4 shows two samples from a DP when imputing a simple input dataset. Once again, we see the *rich get richer* property of the DP. Most of the imputed values look plausible, with the exception of the bottom-right value in the second sample. This value is sampled from the base distribution G_0. Strange results like these become less likely as the amount of data increases because α becomes small relative to N (see equation (4.32)). Inferring the base distribution from the data could also diminish this problem. Note that, for the most part, the data types of each column are maintained.

The imputation samples that are drawn from a single Gaussian, shown in Figure 5.5, illustrate that with a small amount of non-missing data, most of the samples are similar to the prior samples, shown in Figure 5.3. However, the bottom-right element of the first sample illustrates how the posterior distribution has been influenced by the data.

0.1	1	0.1
0.2	1	5
0.3	1	5
?	1	0.1
?	?	?

⟶

0.1	1	0.1
0.2	1	5
0.3	1	5
0.2	1	0.1
0.2	1	5

,

0.1	1	0.1
0.2	1	5
0.3	1	5
0.2	1	0.1
0.1	1	17529

Figure 5.4: Simple DP example. Two imputation samples of a simple input dataset using a DP with $\alpha = 0.5$ and $G_0 = \mathcal{N}(0, 10000)$. The curious value in the bottom-right corner of the second sample was clearly drawn from G_0 and not a repetition of a previous value from that column.

0.1	1	0.1
0.2	1	5
0.3	1	5
?	1	0.1
?	?	?

⟶

0.1	1	0.1
0.2	1	5
0.3	1	5
0.9242	1	0.1
-0.0971	-0.4663	4.2161

,

0.1	1	0.1
0.2	1	5
0.3	1	5
-0.0087	1	0.1
0.3876	-0.0480	-0.0103

Figure 5.5: Simple single Gaussian example. Two imputation samples of a simple input dataset using a single Gaussian with $\mathbf{W}_0 = \mathbf{I}_3$, $\mathbf{m}_0 = (0, 0, 0)$, $\beta_0 = 1$ and $\nu_0 = 1$.

Table 5.1: 5 random rows, with no missing data, from the Boston Housing dataset. The first row is the column number, and last row in this table shows the predicted data type – D indicates discrete and C indicates continuous.

1	2	3	4	5	6	7	8	9	10	11	12	13
6.7177	0	18.1	0	0.713	6.749	92.6	2.3236	24	666	20.2	0.32	17.44
0.1106	0	13.89	1	0.55	5.951	93.8	2.8893	5	276	16.4	396.9	17.92
2.4495	0	19.58	0	0.605	6.402	95.2	2.2625	5	403	14.7	330.0	11.32
0.1717	25	5.13	0	0.453	5.966	93.4	6.8185	8	284	19.7	378.1	14.44
0.0136	75	4	0	0.41	5.888	47.6	7.3197	3	469	21.1	396.9	14.8
C	D	C	C	C	C	C	C	D	D	C	C	C

5.2 Model Comparison

Using model comparison we can estimate the data type classifications for each column in a dataset. Consider the example in Table 5.1, which depicts five rows chosen randomly from the Boston Housing dataset (Harrison and Rubinfeld, 1978), and the data type of each column as predicted by using model comparison. The model for continuous data is a single Gaussian with $\mathbf{W}_0 = \mathbf{I}_1 \times 10000$, $\mathbf{m}_0 = (0, 0, 0)$, $\beta_0 = 1$ and $\nu_0 = 1$ and the model for discrete is a DP with $\alpha = 0.5$ and $G_0 = \mathcal{N}(0,\ 10000)$. Note that the labels "discrete" and "continuous" are slightly misleading because what is really being compared is the evidence for a DP and a single Gaussian distribution. However, with a good choice for the parameters of these models, these evidences can be thought of as proxies for "discrete" and "continuous".

Most of the columns seem reasonably labelled – columns with floating point numbers tend to be labelled as continuous and columns with integers tend to be labelled as discrete. However, column 4, which is a binary variable, has been labelled as continuous. This can be explained by noting that in the 506 rows there are only 35 1's. As a result, the posterior variance of the single Gaussian becomes small and the likelihood becomes large.

Let us once again investigate the behaviour of the tool by considering results

Table 5.2: Toy model comparison example datasets.

(a) 1 & 0

0	0	1
0	1	1
0	?	0
0	?	?
1	?	?
?	?	?

(b) e

2.71828	2.71821	2.71821	1000
2.71828	2.71822	100000	110
2.71828	2.71848	?	9
?	?	?	?
?	?	?	?
?	?	?	?

(c) e & π

2.71828	2.71828	3.14159	2.71828
2.71828	3.14159	3.14159	?
2.71828	?	3.14159	?
2.71828	?	3.14159	?
?	?	2.71828	?
?	?	2.71828	?

from some simple examples datasets, shown in Table 5.2.

First, let us consider how the prior affects the model comparison by comparing two single Gaussian distributions with different prior means m_0. Table 5.3 shows the marginal likelihood of each model for each column in each dataset, and the ratio of the $m_0 = 0$ model's marginal likelihood to the $m_0 = 1$ model's marginal likelihood. For "1 & 0", the results make sense. Firstly, in the case that there are more 1s, the model with a prior mean of 0 has a lower marginal likelihood and vice versa. Secondly, in the case that there are an even amount of 1s and 0s the marginal likelihoods are even. Finally, the higher the ratio of one of the numbers to the other, the higher the ratio of marginal likelihoods for the corresponding model prior. The results for "e" and "e & π" also make sense. For the cases in which the numbers random selections of es and πs, the model with a prior mean of 1 has a higher marginal likelihood, and, for the cases in which there are numbers that are much bigger than both 0 and 1, the marginal likelihoods are approximately equal.

Now let us consider model comparison between a single Gaussian distribution and a Dirichlet process. Table 5.4 shows the marginal likelihood of each model for each column in each dataset, and the ratio of the single Gaussian model's marginal likelihood to the DP model's marginal likelihood. Let us compare columns 2 and 4 of e. Both columns have 3 unique values but for

Table 5.3: Comparison of marginal likelihoods for two single Gaussian distributions. Both Gaussian distributions have the same prior distributions with the exception of the mean: $W_0 = 10000$, $\nu_0 = 1$, and $\beta_0 = 1$.

	1 and 0			e				e and π			
col	1	2	3	1	2	3	4	1	2	3	4
$m_0 = 0$	3.8×10^{-4}	3.4×10^{-3}	4.9×10^{-4}	3.9×10^{-6}	3.9×10^{-6}	9.3×10^{-29}	5.9×10^{-19}	9.6×10^{-8}	1.6×10^{-7}	2.4×10^{-8}	3.0×10^{-6}
$m_0 = 1$	7.2×10^{-5}	3.4×10^{-3}	1.6×10^{-3}	3.9×10^{-5}	3.9×10^{-5}	9.3×10^{-29}	5.9×10^{-19}	6.0×10^{-6}	6.5×10^{-6}	7.8×10^{-7}	1.9×10^{-4}
ratio	0.84	0.50	0.23	0.09	0.09	0.50	0.50	0.02	0.02	0.03	0.02

column 2 the Gaussian has a higher marginal likelihood than the DP. This is because the values in column 2 are much closer together and as a result, the posterior variance of the Gaussian is small where as the posterior variance for column 4 is relatively large. The only other times that the DP has a higher marginal likelihood than the Gaussian are the cases in which there is only one unique value in a column. This might be surprising since one might imagine that the posterior variance of the Gaussian would be small in these cases. However, because the prior variance is relatively high compared to the sample variance, the small sample size results in a posterior variance that is still large. This result, as well as those from Table 5.3, highlight the effect of the prior distribution in Bayesian inference – particularly in cases where there is little data.

5.3 Effect of Missing Data Percentage

As the percentage of missing data increases, one might expect the performance of the imputation method to degrade because there are fewer examples from which to learn the distribution of the data. Figure 5.6 shows how two different metrics are affected by the amount of missing data. The first metric is the root-mean-squared-error between the imputed data $\mathbf{x}_i$ and the

Table 5.4: Comparison of marginal likelihoods for a single Gaussian distribution and a Dirichlet Process. The Gaussian distribution has the following prior parameters: $m_0 = 0$, $W_0 = 10000$, $\nu_0 = 1$, and $\beta_0 = 1$. The DP has parameters $\alpha = 0.5$ and $G_0 = \mathcal{N}(0, 10000)$.

	1 and 0			e				e and π			
col	1	2	3	1	2	3	4	1	2	3	4
SG	3.8×10^{-4}	3.4×10^{-3}	4.9×10^{-4}	3.9×10^{-6}	3.9×10^{-6}	9.3×10^{-29}	5.9×10^{-19}	9.6×10^{-8}	1.6×10^{-7}	2.4×10^{-8}	3.0×10^{-6}
DP	4.0×10^{-11}	2.7×10^{-10}	1.1×10^{-10}	1.1×10^{-5}	2.1×10^{-15}	5.1×10^{-32}	2.1×10^{-15}	9.1×10^{-6}	2.7×10^{-10}	7.3×10^{-12}	2.0×10^{-5}
ratio	1.00	1.00	1.00	0.27	1.00	1.00	0.00	0.01	1.00	1.00	0.13

true data $\mathbf{x}_t$:

$$\text{RMSE} = \sqrt{\frac{\sum_{n\in\mathbf{m}}(x_{in} - x_{tn})^2}{|\mathbf{m}|}} \tag{5.1}$$

where $\mathbf{m}$ is the set of indices for missing data. For this metric *lower* is better. The second metric is the average log-likelihood of the true data $\mathbf{x}_t$:

$$\text{Average LL} = \frac{\sum_{n\in\mathbf{m}} \log p(x_{tn} \,|\, \mathbf{X}_{:\mathbf{o}})}{|\mathbf{m}|} \tag{5.2}$$

where $p(x_{tn} \,|\, \mathbf{X}_{:\mathbf{o}})$ is the probability of the true data given the non-missing data (that was used to infer the parameters of the model). For this metric *higher* is better. As expected, both metrics become worse as the percentage of missing data increases. Note that the relative performance of the different models is not important here, because that simply indicates that for this particular dataset certain models perform better than others.

5.4 Sampling Illustration

Consider the problem of performing principal component analysis (PCA) on a dataset with missing entries. A naive solution would be to do a complete

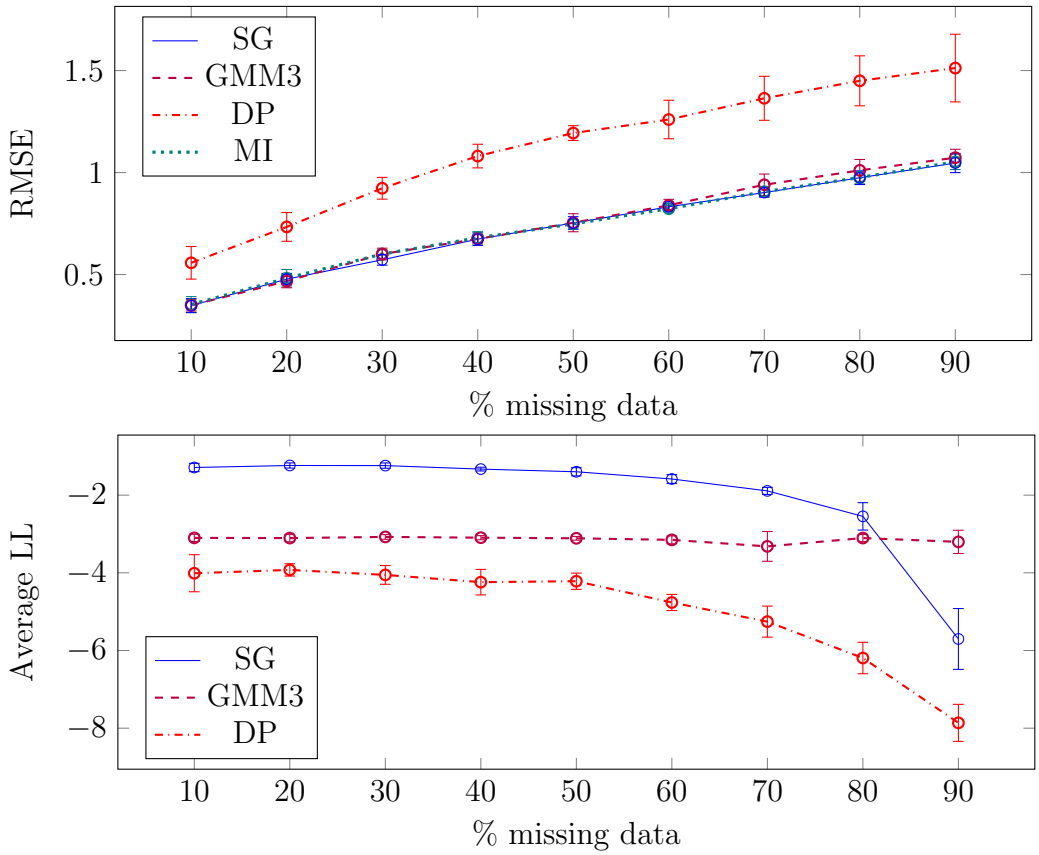


Figure 5.6: Affect of missing data percentage on different performance metrics. The data is MCAR. The results are averaged over 10 runs and 1 standard deviation is depicted by the error bars.

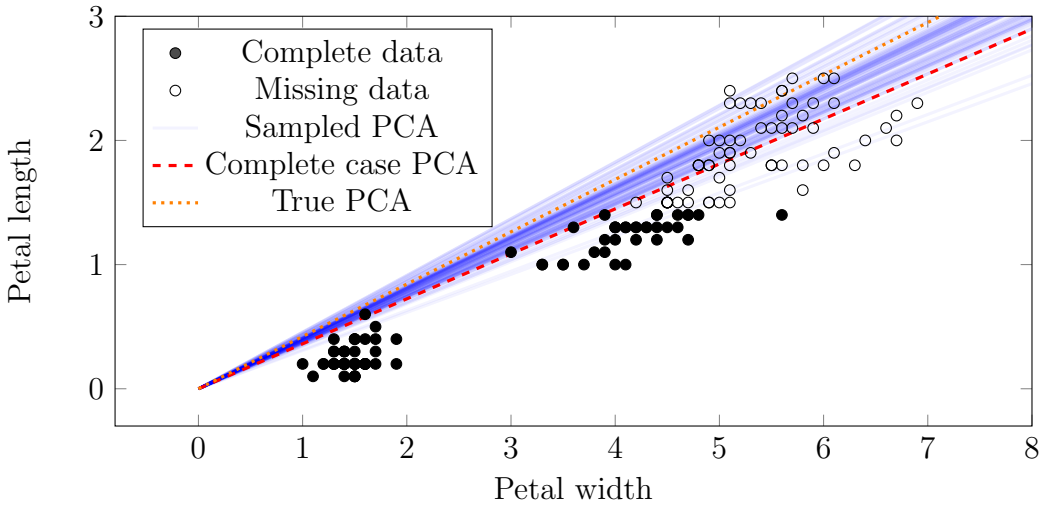


Figure 5.7: Illustration of a use case for sampling multiple imputations before running subsequent analyses. The dotted black line shows correct the principal component of the Iris data set. The dashed red line shows the principal component of the Iris dataset after performing listwise deletion. The partially transparent blue lines show principal components calculated from 100 repaired datasets sampled from a single Gaussian distribution.

case analysis: remove any entries with missing elements, and then perform PCA. However, this would not give any estimate for the variance in the principal components. On the other hand, by performing PCA on samples of imputed datasets we can get a distribution of principal components. Figure 5.7 depicts this scenario. Although many of the principal components calculated from the imputed datasets are incorrect, the distribution of these components gives a much clearer picture of what the true principal component could be. Compare this with the principal component of the dataset after listwise deletion has been performed which is incorrect and does not give any indication of uncertainty.

5.5 Comparison with MissForest

The Boston Housing dataset (Harrison and Rubinfeld, 1978) and Iris dataset (Fisher, 1963) were used to benchmark AutoImpute and MissForest. I wanted to in-

vestigate the performance of AutoImpute and MissForest for different types of missing data as well as different ratios of missing to non-missing data. To create the datasets with missing entries, the following procedures were performed on a dataset $\mathbf{X}$ with N rows and D columns.

Missing completely at random (MCAR): For each entry x_{nd} in the original dataset, randomly erase the element with a probability p.

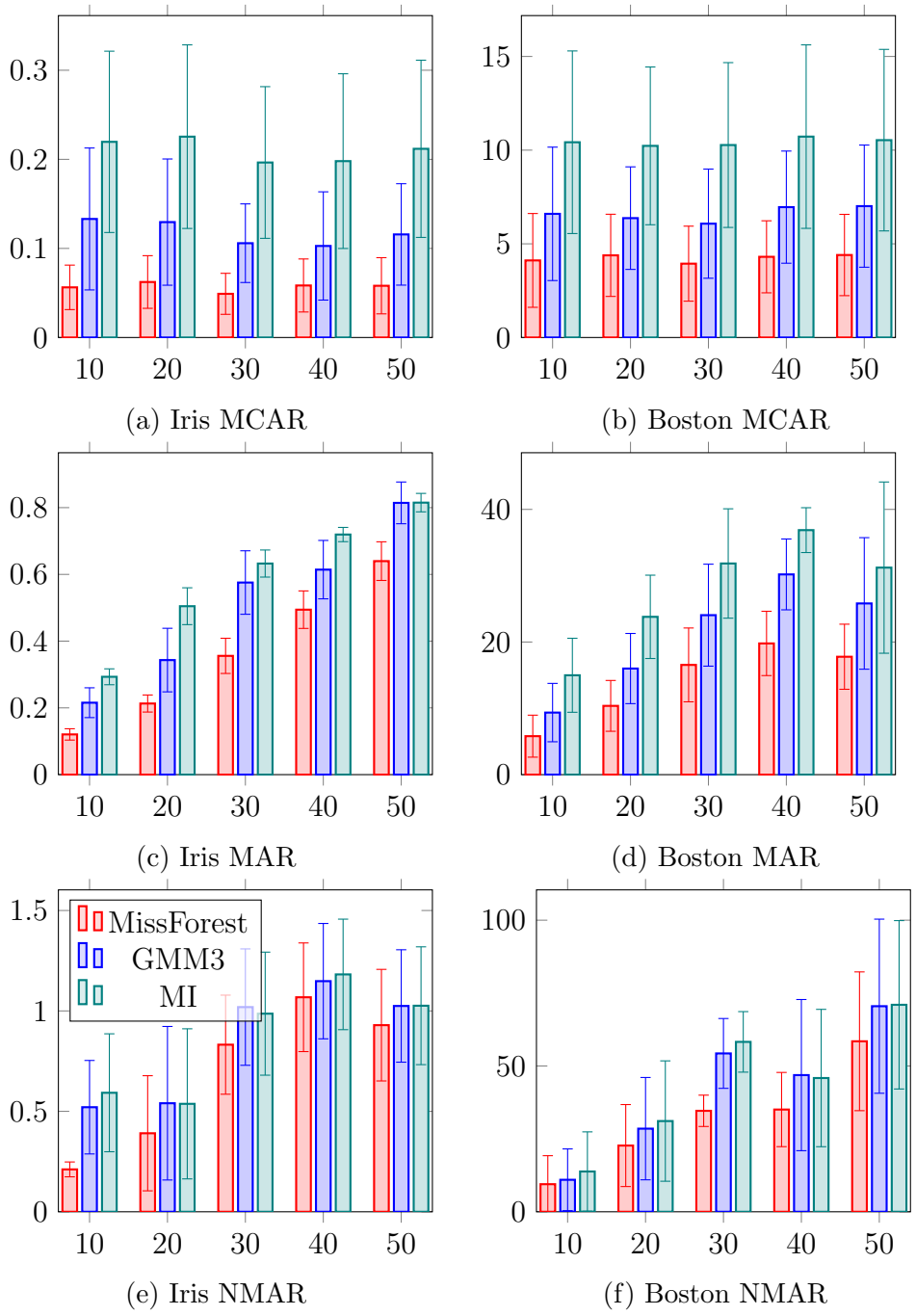
Missing at random (MAR): Randomly choose a set $\mathbf{c}$ of $\lfloor D/5 \rfloor$ columns from $\mathbf{X}$. For each row in $\mathbf{X}$ calculate $\alpha_n = \sum_{d \in \mathbf{c}} x_{nd}$. For each of the rows corresponding to the smallest $\sqrt{p} \times (1 - \lfloor D/5 \rfloor / D)^{-1}$ values of α_n, remove the elements x_{nd} for which the column d is not in $\mathbf{c}$, with a probability $\sqrt{p}$. In this case, approximately $4/5$ columns will have values removed, and for each of those columns, approximately $\sqrt{p} \times 5/4$ of the entries will be removed with a probability of $\sqrt{p}$, thus the overall percentage of removes entries will be approximately p. Note that as p becomes larger, the approximation becomes less accurate.

Not missing at random (NMAR): For each column in $\mathbf{X}$, remove the $\lfloor p \times N \rfloor$ lowest values.

Figure 5.8 shows the results of using MissForest, a GMM with 3 components, and mean imputation to repair the Iris and Boston Housing datasets with the missing data patterns described above. The MissForest imputation used the default settings described in Stekhoven (2016). The GMM parameters were inferred using the EM algorithm to determine the MAP estimate. The hyper-parameters were $\mathbf{m}_0 = (0, ..., 0)^T$, $\mathbf{W}_0 = \mathbf{I}_D$, $\beta_0 = 1$, and $\nu_0 = 1$. The EM algorithm was stopped after the average LL improved by less than 0.01, or after 100 iterations, whichever occurred first.

The results show that in almost every case, MissForest outperforms the GMM. Similarly, the GMM almost always outperforms the baseline of mean imputation.

We can also see that, as expected, for each of the methods, the performance is best for the MCAR case and worst for the NMAR case.



(a) Iris MCAR (b) Boston MCAR

(c) Iris MAR (d) Boston MAR

(e) Iris NMAR (f) Boston NMAR

Figure 5.8: Comparison between MissForest and AutoImpute for the Iris and Boston Housing datasets with MCAR, MAR, and NMAR missing data of different percentages. The x-axes are % missing data and the y-axes are RMSE – see equation (5.1). The results are averaged over 10 runs and 1 standard deviation is depicted by error bars.

Chapter 6

Summary and Conclusions

This dissertation investigated the problem of handling missing data. To facilitate this investigation, a tool called AutoImpute was built. Building this tool highlighted a number of challenging aspects that arise when working with missing data. Probabilistic machine learning techniques offer solutions to some of these challenges. For example, Bayesian inference avoids the problem of over-fitting, and provides a principled way of incorporating prior knowledge about a dataset into the imputation of missing values.

Bayesian model comparison offers a mathematically correct method for determining which model best describes a dataset. Bayesian model comparison may also be useful as a way of inferring the data types of variables in a dataset.

As a result of using probabilistic machine learning methods, multiple imputation is as simple as drawing multiple samples from the posterior distribution of the data. We can also impute by replacing missing values with the mode of the posterior distribution.

However, inferring the distribution of missing data can be challenging. Even if the MAR or MCAR assumptions are made, the missing data makes it difficult to derive closed form equations for the parameters of the distribution, or estimates of them. As a result, we have to resort to using approximate

inference techniques, such as the EM-algorithm and Variational Bayes.

The evaluation of methods for dealing with missing data are difficult, too. It is not possible to compare to the "correct answer" on real datasets with missing values, where the ground truth is not known. Artificial datasets can be created by manually removing values and can be tested objectively because the correct values are known. But it is not clear to what extent these results generalise to real-world datasets.

6.1 Future Work

There are a number of directions for future work on the topic of dataset repair. One such direction is the improvement of the models for missing data. Dirichlet process (DP) models are not confined to numeric values, and a DP could be used to impute missing values in the presence of NaNs and ∞s by treating these poorly behaved floating point numbers as special symbolic values or as strings. It is also possible infer the parameters of the DP which would reduce the requirement of choosing good values for these parameters.

The DP provides another attractive area for future work: the DP mixture model, which can be thought of a mixture model with an infinite number of components. Unfortunately, having an infinite number of components means that exact inference in such models is not generally possible. However, approximate inference techniques such as MCMC and VB exist for such models (Rasmussen, 2000; Blei and Jordan, 2006). By using a DP mixture model, the problem of choosing the correct number of mixture components can be avoided.

Another opportunity for improvement is calculation of the evidence for the GMM. Solving this challenge could help with the problem of choosing the correct number of mixture components, because models with different numbers of components could be compared directly and fairly. It would also allow for better modelling of continuous variables when detecting data types.

Other techniques for imputing missing data, such as Random Forests, could also be incorporated into AutoImpute.

Another direction for future work is improvements to the AutoImpute application. A first step in this direction would be to improve the coverage of the test suite which would streamline other improvements.

The user interface (UI) and user experience (UX) for the tool could also be greatly improved. The error reports might be unhelpful to inexperienced users who might not understand why the application failed. The command line UI is often too verbose and could definitely be streamlined. For example, it would be nice to have `--fast` and `--exhaustive` modes that would perform a simple but fast imputation, or a exhaustive search over different models to find the best imputation method, respectively. Supporting multiple indicators for missing values would be a useful feature. Future work could involve user studies aimed at improving UI and UX.

Finally, there are a number of opportunities for future work in the evaluation of different algorithms for imputing missing data. Adding more datasets to the analyses would be an easy improvement to make.

Benchmarking against additional methods, other than MissForest, would also be interesting. MICE (Van Buuren and Oudshoorn, 1999), KNNimpute (Troyanskaya et al., 2001), and MissPaLasso (Städler et al., 2014), discussed in Section 3.2, are all potential candidates for comparison. The analysis done by Stekhoven and Bühlmann (2012) could be extended to include MAR and NMAR data.

Comparing the different imputation methods by additional metrics could contribute to a clearer picture of when one method might be more appropriate than another. One metric that would be interesting to consider is running time. The Pareto-frontier comparing accuracy and running time of different methods could be interesting to investigate.

Missing data imputation is an open and challenging field which leads to many exciting and interesting opportunities for future work.

Appendix A

Notation

All of the notation used in this dissertation is described as it is introduced. The following table provides a quick reference for all of the notation if needed.

Example	**Meaning**
X	random variable
$\mathbf{x}$	vector
$\mathbf{x} = (x_1, x_2, x_3)$	column vector
$\mathbf{x} = (x_1, x_2, x_3)^T$	row vector
$\mathbf{X}$	matrix
$\mathbf{I}_n$	$n \times n$ identity matrix
$\mathbf{0}_n$	$n \times n$ zero matrix
$\mathbf{X}^T$	matrix transpose
$\mathbf{X}^{-1}$	matrix inverse
$\mathrm{Tr}(\mathbf{X})$	the trace of $\mathbf{X}$
x_i	ith element of a vector $\mathbf{x}$
$\mathbf{x_c}$	vector of elements whose indices are in the set $\mathbf{c}$
x_{ij}	element at the ith row and jth column of a matrix $\mathbf{X}$
$\mathbf{X_{ij}}$	sub-matrix of elements whose row and column indices are in the sets $\mathbf{i}$ and $\mathbf{j}$, respectively
$\mathbf{x}_i$	ith row of the matrix $\mathbf{X}$
$\mathbf{x}_{:j}$	jth column of the matrix $\mathbf{X}$
$\mathbf{X_{ij}}^{-1}$	the inverse of a sub-matrix
$\mathbf{X^{-1}}_{\mathbf{ij}}$	the sub-matrix of an inverse
(a, b)	a range of values between a and b, inclusive
$\hat{y}$	an estimator for the variable y
$\Gamma(z)$	the Gamma function $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}\,\mathrm{d}x$ which generalises the factorial function to real numbers

$D_{\mathrm{KL}}(q \,\|\| \, p)$	the KL-divergence between $q(\mathbf{Z})$ and $p(\mathbf{Z} \,\|\, \mathbf{X})$: $\int_{\mathbf{X}} q(\mathbf{Z}) \log \frac{q(\mathbf{Z})}{p(\mathbf{Z} \,\|\, \mathbf{X})} \, \mathrm{d}\mathbf{X}$
x	the one hot representation of a categorical variable x - $\mathsf{x}_d = 1$ if x is class d, otherwise $\mathsf{x}_d = 0$
$\{x_i\}$	the set containing x_i for all i
$\mathbf{X}_{:\mathbf{m}}$	the elements of the dataset $\mathbf{X}$ that *are* missing
$\mathbf{X}_{:\mathbf{o}}$	the elements of the dataset $\mathbf{X}$ that *are not* missing
$\mathbf{x}_{n\mathbf{m}}$	the elements of the nth row of the dataset $\mathbf{X}$ that *are* missing
$\mathbf{x}_{n\mathbf{o}}$	the elements of the nth row of the dataset $\mathbf{X}$ that *are not* missing
$\lfloor \cdot \rfloor$	the floor operation

Note that the inverse of a sub-matrix ($\mathbf{X_{ij}}^{-1}$) is not the same as the sub-matrix of an inverse ($\mathbf{X^{-1}}_{\mathbf{ij}}$).

Appendix B

Important Probability Distributions

The following distributions are used throughout this dissertation. They serve as the basis for many of the imputation models presented in Chapter 4.

B.1 Gaussian

The Gaussian, or *normal*, distribution is a distribution over continuous values in $\mathbb{R}^D$. If a D-dimensional variable $\mathbf{x}$ is Gaussian distributed, then:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{B.1}$$

where $\boldsymbol{\mu}$ is a D-dimensional vector of means, and $\boldsymbol{\Sigma}$ is a $D \times D$ symmetric and positive definite covariance matrix. It is sometimes mathematically convenient to use an alternative parametrisation of the Gaussian with a precision matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$.

A special case is the 1-dimensional Gaussian distribution which can be written as:

$$\mathcal{N}(x \,|\, \mu,\, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(x-\mu)^2}{2\sigma^2} \tag{B.2}$$

where μ and σ^2 are the mean and variance.

The Gaussian distribution has a number of useful properties. Firstly, the marginal and conditional distributions of a multivariate Gaussian with respect to any subset of the variables are also Gaussian. Concretely, consider a partition of $\mathbf{x}$ into $\mathbf{x_a}$ and $\mathbf{x_b}$. Similarly $\boldsymbol{\mu}$ is partitioned into $\boldsymbol{\mu_a}$ and $\boldsymbol{\mu_b}$, and $\boldsymbol{\Lambda}$ is partitioned into $\boldsymbol{\Lambda_{aa}}$, $\boldsymbol{\Lambda_{ab}}$, $\boldsymbol{\Lambda_{ba}}$ and $\boldsymbol{\Lambda_{bb}}$. Now:

$$p(\mathbf{x_a}) = \mathcal{N}\big(\mathbf{x_a} \,\big|\, \boldsymbol{\mu_a}, \boldsymbol{\Lambda_{aa}}^{-1}\big) \tag{B.3}$$

and

$$p(\mathbf{x_a} \,|\, \mathbf{x_b}) = \mathcal{N}\big(\mathbf{x_a} \,\big|\, \boldsymbol{\mu_a} - \boldsymbol{\Lambda_{aa}}^{-1}\boldsymbol{\Lambda_{ab}}(\mathbf{x_b} - \boldsymbol{\mu_b}), \boldsymbol{\Lambda_{aa}}^{-1}\big) \tag{B.4}$$

which are both Gaussian (Bishop, 2006, ch 2). Figure B.1 gives an example of this property, showing a joint Gaussian distribution as well as examples of Guassian marginal and conditional distributions. Finally, the conjugate-prior (see Section 2.2.1) for $\boldsymbol{\mu}$ is its self a Gaussian. Conjugate-prior for the covariance and precision matrices are the inverse Wishart and Wishart distributions, discussed in Section B.4 and Section B.3.

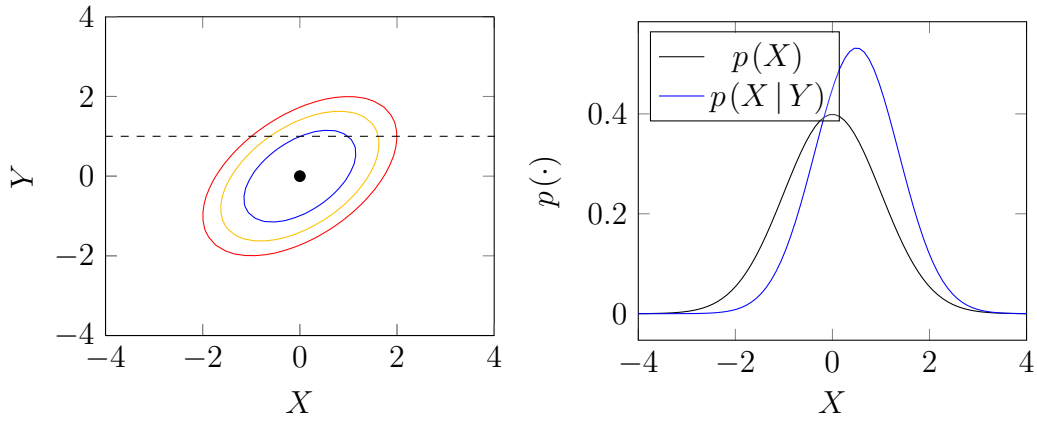


Figure B.1: An illustration of Gaussian distributions. On the left is $p(X, Y)$ the joint distribution between two random variables X and Y. On the right is $p(X)$, the marginal distribution of X, and $p(X \,|\, Y = 1)$, the conditional distribution of X given that $Y = 1$.

B.2 Categorical

The categorical distribution is a multivariate discrete distribution. If $\mathbf{x}$ is a D-dimensional one-hot variable which indicates that $\mathbf{x}$ is one of D different classes:

$$\text{Cat}\left(\mathbf{x} \,|\, \boldsymbol{\pi}\right) = \prod_{d=1}^{D} \pi_d^{x_d} \tag{B.5}$$

where $\boldsymbol{\pi} = (\pi_1, ..., \pi_D)$ is a vector of probabilities for each of the classes $\mathbf{x}$ could take, and x_d is the dth value of $\mathbf{x}$ that is 1 only if $\mathbf{x}$ is class d and is 0 otherwise. Note that $0 \leq \pi_d \leq 1$ and $\sum_d \pi_d = 1$. A conjugate-prior for the categorical is the Dirichlet distribution, discussed in Section B.5.

B.3 Wishart

The Wishart distribution is a distribution over precision matrices. If $\boldsymbol{\Lambda}$ is Wishart distributed then:

$$\mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}, \nu) = \frac{|\boldsymbol{\Lambda}|^{(\nu-D-1)/2} \exp(-\text{Tr}(\mathbf{W}^{-1}\boldsymbol{\Lambda})/2)}{2^{\nu D/2}|\mathbf{W}|^{\nu/2}\pi^{D(D-1)/4}\prod_{d=1}^{D}\Gamma((\nu+1-i)/2)} \tag{B.6}$$

where $\mathbf{W}$ is a $D \times D$ symmetric and positive definite scale matrix (precision prior), and $\nu > D-1$ is the degrees of freedom.

B.4 Inverse Wishart

The inverse Wishart distribution is a distribution over covariance matrices. If $\boldsymbol{\Sigma}$ is Wishart distributed then:

$$\mathcal{W}^{-1}(\boldsymbol{\Sigma}|\mathbf{W}, \nu) = \frac{|\mathbf{W}|^{\nu/2} \exp(-\text{Tr}(\mathbf{W}\boldsymbol{\Sigma}^{-1})/2)}{2^{\nu D/2}|\boldsymbol{\Sigma}|^{-(\nu+D+1)/2}\pi^{D(D-1)/4}\prod_{d=1}^{D}\Gamma((\nu+1-i)/2)} \tag{B.7}$$

where $\mathbf{W}$ is a $D \times D$ symmetric and positive definite scale matrix (covariance prior), and $\nu > D - 1$ is the degrees of freedom.

B.5 Dirichlet

The Dirichlet is a distribution over D-dimensional vectors $\boldsymbol{\pi}$, subject to the constraints:

$$0 \leq \pi_d \leq 1 \quad \text{and} \quad \sum_d \pi_d = 1. \tag{B.8}$$

That is, $\boldsymbol{\pi}$ is Dirichlet distributed if:

$$\text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_d \alpha_d)}{\sum_d \Gamma(\alpha_d)} \prod_{d=1}^{D} \pi_d^{\alpha_d - 1} \tag{B.9}$$

where $\boldsymbol{\alpha}$ is a D-dimensional vector subject to the constraint that $\alpha_d > 0$ for all d. The Dirichlet distribution only has finite density at all points if $\alpha_d \geq 1$ for all d. One useful property of the Dirichlet distribution, when it is used as a conjugate prior for the categorical distribution, is that setting $\alpha_d = 1$ for all d results in a uniform distribution. When the Dirichlet distribution is used as a conjugate prior for the categorical distribution, each α_d can be interpreted as pseudo-count for observations of the dth class in the categorical distribution. A special case of the Dirichlet distribution is when α_d is the same for all values of d – this is called a symmetric Dirichlet distribution and it is often encountered as a conjugate prior. Figure B.2 gives examples showing the effect of α is a symmetric Dirichlet distribution.

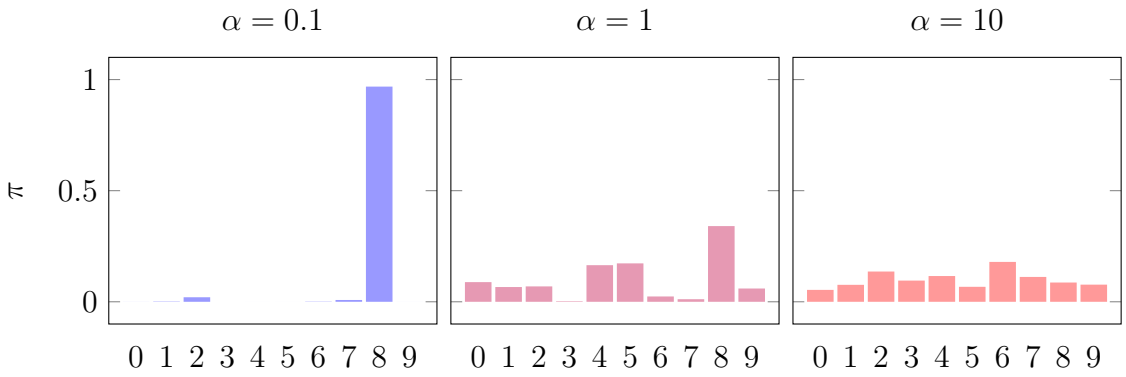


Figure B.2: Categorical distributions with 10 classes sampled from symmetric Dirichlet distributions with different α values. Small values of α, corresponding to a few prior pseudo-counts of each class, generate categorical distributions with high variance. Large values of α, corresponding to large prior pseudo-counts for each category, result in categorical distributions with low variance.

Appendix C

Testing

This appendix describes the unit-tests for AutoImpute, and gives their output.

C.1 Mean imputation

Test name	Description	Output
`NoMissingValuesRMSE`	Tests that when there are no missing values the prediction RMSE is 0.	`OK`
`NoMissingValuesLL`	Tests that the missing data log-likelihoods for a toy dataset are correct.	`OK`
`OneValueResult`	Tests that when there is only one non-missing value, the mean imputation will return exactly that value.	`OK`
`TwoValuesResult`	Tests that when there are only two non-missing values, the mean imputation will return the correct value.	`OK`
`AllMissingValues`	Tests that when there are no observed values the results of mean imputation are all 0s.	`OK`
`AllMissingValuesLL`	Tests that when there are no observed values the missing data log-likelihoods are correct.	`OK`
`NoRows`	Tests that if a dataset with no rows is the input then an error will be thrown.	`OK`

NoCols	Tests that if a dataset with no columns is the input then an error will be thrown.	OK
OneColumnAllMissing	Tests that the mean imputation works if given only 1 column.	OK

C.2 Dirichlet process

Test name	Description	Output
EandPiResult	Tests the predictions of the maximum likelihood imputation of the DP using.	OK
EandPiLL	Tests the missing data log-likelihoods of the maximum likelihood imputation of the DP.	OK
NotAllZerosGivenNoObs	Tests that the sample function works when given only missing data.	OK
SingleColumnSample	Tests that a) the DP works on a single column, and that b) if there are no missing values the output will be the observed data.	OK
SingleColumnMultiple	Tests that all samples from a dataset with no missing values will be the same.	OK
OneColumnAllMissing	Tests that the maximum likelihood prediction of the DP on a dataset with only missing values is the mode of the base distribution G_0.	OK

C.3 Single Gaussian

Test name	Description	Output
NoMissingValuesRMSE	Tests that if there are no missing values the prediction RMSE is 0.	OK
NoMissingValuesLL	Tests that the average missing data log-likelihood is not a number when there is no missing data.	OK

AllMissingValuesMLERes	Tests that the maximum likelihood predictions before and after fitting a single Gaussian with MLE are the same if all the data is missing.	OK
OneValueMLEResult	Tests that the maximum likelihood predictions of an MLE single Gaussian are correct when there is only 1 non-missing value in each column.	OK
TwoValuesMLEResult	Tests that the maximum likelihood predictions of an MLE single Gaussian are correct when there are only 2 non-missing values in each column.	OK
TwoValueMAPResult	Test that the maximum likelihood prediction of a MAP estimated single Gaussian are correct when there are only 2 non-missing values in each column.	OK
TwoValuesSamplesDiff	Tests that samples from the single Gaussian are not identical.	OK
IndependentVsDependent	Tests that a single Gaussian with a diagonal covariance matrix performs worse than one with a full covariance matrix, when the dataset has correlation between its variables.	OK
OneColumnPred	Tests that the single Gaussian works for a dataset with a single column.	OK
OneColumnSample	Tests that samples drawn for the same variable are different.	OK
OneColumnLL	Tests that single Gaussians with diagonal and full covariance matrices will perform equivalently on a single variable dataset.	OK
OneColumnAllMissing	Tests that for a dataset with a single column, if all the values are missing, the predictions before and after using MAP estimation will be the same.	OK

C.4 Guassian Mixture Model

Test name	Description	Output
NoMissingValuesRMSE	Tests that if there are no missing values the prediction RMSE is 0.	OK
NoMissingValuesLL	Tests that the average missing data log-likelihood is not a number when there is no missing data.	OK
AllMissingValuesMLERes	Tests that the maximum likelihood predictions, before and after fitting a single component GMM with MLE, are the same if all the data is missing.	OK
AllMissValu esMLEResMult	Tests that the maximum likelihood predictions, before and after fitting a three component GMM with MLE, are the same if all the data is missing.	OK
OneColumnAllMissing	Tests that the maximum likelihood predictions, before and after fitting a single component GMM with MLE to a single variable dataset, are the same if all the data is missing.	OK
OneValueMLEResult	Tests that the maximum-likelihood predictions of an MLE GMM are correct when there is only 1 non-missing value in each column.	OK
TwoValueMLEResult	Tests that the maximum-likelihood predictions of an MLE GMM are correct when there is only 2 non-missing value in each column.	OK
TwoValuesSamplesDiff	Tests that samples from the GMM are not identical.	OK
IndependentVsDependent	Tests that a GMM with a diagonal covariance matrices for the components performs worse than one with full covariance matrices, when the dataset has correlation between its variables.	OK
OneColumnPred	Tests that the GMM works for a dataset with a single column.	OK
OneColumnSample	Tests that samples drawn for the same variable are different.	OK

`OneColumnLL`	Tests that GMMS which have components with diagonal and full covariance matrices are equivalent on a single variable dataset.	`OK`
`TwoCompLLWorseThan10`	Tests that a MLE GMM with 2 components has a lower missing data log-likelihood than one with 10 components.	`OK`
`MAPandMLEGiveDifferentLL`	Tests that the MAP estimate and MLE for the GMM parameters are different.	`OK`
`TenCompMAPIris50`	Tests that a 10 component GMM with MAP estimation makes predictions for the Iris dataset (Fisher, 1963) with 50 MCAR data.	`OK`

Bibliography

Baum, Leonard E., Petrie, Ted, Soules, George and Weiss, Norman (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, **41**(1) 164–171. ISSN 0003-4851. URL http://projecteuclid.org/euclid.aoms/1177697196.

Bishop, Chris M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer. ISBN 9-780-387-31073-2. URL https://books.google.co.uk/books?id=kTNoQgAACAAJ.

Blei, David M. and Jordan, Michael I. (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, **1**(1) 121–144.

Breiman, Leo (2001). Random forests. *Machine Learning*, **45**(1) 5–32. ISSN 1573-0565. URL https://doi.org/10.1023/A:1010933404324.

Delalleau, Olivier, Courville, Aaron and Bengio, Yoshua (2018). Efficient EM Training of Gaussian Mixtures with Missing Data. URL https://arxiv.org/pdf/1209.0521.pdf.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, **39**(1) 1–38. URL http://www.jstor.org/stable/2984875?seq=1#page_scan_tab_contents.

Faes, C., Ormerod, J. T. and Wand, M. P. (2010). Variational Bayesian Inference for Parametric and Nonparametric Regression with Missing Data. URL http://www.maths.usyd.edu.au/u/jormerod/JTOpapers/faespap.pdf.

Fisher, Ronald A. (1963). The use of multiple measurement in taxonomic problems. *Annals of Eugenics*, **7**(2) 179–188. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x.

Ghahramani, Zoubin and Jordan, Michael I. (1996). Supervised learning from incomplete data via an EM approach. URL http://papers.nips.cc/paper/767-supervised-learning-from-incomplete-data-via-an-em-approach.pdf.

Harrison, David and Rubinfeld, Daniel (1978). Hedonic housing prices and the demand for clean air. In *Journal of Environmental Economics and Management*, volume 5, 81–102.

Hernández-Lobato, José Miguel, Houlsby, Neil and Ghahramani, Zoubin (2014). Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, II–1512–II–1520. JMLR.org. URL http://dl.acm.org/citation.cfm?id=3044805.3045061.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, **349**(6245) 255–260. ISSN 0036-8075. URL http://science.sciencemag.org/content/349/6245/255.

Little, Roderick J. A. and Rubin, Donald B. (2002). *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., 2nd edition. ISBN 978-0-471-18386-0.

MacKay, David J.C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. ISBN 978-0-521-64298-9. URL https://books.google.co.uk/books?id=AKuMj4PN_EMC.

Marlin, Benjamin M., Zemel, Richard S., Roweis, Sam T. and Slaney, Malcolm (2012). Collaborative filtering and the missing at random assumption. URL http://arxiv.org/abs/1206.5267.

Melchior, Peter and Goulding, Andy D (2016). Filling the gaps: Gaussian mixture models from noisy, truncated or incomplete samples. URL https://arxiv.org/pdf/1611.05806.pdf.

Murphy, Kevin P (2007). Conjugate Bayesian analysis of the Gaussian distribution. URL https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf.

Orchard, Terence and Woodbury, Max A (1976). A missing information principle: Theory and applications. URL https://projecteuclid.org/download/pdf_1/euclid.bsmsp/1200514117.

Rasmussen, Carl Edward (2000). The infinite gaussian mixture model. In *Advances in neural information processing systems*, 554–560.

Ricci, Francesco, Rokach, Lior and Shapira, Bracha (2011). *Introduction to Recommender Systems Handbook*, 1–35. Springer US, Boston, MA. ISBN 978-0-387-85820-3. URL https://doi.org/10.1007/978-0-387-85820-3_1.

Rubin, Donald B (1987). *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, New York. ISBN 978-0-471-08705-2.

Rubin, Donald B. (1996). Multiple imputation after 18+ years. *Journal of the American Statistical Association*, **91**(434) 473–489. ISSN 0162-1459. URL http://www.jstor.org/stable/2291635.

Smola, Alex J., Vishwanathan, S. V. N. and Hofmann, Thomas (2005). Kernel Methods for Missing Variables. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.221.8829&rep=rep1&type=pdf#page=334.

Städler, Nicolas, Stekhoven, Daniel and Bühlmann, Peter (2014). Pattern alternating maximization algorithm for missing data in high-dimensional problems. In *Journal of Machine Learning Research*, volume 15, 1903–1928.

Steinruecken, Christian and Iwata, Tomoharu (2013). Advanced sampling. URL http://www.inference.org.uk/tcs27/talks/sampling.html.

Stekhoven, Daniel J. (2016). Nonparametric missing value imputation using random forest. URL https://cran.r-project.org/web/packages/missForest/missForest.pdf.

Stekhoven, Daniel J. and Bühlmann, Peter (2012). MissForest – nonparametric missing value imputation for mixed-type data. *Bioinformatics*, **28**(1) 112–118. URL http://dx.doi.org/10.1093/bioinformatics/btr597.

Sundberg, Rolf (1974). Maximum likelihood theory for incomplete data from an exponential family. *Scandinavian Journal of Statistics*, **1**(2) 49–58. URL https://www.jstor.org/stable/4615553.

Troyanskaya, Olga, Cantor, Michael, Sherlock, Gavin, Brown, Pat, Hastie, Trevor, Tibshirani, Robert, Botstein, David and Altman, Russ B (2001). Missing value estimation methods for dna microarrays. *Bioinformatics*, **17**(6) 520–525.

Van Buuren, Stef and Groothuis-Oudshoorn, Karin (2011). MICE: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**.

Van Buuren, Stef and Oudshoorn, Karin (1999). Flexible multivariate imputation by MICE. Technical Report PG/VGZ/99.054, TNO Prevention and Health.

Williams, Christopher K. I. and Nash, Charlie (2018). Autoencoders and probabilistic inference with missing data: An exact solution for the factor analysis case. URL `https://arxiv.org/pdf/1801.03851.pdf`.