High-Dimensional Bayesian Non-parametric Learning of System Parameters in Different Data Scenarios

Author

Kangrui WANG

Supervisor

Dr. Dalia CHAKRABARTY

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy

Department of *Mathematics*
University of Leicester

2018

Declaration

I, Kangrui Wang, declare that this thesis titled, *High-Dimensional Bayesian Non-parametric Learning of System Parameters in Different Data Scenarios* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Abstract

High-Dimensional Bayesian Non-parametric Learning of System Parameters in Different Data Scenarios

Kangrui Wang

The pursuit of the correlation structure of a high-dimensional random construct, underlines my doctoral studies. This thesis reports on the development of methodologies that help undertake learning of functional relationships between variables, given high-dimensional discontinuous data that exhibit non-stationary correlation structure, with such methods tying in with methods needed to undertake such difficult correlation learning–and its possible intuitive graphical representations as networks. These developed methods are then presented in an application-ready format, in which the relevant inference is typically undertaken with Markov Chain Monte Carlo methods.

I have worked on developing Bayesian methodologies for the supervised learning of the functional relationship between a system vector and another tensor-valued observable that affects the system vector, given real training data that consists of known pairs of values of these variables. The probabilistic learning of the functional relation between these variables is done by modelling this function with a high-dimensional Gaussian Process (GP), and the likelihood is then parametrised by multiple covariance matrices. I have developed on the method of nesting GPs of different dimensionalities, to render covariance kernels non-

stationary, by treating each kernel hyper-parameter as a realisation from a scalar-valued GP. The inner layer of this learning strategy is then built of scalar-valued GPs, which are nested within a tensor-valued GP, and inference is done with Metropolis-within-Gibbs.

It is natural that such interest includes the learning of the correlation structure of multivariate, rectangularly-shaped data, which is manifest in the sought graphical model of this data, where I determine objective uncertainties in the learning of such a graphical models, where such uncertainty learning allows me to quantify the correlation between a pair of such datasets by computing the distance between the (posterior probability densities of the) learnt graphical models of the respective datasets. Applications include the learning of the very large, human disease-symptom network and computation of the distance between the vino-chemical graphical models of red and white Portuguese wines.

Acknowledgement

I would like to express my special appreciation and thanks to my supervisor Dr. Dalia Chakrabarty. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless. My sincere thanks also go to Dr. Bo Wang who has supported me a lot during my Ph.D studies.

A special thanks to my wife, Isabelle, who has spent sleepless nights with me and has always been my support in the moments when there was no one else to answer my queries.

Aims & Summary of Content

Aims, Summary of Content, and New Methods

As we progress through the data revolution, learning the correlation structure of high-dimensional objects becomes important. Knowing this, and if in information on one component of the object becomes available, we can then predict possible values of any other component. Such correlation learning forms the basic aim of my doctoral studies, where my interest is in developing probabilistic – in particular, Bayesian – methodologies that allow for such learning.

This thesis focuses on advancing methodologies that help undertake learning of functional relationships between variables, given different difficult data situations, including high-dimensional discontinuous data that exhibit non-stationary correlation structure. Thus the methods discussed later in the thesis tie in with methods needed to undertake such difficult, (i.e. non-stationary) correlation learning, as well as its possible intuitive graphical representations as networks. These developed methods are then presented in an application-ready format, in which the relevant inference is typically undertaken with Markov Chain Monte Carlo methods, and these methodologies are illustrated within the paradigm of real-life problems, using real data sets.

Thus, one facet of my doctoral research has been dedicated to the development of Bayesian methodologies for the supervised learning of the functional relationship between

a system vector and another tensor-valued (in general) observable that affects the system vector, given the training data that consists of known pairs of values of these variables. The data on this tensor-valued observable is then obtained by stacking multiple measurements of this observable, and it thus becomes shaped as a hyper-cuboid. The probabilistic learning of the functional relation between these variables is done by assigning a probability distribution to the sought function, i.e. by modelling this function with a stochastic process. The stochastic process that I invoke for this purpose is a high-dimensional Gaussian Process (GP), only because of ease of computation and modelling. The likelihood is then rendered a tensor-Normal density that is parametrised by multiple covariance matrices. The following chapters discuss the learning – and particularly – the parametrisation of the unknown covariance matrices. Learning these is equivalent to parametrising the GP that the sought function is a random realisation of. So once the GP is parametrised, we can sample the sought function from it, and use this (sampled) function to predict the value of either of the original two variables, at a test value of the other variable. This is the material covered in the second chapter of the thesis.

It is natural that such interest includes the learning of the correlation structure of multivariate, rectangularly-shaped data, which is manifest in the sought graphical model of this data, where I am keen on determining objective uncertainties in the learning of such a graphical models, as is undertaken in the third chapter of this thesis. Such uncertainty learning allows me to quantify the correlation between a pair of such datasets by computing the distance between the (posterior probability densities of the) learnt graphical models of the respective datasets. This work allows for fast inference, and an application of this therefore includes the learning of the very large, human disease-symptom network. Importantly, the work includes computation of the distance between the vino-chemical graphical

models of red and white Portuguese wines, where this distance is defined in terms of the distance between the posterior probability densities of the pair of graphical models that are learnt given the respective data set. Thus the distance here is the value of a new metric that is the Hellinger metric that is normalised by the uncertainty in the learnt graphical model. This distance computation using this new metric is a new output of my thesis. The learning of graphical models with uncertainties, is another new output from my work.

Returning to my interest in kernel parametrisation of covariance matrices, I have developed on the method of nesting GPs of different dimensionalities, to render covariance kernels non-stationary, by treating each kernel hyper-parameter as a realisation from a scalar-valued GP. This is discussed in the 2nd chapter. The inner layer of this learning strategy is then built of scalar-valued GPs, which are nested within a tensor-valued GP, and inference is done with MCMC.

Chapter Layout

The first chapter of this thesis is an introductory one. It introduces readers to contemporary theoretical ideas in Bayesian learning. Thus, the chapter begins with an exposition of what Bayesian supervised learning is, folding into itself, traditional regression, as well as contemporary Machine Leaping approaches, (while other forms of regression are dealt with later in the chapter), followed by the connection with covariance functions. As classification is another facet of such learning, discussion of classification methods is also included for completion. Similarly, unsupervised learning is also touched upon, though I have not worked in this area for my doctoral studies. The usage of GP-based models are particularly useful when learning in high-dimensions, so this is included as part of the chapter. The chapter ends with a discussion of graphical models which I have worked on extensively.

The content of the second chapter is summarised above. This chapter gives the detailed exposition on my doctoral work in the field of Bayesian supervised learning in high-dimensions. I discuss the GP-model first, and then bring in the added complication of modelling each of the hyper-parameters of the GP parameters, as realisations from another GP still – in order to model discontinuous data. Different ways of parameter prediction are then discussed. An application of the advanced method to such data is included in the chapter as well.

The content of the third chapter is also summarised above. This chapter discusses my work done on the learning of graphical models, at the learnt (partial) correlation structure of the data at hand. Having discussed the model and inference that I make on the random graph give the random partial correlation that is itself learnt given the data, I then discuss the method of computing inter-graph distance, and apply this to a simulated and a real data. Application to the learning of a large network is also included.

Contents

List of Figures

List of Abbreviations

AR	Autoregression
DAG	Directed Acyclic Graphs
DO	Disease Ontology
GP	Gaussian Process
HPD	Highest Probability Density
iid	independent and identically distributed
LHS	Left Hand Side
LOOCV	Leave-One-Out-Cross-Validation
MAP	Maximum a posteriori
MCMC	Markov Chain Monte Carlo
MW	Milky Way
NMPI	Normalised Pointwise Mutual Information
OU	Ornstein-Uhlenbeck
RHS	Right Hand Side
ROCAUC	Receiver Operating Characteristic Area Under the Curve
RQ	Rational Quadratic
r.v.	random variable
s.t.	such that
SQE	Squared Exponential

Chapter 1

Introduction

1.1 Bayesian Supervised Learning

Supervised learning is defined as the learning of functional relationship between variables given observations on these variables. So if the random variable X affects another random variable Y, the aim of supervised learning would be to learn the function $f(\cdot)$, where $Y = f(X)$–using the *training data*, if available, i.e. the set of observed pairs of values of X and the corresponding y. Once the function $f(\cdot)$ is learnt, we can predict the value of one of the variables that is realised at some newly measured value of the other variable.

For supervised learning to be accomplished, training data is necessary, as training data will constrain the sought form of the function $f(\cdot)$. In the Bayesian approach, the supervised learning problem is set in the paradigm of Bayes rule–the posterior probability density of the sought function is written, given the training data. The marginal posterior probability density of each unknown parameter–that characterises this functional form–is then computed, given the data. Such marginals allow us to place comprehensive and objective uncertainties (credible regions) on each unknown–this comprises the learning of the function. Prediction of the value of one of the variables at a newly measured value of the other can proceed by first writing the posterior predictive density of the unknown parameter

given the new measurement, and a model for the functional relationship between the two variables, where this function is itself learnt given the training data. One could also attempt at writing the joint posterior probability of the unknown value of one of the variables and the functional relationship between the two variables, given the new measurement on the other variable, and the training data. The incorporation of uncertainties in our prediction in the two different approaches is then differently undertaken, and makes interesting study.

Let $\{x_1, x_2..., x_n\}$ be a set of design points, i.e. values of X that comprise the training data that is available to learn function $f(\cdot)$ s.t. $f(X) = Y$. Let $\{y_1, y_2, ..., y_n\}$ be the observed values of Y, i.e. values of $f(X)$, where y_i is known to correspond to, i.e. is generated at x_i; $i = 1, \ldots, n$. Bayes rule suggests

$$p(f(\cdot)|Y) \propto p(Y|f(\cdot))p(f(\cdot))$$

Where $f(\cdot)$ is the functional relation we mention above, i.e. it is the mapping $f : \mathcal{X} \longrightarrow \mathcal{Y}$, where $X \in \mathcal{X} \subseteq \mathbb{R}$ and $Y \in \mathcal{Y} \subseteq \mathbb{R}$. Here $p(f(\cdot))$ is the prior of this function, bringing in information about the function $f(\cdot)$, into the model from before the collection of data. In parametric learning, this prior is presented as the prior probability density on the parameters of the parametric model of this function [Tibshirani, 2014]. In non-parametric learning, this probability can be related to the prior on hyper-parameters, or even on covariance parameters of the likelihood function [Christian, 1994]. Informative priors bring usable information into the model. Often, computational convenience motivates the prior and posterior probability densities to be conjugates of each other. On the other hand, priors can be non-informative, in which case, they do not imply preference for any identified subset of the possible models. [Gelman et al., 2014] The other factor in the right-hand-side (RHS) of the statement of Bayes rule above, is $p(Y|f)$ which is the likelihood of the

model of of $f(\cdot)$ given the measured values of the observable Y. In supervised learning, $f(\cdot)$ is assumed to be a one-to-one mapping from $\mathcal{X}$ to $\mathcal{Y}$ [Krishna B. Athreya, 2006]. $p(f(\cdot)|Y)$ is then the posterior probability density of function $f(\cdot)$, given observed value of Y. As with the prior density, the posterior probability of function $f(X)$ is presented as the posterior of parameters/hyper-parameters of the model. Thus, in this framework, we can get the probability distribution of the sought function $f(X)$ given the available training data, i.e. uncertainties of all learnt parameters are obtained.

1.1.1 Regression

The paradigm of supervised learning includes regression methods and classification methods. Regression is an exercise in estimating the function $f(\cdot)$, that as described above, is the mapping $f : \mathcal{X} \longrightarrow \mathcal{Y}$, with the input variable $\boldsymbol{X}$ now considered vector-valued: $\boldsymbol{X} \in \mathcal{X} \subseteq \mathbb{R}^m$ and $\boldsymbol{Y} \in \mathcal{Y} \subseteq \mathbb{R}^k$. Then $f(\boldsymbol{X}) = \boldsymbol{Y}$, where all measurement errors in $\boldsymbol{X}$ and $\boldsymbol{Y}$ are considered subsumed in this definition of the function $f(\cdot)$.

The simplest of the parametric models consider $f(\cdot)$ to be linear in $\boldsymbol{X}$, such that values of the observable $\boldsymbol{Y}$ can be expressed as

$$\boldsymbol{y} = \boldsymbol{x}\boldsymbol{\beta}^T + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon}$ is the value of the error or residual parameter and the matrix $\boldsymbol{\beta}$ of value of coefficients of model parameters is s.t. $\boldsymbol{\beta} \in \mathbb{R}^{(m\times k)}$. Let the residual parameter be $\in \mathbb{R}^k$ as is $\boldsymbol{Y}$. Learning the function $f(\cdot)$ then reduces to the problem of learning the coefficients, i.e. the elements of $\boldsymbol{\beta}$, as well as the parameters $\boldsymbol{\theta}_\epsilon \in \mathbb{R}^k$ of the probability distribution of the error $\boldsymbol{\epsilon}$, when such is not known. For example, when errors are modelled as Normally distributed, $\boldsymbol{\theta}_\epsilon$ reduces to a scalar, namely, the variance σ_ϵ^2 of the Normal error density. We make in-

ference on these unknown parameters by writing the posterior probability density of the unknown parameters given the data, $\pi(\beta_{11},\ldots,\beta_{1k},\beta_{21},\ldots,\beta_{2k},\ldots,\beta_{m1},\ldots,\beta_{mk},\boldsymbol{\theta}_\epsilon|\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^N)$, which by Bayes rule is proportional to the likelihood of these unknown parameters given the data and the prior probability density of these parameters. Then according to the Bayes rule, we can have:

$$p(\boldsymbol{\beta},\boldsymbol{\theta}_\epsilon|\boldsymbol{x},\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{x})p(\boldsymbol{\beta}|\boldsymbol{x})p(\boldsymbol{x}),$$

where $\boldsymbol{\beta} = \{\beta_{11},\ldots,\beta_{1k},\beta_{21},\ldots,\beta_{2k},\ldots,\beta_{m1},\ldots,\beta_{mk}\}$ and $p(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{x})$ is the likelihood of the parameters, given the data.

Indeed, the linear model can be extended to a polynomial model in general, and this has been addressed in the Bayesian setting by [Mitchell and Beauchamp, 1988; Seber and Lee, 2012; Box, 1973; Vaughn, 2008; Raftery et al., 1997].

In a polynomial model, the basis that the unknown $f(\boldsymbol{X})$ is written in terms of, is a polynomial in $\boldsymbol{X}$, of a chosen degree. The basis functions are then of the form $\boldsymbol{X}^0$, $\boldsymbol{X}^1,\ldots,\boldsymbol{X}^n$, with coefficients that are learnt. Likewise, other parametric models of the sought function $f(\cdot)$ can also be invoked [Albert and Chib, 1993; Goldstein, 2006].

The much more interesting domain is that of non-parametric regression, in which the sought function $f(\cdot)$ is not parametrised by some pre-defined model; rather, the function under consideration is treated as a realisation from a chosen Stochastic Process, s.t. the likelihood of this sought function given all data points, can be expressed by accumulating values of the underlying probability density of this process, computed at each datum in the available data set. [Rasmussen and Williams, 2006] In other words, the likelihood is that of the parameters of this Stochastic Process given all data. [MacKay, 1998] If the data points in the available data are *iid*, this accumulation of the underlying density of the process

reduces to the product over the whole data set of the density computed at each datum. In this treatment, the posterior probability density of the unknown function given the data is then proportional to the likelihood just discussed, multiplied by the prior probability density of the parameters of the stochastic process that $f(\boldsymbol{X})$ is considered to be a realisation of.

Good choices for this stochastic process are a Gaussian or a t-Process for reasons of computational convenience and generality of the ensuing model. [Rasmussen and Williams, 2006] In the later chapters, we will focus on our usage of the Gaussian process. An extra advantage of this approach is that it allows us to extend to dimensionality of choice, i.e. both for a scalar-variate unknown function of the design point variable, as well as for a high-dimensional, tensor-variate version of this function. In the former case, the likelihood of N realisations of the function–then scalar variate–is rendered Multivariate Normal, while in the latter case, the likelihood is rendered Tensor Normal. Indeed, errors in the learning of the function are reported within the adopted inference scheme, to perform such learning in this modelling paradigm. Measurement errors can also be accounted for by including the variance of the error density within the covariance structure of the Gaussian Process invoked to model the sought function.

A fundamental advantage of this approach is the flexible learning of the covariance structure that this offers in high-dimensional situations. Compared with standard fitting methods (such as spline fitting/wavelets based learning) that cannot appropriately capture covariance structures of high dimensional functions, the covariance function defined in the Gaussian process can efficiently achieve the correct covariance structure for high-dimensional function. [Chakrabarty et al., 2015] The covariance function is defined by the statistician, i.e. is chosen by us. Defining the covariance function is equivalent to defining the way to generate the covariance structure of the Gaussian process. Hence, an improper

choice of covariance function can lead the Gaussian process to be characterised by an incorrect covariance structure. The choice of this covariance function is then important in the Gaussian process based learning. Detailed discussions on the covariance function will be undertaken in Section 1.2.

Another important advantage of Gaussian Process regression is that it allows us to apply the prior distributions for all the model parameters and hyper-parameters. Furthermore, when defining the covariance function, it is easy to add uncertainty of the parameters into the covariance structure. Unlike the classical methods which the parameter uncertainty is often ignored or artificially incorporated, the uncertainty in the Gaussian process regression can be both defined by hand or leant as an outcome of the undertaken inferential scheme.

1.1.2 Gaussian Process regression

A Gaussian Process (GP) is a continuous valued stochastic process underpinned by the Gaussian probability distribution. Thus, in the scalar-variate case, the stochastic process $\{f(x)\}_{x\in\mathbb{R}}$, indexed by the deterministic variable X is a Gaussian process, if and only if, any finite set of realisations of this process, jointly follows the Multivariate Normal distribution. The mean function of this Gaussian process can be defined as

$$\mu(x) = \boldsymbol{E}(f(x))$$

In some applications, this mean function is commonly taken as the sample mean of the observations [Ley, 2016]

Modelling a Gaussian process, reduces to the problem of learning its covariance structure. The covariance function is defined as

$$K(x_i, x_j) = \boldsymbol{E}((f(x_i) - \mu(x_i))(f(x_j) - \mu(x_j)))$$

One possibility is to represent this covariance function as a parametric function of the input variables. Such a parametrisation then reduces the problem of learning the full covariance function, to the learning of the parameters of this chosen parametric model.

We seek the covariance function that leads to the learning of the function $f(\cdot)$. In the scalar variate case, i.e. when the input variable is $X \in \mathcal{X} \subseteq \mathbb{R}$, we can write:

$$\{f(x_1), \ldots, f(x_n)\} \sim \mathcal{N}(\underline{0}, \boldsymbol{K}),$$

which, by virtue of the equation $Y = f(X)$, is equivalent to:

$$\{y_1, \ldots, y_n\} \sim \mathcal{N}(\underline{0}, \boldsymbol{K}),$$

where $\mathcal{N}(\cdot, \cdot)$ is the multivariate normal density and $\boldsymbol{K}$ is the covariance matrix defined as $\boldsymbol{K} = [K_{ij}]$, with $K(i, j) = K(x_i, x_j)$, $i, j = 1, \ldots, n$.

Thus, in a regression exercise, looking at a Gaussian process as the distribution over the functions $f(\cdot)$ where $Y = f(X)$, it is the parameters of the covariance function that become the sought parameters. Choosing one covariance function over another, can then crucially affect the results of such a regression exercise. Commonly, the continuity of the data drives the choice of the covariance function. In the recent years, many researches have focused on defining different covariance functions for both continuous and discontinuous data. Further discussions on the covariance functions are presented in Section 1.2.

When the random variable Y is vector-valued (referred to as $\boldsymbol{Y}$), the function $f(\cdot)$ is also rendered vector-variate (referred to as $\boldsymbol{f}(\cdot)$). Then a finite number of values of this higher-dimensional function are jointly distributed as a Matrix-Normal distribution. In fact, we will discuss the general higher dimensional version of this situation–when $\boldsymbol{Y}$ is a tensor, so that $\boldsymbol{f}(\cdot)$ is a tensor-variate function, rendering the set of realisations of this function,

jointly Tensor-Normal. The m-th order Tensor-Normal density has m covariance matrices, so the learning of the function $f(\cdot)$ given the training data then involves the learning of these m covariance function. In Chapter 2 of the thesis, I will discuss this, supplemented by the prediction of X at which a new measurement on Y–i.e. the test data–is realised.

Gaussian processes are widely used in Bayesian supervised learning. Compared to parametric approaches to regression, Gaussian process-based regression is more flexible. In the scalar-variate case, when the Squared Exponential covariance function is chosen as the parametric model for the covariance structure of the Gaussian process, GP regression is equivalent to kernel regression with infinite Gaussian kernels [Rasmussen and Williams, 2006] The complexity of Gaussian process regression depends on the complexity of the covariance functions. A complex covariance function with multiple hyperparameters is hard to make inference on. Markov Chain Monte Carlo (MCMC) methods are best suited for inference in such high-dimensional state spaces. Once we learn the covariance matrices, we can write the posterior predictive distribution of the value of the input variable at which test data is realised, given this test data and our learnt model for the Gaussian process. As said in Section 1.1.1, we could alternatively write the joint posterior probability density of this sought value of X and the model for the Gaussian process, given test and training data. In principle inference is possible by maximising the posterior (MAP). But when the number of parameters that are sought is high, i.e. the state space is high-dimensional, manifesting a highly non-linear density, MCMC is a better alternative; importantly, MCMC techniques allow for an organic way of computing the marginal distribution of each of the unknown parameters, allowing for objective uncertainties (such as 95% Highest Probability Density–or HPD–credible regions) to be easily computed for each variable. Additionally Gaussian process regression allows prior distributions to be slapped on the hyper-parameters of the

covariance matrix. Thus, one important advantage of using MCMC based inference methods is that information on prior distribution of any unknown parameter can be incorporated into the inference, to ensure that the chain is avoiding sampling from irrelevant ranges of parameter values, thereby easing convergence.

1.2 Covariance Functions

As we discuss above, if an observable–multiple observations of which make up the data– is modelled as a realisation from a Gaussian process (GP), the relationship between such observations (or different components of such a generic high-dimensional observable), is modelled by the covariance functions of that GP. The selection of covariance function will affect the learning performed with the corresponding GP model. As with any stochastic process, a GP is referred to as stationary, if the joint probability density of n realisations from this GP (i.e. n measurements of this observable), is shift invariant with respect to translations in the input variable (that indexes the process, i.e. $\boldsymbol{X}$ for the purposes of our discussions).

1.2.1 Stationary covariance functions

Stationarity of stochastic processes can be weak or strong. Strong stationarity is defined above as the shift invariance of the joint probability $p(f(x_1),\ldots,f(x_n))$ of the n realisations from the GP, with respect to translations in X.

Weak stationarity on the other hand demands shift invariance only from the first two moments of the stochastic process, i.e. weak stationary for a Gaussian process over X is defined as:

$$\boldsymbol{E}(f(x)) = \mu_0,$$

and

$$K(x_i, x_j) = K(x_i + h, x_j + h)$$

where $K(x_i, x_j)$ is the covariance function computed at values x_i and x_j of X, and h is the value of the shift or the lag parameter. This would require that the covariance function of this GP is shift invariant, which would be ensured if the covariance function depends on the Euclidean distance between any two values of $\boldsymbol{X}$. In a non-stationary covariance function computed at $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the dependence on these values of $\boldsymbol{X}$ is not via the Euclidean distance between these vectors. In other words, for the covariance function, then, such a definition of stationarity means that the function is established on the distance $|x_i - x_j|$, which is non-negative. In the high dimensional cases, matrices generated by this distance function are always positive definite.

Thus, one advantage of stationary covariance functions is that the covariance matrices are rendered positive definite, as is required in computing square roots of the covariance matrices–needed in the computation of the density of the parameters given the data, i.e. in the computation of the likelihood. Covariance matrices that bear information about non-stationary covariance functions, are not necessarily positive definite. Indeed, one of the drivers that provides a constraint on the design of non-stationary covariance functions, is the demand on the ensuing covariance matrix to be positive definite. However, when the model is used to fit a discontinuous data set, the assumption of stationary covariance could be inappropriate [Neal and Nayfeh, 1990]. In that situation, using a stationary covariance function may lead to an incorrect covariance structure.

Stationary covariance functions abound in statistical literature [Rasmussen and Williams, 2006; Abramowitz and Stegun, 1964; Hensman et al., 2013]. The simplest such functional

form is the *power exponential family* [Biegler et al., 2011]:

$$K(x_i, x_j) = A \times exp\left[-\left(\frac{d}{\ell}\right)^{v}\right] \tag{1.1}$$

where $d := |x_i - x_j|$ and $\ell > 0$ is the scale-length parameter, while $A > 0$ is the amplitude parameter. The length-scale parameter ℓ controls how fast the covariance changes over a given distance d. This hyper-parameter then inversely determines the smoothness of the sample function. A sampled function marked by more similar functional values at two values of the input variable separated by a given distance d, than another, is smoother than the latter. Then in this smoother of the two functions, the distance in input space over which the functional value remains similar, is higher, i.e. in the smoother of the two functions, covariance declines more slowly for a given d, i.e. ℓ is lower. So we interpret ℓ as the inverse of the smoothing parameter, i.e. this scale-length parameter ℓ could also be referred to as the reciprocal of the smoothness parameter. In this definition, the parameter v is the power of the scaled distance between the two points at which covariance is being computed.

When $v = 2$, this function becomes the Squared Exponential function or SQE [Lawrence, 2003], which is one of the most commonly used covariance functions in supervised learning. This covariance function is infinitely differentiable and makes the model function very smooth. When the underlying data structure has a rough distribution, the SQE function cannot fit the model well, due to the strong smoothness assumption. The advantage of using the SQE covariance is that the hyperparameters are easy to estimate/learn. Compared with other covariance functions, the power exponential covariance function has the least number of hyperparameters [Rasmussen and Williams, 2006]. Learning the v and ℓ parameters is important when using the power exponential covariance–this can be done given the data,

using MCMC. Alternatively, if we knew the roughness of the function, we can determine the v parameter before we learn the other hyper-parameters.

However, the power exponential covariance function is less flexible than the Matern covariance which is defined by Rasmussen and Williams [2006] as:

$$K(x_i, x_j) = \frac{1}{\Gamma(v)2^{v-1}} \left(\frac{2\sqrt{v}d}{\ell} \right)^v K_v \left(\frac{2\sqrt{v}d}{\ell} \right), \quad d = |x_i - x_j|, \tag{1.2}$$

where $\Gamma(v)$ is the Gamma function of parameter $v > 0$ [Sebah and Gourdon, 2002], and $K_v(\cdot)$ is the Modified Bessel function with parameter v. When $v = 0.5$, the Matern covariance function transforms to the power exponential covariance function.

Using this covariance function, we can model an Ornstein-Uhlenbeck (OU) process in 1 dimensional case which is a stochastic process defined as [Jacobsen et al., 1996]:

$$X_t = c + pX_{t-1} + \epsilon_t$$

Where X_t is the observed value of OU process at index t and ϵ_t a random variable that follows the normal distribution. In time series analysis, we refer to the OU process as the AR(1) process. Generally, when the hyper-parameter v in the Matern covariance has the form: $v + 1/2 = p$, where $p \in \mathbb{Z}$, the continuous AR(p) process results.

When $v \to +\infty$, the Matern covariance function converges to a squared exponential covariance function (SQE). It is the parameter v that controls the roughness of the sampled function. Since the functions learnt from the Matern covariance do not need to be infinitely differentiable, the Matern covariance is a more general model than the SQE covariance. In fact, the observed function is not differentiable for $v = 0.5$. For $v > 7/2$, the function sampled from a GP characterised by the Matern covariance, will be very similar to the function sampled when using the SQE covariance. Thus, we see that the Matern covariance

function is more flexible than the power exponential function, as we discussed above, in the sense that the Matern covariance has more hyper-parameters to control the smoothness of the estimated function. For $v = 0.5$, the estimated function can be very rough.

Rational Quadratic (RQ) Covariance Function is another widely used covariance function [Berger et al., 2001]. This function can be written as:

$$K(x_i, x_j) = \left(1 + \frac{d^2}{2\alpha\ell^2}\right)^{-\alpha}, \quad \text{where} \quad d := |x_i - x_j|$$

This kind of covariance function can be regarded as an infinite sum of SQE covariance functions with an inverse gamma prior placed on the length scale ℓ. This is clarified in the following. Assume $r = \ell^{-2}$ follows the Gamma distribution. We then have the posterior of this inverse squared length parameter to be

$$p(r|\alpha, \beta) \propto r^{\alpha-1} exp(-\alpha r/\beta),$$

using which, we can marginalise over the r dependent covariance kernel, to give the kernel function to be:

$$K(x_i, x_j) = \int p(r|\alpha, \beta) K(x_i, x_j|r) dr$$

$$\propto \int r^{\alpha-1} exp\left(-\frac{\alpha r}{\beta}\right) exp\left(-\frac{rd^2}{2}\right) dr \propto \left(1 + \frac{d^2}{2\alpha l^2}\right)^{-\alpha}, \tag{1.3}$$

so that when $\alpha \longrightarrow \infty$, the RQ covariance function will approach the SQE covariance function with length scale ℓ. In this sense, the RQ covariance can be regarded as a more general form of the SQE covariance. In this section, we have introduced three basic covariance functions. These are: power exponential covariance function, Matern covariance function and rational quadratic covariance function. However, it is easy to use these covariance functions to then define other, new covariance functions. To achieve this, we use the

property that the sum and product of the stationary covariance function are also valid covariance functions [Rasmussen and Williams, 2006]. Thus, the sum of the two covariance functions give: [Hastie and Tibshirani, 1990]

$$f(x) = f_1(x) + f_2(x); \qquad K(x_i, x_j) = K_1(x_i, x_j) + K_2(x_i, x_j)$$

where $f_1(x)$ and $f_2(x)$ are two different and independent functions that are sampled from GPs that are characterised by the covariance functions $K_1(\cdot,\cdot)$ and $K_2(\cdot,\cdot)$ respectively. This equation can then be used to combine the covariance functions with different length scales.

Again, the product of the covariance functions yields another covariance function, as given by:

$$f(x) = f_1(x)f_2(x); \qquad K(x_i, x_j) = K_1(x_i, x_j)K_2(x_i, x_j).$$

If the functions $f_1(x)$ and $f_2(x)$ are sampled from Gaussian processes, the product of the two functions is not a Gaussian process necessarily. However, the Gaussian process with the covariance function $K(x_i, x_j)$ can be regarded as an approximation that characterises the GP, that function $f(x)$ is sampled from [Park and Choi, 2010; Seo et al., 2000].

Thus, combinations of stationary covariance functions are usually valid covariance functions that can still be stationary covariance function. However, if a non-stationary covariance function is added to a stationary covariance function, the new covariance function will be a non-stationary covariance [Rasmussen and Williams, 2006]. Importantly, the adding of non-stationary covariance functions does not lead to a valid covariance function in general.

1.2.2 Non-stationary covariance functions

In general, the assumption of stationarity is an appropriate approximation for a continuous data [Shinozuka and Deodatis, 1991]

However, when the data is not continuous, assuming stationarity of the underpinning Gaussian process is not correct. The concept of the covariance kernel is to connect proximity in values of the input variable $\boldsymbol{X}$, to that in values of the function, where the functional value is also given by the value of the output variable, i.e. the observable $\boldsymbol{Y}$. Stationary covariance functions are structured to establish this connection between the proximity of the values of $\boldsymbol{X}$ and $\boldsymbol{Y}$. Then if it so happens, that the data in $\boldsymbol{Y}$ is discontinuous, then proximity–or rather the lack of the same–in values of $\boldsymbol{X}$ is no longer connected directly to the lack of proximity in values of $\boldsymbol{Y}$ that is manifest in the discontinuous data. In other words, then, stationary covariance kernels do not provide adequate parametrisation of the covariance structure.

The stationary covariance function is defined as a function of the Euclidean distance. The important characteristic of stationary covariance functions is that it generates a symmetric matrix as the covariance matrix for the Gaussian process. Given a stationary covariance kernel, such as the Matern class of covariances, the simplest way to counter stationarity is to use a non-diagonal matrix for the length parameters. [Paciorek and Schervish, 2004] Then recalling the Matern covariance function from Equation 1.2, we can have a non-stationary form of Matern covariance as:

$$K(x_i, x_j) = \frac{|\boldsymbol{\Sigma}_i|^{\frac{1}{4}}|\boldsymbol{\Sigma}_j|^{\frac{1}{4}}}{\Gamma(v)2^{v-1}}\left|\frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2}\right|^{-\frac{1}{2}}\left(\frac{2\sqrt{v}d}{\ell}\right)^{v} K_v\left(\frac{2\sqrt{v}d}{\ell}\right), \quad d = |x_i - x_j|,$$

where $\boldsymbol{\Sigma}_i$ and $\boldsymbol{\Sigma}_j$ are the covariance matrices at x_i and x_j. As with the stationary Matern covariance, when $v \to +\infty$, the above non-stationary covariance function converges to the

SQE form:

$$K(x_i, x_j) = (x_i - x_j)^T \left(\frac{(\mathbf{\Sigma}_i + \mathbf{\Sigma}_j)}{2} \right)^{-1} (x_i - x_j)$$

.

Another simple non-stationary covariance function is the inner product covariance function [MacKay, 1998]

$$K(x_i, x_j) = \sigma_0^2 + \sigma_1^2 x_i \cdot x_j$$

If $\sigma_0^2 = 0$, this covariance function reduces to being stationary. This covariance function can be easily extended in terms of basis functions. Assume that we have a basis function ($\phi(X)$) of X, We can then define a linear kernel as:

$$K(x_i, x_j) = \sigma_0^2 + \sigma_1^2 \phi(x_i)\dot{\phi}(x_j)$$

However, deciding the number of basis functions is also a challenge when using this kernel. For non-linear datasets, one may use a large number of basis functions. On the other hand, this makes the computation of the kernel time-consuming and unrealistic in applications. One way of solving this problem is to assume an infinite number of basis functions and integrate them out.

It is also easy to extend stationary covariance functions to periodic non-stationary covariance functions. Inspired by the SQE covariance function discussed above in Equation 1.1, one form of the periodic covariance function can be written as:

$$K(x_i, x_j) = A \times exp\left(-\frac{2sin^2(\frac{x_i - x_j}{2})}{\ell^2} \right).$$

This can help us achieve a quasi-periodic covariance. This covariance is widely used when analysing time series analysis.

The Wiener process (Brownian motion) can also be presented as a Gaussian process with special covariance function [Revuz and Yor, 2013]. Brownian motion is defined as a continuous-time stochastic process:

$$X_t = \mu t + \sigma W_t$$

Where X_t is the observed data at time t and W_t follows the normal distribution $N(0,t)$. The covariance function for the Brownian motion can be written as:

$$K(x_i, x_j) = \min(x_i, x_j)$$

Assume $B(x)$ is a standard Brownian motion. The covariance between $B(x_i)$ and $B(x_j)$ is:

$$Cov(B(x_i), B(x_j) = E[B(x_i)B(x_j)] - E[B(x_i)]E[B(x_j)] = E[B(x_i)B(x_j)]$$

When $x_i > x_j$, we can have:

$$E[B(x_i)B(x_j)] = E[B^2(x_j)] - E[B(x_j)(B(x_i) - B(x_j))]$$

$$= E[B^2(x_j)] = x_j$$

Similarly, when $x_i < x_j$, we can have $E[B(x_i)B(x_j)] = x_i$. Thus, this covariance function yields results that are not based on the Euclidean distance, and therefore, this covariance function is a non-isotropic and non-stationary covariance function.

1.3 Other regression methods

Kernel regression is a non-parametric regression method used in Statistics and machine learning. The simplest version of it is the Nadaraya-Watson kernel regression method

[Koistinen and Holmström, 1992; Diack, 1999]. In Nadaraya-Watson regression, the kernel function $k(\cdot,\cdot)$ is defined as in:

$$y(x) = \frac{\sum_{i=1}^{n} k(x, x_i) y_i}{\sum_{i=1}^{n} k(x, x_i)}$$

These kernel functions can be regarded as the distance between the observation (y_i) and prediction ($y(x)$). Thus, these kernel functions are by definition, non-negative functions. Like in other nonparametric regression methods, the parameters of the kernel functions serve up to be the hyperparameters in the model. However, the optimisation of those hyperparameters is hard to achieve. One widely used way to make inference on the hyperparameters is Markov Chain Monte Carlo (MCMC). When dealing with complex kernel functions, the usage of MCMC might be time consuming. Since the kernel function is user-defined, it is easy to control the number of hyperparameters in the kernel function. Here are some examples of kernel functions:

Gaussian Kernel:

$$k(x_i, x) = \exp(-r(x_i - x)^2)$$

Multiquadric Kernel:

$$k(x_i, x) = \sqrt{1 + r(x_i - x)^2}$$

Inverse quadratic:

$$k(x_i, x) = \frac{1}{1 + r(x_i - x)^2}$$

The Gaussian kernel is also used in the local linear regression. In that case, the regression function can be written as:

$$f(x) = \sum_{i=1}^{N} (a_i + b_i(x - c_i)) K(x, c_i)$$

where a_i and b_i are the weight parameters for the linear function. c_i is the mid-point of the i-th kernel function. Parameters of the kernel regression can be estimated using classical methods, such as least squared estimation. However, a higher order linear term can also be added into this function. When choosing the appropriate Gaussian kernel, local regression can give a good approximation for the observed function. Normally, this approximation is smoother than the original function. This is because the Gaussian kernel will smooth the observed data to some extent. Choosing a small bandwidth parameter will release this problem, though constraining the bandwidth using the available data is not straightforward, i.e. the effect of this choice can affect results in an arbitrary way.

Kernel functions are widely used in regression and classification problems. Using the kernel functions, the non-linear observed data can be presented as a linear dataset in a high dimensional Euclidean space. Thus, the regression/classification problem can be solved by the learning in the transformed Euclidean space. In machine learning, many supervised learning algorithms are developed based on kernel function [Nasrabadi, 2007; Hofmann et al., 2008].

Usage of kernel functions can be extended to ridge regression, or more specifically, the kernel ridge regression [Vovk, 2013]. Ridge regression has the advantage of being a fast method, and can be used as an online algorithm. The cost function for the classical ridge regression can be written as:

$$C(W) = \sum_i (y_i - W^T x_i)^2 + \lambda ||W||^2$$

where W is the vector of parameters in the linear regression. and $\lambda ||W||^2$ is the penalty cost function.

We aim to minimise the cost function. The solution for the parameter vector W can be

written as:

$$W = (\sum_i x_i x^T + \lambda I)^{-1} \sum_i x_i y_i$$

Then the function $f(x)$, once placed into a high dimensional feature space, where the function is transformed to $\phi(x)$, the cost function can be written as:

$$C(W) = \sum_i (y_i - W^T \phi(x_i))^2 + \lambda ||W||^2$$

Then, the solution for the new cost function is written as:

$$W = (\sum_i \phi(x_i)\phi(x_i)^T + \lambda I)^{-1} \sum_i \phi(x_i) y_i$$

We can re-write $K(x_i, x_j) = \phi(x_i)\phi(x_j)^T$ where $K(x_i, x_j)$ is the kernel function. The advantage of using the kernel ridge regression is that it releases the over-fitting problem of kernel regression. However, the parameter in the penalty cost function is hard to estimate, though this can be tackled using cross validation or leave-one-out estimation. These validation techniques are now briefly discussed. Cross-validation is a technique that is used to validate a model, i.e. to determine the applicability of a model to any new data set that is distinct from the training data that was used to train this model in the first place. In the current context, "applicability" of the model refers to its ability to predict, given a new (or test) data set, where the model itself was inferred upon given the training data. Thus, cross-validation can detect concerns such as overfitting. It can be carried out by scanning across different partitions of the whole of the available data set, into a training part and another mutually exclusive test (or validation part), such that these two parts exhaust the considered data. So for example, the leave-p-out-cross-validation technique works by scanning across partitions of the whole data set into a validation data that comprises p data points, and a training data that is built of the remaining data points; for every considered partition,

prediction using the p-sized validation data is undertaken, subsequent to the training of the model on the rest of the data points. Thus, there are $\binom{n}{p}$ number of partitions that the technique will have to rotate through, for an original data set that comprises n data points. When $p = 1$, the method is referred to as leave-one-out-cross-validation (LOOCV). Indeed, then $\binom{n}{1=n}$ takes less time than general cross-validation (i.e. for non-unit p), but in spite of this less computational expense, LOOCV estimates might have higher variance [Hastie et al., 2009], though isses regarding variance are also affected by n. It is also possible to undertake non-exhaustive cross-validation in which not all ways of splitting the original sample are undertaken.

With the help of high-dimensional kernels, kernel regression methods can also be extended into high dimensional Euclidean space. For three-dimensional data, this kind of problem is known as surface fitting. However, dimensionality of state space can be much higher, as in the problems that we discuss later in Chapters 2 and 3. A well known kernel regression method is the support vector machine.

We can use the kernel density estimation to approximate the probability density function for a random variable. Usually, this kind of approximation is used in complex, low-dimensional (1 to 2 dimensional) distribution, in which the density function is hard to parametrise. The kernel density estimator can be calculated as:

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

where $f(x)$ is the estimated density function and $K(\cdot)$ is the kernel function for the estimated density. h is the smoothing parameter which is also called the bandwidth parameter. A good choice of bandwidth can make the function more accurate with less kernels. The most common optimality criterion used to select this parameter is the expected risk func-

tion. However, the bandwidth parameter can also be learnt using MCMC.

Neural networks are another set of powerful methods useful for supervised learning [Demuth et al., 2014; Rowley et al., 1998; Specht, 1991]. The idea of a neural network comes from modelling the brain. All neural network algorithms are based on a group of neural functions. Each neural function is a simple function that can be user-defined, where the general form of the neural network can be written as:

$$f(x) = K(\sum_i w_i g_i(x))$$

where $f(x)$ is the function defined on the observation x. $K(\cdot)$ is the activation function for the neural functions, $g_i(\cdot)$ is the neural function and w_i is the weight for the neural function. For the simplest neural network, the neural function will be connected in one layer. This makes the function quite similar to a kernel regression function. However, the complex neural network algorithms will have many layers. The activation function $K(\cdot)$ makes the neural network model flexible–using the activation function, the relationship between the different neural functions can be expressed as a direct graph.

Learning of the neural network function is a problem of reducing the cost function. Choosing an appropriate cost function can make the model much easier to fit and get less number of layers of neural functions. One simplest cost function is quadratic:

$$C = E[(f(x) - y)^2]$$

where C is the cost function and $f(x)$ is the predicted value from the neural network function. y is the observed value of function $f(x)$. Once the cost function is minimised, the neural network function is correctly estimated.

Most neural networks have a specified, fixed number of neural functions. However,

increasing the number of hidden neurals to infinity is possible. In that case, the neural network is equivalent to a Gaussian Process with a certain covariance kernel [Neal, 2012].

1.4 Unsupervised Learning

Distinguished from supervised learning (in which a training data set is required), the main characteristic of unsupervised learning is that it does not have a training data set to calibrate the model, [Hastie et al., 2009]. Assume an observed data on an input space variable: $\boldsymbol{D} = \{x_1, \ldots, x_n\}$. Then supervised learning models require measurements $\{y_1, \ldots, y_n\}$ on a related dependent variable $\boldsymbol{Y}$, in order to train the model. This variable $\boldsymbol{Y}$ can be a desired class label in classification or the value of a variable in regression models. The basic idea behind supervised learning is to learn the functional relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$: $\boldsymbol{Y} = f(\boldsymbol{X})$, where a model for $f(\cdot)$ is trained using the available training data.

In unsupervised learning, we only have the observed data set $\boldsymbol{X} = \{x_1, \ldots, x_n\}$ but no data on the dependent variable $\boldsymbol{Y}$. Unsupervised learning algorithms often focus on modelling the underlying data structure, instead of trying to learn the function $f(\cdot)$.

A commonly used unsupervised learning method is clustering-based [Coates et al., 2011; Weber et al., 2000; Cios et al., 2007]. From the observed data $\boldsymbol{D}$ on $\boldsymbol{X}$, where $\boldsymbol{D} = \{x_1, \ldots, x_n\}$, the clustering methods aim to find the partition $\boldsymbol{S} = \{S_1, \ldots, S_k\}$ of the data set, where k is the number of subsets of the observed data [Hartigan and Hartigan, 1975]. Then a function is defined over the number of partitions variable. One such method that seeks k, is the k-means method, in which the idea is to perform partition of the set of observations within the nearest mean [Wagstaff et al., 2001].

As the unsupervised learning methods are oblivious to a training dataset, almost all the unsupervised learning methods avoid over-fitting problems. However, the main concern in

unsupervised learning is the accuracy of the model. This is primarily caused by the lack of training data set.

1.5 Methods for modelling tensor-variate data

Tensors are widely used in physics, mathematics, and often when image processing is undertaken. We refer to $\boldsymbol{Y} \in \mathbb{R}^{k_1 \times \ldots \times k_{m-1}}$ as a tensor of order $m-1$. Indeed, often in real life applications, an observable is found to be tensor-valued, so that n measurements of this $m-1$-th order tensor-valued observable is then an m-th order tensor itself. A tensor of order 1 is a vector; one of order 0 is scalar; a tensor of order 2 is a matrix–though not all matrices are tensors of order 2 of course. This is evident in a simple example that follows. Let us define a vector $\boldsymbol{A} \in \mathbb{R}^d$ expressed in the basis $S = \{e_1, \ldots, e_d\}$, as $\boldsymbol{A} = \boldsymbol{M} \cdot \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} \in \mathbb{R}^d$ and the matrix $\boldsymbol{M} \in \mathbb{R}^{d \times d}$, then if on changing the basis to $S^{/} = \{e_1^{/}, \ldots, e_d^{/}\}$, $\boldsymbol{\alpha}$ changes to the vector $\boldsymbol{\alpha}^{/}$, and matrix $\boldsymbol{M}$ changes to the matrix $\boldsymbol{M}^{/}$, s.t. $\boldsymbol{M}^{/} \cdot \boldsymbol{\alpha}^{/}$ yields $\boldsymbol{A}^{/}$. If we find $\boldsymbol{A}$ to have changed to $\boldsymbol{A}^{/}$ in the same way as $\boldsymbol{\alpha}$ changes to $\boldsymbol{\alpha}^{/}$, then matrix $\boldsymbol{M}$ is a tensor (of order 2).

The outer product or tensor product (for vectors, matrices and tensors) is defined as follows. Outer product of 2 vectors:

Let $\boldsymbol{A}$ be an m-dimensional vector: $\boldsymbol{A} = (A_1, \ldots, A_m)^T$.

Let $\boldsymbol{B}$ be an ℓ-dimensional vector: $\boldsymbol{B} = (B_1, \ldots, B_\ell)^T$.

Then the outer product:

$$A \otimes B = \begin{pmatrix} A_1B_1 & A_1B_2 & \dots & A_1B_\ell \\ A_2B_1 & A_2B_2 & \dots & A_2B_\ell \\ \vdots & \vdots & \ddots & \vdots \\ A_mB_1 & A_mB_2 & \dots & A_mB_\ell \end{pmatrix}$$

Outer product of 2 matrices:

Let $\boldsymbol{A}$ be an $m_1 \times m_2$-dimensional matrix: $\boldsymbol{A} = [A_{ij}]$.

Let $\boldsymbol{B}$ be an $\ell_1 \times \ell_2$-dimensional matrix: $\boldsymbol{B} = [B_{pq}]$.

Then the outer product

$$[A_{ij}B_{pq}e_{ip}^{jq}],$$

where the matrix $\boldsymbol{e} := [e_{ip}^{jq}]$, s.t. $\boldsymbol{e}$ has dimensions of $m_1\ell_1 \times m_2\ell_2$, with the entry of 1 in the $(i-1)\ell_1 + p$-th row and the $(j-1)\ell_2 + q$-th column, and entries of 0 elsewhere.

Thus, the outer product of $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$ is $\begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}a_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}a_{21} & a_{22}b_{22} \end{pmatrix}$

A tensor $\boldsymbol{A}$ of order m with dimensions $d_1 \times \ldots \times d_m$, will form an outer product with the tensor $\boldsymbol{B}$ that is of order ℓ, with dimensions $h_1 \times \ldots \times h_\ell$, to produce the tensor $\boldsymbol{C}$ of order $m + \ell$, with dimensions $g_1 \times \ldots \times g_{m+r}$, s.t.

$$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B} \quad C_{dh} = A_d B_h.$$

On the other hand, inner products of tensors are more simply defined.

The inner product of two tensors of order m and ℓ is either $m + \ell$-2 or 0, whichever is greater.

The inner product of two 2nd-ordered tensors, i.e. matrices is then the standard matrix product in which the product of two matrices $\boldsymbol{A}$ that is $m_1 \times m_2$-dimensional and $\boldsymbol{B}$ that is $\ell_1 \times \ell_2$-dimensional, is allowed only if $m_2 = \ell_1$ and the inner product is itself a matrix of dimensions $m_1 \times \ell_2$. Many multi-variate distributions can be extended to tensor-variate distributions [Xu and Yan, 2015; Guilleminot and Soize, 2010]. The computation of the likelihood will then involve tensor products. An example of this is the extension of the multivariate normal distribution (parametrised by a mean vector and covariance matrix) [Hoff et al., 2011] to the the tensor normal distribution (discussed again in chapter 2); if the latter distribution (in the tensor-valued variable $\mathbf{D} \in \mathbb{R}^{k_1 \times \ldots \times k_m}$) is m-th order tensor normal, then it is parametrised by a mean $\boldsymbol{\mu} \in \mathbb{R}^{k_1 \times \ldots \times k_m}$ that is an m-th order tensor, and m covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m$. The density of this distribution [Basser and Pajevic, 2003] is

$$f(\mathbf{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m) \propto$$

$$\exp\left(- \parallel (\mathbf{D} - \boldsymbol{\mu}) \times_1 \boldsymbol{A}_1^{-1} \times_2 \boldsymbol{A}_2^{-1} \ldots \times_m \boldsymbol{A}_m^{-1} \parallel^2 /2\right), \tag{1.4}$$

where

$$\boldsymbol{\Sigma}_j = \boldsymbol{A}_j \boldsymbol{A}_j^T$$

, $j = 1, \ldots, m$, i.e. $\boldsymbol{A}_j$ is the unique square root of the j-th covariance matrix.

$$(\boldsymbol{\mathcal{Y}} \times_i \boldsymbol{X})_{p_1 p_2 \ldots p_{i-1} q_i p_{i+1} \ldots p_k} := \sum_{p_i} y_{p_1 p_2 \ldots p_{i-1} p_i p_{i+1} \ldots p_k} x_{q_i r_j},$$

where the ij-th element of matrix $\boldsymbol{X}$ is $x_{q_i r_j}$, $q_i = 1, \ldots, q$, $r_j = 1, \ldots, r$. The $p_1 p_2 \ldots p_k$-th element of tensor $\boldsymbol{\mathcal{Y}}$ is $y_{p_1 p_2 \ldots p_{i-1} p_i p_{i+1} \ldots p_k}$. As an example, if in Equation1.4, $m = 3$, i.e.

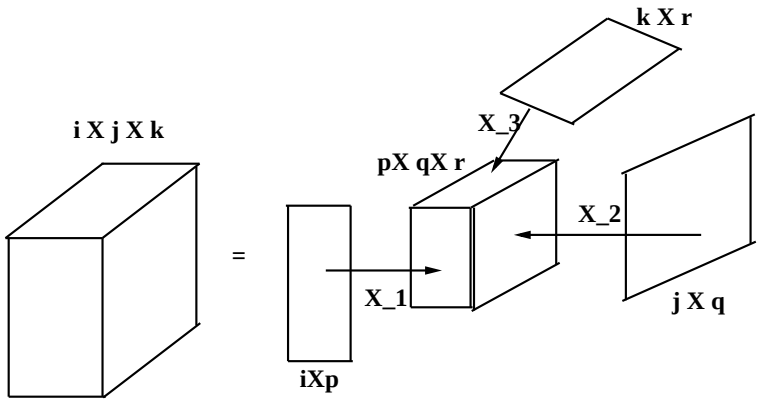

Figure 1.1: Cartoon to represent the sequential multiplication of 3 matrices with a core tensor of order 3, to convert the original core tensor dimensions to that of the final 3-rd ordered tensor. Th $\mathbf{X}_i$ in the figure represents the mode-i multiplication between a tensor and a matrix, $i = 1, 2, 3$.

if the tensor $\mathbf{D}$ is a 3rd ordered tensor, (and so is $\boldsymbol{\mu}$ then), multiplying $\mathbf{D} \in \mathbb{R}^{p\times q\times r}$ with a matrix $\mathbf{B} \in \mathbb{R}^{s\times q}$ via mode-2 multiplication, will yield the tensor $\mathcal{C} \in \mathbb{R}^{p\times s\times r}$:

$$\mathcal{C} = \mathbf{D} \times_2 \mathbf{B},$$

$$c_{ijk} = \sum_t d_{itk} b_{jt}.$$

Thus, in this equation, by such mode-c multiplications–$c = 1, 2, 3$ in this example–the core tensor $\mathbf{D} - \boldsymbol{\mu}$ that is say of dimensions $p \times q \times r$, gets tranformed into a tensor of dimensions $i \times j \times k$, where the dimensions of the square, covariance matrices $\boldsymbol{\Sigma}_c$ (and therefore, the $\boldsymbol{A}_c$) matrices are $i \times i$, $j \times j$, and $k \times k$, respectively, for $c = 1, 2, 3$ [De Lathauwer et al., 2000; Ben-Israel and Greville, 2003; Kolda and Bader, 2009]. This can be visually expressed in Figure 1.1.

Following this general idea of mode multiplications of tensors and matrices, one can then express a k-th ordered tensor $\boldsymbol{X}$ as a "unit" k-th ordered tensor $\boldsymbol{Z}$, multiplied via

mode-i multiplications to k number of square matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k$:

$$\boldsymbol{X} = \boldsymbol{Z} \times_1 \boldsymbol{A}_1 \times_2 \boldsymbol{A}_2 ... \times_k \boldsymbol{A}_k.$$

Hoff et al. [2011] introduced this form of expression of a general tensor in terms of matrix-valued "factors" and uses the maximum likelihood estimation to calculate those matrix-valued factors. However, the maximum likelihood estimation is not the only solution to get the matrix valued factors. Learning directly using MCMC is another possibility.

1.5.1 Tensor-variate Gaussian Process

Assume the $k-1$-dimensional random variable is modelled by a tensor-variate Gaussian process; then collating the measurements of this random variable, we obtain the available data $\boldsymbol{D}$ on this variable, which turns out to be a k-th ordered tensor. Then from the defiition of Gaussian Processes, joint of all these measurements of this variable, i.e. the probability of this k-th ordered data tensor $\boldsymbol{D}$, is the k-dimensional tensor normal distribution. We can write:

$$\boldsymbol{D} \sim \mathcal{TN}(\boldsymbol{M}, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k),$$

where $\boldsymbol{M}$ is the mean tensor of the tensor-Normal density and $\boldsymbol{\Sigma}_p$ is the p-th covariance matrix of this density; $p = 1, \ldots, k$. The bc-th element of $\boldsymbol{\Sigma}_p$ is given by the covariance between the b-th and c-th slices of the data, where each of these slices of the data tensor is realised at a point along the p-th direction in input space. Thus,

$$p(\boldsymbol{D}|\boldsymbol{M}, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k) \propto \exp(-\|(\boldsymbol{D} - \boldsymbol{M}) \times_1 \boldsymbol{A}_1^{-1} ... \times_k \boldsymbol{A}_k^{-1}\|^2/2)$$

where the covariance matrix $\boldsymbol{\Sigma}_p = \boldsymbol{A}_p \boldsymbol{A}_p^T$, the array norm $\|\boldsymbol{X}\|^2$ is calculated as: $\|\boldsymbol{X}\|^2 = \sum_{i_1} ... \sum_{i_k} x_{i_1..i_k}^2$.

The square matrix $\boldsymbol{A}_p$ can be any of the square root of the positive semi-definite covariance matrix $\boldsymbol{\Sigma}_p$. As the covariance matrices are defined as positive semi-definite, the Cholesky decomposition is one of the solution for the square root matrices ($\boldsymbol{A}_p$) [Higham, 1990]. In the two dimensional cases, the tensor variance normal distribution will converge into the matrix normal distribution and $\boldsymbol{A}_p$ will be the square root of the two covariance matrices. Hoff et al. [2011] used a plugin estimate of these covariance matrices in the application he undertook. However, plugin estimates approximate the truth, with sample-size crucially affecting the value. On the other hand these covariance matrices can be sampled from MCMC, as long as $\boldsymbol{\Sigma}_p$ is not too high in its dimensionality. If its size is big ($\gtrsim 10 \times 10$), we will need to fall back on plugin estimates; else, learning the elements of $\boldsymbol{\Sigma}_p$ directly fom MCMC is possible. If $\boldsymbol{\Sigma}_p$ is such that we know the value of the input space variable at which the b-th slice is realised, we can express the covariance between the b-th and c-th slices of the data as a decreasing parametric function of the distance in input space between these slices, i.e. the difference between the input space variable value at which the b-th slice is realised, and the value at which the c-th slice is realised. Here, the b-th and c-th slices of the data $\boldsymbol{D}$ are as defined above, as slices realised at two aribtarily chosen points along the p-th direction in input space.

1.5.2 Other tensor-variate distributions

Many distributions can be extended into their tensor variate version; Xu and Yan [2015] extended the t-distribution to tensor variate t-distribution and developed Bayesian models with the tensor variate t-distribution. The t-distribution can be regarded as a sum of Gaussian distributions. Usually, when we have less observations, the t-distribution can get better results than the normal distribution. For a k-dimensional tensor $D \in \mathbb{R}^{n_1 \times \ldots \times n_k}$, the

tensor-variate t-distribution can be defined as:

$$p(\boldsymbol{D}|\boldsymbol{M},\boldsymbol{\Sigma}_1,...,\boldsymbol{\Sigma}_k) = \frac{\Gamma(\frac{n+v}{2})\prod_{p=1}^{k}|\Sigma_p|^{-\frac{n}{2n_p}}}{\Gamma(\frac{v}{2})(v\pi)^{\frac{n}{2}}}(1+\frac{1}{v}\|(\boldsymbol{D}-\boldsymbol{M})\times_1 \boldsymbol{A}_1^{-1}...\times_k \boldsymbol{A}_k^{-1}\|^2)^{-\frac{1}{2}(n+v)}$$

where the covariance matrix $\boldsymbol{\Sigma}_p = \boldsymbol{A}_p\boldsymbol{A}_p^T$, the array norm $\|\boldsymbol{X}\|^2$ is calculated as: $\|\boldsymbol{X}\|^2 = \sum_{i_1}...\sum_{i_k} x^2_{i_1..i_k}$. The parameter v is the degree of freedom in the tensor variate t-process and $v > 2$.

As the tensor variate Gaussian process, the parameters and covariance matrices in the tensor variate t-process can be estimated by the maximum likelihood estimation. However, a method for learning the covariance matrices is sampling using MCMC.

When undertaking tensor classification, another important method is the tensor-variate kernel function [Zhao et al., 2014]. Since the labels for the tensor classification are usually scalar, tensor classification can be regarded as a functional learning problem in which we write: $Y = f(\boldsymbol{X})$, where Y is the label parameter for the classification and $\boldsymbol{X}$ is the observed tensor. In this situation, one method is to model the observation $\boldsymbol{X}$ using a tensor-variate stochastic process and use the covariance function to map the label values y into this stochastic function. This method is discussed in the section 1.4.1. However, another method is to model the label values as a stochastic process. Since the labels are just scalar, we can use a classic stochastic process to build the model. As we still need to build the one-to-one mapping between labels and the tensor-variate observation. The input for the covariance function will be the tensor variate observation. This kind of covariance function is called tensor variate kernel. Much work has been done to build the tensor variate kernel function. The simplest way to build such kernel is based on the Chordal distance [Signoretto et al., 2011]. Here are some examples for the tensor variate kernel in Chordal distance:

Linear kernel: $k(\boldsymbol{X}_i, \boldsymbol{X}_j) = \vec{(}X_i)\vec{(}X_j)'$

Gaussian kernel: $k(\boldsymbol{X}_i, \boldsymbol{X}_j) = \exp(-\frac{\|\boldsymbol{X}_i - \boldsymbol{X}_j\|^2}{l})$

Where $\boldsymbol{X}$ is the observed tensor variable and l is the hyper-parameter in the kernel function. As discussed in the section 1.2. The covariance functions above are stationary covariance functions since the covariance structure are only based on some distance measurement. As most of the stationary kernel, the advantage of using this kernel is it simplifies the covariance structure. The hyper-parameters in the covariance function are easy to learn. However, the covariance structure is more flexible when modelling the observation directly as a tensor variate process.

1.6 Graphical models

The graph of a given data is an intuitive and illustrative way of representing the correlation structure of the data. So if the data set at hand is shaped like a matrix–composed of n measurements of an observable $\boldsymbol{X} \in \mathbb{R}^p$–then we could construct one graph that represents the correlation amongst the columns of this matrix, and another to represent that amongst the rows of this matrix. The graph $\mathbb{G}(\boldsymbol{V}, \boldsymbol{E})$ consists of nodes, s.t. the i-th node represents the i-th component of the variable $\boldsymbol{X} = (X_1, \ldots, X_p)^T$, with the *vertex set* $\boldsymbol{V} = \{1, 2, \ldots, p\}$. The edges connecting any pair of nodes then is an element in the set $\boldsymbol{E}$. An edge may or may not exist between the i-th and j-th nodes,where $i \neq j; i, j \in \boldsymbol{V}$. In a random graph, i.e. when the graph $\mathbb{G}(\boldsymbol{V}, \boldsymbol{E})$ is itself a random variable, it is possible for the ij-th edge (edge between the ij-th pair of nodes) to exist with some probability ϕ that is a constant irrespective of i and j–such graphs are referred to as *homogeneous graphs*. On the other hand, the probability for the ij-th edge to exist could be ϕ_{ij}, with the matrix $\boldsymbol{\Phi} = [\phi_{ij}]$, i.e. the probability for an edge to exist varies from one pair of nodes to another, as would be the

case in *inhomogeneous graphs*. In light of this, we could change our notation for the graph to $\mathbb{G}(\boldsymbol{V}, \boldsymbol{\Phi})$–this is the notation that we use henceforth. Then a random inhomogeneous graph defined over the vertex set $\boldsymbol{V}$, would be a member of a family of graphs, or a graphical model $\mathcal{G}_{\boldsymbol{V}}(\boldsymbol{\Phi})$, i.e. $\mathbb{G}(\boldsymbol{V}, \boldsymbol{\Phi}) \in \mathcal{G}_{\boldsymbol{V}}(\boldsymbol{\Phi})$. The learning of the graphical model given a data set, can be undertaken Bayesianly or within a frequentist approach. The main classes of graphs include factor graphs, directed graphs, undirected graphs. Common examples of directed graphs are Bayesian networks, while common examples of undirected graphs are Markov networks. Above, one idea that we presented without examination is in regard to the shape of the data set. We said that the data is matrix-shaped above, comprised of n observations of the p-dimensional vector-valued variable $\boldsymbol{X}$. But observed variables are not necessarily vectors–in fact, as we saw in Section 1.5, in general, the observed variable could be a tensor, rendering the data set a tensor as well (one order up on the order of the observable tensor). In such a case, a graphical model that presents the correlation structure of the data is high-dimensional too. But that would render the visualisation of the graphical model difficult, and in fact, defeat the whole purpose of presenting correlation structures via graphs, which as we say above, are quick and easy illustrations of this correlation structure. Thus, in such cases, we could learn the graphical model of each matrix-shaped slice of this tensor-shaped data, and then learn a distance between the other pairs of independent slices of this data. For example, if the data were cuboidal-shaped, i.e. a 3rd-ordered tensor with dimensions n, p, k, then we could learn the graphical model of each of the k, $n \times p$-dimensional matrices, and then find the distance between the graphical model learnt for the m-th and $m^{/}$-pairs; $m, m^{/} = 1, \ldots, k$; $m \neq m^{/}$. One possible such distance (or affinity measure) that can be computed between Bayesianly learnt graphical models, is the Hellinger distance which we discuss in Section 3. So the purpose of this section is to discuss

the construction of a link between the dependence structure of the data and its graphical representation [Sachs et al., 2005], as proxied by conditional independence (denoted by $\perp_P$) between the components X_i and X_j of the observable $\boldsymbol{X}$, and "graphical separation" (denoted with $\perp_G$), where such "separation" is defined (*in terms of cliques in undirected graphs, eg. Markov networks, or in terms of parents in directed graphs, eg. Bayesian networks*). The mapping from the space of such dependency structure of the data to the space of graphical model is called a D-map, while the inverse mapping, i.e. the mapping of the space of graphical separation, into the space of independence amongst components of the observable, is referred to as an I-map.

A graph $\mathbb{G}(\boldsymbol{V}, \boldsymbol{E})$ is a dependency-map, i.e. D-map of the probabilistic dependence structure P of $\boldsymbol{X}$, if there is a one-to-one correspondence between the random variables $X_1, \ldots, X_p$ and the nodes in $\boldsymbol{V}$, s.t. for all disjoint subsets $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ of $\boldsymbol{X}$ we have

$$A \perp_P B|C \Longrightarrow A \perp_G B|C$$

.

Again, $\mathbb{G}(\boldsymbol{V}, E)$ is an independency-map (or I-map) of P if

$$\boldsymbol{A} \perp_P \boldsymbol{B}|\boldsymbol{C} \Longleftarrow \boldsymbol{A} \perp_G \boldsymbol{B}|\boldsymbol{C}$$

.

Being an I-map guarantees that two disjoint sets of nodes A and B found to be separated by another set C in the graph (according to the characterisation of separation for the class of graph under consideration) correspond to independent sets of variables, i.e. components of $\boldsymbol{X}$. If the graphical model given a data is both a D and I-map, then it is called a "perfect" map; P is then isomorphic to $\mathbb{G}(\cdot, \cdot)$. If the dependence structure P of $\boldsymbol{X}$ can be expressed

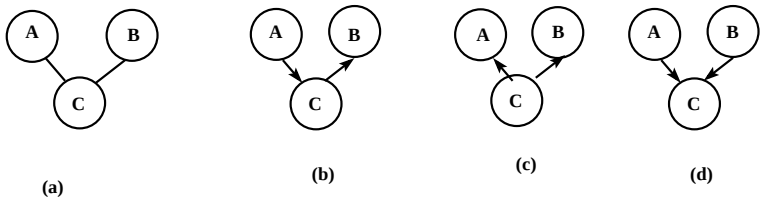

Figure 1.2: Panel (a) represents an undirected graph; (b), (d) and (c) DAGs each, with panel (b) representing a serial graph, (c) a diverging and (d) a converging graph.

by multiple graphs, we must use the one with the minimum number of edges–this defines a *minimal I-map*.

Above, we have mentioned the idea of "separation", and indicated that the definition of separation varies with the class of graphical models that is pertinent (i.e. if the graph is directed or undirected, etc.). We define this for the two main classes first and examine the idea thereafter using some simple examples. In undirected graphs, if $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are three disjoint subsets of nodes, then $\boldsymbol{C}$ is said to separate $\boldsymbol{A}$ from $\boldsymbol{B}$, (denoted $\boldsymbol{A} \perp_G \boldsymbol{B}|\boldsymbol{C}$), if every path between a node in $\boldsymbol{A}$ and a node in $\boldsymbol{B}$ contains at least one node in $\boldsymbol{C}$.

In directed acyclic graphs, for the same three disjoint subsets of nodes $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are , $\boldsymbol{C}$ is said to d-separate $\boldsymbol{A}$ from $\boldsymbol{B}$, (denoted $\boldsymbol{A} \perp_G \boldsymbol{B}|\boldsymbol{C}$), if in every path between a node in $\boldsymbol{A}$ and a node in $\boldsymbol{B}$, there is a node v that is in $\boldsymbol{C}$ where $\boldsymbol{C}$ has no converging edges, or, where none of the descendants from v is in $\boldsymbol{C}$.

Let us look at the example of an undirected graph (panel (a) of Figure 1.2) and of 3 basic directed acyclic graphs or DAGs (panels (b), (c), (d), of Figure 1.2).

Let us examine the P structure, i.e. the joint probability distribution of $x_1, \ldots X_p$ in each of these examples.

- (a) $A \perp B|C \Longrightarrow \Pr(A, B, C) = \Pr(A|C)\Pr(B|C)\Pr(C)$.

- (d) $A \not\perp B|C \implies \Pr(A,B,C) = \Pr(C|A,B)\Pr(B)\Pr(A)$.
- (c) $A \perp B|C \implies \Pr(A,B,C) = \Pr(B|C)\Pr(A|C)\Pr(C)$, which is the same as
- (b) $A \perp B|C \implies \Pr(A,B,C) = \Pr(B|C)\Pr(C|A)\Pr(A)$.

Thus we see that for each of the directed graphs, the global probability distribution can be factorised at each node in terms of the parents of that node. Factorisation of the global distribution is also possible in undirected graphs, but this is fundamentally different from the way factorisation proceeds in DAGs. In DAGs, it is the Markov property that defines decomposition of the global distribution of the data into a set of local distributions which are each represented as the probability of the local node conditional on the parents, i.e.

$$\Pr(\boldsymbol{X}) = \prod_{i=1}^{p} \Pr(X_i|\Pi_{X_i})$$

where Π_{X_i} are the parents of X_i. Note that the local distributions are given by the distribution of the single node X_i and the joint distribution of its parents. However, in undirected graphs, the Markov property that defines decomposition of the global distribution of the data into a set of local distributions is clique-by-clique. Here, by a *clique* we imply the maximal subset of nodes in which each element is adjacent to the others, where *adjacency* is defined as being exactly 1-edge away. Thus, a clique is a complete sub-graph. Then, in a graph that has k cliques $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$,

$$\Pr(\boldsymbol{X}) = \prod_{C_i \in C} \psi(\boldsymbol{C}_i)$$

where $\psi\cdot$ is a *potential* function that has a non-negative value representing the relative probability mass of a clique. This potential reduces to a proper pdf only if the graph is *decomposable*, or *chordal*, i.e. any cycle of length ≥ 4 in the graph has a *chord*, i.e. an

edge between 2 nodes that is not in the cycle itself. Then in an undirected graph, the global distribution factorises as

$$\Pr(\boldsymbol{X}) = \frac{\prod_{C_i \in \mathcal{C}} \Pr(\boldsymbol{C}_i)}{\prod_{S_i \in \mathcal{S}} \Pr(\boldsymbol{S}_i)},$$

where $\boldsymbol{S}_i$ is a (minimal) separator, i.e. a set of nodes s.t. removal of $\boldsymbol{S}_i$ from the graph will separate the graph into two connected components $\boldsymbol{C}_i$ and $\bar{\boldsymbol{C}}_1$ such that each vertex in $\boldsymbol{S}_i$ is adjacent to some vertices in $\boldsymbol{C}_i$ as well as some in $\bar{\boldsymbol{C}}_1$. Then going back to example on simple Markov network as the graph in panel (a) of Figure 1.2, we see that this Markov network has 2 cliques: $C_1 = \{A, C\}$ and $C_2 = \{B, C\}$, separated by the separator $S_1 = \{C\}$. Then the above factorisation for this undirected graph implies

$$\Pr(\boldsymbol{X}) = \frac{\Pr(A, C) \Pr(B, C)}{\Pr(C)} = \Pr(A|C) \Pr(B|C) \Pr(C),$$

as we know this probability to be.

1.6.1 Graphical models via Gaussian process

Thus we see clearly that the learning of the graphical model of a given (matrix-shaped) data set is reliant upon the learning of the correlation between the vectors $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_j$, where $i, j = 1, \ldots, p$, with $\boldsymbol{Y}_i$ the n-dimensional vector that comprises the n number of measurements of the observed variable X_i; $\boldsymbol{X} = (X_1, \ldots, X_p)$. The data $\mathbf{D}$ is then $n \times p$-dimensional. Then if we defined the "between-columns" correlation matrix $\boldsymbol{\Sigma}_C = [S_{ij}]$, it is the ij-th element of $\boldsymbol{\Sigma}_C$ that tells us about the covariance S_{ij} between the X_i and X_j. Of course, if the data is standardised (by its empirical mean and variance), this covariance matrix is equivalent to a correlation matrix. The conditional dependencies that comprise the joint probability distribution of variables X_1 to X_p, are however equivalent to *partial*

correlations, since the effect of X_i conditional on X_j is sought, with the effect of all X_m held fixed, where $m \neq j$; $m = 1, \ldots, p$, in a generic example. Thus, in order to learn the graphical model of the data, we need to learn the partial correlation matrix, any non-diagonal element of which gives the partial correlation between a pair of distinct columns of the given data. One way to achieve this is by learning the between-columns correlation matrix of the data, and then converting the correlation matrix into the partial correlation matrix–using which the graphical model of the data is learnt. As we saw in Section 1.1.2, It is possible to learn the correlation structure of the data by modelling the observable $\boldsymbol{X}$ (n realisations of which comprise the data $\mathbf{D}$) using some underlying generative process, such as the Gaussian process or a t-process. Choosing to model $\boldsymbol{X}$ as a realisation from a GP of corresponding dimensions, it then implies that the joint probability distribution of the n realisations of $\boldsymbol{X}$ must be the matrix-normal GP that is parametrised by the matrix-valued mean $\boldsymbol{\mu}$ and the two covariances: between-columns covariance $\boldsymbol{\Sigma}_C$ and between-rows covariance $\boldsymbol{\Sigma}_R$. But the the n realisations of $\boldsymbol{X}$ constitute the data $\mathbf{D}$. In other words, the probability distribution of the data is matrix-normal, i.e. the likelihood of $\boldsymbol{\mu}, \boldsymbol{\Sigma}_C$ and $\boldsymbol{\Sigma}_R$, given the data $\mathbf{D}$is matrix-normal. While likelihood maximisation techniques can in principle be invoked at this stage in frequentist approaches, in the Bayesian framework, this likelihood is used–along with judiciously chosen priors–to define the posterior of the unknown GP parameters given the data. Posterior sampling using MCMC-based inference schemes can be undertaken to give rise to the joint posterior probability density of all unknowns given the data, which can be used to compute the marginal posterior probability density of each unknown given the data. Such marginal distributions can then be used to compute the 95% Highest Probability density credible regions on each unknown given the data. Once the between-columns correlation $\boldsymbol{\Sigma}_C$ is learnt in this way, it can be implemented

to help learn the graphical model of the data. However this approach is still silent about how to learn this graphical model with Bayesianly interpretable uncertainties on the graph that are learnt given the data, where the graph itself being a random variable, so that its full probability distribution should be inferred upon in a Bayesian setup. Importantly, a robust way of propagating the uncertainties in the learning of $\mathbf{\Sigma}_C$ given the data–especially in the presence of measurement uncertainties–is not presented in the literature. We will address these shortcomings in Chapter 3.

Chapter 2

Tensor variate Gaussian process

2.1 Introduction

Statistical modelling allows for the learning of the relationship between two variables, where the said relationship is responsible for generating the data available on the variables. It then follows that such learning should be inclusive of the errors made in predicting the value of one of the variables, given the other. Thus, let $\boldsymbol{X}$ be a random variable that represents a behavioural or structural parameter of the system, and $\boldsymbol{Y}$ is another variable that bears influence on $\boldsymbol{X}$ in either the sense that if value of $\boldsymbol{X}$ is changed, $\boldsymbol{Y}$ changes, or in the sense that if $\boldsymbol{X}$ changes, then a set of other variables: $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_n$ change, and the change in $\boldsymbol{W}_i$, causes the value of $\boldsymbol{Y}$ to change, where $i \in \{1, \ldots, n\}$. In either case, if we are to represent the relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ as: $\boldsymbol{Y} = f(\boldsymbol{X})$, then the functional relation $f(\cdot)$ that we seek to learn, is endowed with information about the error made in predicting the values of $\boldsymbol{Y}$ (or $\boldsymbol{X}$) at which the noise-included measurement of $\boldsymbol{X}$ (or $\boldsymbol{Y}$) has been realised. In other words, the errors of the learning of the sought functional relationship $f(\cdot)$, are considered subsumed within $f(\cdot)$.

In the above discourse, we did not commit to the structure of the two variables. In general, either of both variables could of course be tensor-valued, such that, data comprising

measurements of either variable, is then shaped as a hypercuboid. Typically, the structure/behaviour of a system is parametrised using a set of scalar-valued parameters, (say d number of such parameters), which can, in principle be collated into a d-dimensional vector. In other words, in the jargon motivated in the last paragraph, the random variable $\boldsymbol{X}$ is a d-dimensional variable. We write, $\boldsymbol{X} \in \mathcal{X} \subseteq \mathbb{R}^d$. This is the typical structure of the system parameter vector $\boldsymbol{X}$.

The other, observed variable $\boldsymbol{Y}$, that–as we say above–bears influence on $\boldsymbol{X}$, can be tensor-valued in general. Let $\boldsymbol{Y}$ be a k-th ordered tensor-valued variable, i.e. is $m_1 \times m_2 \times \ldots \times m_k$-dimensional, where $m_i \in \mathbb{Z}, \forall\, i = 1, \ldots, k$. Then we write, $\boldsymbol{Y} \in \mathcal{Y} \subseteq \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_k}$.

Given this structure of $\boldsymbol{Y}$, and the schematic relationship $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X})$ between $\boldsymbol{X}$ and $\boldsymbol{Y}$, we realise that the sought function $\boldsymbol{f}(\dot{)}$ is a map of the following nature; $\boldsymbol{f} : \mathcal{X} \subseteq \mathbb{R}^d \longrightarrow \mathcal{Y} \subseteq \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_k}$, i.e. $\boldsymbol{f}(\cdot)$ is itself a high-dimensional function. To be precise, it is a k-ordered tensor-variate function of dimensions $m_1 \times m_2 \times \ldots \times m_k$.

What does this "tensor-variate" function $\boldsymbol{f}(\cdot)$ mean? Learning $\boldsymbol{f}(\cdot)$ is in fact equivalent to learning all the $\prod_{i=1}^{k} m_i$-number of component functions, with these components suffering inter-correlations. Thus, the learning of $\boldsymbol{f}(\cdot)$ is equivalent to learning the multiple functions, and such learning is inclusive of learning the correlation amongst these component functions.

If, in the definition of $\boldsymbol{Y}$ as a k-ordered tensor-valued r.v., k=2, i.e. $\boldsymbol{Y}$ is a vector ($\in \mathbb{R}^q$, say), then $\boldsymbol{f}(\cdot)$ would be rendered a vector-variate function, i.e. a function with q number of component functions, $f_1(\cdot), \ldots, f_q(\cdot)$, s.t. at $\boldsymbol{X} = \boldsymbol{x}_i$, $f_j(\boldsymbol{x}_i) = y_j^{(i)}$, $j = 1, \ldots, q$; $i = 1, \ldots, N$, where the q-dimensional vector $\boldsymbol{y}_i = (y_1^{(i)}, \ldots, y_q^{(i)})^T$, and the correlation amongst $y_1^{(i)}, \ldots, y_q^{(i)}$ is the same as that amongst $f_1(\boldsymbol{x}_i), \ldots, f_q(\boldsymbol{x}_i)$. Now I

seek the inverse of this function $\boldsymbol{f}(\cdot)$, We will define this as the q-dimensional vector of the inverses of the component functions. So to find the value $\boldsymbol{x}^{(new)}$ of the d-dimensional $\boldsymbol{X}$, at which a new (measured) value $\boldsymbol{y}^{(new)}$ of $\boldsymbol{Y}$ is realised, we compute:

$$\boldsymbol{x}^{(new)} := \boldsymbol{f}^{-1}(\boldsymbol{Y})|_{\boldsymbol{Y}=\boldsymbol{y}^{(new)}}.$$

Here, the q-dimensional vector $\boldsymbol{y}^{(new)} := (y_1^{(new)}, \ldots, y_q^{(new)})^T$. Then I will in fact have to operate the inverse of the j-th component function on the j-th component Y_j of $\boldsymbol{Y}$, and compute this at $Y_j = y_j^{(new)}$, and do this $\forall j = 1, \ldots, q$. The first point that we realise is that, for the above equation to hold, and offer solutions, we require $d \leq q$. Secondly, the equation fundamentally represents an over-determined system, and unique solutions are not expected unless $d = q$, though, correlation amongst the q-number of component functions is s.t. $f_j^{-1}(y_j^{(new)})$ and $f_{j'}^{-1}(y_{j'}^{(new)})$ ensure consistency within the uncertainty levels in the values of each component of $\boldsymbol{x}^{(new)}$. The over-determinedness is not expected to always be exactly compensated by the learnt correlation structure, but in a Bayesian setting, the non-uniqueness of the solutions only contributes to inflating the 95% Highest Probability Density credible regions on the learnt function.

The learning of this "tensor-variate" function discussed above, is of course challenging. In particular, capturing the correlation structure amongst the component functions of such a high-dimensional function is a daunting task. We realise that the correlation amongst the different components of $\boldsymbol{f}(\cdot)$, at each realisation of the variable $\boldsymbol{X}$, is in fact, synonymous to the correlation amongst the different components of the realisation of $\boldsymbol{f}(\boldsymbol{X})$, i.e. of $\boldsymbol{Y}$ (by virtue of the equation $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X})$). Thus, learning the correlation structure of the data on $\boldsymbol{Y}$ is closely related to us seeking the form of the function $\boldsymbol{f}(\cdot)$. Of course, we do not want to just stop at learning this function, but thereafter, employ it, to predict a value for $\boldsymbol{X}$

(or $\boldsymbol{Y}$) at which, a recorded value of $\boldsymbol{Y}$ (or $\boldsymbol{X}$) is realised.

In summary, we state that interest in learning such a high-dimensional function $f(\cdot)$, given hypercuboidally-shaped data comprising multiple measurements of a tensor-valued variable $\boldsymbol{Y}$, is indeed an exercise involving "big data", i.e. data that is "big" in the sense that it is high-dimensional, as manifest in its hypercuboidal morphology. But one may ask why we would not resort to seek the vector-variate function that takes the tensor-valued variable $\boldsymbol{Y}$ as the input, and offer a vector-valued output, namely, $\boldsymbol{X}$. Why indeed do we instead opt to undertake the harder task of learning the higher-dimensional, tensor-variate function $f(\cdot)$ that takes the vector $\boldsymbol{X}$ as the input, and outputs the tensor-valued $\boldsymbol{Y}$?

This is an important question–in fact, fundamental to the study that I undertake during the main part of my Ph.D.

The answer to this question relates mainly to the question of why we want to learn the function $f(\cdot)$ that describes the functional relationship between the vector-valued $\boldsymbol{X}$, and the tensor-valued $\boldsymbol{Y}$? Having learnt the functional relationship between the variables, ultimately, we would like to predict one variable, given a measured value of another. So the question posed in the last paragraph reduces to the following: is it feasible to invert a (learnt) mapping that maps a high-dimensional domain to a lower dimensional one, or is the converse true?

To answer this question let us consider the pair of ways in which we can express the functional relationship betwen two differently dimensional, example random variables, S and $\boldsymbol{V}$. Let us consider the mapping from the space of d-dimensional vectors to that of reals, as $g : \mathbb{R}^p \longrightarrow \mathbb{R}$, so that, if the r.v. $\boldsymbol{V} \in \mathbb{R}^p$ and the r.v, $S \in \mathbb{R}$, we can write $S = g(\boldsymbol{V})$. Then assuming that $g(\cdot)$ is learnt, prediction of the value $s^{(new)}$ of S at which a measured value $\boldsymbol{v}^{(new)}$ of $\boldsymbol{V}$ is realised, is possible as $s^{(new)} := g(\boldsymbol{V})|_{\boldsymbol{V}=\boldsymbol{v}^{(new)}}$. Similarly,

if we want to predict the value of $\boldsymbol{V}$, at which a new datum on S has been measured, we would think that this sought value of $\boldsymbol{V}$ can imply be defined as $\boldsymbol{v}^{(new)} := g^{-1}(S)|_{S=s^{(new)}}$. However the RHS of this suggested equation is a scalar, while the LHS is a p-dimensional vector, i.e. this equation is valid only for $p = 1, 0$. Thus it is clear that inverse prediction of the high-dimensional argument (as compared to a lower-dimensional output) of a learnt function is not possible.

However, if we were to consider the mapping $\boldsymbol{h} : \mathbb{R} \longrightarrow \mathbb{R}^p$, so that, if the r.v. $\boldsymbol{V} \in \mathbb{R}^p$ and the r.v, $S \in \mathbb{R}$, we can write $\boldsymbol{V} = \boldsymbol{h}(S)$. Then if the vector-variate function $\boldsymbol{h}(\cdot)$ is learnt, prediction of the value of $\boldsymbol{V}$ at which a measured value of $\boldsymbol{S}$ is realised, is possible, using the forward operation of $\boldsymbol{h}(\cdot)$ on S, computed at this new measured value. Also, the inverse prediction of the value $s^{(new)}$ of S, at which a measured value $\boldsymbol{v}^{(new)}$ of $\boldsymbol{V}$ is realised, is given by: $s^{(new)} := \boldsymbol{h}^{-1}(\boldsymbol{V})|_{\boldsymbol{V}=\boldsymbol{v}^{(new)}}$; this is indeed an overdetermined system, so multiple solutions exists, but the situation stands mitigated in light of the fact that not all the p equations (that this single equation includes) will bear independent information, since the p-number of component functions of $\boldsymbol{h}(\cdot)$ are correlated. However in the modelling of the function with a higher-dimensional input than output (as in the example function $g(\cdot)$ discussed in the previous paragraph), the solution to the inverse prediction does not exist.

Thus, using this example of the two ways according to which we can exress the relation between a pair of example random variables, that are differently dimensional, we realise that we need to define our sought functional relationship between the vector-valued $\boldsymbol{X}$ and the tensor-valued $\boldsymbol{Y}$, as $\boldsymbol{f}(\cdot)$, where $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X})$, (and not as the unknown function $\boldsymbol{f}^{-1}(\cdot)$), so that we are in principle, not incapacitated from performing prediction of either variable, given a measured value of the other. This holds, even though, it is harder to learn the tensor-variate function $\boldsymbol{f}(\cdot)$, than the lower-dimensional, vector-variate function

$\boldsymbol{f}^{(-1)}(\cdot)$, i.e. there are more component functions to be learnt. This is a fundamental concept in inverse problems.... In addition, there are other advantages of modelling the sought functional relationship as $\boldsymbol{f}(\cdot)$, than $\boldsymbol{f}^{(-1)}(\cdot)$, such as inclusion of measurement uncertainties; we discuss this in greater detail below. We now discuss the connection of the sought learning and prediction, with the availability of data.

To understand how data guides a model for the sought function $\boldsymbol{f}(\cdot)$, let us consider the simple scalar-variate case first. The learning of the function $f(\cdot)$–where, $Y = f(X); f : \mathbb{R} \longrightarrow \mathbb{R}$–is possible, as long as we have access to pairs of (N number of) known values of X, and the corresponding Y, i.e. the training dataset $\{(x_i, y_i)\}_{i=1}^{N}$. Then we can in principle learn which curve $f(\cdot)$ is to be passed through these points that comprise the training data. Thus, the training data consists of pairs of: *chosen* or *designed* value of X, and the corresponding value of Y that maybe is known by some means–empirical observations, experiments, surveys, etc. The upshot is that training data is available, i.e. pairs of (x_i, y_i) are known, and we want to find the function $f(\cdot)$ that is defined via $Y = f(X)$.

Similarly, in the situation that we discuss above, in which $\boldsymbol{X}$ is a vector-valued and $\boldsymbol{Y}$ is a tensor-valued r.v., s.t. $\boldsymbol{f}(\cdot)$ is tensor-variate–we learn the unknown function as long as we have access to the training data $\mathbf{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$. Our ulterior aim is the prediction of either of $\boldsymbol{Y}$ or $\boldsymbol{X}$, given *test data* on $\boldsymbol{X}$ or $\boldsymbol{Y}$ respectively. In conventional framework, the inverse prediction of $\boldsymbol{X} = \boldsymbol{x}^{(test)}$, at which test datum $\boldsymbol{y}^{(test)}$ on $\boldsymbol{Y}$ is realised, is undertaken as:

$$\boldsymbol{x}^{(test)} := \boldsymbol{f}^{-1}(\boldsymbol{Y})\big|_{\boldsymbol{y}^{(test)}}$$

. However, this formulation renders the learning of the uncertainty in the prediction difficult, and there is no clear way on including the uncertainties learnt in the learning of the

function $\boldsymbol{f}(\cdot)$, to propagate into the uncertainty of this prediction. This leads us to consider Bayesian prediction of one variable, given test data on the other, subsequent to the Bayesian learning of the function $\boldsymbol{f}(\cdot)$. Before we discuss this, let us better motivate the need for a Bayesian framework.

Indeed, the curve-fitting exercise could be handled using fitting based on Ordinary Least Squares, or fitting that employs splines/wavelets. However, such fitting methods fumble under multiple realistic complications. Firstly, the measured values of both/either of the r.v.s X and Y, can be accompanied by measurement errors; in light of this, it becomes difficult to infer the function that fits the data the best. In fact, the uncertainty in the learning of the sought function is also then difficult to quantify–how to identify the set of functions that "best fit" the training data? Secondly, how do we judge what the smoothness of the curve should be? So for scalar-valued r.v.s X and Y, that are related via the sought $f(\cdot)$, how do we know what the smoothness of the curve should be between the successive points (x_i, y_i) and (x_{i+1}, y_{i+1}), $(i = 1, \ldots, N-1)$, in the training data? Ideally, we would prefer to learn this smoothness from the data itself. However, there is nothing intrinsic to the fitting-with-splines/wavelets method that can in principle, quantity the smoothness of the curve, given a training data. Lastly, when Y is an r.v. that is no longer a scalar, but higher-dimensional (say tensor-valued in general), fitting with splines/wavelets starts to become useless, since in such cases of sought tensor-variate function $\boldsymbol{f}(\cdot)$ (in general), the component functions of $\boldsymbol{f}(\cdot)$ are correlated, but methods such as parametric fitting approaches, cannot capture such correlation, given the training data. As we have remarked above, such correlation amongst the components functions of $\boldsymbol{f}(\cdot)$ is the same correlation structure amongst the components of the tensor-valued $\boldsymbol{Y}$–so in principle, the sought correlation can be learnt from the training data.

In this case, given the general vector-valued $\boldsymbol{X}$, and the tensor-valued $\boldsymbol{Y}$ (of order k, say), the sought function is rendered a k-th ordered tensor-variate function. One objective way of learning the set of functional forms for the relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$, i.e. for $\boldsymbol{f}(\cdot)$, is then to sample a function from a "bag of functions", in which functions exists with varying posterior probabilities given the data at hand. But what is such a "bag of functions", if not a Stochastic Process, for Stochastic Processes are distributions over function space. So, we identify a relevant Stochastic Process that can give a general, non-restrictive description of the sampled functions–a Gaussian Process for example–and for a set of functions sampled from the identified process, we will attempt to compute the posterior probability density of this set of realisations from this Stochastic Process, given the training data. Of course, the parameters of the Process that generates these functions are not fixed arbitrarily, but are learnt from the data, so that learning these Process parameters then enables us to generate the very functions that are compatible with the data at identified probabilities. In other words, when we speak of computing the posterior probability density of a set of functions sampled from the Process, it is synonymous to us computing the posterior probability density of the Process parameters given the data. Generating samples from this posterior density then allows for the identification of the 95% HPD credible regions on these Process parameters, i.e. on the learnt function $\boldsymbol{f}(\cdot)$. It is possible to learn the smoothness of the function generated from this Process, via parameterisation of the covariance kernels of the covariance structure of the Stochastic Process under consideration. Thus, one of the major sources of worry for us will be the pursuit of adequate covariance kernel parametrisation.

Once the Process parameters are learnt (via sampling from the posterior probability density of these parameters given the data), the inverse prediction of the value of $\boldsymbol{X}$, given test data on $\boldsymbol{Y}$, can be considered by writing the posterior probability density of $\boldsymbol{X}$ given

the test data, and the learnt model of the function $f(\cdot)$, i.e. the parameters of the Stochastic Process from which this function is generated. This can be undertaken by two ways: either by writing the posterior predictive density of $\boldsymbol{X}$ given the test data and the learnt Process parameters, or by writing the joint posterior probability density of the sought value of $\boldsymbol{X}$ and all Process parameters, given test and training data. Our favoured method of generation of posterior samples, namely MCMC techniques, then allows for the learning of the marginals of each sought parameter, and the corresponding 95% HPD credible interval. The latter approach (sampling from the joint posterior density) acknowledges errors of learning and prediction well.

A last point that remains to be addressed in this introductory section of the 2nd chapter, is the relevance of high-dimensional data. Are there really hypercuboidally-shaped data that show up in real-world applications, that we need to find the correlation structure of, and/or, perform learning+prediction with? The answer is an emphatic yes [Mardia and Goodall, 1993; Bijma et al., 2005; Werner et al., 2008; Theobald and Wuttke, 2008; Barton and Fuhrmann, 1993]. For example, in computer vision, the image of one person might be a matrix of dimensions $a \times b$, i.e. image with resolution of a pixels by b pixels. Then, repetition across n persons inflates the data to a cuboidally-shaped dataset. Examples of handling high-dimensional datasets within computer vision exist [Dryden et al., 2009; Fu, 2016; Pang et al., 2016; Wang, 2011; Qiang and Fei, 2011]. In health care, the p number of health parameters of n patients, when charted across k time-points, again generates a high-dimensional data, which gets further enhanced, if the experiment involves tracking for changes across ℓ groups of n patients each, where each such group is identified by the level of intervention [Chari, Thu, Wilson, Lockwood, Lonergan, Coe, Malloff, Gazdar et al., 2010; Chari, Coe, Vucic, Lockwood and Lam, 2010; Clarke et al., 2008; Oberg et al., 2015; Sarkar,

2015; Wang et al., 2015; Fan, 2017]. That we treated these groups as independent–or for that matter, even the variation in health parameter values of any group across the k time points, is ignored, and a mere snapshot of each group at a given time is all that is traditionally considered–is a shortcoming, of the traditional modelling strategies. We are advancing a method for the consideration of parameters across all relevant levels within one integrated framework, to enable the learning of correlations across all such levels, thus permitting the prediction of a health parameter (of a patient at some desired time and identified patient group), with meaningful uncertainties and avoiding information loss associated with categorisation of data. Again, in ecological datasets, there could be n spatial locations at each of which, p traits of k species could be tracked, giving rise to a high-dimensional data [Leitao et al., 2015; Warton, 2011; Dunstan et al., 2013].

That we treated these groups as independent–or for that matter, even the variation in parameter values of any group across the k time points, is ignored, and a mere snapshot of each group is traditionally considered one at a time–is a shortcoming, of such traditional modelling strategies. In this work, we advance a method for the consideration of parameters across all relevant levels of measurement, within one integrated framework, to enable the learning of correlations across all such levels, thus permitting the prediction of the system parameter vector, with meaningful uncertainties and avoid information loss associated with categorisation of data.

While discussing the generic methodology that helps address the problem of learning the inter-variable relationship $\boldsymbol{f}(\cdot)$, given general hypercuboid-shaped data–comprising multiple measurements of the observable $\boldsymbol{Y}$, where each value of $\boldsymbol{Y}$ is generated at a given value of the system parameter $\boldsymbol{X}$–we focus on developing such learning when this data displays discontinuities. In such a learning exercise, the functional relation $\boldsymbol{f}(\cdot)$ between

the variables needs to be modelled using a high-dimensional stochastic process (a tensor-variate Gaussian Process, for example), the covariance function of which is non-stationary. The correlation between a pair of data slices, (defined by two such measured values of $\boldsymbol{Y}$, each realised at two distinct values of the system parameter $\boldsymbol{X}$), is sometimes parametrically modelled as a function of the distance between the values of the system parameter at which these slices are realised, i.e. "similarity" in values of $\boldsymbol{Y}$ can be modelled as a function of "similarity" in the corresponding $\boldsymbol{X}$ values. However, if there are discontinuities in the data, then such a mapping between "similarities" in $\boldsymbol{X}$ and $\boldsymbol{Y}$ no longer holds. In other words, discontinuities in data call for a model of the correlation that adapts to the discontinuities in the data. We present such correlation modelling in this paper, by modelling each scalar-valued hyperparameter of the correlation structure of the high-dimensional stochastic process, as a random function of the sample path of that process; this random function then, can itself be modelled as a realisation of a scalar-variate stochastic process–a scalar-variate Gaussian Process (GP) for example. Thus, the learning of $\boldsymbol{f}(\cdot)$ is double-layered in which multiple scalar-variate GPs inform a high-dimensional (tensor-variate) GP. The data on the observable $\boldsymbol{Y}$ can be shown to be sampled from a compound tensor-variate and multiple scalar-variate Gaussian Processes.

Acknowledgement of nonstationarity in correlation learning is not new [Paciorek and Schervish, 2004]. In some approaches, a transformation of the space of the input variable is suggested, to accommodate non-stationarity [Sampson and Guttorp, 1992; Schmidt and O'Hagan, 2003; Snoek et al., 2014]. When faced with learning the dynamically varying covariance structure of time-dependent data, others have resorted to learning such a covariance, using Generalised Wishart Process [Wilson and Ghahramani, 2010]. In another approach, latent parameters that bear information on non-stationarity, have been modelled with GPs

and learnt simultaneously with the sought function [Tolvanen et al., 2014], while others have used multiple GPs to capture the non-stationarity [Gramacy, 2005; Heinonen et al., 2016]. However, what has not been presented, is a template for including non-stationarity in high-dimensional data, by nesting lower-dimensional Gaussian Processes with distinct covariances, within a tensor-variate GP (Section 2.2), using a Metropolis-within-Gibbs inference scheme (Section 2.4), to perform with-uncertainties learning of a high-dimensional function, given discontinuities that show up in the hypercuboidally-shaped datasets in general, and illustration of the method on a cuboidally-shaped, real-world dataset (Section 2.5, Section 2.6). This is what we introduce in this paper. Our model is capacitated to learn the temporally-evolving covariance of time-dependent data, if such is the data at hand, but the focus of our interest is to follow the learning of the sought tensor-valued functional relation between a system parameter vector and a tensor-valued observable, with inverse Bayesian prediction of the system parameter values, at which test data on the observable is measured (Section 2.7, Section 2.6). Additionally, flexibility of our model design allows us to undertake both inverse and forward predictions. So we also predict new data at chosen system parameter values given our model and results, and perform model checking, by comparing such generated data against the empirically observed data (Section 2.8).

2.2 The model

We define the relationship between $k-1$ dimensional observable $\boldsymbol{V}$ and model parameter $\boldsymbol{S}$ as $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$, where $\boldsymbol{V} \in \mathbb{R}^{m_1 \times m_2 \cdots \times m_{k-1}}$ and m_i is a positive integer. The $k-1$ dimensional function $\boldsymbol{\xi}(\dot{)}$ is defined as an unknown function with parameter $\boldsymbol{S} \in \mathbb{R}^d$. We are going to estimate value $\boldsymbol{s}^{(test)}$ of using test data $\mathbf{v}^{(test)}$. To do this, we need to estimate function $\boldsymbol{\xi}(\cdot)$ given the training data $\boldsymbol{D} = \{(s_1, v_1), \cdots, (s_n, v_n)\}$ where s_i is the i-th

point that the value of the function is observed. This kind of supervised learning can be done using parametric regression technique like splines/wavelets. The disadvantage of using the splines/wavelets is that it cannot learn the correlation between the components in the high dimensional function $\boldsymbol{\xi}(\cdot)$. Furthermore, such parametric regression causes computational difficulties when the dimensionality of the observation increases. Thus, we are using a high-dimensional Gaussian Process (GP) , i.e. a tensor-variate GP, to the model this high-dimensional data. We treat the $k-1$ dimensional tensor-variate observation as a set of realisation from a $k-1$ dimensional tensor variate GP. By learning the parameters of this $k-1$ dimensional GP using the training data, we are able to predict the value of s^* from the posterior distribution of $\boldsymbol{S}$ given training data and GP parameters. We learn the GP parameters and the posterior distribution for $\boldsymbol{S}$ via MCMC based inference scheme.

2.2.1 Method

We treat the observation $\boldsymbol{v} = \boldsymbol{\xi}(\cdot)$, as a realisation from a $k-1$ dimensional tensor-variate GP. Then, the likelihood function of a set of n realisations of $\boldsymbol{V}$ (that reside within the training data), follows the k dimensional tensor normal distribution [Kolda and Bader, 2009; Richter et al., 2008; McCullagh, 1987; Manceur and Dutilleul, 2013]. As introduced in Chapter 1, the covariance tensor of a k-th order tensor normal distribution, can be decomposed by Tucker decomposition, into k different covariance matrices [Manceur and Dutilleul, 2013; Hoff et al., 2011; Manceur and Dutilleul, 2013; Kolda and Bader, 2009; Xu and Yan, 2015]. The parameters of this tensor normal distribution are a k-th ordered mean tensor $\boldsymbol{M}$ of dimensions $m_1 \times \ldots \times m_k$, and the k number of covariance matrices, $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k$, where the i-th covariance matrix $\boldsymbol{\Sigma}_j$ is an $m_j \times m_j$-dimensional square matrix, $j = 1, \ldots, k$. Then the observed values $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ of $\boldsymbol{V}$–where the i-th such observation occurs at the i-th

design point in the training dataset $\mathbf{D} = \{(\boldsymbol{s}_i, \boldsymbol{v}_i)\}$–has the following joint distribution.

$$[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \sim \mathcal{TN}(\boldsymbol{M}, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k),$$

. s.t., the density of this joint probability distribution of the n observations on $\boldsymbol{V}$ in the training data, is the Tensor-Normal density [Manceur and Dutilleul, 2013; Hoff et al., 2011; Kolda and Bader, 2009]. Let the n observed values of the $k-1$-th ordered tensor-valued variable $\boldsymbol{V}$ be collated to form the k-th ordered tensor $\mathbf{D}_{\boldsymbol{V}}$, i.e $\mathbf{D}_{\boldsymbol{V}} := (\boldsymbol{v}_1, \vdots, \boldsymbol{v}_2, \vdots, \ldots, \vdots, \boldsymbol{v}_n)$, s.t. $\mathbf{D}_{\boldsymbol{V}} \in \mathbb{R}^{m_1 \times \ldots \times m_k}$. Then the joint probability density of the observations is tensor-normal, as stated in the following.

$$f(\mathbf{D}_{\boldsymbol{V}} | \boldsymbol{M}, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k) \propto \exp(-\|(\mathbf{D}_{\boldsymbol{V}} - \boldsymbol{M}) \times_1 \boldsymbol{A}_1^{-1} \times_2 \boldsymbol{A}_2^{-1} ... \times_k \boldsymbol{A}_k^{-1}\|^2/2), \quad (2.1)$$

where the covariance matrix $\boldsymbol{\Sigma}_j = \boldsymbol{A}_j \boldsymbol{A}_j^T, j = 1, ..., k$, i.e. $\boldsymbol{A}_j$ is the unique square-root of the positive definite covariance matrix $\boldsymbol{\Sigma}_j$. One example of a computational algorithm that can be invoked to realise such a square root of a matrix, is Cholesky decomposition [Dereniowski and Kubale, 2003; Higham, 1990; Krishnamoorthy and Menon, 2013]. Equation 2.1 is equivalent to suggesting that the likelihood of the mean $\boldsymbol{M}$ and covariance matrix parameters $(\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k)$ of the k-th ordered tensor-normal density, given the observations that comprise the tensor $\mathbf{D}_{\boldsymbol{V}}$, is k-th ordered tensor-normal. We will employ this likelihood to write the joint posterior probability density of the GP parameters, given the data. But prior to doing that, we examine the possibility of reducing the number of parameters that we seek to learn. In other words, we review the mean and covariance structure of the tensor-variate GP that we invoke to model the data with, to identify those parameters–if any–that can be estimated in a pre-processing stage of the inference, in order to reduce the computational burden of inference. At this point, we also note that seeking to learn

any of the relevant GP parameters implies the need to learn multiple scalar components of these tensor/matrix-valued parameters. Therefore, it would be useful to find ways of parametrising the sought GP mean-tensor/covariance-matrices, thereby reducing the number of parameters that we need to learn in reality. We will achieve this, via kernel-based parametrisation of covariance matrices, when possible.

We will in fact, embark upon the modelling of some covariance matrices using a kernel parametrisation technique that allows for the potently simple–though unrealistic–assumption of stationarity to be included in the global structure of the covariance matrix, while then allowing for the modelling of the parameters of such a simplistic kernel using independent Gaussian Processes. This then effectively challenges the original stationarity assumption, and renders the covariance structure realistic given the expected discontinuities in real-life datasets.

To this effect, we undertake the estimation of the mean tensor is $\boldsymbol{M} \in R^{m_1 \times m_2 \ldots \times m_k}$. It may be estimated as the sample mean $\overline{v}$ of the sample $\{v_1, \ldots, v_n\}$, s.t. n repetitions of $\overline{v}$ form the value $\boldsymbol{m}$ of $\boldsymbol{M}$. Then once $\boldsymbol{m}$ is removed from the data $\mathbf{D}_V$, the data can be modelled with a zero-mean k-th ordered tensor-variate GP. A general method of estimation, like maximum likelihood estimation or least square estimation, can be used, [Hoff, 1997; Veraart et al., 2013]. Then, the Gaussian Process can be converted into a zero mean GP. However, if necessary, the mean tensor itself can be regarded as a random variable and learnt from the data [Chakrabarty et al., 2015], The modelling of the covariance structure of this GP is discussed in the following subsection.

Once we can write the joint posterior probability density of the relevant unknowns, given the data, we will generate marginals for each unknown separately using, to then identify the 95% Highest Probability Density credible regions (HPDs) on each unknown.

We will employ flavours of MCMC that will suit our purpose best.

Of course, the whole point in learning $\boldsymbol{\xi}(\cdot)$ (where the tensor-valued output variable $\boldsymbol{V}$ is related to the vector-valued input variable $\boldsymbol{S}$ as in $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$), is to predict the value of either variable, at which a new or test data on the other variable is observed. So when it comes to the inverse prediction of the value $\boldsymbol{s}^{(test)}$ of the input variable $\boldsymbol{S}$, at which test data $\boldsymbol{v}^{(test)}$ on $\boldsymbol{V}$ is realised, we can use two approaches. Under one, we sample from the posterior probability density of $\boldsymbol{S}$ given the test data $\boldsymbol{v}^{(test)}$, and the tensor-valued parameters of the k-th ordered tensor-variate GP invoked to model $\boldsymbol{\xi}(\cdot)$–learnt using the training data. Another is to write the joint posterior probability density of $\boldsymbol{s}^{(test)}$ and all other parameters of this tensor-variate GP given training data, as well as test data, and sample from this joint posterior density using MCMC, for the 95% HPDs on $\boldsymbol{s}^{(test)}$.

To discuss the relative merits of these two methods of prediction, we realise that for a large and diverse training data set, the two methods will give similar results. However, since the marginal distribution for GP parameters are learnt additionally in the first method, the computational speed of the first approach, is much higher. On the other hand, when the training data is small, or if the training data is not representative of the test data at hand, the learning of $\boldsymbol{s}^{(test)}$ via the first method may affect the learning of the GP parameters, and worries about misrepresentative training data stand mitigated under the second method of sampling from the joint posterior density of all unknowns, given all data. The other, slight concern about sampling from the posterior density of $\boldsymbol{s}^{(test)}$ given the test data, at parameters of the tensor-variate likelihood is that we need to choose which summary of the marginal distribution of any such parameter (given the training data) we will perform the prediction of $s^{(test)}$ at. In general, we choose to perform prediction at the learnt modal value of any such parameter (given the training data), but other summaries – such as median,

mean – are also possible.

2.2.2 Covariance structure

Tucker decomposition of the k-th ordered covariance tensor is employed to obtain the k-number of covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k$ of the k-th ordered tensor-variate Gaussian Process. A k-th ordered random tensor $\boldsymbol{\Sigma} \in R^{m_1 \times m_2 \ldots \times m_k}$ can be decomposed to a unit k-th ordered tensor ($\boldsymbol{Z}$) and k number of covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k$ by **Tucker product** [Hoff, 1997; Manceur and Dutilleul, 2013]:

$$\boldsymbol{\Sigma} = \boldsymbol{Z} \times_1 \boldsymbol{\Sigma}_1 \times_2 \boldsymbol{\Sigma}_2 \ldots \times_k \boldsymbol{\Sigma}_k, \tag{2.2}$$

where the j-th covariance matrix is an $m_j \times m_j$ matrix; $m_j \in \mathbb{Z}_{>0}, m_j \in \{m_1, m_2, \ldots, m_k\}$. The notation $\times_j$ in Equation 2.2 presents the j-mode product of a matrix and a tensor [Oseledets, 2011]. It can be proved that all tensors can be decomposed into a set of covariance matrices [Xu and Yan, 2015]. However, the disadvantage is that the decomposition of many tensors is not unique. In other words, some tensors can be decomposed into multiple sets of matrices. This may cause difficulty in finding the correct combination of covariance matrices that present the correlation structure of the data at hand. One way to solve this problem is to use a prior probability density for the covariance parameters.

The k-th ordered tensor-normal density that represents the likelihood of the mean and covariance parameters of the high-dimensional GP that is invoked to model the data, can easily incorporate the Tucker decomposition of the covariance tensor of this GP. Recalling this density, we can get:

$$f(\boldsymbol{M}, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k | \mathbf{D}_V) = (2\pi)^{-m/2} \prod_{i=1}^{k} |\boldsymbol{\Sigma}_i|^{-m/2m_i}) \\ \times \exp(-\|(\boldsymbol{V} - \boldsymbol{M}) \times_1 \boldsymbol{A}_1^{-1} \times_2 \boldsymbol{A}_2^{-1} ... \times_k \boldsymbol{A}_k^{-1}\|^2/2 \tag{2.3}$$

where $m = \prod_{i=1}^{k} m_i$ and $\boldsymbol{\Sigma}_j = \boldsymbol{A}_j \boldsymbol{A}_j^T$. Any kind of the matrix factor can be used here to get the $\boldsymbol{A}_j$.

This probability density function is well structured. As discussed above the mean tensor $\boldsymbol{M}$ can be estimated using maximum likelihood, so that the learning problem can be reduced to the problem of learning the covariance structure of a zero-mean, tensor-normal GP. We can in principle learn all the covariance matrices directly, given the data. [Hoff, 1997] discusses the implementation of a maximum likelihood estimation technique to estimate the covariance matrices of such a tensor-variate GP. This is however an approach that is inadequate when the sample size is small because the sample-based estimate will tend to be incorrect; indeed discontinuities and steep gradients in the data-especially a small sample and high-dimensional data–will render such estimates of the covariance structure incorrect representation of the correlation structure of such data. Importantly, such an approach does not leave any scope for identifying the smoothness in the function $\boldsymbol{\xi}(\cdot)$ that represents the functional relationship between the input and output variables. Lastly, the uncertainties in the estimated covariance structure of the GP remain inadequately known.

Instead, we learn the covariance structure of the GP whenever we can, using Bayesian inference (MCMC techniques) that allows for proper learning of the uncertainties (as the 95% HPDs). Our employment of kernel parametrisation techniques will also allow for an organic learning of the smoothing length scales of the function $\boldsymbol{\xi}(\cdot)$ sampled from the high-dimensional GP. Lastly, our Bayesian framework allows us to use prior probability densities

for the covariance matrices. As discussed above, some covariance tensors may have multiple decompositions into the covariance matrices. The application of even weak priors allow for the elimination of solutions that are rendered inappropriate given the data at hand. For both the maximum likelihood approach and our Bayesian method, the computational complexity increases with dimensionality of the data. The total number of distinct parameters in the (symmetric) j-th covariance matrix that is $m_j \times m_j$-dimensional, is $\frac{(m_j+1)\times m_j}{2}$. So if all the scalar components of all the k covariance matrices were to be learnt directly by MCMC, we could end up learning a very large number of parameters indeed, with the number of sought parameters increasing quadratically with both dimensionality of the data tensor and size of the dataset. This could rise an unaffordable calculating time. One solution is to implement a parametric model for the covariance matrices, such as kernel parameterisation. This could efficiently reduce the number of parameters in the covariance matrix thus parametrised. Another advantage of kernel parameterisation is that it guides the covariance structure to not go into wrong solutions, given the data at hand, while also allowing for–in principle–for smoothing of the functions sampled from the GP, to be learnt. Kernel functions are usually distance based. Thus, the function itself, together with the distance observations, will bring in the information about the covariance structure. The trick is in choosing a kernel function that is as less restrictive in form as possible, so that it gets guided well by data than be model-driven, while at the same time, not allowing for an inflation in the number of sought parameters to avoid an insurmountable computational challenge.

2.2.3 Different ways of learning covariance matrices

As discussed above, we can learn elements of covariance matrices–or at least of the square-root of each such matrix–directly, by treating each of these scalar-valued components as unknowns. Then however, we will need to learn a very large number of parameters, and this number will only increase as the number and dimensionality of observations increases. The computational complexity of learning the elements of covariance matrices, then increases rapidly with number of observations, rendering the inference task infeasible. In order to reduce the number of parameters in the covariance matrices, one possibility is to use kernel parametrisation of one or more of the covariance matrices $\mathbf{\Sigma}_1, ..., \mathbf{\Sigma}_k$, and then learn the parameters of these kernels using MCMC.

For the p-th covariance matrix, let the ij-th element be $\sigma_{ij}^{(p)}$, i.e. $\mathbf{\Sigma}_p = [\sigma_{ij}]$, where $\mathbf{\Sigma}_p$ is $m_p \times m_p$-dimensional, so that $j, i = 1, \ldots, m_p$. Then $\sigma_{ij}^{(p)}$ bears information about the covariance amongst the i-th and j-th slices of the data, where the data being shaped as as a k-th ordered tensor, such i-th and j-th "slice" is each shaped like a $k-1$-th ordered tensor. We recall that the training data $\mathbf{D}$ comprises pairs of chosen (or designed) value of the input parameter $\boldsymbol{S}$ and the corresponding value of the $k-1$-th ordered tensor-valued output variable $\boldsymbol{V}$, realised at this design point (i.e. chosen value of $\boldsymbol{S}$). In other words, $\mathbf{D} = \{(\boldsymbol{s}_q, \boldsymbol{v}_q)\}_{q=1}^n$, where $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$. Then the i-th "slice" of the data is the value $\boldsymbol{v}_i$ of the $k-1$-th ordered tensor-valued variable $\boldsymbol{V}$ that is realised at the i-th design point. Similarly, the j-th slice is $\boldsymbol{v}_j$ that is realised at the value $\boldsymbol{s}_j$ of the input variable $\boldsymbol{S}$.

The correlation–and therefore, the covariance $\sigma_{ij}^{(p)}$–between the i-th and j-th slices of the data then decreases as the slices get increasingly more disparate. This disparity between a pair of slices realised at respective values of input variable $\boldsymbol{S}$, can be treated as increasing,

with increasing difference in the values of S at which these slices are realised, i.e. with increasing $|s_i - s_j|$. In fact, we can model the correlation $\sigma_{ij}^{(p)}$ between the i-th and j-th slices of the data as a decreasing function of this disparity, i.e. of $|s_i - s_j|$. In an even more generalised model, $\sigma_{ij}^{(p)}$ may be modelled as a decreasing function of some components of $s_i - s_j$, but trending differently with other components of s_i and s_j, where S is a higher-dimensional than a scalar. In either modelling strategy, we suggest that $\sigma_{ij}^{(p)}$ be modelled as a function $K(s_i, s_j)$. One of the simplest of these modelling strategies will then model this component $\sigma_{ij}^{(p)}$ of the covariance matrix $\mathbf{\Sigma}_p$, using a "stationary kernel" that is a function of the Euclidean distance between s_i and s_j, i.e. when $K(s_i, s_j) = f((s_i - s_j)^\alpha)$, where $f(\cdot)$ is any mapping from $\mathbb{R}^d \times \mathbb{R}^d$ to $\mathbb{R}$ (recalling that $S \in \mathbb{R}^d$) and $\alpha \in \mathbb{R}/\{0\}$.

So in general, we can then define $\sigma_{ij}^{(p)} = K_p(s_i, s_j)$, where $K_p(s_i, s_j)$ is the kernel function $K_p(\cdot, \cdot)$, computed at the i-th and j-th input variables. Thus, the number of distinct unknown parameters involved in the learning of $\mathbf{\Sigma}_p$ would reduce from $m_p(m_p + 1)/2$, to the number of hyper-parameters that parametrise the kernel function $K_p(\cdot, \cdot)$. Thus, the exact reduction in the number of sought parameters will vary with the choice of the kernel function; for example, in an SQE kernel (discussed in Chapter 1), in which the hyper-parameters are the length scales along each direction in input space, the number of hyper-parameters of $K_p(\cdot, \cdot)$ is simply the dimensionality of the input space. So, such kernel parametrisation does help reduce the number of unknowns that need to be sampled by MCMC.

However, kernel parametrisation is not always possible. There are two situations when we clearly cannot use kernel parametrisation. Firstly, this parametrisation may cause information loss and this may not be acceptable [Aston and Kirch, 2012]. One example of this occurs when one uses stationary kernel to model the covariance structure of discontiuous

data. Although this can be solved by changing into an appropriate, non-stationary kernel function, condensing all the information of the covariance matrix through such parametrisation may not be necessarily be lossless; so ideally, one may wish to learn each element of the covariance matrix directly (perhaps using MCMC, to gain the advantage of comprehensive 95% HPDs). Another situation when we will necessarily avoid kernel parametrisation of a covariance matrix, is when we cannot find input parameters, at which the corresponding slices in the data are realised, where the covariance matrix in question is composed of elements that are each, the pairwise covariance between a pair of such slices.

In such situations, we will learn the elements of the covariance matrix directly using MCMC. However, as discussed above, this can entail the learning of an infeasibly large number of parameters. So a direct learning of all distinct elements of $\mathbf{\Sigma}_p$ is feasible, as long as total number of all unknowns learnt by MCMC $\lesssim 200$. One way to solve this problem is to use an empirical estimation for the covariance matrix. An empirical estimation of $\mathbf{\Sigma}_p$ can be performed by collapsing each of the m_p number of high-dimensional ($k-1$-th ordered tensor-shaped) slices of the data, along all-but-one axis in output space, i.e. the native space of $\boldsymbol{V}$, namely along the p-axis. Such an operation then reduces each of the high-dimensional m_p slices to a vector. The covariance computed using a pair of such vectors (adjusted for sample size), is then an element of $\mathbf{\Sigma}_p$. Each such reduced vector then possesses the compressed information from all the relevant dimensions of the data. The covariance matrix $\mathbf{\Sigma}_p$ is thus approximated by an empirical estimate of the covariance amongst such vectors.

Indeed such an empirical estimate of any covariance matrix may then be easily generated, but it indulges in linearisation amongst the different dimensionalities of the observable $\boldsymbol{V}$. So when the $\mathbf{\Sigma}_p$ covariance matrix bears information about the m_p number

of high-dimensional slices of the data, such linearisation will cause loss of information about the covariance structure amongst the components of these high-dimensional slices. Also, smaller sample sizes will render the empirical estimate a worse approximation of the covariance matrix, than larger samples.

In summary, we model the covariance matrices as kernel parametrised, or empirically-estimated, or learnt directly using MCMC.

An accompanying computational worry is the inversion of any of the covariance matrices; for a covariance matrix that is an $m_p \times m_p$-dimensional matrix, the computational order for matrix inversion is well known to be $O(m_p^3)$ [Knuth, 1997].

Often, including in our first application, a simple stationary covariance kernel, such as the Squared Exponential (SQE) covariance kernel is used. The SQE kernel computed at two values $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ of the input space variable $\boldsymbol{S}$, depends on the Euclidean distance between $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ as:

$$K(\boldsymbol{s}_i, \boldsymbol{s}_j) = a_{ij} \exp\left(-(\boldsymbol{s}_i - \boldsymbol{s}_j)^T \boldsymbol{Q}^{-1} (\boldsymbol{s}_i - \boldsymbol{s}_j)\right) \tag{2.4}$$

where $\boldsymbol{Q}$ is a diagonal matrix, the diagonal elements of which are the length scale parameters of the covariance matrix. These (unknown) length scales bear information about the extent of input space, over which correlations persist, within a function that is sampled from the (high-dimensional) GP, a covariance matrix of which is being modelled using the SQE kernel above. These length scale parameters $\ell_1, \ldots, \ell_d$ tell us how quickly the correlation fades away along a given direction in the input space, i.e. the space of the d-dimensional vector-valued input variable $\boldsymbol{S}$; thus, higher is ℓ_q, the correlation persists over longer intervals of S_q, where $\boldsymbol{S} = (S_1, \ldots, S_d)^T$ and $q = 1, \ldots, d$. Then the matrix $\boldsymbol{Q}^{-1}$ is the inverse of the diagonal matrix of the length scales, and is therefore diagonal itself, with the

diagonal elements given as $\frac{1}{\ell_1}, \ldots, \frac{1}{\ell_d}$, where we interpret $1/\ell_q$ as the smoothness parameter, given that it is the reciprocal of the length scale parameter ℓ_q. These are d-number of unknown parameters that we learn from the data, using MCMC; indeed, these are treated as the hyperparameters of the covariance kernel. Here a_{ij} is the amplitude of the covariance matrix. We can learn all the a_{ij} parameters directly from MCMC. It merits mention that the SQE model is designed s.t. identifiability of both a_{ij}, and ℓ_q is assured. In other words, it is not possible to subsume the a_{ij} in the definition of the SQE covariance kernel (Equation 2.4), into the length scale parametrisation, since any attempt towards subsuming the i, j-dependent a_{ij} parameter into the i, j-independent length scale, that will result in a new length scale parameter that will not be independent of i, j any more (as easily verified using the example of a 1-dimensional input space, i.e. for d=1). However, learning the a_{ij} parameters directly from MCMC will amount to a very large number of parameters of the covariance matrix that we will then need to make inference upon. To avoid this, we interpret the SQE kernel parametrisation to be endowed a global amplitude instead, while the length scale parameters are also left as unknowns that are learnt from the data. The resulting covariance structure then crucially depends on the distance between the pair of values of the input variable $\boldsymbol{S}$ that the kernel is computed at. Below, we state the kernel function that we will use to parametrise our covariance matrices with–when in possession of information about the value of the input variable that a slice of the data is realised at, where covariance between a pair of such slices comprises an element of the covariance matrix in question.

$$K(\boldsymbol{s}_i, \boldsymbol{s}_j) := A \left[\exp\left(-(\boldsymbol{s}_i - \boldsymbol{s}_j)^T \boldsymbol{Q}^{-1} (\boldsymbol{s}_i - \boldsymbol{s}_j) \right) \right],$$

where A is a global scale. An interpretation of this is that we have scaled all local ampli-

tudes a_{ij} to be < 1 using the global factor A, and these scaled local amplitudes are then subsumed into the argument of the exponential in the RHS of the last equation, s.t. the reciprocal of the correlation length scales, that are originally interpreted as the elements of the diagonal matrix $\boldsymbol{Q}^{-1}$, are now interpreted as the smoothing parameters modulated by such local amplitudes. The global scale A is itself subsumed as a scale factor, in one of the covariance matrices of the tensor-normal distribution at hand–this is the matrix, the distinct elements of which are learnt directly using MCMC, i.e. without resorting to any parametrisation or to any form of empirical estimation. If we do not use any empirical estimation or do not have any covariance matrices learnt directly, we can still learn this parameter using MCMC. However, it has to be kept in mind that the above interpretation is only loose, since the same smoothness parameters cannot accommodate all (scaled by a global factor) local amplitudes$\in (0,1]$, for all $\boldsymbol{s}_i - \boldsymbol{s}_j$. This is why, our definition of the kernel function $K(\boldsymbol{s}_i, \boldsymbol{s}_j)$ is best understood as our choice of model for the covariance kernel (see last equation). This model helps to reduce our number of parameters that are learnt y MCMC.

Let us revisit our likelihood stated in Equation 2.1. We had stated that the likelihood is

$$f(\mathbf{D}_{\boldsymbol{V}}|\boldsymbol{M}, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k) \propto \exp(-\|(\mathbf{D}_{\boldsymbol{V}} - \boldsymbol{M}) \times_1 \boldsymbol{A}_1^{-1} \times_2 \boldsymbol{A}_2^{-1} ... \times_k \boldsymbol{A}_k^{-1}\|^2/2), \quad (2.5)$$

where the covariance matrix $\boldsymbol{\Sigma}_p = \boldsymbol{A}_p \boldsymbol{A}_p^T, p = 1, ..., k$. However, we recall that some of our covariance matrices might be parametrised using kernel functions, that depend on the design points. In light of this, we need to update our statement for the likelihood of the mean and covariance parameters of the tensor-normal density that is the joint probability density of $m_k \equiv n$ realisations of the output variable $\boldsymbol{V}$, given the n observations on $\boldsymbol{V}$, as well as the corresponding design point at which each observation of $\boldsymbol{V}$ is realised, i.e. the

likelihood is the probability density not only of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$, but also of $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$, or to put is succinctly, of training data $\mathbf{D} := \{(\boldsymbol{s}_i, \boldsymbol{v}_i)\}_{i=1}^n$–given the mean tensor $\boldsymbol{M}$ and covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k$. Indeed we estimate $\boldsymbol{M}$ and subtract this estimated value from the data. Indeed, it is understood hereon, that the data $\mathbf{D}$ discussed above is the data reduced by the estimated mean. Using this likelihood and adequately chosen prior probability densities on the unknown parameters, we can write the joint posterior probability density of the unknown parameters given the training data, where the unknowns include parameters of the covariance matrices–if kernel parametrised, or elements of such a matrix itself–if being directly learnt using MCMC. Once this is achieved, we use the MCMC techniques to sample from the joint posterior probability density of the unknown parameters of each covariance matrix and mean tensor, given the training data. Thus, the updated statement of the likelihood is

$$f(\mathbf{D}|\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k) \propto \exp(-\|(\mathbf{D}) \times_1 \boldsymbol{A}_1^{-1} \times_2 \boldsymbol{A}_2^{-1} ... \times_k \boldsymbol{A}_k^{-1}\|^2/2), \tag{2.6}$$

where the covariance matrix $\boldsymbol{\Sigma}_p = \boldsymbol{A}_p \boldsymbol{A}_p^T, p = 1, ..., k$.

We generate the marginal posterior probability densities of each unknown parameter given training data, and identify the 95% Highest Probability Density (HPD) credible regions on each parameter.

Thereafter, the prediction of $\boldsymbol{s}^{(test)}$ can be performed, by generating posterior samples using MCMC from the posterior predictive of $\boldsymbol{S}$ given all data and the learnt/estimated GP mean and covariance structures. We also write the joint posterior probability density of $\boldsymbol{s}^{(test)}$ and all the other GP parameters given test+training data and then use MCMC to identify the marginal distribution of $\boldsymbol{S}$ given the test data and training data, in addition to the marginals of the tensor-variate parameters.

2.3 Imposing non-stationarity by modelling hyperparameters of covariance kernels as realisations of Stochastic Process

In our definition of the kernel function that we employ to parametrically model a covariance matrix $\mathbf{\Sigma}_p^{(m_p \times m_p)}$, we have included a smoothing parameter (reciprocal of the correlation length scale), and a global amplitude, as the hyperparameters of the kernel. As is the convention, these hyperparameters are treated as unknown constants, that we aim to learn from the data. Then by definition of this kernel function, (Equation 2.2.3), all sample paths are endowed with the same smoothing parameters $\ell_1, \ldots, \ell_d$, and global amplitude A–each with the 95% HPD uncertainty that its MCMC-based learning entails. However, if the sampled function $\boldsymbol{\xi}(\cdot)$–in our original definition of the problem as $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$–is in itself a discontinuous function, then defining the correlation between $\boldsymbol{\xi}(\boldsymbol{s}_i)$ and $\boldsymbol{\xi}(\boldsymbol{s}_j)$, as dependent on $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, only via the Euclidean distance between them, will be an incorrect representation of the correlation at these two points in the input space. Indeed, if the data on the output variable $\boldsymbol{V}$ is not continuous, i.e. if the sampled function is discontinuous, such a stationary definition of the correlation between any pair of points in the function domain, i.e. $\forall \boldsymbol{s}_i, \boldsymbol{s}_j$, is wrong. In other words, it is not correct to define the covariance matrix that bears information about covariance between any pair of slices of the data–with each slice realised at a given point in the input space–using a stationary kernel.

One way to generalise the model for the covariance kernel is to suggest that the hyperparameters vary with the sample path. We understand the effect of this undertaking, using the following construction. Consider a sample path of the tensor-variate Gaussian Process that we invoke to model the sought function $\boldsymbol{\xi}(\cdot)$. Now maintaining everything else, in-

troduce stochasticity to this particular sampled function, by varying the hyperparameters of the kernel function that is used to parametrically model a covariance matrix $\mathbf{\Sigma}_p$ of the tensor-normal likelihood. At every value of $\ell_1, \ldots, \ell_d, A$, record the value $\boldsymbol{\xi}(\boldsymbol{s}_i)$, where $\boldsymbol{s}_i$ is arbitrarily chosen. For each choice of $\ell_1, \ldots, \ell_d, A$, we record the value of $\boldsymbol{\xi}(\boldsymbol{s}_i)$, and include each such generated $\boldsymbol{\xi}(\boldsymbol{s}_i)$ in the sample called $\mathcal{S}_1$, Then each element of sample $\mathcal{S}_1$ is generated by at different set of kernel hyperparameter values. Also, at a fixed set of values of the kernel hyperparameters, define sample $\mathcal{S}_2$ that comprises values of $\boldsymbol{\xi}(\boldsymbol{S})$, across a set of values of the input variable: $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{max}$. Thus, while sample $\mathcal{S}_1$ is generated by sampling across kernel hyperparameter values, $\mathcal{S}_2$ is generated by sampling across input space variable values. Then if ergodicity is assumed to hold in the system, distribution of $\mathcal{S}_1$ and $\mathcal{S}_2$ will be the same. Loosely speaking, trends in the distribution of elements of both samples will be the same.

However, what we need to assume, is weaker than ergodicity, namely, lack of continuity in the variation within elements of one sample suggests the same within elements of the other sample. Then it follows that:

discontinuous variation amongst members of sample $\mathcal{S}_2 \implies$ discontinuity amongst elements of sample $\mathcal{S}_1$,

i.e. discontinuities in variation amongst $\boldsymbol{\xi}(\boldsymbol{s}_1), \ldots, \boldsymbol{\xi}(\boldsymbol{s}_{max}) \implies$ discontinuity within $\mathcal{S}_1$,

i.e. discontinuities in variation amongst $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{max} \implies$ discontinuity within $\mathcal{S}_1$,

i.e. discontinuities in the data on $\boldsymbol{V} \implies$ discontinuities amongst values of $\boldsymbol{\xi}(\boldsymbol{s}_i)$ generated by spanning across the kernel hyperparameter values – effectively creating different sample paths in the process.

So one way of acknowledging discontinuities in data, is by relaxing the restriction that the same hyperparameter values parametrise all sampled functions; instead, stochastically

varying hyperparameters are sought.

In this model, the kernel hyperparameters are themselves randomly varying, to accompany every realisation of a sample function from the tensor-variate GP. Thus, we propose treating each kernel hyperparameter as a realisation from a stochastic process. Such a stochastic process is independent of the tensor-variate GP of course. Indeed, we sample each hyperparameter from a distinct Gaussian Process. In practice, it is only the correlation length scale parameters $\ell_1, \ldots, \ell_d$ that we model in this way.

The underlying principle then is to sample a new length scale parameter of the kernel function that parametrises a covariance matrix of the tensor-normal likelihood, at every new realisation of the tensor-variate function $\boldsymbol{\xi}(\cdot)$. To be clearer, we suggest modelling ℓ_c as a function of $\boldsymbol{\xi}(\cdot)$, where this function is then modelled as a realisation from a Gaussian Process; $c = 1, \ldots, d$. But, within our MCMC-based inference procedure, a new $\boldsymbol{\xi}(\cdot)$ function is sampled from the tensor-variate GP, at each iteration. The equivalent statement of this is, that at the t-th iteration, $\ell_c = g_c(t)$, $\forall c = 1, \ldots, d$, where $t = 0, 1, \ldots, t_{max}$, i.e. the MCMC chain that we run, starts with the 0-th and ends with the t_{max}-th iteration. Here, the function $g_c(\cdot)$ that is modelled using a scalar-variate GP. Then at the t-th iteration, a new function is sampled from the tensor-variate GP, the covariance structure of which is specified in that iteration, upon the sampling of the $\ell_1, \ldots, \ell_d$ parameter values from d distinct and independent scalar-variate GPs. Thus we suggest an ordered sequence of sampling–first of the length scale parameters, each from the d-number of scalar-variate GPs–and then at the sampled ℓ_c values that fix the covariance structure of the generative tensor-variate GP, of the high-dimensional function $\boldsymbol{\xi}(\cdot)$. Such an ordered sequence of sampling is possible within a Metropolis-within-Gibbs scheme that we propose for our inference (discussed below).

As the length scale parameter ℓ_c is modelled as a function of (the sampled function $\boldsymbol{\xi}(\cdot)$, and thereby of) the iteration number variable $T \in \mathbb{Z}_{\geq 0}$, the scalar-variate GP that it is sampled from, in this modelling strategy, is given as $\ell_c = g_c(T), \quad , \forall c = 1, \ldots, d$, s.t. the joint probability distribution of t_0 number of ℓ_c values, with each value realised from this scalar-variate GP, at each of t_0 number of iterations, is multivariate normal, with mean vector $\boldsymbol{M}_c$ and a covariance matrix $\boldsymbol{S}_c$, i.e.

$$[g_c(t-t_0), \ldots, g_c(t-2), g_c(t-1)] \sim \mathcal{MN}(\boldsymbol{M}_c, \boldsymbol{S}_c),$$

where $\boldsymbol{M}_c$ is a t_0-dimensional vector and $\boldsymbol{S}_c$ is a $t_0 \times t_0$-dimensional covariance matrix.

It is to be noted that there is no data-driven constraint on what we can choose for the number t_0 that gives the sample size of the realisations of each of the d length scale parameters. On the other hand, the value of t_0 is driven to ensure feasibility of computational manipulation of working with structures, the dimensionality of which is affected by t_0. One such structure is the t_0-dimensional mean vector $\boldsymbol{M}_c$, which is estimated as the empirical mean of ℓ_c, given by a t_0-dimensional vector, all components of which are given by the scalar $m_c^{(t)} := [g_c(t-t_0) + \ldots + g_c(t-2) + g_c(t-1)]/t_0 \equiv [\ell_c^{(t-t_0)} + \ldots + \ell_c^{(t-1)}]/t_0$, i.e. the empirical mean of the realisations of ℓ_c in the $t-1$-th iteration, to the $t-t_0$-th iteration. Thus, computational load is not affected by t_0 as far as this mean vector is concerned. In fact, in the t-th iteration, upon the empirical estimation of the mean as given above, it is subtracted from the "data" $\mathbf{D}_{c'}^{(t)} := \{\ell_c^{(t-t_0)}, \ldots, \ell_c^{(t-1)}\}$, so that the subsequent mean-subtracted data $\mathbf{D}_c^{(t)} := \{\ell_c^{(t-t_0)} - m_c^{(t)}, \ldots, \ell_c^{(t-1)} - m_c^{(t)}\}$. It is indeed this mean-subtracted sample that we model using a scalar-variate, zero-mean GP, s.t. the joint probability distribution of elements of this sample (of size t_0) is multivariate normal with mean vector $\mathbf{0}$ and covariance matrix $\boldsymbol{S}_c$, ($\forall c = 1, \ldots, d$, where d is the dimensionality of

the input variable $\boldsymbol{S}$ in the definition of the original learning problem: $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$).

Thus, the likelihood of the covariance matrix $\boldsymbol{S}_c$, given realisations of the c-th length scale parameter, in the last t_0 number of iterations is

$$\mathcal{L}(\boldsymbol{S}_c|\mathbf{D}_c^{(t)}) = \mathcal{MN}(\mathbf{0}, \boldsymbol{\Sigma}_c), \quad , \forall c = 1, \ldots, d,$$

where for $t \geq t_0$, we model $\boldsymbol{S}_c$ as $\boldsymbol{S}_c = [s_{\alpha\beta}^{(c)}]$, with

$$s_{ab}^{(c)} = a_c \exp\left[-\frac{(\alpha-\beta)^2}{2\delta_c^2}\right], \quad \alpha, \beta = t - t_0, \ldots, t-1, \quad \forall c = 1, \ldots, d.$$

Thus, we model the covariance structure of the scalar-variate GP that the c-th length scale parameter is considered a realisation from, using a Square Exponential kernel function, the global amplitude parameter of which is a_c, and the length scale parameter of which is δ_c. We will learn the hyperparameter values a_c and δ_c, $\forall c = 1, \ldots, d$, from the data. We realise that the q-th diagonal element of $\boldsymbol{S}_c$ is a_c, $\forall q = 1, \ldots, t_0$.

In the following section, we discuss the learning of the hyperparameters of the scalar-variate GPs, that the hyperparameters of the covariance kernels of the generative tensor-variate GP–that generates samples of the function $\boldsymbol{\xi}(\cdot)$.

2.4 Inference

We discuss the MCMC-based inference on the unknowns here. To begin with, we will discuss inference for the model in which each of the length scale hyperparameters of the covariance kernel functions, is modelled using a scalar-variate, zero-mean GP, the covariance matrix of which parametrised by a stationary kernel, thereby prompting the need to learn the two scalar-valued hyperparameters of each such kernel function. This illustrative case is succinctly referred to as the "nested-GP" model. To summarise, the unknowns in this "nested GP" model include

1(a) hyperparameters (global amplitude a_c and correlation length scale δ_c) of the c-th kernel function that parametrises the covariance matrix of the zero-mean multivariate normal likelihood that equivalently represents the joint probability density of t_0 realisations of the c-th length scale ℓ_c: $c = 1, \ldots, d$. These hyperparameters are relevant at the t-th iteration, where $t \geq t_0$. At such iterations, the updated values of a_c and δ_c for each c will provide a value for ℓ_c in each iteration. Here $\ell_1, \ldots, \ell_d$ are the unknown correlation length scales of one of the covariance matrices of the k-th ordered tensor normal likelihood that represents the joint probability of n realisations of the observable $\boldsymbol{V}$, where $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$, with $\boldsymbol{S} \in \mathbb{R}^d$. For our illustration of the suggested inference, we will assume that only one of the covariance matrices (say $\boldsymbol{\Sigma}_p$, where $p \in \{1, 2, \ldots, k\}$) of this tensor-normal density is kernel parametrised, so that we need to model only the d length scale parameters of this matrix, each as a realisation from a scalar-variate GP, with the global amplitude parameter of this kernel, subsumed as a global scale for other covariance matrices that are learnt directly from MCMC. Thus, in this illustration, there are $2d$ kernel hyperparameters to learn in total: $a_1, \ldots, a_d, \delta_1, \ldots, \delta_d$.

1(b) the correlation length scales $\ell_1, \ldots, \ell_d$ of the kernel function that parametrises $\boldsymbol{\Sigma}_p$, at the t-iteration, where $t < t_0$. At such early iterations, the hyperparameters of the kernel function that parametrises $\boldsymbol{\Sigma}_p$, are not modelled as realisations of a scalar-variate GP, but are learnt directly using MCMC.

2 distinct elements of other covariance matrices of the tensor-normal density mentioned in the last bullet point, that are learnt directly using MCMC. We use the qualification "distinct" to clarify that we seek only the upper (or lower) triangle of

these symmetrical covariance matrices. Let these elements be jointly referred to as $\{\sigma_q\}_{q=1}^{q_{max}}$, where q_{max} is known, given the number of such covariance matrices (out of the total of $k-1$ that are not kernel parametrised) that are directly inferred upon by MCMC, and the dimensionality of each such matrix.

We undertake a Metropolis-within-Gibbs strategy to implement the MCMC-based inference. For all iterations with iteration-number $t \geq t_0$, we partition the sought unknowns to 2 sets: $\{a_c, \delta_c\}_{c=1}^{d}$ and $\{\sigma_q\}_{q=1}^{q_{max}}$, and update the first set of unknowns in the first block update, while the second set is updated in the second block update, at the updated values of $\{a_c, \delta_c\}_{c=1}^{d}$. For $t < t_0$, we learn $\ell_1, \ldots, \ell_d$ in the first block update and the σ_q parameters in the next. The algorithm for the inference strategy is as follows.

1 In the 0-th iteration, set all unknown parameters to arbitrarily chosen seed values : a_c is set to the seed $a_c^{(0)}$ and δ_c is set to the seed $\delta_c^{(0)}$ $\forall c = 1, \ldots, d$; σ_q is set to the seed $\sigma_q^{(0)}$ $\forall q = 1, \ldots, q_{max}$. We also set the length scales of the kernel-parametrised covariance matrix $\mathbf{\Sigma}_p$ to their respective seed values, i.e. set $\ell_c := \ell_c^{(0)} \forall c = 1, \ldots, d$.

2(a) At the beginning of the t-th iteration, for $t < t_0$, the current value of the ell_c parameter is $\ell_c^{(t-1)}$. We propose the new value, $\ell_c^{(t\star)}$ from a Gaussian distribution, the mean of which is the current value of this parameter, namely $\ell_c^{(t-1)}$, and the variance of which is chosen experimentally, to be the constant v_c, i.e.

$$\ell_c^{(t\star)} \sim \mathcal{N}(\ell_c^{(t-1)}, v_c).$$

This proposing is undertaken $\forall c = 1, \ldots, d$. We choose adequate prior probability densities (often Gaussian priors with mean $\ell_c^{(0)}$ and large constant variances) on all the ℓ_c parameters. We refer to these prior probability densities as $\pi_0(\ell_1, \ldots, \ell_d)$. The

proposed ℓ_c parameters then inform the kernel function $K_p(\cdot,\cdot)$ that is used to kernel parametrise the covariance matrix $\mathbf{\Sigma}_p$, s.t. the proposed kernel-parametrised covariance matrix in the t-th iteration, $t < t_0$, is $\mathbf{\Sigma}_p^{(t\star)} = \boldsymbol{A}_p^{(t\star)}(\boldsymbol{A}_p^{(t\star)})^T$, while the current values of the ℓ_c parameters suggest that the current kernel-parametrised covariance matrix is $\mathbf{\Sigma}_p^{(t-1)} = \boldsymbol{A}_p^{(t-1)}(\boldsymbol{A}_p^{(t-1)})^T$. We compute $\mathbf{\Sigma}_p^{(t\star)} = \left[\exp\left(-\frac{(s_i-s_j)^2}{2(\boldsymbol{\ell}^{(t\star)})^2}\right)\right]$, where $\boldsymbol{\ell}^{(t\star)} := (\ell_1^{(t\star)},\ldots,\ell_d^{(t\star)})^T$. Similarly, $\mathbf{\Sigma}_p^{(t-1)}$ is defined in terms of the vector of the length scales, $\boldsymbol{\ell}^{(t-1)}$ that is the current value of the end of the $t-1$-th iteration.

3(a) We compute the ratio of the posterior probability densities of the proposed ℓ_c parameters given the training data $\mathbf{D}$ to the posterior density of the current ℓ_c values. The ratio of the proposal densities does not get invoked since the proposal is symmetric. Thus, the ratio that we compute is

$$r := \frac{\exp(-\|(\mathbf{D}) \times_1 \boldsymbol{A}_1^{-1} \ldots \times_p (\boldsymbol{A}_p^{(t\star)})^{-1} \ldots \times_k \boldsymbol{A}_k^{-1}\|^2/2)\pi_0(\ell_1^{(t\star)},\ldots,\ell_d^{(t\star)})}{\exp(-\|(\mathbf{D}) \times_1 \boldsymbol{A}_1^{-1} \ldots \times_p (\boldsymbol{A}_p^{(t-1)})^{-1} \ldots \times_k \boldsymbol{A}_k^{-1}\|^2/2)\pi_0(\ell_1^{(t-1)},\ldots,\ell_d^{(t-1)})},$$

and compare r with the value of the uniform random variate $U \sim \mathcal{U}[0,1]$.
–If $u \geq r$, we reject the proposed values $\ell_1^{(t\star)},\ldots,\ell_d^{(t\star)}$, and set the current value of the ℓ_c parameter at the end of the t-th iteration to be $\ell_c^{(t)} = \ell_c^{(t-1)} \forall c = 1,\ldots,d$.
–If $u < r$, we accept the proposed values $\ell_1^{(t\star)},\ldots,\ell_d^{(t\star)}$, and set the current value of the ℓ_c parameter at the end of the t-th iteration to be $\ell_c^{(t)} = \ell_c^{(t\star)}, \forall c = 1,\ldots,d$.
Thus, for iterations $t < t_0$, the first block update is a manifestation of Random Walk.

2(b) If the iteration number t is s.t. $t \geq t_0$, then we model the ℓ_c parameters, each as a realisation from a distinct scalar-variate GP, the covariance structure of which is kernel-parametrised s.t. these kernel hyperparameters are a_c and δ_c, $\forall c = 1,\ldots,d$. Then the counterpart of point 2(a), within the "nested GP" approach, is now discussed. Let the current values of a_c and δ_c be $a_c^{(t-1)}$ and $\delta_c^{(t-1)}$. We propose values of these parameters in the t-th iteration, respectively from a Truncated-Normal density (left-truncated at 0, mean $a_c^{(t-1)}$, and experimentally chosen constant variance $v_a^{(c)}$), and a Normal (mean $\delta_c^{(t-1)}$, and experimentally chosen constant variance $v_\delta^{(c)}$), i.e.

$$a_c^{(t\star)} \sim \mathcal{TN}(a_c^{(t-1)}, 0, v_a^{(c)}), \quad \forall c = 1,\ldots,d,$$
$$\delta_c^{(t\star)} \sim \mathcal{N}(\delta_c^{(t-1)}, 0, v_\delta^{(c)}), \quad \forall c = 1,\ldots,d.$$

Now the GP that ℓ_c is modelled with, currently has a covariance structure that is parametrised by the $t_0 \times t_0$-dimensional covariance matrix $\boldsymbol{S}_c$ s.t. currently the ij-th element of this matrix is the covariance between the value of ℓ_c that was current in the $t-i$-th iteration and the value current in the $t-j$-th iteration, i.e. $\boldsymbol{S}_c^{(t-1)} = \left[a_c^{(t-1)} \exp\left(-\frac{(i-j)^2}{2(\delta_c^{(t-1)})^2}\right)\right]; i,j = 1,\ldots,t_0.$

Thus, at any fixed value (say i) of the input variable–the iteration number–the i-th diagonal element $a_c^{(t-i)}$ of the covariance matrix $\boldsymbol{S}$, gives the variance of the Gaussian distribution that $\ell_c^{(t-i)}$ can be considered to be sampled from. Following this, we reduce this scalar-variate GP to a Gaussian distribution, by fixing the value of the input-space variable, (which in this situation is the iteration number), to t. Then the proposed variance of the Gaussian distribution that ℓ_c is sampled from, at the t-th iteration, is the proposed value of the a_c parameter in this iteration, i.e. $a_c^{(t\star)}$. Under a Random Walk paradigm, the mean of this Gaussian distribution is the current value of the ℓ_c parameter. In other words, the model suggests that

$$\ell_c^{t\star} \sim \mathcal{N}(\ell_c^{(t-1)}, a_c^{(t\star)}).$$

This is essentially suggesting an adaptive Random Walk updating scheme for the ℓ_c parameter, $\forall c = 1, \ldots, d$.

3(b) This is the counterpart of point 3(a) for the $t \geq t_0$ iterations, i.e. when the "nested GP" model is in play. Again, as during the discussion of 3(a), here we compute the ratio of the posterior probability densities of the proposed to the current values of the unknowns that are updated in the first block. This posterior density has the contribution from the k-th ordered tensor-normal likelihood that the observable $\boldsymbol{V}$ $(= \boldsymbol{\xi}(\boldsymbol{S}))$ is modelled as a realisation from. But the covariance matrix $\boldsymbol{\Sigma}_p$ of this tensor-normal likelihood is kernel-parametrised, with a GP prior imposed on each length scale parameters $\ell_1, \ldots, \ell_d$ of this kernel function. Then the joint probability density of the set of t_0 number of realisations $\{\ell_c^{(t-t_0)}, \ldots, \ell_c^{(t-1)}\}$ of the parameter ℓ_c, from the scalar-variate, zero-mean GP, is multivariate normal with mean vector $\boldsymbol{0}$ and covariance matrix $\boldsymbol{S}_c = [s_c^{(ij)}]$, which is kernel-parametrised as $s_c^{ij} = a_c \exp\left[-\dfrac{(i-j)^2}{2\delta_c^2}\right]$, s.t. the current value of the covariance matrix in the t-th iteration is $\boldsymbol{S}_c^{(t-1)} = \left[a_c^{(t-1)} \exp\left[-\dfrac{(i-j)^2}{2(\delta_c^{(t-1)})^2}\right]\right]$, and the proposed value of the covariance matrix in the t-th iteration is $\boldsymbol{S}_c^{(t\star)} = \left[a_c^{(t\star)} \exp\left[-\dfrac{(i-j)^2}{2(\delta_c^{(t\star)})^2}\right]\right]$. In other words, the prior probability density on the t_0-dimensional vector $\boldsymbol{\ell}_c^{(t_0)} := (\ell_c^{(t-t_0)}, \ldots, \ell_c^{(t-1)})^T$ of values of the c-th length scale parameter, over the last t_0 iterations is multivariate normal, with mean vector $\boldsymbol{0}$ and covariance matrix $\boldsymbol{S}$, i.e.

$$\pi_0(\ell_c^{(t-t_0)}, \ldots, \ell_c^{(t-1)}) = \frac{1}{\sqrt{\det(2\pi\boldsymbol{S})}} \exp\left[-\frac{1}{2}(\boldsymbol{\ell}_c^{(t_0)})^T \boldsymbol{S}^{-1} (\boldsymbol{\ell}_c^{(t_0)})\right],$$

where $\boldsymbol{\ell}_c^{(t_0)}$ and the current and proposed $\boldsymbol{S}$ (as a function of current and proposed a_c and δ_c values) are defined above. This is true $\forall c = 1, \ldots, d$. Then the ratio of the posterior probability density of the proposed to the current values of the unknowns $a_1, \ldots, a_d, \delta_1, \ldots, \delta_c$, given the data is

$$r := \frac{\exp(-\|(\mathbf{D}) \times_1 \boldsymbol{A}_1^{-1} \ldots \times_p (\boldsymbol{A}_p^{(t\star)})^{-1} \ldots \times_k \boldsymbol{A}_k^{-1}\|^2/2) \prod_{c=1}^{d} \frac{1}{\sqrt{\det(2\pi \boldsymbol{S}^{(t\star)})}} \exp\left[-\frac{1}{2}(\boldsymbol{\ell}_c^{(t_0)})^T (\boldsymbol{S}^{(t\star)})^{-1} (\boldsymbol{\ell}_c^{(t_0)})\right] \prod_{c=1}^{d} \mathcal{TN}(a_c^{(t\star)}, 0, v_a^{(c)})}{\exp(-\|(\mathbf{D}) \times_1 \boldsymbol{A}_1^{-1} \ldots \times_p (\boldsymbol{A}_p^{(t-1)})^{-1} \ldots \times_k \boldsymbol{A}_k^{-1}\|^2/2) \prod_{c=1}^{d} \frac{1}{\sqrt{\det(2\pi \boldsymbol{S}^{(t-1)})}} \exp\left[-\frac{1}{2}(\boldsymbol{\ell}_c^{(t_0)})^T (\boldsymbol{S}^{(t-1)})^{-1} (\boldsymbol{\ell}_c^{(t_0)})\right] \prod_{c=1}^{d} \mathcal{TN}(a_c^{(t-1)}, 0, v_a^{(c)})},$$

and compare r with the value of the uniform random variate $U \sim \mathcal{U}[0,1]$.
–If $u \geq r$, we reject the proposed values of the unknowns, and set the current value of the δ_c and a_c parameters at the end of the t-th iteration to be $a_c^{(t)} = a_c^{(t-1)}, \delta_c^{(t)} = \delta_c^{(t-1)}, \forall c = 1, \ldots, d$.
–If $u < r$, we accept the proposed values, and set $a_c^{(t)} = a_c^{(t\star)}, \delta_c^{(t)} = \delta_c^{(t\star)}, \forall c = 1, \ldots, d$.
Thus, the updating for the δ_c parameters is Random Walk as they are proposed from a Gaussian.

4 In this point, we discuss the updating of the remaining unknowns, $\sigma_1, \ldots, \sigma_{q_{max}}$, i.e. the elements of covariance matrices of the tensor-normal joint probability distribution of a set of realisations of $\boldsymbol{V}$ $(= \boldsymbol{\xi}(\boldsymbol{S}))$, that are not kernel-parametrised, but learnt directly by MCMC. These elements can in general be positive or negative, and so, in the t-th iteration, we propose them from a Truncated Normal with mean given by their current value $\sigma_q^{(t-1)}$, and experimentally fixed variance v_q, $q = 1, \ldots, q_{max}$, i.e. the proposed value is

$$\sigma_q^{(t\star)} \sim \mathcal{TN}(\sigma_q^{(t-1)}, v_q) \quad \forall q = 1, \ldots, v_q.$$

Then using these proposed values of the elements, the proposed values of all covariance matrices other than $\boldsymbol{\Sigma}_p$ that is kernel-parametrised, are $\boldsymbol{\Sigma}_1^{(t\star)}, \ldots, \boldsymbol{\Sigma}_{p-1}^{(t\star)}, \boldsymbol{\Sigma}_{p+1}^{(t\star)}, \ldots, \boldsymbol{\Sigma}_k^{(t\star)}$, while their current values (populated by the current values $\sigma_q^{(t-1)}$ of elements) are:
$\boldsymbol{\Sigma}_1^{(t-1)}, \ldots, \boldsymbol{\Sigma}_{p-1}^{(t-1)}, \boldsymbol{\Sigma}_{p+1}^{(t-1)}, \ldots, \boldsymbol{\Sigma}_k^{(t-1)}$. The prior probability densities on the σ_q parameters are treated as Gaussians with mean given by the seed value of $\sigma_q^{(0)}$ and experimentally chose, large variance, to suggest vague priors. Thus, the ratio of the posterior probability of the proposed and current $\sigma_1, \ldots, \sigma_{q_{max}}$, parameters, given the training data $\mathbf{D}$, at the already updated $\boldsymbol{\Sigma}_p$ to value $\boldsymbol{\Sigma}_p^{(t)} = (A_p^{(t)})^T A_p^{(t)}$, is

$$r := \frac{\exp(-\|(\mathbf{D}) \times_1 (\boldsymbol{A}_1^{(t\star)})^{-1} \ldots \times_{p-1} (\boldsymbol{A}_{p-1}^{(t\star)})^{-1} \times_p (\boldsymbol{A}_p^{(t)})^{-1} \times_{p+1} (\boldsymbol{A}_{p+1}^{(t\star)})^{-1} \ldots \times_k (\boldsymbol{A}_k(t\star))^{-1}\|^2/2)\pi_0(\sigma_1^{(t\star)}, \ldots, \sigma_{q_{max}}^{(t\star)})}{\exp(-\|(\mathbf{D}) \times_1 (\boldsymbol{A}_1^{(t-1)})^{-1} \ldots \times_{p-1} (\boldsymbol{A}_{p-1}^{(t-1)})^{-1} \times_p (\boldsymbol{A}_p^{(t\star)})^{-1} \times_{p+1} (\boldsymbol{A}_{p+1}^{(t-1)})^{-1} \ldots \times_k (\boldsymbol{A}_k(t-1))^{-1}\|^2/2)\pi_0(\sigma_1^{(t-1)}, \ldots, \sigma_{q_{max}}^{(t-1)})},$$

and compare r with the value of the uniform random variate $U \sim \mathcal{U}[0,1]$.
It is possible that some of these $k-1$ covariance matrices are not learnt using MCMC, but empirically estimated–in that case, the empirically estimated value of the corresponding covariance matrix is used in both denominator and numerator in the definition of the likelihood above, instead of its current and proposed values respectively. –If $u \geq r$, we reject the proposed values of the unknowns, and set the current value of the σ_q at the end of the t-th iteration to be $\sigma_q^{(t)} = \sigma_q^{(t-1)}, \forall q = 1, \ldots, q_{max}$.
–If $u < r$, we accept the proposed values, and set $\sigma_q^{(t)} = \sigma_q^{(t\star)}, \forall q = 1, \ldots, q_{max}$.
Thus, the updating for the σ_q parameters is Random Walk as they are proposed from a Gaussian.

5 Repeat steps 2-5 until $t = t_{max}$, the length of the chain.

From an inference point of view, it is important to remind ourselves that only those kernels that allow for positive definiteness of the corresponding covariance matrix, are to be used.

Since in this work, $\mathbf{\Sigma}_p$ is kernel parametrised, s.t. the diagonal elements of $\mathbf{\Sigma}_p$ designed to be unity, and any off-diagonal element is less than unity. So its positive definiteness is assured. The proposing of the diagonal elements (as well as the off-diagonal ones) of the other covariance matrices that are learnt directly from MCMC, is from a truncated Normal, thus ensuring positive definteness of the diagonal elements. In addition, at every iteration we will need to keep an eye on the overall adherence to postive definteness, and reject samples that imply deviation from positive definiteness of such covariance matrices that are learnt directly from MCMC. This is done in the application discussed below, by checking that the determinant of the only relevant, directly-learnt covariance matrix (that is 2×2-dimensional), is always positive.

If the "nested GP" model is not invoked at all, i.e., the length scale parameters $\ell_1, \ldots \ell_d$ of the kernel function of the kernel-parametrised covariance function $\mathbf{\Sigma}_p$ are treated as unknown constants that are learnt using MCMC, then steps 2(b) and 3(b) can be ignored in the above algorithm, and steps 2(a) and 3(a) imposed for all iterations, for all $t = 1, \ldots, t_{max}$. In that case, marginals (allowing for 95% HPDs) of $\sigma_1, \ldots, \sigma_{qmax}, \ell_1, \ldots, \ell_d$ are learnt. On the other hand, when the "nested GP" model is invoked, with the "lookback time" chosen to be set as t_0 iterations behind the current iteration, then marginals (with 95% HPDs) on $\sigma_1, \ldots, \sigma_{qmax}, a_1, \ldots, a_d, \delta_1, \ldots, \delta_d$ are obtained. We can, in this case, also generate values of $\ell_1, \ldots, \ell_d$ at the end of the first block update (on the a_c, δ_c parameters) in every iteration past iteration number t_0, since the generative GP that ℓ_c is a realisation of, is updated in this block update. Thus, values of $\ell_1, \ldots, \ell_d$ are predicted at every iteration, when this "nested GP" model is invoked.

2.5 Application

We are going to illustrate our method using an application on astronomical data. In this application, we are going to learn the location of Sun in the two dimensional Milky Way (MW) disk. The training dataset comprises a set of 2-dimensional velocity vectors of a sample of stars around the Sun, where such velocity vectors are generated via numerical simulations conducted with varying astronomical models of the Galaxy. Given each such astronomical model of the Galaxy, a set of 2-dimensional vectors of this sample of stars is recorded, i.e. the data recorded at each choice of the Galactic model, is a set of $p = 2$ velocity components of the same $k = 50$ stars, rendering the data recorded for each choice of the galactic model, a matrix with $k = 50$ rows and $p = 2$ columns. In fact, a total of $n = 216$ of such Galactic models were chosen, s.t. the full training data is a cuboid comprising $n = 216$ number of matrices, each of which is 50×2-dimensional. Each such chosen Galactic model corresponds to a choice of a 2-dimensional vector $\boldsymbol{S}$ that bears information about the Milky Way features. Then we treat this situation to state that at each value of the $d = 2$-dimensional vector-valued input variable $\boldsymbol{S} = (S_1, S_2)^T$, a $k \times p$-dimensional (50×2-dimensional) matrix-valued observable $\boldsymbol{V}$ is generated. There are $n = 216$ such choices of values of $\boldsymbol{S}$, i.e. there are n design points, and the value of the observable $\boldsymbol{V}$ at each of the n design points, is identifiable. Thus, the training data is $\mathbf{D} = \{(\boldsymbol{s}_i, \boldsymbol{v}_i)\}_{i=1}^{n}$.

At the same time, there exists the test data $\boldsymbol{v}^{(test)}$ that comprises the $p = 2$-dimensional velocity vectors of the same $k = 50$ number of stellar neighbours of the Sun, as measured by the Hipparcos satellite [Chakrabarty, 2007]. However, we do not know the real Milky Way feature parameter vector $\boldsymbol{s}^{(test)}$ at which $\boldsymbol{V} = \boldsymbol{v}^{(test)}$ is realised.

We frame this problem to state that $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$, and we want to predict the value $\boldsymbol{s}^{(test)}$ of $\boldsymbol{S}$, at which $\boldsymbol{V} = \boldsymbol{v}^{(test)}$, upon learning $\boldsymbol{\xi}(\cdot)$, using the training data $\mathbf{D}$.

Since we are observing the velocities of stars around the Sun, the observed velocities will be impacted by certain Galactic features. These features include location of the Sun. In other words, the observed velocities of our stellar neighbours, as expressed as the matrix $\boldsymbol{v}^{(test)}$ that is the measured test data, can be regarded as resulting from the Galactic features (including the sought solar location) to bear certain values. Put another way, the matrix of the observed velocities $\boldsymbol{V}$ is treated to be (functionally) related to the location vector of the Sun $\boldsymbol{S}$, and this way we can write $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$. On galactic length scales, the Earth-bound observer's location in the Milky Way disk is equivalent to the location of eh Sun in the Miky Way disk. So in the discussion below, the "observer's location" is held the same as the solar location.

Here for $\boldsymbol{V} \in \mathbb{R}^{m_1 \times m_2 \times m_2}$ and $\boldsymbol{S} \in \mathbb{R}^d$, $\boldsymbol{\xi} : \mathbb{R}^d \longrightarrow \mathbb{R}^{m_1 \times m_2 \times m_3}$. So, an observer located at the location $\boldsymbol{S} = \boldsymbol{s}_1$ will observe the neighbouring stars to have a velocity matrix that is different from the velocity matrix of the neighbouring stars measured by another observer at $\boldsymbol{S} = \boldsymbol{s}_2$, and the velocity matrix measured by observer at location $\boldsymbol{s}_i$ bears the stamp of this location, $i = 1, \ldots, n$.

We learn this function $\boldsymbol{\xi}(\cdot)$ using training data which includes n pairs of values of chosen solar location vector and the stellar velocity matrix observed at this solar location. Thus, the full training data is a 3-dimensional tensor which has the dimensionality of $m_1 \times m_2 \times n$. For the i-th slice of the tensor is a $m_2 \times m_1$ matrix, which is observed at the location $\boldsymbol{s}_i$, $i = 1, \ldots, n$. We use the astronomical simulated data (presented by [Chakrabarty, 2007]) as our training data. In this application, we will learn the covariance structure of the training data and predict the value of the solar/observer location parameter $\boldsymbol{S}$, at which the measured

or test data is realised.

In [Chakrabarty et al., 2015], the matrix of velocities was vectorised, so that the observable was then a vector. In our case, the observable is $\boldsymbol{V}$–a matrix.

By this process of vectorisation, [Chakrabarty et al., 2015] miss out on the opportunity to learn the covariance amongst the columns of the velocity matrix, (i.e. amongst the components of the velocity vector), distinguished from the covariance amongst the rows, (i.e. amongst the stars that are at distinct relative locations with respect to the observer). Our work allows for clear quantification of such covariances. More importantly, our work provides a clear methodology for learning given high-dimensional data comprising measurements of a tensor-valued observable.

In our application we realise that the location vector of the observer is 2-dimensional, i.e. d=2 since the Milky Way disk is assumed to be 2-dimensional. Also, each stellar velocity vector is also 2-dimensional, i.e. m_1=2. [Chakrabarty, 2007] generated such training data by first placing a regular 2-dimensional polar grid on a chosen annulus in an astronomical model of the MW disk. In the centroid of each grid cell, an observer was placed. There were n grid cells, so, there were n observers placed in this grid, such that the i-th observer measured the velocities of m_{2i} stars that landed in her grid cell, at the end of a simulated evolution of a sample of stars that were evolved in this model of the MW disk, under the influence of the feature parameters that mark this MW model. We indexed the m_{2i} stars by their location with respect to the observer inside the grid cell, and took a stratified sample of m_2 stars from this collection of m_{2i} stars while maintaining the order by stellar location inside each grid; $i = 1, \ldots, n$. Thus, each of the observers records a sheet of information that contains the 2-dimensional velocity vectors of m_2 stars, i.e. the training data $\mathbf{D}$ comprises n-number of $m_2 \times 2$-dimensional matrices, with each generated at a design point.

Then the velocity matrices that contribute to the training data is a 3-tensor. We call this the "observed tensor": $\boldsymbol{D}_{\boldsymbol{V}}^{(2\times m_2\times n)}$. We realise that the i-th velocity matrix or sheet in the training data $\mathbf{D}$, is realised at the observer location $\boldsymbol{s}_i$ that is the i-th design point in our training data. We use n=216 and m_2=50. The test data measured by the Hipparcos satellite is then the 217-th sheet, except we are not aware of the value of $\boldsymbol{S}$ that this sheet is realised at. We clarify that in this polar grid, observer location $\boldsymbol{S}$ is given by 2 coordinates: the first S_1 tells us about the radial distance between the Galactic centre and the observer, while the second coordinate of S_2 denotes the angular separation between a straight line that joins the Galactic centre to the observer, and a pre-fixed axis in the MW. This axis is chosen to be the long axis of an elongated bar of stars that lies pivoted at the Galactic centre, as per the astronomical model of the MW that was used to generate the training data.

As mentioned above, the maximum likelihood estimate of the mean tensor is removed from the data to allow us to work with a zero mean tensor normal density that represents the likelihood.

Since the data is the oserved 3-tensor $\boldsymbol{D}_{\boldsymbol{V}}$ (built of n observations of the 50×2-dimensional matrix-variate observable $\boldsymbol{V}$), the likelihood is a 3rd-order tensor Normal distribution, with zero mean tensor (following the removal of the estimated mean) and 3 covariance matrices that measure:

–amongst-observer-location covariance ($\boldsymbol{\Sigma}_3^{(216\times 216)}$),

–amongst-stars-at-different-relative-position-w.r.t.-observer covariance ($\boldsymbol{\Sigma}_2^{(50\times 50)}$), and

–amongst-velocity-component covariance ($\boldsymbol{\Sigma}_1^{(2\times 2)}$).

We perform kernel parametrisation of $\boldsymbol{\Sigma}_3$, using the SQE kernel such that the jp-th element of $\boldsymbol{\Sigma}_3$ is kernel-parametrised as $[\sigma_{jp}] = \exp\left(-(\boldsymbol{s}_j - \boldsymbol{s}_p)^T \boldsymbol{Q}^{-1}(\boldsymbol{s}_j - \boldsymbol{s}_p)\right), j, p = 1, \ldots, 216$. Since $\boldsymbol{S}$ is a 2-dimensional vector, $\boldsymbol{Q}$ is a $2\times$ 2 square diagonal matrix, the

elements ℓ_1 and ℓ_2 of which, represent the the correlation length scales.

Indeed, unless the "nested GP" model is invoked, this model of the covariance function suggests the same correlation length scales between the values the sampled function at any two points in its support, any two values of the input space variable $\boldsymbol{S}$, and this is a simplification. However, this non-nested GP model still implies that the learning of the 216×216-dimensional covariance matrix $\boldsymbol{\Sigma}_3$, has been reduced to the learning of 2 length scale parameters ℓ_1, ℓ_2. Here the global amplitude of the kernel function is subsumed into the global scale that multiplies the covariance matrix $\boldsymbol{\Sigma}_1$, elements of which are learnt directly by MCMC.

Under the "nested GP" model, ℓ_c is modelled as a realisation from a scalar-variate, zero-mean GP, s.t. the joint probability of t_0-number of realisations of ℓ_c–obtained over the last t_0 iterations of the Metropolis-within-Gibbs chain–is multivariate normal, with covariance matrix $\boldsymbol{S}_c = [s_{\alpha\beta}^{(c)}] = \left[a_c \exp\left(-\frac{(\alpha-\beta)^2}{2\delta_c^2}\right)\right], \alpha, \beta = t - t_0, \ldots, t-1, c = 1, 2$. We are free to choose t_0, and choose it as large as possible, urged to achieve as wide a coverage of the chain's history as possible, subject to computational constraints–after all, the bigger is t_0, the larger is the matrix $\boldsymbol{S}$, and therefore more computationally challenging is the inversion of $\boldsymbol{S}$, that we need to undertake at every iteration of our MCMC-based inference scheme, (see point 3(b) of the algorithm, discussed above). In light of this challenge, we use t_0 to typically be about 10^2, with $t_0 \leq 500$ in all our experiments. Thus, at each iteration, under the invoked "nested GP" model, we update $a_1, a_2, \delta_1, \delta_2$, and sample a value of each of ℓ_1 and ℓ_2, from the updated scalar-variate, zero-mean GP that this parameter is treated as a realisation from. For the 0-th to the $t_0 - 1$-th iterations however, the "nested GP" model is not implemented, and we learn the ℓ_1 and ℓ_2 parameters only, in the first block update of our Metropolis-within-Gibbs scheme.

$\mathbf{\Sigma}_1$ measures covariance amongst the matrices or sheets obtained at distinct components of the velocity vector. As there are only such 2 components, there are 2 such sheets. However, we are not aware of any input variable at which these sheets are realised. Therefore we need to learn the 4 elements of this matrix directly from MCMC. As the covariance matrix is symmetric, we need to learn only 3 of the 4 parameters. We are going to learn the two diagonal elements and one non-diagonal element in the $\mathbf{\Sigma}_1$ matrix. The two diagonal elements will be learnt by our MCMC algorithm directly. However, the non-diagonal element $\sigma_{12}^{(1)}$ can be written as $\sigma_{12}^{(1)} = \rho\sqrt{\sigma_{11}^{(1)}\sigma_{22}^{(1)}}$ where ρ is the correlation amongst these two vertical sheets in the training data on the observable $\boldsymbol{V}$. Thus, instead of learning the $\sigma_{12}^{(1)}$ directly, we choose to learn the correlation parameter ρ, using our MCMC algorithm.

The elements of the $m_2 \times m_2 = 50 \times 50$-dimensional $\mathbf{\Sigma}_2$ covariance matrix are not learnt by MCMC. Firstly, there is no input space variable that can be identified, at which the ij-th element of $\mathbf{\Sigma}_2$ can be considered to be realised, where i and j are arbitrarily assigned indices of a pair of stars observed in the 50-star-strong sample of stellar neighbours of the Sun, i.e. $i, j = 1, \ldots, 50$. So, the ij-th element of $\mathbf{\Sigma}_2$ gives the covariance amongst the i-th and j-th, 216-dimensional matrices of the 2-dimensional velocity vectors respectively, of the i-th and j-th sampled stars, generated in the astronomical simulation (that generates the training data), at the 216 different chosen values of the Sun's location on the Milky Way disk; $\forall i, j = 1, \ldots, 50$. Effectively, the 41st star could have been referred to as the 3rd star in this sample, and the vice versa, i.e. there is no meaningful ordering in the labelling of the sampled stars with these indices. Therefore, we cannot use these labels as input space variable, in terms of which, the covariance between the i-th and j-th 216×2-dimensional velocity matrices can be kernel-parametrised. One possibility then is to learn elements in the upper (or lower) triangle of $\mathbf{\Sigma}_2$ directly, using MCMC. However, there are 25×49

number of such elements, which is too large a number to allow for direct MCMC-based learning. In light of this, we will perform empirical estimation of the elements of $\mathbf{\Sigma}_2$. So the ij-th element of $\mathbf{\Sigma}_2$, i.e. the covariance between the 216×2-dimensional matrix $\boldsymbol{V}_i := [v_{pq}^{(i)}]$ of the i-th and the matrix $\boldsymbol{V}_j := [v_{pq}^{(j)}]$ of the j-th labelled sampled stars, $(p = 1, \ldots, 216; q = 1, 2)$, is estimated as $\widehat{\sigma_{ij}^{(2)}}$, where:

$$\widehat{\sigma_{ij}^{(2)}} = \frac{1}{2-1} \times \sum_{q=1}^{2} \left[\frac{1}{216} \times \left(\sum_{p=1}^{216} (v_{pq}^{(i)} - \bar{v}_q^{(i)}) \times (v_{pq}^{(j)} - \bar{v}_q^{(j)}) \right) \right],$$

where $\bar{v}_q^{(i)} = \dfrac{\left(\sum_{p=1}^{216} v_{pq}^{(i)}\right)}{216}$ is the sample mean of the q-th column of the matrix $\boldsymbol{V}_i = [v_{pq}^{(i)}]$.

Thus, within the non-"nested GP" model, from the training data, we have 5 parameters to learn: ℓ_1,ℓ_2,$\sigma_{11}^{(1)}$,ρ,$\sigma_{22}^{(1)}$, where these parameters are defined as in:

$$\boldsymbol{Q} = \begin{pmatrix} \ell_1 & 0 \\ 0 & \ell_2 \end{pmatrix}; \mathbf{\Sigma}_3 = \begin{pmatrix} \sigma_{11}^{(1)} & \sigma_{12}^{(1)} \\ \sigma_{12}^{(1)} & \sigma_{22}^{(1)} \end{pmatrix}; \rho = \frac{\sigma_{12}^{(1)}}{\sqrt{\sigma_{11}^{(1)} \sigma_{22}^{(1)}}}$$

The likelihood of the GP parameters given the training data is then given as per Equation 2.1:

$$\begin{aligned} f(\mathbf{D}|\ell_1, \ell_2, \sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \rho) &= (2\pi)^{-m/2} \left(\prod_{i=1}^{3} |\mathbf{\Sigma}_i|^{-m/2m_i}\right) \\ &\times \exp(-\|(\mathbf{D} - \hat{\boldsymbol{M}}) \times_1 {\boldsymbol{A}_1}^{-1} \times_2 {\hat{\boldsymbol{A}}_2}^{-1} \times_3 \boldsymbol{A}_3^{-1}\|^2/2). \end{aligned} \tag{2.7}$$

where $\mathbf{\Sigma}_p = \boldsymbol{A}_p \boldsymbol{A}_p^T$, $p = 1, 2, 3$ and $\hat{\boldsymbol{M}}$ is the empirical estimate of the mean tensor and $\hat{\mathbf{\Sigma}_2}$ is the empirical estimate of the covariance matrix $\mathbf{\Sigma}_2$ such that $\hat{\mathbf{\Sigma}_2} = \hat{\boldsymbol{A}}_2 {\hat{\boldsymbol{A}}_2}^T$. Here $m_3 = 216$, $m_2 = 50$, $m_1 = 2$, and $m = m_1 m_2 m_3$.

This allows us to write the joint posterior probability density of the unknown parameters given the training data $\mathbf{D}$. We generate posterior samples from it using MCMC. To write this posterior density, we impose non-informative prior probability densities $\pi_0(\cdot)$ on each

of our unknowns (Gaussian with wide, experimental chosen variances, and mean that is the arbitrarily chosen seed value of $\ell.$, and Jeffry's on $\mathbf{\Sigma}_1$). The posterior probability density of our unknown GP parameters, given the training data is then

$$\begin{aligned}&\pi(\ell_1, \ell_2, \sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \rho|\mathbf{D}) \propto \\ &\ell(\mathbf{D}|\mathbf{\Sigma}_1, \mathbf{\Sigma}_3) \times \pi_0(\ell_1)\pi_0(\ell_2)\pi_0(\mathbf{\Sigma}_1).\end{aligned} \tag{2.8}$$

The results of our learning and estimation of the mean and covariance structure of the GP used to model this tensor-variate data, is discussed below in Section 2.7.

Within the "nested GP" model, from the training data, we have 7 parameters to learn: $a_1, a_2, \delta_1, \delta_2, \sigma_{11}^{(1)}, \rho, \sigma_{22}^{(1)}$, where the learnt values of a_c, δ_c, $(c = 1, 2)$ inform on the zero-mean multivariate Normal joint probability density of t_0 realisations of the length scale parameter ℓ_c. In every iteration, a value of ℓ_c is sampled, from the updated (at the updated a_c, δ_c values), scalar-variate GP that is the generative process for ℓ_c.

The joint posterior probability density of the unknown parameters given the training data $\mathbf{D}$, under the "nested GP" model is given by

$$\begin{aligned}&\pi(\delta_1, \delta_2, a_1, a_2.\ell_1, \ell_2, \sigma_{11}^{(1)}, \sigma_{22}^{(1)}, \rho|\mathbf{D}) \propto \\ &(2\pi)^{-m/2}(\prod_{i=1}^{3}|\mathbf{\Sigma}_i|^{-m/2m_i}) \\ &\times \exp(-\|(\mathbf{D} - \hat{\boldsymbol{M}}) \times_1 \boldsymbol{A}_1{}^{-1} \times_2 \hat{\boldsymbol{A}}_2{}^{-1} \times_3 \boldsymbol{A}_3^{-1}\|^2/2) \\ &\times \prod_{c=1}^{2} \frac{1}{\sqrt{\det(2\pi \boldsymbol{S})}} \exp\left[-\frac{1}{2}(\boldsymbol{\ell}_c^{(t_0)})^T \boldsymbol{S}^{-1}(\boldsymbol{\ell}_c^{(t_0)})\right] \times \pi_0(\mathbf{\Sigma}_1),\end{aligned} \tag{2.9}$$

where $\boldsymbol{\ell}_c^{(t_0)} := (\ell_c^{(t-t_0)}, \ldots, \ell_c^{(t-1)})^T$, and $\boldsymbol{S}_c = \left[a_c \exp\left[-\frac{(i-j)^2}{2(\delta_c)^2}\right]\right]$.

We generate posterior samples using MCMC, to identify the marginal posterior probability distribution of each unknown. The marginal then allows for the computation of the 95% HPD.

2.6 Predicting

The aim of learning all GP parameters, i.e. parametrising the generative process that gives rise to the function $\boldsymbol{\xi}(\cdot)$ (where the observable $\boldsymbol{V} = \boldsymbol{\xi}(\boldsymbol{S})$), is that we are going to predict the value $\boldsymbol{s}^{(test)}$ of the input variable $\boldsymbol{S}$, at which test data on $\boldsymbol{V}$ is realised. In the context of our application, $\boldsymbol{s}^{(test)}$ is the location vector of the Sun in the Milky Way disk, i.e. it is the location of the observer who observes the velocity matrix of a sample of nearby stars, as included in the test data. This test data $\boldsymbol{v}^{(test)}$ is a 50×2 matrix that includes measurements of velocities of 50 stars that are neighbours of the Sun. We use two different methods for making inference on $\boldsymbol{s}^{(test)} = (s_1^{(test)}, s_2^{(test)})^T$, in next section.

In one method we learn the GP parameters and $s_1^{(test)}$ and $s_2^{(test)}$ simultaneously from the same MCMC chain run using both training and test data. The tensor that includes both test and training data has dimensions of $217 \times 50 \times 2$. We call this augmented data $\boldsymbol{D}^* = \{\boldsymbol{v}_1, ..., \boldsymbol{v}_{50}, \boldsymbol{v}^{(test)}\}$, to distinguish it from the observed tensor $\boldsymbol{D_V}$ that contributes to the training data $\mathbf{D}$. This 217-th sheet of (test) data is realised at the unknown value $\boldsymbol{s}^{(test)}$ of $\boldsymbol{S}$, and upon its addition, the updated covariance amongst the sheets generated at the different values of $\boldsymbol{S}$, is renamed $\boldsymbol{\Sigma}_1^*$, which is now rendered 217×217-dimensional. Then $\boldsymbol{\Sigma}_1^*$ includes information about $\boldsymbol{s}^{(test)}$ via the SQE-based kernel parametrisation discussed in Section 2.2.1. The effect of the inclusion of the test data on the other covariance matrices is less; we refer to them as (empirically estimated) $\hat{\boldsymbol{\Sigma}}_2^*$ and $\boldsymbol{\Sigma}_3^*$. The updated (empirically estimated) mean tensor is $\hat{\boldsymbol{M}}^*$. The likelihood for the augmented data is:

$$
\begin{aligned}
f(\boldsymbol{D}^* | \boldsymbol{s}^{(test)}, \boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_3^*) = (2\pi)^{-m/2} \left(\prod_{i=1}^{3} |\boldsymbol{\Sigma}_i^*|^{-m/2m_i} \right) \times \\
\exp \left[-\| (\boldsymbol{D}^* - \hat{\boldsymbol{M}}^*) \times_1 (\boldsymbol{A}_1^*)^{-1} \times_2 (\hat{\boldsymbol{A}}_2^*)^{-1} \times_3 (\boldsymbol{A}_3^*)^{-1} \|^2 / 2 \right]
\end{aligned}
\tag{2.10}
$$

where $\hat{\boldsymbol{A}}_2^*$ is the square root of $\hat{\boldsymbol{\Sigma}}_2^*$. Here $m_1 = 217$, $m_2 = 50$, $m_3 = 2$, and $m = m_1 m_2 m_3$.

Here $\boldsymbol{A}_1^*$ is the square root of $\boldsymbol{\Sigma}_1^*$ and depends on $\boldsymbol{s}^{(test)}$.

The posterior probability density of the unknowns given the test+training data is:

$$\pi(s_1^{(test)}, s_2^{(test)}, \boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_3^* | \boldsymbol{D}^*) \propto$$

$$f(\boldsymbol{D}^* | s_1^{(test)}, s_2^{(test)}, \boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_3^*) \times$$
$$\pi_0(s_1^{(test)}) \pi_0(s_2^{(test)}) \pi_0(q_2^{(*)}) \pi_0(q_1^{(*)}) \pi_0(\boldsymbol{\Sigma}_3^*) \tag{2.11}$$

.

As discussed above, we use non-informative prior probability densities on all GP parameters and uniform priors on $s_1^{(test)}$ and $s_2^{(test)}$. So $\pi_0(s_p^{(test)}) = \mathcal{U}(l_p, u_p)$, $p = 1, 2$, where l_p and u_p are chosen depending on the spatial boundaries of the fixed area of the Milky Way disk that was used in the astronomical simulations of [Chakrabarty, 2007]. Recalling that the observer is located in a two-dimensional polar grid, [Chakrabarty, 2007] set the lower boundary on the value of the angular position of the observer to 0 and the upper boundary is $\pi/2$ radians, i.e. 90 degrees, where the observer's angular coordinate is the angle made by the observer-Galactic centre line to the long-axis of the elongated Galactic bar made of stars that rotates pivoted at the Galactic centre (discussed in Section 1). The observer's radial location is maintained within the interval [1.7,2.3] in model units, where the model units for length are related to galactic unit for length, as discussed in Section 2.7.2.

In the second method, we infer $\boldsymbol{s}^{(test)}$ by sampling from the posterior density of $\boldsymbol{s}^{(test)}$ given the test+training data and the modal values of the parameters $q_1, q_2, \sigma_{11}^{(1)}, \rho, \sigma_{22}^{(1)}$ that were learnt using the training data. The modal value of $\boldsymbol{\Sigma}_3$, learnt using trainng data alone is $[(\sigma_3^{(M)})_{jp}]_{j=1;p=1}^{217,217}$, where $\boldsymbol{\Sigma}_3^\star = (\sigma_3^{(M)})_{jp} = \left[\exp\left(-(\boldsymbol{s}_j - \boldsymbol{s}_p)^T \boldsymbol{Q}^{(M)} (\boldsymbol{s}_j - \boldsymbol{s}_p)\right)\right]$, with the unknown $\boldsymbol{s}_{217} = \boldsymbol{s}^{(test)}$ and the diagonal elements of $\boldsymbol{Q}$ given as the modal values $q_1^{(M)}$ and $q_2^{(M)}$ that were learnt using training data alone. Similarly, $\boldsymbol{\Sigma}_1$ is retained as the modal

value $\boldsymbol{\Sigma}_1^{(M)}$ that was learnt using the training data alone.

The posterior probability density of $\boldsymbol{s}^{(test)}$, at learnt (modal) values is then

$$\begin{aligned}
&\pi(s_1^{(test)}, s_2^{(test)} | \boldsymbol{D}^*, \boldsymbol{\Sigma}_1^{(M)}, \boldsymbol{\Sigma}_3^{\star}) \propto \\
&f(\boldsymbol{D}^* | s_1^{(test)}, s_2^{(test)}, \boldsymbol{\Sigma}_1^{(M)}, \boldsymbol{\Sigma}_3^{\star}) \times \pi_0(s_1^{(test)}) \pi_0(s_2^{(test)}) \\
&\times \pi_0(q_2^{(M)}) \pi_0(q_1^{(M)}) \pi_0(\boldsymbol{\Sigma}_3) | \boldsymbol{V}^*).
\end{aligned} \tag{2.12}$$

where $f(\boldsymbol{D}^* | s_1^{(test)}, s_2^{(test)}, \boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_3^{(M)})$ is as given in Equation 2.3, with $\boldsymbol{\Sigma}_3^*$ replaced by $\boldsymbol{\Sigma}_3$, and $\boldsymbol{\Sigma}_1$ replaced by its modal value $\boldsymbol{\sigma}_1^{(M)}$. The prior probability densities on $s_1^{(test)}$ and $s_2^{(test)}$ are as discussed above. For all parameters, we use Normal proposal densities that have experimentally chosen variances.

Prediction according to this second method is faster, since the number of parameters that we are learning is less than when we sample according to the first method suggested above, i.e. sampling from the joint posterior of all uknowns ($s^{(test)}$ as well as relevant uknown parameters of the general tensor-Normal density that represents our likelihood), given all data (training and test). In fact, in that first method, we are suggesting making inference on all relevant parameters anew, every time we have a new test datum. This may indeed be an overkill. However, in this method, concerns about mis-representative nature of the training data, may stand mitigated. For the sake of quicker prediction, the second method of predicting is used.

2.7 Results

In this section, we present the results of learning the unknown parameters of the 3rd-order tensor-normal likelihood, given the training as well as the training+test data.

While Figure 2.1 and Figure 2.2 depict results obtained from using the "non-nested"

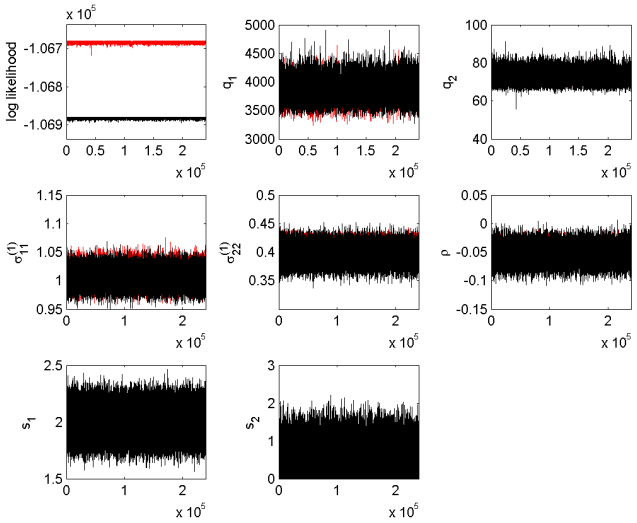

Figure 2.1: Results from run done with observed matrix $\boldsymbol{D_V}$ that contributes to the training data $\mathbf{D}$, with only the top layer of GP included in the model, are shown in grey (or red in a coloured or the electronic copy of the thesis) while results from run undertaken with training and test data, $\boldsymbol{D}^\star$, in this "non-nested" model, are depicted in black. Traces of the logarithm of the likelihood are displayed from the two runs in the top left panel. Reciprocal of the length scale parameters are the shown in the top middle and right panels; here $q_c = \ell_c^{-1}$, $c = 1, 2$. Traces of the learnt diagonal elements $\sigma_{11}^{(1)}$ and $\sigma_{22}^{(1)}$, of the covariance matrix $\boldsymbol{\Sigma}_1$, are shown in the mid-row, left and middle panels. Trace of the correlation $\rho = \dfrac{\sigma_{12}}{\sqrt{\sigma_{11}^{(1)}\sigma_{22}^{(1)}}}$ is displayed in the mid-row right panel. Prediction of the values of the input parameter $\boldsymbol{S} = (S_1, S_2)^T$ is possible only in the run performed with both training and test data. Traces of S_1 and S_2 values learnt via MCMC-based sampling from the joint posterior probability density of all unknown parameters given $\boldsymbol{D}^\star$, are shown in the lower panel.

.

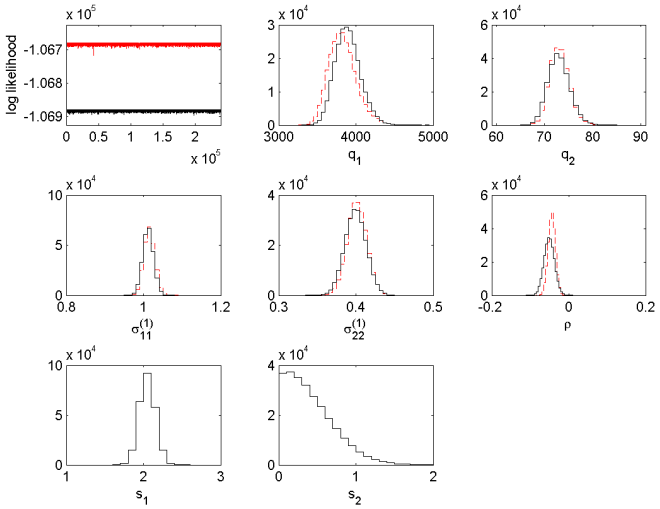

Figure 2.2: Results from run done with observed matrix $\boldsymbol{D}_V$ that contributes to the training data $\mathbf{D}$, with only the top layer of GP included in the model, are shown in grey (or red in a coloured or the electronic copy of the thesis) while results from run undertaken with training and test data, $\boldsymbol{D}^\star$, in this "non-nested" model, are depicted in black. The histograms presented here represent histograms of the marginal posterior probability distributions of a parameter given training data $\mathbf{D}$ (in grey broken lines–or red broken lines, in the electronic version), and given test+training data $\boldsymbol{D}^\star$ in black solid lines.

.

GP, in the Figures 2.3, 2.4, 2.5, 2.6, results of the learning of from the "nested" GP models are included. In this "nested" GP model, the covariance matrix $\mathbf{\Sigma}_3$ (that bears information about the covariance structure between sheets of data generated at different values of the input variable $\boldsymbol{S} = (S_1, S_2)^T$), is parameterised using a kernel, each length-scale hyperparameter of which is itself considered sampled from a GP. For each such scalar-variate GP that generates the length-scale ℓ_c, $c = 1, \ldots, d = 2$ the covariance matrix is itself kernel-parametrised using a stationary kernel, with an amplitude parameter A_c and length-scale parameter δ_c. Figures 2.3 to 2.6 that depict results from the nested-GP approach will then include results of the learning of values of these A_c and value δ_c of the reciprocal of the δ_c parameters. Also, our modelling under the nested-GP paradigm relies on a lookback-time parameter T_0, which gives a chosen number of iterations. To quickly recap the essence of T_0 as discussed earlier in Section 2.5: we model ℓ_c as a function of the iteration number, where this function is modelled as a realisation from a scalar-variate GP, s.t. the joint probability of t_0-number of realisations of this function (i.e. the t_0-number of values of ℓ_c generated in these t_0 iterations), is Multivariate Normal–the $t_0 \times t_0$ covariance matrix of which is kernel-parametrised using the stationary kernel with hyperparameters A_c and δ_c. Thus, T_0 is a global model parameter, which once set, gives rise to a nested-GP model. As said above in Section 2.5, we do not exceed $t_0 = 500$ in any of our runs.

One difference between the learning of parameters from the nested-GP, as distinguished from the non-nested GP models is the quality of the inference, in the sense that the uncertainty of parameters (i.e. the 95% HPDs) learnt using the nested-GP models, is less than that learnt using the non-nested GP models. This difference in the learnt HPDs is most marked for the learning of values of Q_1 and S_1, and S_2 is a lesser extent. We will attempt explaining this later, by invoking the discontinuity in the data, the kernel-parametrised (as

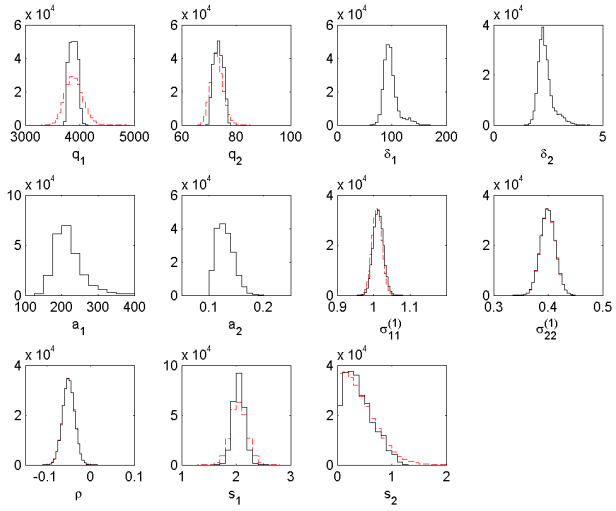

Figure 2.3: Results from run done with test+training data $\boldsymbol{D}^{\star}$ within the nested-GP model, shown in black, as distinguished from the results of learning given the same data, and the "non-nested" GP model depicted in grey (or red in the electronic copy of the thesis). Here the used value of T_0 is 200 iterations. Histograms approximating the marginal posterior probability densities of each sought unknown is depicted. Here d_c is defined as the value of the reciprocal of δ_c. Indeed, the hyperparameters A_c and δ_c are relevant only to the nested-GP model ($c = 1, 2$). Here, we have undertaken sampling from the joint posterior of all parameters, including the input parameter values $s_1^{(test)}$ and $s_2^{(test)}$, at which the test data are realised. Histograms approximating marginal posterior probability densities of each learnt unknown are presented.

.

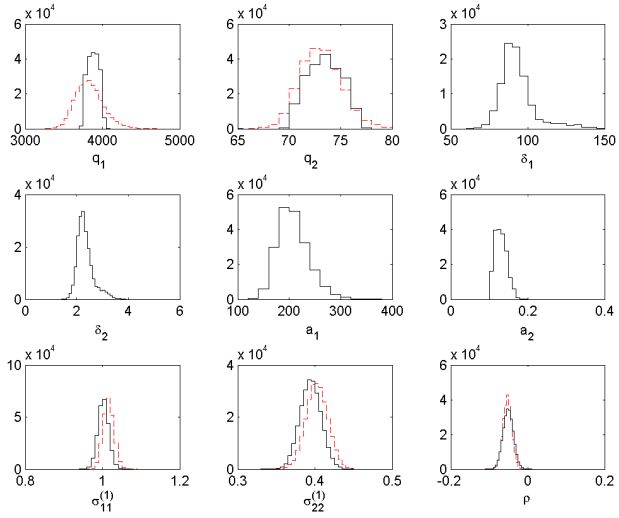

Figure 2.4: Results from run done with observed tensor $\boldsymbol{D_V}$ that contributes to the training data, within the nested-GP model, shown in black, as distinguished from the results of learning given the same data, and the "non-nested" GP model depicted in grey (or red in the electronic copy of the thesis). Here the used value of T_0 is 200 iterations. Histograms approximating the marginal posterior probability densities of each sought unknown is depicted. Here d_c is defined as the value of the reciprocal of δ_c. Indeed, the hyperparameters A_c and δ_c are relevant only to the nested-GP model ($c = 1, 2$). Histograms approximating marginal posterior probability densities of each learnt unknown are presented.

.

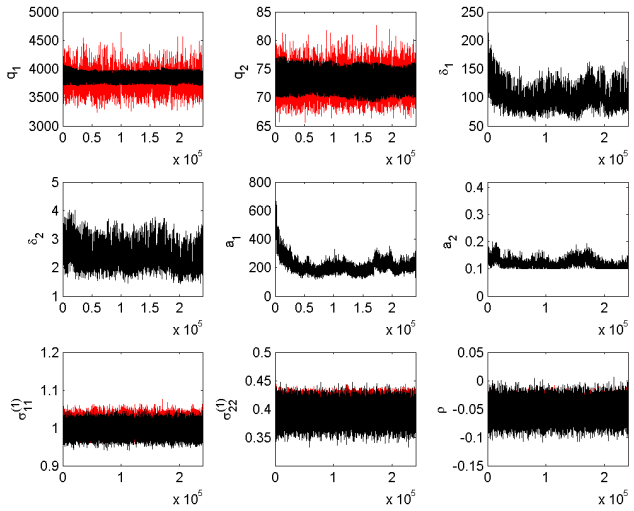

Figure 2.5: Same as in Figure 2.4, except that here we present traces of learnt parameters. Traces of parameters learnt within the non-nested are in grey (or red in the e-version) while the traces obtained using the nested-GP model are shown in black.

.

a function of $\boldsymbol{S}$) covariance matrix ($\boldsymbol{\Sigma}_3$) of which, is affected by a sharply discontinuous probability distribution of S_1, and a less sharp discontinuity in the distribution of S_2.

We refer to Figure 8 of Chakrabarty [2007] (that corresponds to the base astronomical model used in the simulations that generate the training data that we employ), in evidence; this figure tells us about the distribution of location $\boldsymbol{S}$, by compatibility of the stellar velocity matrix $\boldsymbol{v} = \boldsymbol{\xi}(\boldsymbol{s})$ realised (in astronomical simulations) at an $\boldsymbol{s}$, to the test velocity matrix $\boldsymbol{v}^{(test)}$ (recorded by the *Hipparcos* satellite). 50×2-dimensional matrix-variate, stellar velocity r.v. $\boldsymbol{V}$, takes This is a contour plot of the distribution of such a compatibility parameter, in the space $\mathcal{D}$–where $\boldsymbol{S} \in \mathcal{D} \subset \mathbb{R}^2$; here the two components of $\boldsymbol{S}$ are represented in polar coordinates, with S_1 the radial and S_2 the angular component. We see clearly from this figure, that the distribution across S_1 is highly discontinuous, at given

values of S_2 (i.e. at fixed angular bins). In fact, this distribution is visually more discontinuous, than the distribution across S_2, at given values of S_1, i.e. at fixed radial bins (each of which is represented by an arcs between two bounding radii). In other words, the velocity matrices that are astronomically simulated at different $\boldsymbol{S}$ values, are differently compatible with a reference velocity matrix ($\boldsymbol{v}^{(test)}$)–and, this difference is discontinuous across vales of $\boldsymbol{S}$. Thus, this figure indicates the discontinuity in the training data, with the input-space variable $\boldsymbol{S}$. Then, it is incorrect to use a stationary kernel to parametrise the covariance $\boldsymbol{\Sigma}_3$, that informs on the covariance between such velocity matrices. Our implementation of the nested-GP model tackles this shortcoming of the model. However, when we implement the non-nested GP model, the inferential algorithm (Metropolis) needs to explore a wider volume of the state space to accommodate parameter values, given the data at hand–and even then, there is a possibility for incorrect inference under the stationary kernel model. This explains the noted trend of higher 95% HPDs on most parameters learnt using the non-nested GP model, compared to the nested-GP model, as observed in comparison of results from runs done with training data alone, or both training and test data (i.e. when prediction from the joint posterior probability density of all parameters give all data is undertaken, to learnt $\boldsymbol{s}^{(test)}$); see Figure 2.3, Figure 2.4, Figure 2.5. Indeed, this also explains the bigger difference noted in these figures when we compare the learning of q_1 over q_2, in runs that use the stationary model, as distinguished from the non-stationary model. After all, the discontinuity across S_1 is discussed above, to be higher than across S_2.

To check for the effect of the lookback time (parametrised by T_0) we present traces of the parametrised kernel parameters and hyperparameters learnt from runs undertaken within the nested-GP model, at different T_0 values of 50 and 100, in Figure 2.6, which we can compare to the traces obtained in runs performed under the nested-GP model, with

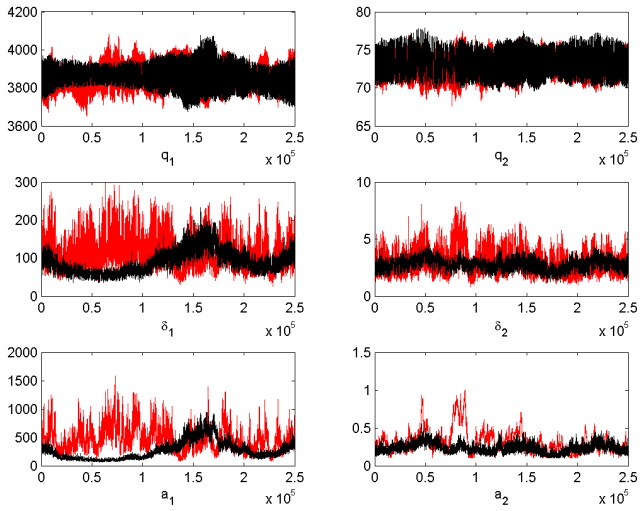

Figure 2.6: Comparison of traces of unknown smoothness parameters of $\mathbf{\Sigma}_3$ and hyperparameters of GPs invoked to model these parameters, obtained in runs performed with training data $\mathbf{D}$ and $t_0 = 50$ (in grey, or red in the e-version) and $t_0 = 100$ (in black).
.

$t_0 = 200$, as displayed in Figure 2.5.

It is indeed interesting to note the trends in traces of the smoothness, i.e. the reciprocal of ℓ parameters, and the amplitude A and value δ of the correlation-length hyperparameters (evidenced in Figure 2.6 and the results in black in Figure 2.5). We note the increase in the amplitude of the fluctuations in the traces of these hyperparameters with decreasing t_0. For smaller values of lookback time T_0, the average covariance between $g_c(t_1)$ and $g_c(t_2)$ is higher than when t_0 is higher, where the averaging is performed over a t_0-iteration long interval that has its right edge on the current iteration; here $c = 1, 2$ and as introduced above, we model the length scale parameter of the kernel that parametrises $\mathbf{\Sigma}_3$, as $\ell_c = g_c(T)$. Here $g_c(\cdot)$ is modelled as a realisation from a scalar-variate GP with covariance matrix $\mathbf{S}_c$ that is itself kernel-parametrised using an SQE kernel with amplitude A_c and

correlation-length δ_c. Then higher covariances between values of $g(\cdot)$ at different T-values in general would suggest higher values of the global amplitude of this parametrised kernel, and higher values of the length-scales of this SQE kernel.

A very interesting trend noted in the parameter traces presented in Figure 2.6 for $t_0 = 50, 100$, and to a lesser extent for $t_0 = 200$, in the results in black in Figure 2.5, is the global near-periodic existence of crests and troughs in these traces. This periodic fluctuation is more marked for q_1 values and the parameters of the scalar-variate GP used to model $g_1(\cdot)$ from, (where the reciprocal of the smoothness Q_1 is $\ell_1 = g_1(T)$), than for q_2 (and a_2 and δ_2). Let us first seek an explanation for this observed global periodic trend–which is a difficult undertaking in itself, and then explore comparison of the strength of this periodicity across different sets of parameters.

Above, we have convinced ourselves of the fact that the covariance ($\mathbf{\Sigma}_3$) amongst velocity slices simulated (astronomically) at different S values, is incorrectly parametrised with a stationary kernel, and that a non-stationary kernel is required to model such covariance. We implement such non-stationarity by kernel parametrising $\mathbf{\Sigma}_3$ using a kernel, the length-scale ℓ of which is itself stochastic–this stochasticity is generated by a scalar-variate GP according to our model. At any given iteration of our inference scheme, the said model is informed by invoking the past t_0 realisations from such a stochastic process. Then, loosely speaking, the value of ℓ in any iteration, is a moving average over the values indicated over this t_0-iterations wide window–and such a moving average will manifest the result of superposition of the different (discontinuous) modal neighbourhoods present in the data. The more multimodal the data, i.e. larger the number of "classes" (by correlation-length scales) of functional form $\boldsymbol{\xi}(\cdot)$ sampled from the tensor-variate GP, superposition of the sample paths will cause a washing-out of the effect of the different modes, and a less prominent

global trend will be manifest in the traces. However, for data that is globally bimodal, the superposition of the two "classes" of sampled functions $\boldsymbol{\xi}(\cdot)$ will create a periodicity in the global trend of the generated ℓ values (and thereby of the smoothness parameter values q, where $q = \ell^{-1}$).

Again, the larger the value t_0 of the lookback-time parameter, the moving average is over a larger number of samples, and hence greater is the washing-out effect. In fact, depending on the discontinuity in the data, it is anticipated that there is range of optimal lookback-time values, s.t. the global periodicity is most marked. Thus, the trace of q_1 at $t_0 = 100$ displays the global periodicity more strongly than that at $t_0 = 200$ (see Figure 2.6 and Figure 2.5). At the same time, the expected higher amplitude in the trace of q_1 with decreasing t_0, also make the visual identification of this trend more difficult at lower, than at higher t_0; this is clear in the comparison of the trends of the trace of q_1 generated at t_0=50, as compared to t_0=100 (see Figure 2.6). Another point is that the strength this global periodic trend will be stronger for the correlation-length scale along that direction in input-space, the discontinuity along which is stronger. Indeed, as we have discussed above, the discontinuity in the data along S_1 is anticipated to be higher than along S_2. So we would expect a more prominent periodic trend in the trace of q_1 than q_2. This is indeed what to note in Figure 2.6. A simulation study can be undertaken to explore the effects of empirical discontinuities.

The arguments above qualitatively explain the observed trends in the traces of the hyperparameters, obtained from runs using different t_0. That in spite of discrepancies in A_c and δ_c, with t_0, values of the length scale parameter ℓ_c (and therefore its reciprocal q_c) are concurrent within the 95% HPDs, is testament to the robustness of the model. Stationarity of the traces betrays the achievement of convergence of the chain. 95% HPD credible

regions computed on each learnt parameter given training data alone, are displayed in Table 2.1.

Table 2.1: 95% HPD credible regions on each learnt parameter, using three different inference schemes

Parameters	using only training data	sampling from posterior predictive	sampling from joint posterior density
q_1	[4572.4,5373.2]		[4566.8,5460.4]
q_2	[82.50,93.12]		[82.30,93.44]
$\sigma_{11}^{(1)}$	[0.9884,1.0337]		[0.9848,1.0310]
ρ	[-0.0627,-0.0310]		[-0.0620,-0.0304]
$\sigma_{22}^{(1)}$	[0.4087,0.4270]		[0.4116,0.4306]
s_1	-	[1.7496,2.0995]	[1.7547,2.0816]
s_2	-	[0.079,0.7609]	[0.0393,0.8165]

From Table 2.1, we notice that the reciprocal correlation length scale q_1 is a couple of orders of magnitude higher than q_2; correlation between values of the sampled function $\boldsymbol{\xi}(\cdot)$, at 2 different S_1 values (at the same s_2), then wanes more quickly than correlation between sampled functions computed at same s_1 and different S_2 values. Here $\boldsymbol{s} = (s_1, s_2)^T$ and given that $\boldsymbol{S}$ is the location of the observer who observes the velocities of her neighbouring stars on a two-dimensional polar grid, S_1 is interpreted as the radial coordinate of the observer's location in the Galaxy and S_2 is the observer's angular coordinate. Then it appears that the velocities measured by observers at different radial coordinates, but at the same angle, are correlated over shorter radial-length scales than velocities measured by observers at the same radial coordinate, but different angles. This is understood to be due to the astro-dynamical influences of the Galactic features included by [Chakrabarty, 2007] in the simulation that generates the training data that we use here. This simulation incorporates the joint dynamical effect of the Galactic spiral arms and the elongated Galactic bar (made of stars) that rotate at different frequencies (as per the astronomical model

responsible for the generation of our training data), pivoted at the centre of the Galaxy. An effect of this joint handiwork of the bar and the spiral arms is to generate distinctive stellar velocity distributions at different radial (i.e. along the S_1 direction) coordinates, at the same angle (s_2). On the other hand, the stellar velocity distributions are more similar at different S_2 values, at the same s_1. This pattern is borne by [Chakrabarty, 2004], in which the radial and angular variation of the standard deviations of these bivariate velocity distributions are plotted. Then it is understandable why the correlation length scales are shorter along the S_1 direction, than along the S_2 direction. Furthermore, for the correlation parameter ρ, physics suggests that the correlation will be zero among the two components of a velocity vector. These two components are after all, the components of the velocity vector in a 2-dimensional orthogonal basis. However, the MCMC chain shows that there is a small (negative) correlation between the two components of the stellar velocity vector.

2.7.1 Predicting $s^{(test)}$

In one method, we perform posterior sampling using Metropolis-Hastings, from the joint posterior probability density of all parameters (GP parameters as well as solar location vector), given test+training data. In Figure 2.1 and Figure 2.2, we present respectively, traces and histogram-representations of marginal posterior probability densities of the solar location coordinates $s_1^{(test)}$, $s_2^{(test)}$; q_1 and q_2 that get updated once the test data is added to augment the training data, and parameters σ_{11}^1, σ_{22}^1 and ρ that are learnt given this augmented data. 95% HPD credible regions computed on each parameter in this inference scheme, are displayed in Table 2.1. These figures display these parameters in the non-nested GP model. When the nested-GP model is used, histogram-representations of the marginals of the aforementioned parameters, are displayed in Figure 2.3.

We notice that the values of the inverse correlation length are almost the same as the values with training data only.

Prediction of $\boldsymbol{s}^{(test)}$ using the nested GP models gives rise to very similar results as when the non-nested models are used, (see Figure 2.3 that compares the marginals of the solar location parameters sampled from the joint posterior density of all unknowns, given all data, in nested GP models, against those obtained when non-nested GP models are used).

The marginal distribution of $s_1^{(test)}$ indicates that the marginal is unimodal and converges well, with modes at about 2 in model units. The distribution of $s_2^{(test)}$ on the other hand is quite strongly skewed towards values of $s_2^{(test)} \lesssim 1$ radians, i.e. $s_2^{(test)} \lesssim 57$ degrees, though the probability mass in this marginal density falls sharply after about 0.4 radians, i.e. about 23 degrees. These values tally quite well with previous work [Chakrabarty et al., 2015]. In that earlier work, using the training data that we use in this work, (constructed using the astronomical model $sp3bar3\_18$ discussed by [Chakrabarty et al., 2015]), the marginal distribution of $s_1^{(test)}$ was learnt to be bimodal, with modes at about 1.85 and 2, in model units. The distribution of $s_2^{(test)}$ found by [Chakrabarty et al., 2015] is however more constricted, with a sharp mode at about 0.32 radians (i.e. about 20 degrees). We do notice a mode at about this value in our inference, but unlike in the results of [Chakrabarty et al., 2015], we do not find the probability mass declining to low values beyond about 15 degrees. One possible reason for this lack of compatibility could be that in [Chakrabarty et al., 2015], the matrix of velocities $\boldsymbol{V}$ was vectorised, so that the training data then resembled a matrix, rather than a 3-tensor as we know it to be. Such vectorisation could have led to some loss of correlation information, leading to the results of [Chakrabarty et al., 2015].

When we predict $\boldsymbol{s}^{(test)}$ using test+training data, at the (modal values of the) GP pa-

rameters that are learnt from the training data, we generate samples from the posterior predictive of $\boldsymbol{s}^{(test)}$ (Equation 2.12) using Metropolis-Hastings. The results are presented in Table 2.1.

2.7.2 Astronomical implications

The radial coordinate of the observer in the Milky Way, i.e. the solar radial location is dealt with in model units, but will need to be scaled to real galactic unit of distance, which is kilo parsec (kpc). Now, from independent astronomical work, the radial location of the Sun is set as 8 kpc [Binney and Merrifield, 1998]. Then our learnt value of $S_1^{(test)}$ is to be scaled to 8 kpc, which gives 1 model unit of length to be $\frac{8\text{kpc}}{\text{our estimate of } S_1^{(test)}}$. Our main interest in learning the solar location is to find the frequency Ω_{bar} with which the Galactic bar is rotating, pivoted at the galactic centre, loosely speaking. Here $\Omega_{bar} = \frac{v_0}{1 \text{ model unit of length}}$, where $v_0 = 220$ km/s (see [Chakrabarty, 2007] for details). The solar angular location being measured as the angular distance from the long-axis of the Galactic bar, our prediction for S_2 actually tells us the angular distance between the Sun-Galactic centre line and the long axis of the bar.

Table 2.2: 95% HPD on each Galactic feature parameter learnt from the solar location coordinates learnt using the two predictive inference schemes listed above and as reported in a past paper for the same training and test data.

	95% HPD for Ω_{bar} (km/s/kpc)	for angular distance of bar to Sun (degrees)
from posterior predictive	$[48.11, 57.73]$	$[4.53, 43.62]$
from joint posterior density	$[48.25, 57.244]$	$[2.25, 46.80]$
from Chakrabarty et. al (2015)	$[46.75, 62.98]$	$[17.60, 79.90]$

Table 2.2 displays the Galactic feature parameters that are derived from the learnt solar location parameters, under the different inference schemes using the non-nested model,

namely, sampling from the joint posterior probability of all parameters given all data, and from the posterior predictive of the solar location coordinates given all data and GP parameters already learnt from training data alone. The derived Galactic feature parameters are the bar rotational frequency Ω_{bar} in the real astronomical units of km/s/kpc and the angular distance between the bar and the Sun, in degrees. The table also includes results from Chakrabarty et. al (2015). As we see, the bar in the Milky Way galaxy is a relatively fast bar with its predicted rotational frequency Ω_{bar} that accommodates the suggested frequency of [54.1,60.2] in km/s/kpc advanced by [Chakrabarty, 2007] on the basis of numerical modelling of the MW disk. The low angular separation of Sun-Galactic-centre line from the long-axis of the bar, is also corroborated by that earlier work, and multiple astronomical references cited therein.

2.8 Model Checking

One way to check for the model and results, given the data at hand, is to generate data from the learnt model, and then compare this generated data with the observed data. Now, the model that we learn, is essentially the tensor-variate GP that is used to model the functional relationship $\boldsymbol{\xi}(\cdot)$ between the observable $\boldsymbol{V}$ and the input-space parameter $\boldsymbol{S}$. By, saying that we intend on generating new data, we imply the prediction of a new value of $\boldsymbol{V}$, given the learnt model of this GP.

This prediction of new datum on $\boldsymbol{V}$, is fundamentally different from the inverse prediction of the value $\boldsymbol{s}^{(test)}$ of the input-space parameter $\boldsymbol{S}$ that we have undertaken–as discussed above in Section 2.6–where the sought $\boldsymbol{s}^{(test)}$ is the value of $\boldsymbol{S}$ at which test data $\boldsymbol{v}^{(test)}$ on $\boldsymbol{V}$ is recorded. There is no closed-form solution to the posterior predictive of $\boldsymbol{s}^{(test)}$ given the test data and the learnt GP parameters.

In fact, at chosen values of $\boldsymbol{S}$–chosen to be the design points in the training data, for convenience–the covariance function $\boldsymbol{\Sigma}_3$ of this GP, (modelled as a GP with an estimated mean), is known, given the learnt values of the parameters of the kernel used to parametrise $\boldsymbol{\Sigma}_3$. However, in our Bayesian inference, we do not really learn a value of any parameter, but learn the marginal posterior density of each unknown parameter, given the data. Thus, in order to pin the value of the covariance matrix, we identify the parameter value corresponding to a selected summary of this posterior distribution. For example, we could choose to define $\boldsymbol{\Sigma}_3$ at pairs of known design points $\boldsymbol{s}_i$, $\boldsymbol{s}_j$, and the modal value of ℓ_c–identified from the marginal posterior density of ℓ_c inferred upon, given the data. Here $i, j \in \{1, \ldots, n = 216\}$. The resulting value of the ij-th element of $\boldsymbol{\Sigma}_3$ will then provide one summary, of the covariance between the 50×2 stellar velocity matrix $\boldsymbol{v}_i$ realised at $\boldsymbol{S} = \boldsymbol{s}_i$, and $\boldsymbol{v}_j$ realised at $\boldsymbol{S} = \boldsymbol{s}_j$. Similarly, the learnt modal values of the parameters $\sigma_{11}^{(1)}$, $\sigma_{22}^{(1)}$ and ρ define one summary of the covariance matrix $\boldsymbol{\Sigma}_1$ that informs on te covariance between the $2\,216 \times 50$-dimensional sheets of data on each component of the 2-imensional stellar velocity vector. Again, other summaries of the parameter values could be used as well, for example, the parameter value identified at the mean of the marginal posterior density of this parameter, as learnt given the training data, is also used.

In this model checking exercise, the unknowns are certain elements of the cuboidally-shaped data comprising the 216 number of 50×2-dimensional stellar velocity matrices generated by astronomical simulation, at chosen design points $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{216}$, i.e. the 3rd-order tensor $\mathbf{D}_V := \{\boldsymbol{v}_1, \vdots \boldsymbol{v}_2, \vdots \ldots, \vdots \boldsymbol{v}_{216}\}$. In the first attempt to model checking, we model all elements of the r-th such simulated stelar velocity matrix $\boldsymbol{v}_r$, (that is generated at the known design point $\boldsymbol{s}_r$), to be the $50 \times 2 = 100$ unknowns. We refer to these unknown elements of $\boldsymbol{v}_r$ as $v_{11}^{(r)}, v_{12}^{(r)}, v_{21}^{(r)}, \ldots, v_{50,2}^{(r)}$. The 3rd-ordered tensor without the r-th slice, is

referred to as $\mathbf{D}_V^{(-r)} := \{\boldsymbol{v}_1, \vdots \boldsymbol{v}_2, \vdots \ldots, \vdots \boldsymbol{v}_{r-1}, \vdots \boldsymbol{v}_{r+1}, \vdots \boldsymbol{v}_{216}\}$. The joint posterior probability density of the 100 unknowns, at the learnt modal values $r_1^{(mode)}, r_2^{(mode)}, \sigma_{11}^{(1,mode)}, \sigma_{22}^{(1,mode)}, \rho^{(mode)}$ is

$$\pi\left(v_{11}^{(r)}, v_{12}^{(r)}, v_{21}^{(r)}, \ldots, v_{50,2}^{(r)} | \mathbf{D}_V^{(-r)}\right) \propto \mathcal{TN}_{2\times 50\times 216}(\hat{\boldsymbol{M}}, \boldsymbol{\Sigma}_1^{(mode)}, \hat{\boldsymbol{\Sigma}}_2, \boldsymbol{\Sigma}_3^{(mode)}),$$

where,

–the 3rd-ordered tensor-valued data that enters the parametric form of the 3rd-ordered tensor-normal density on the RHS, has elements of its r-th slice, (of the total of 216 slices), unknown. All other elements of this $2 \times 50 \times 216$-dimensional tensor are known;
–uniform priors are used on the unknowns; –$\boldsymbol{\Sigma}_1^{(mode)}$ is the learnt modal value of the 2×2-dimensional covariance matrix $\boldsymbol{\Sigma}_1$ s.t. its $1,1$-th element is $\sigma_{11}^{(1,mode)}$, $2,2$-th element is $\sigma_{22}^{(1,mode)}$, $1,2$-th element is $\rho^{(mode)}\sqrt{\sigma_{22}^{(1,mode)}\sigma_{11}^{(1,mode)}}$, and the $2,1$-th element is equal to the $1,2$-th element (as this is a covariance matrix);
–$\boldsymbol{\Sigma}_3^{(mode)}$ is the learnt modal value of the 216×216-dimensional covariance matrix $\boldsymbol{\Sigma}_3$, s.t. its ij-th element is $\exp\left[-(\boldsymbol{s}_i - \boldsymbol{s}_j)^T \boldsymbol{Q}^{(mode)}(\boldsymbol{s}_i - \boldsymbol{s}_j)\right]$, with the non-zero elements of the diagonal 2×2-dimensional $\boldsymbol{Q}^{(mode)}$-matrix given by $r_1^{(mode)}$ and $r_2^{(mode)}$. $\boldsymbol{s}_i$ being the i-th design point, is known $\forall i, j = 1, \ldots, 216$.

To learn the 100 unknowns $v_{11}^{(r)}, v_{12}^{(r)}, v_{21}^{(r)}, \ldots, v_{50,2}^{(r)}$, we run a Random Walk (RW) Metropolis-Hastings chain, with the data defined as above, the known 216 number of design points, and all the learnt, modal parameter values. The joint posterior probability density of the unknowns that defines the acceptance ratio in this chain, is given as in the last equation. The chain is run for 20,000 iterations, for r=200, and the mean of the last 1000 samples of $v_{ij}^{(200)}$ is recorded, where $i = 1, \ldots, 50$, $j = 1, 2$. These sample means $\bar{v}_{ij}^{(200)}$ then constitute the learnt value of the 100 elements of the 200-th stellar ve-

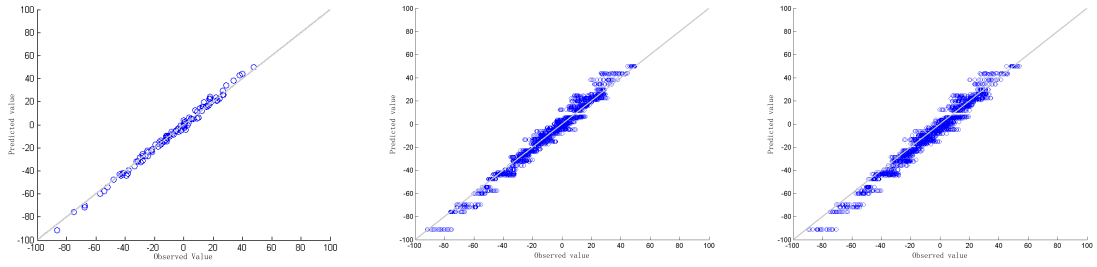

Figure 2.7: *Left:* Comparison of the observed and predicted values of elements of the r-th 50×2-dimensional stellar velocity matrix $\boldsymbol{v}_r$, where 216 such matrices constitute the training data $\mathbf{D}_V$ (on velocities of 50 stellar neighbours of the Sun) that is generated by astronomical simulations. The predicted or learnt values are obtained from a Random Walk (RW)-MCMC chain undertaken with the all elements of the 3rd-order tensor $\mathbf{D}_V$ known, except for the elements of its r-th slice, and the learnt values of the parameters of the GP used to model the data at hand, at a chosen summary, namely the mode, of the marginal posterior density of each such learnt GP parameter. Here r=200. Equality of the observed and predicted values of he elements of $\boldsymbol{v}_r$ is indicated by the point lying on the drawn straight line with unit slope; the predicted values are found to lie close to this line. *Middle:* Depicts a similar comparison, as displayed in the left panel, but for 20 distinct values of r, namely for $r = 190, 191, \ldots, 210$. *Right:* Depicts the same comparison of observed and predicted values of elements of 20 slices $\boldsymbol{v}_{190}, \ldots, \boldsymbol{v}_{210}$, but this time, the employed GP parameters are the means of their respective marginals. Thus, this model-checking exercise checks for the used models and results obtained (given the data at hand) at the mean of the respective posterior density.

.

locity matrix v_{200}. We plot the pairs of learnt value $\bar{v}_{ij}^{(200)}$ of elements of the v_{200} matrix, against the empirically observed value of this element, $\forall i = 1,\ldots,50$, $\forall j = 1,2$. The plot is presented in the left panel of Figure 2.7. Thus, each point on this plot is a pair (empirically observed value of $v_{ij}^{(200)}, \bar{v}_{ij}^{(200)}$), and there are $50 \times 2 = 100$ points in this plot. The points are found to lie around the straight line with slope 1. In other words, the values of the elements in the r-th (=200-th) slice of the training data that we learn using our model, are approximately equal to the empirically observed values of these elements. This is corroboration of our models and results.

We attempt a similar prediction of elements of the training data for other values of r, namely for $r = 190,\ldots,210$. The learnt values of elements of v_r, for each r, is plotted against the empirically observed elements of v_r. We have superimposed results for all 20 values of r in the same plot, resulting in the middle panel of Figure 2.7. Again, the values predicted for all 20 slices, are found to be close to the empirical observations, as betryaed by the points lying close to the straight line of unit slope.

Lastly, we wanted to ensure that the encouraging results from our model checking exercise are robust to changes in the posterior density summary of the learnt GP parameters. Thus, we switch to using the mean of the parameter marginal posterior from the posterior mode, and carry out the same exercise of predicting elements of slices $v_{190},\ldots,v_{210}$. Results are displayed in the right panel of Figure 2.7. Again, very encouraging corroboration of our used models and results (of learning the GP parameters) is noted. Indeed, in such model checking exercises, encouraging match between the predictions and the empirical observations lends confidence in the used models and results obtained therefrom, given the data at hand–such models and results are the inputs to this exercise. However, if lack of compatibility is noted in such a model checking exercise, between empirical observations

and predictions, then it implies that either the used modelling is wrong, and/or the results obtained there from given the data are wrong. However, the model checking exercise that we undertake, vindicates our models and results, given the data at hand.

2.9 Conclusion

In this chapter I have presented work a method for learning tensor-valued functional relations between two random variables, where at least one of the variables is high-dimensional, i.e. tensor-valued in general. This situation then renders the unknown (and sought) functional relation between the variables to be a tensor-valued function, if we express the model of the relationship between the variables to be s.t. it retains capability to predict either variable, given test data measured on the other. The sought function is treated as a random-valued function that is learnt probabilistically, i.e. a probability distribution is assigned to it, which is equivalent to stating that this function is treated as a random realisation from a stochastic process. The particular stochastic process that we choose to work with is a high-dimensional (tensor-variate) Gaussian Process, of corresponding dimensionalities, s.t. the joint probability density of a finite number of values of this function (at each of the design points in the trainning data), is the high-diemsnional equivalent of a multivariate Normal, namely a Tensor Normal density. In other words, the likelihood is Tensor Normal, and we use this in Bayes rule with chosen prior probability densities to write the joint posterior probability desity of the parameters of the Tensor Normal likelihood, given the training data. We sample from this posterior density using MCMC. The focus of my work has been hypercuboidally-shaped data, that is in general not continuous, thus demanding a non-stationary covariance structure of the invoked tensor-variate GP. Such a covariance structure is attained by generalising a stationary covariance to one in which the hyperpa-

rameters (correlation length scales along each direction in input space) are treated as dependent on the sample function of the invoked GP, i.e. as a dynamically varying, scalar-valued function that is modelled as a realisation from a scalar-variate GP with distinct covariance structure, that we parametrise. We employ Metropolis-within-Gibbs-based inference. Subsequent to the learning of the sought tensor-valued function, we make an inverse Bayesian prediction of the system parameter values at which test data on the observable is realised. Thus this work permits methodology for learning given discontinuous data.

Chapter 3

With-uncertainty Graphical Models and Inter-graph Distance

3.1 Introduction

Graphical models of a complex, highly multivariate dataset, manifest an intuitive illustration of the correlation structure of the data, and are of interest in different disciplines [Benner et al., 2014; Airoldi, 2007; Carvalho and West, 2007; Bandyopadhyay and Canale, 2016; Whittaker, 2008]. Much work has been undertaken to study the correlation of a rectangularly-shaped, multivariate dataset, by treating the vector-valued observable–multiple measurements of which comprise the data–as a realisation from a Gaussian Process (GP), rendering the likelihood of the unknown GP parameters given the data, a matrix-normal distribution [Wang and West, 2009; Ni et al., 2017; Gruber and West, 2016].

In this chapter, I discuss the simultaneous Bayesian inference on the partial correlation structure and graphical model of a multivariate dataset, learning each, along with well-defined uncertainties, namely, HPDs, while acknowledging possible errors of measurement. It is to this effect that we perform a Metropolis, by a 2-block-update version of MCMC [Robert and Casella, 2004], on the correlation matrix given the data, and on the graph given the updated correlation.

Objective and comprehensive uncertainties on the Bayesianly learnt graphical model of the given multivariate data, are sparsely available in the literature. Such uncertainties can potentially be very useful in informing us about the range of models that describe the partial correlation structure of the data at hand, where by a model, we refer to an identified set of nodes that are connected by edges, at learnt probabilities. Madigan and Raftery [1994] discuss a method for computing model uncertainties by averaging over a set of identified models, and they advance ways for computing the posterior probabilities, by taking advantage of the graphical structure, for two classes of considered models, namely, the recursive causal models [Kiiveri et al., 1984] and the decomposable loglinear models [Goodman, 1970]. This method allows them to select the "best models", while accounting for model uncertainty.

On the other hand, the method of simultaneous learning of correlation strcture and graphical model of a rectangularly-shaped data set that is introduced in this chapter, provides a simple and well-defined way of learning uncertainties of the graphical model of a given multivariate data.

At every update of the graphical structure of the data, the graph is updated; graphs thus learnt, if identified to lie within an identified range of values of the posterior probability values, comprise the uncertainty-included graphical model of the data. Thus, the method allows for acknowledgement of uncertainties in the learning of the graphical model of the data. In addition, this method permits incorporation of measurement errors into the learning of the graphical model, and permits fast learning of large networks (demonstrated on the learning of the human disease-symptom network, with $\geq$8000 nodes). The uncertainty learning mentioned above is empirically illustrated in Section 3.3, (along with model checking and effect of measurement error incorporation), on a small simulated dataset,

as well as on the learning of the graphical models of the real vino-chemical datasets of Portuguese red and white wine samples.

In fact, via this vino-chemical example, we illustrate an important contribution of our work, namely, the computing of inter-graph distances. Hoff et al. [2011]; Xu and Yan [2015]; Wang & Chakrabarty (*under preparation*), advance methods to learn the correlation in data that is high-dimensional in general, eg. a cuboidally-shaped dataset that comprises multiple measurements of a matrix-variate observable. In the general case, a k-th ordered tensor-variate observable is then modelled using a high-dimensional Gaussian Process, rendering the likelihood, $k+1$-variate Tensor Normal. The pioneering work by Wang and West [2009] allows for the learning of both the between-rows and between-columns covariance matrices of a rectangularly-shaped multivariate dataset, and therefore, of two graphical models for such data. Ni et al. [2017] extend this approach to high-dimensional data. However, a high-dimensional graph showing the correlation structure amongst the multiple components of a general hypercuboidally-shaped dataset, is not easy to visualise or interpret. Instead, in this paper, we treat the data to be built of correlated rectangularly-shaped slices, given each of which, the between-columns (partial) correlation structure and graphical model are Bayesianly learnt, along with uncertainties, subsequent to our closed-form marginalisation over all between-rows correlation matrices (unlike in the work of Wang and West [2009]). We then compute the Hellinger distance [Matusita, 1953; Banerjee et al., 2015] between the posterior probability densities of the pair of graphical models that are learnt given the respective pair of such rectangularly-shaped data slices. Such a distance will then tell us about the independence of the probability density functions that each such data slice is sampled from.

For example, as Guinness et al. [2014] state, the presence of spatial correlation amongst

sets of multivariate observations in a given dataset, cannot be correctly captured by "computing partial correlation coefficients and by specifying and fitting more complex graphical models". However, we suggest treating each such set of multivariate observation as a separate (rectangularly-shaped) dataset–where one such dataset is correlated with another–and learn the partial correlation structure and graphical model of each dataset, followed by computing the pairwise Hellinger distance between posterior probability densities of each learnt pair of graphs. The corresponding Hellinger affinity measure then bears information about the correlation between the original sets of multivariate observations. Indeed our method offers the inter-graph distance between two differently sized datasets, i.e. even when the two datasets contain different numbers of multivariate observations.

Our learnt graphical model of the given data, comprises a set of random inhomogeneous graphs [Frieze and Karonski, 2016] that lie within the credible regions that we define, where each such graph is a generalisation of a Binomial graph [Frieze and Karonski, 2016], in which the probability of existence of the edge between a given pair of vertices is dependent on the partial correlation of the components of the observable vector corresponding to these vertices, where the partial correlation matrix is itself computed using the correlation matrix that is updated given the data at hand, within a Metropolis with a dual-block updated-based Bayesian inference scheme [Robert and Casella, 2004].

In our used inference scheme, we do not make inference on the graph (writing its posterior) clique-by-clique, and neither are we reliant on the closed-form nature of the posteriors to sample from. In other words, we do not need to invoke conjugacy to affect our learning–either of the partial correlation structure of the data or of the graphical model. Often, in Bayesian learning of Gaussian undirected graphs, a Hyper-Inverse-Wishart prior is typically imposed on the covariance matrix of the data, as this then allows for a Hyper-Inverse-

Wishart posterior of the covariance, which in turn implies that the marginal posterior of any clique is Inverse-Wishart–a known, closed-form density [Dawid and Lauritzen, 1993; Lauritzen, 1996].

Inference is then rendered easier, than when posterior sampling from a non-closed form posterior needs to be undertaken, using numerical techniques such as MCMC. Now, if the graph is not decomposable, and a Hyper-Inverse-Wishart prior is placed on the covariance matrix, the resulting Hyper-Inverse-Wishart joint posterior density that can be factorised into a set of Inverse-Wishart densities, cannot be identified as the clique marginals. Expressed differently, the clique marginals are not closed-form when the graph is not decomposable. However, this is not a worry in our learning, i.e. we can undertake our learning irrespective of the validity of decomposability.

3.2 Learning correlation matrix and graphical model given data, using block-update Metropolis

Let $\boldsymbol{X} \in \mathcal{X} \subseteq \mathbb{R}^p$ be a p-dimensional observed vector, with $\boldsymbol{X} = (X_1, \ldots, X_p)^T$. Let there be n measurements of X_j, $j = 1, \ldots, p$, so that the $n \times p$-dimensional matrix $\mathbf{D} = [x_{ij}]_{i=1;j=1}^{n;p}$ is the data that comprises n measurements of the p-dimensional observable $\boldsymbol{X}$. Let the i-th realisation of $\boldsymbol{X}$ be $\boldsymbol{x}_i$, $i = 1, \ldots, n$.

We model $\boldsymbol{X}$ using a high-dimensional GP, so that the set of realisations of this variable that comprises the data $\mathbf{D}$, is jointly matrix-normal, i.e.

$$\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \sim \mathcal{MN}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_R, \boldsymbol{\Sigma}_C),$$

where this matrix-normal density is parametrised by an $n \times p$-dimensional mean matrix $\boldsymbol{\mu}$, an $n \times n$-dimensional covariance matrix $\boldsymbol{\Sigma}_R$, an element of which is the covariance

between a pair of rows in $\mathbf{D}$, and a $p \times p$-dimensional covariance matrix $\mathbf{\Sigma}_C$ that manifests information about between-columns covariance in data $\mathbf{D}$. This is synonymous to saying that the likelihood of $\boldsymbol{\mu}$, $\mathbf{\Sigma}_C$ and $\mathbf{\Sigma}_R$, given data $\mathbf{D}$, is matrix normal.

3.2.1 Learning the correlation structure in the data

We standardise the data $\mathbf{D}$ by the empirical mean and standard deviation, to then model this standardised data $\mathbf{D}_S$ using a high-dimensional GP with zero mean. Thus, the $n \times p$-dimensional matrix $\mathbf{D}_S = [z_{ij}]$, with $z_{ij} = \dfrac{x_{ij} - \bar{x}_j}{\mathrm{Y}_j}$, where $\bar{x}_j := \dfrac{\sum_{i=0}^{n} x_{ij}}{n}$ and $\mathrm{Y}_j^2 := \dfrac{\sum_{i=0}^{n} x_{ij}^2}{n} - \left(\dfrac{\sum_{i=0}^{n} x_{ij}}{n} \right)^2$. Then modelling the standardised observable $\boldsymbol{Z} = (Z_1, \ldots, Z_p)^T$ with a zero-mean high-dimensional GP, we get the joint probability distribution of the n values of $\boldsymbol{Z}$ that comprise $\mathbf{D}_S$ to be

$$\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\} \sim \mathcal{MN}(\mathbf{0}, \mathbf{\Sigma}_R^{(S)}, \mathbf{\Sigma}_C^{(S)}),$$

i.e. the likelihood of the covariance matrices $\mathbf{\Sigma}_R^{(S)}$ and $\mathbf{\Sigma}_C^{(S)}$, given data $\mathbf{D}_S$, is matrix-normal:

$$\ell(\mathbf{\Sigma}_R^{(S)}, \mathbf{\Sigma}_C^{(S)} | \mathbf{D}_S) = \frac{1}{(2\pi)^{\frac{np}{2}} |\mathbf{\Sigma}_C^{(S)}|^{\frac{p}{2}} |\mathbf{\Sigma}_R^{(S)}|^{\frac{n}{2}}} \times \exp\left[-\frac{1}{2} tr\left\{ (\mathbf{\Sigma}_R^{(S)})^{-1} \mathbf{D}_S (\mathbf{\Sigma}_C^{(S)})^{-1} (\mathbf{D}_S)^T \right\} \right], \tag{3.1}$$

Here $\mathbf{\Sigma}_R^{(S)}$ generates the covariance between the standardised variables $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_{i'}$, $i, i' = 1, \ldots, n$, (while $\mathbf{\Sigma}_R$ generates the covariance between $\boldsymbol{X}_i$ and $\boldsymbol{X}_{i'}$). In other words, $\mathbf{\Sigma}_R^{(S)}$ generates the correlation between rows of the standardised data set $\mathbf{D}_S$. Similarly, $\mathbf{\Sigma}_C^{(S)}$ generates the correlation between columns of $\mathbf{D}_S$.

Importantly, we use uniform prior on $\mathbf{\Sigma}_C^{(S)}$, and Jeffry's prior on $\mathbf{\Sigma}_R^{(S)}$: $\pi_0(\mathbf{\Sigma}_R^{(S)}) = \left|\mathbf{\Sigma}_R^{(S)}\right|^{\alpha}$, $\alpha = -\left(\frac{n}{2}+1\right)$.

Theorem 3.2.1. *The joint posterior probability density of the correlation matrices $\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)}$, given the standardised data $\mathbf{D}_S$ is*

$$\left[\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)} | \mathbf{D}_S\right] \propto \ell(\mathbf{\Sigma}_R^{(S)}, \mathbf{\Sigma}_C^{(S)} | \mathbf{D}_S) \left[\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)}\right],$$

where $\ell(\mathbf{\Sigma}_R^{(S)}, \mathbf{\Sigma}_C^{(S)} | \mathbf{D}_S)$ is the likelihood of $\mathbf{\Sigma}_R^{(S)}, \mathbf{\Sigma}_C^{(S)}$ given data $\mathbf{D}_S$. This can be marginalised over the $n \times n$-dimensional between-rows' correlation $\mathbf{\Sigma}_R^{(S)}$, to yield

$$[\mathbf{\Sigma}_C^{(S)} | \mathbf{D}_S] \propto \frac{1}{c\left(\mathbf{\Sigma}_C^{(S)}\right) \left|\mathbf{\Sigma}_C^{(S)}\right|^{p/2} \left|\mathbf{D}_S (\mathbf{\Sigma}_C^{(S)})^{-1} (\mathbf{D}_S)^T\right|^{\frac{n+1}{2}}},$$

where the prior on $\mathbf{\Sigma}_C^{(S)}$ is uniform; prior on $\mathbf{\Sigma}_R^{(S)}$ is the non-informative $\pi_0(\mathbf{\Sigma}_R^{(S)}) = \left|\mathbf{\Sigma}_R^{(S)}\right|^{\alpha}$, $\alpha = -\frac{n}{2} - 1$, and $\mathbf{\Sigma}_C^{(S)}$ is assumed invertible. Here, $c\left(\mathbf{\Sigma}_C^{(S)}\right)$ is a function of $\mathbf{\Sigma}_C^{(S)}$ that normalises the likelihood.

Proof. The joint posterior probability density of $\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)}$, given data $\mathbf{D}_S$:

$$\begin{aligned}
\left[\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)} | \mathbf{D}_S\right] &\propto \ell\left(\mathbf{\Sigma}_R^{(S)}, \mathbf{\Sigma}_C^{(S)} | \mathbf{D}_S\right) \left[\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)}\right], \quad \text{i.e.} \\
\left[\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)} | \mathbf{D}_S\right] &\propto \frac{1}{(2\pi)^{\frac{np}{2}} \left|\mathbf{\Sigma}_C^{(S)}\right|^{\frac{p}{2}} \left|\mathbf{\Sigma}_R^{(S)}\right|^{\frac{n}{2}}} \times \\
&\quad \exp\left[-\frac{1}{2} tr\left\{(\mathbf{\Sigma}_R^{(S)})^{-1} (\mathbf{D}_S) (\mathbf{\Sigma}_C^{(S)})^{-1} (\mathbf{D}_S)^T\right\}\right] \left|\mathbf{\Sigma}_R^{(S)}\right|^{-\frac{n}{2}-1},
\end{aligned} \tag{3.2}$$

using the likelihood from Equation 3.1; using prior on $\mathbf{\Sigma}_R^{(S)}$ to be $\pi_0(\mathbf{\Sigma}_R^{(S)}) = \left|\mathbf{\Sigma}_R^{(S)}\right|^{\alpha}$ where $\alpha = -\frac{n}{2} - 1$; using prior on $\mathbf{\Sigma}_C^{(S)}$ to be uniform.

Marginalising $\mathbf{\Sigma}_R^{(S)}$ out from the joint posterior $\left[\mathbf{\Sigma}_C^{(S)}, \mathbf{\Sigma}_R^{(S)} | \mathbf{D}_S\right]$, we get:

$$\left[\mathbf{\Sigma}_C^{(S)} | \mathbf{D}_S\right] \propto$$

$$\frac{1}{\left|\mathbf{\Sigma}_C^{(S)}\right|^{\frac{p}{2}}} \times \int_{\mathcal{R}} \frac{1}{\left|\mathbf{\Sigma}_R^{(S)}\right|^{\frac{n}{2}}} \left|\mathbf{\Sigma}_R^{(S)}\right|^{-\frac{n}{2}-1} \times \exp\left[-\frac{1}{2} tr\left\{(\mathbf{\Sigma}_R^{(S)})^{-1} \mathbf{D}_S (\mathbf{\Sigma}_C^{(S)})^{-1} (\mathbf{D}_S)^T\right\}\right] d(\mathbf{\Sigma}_R^{(S)}) \tag{3.3}$$

Here $\mathbf{\Sigma}_R^{(S)} \in \mathcal{R} \subseteq \mathbb{R}^{(n \times n)}$. Now,

– let $\boldsymbol{\Upsilon} := (\boldsymbol{\Sigma}_R^{(S)})^{-1}$. Then $d(\boldsymbol{\Sigma}_R^{(S)}) = |\boldsymbol{\Upsilon}|^{-(n+1)} d\boldsymbol{\Upsilon}$ [Mathai and G.Pederzoli, 1997],

– let $\boldsymbol{V}^{-1} := \mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T$, $\Longrightarrow tr\left[(\boldsymbol{\Sigma}_R^{(S)})^{-1}\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right] \equiv tr\left[\boldsymbol{V}^{-1}\boldsymbol{\Upsilon}\right]$ (using commutativeness of trace),

so that in Equation 3.3, we get

$$\left[\boldsymbol{\Sigma}_C^{(S)}|\mathbf{D}_S\right] \propto \frac{1}{\left|\boldsymbol{\Sigma}_C^{(S)}\right|^{\frac{p}{2}}} \int_{\mathcal{R}} |\boldsymbol{\Upsilon}|^{\frac{n}{2}} |\boldsymbol{\Upsilon}|^{\frac{n}{2}+1} \times \exp\left[-\frac{1}{2} tr\left\{\boldsymbol{V}^{-1}\boldsymbol{\Upsilon}\right\}\right] |\boldsymbol{\Upsilon}|^{-(n+1)} d\boldsymbol{\Upsilon}. \tag{3.4}$$

The integral in the RHS of Equation 3.4 represents the unnormalised Wishart pdf $W_n(\boldsymbol{V}, q)$, over all values of the random matrix $\boldsymbol{\Upsilon}$, where the scale matrix and degrees of freedom of this pdf are $\boldsymbol{V}$ and $q = n+1$ respectively, i.e. $q > n-1$.
Thus, integral in the RHS of Equation 3.4 is the integral of the unnormalised pdf of $\boldsymbol{\Upsilon} \sim W_n(\boldsymbol{V}, q)$, over the full support of $\boldsymbol{\Upsilon}$ $\left(\equiv \left(\boldsymbol{\Sigma}_R^{(S)}\right)^{-1}\right)$,
i.e. the integral in the RHS of Equation 3.4 is the normalisation of this pdf:

$$2^{\frac{qn}{2}} \Gamma_n\left(\frac{q}{2}\right) |\boldsymbol{V}|^{\frac{q}{2}} \equiv$$

$$2^{\frac{(n+1)(n)}{2}} \Gamma_n\left(\frac{n+1}{2}\right) \left| \left(\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right)^{-1} \right|^{\frac{n+1}{2}},$$

i.e. integral on RHS of Equation 3.4 is proportional to $\left| \left(\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right)^{-1} \right|^{\frac{n+1}{2}}$, i.e.

$$\left[\boldsymbol{\Sigma}_C^{(S)}|\mathbf{D}_S\right] \propto \frac{1}{\left|\boldsymbol{\Sigma}_C^{(S)}\right|^{\frac{p}{2}}} \left| \left(\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right)^{-1} \right|^{\frac{n+1}{2}} \tag{3.5}$$

Now, if $\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T$ is invertible, $\left| \left(\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right)^{-1} \right|^{\cdot} = \left|\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right|^{-\cdot}$.

– It is given that $\boldsymbol{\Sigma}_C^{(S)}$ is invertible, i.e. $\left(\boldsymbol{\Sigma}_C^{(S)}\right)^{-1}$ exists.

– The original dataset is examined to discard rows that are linear transformations of each other, leading to data matrix $\mathbf{D}_S$, no two rows of which are linear transformations of each other

$\Longrightarrow \mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T$ is positive definite, i.e. $\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T$ is invertible,
$\Longrightarrow \left| \left(\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right)^{-1} \right|^{(n+1)/2} = \left|\mathbf{D}_S(\boldsymbol{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right|^{-(n+1)/2}$.

Using this in Equation 3.5:

$$\left[\mathbf{\Sigma}_C^{(S)}|\mathbf{D}_S\right] \propto \left|\mathbf{\Sigma}_C^{(S)}\right|^{-p/2}\left|\mathbf{D}_S(\mathbf{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right|^{-(n+1)/2}. \tag{3.6}$$

This posterior of the between-columns correlation matrix $\mathbf{\Sigma}_C^{(S)}$ given data $\mathbf{D}_S$, is normalised over all possible datasets, where the possible datasets abide by a column-correlation matrix of $\mathbf{\Sigma}_C^{(S)}$, as:

$$c\left(\mathbf{\Sigma}_C^{(S)}\right) = \int_{\mathcal{Z}} \dots \int_{\mathcal{Z}} \frac{1}{\left|\left(\mathbf{D}^{/}(\mathbf{\Sigma}_C^{(S)})^{-1}(\mathbf{D}^{/})^T\right)\right|^{\frac{n^{/}+1}{2}}} dz_{11}^{/} dz_{11}^{/} \dots dz_{n^{/}p}^{/}, \tag{3.7}$$

where $\mathbf{D}^{/} = [z_{ij}^{/}]_{i=1;j=1}^{i=n^{/};j=p}$ is a dataset with $n^{/}$ rows and p columns, comprising values of random standardised variables $Z_{ij}^{/} \in \mathcal{Z}$, simulated to bear between-column correlation matrix of $\mathbf{\Sigma}_C^{(S)}$, s.t. $\mathbf{D}^{/}(\mathbf{\Sigma}_C^{(S)})^{-1}(\mathbf{D}^{/})^T$ is positive definite $\forall \mathbf{D}^{/} \in \mathcal{D}$. Choosing the same number of rows for all choices of the random data matrix $\mathbf{D}^{/}$, i.e. for a constant $n^{/}$, $\mathcal{D} \subseteq \mathbb{R}^{(n^{/}\times p)}$. Then $c\left(\mathbf{\Sigma}_C^{(S)}\right)$ is a positive definite function of $\mathbf{\Sigma}_C^{(S)}$.

Using this normalisation on the posterior of $\mathbf{\Sigma}_C^{(S)}$ given $\mathbf{D}_S$, in Equation 3.6 we get

$$\pi\left(\mathbf{\Sigma}_C^{(S)}|\mathbf{D}_S\right) = \frac{1}{c\left(\mathbf{\Sigma}_C^{(S)}\right)\left|\mathbf{\Sigma}_C^{(S)}\right|^{\frac{p}{2}}} \frac{1}{\left|\left(\mathbf{D}_S(\mathbf{\Sigma}_C^{(S)})^{-1}(\mathbf{D}_S)^T\right)\right|^{\frac{n+1}{2}}}, \tag{3.8}$$

where $c\left(\mathbf{\Sigma}_C^{(S)}\right) > 0$ is defined in Equation 3.7. □

The posterior $\left[\mathbf{\Sigma}_C^{(S)}|\mathbf{D}_S\right]$ as given by Theorem 3.2.1, suggests that we have to compute the normalisation $c\left(\mathbf{\Sigma}_C^{(S)}\right)$, in every iteration, i.e. for every updated value of $\mathbf{\Sigma}_C^{(S)}$. We recall that this normalisation is given by Equation 3.7, where the standardised $Z_{ij}^{/}$, $i = 1,\dots,n^{/};\ j = 1,\dots,p$ are simulated s.t. the column correlation between $(Z_{1m}^{/},\dots Z_{n^{/}m}^{/})^T$ and $(Z_{1q}^{/},\dots Z_{n^{/}q}^{/})^T$ is s_{mq}, with $\mathbf{\Sigma}_C^{(S)} = [S_{mq}]_{m=1;q=1}^{m=p;q=p}$. However, it is hard to compute $c\left(\mathbf{\Sigma}_C^{(S)}\right)$, defined in Equation 3.7, as a closed-form integral. Instead, we define an estimator $\hat{c}_t$ of the integral representing the all-data averaged normalisation, in the t-th iteration of the N-iteration long MCMC chain that we run, $(t = 0,\dots,N)$.

So, we estimate the integral in the RHS of Equation 3.7 using its unbiased estimator. We rephrase the integrand as $h(Z^{/}_{11}, \ldots, Z^{/}_{n^{/}p}, \mathbf{\Sigma}_C^{(S)})$, i.e.

$$h(Z^{/}_{11}, \ldots, Z^{/}_{n^{/}p}, \mathbf{\Sigma}_C^{(S)}) := \frac{1}{\left| \left(\mathbf{D}^{/} (\mathbf{\Sigma}_C^{(S)})^{-1} (\mathbf{D}^{/})^T \right) \right|^{\frac{n^{/}+1}{2}}}.$$

Then the estimator of the normalisation in Equation 3.7 is

$$\hat{c}\left(\mathbf{\Sigma}_C^{(S)}\right) = \mathbb{E}_{Z^{/}_{n^{/}p}}\left[\ldots\left[\mathbb{E}_{Z^{/}_{11}}\left[h(Z^{/}_{11}, \ldots, Z^{/}_{n^{/}p}, \mathbf{\Sigma}_C^{(S)})\right]\right]\ldots\right].$$

However, it is difficult to implement this estimator by sequentially computing expectations w.r.t. distribution of each of the elements of $\mathbf{D}^{/}$. Instead, we compute the expectation w.r.t. the block $\mathbf{D}^{/}$ of these elements, where $\mathbf{D}^{/}$ abides by a column-correlation of $\mathbf{\Sigma}_C^{(S)}$, i.e., we compute

$$\hat{c}^{/}\left(\mathbf{\Sigma}_C^{(S)}\right) = \mathbb{E}_{\mathbf{D}^{/}_S}\left[h(Z^{/}_{11}, \ldots, Z^{/}_{np}, \mathbf{\Sigma}_C^{(S)})\right].$$

During the t-th iteration of the MCMC chain, let the column correlation matrix be $\mathbf{\Sigma}_t$ and the normalisation be $\hat{c}_t$. Then $\hat{c}_t$ is defined using $\mathbf{\Sigma}_t$ and the sample of $n^{/} \times p$-dimensional data sets $\{\mathbf{D}^{t/}_1, \ldots, \mathbf{D}^{t/}_K\}$, s.t. $\mathbf{D}^{t/}_k (\mathbf{\Sigma}_t)^{-1} (\mathbf{D}^{t/}_k)^T$ is positive definite $\forall k = 1, \ldots, K$, at each t. Then:

$$\hat{c}_t := \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\left| \left(\mathbf{D}^{t/}_k (\mathbf{\Sigma}_t)^{-1} (\mathbf{D}^{t/}_k)^T \right) \right|^{\frac{n^{/}+1}{2}}}. \tag{3.9}$$

3.2.2 Learning the graphical model

Bayesian learning of the inhomogeneous, Generalised Binomial random graph, is undertaken, given the learnt $p \times p$-dimensional, between-columns correlation matrix $\mathbf{\Sigma}_C^{(S)}$, of the multivariate data set $\mathbf{D}_S := (\boldsymbol{Z}_1, \vdots, \ldots, \vdots, \boldsymbol{Z}_p)^T$, that results from the standardisation of the available data $\mathbf{D} := (\boldsymbol{X}_1, \vdots, \ldots, \vdots, \boldsymbol{X}_p)^T$.

The graph $\mathbb{G}(p, \boldsymbol{R})$ is a random variable in this learning, and it gets updated in every iteration of the used Bayesian inference scheme (Metropolis with 2-blocks update), at the updated (partial) correlation matrix given the data. Here, the graph $\mathbb{G}(p, \boldsymbol{R})$, has the vertex set $\boldsymbol{V}$ and the between-columns partial correlation matrix of data $\mathbf{D}_S$ is $\boldsymbol{R} = [R_{ij}]$, s.t. R_{ij} takes the value ρ_{ij}, $i \neq j$, and $\rho_{ii} = 1$. The vertex set is $\boldsymbol{V} = \{1, \ldots, p\}$ s.t. vertices $i, j \in \boldsymbol{V}$, $i \neq j$, are joined by the edge G_{ij} that is a random binary variable taking values of g_{ij}, where g_{ij} is either 1 or 0, and is the ij-th element of the edge matrix $\boldsymbol{G} = [G_{ij}]$.

Given a learnt value of the between-columns correlation matrix $\mathbf{\Sigma}_C^{(S)}$, to compute the value ρ_{ij} of the partial correlation variable R_{ij}, we first invert $\mathbf{\Sigma}_C^{(S)}$ to yield: $\mathbf{\Psi} := \left(\mathbf{\Sigma}_C^{(S)}\right)^{-1}$; $\mathbf{\Psi} = [\psi_{ij}]$, s.t.

$$R_{ij} = -\frac{\psi_{ij}}{\sqrt{\psi_{ii}\psi_{jj}}}, \quad i \neq j, \tag{3.10}$$

and $\rho_{ii} = 1$ for $i = j$.

The posterior probability density of the graph $\mathbb{G}(p, \boldsymbol{R})$ defined for the edge matrix $\boldsymbol{G}$, is given as

$$\pi(G_{12}, \ldots G_{p\,p-1}|\boldsymbol{R}) \propto \ell(G_{12}, \ldots G_{p\,p-1}|\boldsymbol{R})\ \pi_0(G_{11}, G_{12}, \ldots G_{p\,p-1}),$$

where $\pi_0(G_{12}, \ldots G_{p\,p-1})$ is the prior probability density on the edge parameters $\{G_{ij}\}_{i\neq j; i,j=1}^{p}$, and $\ell(G_{12}, \ldots, G_{1p}, G_{21}, G_{23}, \ldots, G_{p\,p-1}|\boldsymbol{R})$ is the likelihood of the edge parameters, given the partial correlation matrix $\boldsymbol{R}$ (that is itself computed using the between-columns correlation matrix $\mathbf{\Sigma}_C^{(S)}$, learnt given $\mathbf{D}_S$, (see Equation 3.8).

The prior on G_{ij} is set as $Bernoulli(0.5)$, i.e. $\pi_0(G_{12}, \ldots G_{p\,p-1}) = \prod_{i,j=1; i\neq j}^{p} 0.5^{g_{ij}} 0.5^{1-g_{ij}}$; thus, the prior is independent of the edge parameters.

The likelihood of any edge parameter given the corresponding partial correlation, is

defined to be a Normal density. Thus, likelihood of the edge parameters given $\boldsymbol{R}$ is

$$\prod_{i\neq j;i,j=1}^{p} \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left[-\frac{\left(G_{ij}-R_{ij}\right)^2}{2\sigma_{ij}^2}\right],$$

where the variance parameters $\{\sigma_{ij}\}_{i\neq j;i,j=1}^{p}$ are indeed hyperparameters that are also learnt from the data; these variance parameters have uniform prior probabilities imposed on them.

In light of the fact that parameters that are learnt, are the edge parameters as well as the variance parameters, the likelihood is rephrased as:

$$\ell(G_{12},\ldots,G_{1p},G_{23},\ldots,G_{2p},G_{34},\ldots,G_{p\,p-1},\sigma_{12},\ldots,\sigma_{1p},\sigma_{21},\sigma_{23},\ldots,\sigma_{p\,p-1},|\boldsymbol{R}) = \prod_{i\neq j;i,j=1}^{p} \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left[-\frac{\left(G_{ij}-R_{ij}\right)^2}{2\sigma_{ij}^2}\right]. \tag{3.11}$$

3.2.3 Inference using Metropolis-with 2-block update

The p-dimensional vector-valued observable (n realisations of which together comprise the standardised data $\mathbf{D}_S$), is modelled using a Gaussian Process, s.t. the joint probability distribution of the n realisations of this observable is Matrix-Normal with zero mean; while the between-row matrix can be marginalised out from this likelihood, the between-columns, $p \times p$-dimensional correlation matrix $\boldsymbol{\Sigma}_C^{(S)}$ is learnt given data $\mathbf{D}_S$. Upon learning $\boldsymbol{\Sigma}_C^{(S)}$–using which the partial correlation matrix $\boldsymbol{R}$ is computed–the graphical model comprising the credible-region defining set of random Binomial graphs $\{\mathbb{G}(p,\boldsymbol{R})\}$ is learnt, where the vertex set of each graph in this set is fixed as $\boldsymbol{V}$; the "credible region" in question is defined below in Section 3.2.5.

Metropolis-with 2-block update-based inference is carried out on the unknowns. Then, at the beginning of any iteration, $\boldsymbol{\Sigma}_C^{(S)}$ is updated given $\mathbf{D}_S$, following which, $\mathbb{G}(p,\boldsymbol{R})$ is updated at the newly updated $\boldsymbol{\Sigma}_C^{(S)}$.

To be precise, updating $\mathbf{\Sigma}_C^{(S)}$ implies updating the $\frac{p^2-p}{2}$ non-diagonal terms of the upper (or lower) triangle of the symmetric $p \times p$ correlation matrix $\mathbf{\Sigma}_C^{(S)}$, i.e. the parameters $S_{12}, S_{13}, \ldots, S_{1p}, S_{23}, \ldots, S_{2p}, S_{34}, \ldots, S_{p-1\,p}$. Given the nature of the likelihood of the S_{ij} parameters, $i < j$; $i, j = 1, \ldots, p$, (see Equation 3.8), this updating involves inversion of $\mathbf{\Sigma}_C^{(S)}$, and computing of the determinants of $\mathbf{\Sigma}_C^{(S)}$ and $\mathbf{D}_S \left(\mathbf{\Sigma}_C^{(S)}\right)^{-1} (\mathbf{D}_S)^T$. The inversion and determinant computation will need to be undertaken in every iteration. Such is possible with the computation of the square root of the column correlation matrix $\mathbf{\Sigma}_C^{(S)}$, as well as the factorisation of $\mathbf{D}_S \left(\mathbf{\Sigma}_C^{(S)}\right)^{-1} (\mathbf{D}_S)^T$ into two triangular matrices, since determinant of a triangular matrix is easily computed, as the product of the diagonal elements of such a matrix. Thus, in any iteration, we can compute the $p \times p$-dimensional, (lower by choice) triangular matrix, that is the square root $\boldsymbol{L}_C^{(S)}$ of $\mathbf{\Sigma}_C^{(S)}$, i.e. $\mathbf{\Sigma}_C^{(S)} = \boldsymbol{L}_C^{(S)} (\boldsymbol{L}_C^{(S)})^T$. Here $\boldsymbol{L}_C^{(S)}$ is computed via Cholesky decomposition. Cholesky decomposition of $\mathbf{D}_S \left(\mathbf{\Sigma}_C^{(S)}\right)^{-1} (\mathbf{D}_S)^T$ into the (lower) triangular matrix $\boldsymbol{L}$ and $\boldsymbol{L}^T$ is also undertaken, following the inversion of $\mathbf{\Sigma}_C^{(S)}$ into $(\mathbf{\Sigma}_C^{(S)})^{-1}$. Since the factors of $\mathbf{\Sigma}_C^{(S)}$ are already computed (Cholesky decomposed) as $\boldsymbol{L}_C^{(S)}$ and $(\boldsymbol{L}_C^{(S)})^T$, the inversion of $\mathbf{\Sigma}_C^{(S)}$ is undertaken using a forward substitution algorithm, i.e. we set $(\mathbf{\Sigma}_C^{(S)})^{-1} = \left((\boldsymbol{L}_C^{(S)})^T\right)^{-1} (\boldsymbol{L}_C^{(S)})^{-1}$, where $(\boldsymbol{L}_C^{(S)})^{-1}$ is computed using $\boldsymbol{L}_C^{(S)}$ in a forward substitution scheme; here $(\boldsymbol{L}_C^{(S)})^{-1} \boldsymbol{L}_C^{(S)} = \boldsymbol{I}$. Then $\mathbf{D}_S \left((\boldsymbol{L}_C^{(S)})^T\right)^{-1} (\boldsymbol{L}_C^{(S)})^{-1} (\mathbf{D}_S)^T$ is Cholesky decomposed into $\boldsymbol{L}$ and $\boldsymbol{L}^T$. The underlying schemes for forward substitution and Cholesky Decomposition are discussed in Section 3.2.4.

It is assumed that at every iteration, the proposed $\mathbf{\Sigma}_C^{(S)}$ and $\mathbf{D}_S \left(\mathbf{\Sigma}_C^{(S)}\right)^{-1} (\mathbf{D}_S)^T$ are positive definite–to implement which, we start by identifying rows in the original data set that are linearly dependent; for each set of rows that are identified as linearly dependent,

only one row is retained and the rest discarded from the original data set. While this is undertaken prior to the initiation of the MCMC chain, it is still possible that during an iteration, a proposed $\mathbf{\Sigma}_C^{(S)}$ is s.t. its square root $\boldsymbol{L}_C^{(S)}$ is not positive definite, within some pre-set numerical threshold. One possible solution to this problem is numerical, for example, implementing ridge adjustment [Wothke, 1993], i.e. adding a "small" number to the diagonal elements of $\boldsymbol{L}_C^{(S)}$, where "small" is typically $\lesssim 10^{-14}$times a diagonal element of the matrix in our implementation. Another, less *ad hoc* solution is to propose $\boldsymbol{L}_C^{(S)}$ in any iteration–instead of $\mathbf{\Sigma}_C^{(S)}$–where the diagonal elements of $\boldsymbol{L}_C^{(S)}$ are proposed as positive definite, while ensuring that the $\mathbf{\Sigma}_C^{(S)}$ generated as $\boldsymbol{L}_C^{(S)}(\boldsymbol{L}_C^{(S)})^T$ abides by the constraints of a correlation matrix, i.e. the diagonal elements are 1 and non-diagonal elements are $\in [-1,1]$. Adhering to such constraints is as difficult a numerical challenge as the original one. Given this, we opt to propose $\mathbf{\Sigma}_C^{(S)}$ in every iteration and perform its Cholesky decomposition, while implementing the ridge adjustment discussed above.

As we saw in Section 3.2, the joint posterior probability density of the elements S_{ij} of the $(p^2-p)/2$ upper/lower triangle of the correlation matrix, given data $\mathbf{D}_S$, is normalised, with the normalisation factor that needs updating at each update of $\Sigma_C^{(S)}$; $i < j$; $i,j = 1,\ldots,p$. This normalisation factor is $c\left(\mathbf{\Sigma}_C^{(S)}\right)$ defined in Equation 3.7. An estimator $\hat{c}_t$ of this normalisation in the t-th iteration is given in Equation 3.9–as the arithmetic mean of K number of random realisations of the integrand in the integral representing the normalisation $c\left(\mathbf{\Sigma}_C^{(S)}\right)$. Each such realisation results from a sampled data set $\mathbf{D}_k^{t/}$, that bears the column correlation matrix that is the column correlation updated in this t-th iteration of the MCMC chain; $t = 0,\ldots,N$, $k = 1,\ldots,K$. Here, each of the K sampled data sets is generated with $n^/$ rows (and the p columns, as $\mathbf{\Sigma}_C^{(S)}$ is $p \times p$-dimensional). To generate $\mathbf{D}_k^{t/}$ as a randomly sampled $n^/ \times p$-sized data set with column correlation as in the

t-th iteration, we use established sampling techniques for sampling data given a correlation structure.

The algorithm followed for our Metropolis-with 2-block-update-based inference is the following.

1(i) At the 0-th iteration, a seed value $\mathbf{\Sigma}_0$ of the correlation matrix $\mathbf{\Sigma}_C^{(S)}$ is chosen. To be precise, we choose a seed value $s_{ij}^{(0)}$ of the parameter S_{ij}, $i < j$; $i = 1, \ldots, p$. At this iteration–as at every iteration–Cholesky decomposition of the column correlation matrix is undertaken. Thus, in the 0-th iteration, we compute the $p \times p$-dimensional, lower (by choice) triangular matrix that is the square root $\boldsymbol{L}_0$ of $\mathbf{\Sigma}_0$, i.e. $\mathbf{\Sigma}_0 = \boldsymbol{L}_0 \boldsymbol{L}_0^T$. The Cholesky decomposition of $\mathbf{\Sigma}_0$ is performed as delineated in Section 3.2.4. This allows for $|\mathbf{\Sigma}_0|$ to be computed as square of the product of the diagonal elements of $\boldsymbol{L}_0$. Using $\boldsymbol{L}_0$, its inverse $\boldsymbol{L}_0^{-1}$ is computed using the forward substitution algorithm delineated in Section 3.2.4, and the inverse $\mathbf{\Sigma}_0^{-1}$ of $\mathbf{\Sigma}_0$ is then $\mathbf{\Sigma}_0^{-1} = (\boldsymbol{L}_0^{-1})^T \boldsymbol{L}_0^{-1}$. Next, $\mathbf{D}_S \mathbf{\Sigma}_0^{-1} (\mathbf{D}_S)^T$ is Cholesky decomposed, to allow for the determinant of this matrix to be computed. Using the computed precision matrix $\mathbf{\Sigma}_0^{-1}$ in the 0-th iteration, the partial correlation matrix $\boldsymbol{R}_0$ in this iteration is computed, i.e. $\rho_{ij}^{(0)}$ is computed $\forall i \neq j$; $i, j = 1, \ldots, p$, where $\boldsymbol{R}_0$ takes the value $[\rho_{ij}^{(0)}]$. For this seed correlation $\mathbf{\Sigma}_0$, the estimator of the normalisation factor $\hat{c}_0$ is computed using Equation 3.9, following the random selection of K number of $n^{/} \times p$-dimensional data sets $\mathbf{D}_k^{0/}$ with column correlation $\mathbf{\Sigma}_0$, $k = 1, \ldots, K$. We use $n^{/} = n$. The joint posterior probability density $\pi(s_{12}^{(0)}, \ldots, s_{p-1\,p}^{(0)} | \mathbf{D}_S)$ of the non-diagonal, upper triangle elements of $\mathbf{\Sigma}_0$, is computed, given data $\mathbf{D}_S$, using Equation 3.8.

1(ii) In the 0-th iteration, seed values ($g_{ij}^{(0)} = 1$) of the edge parameter G_{ij}, are assigned.

In addition, seed values $\sigma_{ij}^{(0)}$ are assigned to the σ_{ij} parameter; $i < j;\ i, j \in \boldsymbol{V}$. The joint posterior probability density of $g_{ij}^{(0)}$ and $\sigma_{ij}^{(0)}$, $\forall i, j = 1, \ldots, p;\ i < j$, is given as the product of the likelihood in Equation 3.11, the uniform prior on each variance parameter and the Bernoulli prior with rate parameter 0.5 on each edge parameter.

2(i) As the t-th iteration begins, $t = 1, \ldots, N$, a value $s_{ij}^{(t*)}$ for the ij-th element of the correlation matrix $\boldsymbol{\Sigma}_C^{(S)}$ is proposed, such that the proposed value of this matrix is $\boldsymbol{\Sigma}_t^{\star}$ in the t-th iteration. This is done by proposing $s_{ij}^{(t*)}$ from a Truncated Normal density that is left truncated at -1 and right truncated at 1, i.e.

$$s_{ij}^{(t*)} \sim \mathcal{TN}(s_{ij}^{(t*)}; s_{ij}^{(t-1)}, v_{ij}, -1, 1), \quad , \forall\, i, j = 1, \ldots, p;\ i \neq j,$$

where v_{ij} is the experimentally chosen variance of the Truncated Normal proposal density, and the mean of this density is the value $s_{ij}^{(t-1)}$ that is the current value of the parameter S_{ij} at the end of the $t-1$-th iteration. Here the current correlation matrix is $\boldsymbol{\Sigma}_{t-1}$, and $v_0 = v_{ij} \forall\, i, j$ by choice. We refer to the estimator of the normalisation at the proposed $\boldsymbol{\Sigma}_t^{\star}$ as $\hat{c}_t^{\star}$, and compute it using the generated sample $\{\mathbf{D}_1^{t/}, \ldots, \mathbf{D}_K^{t/}\}$ in Equation 3.9. The current normalisation $\hat{c}_{t-1}$ at the current correlation matrix $\boldsymbol{\Sigma}_{t-1}$ is computed using the generated sample $\{\mathbf{D}_1^{t-1/}, \ldots, \mathbf{D}_K^{t-1/}\}$ in Equation 3.9. Then we accept the proposed $\boldsymbol{\Sigma}_t^{\star}$, at a probability of

$$a(\boldsymbol{\Sigma}_t^{\star}, \boldsymbol{\Sigma}_{t-1}) := \min\left(1, \frac{\hat{c}_{t-1} \times \pi\left(\boldsymbol{\Sigma}_t^{\star} | \mathbf{D}_S\right)}{\hat{c}_t^{\star} \times \pi\left(\boldsymbol{\Sigma}_{t-1} | \mathbf{D}_S\right)} \frac{\mathcal{TN}(\boldsymbol{\Sigma}_{t-1}; \boldsymbol{\Sigma}_t^{\star}, v_0, -1, 1)}{\mathcal{TN}(\boldsymbol{\Sigma}_t^{\star}; \boldsymbol{\Sigma}_{t-1}, v_0, -1, 1)}\right),$$

where $\pi(\cdot|\mathbf{D}_s)$ is given in Equation 3.6, as $\propto \left|\boldsymbol{\Sigma}_C^{(S)}\right|^{-p/2} \left|\mathbf{D}_S (\boldsymbol{\Sigma}_C^{(S)})^{-1} (\mathbf{D}_S)^T\right|^{-(n+1)/2}$. If $a(\boldsymbol{\Sigma}_t^{\star}, \boldsymbol{\Sigma}_{t-1}) > u$, where $u \sim \mathcal{U}[0, 1]$, the correlation matrix is updated to $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_t^{\star}$; else $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{t-1}$. We then invert the correlation matrix that is current in the t-th iteration, and employ $\boldsymbol{\Sigma}_t^{-1}$ to compute the current value of the partial correlation

matrix $\boldsymbol{R}_t = [\rho_{ij}^{(t)}]$ (see Equation 3.10).

2(ii) At the 2nd stage of the t-th iteration, the graph variable $\mathbb{G}(p, \boldsymbol{R})$ is updated, given the current partial correlation matrix $\boldsymbol{R}_t$. To be precise, we propose $g_{ij}^{(t\star)} \sim Bernoulli(g_{ij}^{(t\star)}; \rho_{ij}^{(t)})$ pmf, $\forall\, i, j = 1, \ldots, p; i < j$. Also, for each i, j, we propose $\sigma_{ij}^{(t\star)}$ from a Normal density $\mathcal{N}(\sigma_{ij}^{(t\star)}; \sigma_{ij}^{(t-1)}, w_{ij}^2)$, where w_{ij}^2 are the experimentally chosen variance of the proposal density and the mean of this density is the value of σ_{ij} that is current at the end of the $t-1$-th iteration. Then we accept the proposed $g_{ij}^{(t\star)}, \sigma_{ij}^{(t\star)}$, $\forall i < j;\ i, j = 1, \ldots, p$, at the probability of

$$\min\left(1, \frac{\prod_{i<j;i,j=1}^{p} \mathcal{N}\left(g_{ij}^{(t\star)}; \rho_{ij}^{(t)}, \sigma_{ij}^{(t\star)}\right)}{\prod_{i<j;i,j=1}^{p} \mathcal{N}\left(g_{ij}^{(t-1)}; \rho_{ij}^{(t)}, \sigma_{ij}^{(t-1)}\right)} \frac{Bernoulli(g_{ij}^{(t-1)}; \rho_{ij}^{(t)})}{Bernoulli(g_{ij}^{(t\star)}; \rho_{ij}^{(t)})}\right),$$

where $\mathcal{N}\left(g_{ij}; \rho_{ij}, \sigma_{ij}\right) = \dfrac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left[-\dfrac{\left(g_{ij} - \rho_{ij}\right)^2}{2\sigma_{ij}^2}\right]$. If the acceptance probability $> u'$, where $u' \in \mathcal{U}[0, 1]$, we accept the proposed values of the edge and variance parameters, i.e. set $g_{ij}^{(t)} = g_{ij}^{(t\star)}$ and $\sigma_{ij}^{(t)} = \sigma_{ij}^{(t\star)}$, $\forall\, i < j; i, j \in V$. Otherwise, the proposed values of the parameters are rejected and we set the current values in iteration t to be those of the previous iteration. The graph at the end of the t-th iteration is $\mathbb{G}^{(t)}(p, \boldsymbol{R}_t)$, where $\boldsymbol{R}_t$ is the updated partial correlation matrix in the t-iteration.

3 Stop if $t = N$; else repeat Steps 2.

3.2.4 Cholesky Factorisation and Matrix Inversion by Forward Substitution

Let a $p \times p$-square positive-definite (correlation) matrix be $\boldsymbol{\Sigma}_C^{(S)} = \boldsymbol{L}_C^{(S)}(\boldsymbol{L}_C^{(S)})^T$. The Cholesky factorisation of $\boldsymbol{\Sigma}_C^{(S)} = [s_{ij}]$ into its unique square root $\boldsymbol{L}_C^{(S)} = [l_{ij}]$ can be shown to be defined by the following scheme:

$$\begin{aligned}
l_{11} &= \sqrt{s_{11}}, \\
l_{i1} &= \frac{s_{i1}}{l_{11}}, \quad i = 1, \ldots, p, \\
l_{ij} &= \frac{\sqrt{s_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{kj}}}{l_{jj}} \quad j = 1, \ldots, i-1;\ i = 1, \ldots, p, \\
l_{ii} &= \sqrt{s_{ii} - \sum_{k=1}^{i-1} l_{ik}^2} \quad i = 2, \ldots, p,
\end{aligned} \tag{3.12}$$

while forward substitution seeks $\boldsymbol{L}_C^{-1}$ s.t. $\boldsymbol{L}_C \boldsymbol{L}_C^{-1} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the pXp-dimensional identity matrix. Then the scheme for forward substitution is the following:

$$\begin{aligned}
m_{11} &= \frac{1}{l_{11}}, \\
l_{i1} &= \frac{s_{i1}}{l_{11}}, \quad i = 1, \ldots, p, \\
l_{ij} &= \frac{\sqrt{s_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{kj}}}{l_{jj}} \quad j = 1, \ldots, i-1;\ i = 1, \ldots, p, \\
l_{ii} &= \sqrt{s_{ii} - \sum_{k=1}^{i-1} l_{ik}^2} \quad i = 2, \ldots, p,
\end{aligned} \tag{3.13}$$

3.2.5 Definition of 95% HPD credible regions on the random graph variable and the learnt graphical model

Bayesian inference is made on the random graph variable $\mathbb{G}(p, \boldsymbol{R})$, leading to one sampled graph at the end of each of the $N+1$ iterations of our inference scheme (Metropolis-with 2-block-update). The sample of graphs obtained from the post-burnin part of the MCMC chain then encompasses the learning of the graphical model of the data $\mathbf{D}_S$. In order to acknowledge uncertainties in the Bayesian learning of this graphical model, we need to include in its definition, only those sampled graphs that lie within an identified 95% HPD credible region. How can we model this uncertainty, and in particular, present a single representation of the learnt graphical model of $\mathbf{D}_S$, inclusive of such learnt uncertainties?

This concern is addressed by defining the fraction N_{ij} of the post-burnin number N_{post} of iterations (where $N_{post} < N+1$), in which the ij-th edge exists, i.e. G_{ij} takes the value $1, \forall\, i, j = 1, 2, \ldots, p,\ i \neq j$. Thus, the variable N_{ij} is defined to take the value

$$n_{ij} := \frac{\sum_{t=N-N_{post}+1}^{N} \mathbb{I}_1(g_{ij}^{(t)})}{N_{post}}, \quad i < j;\ i, j = 1, \ldots, p, \tag{3.14}$$

where the indicator function

$$\mathbb{I}_1(g_{ij}^{(t)}) = 1 \quad \text{if} \quad g_{ij}^{(t)} = 1$$
$$\mathbb{I}_1(g_{ij}^{(t)}) = 0 \quad \text{if} \quad g_{ij}^{(t)} = 0$$

Then N_{ij} is the fractional number of sampled graphs, in which an edge exists between vertices i and j. This leads us to interpret $\{N_{ij}\}_{i,j \in V;\, i<j}$ as carrying information about the uncertainty in the graph learnt given data $\mathbf{D}_S$; in particular, n_{ij} approximates the probability of existence of the edge between the i-th and j-th nodes in the graphical model of the data at hand. Indeed the N_{ij} parameters are functions of the partial correlation matrix $\boldsymbol{R}$ that is

learnt given this data, but for the sake of notational brevity, we do not include this explicit $\boldsymbol{R}$ dependence in our notation to denote the edge probability parameters.

So the set $\{\mathbb{G}(p,\boldsymbol{R}_t)\}_{t=N-N_{post}+1}^{N}$ of graphs on vertex set $\boldsymbol{V}=\{1,\ldots,p\}$ and edge matrix $\boldsymbol{G}_t$ that is updated given $\boldsymbol{R}_t$, is treated equivalently as the post-burnin sample $\{g_{12}^{(t)},g_{13}^{(t)},\ldots,g_{1p}^{(t)},g_{23}^{(t)},\ldots,g_{p\,p-1}^{(t)}\}_{t=N-N_{post}+1}^{N}$ of edge parameters. Only those edge parameters are included in the defined 95% HPD credible region, that occur with probability ≥ 0.05 in this sample. In other words, only for ij pairs s.t. $N_{ij}\geq 0.05$, define the g_{ij} parameters included in the set that comprises the 95% HPD credible region on the edge parameters, in our definition. Indeed, the graphical model of the data is then the set of those graphs on vertex set $\boldsymbol{V}=\{1,\ldots,p\}$, the existing edges of which are those G_{ij} parameters that lie within this defined 95% HPD credible region.

Theorem 3.2.2. *The graphical model of data $\mathbf{D}_S$ for which the between-column partial correlation matrix is $\mathbf{R}$, is the $\mathbf{R}$-dependent set or family $\mathcal{G}_{p,\boldsymbol{\Phi}(\boldsymbol{R})}$ of all inhomogeneous Binomial graphs $\mathbb{G}(p,\boldsymbol{R})$, the edge probabilities in which is given by the matrix $\boldsymbol{\Phi}(\boldsymbol{R})=[\phi_{ij}(R_{ij})]$, s.t. probability of the edge between the i-th and j-th nodes ($i\neq j$; $i,j\in V$) is*

$$\phi_{ij}(R_{ij})=\left[H(n_{ij}-0.05)\right]n_{ij}. \tag{3.15}$$

Here, n_{ij} is the value of the parameter N_{ij} defined in Equation 3.14, and $H(\cdot)$ is the Heaviside function [Duff and Naylor, 1966] s.t. the Heaviside or step-function of $A\in\mathbb{R}$ is

$$\begin{aligned} H(a) &= 1 \quad \textit{if} \quad a\geq 0 \\ &= 0 \quad \textit{if} \quad a<0. \end{aligned}$$

Only edges with non-zero edge probability $\phi_{ij}(R_{ij})$, are marked on the learnt graphical model, and the corresponding value of N_{ij} is written next to each such marked edge. Then by this definition, any graph $\mathbb{G}(p,\boldsymbol{R})\in\mathcal{G}_{p,\boldsymbol{\Phi}(\boldsymbol{R})}$ is sampled from within the 95% HPD credible region on inhomogeneous random Binomial graphs given the partial correlation matrix $\mathbf{R}$ of the data.

Thus, the binary edge parameter G_{ij} between the i-th and j-th nodes, takes the value 1 (i.e. the edge exists), with a learnt probability–in fact, the joint posterior of all G_{ij} parameters is learnt given the learnt correlation structure of the data, while acknowledging the

propagation of uncertainties in our learning of the correlation given the data, into the learning of the distribution of the G_{ij} parameters given this learnt partial correlation matrix $\boldsymbol{R}$. A summary of this learnt distribution is then the edge probability parameter $\phi_{ij}(R_{ij})$, the value of which is marked on the visualisation of the graphical model of the data against the edge between the i-th and j-th nodes, as long as $\phi_{ij}(R_{ij}) > 0$, i.e. $n_{ij} \geq 0.05$; $i \neq j$; $i, j \in \boldsymbol{V}$. In other words, only edges occurring with posterior probabilities in excess of 5% are included in this graphical model.

3.2.6 Incorporating measurement uncertainties in the learnt graphical model

If measurement errors affect the values of the i-th component Z_i of the p-dimensional vector-valued observable $\boldsymbol{Z}$, where measurements of Z_i comprise the i-th column of data $\mathbf{D}_S$, $(i = 1, \ldots, p)$, the variance of the probability distribution of such errors–if unknown–can be learnt given the data. So let the error in Z_i be ϵ_i that we assume is Normally distributed with variance v_{ϵ_i}, i.e. $\epsilon_i \sim \mathcal{N}(0, v_{\epsilon_i})$. Then if the unknown error variance v_{ϵ_i} is proposed in the t-th iteration of our MCMC chain to be $v_{\epsilon_i}^{(t\star)}$, the correlation $s_{ij}^{(t\star)}$ has to be adjusted by the factor $1/\sqrt{1 + v_{\epsilon_i}^{(t\star)}}$, $\forall j \neq i$.

3.3 Empirical illustration: simulated data

The simulated data used in this section, is a 5-columned data set $\boldsymbol{D}_{orig}$ (p=5) with number of rows $n_{orig} = 4000$, where $\boldsymbol{D}_{orig}$ is simulated to bear a chosen between-columns correlation

matrix $\mathbf{\Sigma}_C^{(true)}$ that is given as:

$$\begin{pmatrix} 1 & 0.9914 & -0.8964 & 0.02526 & 0.0656 \\ & 1 & -0.8916 & 0.01981 & 0.6647 \\ & & 1 & -0.009747 & -0.06140 \\ & & & 1 & 0.03622 \\ & & & & 1 \end{pmatrix}$$

which when inverted, allows for the computation of the empirical partial correlation matrix, following Equation 3.10. This empirical partial correlation matrix is $\boldsymbol{R}^{(true)}$:

$$\begin{pmatrix} 1 & 0.9574 & -0.2114 & 0.004786 & 0.005037 \\ & 1 & -0.04897 & 0.03900 & 0.01206 \\ & & 1 & 0.02736 & -0.006288 \\ & & & 1 & 0.03527 \\ & & & & 1 \end{pmatrix}$$

We randomly sample n (=300 typically) rows from this simulated data set $\boldsymbol{D}_{orig}$, to define our toy data set $\mathbf{D}_T$, that we will implement in the method, to

- learn the between-columns correlation matrix $\mathbf{\Sigma}_C^{(S)} = [S_{ij}]_{i=1;j=1}^{n,p}$ given the standardised version $\mathbf{D}_T^{(S)}$ of $\mathbf{D}_T$, and thereafter, learn the graphical model of data $\mathbf{D}_T^{(S)}$, as defined in Definition 3.2.2 with p=5 and partial correlation matrix $\boldsymbol{R} = [R_{ij}]_{i=1;j=1}^{n,p}$, where elements of $\boldsymbol{R}$ are computed using the learnt $\mathbf{\Sigma}_C^{(S)}$ in Equation 3.10. Here $\mathbf{D}_T^{(S)}$ comprises n simulated values of the variables $Z_1, \ldots, Z_5$.

- perform model checking using $\mathbf{D}_T^{(S)}$. To be precise, we predict the distribution of Z_i when in the identified test data, Z_j is restricted to take values in the chosen, narrow

interval $[z_j^{(0)} - \delta_j, z_j^{(0)} + \delta_j]$, for $j \neq i$; $i, j = 1, \ldots, 5$–and then compare the empirical distribution of Z_i in the test data, with the posterior predictive distribution of Z_i, given the correlation matrix learnt using $\mathbf{D}_T^{(S)}$. Also, given $\mathbf{D}_T^{(S)}$ and Z_j, MCMC-based sampling is performed from the joint posterior of $\{Z_i\}_{i=1;i\neq j}^{i=p}$ and $\mathbf{\Sigma}_C^{(S)}$. This is discussed in Section 3.3.

– learn the correlation matrix and graphical model of the data, where a chosen measurement error is placed on Z_i, $i = 1, \ldots, p$; the unknown variance v_{ϵ_i} of this error density is also learnt.

Plots of Z_i against Z_1 are included in Figure 3.1; $i = 2, 3, 4, 5$.

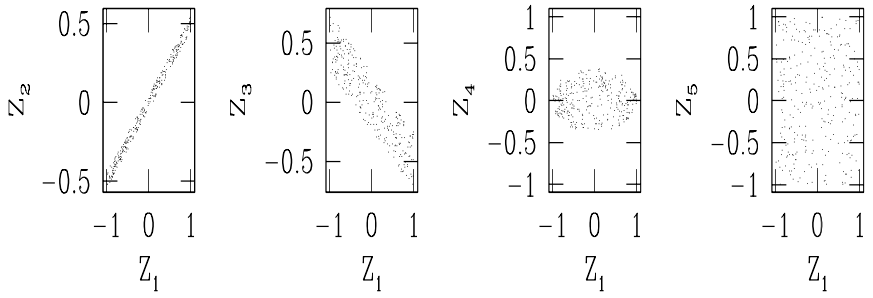

Figure 3.1: Plots of Z_i against Z_1 in the standardised version of the toy data $\mathbf{D}_T^{(S)}$ simulated to bear the empirical column-correlation matrix $\mathbf{\Sigma}_C^{(true)}$; here $i = 2, 3, 4, 5$. The toy data $\mathbf{D}_T^{(S)}$ comprises n measurements of the variables $Z_1, ..., Z_5$, with a typical n of 300.

3.3.1 Learning correlation matrix & graph given toy data $\mathbf{D}_T^{(S)}$

The between-columns correlation matrix $\mathbf{\Sigma}_C^{(S)}$ is learnt, given the standardised toy data $\mathbf{D}_T^{(S)}$ by employing the algorithm discussed in Section 3.2.3. Here $n = 300$, $p = 5$, and with the aim of estimating the normalisation $\hat{c}_t$ of the posterior in the t-th iteration, $K = 20$

is the chosen number of sampled data sets with $n^{/}$ rows and p columns, generated in each iteration, to bear the column-correlation matrix proposed in that iteration. Indeed, we set $n^{/} = n$. Here $t = 0, \ldots, N$.

In the t-th iteration of our MCMC chain, the first block update in our Metropolis-with 2-block-update-based inference scheme, leads to the updating of the column correlation matrix to $\mathbf{\Sigma}_t$ given the data $\mathbf{D}_T^{(S)}$, using which the value of the partial correlation matrix $\boldsymbol{R}_t = [\rho_{ij}^{(t)}]$ is computed in this iteration. Then the second block update leads to the updating of the values of the binary graph edge parameters to $g_{ij}^{(t)}$ and variance parameters to $\sigma_{ij}^{(t)}$, given $\boldsymbol{R}_t$. Traces of the marginal posterior probability of five of the S_{ij} parameters given data $\mathbf{D}_T^{(S)}$ are shown in the top left panel Figure 3.2, while the joint posterior of all G_{ij} and σ_{ij} parameters given the learnt partial correlation matrix, is shown in the top left panel Figure 3.3. Histograms representing approximations of marginals of individual R_{ij} and σ_{ij} parameters, given the data and the learnt partial correlation respectively, occupy other panels of Figure 3.2 and Figure 3.3 respectively. Here $i < j; i, j = 1, \ldots, p$.

The graphical model of the data $\mathbf{D}_T^{(S)}$ is presented in Figure 3.4. The fraction n_{ij} of post-burnin samples of g_{ij} with a value of 1, i.e. an approximation to the probability of existence of the edge joining nodes i and j, is marked next to each edge of the graph, as long as $n_{ij} \geq 0.05$, i.e. the edge probability parameter $\phi_{ij}(R_{ij})$ is non-zero.

We note that the column correlation matrix $\mathbf{\Sigma}_C^{(S)}$ of the Gaussian Process that models the data, is such that the partial correlation ρ_{12} between Z_1 and Z_2 is learnt to be in the 95% HPD credible region of $\in [0.86, 0.95]$ approximately, which is close to the empirical value of 0.96. Again, the empirical value of ρ_{13} is about -0.2, and the learnt value is $\in [-0.44, -0.27]$ approximately; empirical value of ρ_{23} is about 0.04, and the learnt value is $\in [-0.11, 0.05]$ approximately. The other partial correlation parameters have smaller

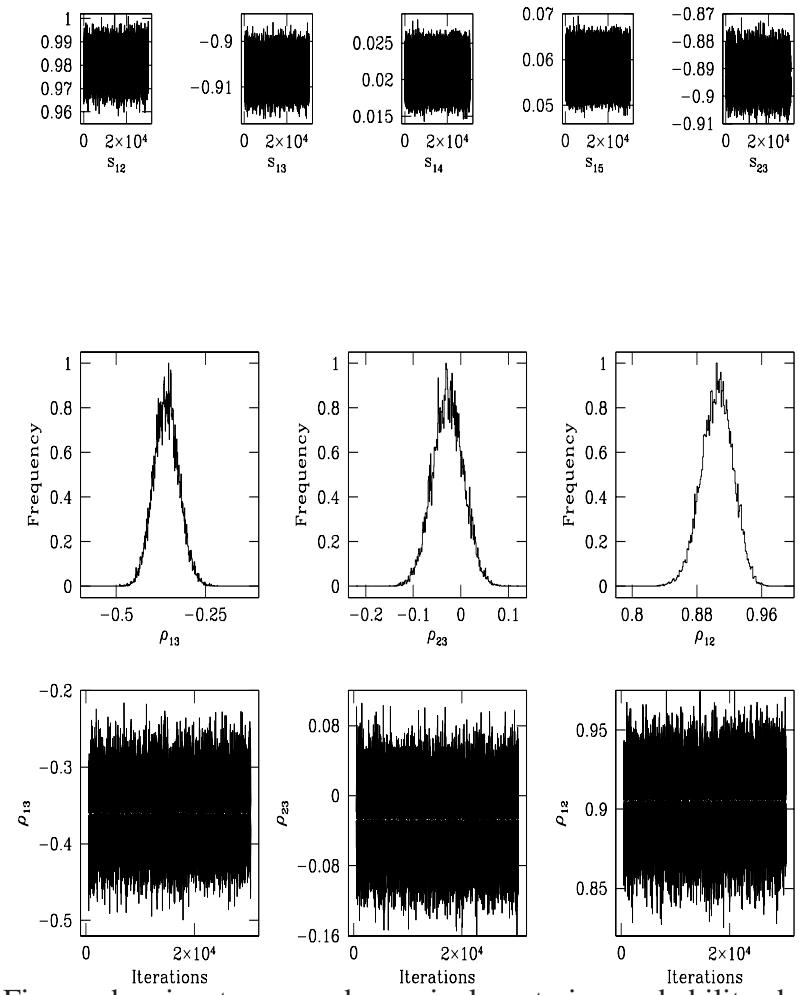

Figure 3.2: Figure showing traces and marginal posterior probability densities (as histograms) of elements of the correlation matrix $\mathbf{\Sigma}_C^{(S)}$, and partial correlation matrix $\boldsymbol{R}$, learnt given the toy data $\mathbf{D}_T^{(S)}$, in the method in which the data is modelled using a matrix-variate Gaussian Process, and the likelihood obtained by marginalising over the between-row correlation matrix. The top panel displays traces of the five correlation parameters $s_{12}, s_{13}, s_{14}, s_{15}, s_{23}$ given this toy data. The lower-most panel displays traces of the partial correlation parameters ρ_{12}, ρ_{13}, ρ_{23}, computed using correlation matrix $\mathbf{\Sigma}_C^{(S)}$ learnt given $\mathbf{D}_T^{(S)}$, in Equation 3.10. The middle panel presents the marginals of these partial correlation parameters as histograms.

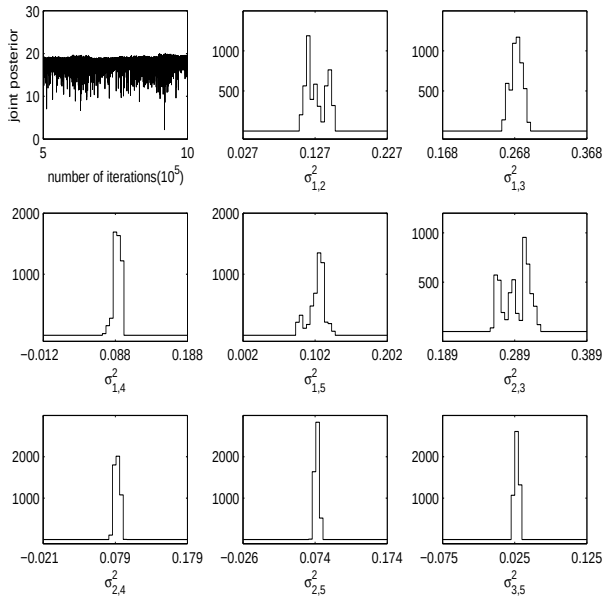

Figure 3.3: *Top left:* trace of joint posterior probability density of the graph edge parameters g_{ij} and variance parameters σ^2_{ij}, given the partial correlation matrix learnt in the first block update of our Metropolis-with 2-block-update inference scheme, given the 5-columned toy data set $\mathbf{D}_T^{(S)}$. *Other panels:* histogram approximations to the marginal posterior probability density of three of the variance parameters.

values in the chosen correlation structure that the data is simulated to bear–each of which is close to the corresponding learnt value. This offers confidence in our method of learning the correlation matrix $\mathbf{\Sigma}_C^{(S)}$ of the standardised toy data $\mathbf{D}_T^{(S)}$.

3.3.2 Effect of measurement uncertainties in learning of correlation and graph

As discussed in Section 3.2.6, in this approach, incorporate measurement errors in one or more of the variables $Z_1, \ldots, Z_p$, can be incorporated into the learning of the between-columns correlation structure given the toy data set $\mathbf{D}_T^{(S)}$, that then affects the learning of the graphical model of this data. Let the unknown variance of the error distribution of Z_i be v_{ϵ_i}, $(i = 1, \ldots, p)$. Then, one model of the total variance of Z_i is $s^2_{ii} + v_{\epsilon_i}$, where s^2_{ii} would

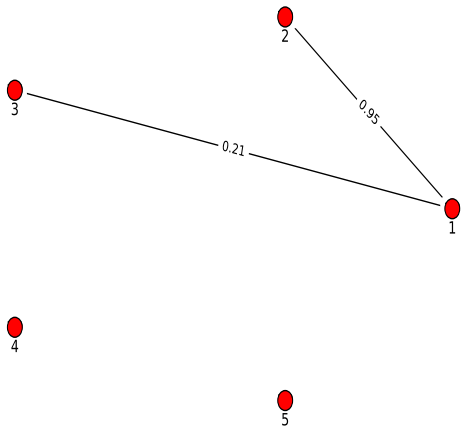

Figure 3.4: Figure showing graphical model of toy data $\mathbf{D}_T^{(S)}$–learnt in the Metropolis-with 2-block-update inference scheme in which the correlation matrix $\mathbf{\Sigma}_C^{(S)}$ of the data is learnt, simultaneously with the graph. The observables $Z_1, ...Z_5$, measurements of which comprise the data, are marked by filled red circles, as the 5 nodes in this graph. The probability of the edge parameter g_{ij} to exist (i.e. for g_{ij} to be 1)–$i \neq j$, $i, j = 1, \ldots, 5$–is approximated by the fraction n_{ij} of post-burnin iterations in which the current value of g_{ij} is 1. This value of n_{ij} is marked against the edge joining the i-th and j-th nodes, as long as $n_{ij} > 0.05$.

be the variance of Z_i, had Z_i been free of any measurement errors. However, Z_i results from standardising the i-th observable X_i, by its empirical variance, i.e. s_{ii}^2 is unity by design. Thus, when measurement error is no longer absent, the variance of Z_i increases to $1 + v_{\epsilon_i}$ assuming Z_i to be independent of the error in Z_i, so that the variances add linearly. Indeed, in the presence of measurement error in X_i, the absolute value of the correlation s_{ij} between Z_i and Z_j decreases (by a factor of $\sqrt{1 + v_{\epsilon_i}}$ in the model in which variances add linearly).

On the other hand, the partial correlation ρ_{ij} may increase or decrease [Liu, 1988]. That such is a possibility, is corroborated in the correlation and partial correlation structures of an example data set that comprises measurements of a 3-dimensional observable vector

$(Z_1, Z_2, Z_3)^T$. Then, $\rho_{ij} = \frac{s_{ij} - s_{ik}s_{jk}}{\sqrt{(1-s_{ik}^2)(1-s_{jk}^2)}}$, $i \neq j, i \neq k, k \neq j; i, j, k = 1, 2, 3$. It follows that if $|s_{ij}|$ and $|s_{ik}|$ decrease, ρ_{ij} can either increase or decrease. But ρ_{ij} is the probability for the edge between the i-th and j-th nodes of the graph of this data, to exist, i.e. $\rho_{ij} = \Pr(g_{ij} = 1)$. Then it is possible that while in the absence of measurement errors, $g_{ij} = 1$ during a fraction $n_{ij} < 0.05$ of the number of post-burnin iterations, in the presence of measurement error in X_i, ρ_{ij} increases sufficiently to ensure that the fraction of iterations during which this edge exists is in excess of 0.05. If this happens, the edge between the i-th and j-th nodes will be included in the graphical model of the data when measurement error in X_i is acknowledged, but not when such error is not. In other words, ignoring measurement uncertainties can lead to a potential misrepresentation of the graphical model of the data at hand.

It is possible to produce graphs while ignoring, as well as acknowledging the measurement uncertainty in one or more components of the p-dimensional observable vector, n measurements of which results in the rectagularly-shaped data at hand. In fact, it is also possible to learn the variance of the error density of the components of this observable. This is demonstrated in the experiment discussed below.

In this implementation, we add measurement error to the 2nd component X_2 of the 5-dimensional observable vector, n standardised measurements of which comprise data $\mathbf{D}_T^{(S)}$. Let us impose Gaussian measurement errors on Z_2, s.t. this Gaussian error density is $\epsilon_2 \sim \mathcal{N}(0, 0.01)$. We then define a data set that is the same as $\mathbf{D}_T^{(S)}$, except that the 2-nd column of this data is now sampled from a Gaussian with zero mean and variance given by 1+0.01, i.e. sampled from the convolution of a standard Normal, with the density $\mathcal{N}(0, 0.01)$. The resulting data set is referred to as $\mathbf{D}_T^{(err)}$. Thus, the true value of the

variance v_{ϵ_2} of the 2nd column of the data $\mathbf{D}_T^{(err)}$ is 0.01. We will treat this variance as an unknown and in fact, learn this value using $\mathbf{D}_T^{(err)}$.

We learn the column-correlation matrix of this data using the method delineated in Section 3.2.3 above, using an MCMC chain that we run with this data $\mathbf{D}_T^{(err)}$. The only exception to the method of learning the s_{ij} parameters is that the correlation between the Z_i and Z_j is given by $\frac{s_{ij}}{\sqrt{(1+v_{\epsilon_i})(1+v_{\epsilon_j})}}$ in the model in which the variances are assumed to add linearly; $i \neq j; i,j = 1,\ldots,p$. Thus, in addition to the $p(p-1)/2$ number of s_{ij} parameters, we now also learn the p number of v_{ϵ_i} parameters, where the latter is the variance of the error distribution of Z_i. We actually learn the standard deviation of the error density on Z_i, namely γ_i, i.e. $v_{\epsilon_i} = \gamma_i^2$. In the t-th iteration, we propose γ_i from a Gaussian proposal density that has the mean given by the current value of the parameter in this iteration, and an experimentally chosen variance. Here $t = 0,\ldots,N$. This is undertaken $\forall i = 1,\ldots,p$. The S_{ij} parameters are always proposed from Truncated Normal proposal densities that are left and right truncated at -1 and 1 respectively and have mean given by the current parameter value, while the variance is fixed. Then the correlation parameters that define the correlation matrix in the t-th iteration, are $s_{ij}^{(t\star)} / \sqrt{(1+(\gamma_{\epsilon_i}^{(t\star)})^2)(1+(\gamma_{\epsilon_j}^{(t\star)})^2)}$, $i \neq j; i,j = 1,\ldots,p$. Gaussian priors are used for the S_{ij} parameters, where such a Gaussian is centred on the empirical correlation between Z_i and Z_j in the data, while uniform priors are used on all other parameters. Using the proposed and current correlation matrices in our Metropolis-Hastings inferential scheme, we compute the marginals of the individual S_{ij} parameters as well as the γ_i parameters ($\gamma_i^2 = v_{\epsilon_i}$).

Histogram representations of the marginals (normalised to 1 at the mode), of some of these parameters are displayed in Figure 3.5. The 95% HPD credible region on γ_2 that are

learnt given this data is [-0.2,0.2] approximately. The learnt standard deviations of the error densities of variables other than Z_2, are 0 approximately. We also note from this figure that the changes in the partial correlations introduced by the introduction of the measurement error in one variable, can be both an increase and decrease–this is discussed above. The effect on introducing this measurement error on Z_2, on the graphical model of the data $\mathbf{D}_T^{(err)}$, is presented in Figure 3.6. In this graphical model, the edge G_{23} between the 2-nd and 3-rd nodes takes the value 1, with probability of about 0.16, while n_{23} was less than 0.05 in the graphical model of data $\mathbf{D}_T$–which differs from $\mathbf{D}_T^{(err)}$ only in that the 2nd column is imposed with a Gaussian error of variance 0.01. Thus, the effect of introducing this error to measurements of the variable Z_2 propagates into the (partial) correlation structure of the data, to then affect the graphical model. Comparing this learnt graph to the graph of the toy data $\mathbf{D}_T^{(S)}$, we recognise that measurement errors can distort the graphical model of a data.

3.4 Model checking

Section 3.3 discusses the learning of $\mathbf{\Sigma}_C^{(S)}$ using the n rows of the standardised toy data $\mathbf{D}_T^{(S)}$, which is a 300-row subset from the 5-columned simulated dataset $\boldsymbol{D}_{orig}$, discussed in the previous section, where $\boldsymbol{D}_{orig}$ is generated to abide by a chosen correlation matrix $\mathbf{\Sigma}_C^{(true)}$ that is defined above in Section 3.3. Then $\mathbf{D}_T^{(S)}$ comprises 300 different measurements of the 5-columned vector $\boldsymbol{Z} := (Z_1, Z_2, Z_3, Z_4, Z_5)^T$, where Z_i is a standardised variable $i = 1, \ldots, 5$. Having learnt the parameters of the Gaussian Process in Section 3.3–of which the standardised observable $\boldsymbol{Z} \in \mathbb{R}^p$ is a realisation–here we want to predict values of Z_i for values of Z_j as given in a new or test data, ($j \neq i$; $i, j = 1, \ldots, p$); for our purposes, p=5. This test data $\mathbf{D}_{test}$ is built to be independent of the training data $\mathbf{D}_T^{(S)}$,

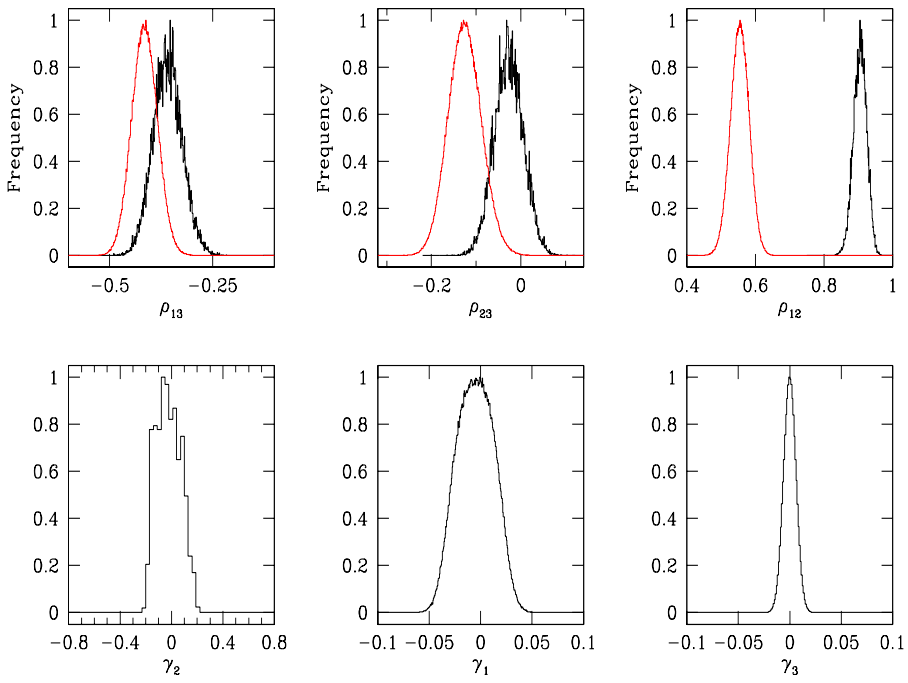

Figure 3.5: *Top panels:* comparison of histogram representation (in black) of the marginal posterior density of some partial correlation parameters (ρ_{ij}) learnt given toy data $\mathbf{D}_T^{(S)}$, with the marginals (in grey, or red in the electronic version), of the same parameter, learnt given the data $\mathbf{D}_T^{(err)}$, which differs from $\mathbf{D}_T^{(S)}$, in only that Gaussian errors of variance 0.01 are imposed on the variable Z_2. i.e. the 2nd component of the 5-dimensional observable vector $(Z_1, Z_2, Z_3, Z_4, Z_5)^T$, measurements of which comprise the data. Here $i, j = 1, ..., 5; i \neq j$. From left to right, are presented the results for ρ_{12}, ρ_{13} and ρ_{23}. *Lower panels:* histogram representations of the standard deviation γ_i of the error density in the measurement of Z_i, learnt using data $\mathbf{D}_T^{(err)}$, for $i = 2, 1, 3$ from the left to the right panels, where in this data, Z_2 is the only one of the 5 variables that has an error (of standard deviation 0.1) imposed on it.

as q rows of the standardised version of the bigger data set $\mathbf{D}_{orig}$–of which $\mathbf{D}_T^{(S)}$ is also a subset–although the q rows of $\mathbf{D}_{orig}$ that comprise $\mathbf{D}_{test}$, are chosen as distinct from the n rows of the training data $\mathbf{D}_T^{(S)}$. The standardised test data $\mathbf{D}_{test}$ has $p = 5$ columns and q rows; in fact, q is set s.t. $q = n$.

Though the first part of this paragraph was suggested performing model checking using

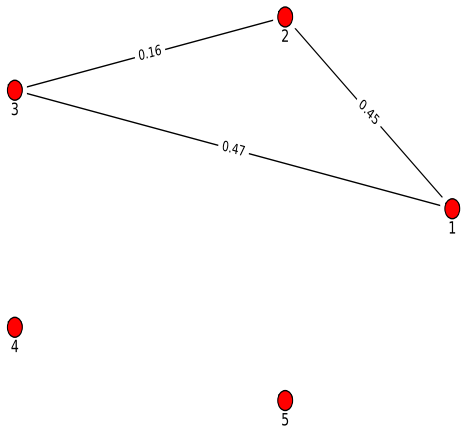

Figure 3.6: Figure showing graphical model of data $\mathbf{D}_T^{(err)}$ that differs from the toy data $\mathbf{D}_T^{(S)}$ only in that Gaussian errors (with variance 0.01) are added to the 2nd column of $\mathbf{D}_T^{(S)}$, to realise $\mathbf{D}_T^{(err)}$. The inclusion of measurement noise in this column of the toy data is noted in the learnt graphical model of the resulting error-bearing data $\mathbf{D}_T^{(err)}$, which manifests the edge between variables Z_2 and Z_3, while this edge is absent in the graphical model of the error-free data $\mathbf{D}_T^{(S)}$; see Figure 3.4.

predicted and true values of all 4 of Z_2 to Z_5, at given values of Z_1, given that the simulated Z_5 is known to be uncorrelated to Z_1, and occupying the interval [-1,1] uniformly, at any z_1. So it only Z_2, Z_3, Z_4 that are predicted at each of the known q (=n) values of Z_1 in the test data $\mathbf{D}_{test}$, given the GP parameters (i.e. the between-columns covariance matrix $\mathbf{\Sigma}_C^{(S)}$) that we learn using the training data. No prediction of Z_5 is undertaken as Z_5 is not correlated . In fact, we will sample from the posterior predictive density of Z_2, Z_3, Z_4, given the correlation matrix learnt using training data $\mathbf{D}_T^{(S)}$, and values of Z_1 in the test data $\mathbf{D}_{test}$. We compare the predicted values of Z_2, Z_3, Z_4 against their empirical values in the test data. Such a comparison constitutes the checking of our models s well as the results (of the learning of $\mathbf{\Sigma}_C^{(S)}$ given the training data $\mathbf{D}_T^{(S)}$). We clarify this prediction now.

As we learn the marginal posterior probability density of each correlation parameter S_{ij} given $\mathbf{D}_T^{(S)}$, we need to choose a summary of this marginal distribution, at which the prediction of the z_{ik} is undertaken, $i = 2, 3, 4$, $k = 1, \ldots, n$. We choose the mode of the marginal as this summary. Denoting the value of Z_i in the k-th row of the test data as z_{ik}, $(k = 1, \ldots, q = n)$, the learning of $\{z_{2k}, z_{3k}, z_{4k}\}_{k=1}^{n}$ is undertaken, in the test data $\mathbf{D}_{test}$, given values of $\{z_{1k}\}_{k=1}^{n}$ in $\mathbf{D}_{test}$ and the modal values of S_{ij} learnt using the training data $\mathbf{D}_T^{(S)}$. In this Bayesian, MCMC-based inferential approach, this learning is equivalent to sampling from the posterior predictive of the unknowns, i.e. performing MCMC-based posterior sampling from

$$\pi(z_{21}, z_{31}, z_{41}, \ldots, z_{2n}, z_{3n}, z_{4n} | z_{11}, \ldots, z_{1n}, s_{12}^{(M)}, \ldots, s_{1p}^{(M)}, s_{23}^{(M)}, \ldots, s_{2p}^{(M)}, \ldots, s_{p-1\,p}^{(M)}),$$

where $s_{ij}^{(M)}$ represents the modal value of the correlation parameter S_{ij} that we learn given the training data $\mathbf{D}_T^{(S)}$. The learnt "modal" correlation matrix is defined to be $\mathbf{\Sigma}_C^{(M)} = [s_{ij}^{(M)}]$.

In the t-iteration, a value $z_{ik}^{(t\star)}$ is proposed from a Gaussian proposal density with mean given by the current value $z_{ik}^{(t-1)}$ of this variable, and fixed variance ν_{ik}, i.e. the proposed value is $z_{ik}^{(t\star)} \sim \mathcal{N}(z_{ik}^{(t-1)}, \nu_{ik})$; this is done for $i = 2, 3, 4$ and $\forall k = 1, \ldots, n$, at each $t = 0, \ldots, N$. Then the proposed data in the t-th iteration is $\mathbf{D}^{(t\star)} = \left(\boldsymbol{z}_1, \boldsymbol{z}_2^{(t\star)}, \boldsymbol{z}_3^{(t\star)}, \boldsymbol{z}_4^{(t\star)}, \boldsymbol{z}_5\right)$, where $\boldsymbol{z}_i = (z_{i1}, \ldots, z_{in})^T$, $i = 1, \ldots, 5$. The posterior of the unknowns is then given as in Equation 2.8, with the data given by $\mathbf{D}^{(t\star)}$ and the modal correlation matrix given by $\mathbf{\Sigma}_C^{(M)}$ learnt using the training data set $\mathbf{D}_T^{(S)}$. The normalisation of the posterior is computed in the t-th iteration in the way described above in SEction 3.2.1, at the $\mathbf{\Sigma}_C^{(M)}$. Uniform priors are used on all unknowns. So in each iteration, we (use Random-Walk Metropolis to) sample from the posterior of the unknown variables, given $\mathbf{\Sigma}_C^{(M)}$ and the data on the $q = n$

number of Z_1 values in the test data $\mathbf{D}_{test}$. We implement such posterior sampling to compute marginal predictive of each of the unknowns. We compare this marginal predictive of of Z_2, Z_3, Z_4, to the empirical distribution of Z_2, Z_3, Z_4 in the test data $\mathbf{D}_{test}$. We also compare the plots of the predicted Z_i and the known Z_1 values, to the corresponding plot of empirical value of Z_i and Z_1; $i = 2, 3, 4$. The results of this comparison for Z_2, Z_3 and Z_4 are included in Figure 3.7.

Figure 3.7 shows that the plots of the predicted values of Z_i, $i = 2, 3, 4$, against Z_1 (in red filled circles in the electronic version, and grey circles in the monochrome version), compare favourably–visually speaking–to the plots of the empirical Z_i (in the test data), against Z_1. To be precise, the red (or grey) circles comprise predicted (or learnt) pair $(z_{1k}, z_{ik}^{(mode)})$ for $k = 1, \ldots, q = n$, where $z_{ik}^{(mode)}$ is the modal value of the marginal posterior density of Z_{ik} given known values of Z_1 in the test data, and the (modal) correlation matrix $\mathbf{\Sigma}_C^{(M)}$ (itself learnt given the training data). The black circles represent the empirical values (z_{1k}, z_{ik}) for $k = 1, \ldots, n$, i.e. the pair in the k-th row of the test data. We also plot the marginal of the learnt values of Z_i given the data, superimposed on the frequency distribution of the empirical value of Z_i in the test data–we do this for each $i = 2, 3, 4$. Again, the overlap between the results is encouraging. Thus, the predictions offer confidence in our model, as well as the results of our learning of the correlation structure of the data.

However, conditioning the posterior predictive of Z_i on a summary–modal in our earlier implementation–correlation matrix learnt given training data $\mathbf{D}_T^{(S)}$ is restrictive in that this approach ignores the learnt distribution of the correlation matrices. After all, learning of the correlation matrix given $\mathbf{D}_T^{(S)}$ is MCMC-based, generating a value of $\mathbf{\Sigma}_C^{(S)}$ in each iteration. In light of this, the marginal posterior of Z_i obtained by marginalisation over the joint posterior probability density of all unknown components of $\boldsymbol{Z}$ and $\mathbf{\Sigma}_C^{(S)}$ is a possibility.

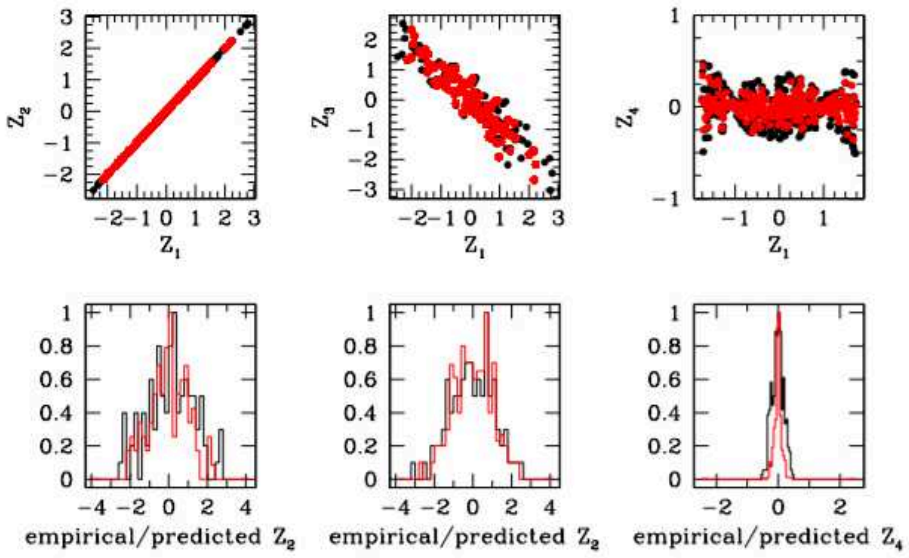

Figure 3.7: *Top panels:* figures comparing plots of empirical and predicted values of Z_i against values of Z_1, for $i = 2, 3, 4$ moving from left to the right panel. Grey (red in the electronic version) circles depict pairs of (z_{1k}, z_{ik}) in the test data $\mathbf{D}_{test}$, while black circles depict Z_i values learnt given the first column of the test data and the modal correlation matrix $\mathbf{\Sigma}_C^{(M)}$ that is itself learnt using the training data set $\mathbf{D}_T^{(S)}$. *Lower panels:* marginal of Z_i given 1st column of test data and $\mathbf{\Sigma}_C^{(M)}$, plotted as a histogram in grey (or red in the electronic version), over its empirical distribution in black, i.e. the histogram of the i-th column of the test data. Here, $i = 2, 3, 4$ as we move from left to right.

Thus, we learn $\mathbf{\Sigma}_C^{(S)}$ simultaneously with Z_2, Z_3, Z_4, i.e. the 2nd, 3rd and 4th columns of the test data, given the training data and the 1st column of the test data. Subsequently, MCMC-based posterior sampling is performed from the joint posterior probability density:

$$\pi\left(s_{12}, \ldots, s_{1p}, s_{23}, \ldots, s_{2p}, \ldots, s_{p-1\,p}, z_{21}, \ldots, z_{2n}, z_{31}, \ldots, z_{3n}, z_{41}, \ldots, z_{4n} | z_{11}, z_{1n}, \mathbf{D}_T^{(S)}\right). \tag{3.16}$$

In order to implement this, propose $z_{21}^{(t\star)}, \ldots, z_{2n}^{(t\star)}, z_{31}^{(t\star)}, \ldots, z_{3n}^{(t\star)}, z_{41}^{(t\star)}, \ldots, z_{4n}^{(t\star)}$ in each of the t iterations, $t = 0, \ldots, N$. Each of these parameters is proposed from a Gaussian proposal density (with mean given by the current value and an experimentally chosen variance). At the same time, the s_{ij} parameters, $i \neq j$, $i, j = 1, \ldots, p$ are proposed from a

Truncated Normal proposal density, truncated at -1 and 1, with mean given by the current value of the parameter, and chosen variance.

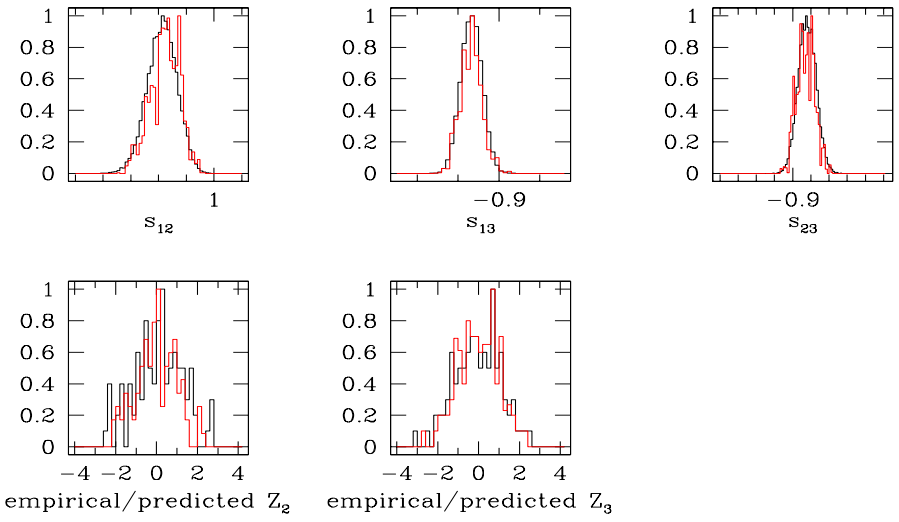

Figure 3.8: *Top panels:* grey (red in the electronic version) coloured histograms represent the marginal posterior density of S_{ij} learnt, (along with the Z_i parameters; $i = 2, 3, 4$), given the training data $\mathbf{D}_T^{(S)}$, and the known 1st column of the test data $\mathbf{D}_{test}$. This is compared to the marginal of S_{ij} learnt (when the column-correlation matrix is learnt alone), given training data–presented as the histograms in black. Panels from left to right correspond to the results for S_{12}, S_{13} and S_{23} respectively. The lower panels present the comparison between the empirical distribution of the i-th column of the test data $\mathbf{D}_{test}$–in black–and the joint posterior of Z_i, (learnt along with the S_{ij} parameters), given $\mathbf{D}_T^{(S)}$, and the 1st column of $\mathbf{D}_{test}$, (in grey, or red in the electronic version). Here $i = 2$, in the bottom left panel and $i = 3$ in the right.

For this implementation, at the t-th iteration, the augmented data $\mathbf{D}_A^{(t\star)}$ is defined; this is the training data $\mathbf{D}_T^{(S)}$, augmented by the data set $\mathbf{D}^{(t\star)}$ proposed in the t-th iteration, (defined above), where the 1st and 5th columns of $\mathbf{D}^{(t\star)}$ are the known 1st and 5th columns of the test data $\mathbf{D}_{test}$, and the i-th column is the proposed vector $(z_{i1}^{t\star}, \ldots, z_{in}^{t\star})^T$, $i = 2, 3, 4$. Thus, as the proposed $\mathbf{D}^{(t\star)}$ varies from one iteration to the next, the augmented data $\mathbf{D}_A^{(t\star)}$

also varies. This augmented data then has p columns nd $n+q$ rows, i.e. $2n$ rows, given our choice of $q = n$. In the t-th iteration, the posterior probability density of the unknowns given this augmented data $\mathbf{D}_A^{(t\star)}$ is computed, using the posterior defined in Equation 2.8 of WMC in which the generic data $\mathbf{D}_S$ is now replaced by $\mathbf{D}_A^{(t\star)}$. While uniform priors are placed on the z_{ik} parameters, Gaussian priors on s_{ij} are used, with such a prior centred at the empirical value of the correlation between the i-th and j-th columns of the data, $(i, j = 1, \ldots, p)$; the variance of these Gaussian priors are experimentally chosen.

Some results of sampling from the joint defined in Equation 3.16 are shown in Figure 3.8. These include comparison of the histogram representations of the marginals of 3 correlation parameters S_{12}, S_{13}, S_{23}, learnt in this implementation given the augmented data, with the marginal of the same correlation parameter learnt given training data $\mathbf{D}_T^{(S)}$. The figure also includes a comparison of the empirical and predicted marginals of Z_2 and Z_3.

3.5 Implementation on real data

This section presents a real-data application of the Bayesian learning of the between-columns correlation matrix of a multivariate data set, and simultaneous learning of the graphical model of the data at hand, given the updated correlation structure. We use the relatively well-known data sets on 11 different chemical attributes and "quality" classes of red and white wines, grown in the Minho region of Portugal (referred to a "vinho verde"); these data sets have been considered before by Cortez et al. [1998] and discussed in the website https://onlinecourses.science.psu.edu/stat857/node/223. The data consists of information on 1599 red wines and 4898 white wines. Each of these data sets consists of 12 columns that contain information on physiochemical attributes of the

sampled wines; these properties are assigned the following names: "fixed acidity" (X_1), "volatile acidity" (X_2), "citric acid" (X_3), "residual sugar" (X_4), "chlorides" (X_5), "free sulphur dioxide" (X_6), "total sulphur dioxide" (X_7), "density" (X_8), "pH" (X_9), "sulphates" (X_{10}), "alcohol" (X_{11}) and "quality" (X_{12}). Then the n-th row and i-th column of the data matrix carries measured/assigned value of the i-th property of the n-th wine in the sample, where $i = 1, \ldots, 12$ and $n = 1, \ldots, n_{orig} = 1599$ for the red wine data $\mathbf{D}_{orig}^{(red)}$, while $n = 1, \ldots, n_{orig} = 4898$ for the white wine data $\mathbf{D}_{orig}^{(white)}$. We refer to the i-th vinous property to be X_i. Then $X_i \in \mathbb{R}_{\geq 0}\ \forall i = 1, \ldots, 11$, while X_{12} that denotes the perceived "quality" of the wine is a categorical variable. Each wine in these samples was assessed by at least three experts who graded the wine on a categorical scale of 0 to 10, in increasing order of excellence. The resulting "sensory score" or value of the "quality" parameter was a median of the expert assessments [Cortez et al., 1998]. We seek the graphical model given each of the wine data sets, in which the relationship between any X_i and X_j is embodied, $i \neq j;\ i, j = 1, \ldots, 12$. Thus, we seek to find out how the different vino-chemical attributes affect each other, as well as the quality of the wine, in the sample at hand.

Here, $X_1, \ldots, X_{11}$ are real-valued, while X_{12} is a categorical variable, and our methodology allows for the learning of the graphical model of a data set that in its raw state bears measurements of variables of different types. In fact, we standardise our data, s.t. X_i is standardised to Z_i, $i = 1, \ldots, p$, $p = 12$. We work with only a subset data set, (comprising only $n < n_{orig}$ rows of the available $\mathbf{D}_{orig}^{(\cdot)}$; $n = 300$ typically). Thus, the data sets with n rows, containing Z_i values, ($i = 1, \ldots, p = 12$), are $n \times p$-dimensional matrices each; we refer to these data sets that we work with, as $\mathbf{D}_S^{(white)}$ and $\mathbf{D}_S^{(red)}$, respectively for the white and red wines.

The aim here is to learn the between-column correlation matrix $\mathbf{\Sigma}_S^{(m)}$ given data $\mathbf{D}_S^{(m)}$,

and simultaneously learn the graphical model of this data using the methodology developed above; $m = white,\ red$;

The motivation behind choosing these data sets are basically three-fold. Firstly, multivariate, rectangularly-shaped, real-life data were sought, where such data would admit graphical modelling of the correlations between the different variables involved. Also, we wanted data, results from–at least a part of–which exists in the literature. Comparison of these published results, with our independent results can then illustrate strengths of our method. Thirdly, treating the red and white wine data as data realised at different experimental conditions, we would want to address the question of the distance between these data, and this is done by computing the distance between the graphical models of the two data sets. Hence our choice of the popular Portuguese red and white wine data sets, as the data that we implement to illustrate our method on. It is to be noted that a rigorous vinaceous implications of the results, is outside the scope and intent of this paper. However, we will make a comparison of our results with the results of the analysis of white wine data that is reported in: https://onlinecourses.science.psu.edu/stat857/node/223 precludes analysis of the red wine data.

3.5.1 Results given data $\mathbf{D}_S^{(white)}$

As per the underlying principle of our Bayesian learning methodology discussed above (Section 3.2), we model the observable vector $(Z_1, \ldots, Z_p)^T$ using a Gaussian Process. Within a Metropolis-with 2-block-update inferential scheme (discussed in Section 3.2.3), we first perform the updating of the $p(p-1)/2$ parameters $S_{12}, \ldots, S_{p-1\,p}$ that are the elements of the upper triangle of the between-column correlation matrix $\mathbf{\Sigma}_S^{(white)}$, given data $\mathbf{D}_S^{(white)}$ that has $n = 300$ number of rows and $p = 12$ number of columns. The

proposal and prior densities on the S_{ij} parameters are as discussed in Section 3.2. The posterior probability density of the unknowns given the data is computed using these chosen priors and likelihood that is given in Equation 3.8. The updated correlation matrix is then used to compute the updated partial correlation matrix, given which, we update the binary graph edge parameters G_{ij} and the parameters σ_{ij}^2 that provide the variance of the likelihood function defined in Equation 3.11. Priors and proposal densities of the unknowns are as discussed in Section 3.2.2. Marginals of all unknowns are computed, and traces of the joint posterior probability density in each of the two block updates in our MCMC chain, are examined. The graph edge probability parameter $\phi_{ij}(R_{ij})$ is also computed from the graphs sampled in the post-burnin part of the MCMC chain.

The top left-hand panel of Figure 3.9 presents the trace of the joint posterior probability density of the correlation parameters S_{ij} given the data $\mathbf{D}_S^{(white)}$. All the other panels of this figure include marginal posterior probabilities of some of the partial correlation parameters, with value ρ_{ij}, where the i-th variable is the i-th vinous parameter listed above, with $i = 1, \ldots, 12$; $j \neq i, j = 1, \ldots, 12$. Figure 3.10 presents the trace of the joint posterior of the G_{ij} and σ_{ij}^2 parameters, updated in the 2nd block of each iteration of our MCMC chain, at the updated (partial) correlation matrix. The other panels of this figure depict the histogram representation of the marginals of some of the σ_{ij}^2 parameters. Thus the sample of graphs, $\{\mathbb{G}^{(t)}(p, \boldsymbol{R}_t)\}_{t=N-N_{post}+1}^{N}$, was obtained, where each graph is on the vertex set $\boldsymbol{V} = \{1, \ldots, p\}$ and is learnt given the partial correlation matrix $\boldsymbol{R}_t$ in the t-th iteration of our MCMC chain. The graph edge probability parameter $\phi_{ij}(R_{ij})$ is computed for each ij-pair of nodes in this sample, and include only those edges in the graphical model of the $\mathbf{D}_S^{(white)}$ data, that have non-zero $\phi_{ij}(R_{ij})$, i.e. $n_{ij} \geq 0.05$ (see Section 3.2.5). For these edges, the value n_{ij} is marked against the edge between the i-th and j-th nodes in the repre-

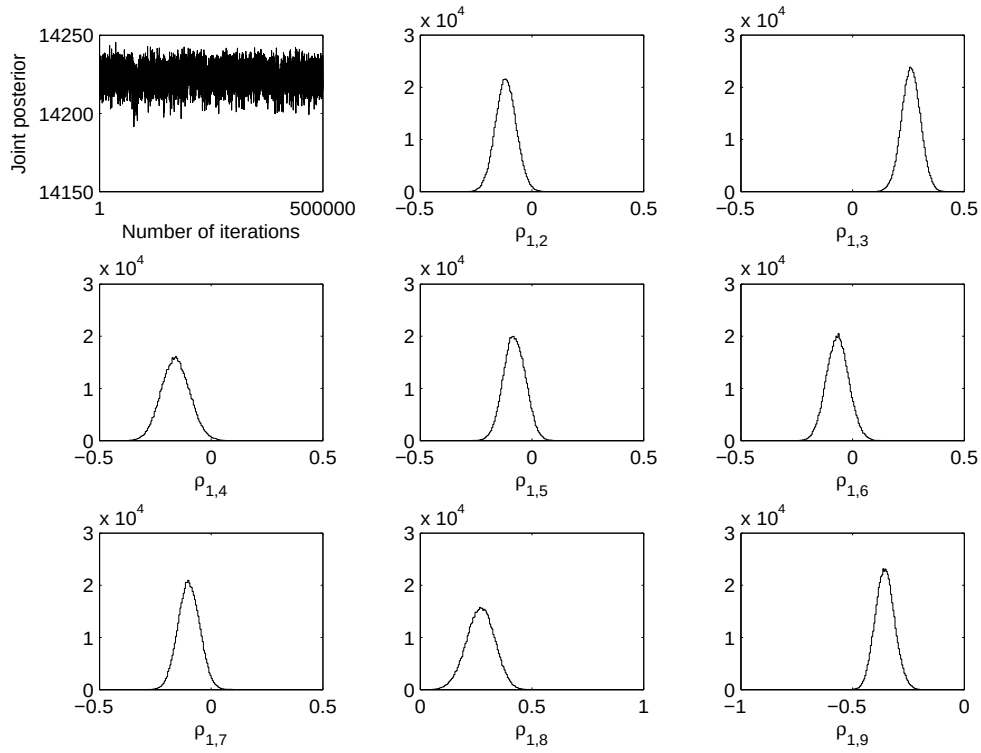

Figure 3.9: *Top left panel:* trace of the joint posterior probability density of the elements of the upper triangle of the between-columns correlation matrix of the standardised version of the real data $\mathbf{D}_S^{(white)}$ on Portuguese white wine samples [Cortez et al., 1998]; this data has $n = 300$ rows nd $p = 12$ columns, and is constructed as a randomly sampled subset of the original data, the sample size of which is 4898. *All other panels:* histogram representations of marginal posterior probability densities of some of the partial correlation parameters computed using the correlation matrix learnt given data $\mathbf{D}_S^{(white)}$. Random variable indexed with "1" is fixed acidity ; with '2' is volatile acidity; with '3' is citric acid; with '4' is residual sugar; with '5' is chlorides; with '6' is free sulphur dioxide; with '7' is total sulphur dioxide; with '8' is pH; with '9' is sulphate; with '10' is alcohol;

sentation of this graphical model of this white wine data set, that is shown in Figure 3.11.

Here $i \neq j$, $i, j = 1, \ldots, p = 12$.

3.5.1.1 Comparing against previous work done with white wine data

The graphical model of the white wine data presented in Figure 3.11 is strongly corroborated by the simple empirical correlations between pairs of different vino-chemical

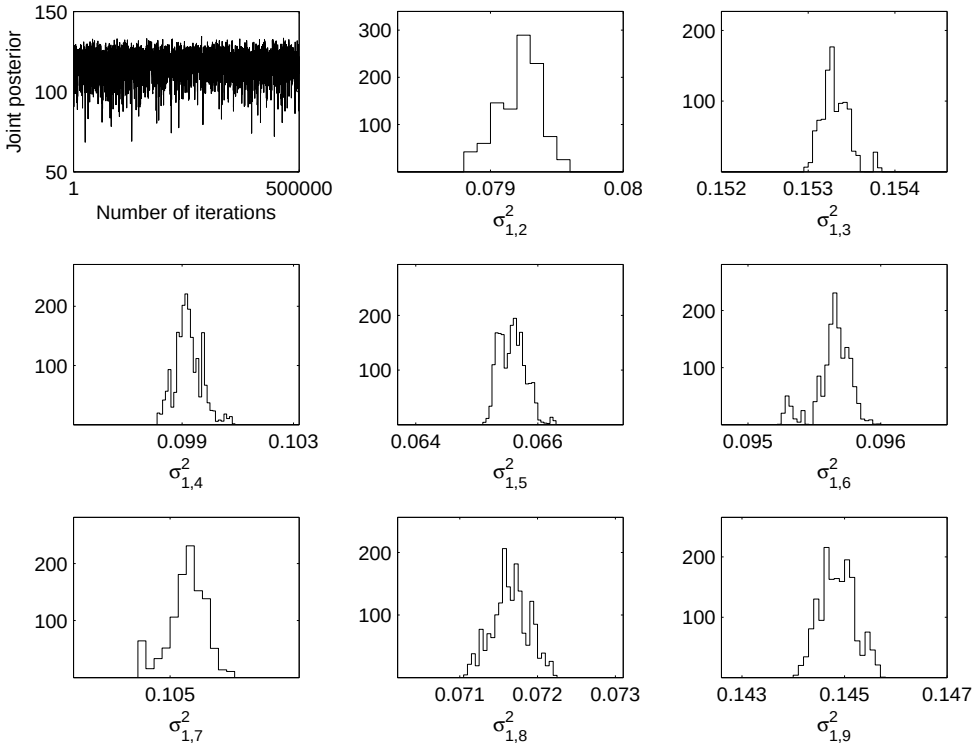

Figure 3.10: *Top left panel:* trace of the joint posterior probability of the graph edge parameters G_{ij} and the variance parameters σ^2_{ij} that are the variances used in the likelihood function defined in Equation 3.11; these parameters are updated within the 2nd block update of our Metropolis-with 2-block-update inferential scheme, at the correlation matrix that is updated given the data $\mathbf{D}_S^{(white)}$ of Portuguese white wine samples. Here $i \neq j$; $i, j = 1, \ldots, 12$. *All other panels:* histogram representations of marginal posterior probability densities of some of the variance parameters learnt given the correlation matrix that is itself learnt, given data $\mathbf{D}_S^{(white)}$.

properties–this correlation structure is apparent in the "scatterplot of the predictors" included as part of the results of the "Exploratory Data Analysis" reported in:

https://onlinecourses.science.psu.edu/stat857/node/224 on the white wine data. They use the full white wine data set $\mathbf{D}_{orig}^{(white)}$, to construct a matrix of scatterplots of X_i against X_j, where $i \neq j$; $i, j = 1, \ldots, 11$. It is to be noted that in the data analysis reported in:

https://onlinecourses.science.psu.edu/stat857/node/224, the ma-

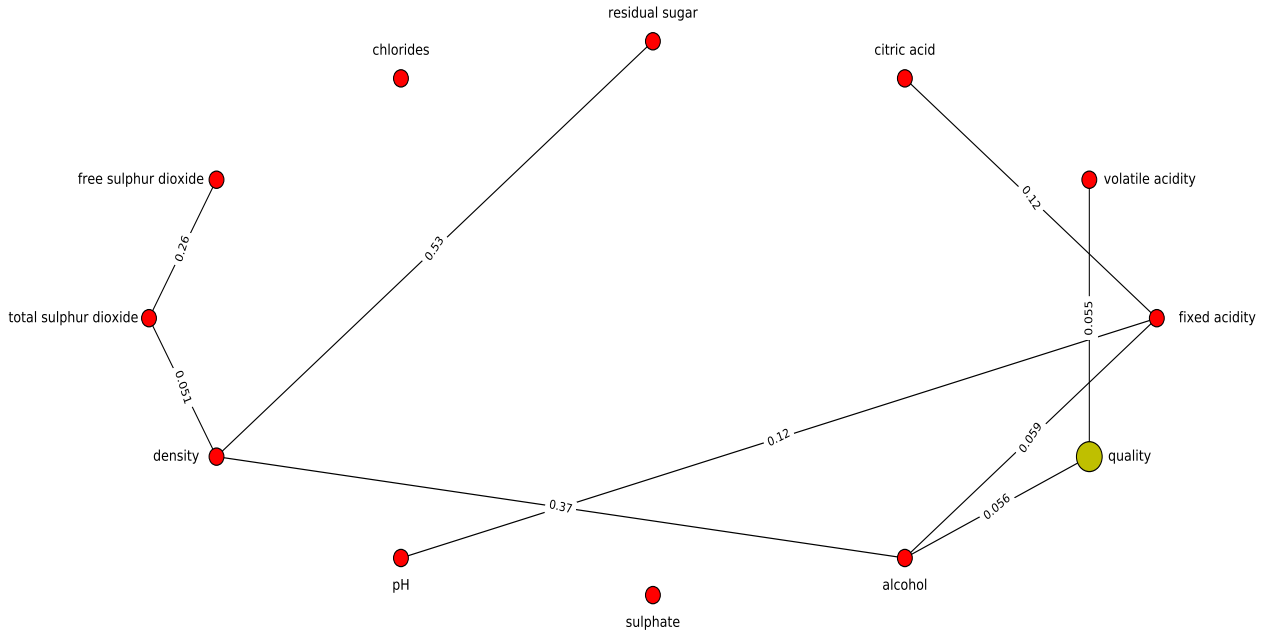

Figure 3.11: Figure showing graphical model of standardised version of the real data $\mathbf{D}_S^{(white)}$ on Portuguese white wine samples [Cortez et al., 1998]. Each of the first 11 columns of this data gives the measured value of each of 11 different vino-chemical properties of the wines in the sample–marked as nodes in the graph above, by filled red (or grey in the printed version) circles, with the name of the property included in the vicinity of the respective node. The 12-th column in the data includes values of the assessed quality of a wine in the sample, (a node that we mark with a green circle in the electronic version; the bigger grey circle in a monochromatic version of the paper). The probability for an edge to exist in the post-burnin sample of graphs generated in our MCMC-based inferential scheme, is marked against an existing edge, where edges with such probabilities that are < 0.05 are omitted from this graphical model, as included within a pre-defined 95% HPD credible region (defined in Section 3.2.5) on the MCMC-based sample of graphs.

trix of scatterplots of pairs of variables i and j was included, where this set of variables excluded the last column of the white wine data–the column that informs us of the assessed "quality" of the wine.

When comparing the learnt graphical model with the results of this reported "Exploratory Data Analysis", we recall that partial correlation (that drives the probability of the edge between the i-th and j-th nodes), is often smaller than the correlation between the i-th and j-th variables, computed before the effect of a third variable has been removed [Sheskin, 2004]. If this is the case, then an edge between nodes i and j in the learnt graphical model, is indicative of a high correlation between the i-th and j-th variables in the data. However, in the presence of a suppressor variable (that may share a high correlation with the i-th variable, but low correlation with the j-th), the absolute value of the partial correlation parameter can be enhanced to exceed that of the correlation parameter. In such a situation, the edge between the nodes i and j in the learnt graphical model may show up (within our defined 95% HPD credible region on edge probabilities, i.e. at probability higher than 0.05), though the empirical correlation between these variables is computed as low [Sheskin, 2004]. So, to summarise, if the empirical correlation between two variables reported for a data set is high, our learnt graphical model should include an edge between the two nodes. But the presence of an edge between pair of nodes is not necessarily an indication of high empirical correlation between a pair of variables–as in cases where suppressor variables are involved. Guessing the effect of such suppressor variables via an examination of the scatterplots is difficult in this multivariate situation. Lastly, it is appreciated that empirical trends are only indicators as to the Gaussian-process based model of the learnt correlation structure (and the graphical model learnt thereby) given the data at hand.

As mentioned above, in this comparative exercise, $i \neq j$; $i, j = 1, \ldots, 11$. (Existence of edges to/from Z_{12}, i.e. the "quality" variable is corroborated by examining results reported in https://onlinecourses.science.psu.edu/stat857/node/225

on regressing this variable against the others). Indeed, these empirical scatterplots visually appear to suggest stronger correlations between fixed acidity and pH; residual sugar and density; free sulphur dioxide and total sulphur dioxide; density and total sulphur dioxide; density and alcohol; alcohol and density–than amongst other pairs of variables. In our learnt graphical model, these are in fact the very node pairs that are identified to have edges (at probability in excess of 0.05) between them. Also, in https://onlinecourses.science.psu.edu/stat857/node/225, the multiple and polynomial regression analysis of the predictors $X_1, \ldots, X_{11}$ on the response variable termed "quality", i.e. X_{12}, suggested the variables alcohol and volatile acidity to have maximal effect on quality; again, alcohol and volatile acidity are the two variables included in this work for further attempts at classification of the white wines in the sample (using tree-based regression and random forests). Indeed, this is corroborated in the learning of the graphical model presented above as this manifests edges between the nodes corresponding to variables: alcohol-quality, and volatile acidity-quality.

3.5.2 Results given data $\mathbf{D}_S^{(red)}$

The $\mathbf{D}_S^{(red)}$ data is the standardised version of a subset of the original red wine data set $\mathbf{D}_{orig}^{(red)}$. $\mathbf{D}_S^{(red)}$ comprises $n = 300$ rows and $p = 12$. Thus, the $\mathbf{D}_S^{(red)}$ data comprises n measurements of the p-dimensional observable vector $(Z_1, \ldots, Z_p)^T$, where Z_i is the standardised X_i, where X_i has been described above in Section 3.2; $i = 1, \ldots, p = 12$. As with the white wine data, we implement data $\mathbf{D}_S^{(red)}$ in a Metropolis-with 2-block-update-based inference scheme to learn the between-column correlation matrix $\mathbf{\Sigma}_S^{(red)}$ of the red wine data, and learn the graphical model of data $\mathbf{D}_S^{(red)}$, given the learnt correlation $\mathbf{\Sigma}_S^{(red)}$.

The inferred graphical model of the red wine data is included in Figure 3.15. The

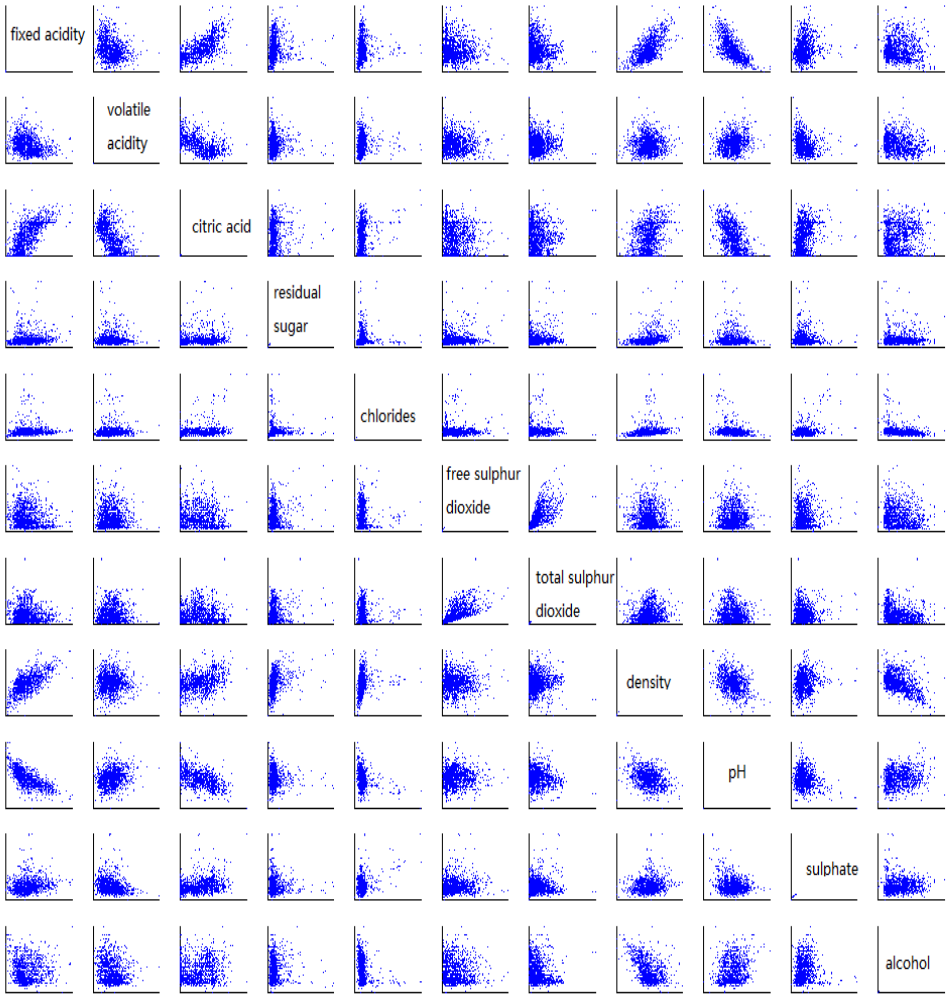

Figure 3.12: Matrix of scatterplots of the 11 different vino-chemical variables $X_1, \ldots, X_{11}$ that form the first 11 columns of the red wine data $\mathbf{D}_{orig}^{(red)}$. Here X_j is plotted against X_i, $i \neq j, i, j = 1, \ldots, 11$. The X_i relevant to the i-th row is named in the diagonal element of the i-th row; j increases from 1 to 11 from left to right.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.401
Model:	OLS	Adj. R-squared:	0.396
Method:	Least Squares	F-statistic:	89.48
Date:	Tue, 02 May 2017	Prob (F-statistic):	3.45e-141
Time:	06:53:50	Log-Likelihood:	-1914.0
No. Observations:	1350	AIC:	3850.
Df Residuals:	1339	BIC:	3907.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]	
const	-766.3330	28.797	-26.611	0.000	-822.825	-709.841
x1	-0.6843	0.041	-16.824	0.000	-0.764	-0.604
x2	-0.3570	0.202	-1.771	0.077	-0.752	0.038
x3	0.3193	0.251	1.271	0.204	-0.173	0.812
x4	-1.7821	0.703	-2.534	0.011	-3.162	-0.403
x5	0.0081	0.004	2.163	0.031	0.001	0.015
x6	0.0036	0.001	2.882	0.004	0.001	0.006
x7	781.9845	29.412	26.587	0.000	724.286	839.683
x8	-3.9047	0.299	-13.042	0.000	-4.492	-3.317
x9	-1.2234	0.185	-6.622	0.000	-1.586	-0.861
x10	0.8462	0.037	22.567	0.000	0.773	0.920

Omnibus:	1023.119	Durbin-Watson:	1.782
Prob(Omnibus):	0.000	Jarque-Bera (JB):	25392.413
Skew:	3.290	Prob(JB):	0.00
Kurtosis:	23.202	Cond. No.	9.18e+04

Figure 3.13: Output of ordinary least square analysis of regressing residual sugar on the other 10 vino-chemical attributes in the red wine data. Here X_1 is the notation used in the MATLAB output table for "fixed acidity", X_2 for "volatile acidity", X_3 for "citric acid", X_4 for "residual sugar", X_5 for "chlorides", X_6 for "free sulphur dioxide", X_7 for "total sulphur dioxide", X_8 for "pH", X_9 for "sulphate", X_{10} for "alcohol".

OLS Regression Results

Dep. Variable:	y	R-squared:	0.365
Model:	OLS	Adj. R-squared:	0.360
Method:	Least Squares	F-statistic:	76.89
Date:	Mon, 01 May 2017	Prob (F-statistic):	1.74e-124
Time:	02:14:11	Log-Likelihood:	-1318.0
No. Observations:	1350	AIC:	2658.
Df Residuals:	1339	BIC:	2715.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf.	Int.]
const	4.2567	0.653	6.523	0.000	2.977	5.537
x1	0.0059	0.018	0.335	0.738	-0.029	0.041
x2	-1.0993	0.128	-8.558	0.000	-1.351	-0.847
x3	-0.1662	0.162	-1.029	0.304	-0.483	0.151
x4	-0.0013	0.014	-0.091	0.927	-0.029	0.027
x5	-1.7190	0.449	-3.832	0.000	-2.599	-0.839
x6	0.0033	0.002	1.371	0.171	-0.001	0.008
x7	-0.0035	0.001	-4.319	0.000	-0.005	-0.002
x8	-0.4105	0.167	-2.463	0.014	-0.738	-0.084
x9	0.8068	0.118	6.855	0.000	0.576	1.038
x10	0.2939	0.018	15.975	0.000	0.258	0.330

Omnibus:	20.370	Durbin-Watson:	1.761
Prob(Omnibus):	0.000	Jarque-Bera (JB):	28.835
Skew:	-0.161	Prob(JB):	5.48e-07
Kurtosis:	3.640	Cond. No.	2.40e+03

Here X1 to X10 are: 1'fixed acidity',2'volatile acidity',3'citric acid',4:'residual sugar',5:'chlorides',6:'free sulphur dioxide',7:'total sulphur dioxide',8:'pH',9:'sulphate',10:'alcohol'

Figure 3.14: Output of ordinary least square analysis of regressing quality on the vino-chemical attributes of red wine samples in the red wine data.

marginal posterior of some of the partial correlation parameters ρ_{ij} computed using the elements of the correlation matrix $\mathbf{\Sigma}_S^{(red)}$ that is updated in the first block of Metropolis-with 2-block-update, are presented in Figure 3.16. In the second block, the edge parameters

G_{ij} of the graph $\mathbb{G}(p, \boldsymbol{R})$ are updated, given the newly updated $\boldsymbol{R}$. Figure 3.17 presents the trace of the joint posterior probability of the G_{ij} parameters and the variance parameters σ^2_{ij} (of the Normal likelihood). The marginal of some of the variance parameters are also shown in the other panels of this figure.

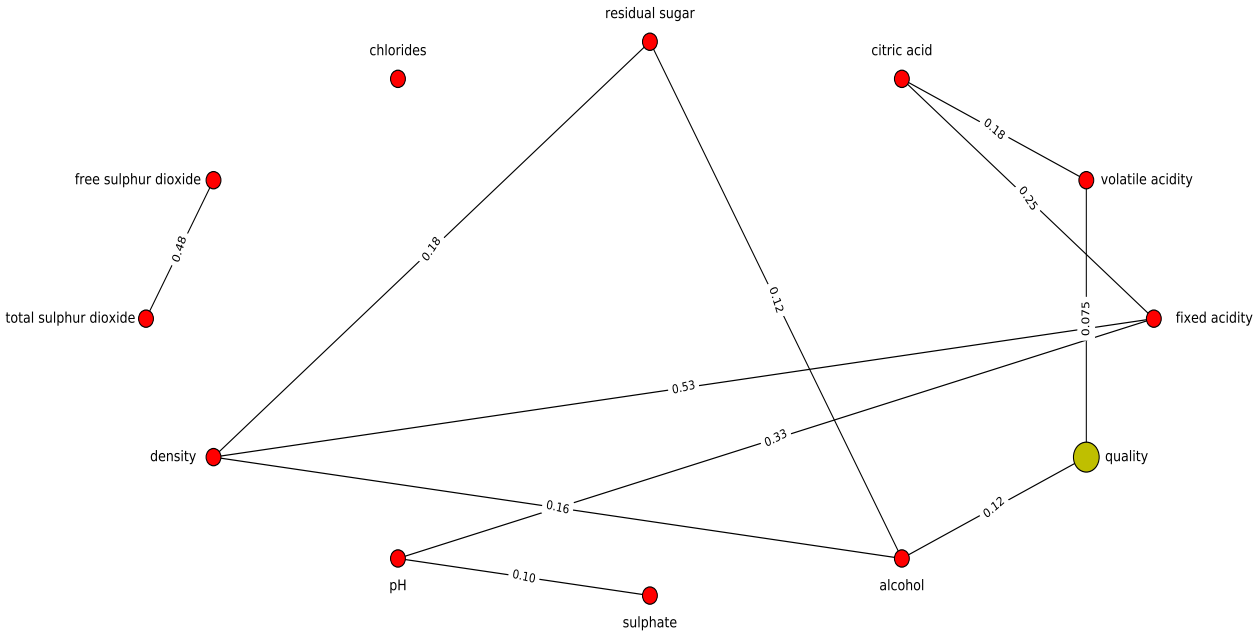

Figure 3.15: Graphical model of standardised version of the real data $\mathbf{D}_S^{(red)}$ on Portuguese red wine samples [Cortez et al., 1998]. Figure is similar to Figure 3.11, except that this is the graphical model learnt for the red wine data.

3.5.2.1 Comparing against empirical work done with red wine data

To the best of my knowledge, analysis of the red wine data has not been reported in the literature. In lieu of that, I present a matrix of pairwise scatterplots of the first 11 columns

of the red wine data in Figure 3.12. Firstly, the learnt correlations (see plots of R_{1j}, for $j = 2,\ldots,9$, as displayed in Figure 3.16), are compared to the correlations manifest in this matrix of scatterplots, to check for compatability between the learnt and empirical results. We note that all moderately correlated variable pairs, as represented in the scatterplots in Figure 3.12, are joined by edges in our learnt graphical model of the red wine data–as is to be expected if the learning of the graphical model is correct. Such pairs include fixed acidity-citric acid, fixed acidity-density, fixed acidity-pH, volatile acidity-citric acid, free sulphur dioxide-total sulphur dioxide, density-alcohol. However, an edge may exist between a pair of variables even when the apparent empirical correlation between these variables is low (see Section 3.5.1.1); this owes to the effect of other variables. Such

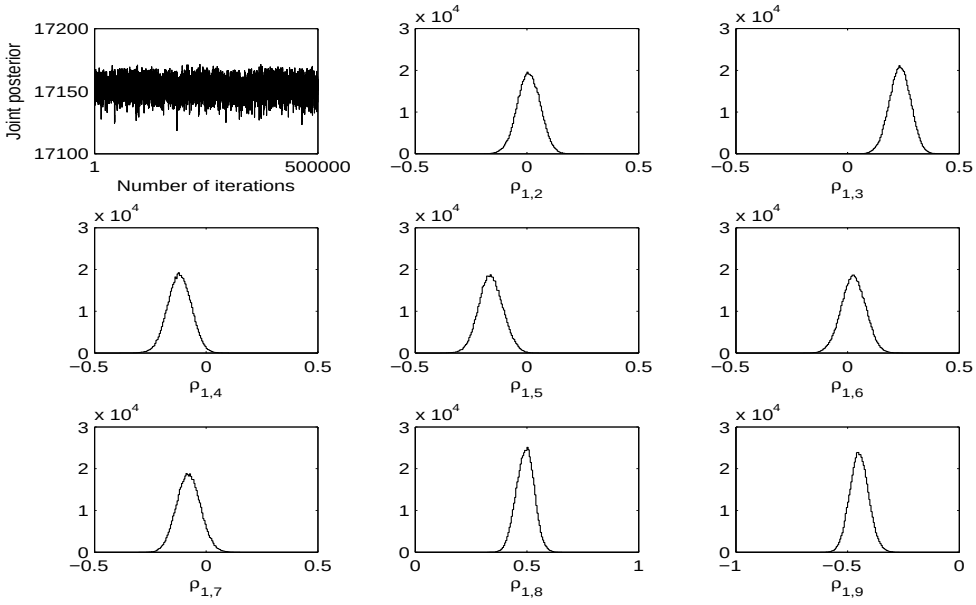

Figure 3.16: The marginal posterior of some of the partial correlation parameters ρ_{ij} computed using the elements of the correlation matrix $\mathbf{\Sigma}_S^{(red)}$ that is updated in the first block of our MCMC chain, run with the red wine data $\mathbf{D}_S^{(red)}$ of Portuguese red wine samples; $i \neq j$; $i, j = 1,\ldots,p = 12$. The top left hand panel of this figure presents the trace of the joint posterior probability density of the elements of the upper triangle of $\mathbf{\Sigma}_S^{(red)}$.

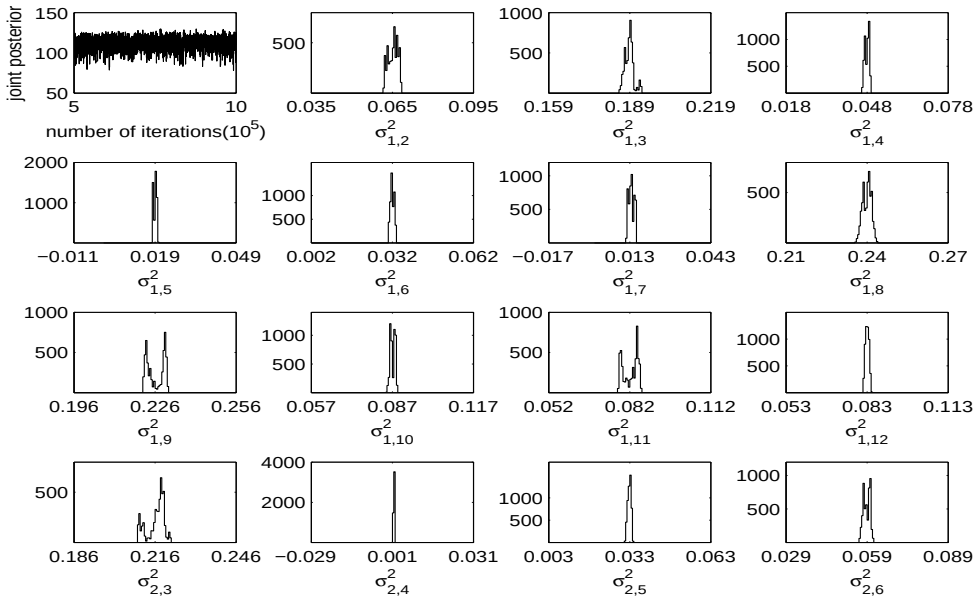

Figure 3.17: The upper panel of this figure presents the trace of the joint posterior probability of the G_{ij} parameters and the variance parameters σ^2_{ij} (of the Normal likelihood) used in this second block update of our MCMC chain, run with the red wine data $\mathbf{D}_S^{(red)}$ of Portuguese red wine samples; $i \neq j$; $i, j = 1, \ldots, p = 12$. The marginal of some of the variance parameters are also shown in the other panels of this figure.

effects give rise to the remaining edges seen in our learnt graphical model of the red wine data, namely, the edges between residual sugar-density and residual sugar-alcohol. Indeed, this is examined more closely by regressing residual sugar against the remaining 10 variables (other than quality); the results of this Ordinary Least Squares regression exercise are included in Figure 3.13. It shows that in this regression model, Z_7 (density) and Z_{10} (alcohol) affect residual sugar more than any of the other covariates–indeed, in the learnt graphical model of the red wine data, it is density and alcohol that residual sugar has edges with, respectively.

Modelling of the relationship between the response variable "quality" (Z_{12}) and the other 11 covariates (Z_1 to Z_{11}), via an OLS regression is undertaken. In this model, quality is regressed over the other vino-chemical attributes. This modelling suggests the strongest

effect of alcohol and volatile-acidity on quality (see Figure 3.14); this trend is replicated in the learnt graphical model of the red wine data.

3.6 Hellinger distance between posterior probability densities of graphs given white and red wine datasets

We are interested in answering the question of whether two multivariate datasets $\mathbf{D}_1$ and $\mathbf{D}_2$, are independent of each other or not; in effect, the question addresses the possible independence of the pdfs that the two data sets at hand are sampled from. This is of course a hard question to address when the data comprise unequal, but large number of measurements (n_1 and n_2 respectively) of two high-dimensional vector-valued observables, s.t. $\mathbf{D}_m$ comprises measurements of the standardised variable $\boldsymbol{Z}_m \in \mathcal{Z}_m \subseteq \mathbb{R}^p$, $m = 1, 2$. This is addressed by learning the graphical model of each dataset as per the methodology discussed above, and then by computing the Hellinger distance between the posterior probability density of the graphical model $\mathcal{G}_{p,\boldsymbol{\Phi}_1(\boldsymbol{R}_1)}$ of data $\mathbf{D}_1$, the between-columns partial correlation matrix of which is $\boldsymbol{R}_1^{(p\times p)}$, and the posterior of the graphical model $\mathcal{G}_{p,\boldsymbol{\Phi}_2(\boldsymbol{R}_2)}$ given the other data set, the partial correlation matrix of which is learnt to be $\boldsymbol{R}_2^{(p\times p)}$. Here $\boldsymbol{\Phi}_m(\boldsymbol{R}_m)$ is the matrix, the ij-th element of which is the edge probability $\phi_{ij}(R_{ij}) = n_{ij}$ if $n_{ij} \geq 0.05$ and $\phi_{ij}(R_{ij}) = 0$ if $n_{ij} < 0.05$. $i \neq j$; $i, j = 1, \ldots, p_m$; $m = 1, 2$. We need to consider the Hellinger distance between the posteriors of the graphical models of two data sets with the same number of columns, as this distance is defined between densities that share a common domain.

First let us remind ourselves of the Hellinger distance between two generic densities, and then clarify the computation of this distance in the context of the distance between the posterior probability densities of the edge parameters of the graphical model given a data

set, and that given another data set. Lastly, such a computation is contextualised to the graphical models learnt given the standardised versions of the real Portuguese white and red wine data sets $\mathbf{D}_S^{(white)}$ and $\mathbf{D}_S^{(red)}$ respectively, to probe the independence of these two data that report the vino-chemical attributes of the red and white wine samples.

The square of the Hellinger distance between two probability density functions $g(\cdot)$ and $h(\cdot)$ over a common domain $\mathcal{X} \in \mathbb{R}^m$, with respect to a chosen measure, is

$$\begin{aligned} D_H^2(g,f) &= \int \left(\sqrt{g(\boldsymbol{x})} - \sqrt{h(\boldsymbol{x})}\right)^2 d\boldsymbol{x} \\ &= \int g(\boldsymbol{x})d\boldsymbol{x} + \int h(\boldsymbol{x})d\boldsymbol{x} - 2\int \sqrt{g(\boldsymbol{x})}\sqrt{h(\boldsymbol{x})}d\boldsymbol{x} \\ &= 2\left(1 - \int \sqrt{g(\boldsymbol{x})}\sqrt{h(\boldsymbol{x})}d\boldsymbol{x}\right). \end{aligned} \tag{3.17}$$

Thus we see that $D_H^2(\cdot,\cdot)$ takes values in [0,2], where the value of 0 is attained when the two densities are equal, and the value of 2 is attained when the densities are singular. The Hellinger distance is closely related to the Bhattacharyya distance between two densities: $D_B(g,f) = -log\left[\int \left(\sqrt{g(\boldsymbol{x})}\sqrt{h(\boldsymbol{x})}\right)^2 d\boldsymbol{x}\right]$ [Bhattacharyya, 1943].

For the standardised wine data $\mathbf{D}_S^{(m)}$ where $m = red, white$, the posterior probability density of the graph edge parameters $G_{ij}^{(m)}$ is $\pi(G_{11}^{(m)}, G_{12}^{(m)}, \ldots, G_{p\,p-1}^{(m)} | \boldsymbol{R}_m)$, where the partial correlation matrix $\boldsymbol{R}_m$ is learnt, given this data. Indeed, during the second block update of the Metropolis-with 2-block-update inference, the value of the joint posterior probability of all the G_{ij} and σ_{ij}^2 parameters is computed, given the partial correlation matrix– that is updated given the data during the first block update. So we marginalise the σ_{ij}^2 out of this joint posterior of all edge and variance parameters, i.e. marginalise out all σ_{ij}^2 for all $i,j = 1,\ldots,p\; i \neq j$, to achieve the joint posterior probability density of the graph edge parameters given the partial correlation matrix of the data at hand.

So, at the end of the t-th iteration, value of the posterior $\pi(G_{11}^{(mt)}, G_{12}^{(mt)}, \ldots, G_{p\,p-1}^{(mt)} | \boldsymbol{R}_{mt})$,

$t = 0, \ldots, N_{iter}$ is computed. Given the availability of the posterior at discrete points in its support, implementation of the integral of the relevant posterior probability density in the definition of the Hellinger distance is replaced by the discretised version of this definition. So the square of the Hellinger distance $D_H^2(white, red)$ between the posterior probability densities

$$p_{white} := \pi(G_{11}^{(white)}, G_{12}^{(white)}, \ldots, G_{p\ p-1}^{(white)} | \boldsymbol{R}_{white})$$

and

$$p_{red} := \pi(G_{11}^{(red)}, G_{12}^{(red)}, \ldots, G_{p\ p-1}^{(red)} | \boldsymbol{R}_{red})$$

is discretised as

$$D_H^2(p_{white}, p_{red}) = \frac{\sum_{t=N_{burnin}+1}^{N_{iter}} \left(\sqrt{p_{white}^{(t)}} - \sqrt{p_{red}^{(t)}} \right)^2}{N_{iter} - N_{burnin}}, \tag{3.18}$$

where for the m-th data set, $m = white, red$, $p_m^{(t)}$ is the value of the posterior of the graph edge parameters given the partial correlation matrix, in the t-th iteration, and only post-burnin posterior samples are considered for the computation of the Hellinger distance. In other words, $p_m^{(t)} := \pi(G_{11}^{(mt)}, G_{12}^{(mt)}, \ldots, G_{p\ p-1}^{(mt)} | \boldsymbol{R}_{mt})$. The Bhattacharyya distance can be similarly discretised.

However, MCMC does not provide normalised posterior probability densities–as uniform priors on the variance parameters are employed, the marginalised posterior probability of the edge parameters is known only up to an unknown scale. In fact, what is recorded at the end of the t-th iteration, is the logarithm $\ln(p_m^{(t)})$ of the un-normalised posterior of the edges of the graph given the m-th data ($m = red,\ white$). Hence the Hellinger distance between the red and white wine graphs that is computed is only known upto a constant normalisation S that scales $p_{white}^{(t)}$ and $p_{red}^{(t)}$, $\forall\ t = 0, \ldots, N_{iter}$. Let this scale parameter S,

be chosen to ensure that the scaled, log posterior of the graph in the t-th iteration, is easily exponentiable, as in $\exp\left(\frac{\ln(p_m^{(t)})}{s}\right)$. One way of achieving this is to choose the global scale S as: $\max\{(\ln(p_{red}^{(0)}), \ln(p_{red}^{(1)}), \ldots, \ln(p_{red}^{(N_{iter})}), \ln(p_{white}^{(0)}), \ldots, \ln(p_{red}^{(N_{iter})})\}$. Indeed, this definition yields the value of the global scale S to be $s = \ln(p_{red}^{(1474)}) \approx 142.7687$; we then use $\exp(\ln(p_m^{(t)})/s)$ in Equation 3.18. The computed Hellinger distance will of course be affected by the global scaling parameter that is uses. The Bhattacharyya distance can be similarly computed, using the (similarly scaled) logarithm of the posterior values as obtained from MCMC; again the Bhattacharyya distance is unnormalised given the incompleteness of knowledge of the posterior of the graph at any iteration.

Alternatively, a (discretised version of the) odds ratio of unscaled logarithm of the unnormalised posterior densities of the graphical models can be defined, where the said graphical models are learnt using MCMC, given the two real wine data sets, as

$$\int (\log(g(\boldsymbol{x})) - \log(h(\boldsymbol{x}))) \, d\boldsymbol{x}$$

; such is then a divergence measure that is defined as

$$O_\pi(p_{white}, p_{red}) := \sum_{t=N_{burnin}+1}^{N_{iter}} \left[\log(p_{white}^{(t)}) - \log(p_{red}^{(t)})\right]. \tag{3.19}$$

Then scaling the log posterior given either data set, at any iteration, by the scale value of s=142.7687 approximately–which is the maximal value of the log posterior of the graph in the 1474-th iteration, given the red wine data– $D_H(p_{white}, p_{red}) \approx 0.1153$, so that the logarithm of this value of the Hellinger distance is $\ln(0.1153) \approx -2.1602$. Similarly, using the same scale, the Bhattacharyya distance is $D_B(p_{white}, p_{red}) \approx -1.7623$, where is is recalled that this measure is a logarithm of the distance. Indeed, values of these distances are affected by our choice of the scale S.

However, what is of interest is the comparison of the ratio of the Hellinger distance between posterior probability of graphical models given two datasets, for a given choice of the scale S, to the uncertainty inherent in the graphical model of either data, as computed at that chosen S. This uncertainty inherent to the graphical model given the m-th data can be computed as the difference

$$\begin{aligned} D_{max,s}(m) \;:=\; & \max\{\exp(\ln(p_m^{(0)})/s), \exp(\ln(p_m^{(1)})/s), \ldots, \exp(\ln(p_m^{(N_{iter})})/s)\} - \\ & \min\{\exp(\ln(p_m^{(0)})/s), \exp(\ln(p_m^{(1)})/s), \ldots, \exp(\ln(p_m^{(N_{iter})})/s)\}, \end{aligned}$$

computed for this choice of S. By this definition, $D_{max,s}(m)$ provides the separation between the maximal and minimal posteriors of graphs scaled by a chosen scale S, generated in the MCMC run using the m-th data; $m = white, red$. So for a chosen scale, the ratio of the distance between the two graphical models is computed, to the uncertainty inherent in a graphical model, namely $\sqrt{D_H^2(p_{white}, p_{red})}/D_{max,s}(red)$, and compare that with $\sqrt{D_H^2(p_{white}, p_{red})}/D_{max,s}(white)$.

This comparison is depicted in the left panel of Figure 3.18 that shows that the difference $D_{max,s}(white)$ between the scaled posterior of graphs given the white wine data is about 0.0694 while $D_{max,s}(red)$ given the red wine data is about 0.05521, These values are compared to the Hellinger distance (between scaled posteriors) of about 0.1153, between graphs given the red and white wine data. Thus, $D_H(p_{red}, p_{white})$ is about $1.66 D_{max,s}(white)$ and about $2.1 D_{max,s}(red)$, i.e. the distance between the graphical models given the two data sets is higher than the internal uncertainties within the graphs inferred upon, given either data set. Then intuitively speaking, the Hellinger distance between the graphical models given the red and white wine datasets, may suggest independence of the data sets, but the question of interpretation of the computed values of distance/divergence between a pair of

graphs, cannot be properly addressed unless a test of hypothesis can be undertaken to test if the computed distance is different from 0, i.e. the two graphical models (given the real red and white wine Portuguese data sets in our work), are different.

Compared to these, the sample mean of the log odds of the posterior of the graphs generated in the post-burnin iterations, given the two data is 18.9273, which is about 1.9 times the maximal difference between the log posterior values of graphs achieved in the MCMC run with the white wine data, and about 2.4 times that for the red wine data (see Figure 3.18). Again, this suggests that the log odds as a measure of distance between the graphical models given these two wine data sets, is significantly higher than the uncertainty internal to the results for each data.

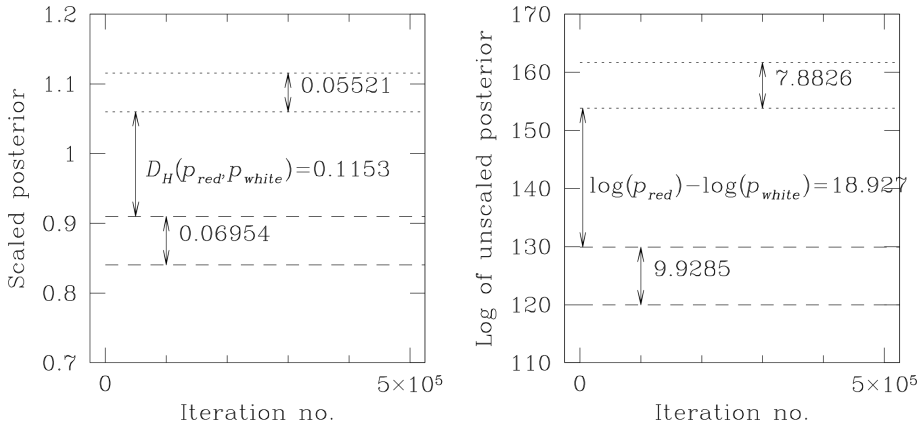

Figure 3.18: *Left*: minimum and maximum values of the scaled posterior probability density of the graph sampled in an iteration in the MCMC chain run with the red wine data, plotted in dotted lines against the number of the iteration. The difference between these values is depicted within the band delineated by these lines. The broken lines show the same for the results obtained from the MCMC chain run using the white wine data. The value of the Hellinger distance $D_H(p_{red}, p_{white})$ computed using the scaled posterior probabilities of the graphical models given the two wine data sets, is also marked, as about 0.1153. All log posterior values are scaled by a chosen global scale and exponentiated (as discussed in the text). *Right*: similar to the left panel, except that here, the ratio of the logarithm of the unscaled posteriors is used; the value of the log odds between the posteriors of the red and white wine data sets is marked to be about 18.927.

This work shows a simple and easily calculable method for comparing multiple, high-dimensional data sets, for their independence.

3.7 Learning the human disease-symptom network

The methodology for learning the graphical model of a given data, can be implemented even for a highly multivariate data, i.e. one that generates a graph with a very large number of nodes. In this section, I discuss such a graph (with $\gtrsim$8000 nodes) that describes the correlation structure of the human disease-symptom network.

3.7.1 Background

Hoehndorf et al. [2015] (HSG hereon) learn this network by considering the similarity parameter for each pair of diseases that are elements of an identified set of diseases in the Human Disease Ontology (DO), that contains information about rare and common diseases, and spans heritable, developmental, infectious and environmental diseases. Here, the "similarity parameter" between one disease and another, is computed using the ranked vectors of "normalised pointwise mutual information" (NMPI) parameters for the two diseases, where the NMPI parameter describes the relevance of a symptom (or rather, a phenotype), to the disease in question. HSG define the NMPI parameter semantically, as the normalised number of co-occurrences of a given phenotype and a disease in the titles and abstracts of 5 million articles in Medline. To do this, they make use of the Aber-OWL: Pubmed infrastructure that performs such semantical mining of the Medline abstracts and titles. The disease-disease pairwise semantic similarity parameters– computed using the degree of overlap in the relevance ranks of phenotypes associated with each disease–result in a similarity matrix, which HSG turn into a diseasedisease net-

work based on phenotypes. To do this, they only choose from the top-ranking 0.5% of disease-disease similarity values. The phenotypes associated with the diseases, and the corresponding scoring functions (such as the NPMI), exist in the file "doid2hpo-fulltext.txt.gz" at http://aber-owl.net/aber-owl/diseasephenotypes. In fact, at the site http://aber-owl.net/aber-owl/diseasephenotypes/data/, HSG have uploaded all the data that they have used. this file "doid2hpo-fulltext.txt.gz" contains information about N_{dis} diseases, and the semantic relevance of each of the N_{pheno} phenotypes to each disease, as quantified by NPMI parameter values, in addition to other scores such as t-scores and z-scores. In this file, N_{dis} is 8676 and N_{pheno} is 19323.

In the phenotypic similarity network between diseases that HSG report, diseases are the nodes, and the edge between two nodes exists in this undirected graph, if the similarity between the nodes (diseases) is in the highest-ranking 0.5% of the 38,688,400 similarity values. They remove all self-loops from the network and all nodes with a degree of 0. Their network is presented in http://aber-owl.net/aber-owl/diseasephenotypes/network/ The network analysis was performed using standard softwares and they identify multiple clusters in their network, with agglomerates of some clusters (of diseases), found to correspond to known disease-classes. The "Group Selector" function on their visualisation kit, allows for the identification of 19 such clusters in their disease-disease network, with each cluster corresponding to a disease-class. This function also allows identification of the number of diseases (i.e. nodes) in each disease-class (see left panel of Figure 3.20). The sum of the number of nodes over their identified 19 clusters, is 5059. The number of edges in their network is reported to be 65,795. The average node degree is then about 26.2. The right panel of Figure 3.20 displays the ratio of intra-class variance to the inter-class variance of each disease-class; the value of the area under the Receiver Operating

Characteristic Area Under the Curve (ROCAUC) for each cluster is opverplotted, where the ROCAUC value for the i-th cluster can be interpreted as the probability that a randomly chosen node is ranked as more likely to be in the i-th class than in the j-th class, with $i \neq j$; $i, j = 1, \ldots, 19$ [Hajian-Tilaki, 2013].

3.7.2 Learning of the disease-disease network in phenotype space using our graphical models learning

HSG's network then manifests a similarity-structure that is computed using the available NPMI parameter values. The interest here, is in learning the disease-disease graphical model, with each edge of such a graphical model learnt to exist at a learnt probability. Such learning is performed using the NPMI semantic-relevance data that is made available for each of the N_{dis} number of diseases, by HSG–this data is referred to as the human disease-phenotype data $\mathbf{D}_{DPh}$. Using $\mathbf{D}_{DPh}$, we first compute the partial correlation between any pair of diseases, for each of which, information on the ranked (semantic) relevance of each of the N_{pheno} phenotypes exist, in this given dataset. Upon computation of the pairwise partial correlations, the graphical model for the $\mathbf{D}_{DPh}$ data is learnt.

The partial correlation R_{ij} between the i-th and j-th diseases in the $\mathbf{D}_{DPh}$ data, is computed ($i, j = 1, \ldots, N_{dis}$, $i \neq j$), in the following way. The NPMI parameter values for the i-th disease and each of the N_{pheno} phenotypes are ranked, with the phenotype of the highest semantic relevance to the i-th disease assigned a rank 1. Let the rank vector of phenotypes, by semantic relevance to the i-th disease take the value $r_{\mathbf{i}}$ and similarly, the rank vector of phenotypes relevant to the j-th disease is $r_{\mathbf{j}}$. The Spearman rank correlation $s_{ij}^{(rank)}$, of vectors $r_{\mathbf{i}}$ and $r_{\mathbf{j}}$, is computed. Then we compute the partial correlation R_{ij} $\forall\, i, j = 1, \ldots, N_{dis}$; $i \neq j$, between the i-th and j-th nodes of our undirected graph, using

the computed values of the Spearman rank correlation in $\{s_{ij}^{(rank)}\}$. It is useful to define the partial correlation using the Spearman rank correlation, rather than the correlation between the vector of normalised NPMI values, since we intend to correlate the i-th disease with the j-th disease depending on how relevant a given list of phenotypes is, to each disease, i.e. depending on the ranked relevance of the phenotypes.

To learn the graphical model given this partial correlation structure in $\boldsymbol{R} = [R_{ij}]$, that is itself computed from the data $\mathbf{D}_{DPh}$), as explained in the previous sections, using an MCMC-based inference strategy, that helps us learn the edge parameters, as well as the variance of the likelihood. However, the data that we want to learn the graphical model for, is so highly multivariate–i.e. there are so many edges in the proposed graph–that we forego iterating over the multiple samples of edge and variance parameter values, and compute the graphical model for this data, by computing the posterior probability for each edge, given the computed partial correlation structure. In fact, the graphical model of data $\mathbf{D}_{DPh}$ that is presented, comprises only those edge parameters, the posterior probability of which exceeds 0.9.

Here, the posterior probability density of the edge G_{ij} (=0 or 1) between the i-th and j-th diseases, is proportional to the likelihood and prior:

$$\pi(G_{ij}|R_{ij}) \propto \ell(G_{ij}|R_{ij})\pi_0(G_{ij}),$$

where the prior on G_{ij} is $Bernoulli(0.5)$ $\forall i, j$, and the likelihood is the Normal likelihood that we chose to work with in our learning, as discussed before in Section 3.2.2, i.e. likelihood given $\boldsymbol{R} = [R_{ij}]$ is

$$\prod_{i\neq j;i,j=1}^{N_{dis}} \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left[-\frac{\left(G_{ij}-R_{ij}\right)^2}{2\sigma_{ij}^2} - \frac{\left(G_{ij}+R_{ij}\right)^2}{2\sigma_{ij}^2}\right],$$

where the variance parameters $\{\sigma_{ij}\}_{i\neq j;i,j=1}^{p}$ are defined as $\sigma_{ij}^2 = R_{ij}(1-R_{ij})$.

3.7.3 Results

The visualised graph is a sub-graph of the full graph $\mathbb{G}(N_{dis}, \boldsymbol{R})$ of data $\mathbf{D}_{DPh}$, the between-columns partial correlation matrix of which is $\boldsymbol{R} = [R_{ij}]$, $i \neq j,\ i,j = 1,\ldots,N_{dis}$, such that this visualised graph is defined to consist only of edges in the set: $\boldsymbol{E}^{/} := \{G_{ij} = 1 | \pi(G_{ij}|R_{ij}) \geq 0.9;\ i \neq j,\ i,j = 1,\ldots,N_{dis}\}$. This visualised graph has 6052 number of nodes (diseases) and 145210 edges, so that the average node degree is about 24. It is a random undirected graphical model and represents our learning of the human disease phenotype graph (displayed in Figure 3.19). Diseases belonging to the same medically-recognised disease-class are displayed in our learnt network in the same colour and symbol-type. We find in our learnt network that diseases of the same class often tend to sit in a cluster inside this network.

3.8 Conclusion

This work presents a methodology that allows for the simultaneous learning of the inter-column correlation of a rectangularly-shaped dataset, and the graphical model of such data, where this undirected graphical model comprises random edge variables that can take values of either 1 or 0.

Thus, the between-columns correlation matrix and the graph of the data, are both treated as random variables, and learnt within a Metropolis-with 2-block-update inference scheme in which the correlation matrix is first updated given the data, and the graph is then updated at the freshly updated correlation, without requiring to resort to the assumption of decomposability. We marginalise over all between-row correlation matrices, to achieve a closed-form likelihood for the between-column correlation matrix, given the data. The

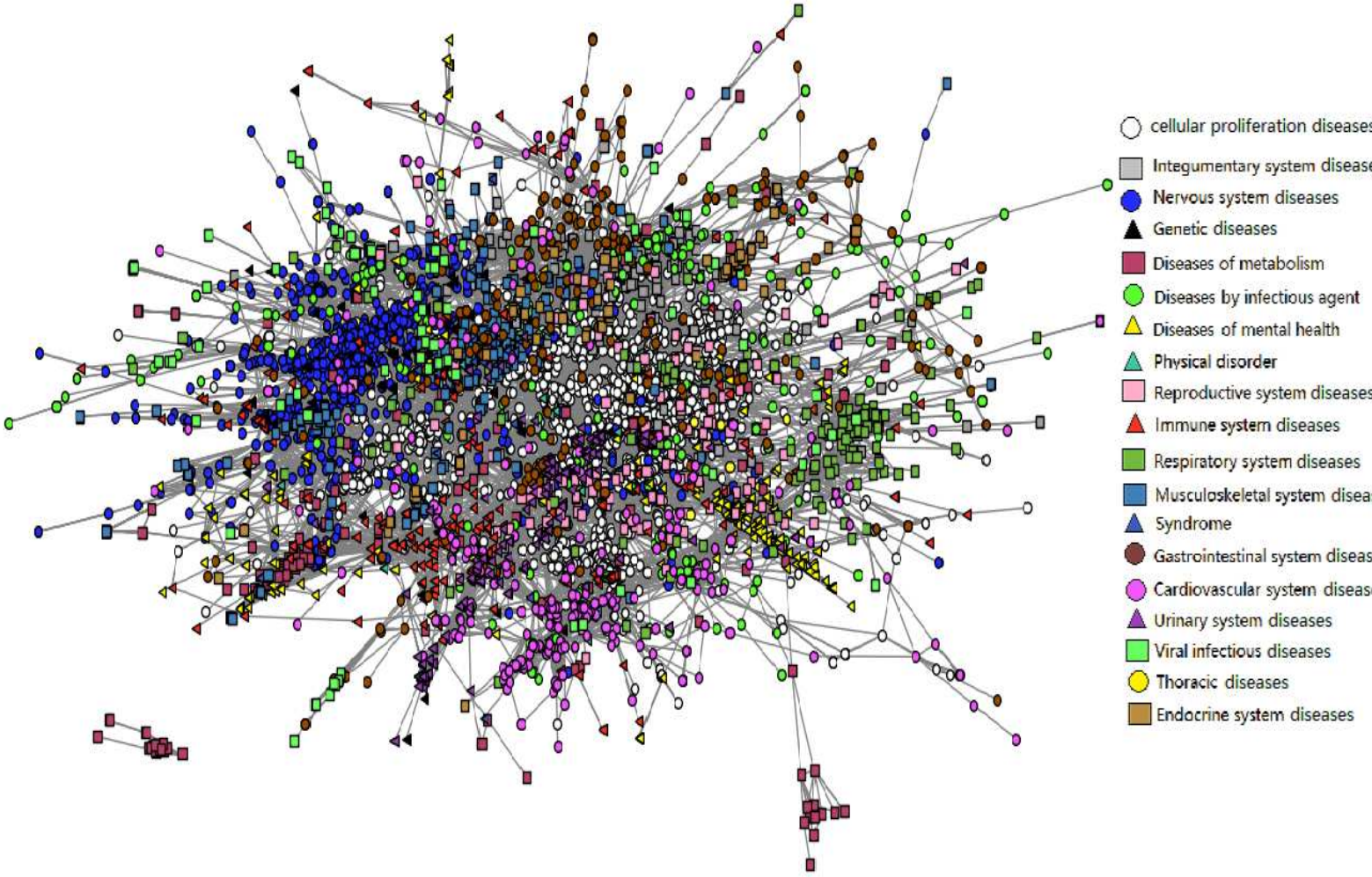

Figure 3.19: The human disease phenotype graphical model that we learn using the disease-disease partial correlation obtained using the computed Spearman rank correlation between the rank vectors of a list of phenotypes, where the phenotype ranking reflects semantic relevance of a phenotype to the disease in question (quantified by HSG as the NPMI parameter in the $\mathbf{D}_{DPh}$ dataset). Only edges with posterior probability $\geq$ 0.9 are included in this graph, and nodes that have edges with posterior less than 0.9, are discarded, resulting in 6052 diseases (nodes) remaining in this graph. There are 145210 edges in the displayed graph. All diseases identified by name by HSG, to belong to one of the 19 given disease class, are presented above in the same colour; the colour key identifying these classes, is attached. To draw the graph, we used a Python-based code that implements the Fruchterman-Reingold force-directed algorithm.

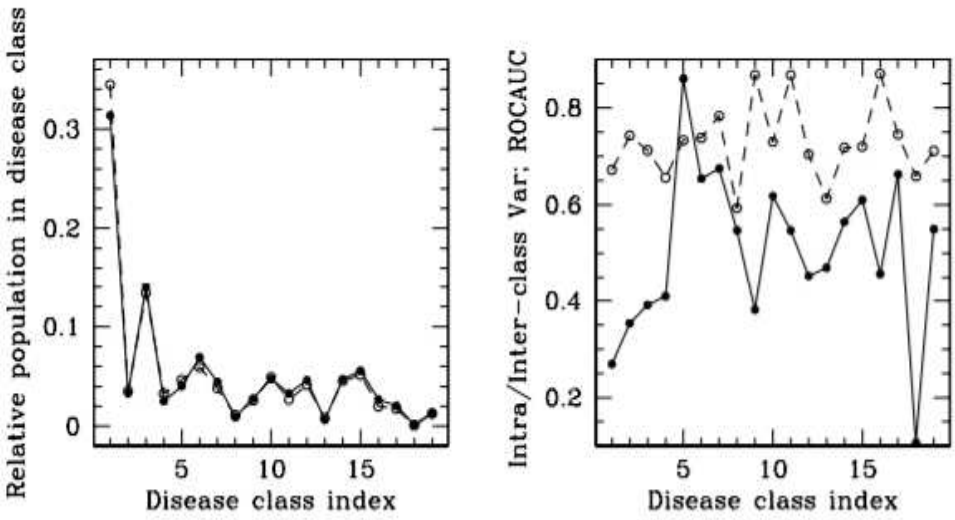

Figure 3.20: *Left:* comparison of the relative number of nodes (diseases) that are recovered in each of the 19 disease classes that HSG classify their reported network to be classified into, with the relative class-membership reported by HSG. Our results are shown as filled circles joined by solid lines. In open circles threaded by broken lines, we overplot the relative number of diseases in each of the 19 classes, as reported by HSG. Similarity of the relative populations in the different disease classes, indicate that the learnt clustering distribution is similar to that obtained by HSG. *Right:* computed ratios of the averaged intra-class to inter-class variance for each of the 19 classes, shown in filled circles; the ROC Area Under Curve values reported by HSG for each class, is overplotted as open circles joined by broken lines. The disease class indices, from assigned values of 1 to 19, are the following respectively: cellular proliferation diseases, integumentary diseases, diseases of the nervous system, genetic diseases, diseases of metabolism, diseases by infectious agents, diseases of mental health, physical disorders, diseases of the reproductive system, of the immune system, of the respiratory system, of the muscleoskeletal system, syndromes, gastrointestinal diseases, cardiovascular diseases, urinary diseases, viral infections, thoracic diseases, diseases of the endocrine system.

likelihood of an edge parameter of the graph, given the between-columns correlation, is chosen to be the Normal density with mean given by the corresponding partial correlation (computed using the updated correlation matrix) and variance that is also learnt.

Consequently, the method is capable of acknowledging errors in the measurement of any observable–repeated measurement of which comprise a column in the dataset. The ef-

fect of ignoring such existent measurement errors, on the learning of the between-columns correlation matrix, and ultimately on the graphical model of the data, is demonstrated using a simple, low-dimensional simulated dataset. Even in such a low-dimensional example, the difference made to the graph, by the inclusion of measurement errors, is clear.

In addition, the method allows for the learning of the correlation matrix and the graphical model with objective uncertainties, namely the 95% HPD credible regions. On the graphical model, these uncertainties are imposed by choosing to compose the model with only those edges, the probability of existence of which (within the post-burnin part of the MCMC run), is ≥ 0.05. Each included edge is presented with this probability of existence marked against the graph.

Upon learning the graphical model given a data set, the distance between the graphs can then be computed. This is demonstrated by computing the distance between the learnt graphical model of 11 different vino-chemical parameters of a sample of Portuguese white wines, and that of Portuguese red wines. The Hellinger distance between the posterior probability of the graphical models, given the red and white wine datasets, was expressed in units of the uncertainty in learning either graphical model. This inter-graphical-model distance is sufficiently higher than the intra-graphical-model distance, for both the red-wine and the white wine datasets, to intuitively suggest inequality of the pdfs that the red and white wine vino-chemical datasets are sampled from.

While in the learning of the correlation structure and graphical model of a given dataset, we have in general employed MCMC-based inference methods, (Metropolis-with 2-block-update, to be precise), we can avoid such inference, when faced with the task of learning very large graphs, i.e. a graphical model of a highly multivariate dataset. Such a graphical model–comprising ≥ 8000 nodes–is learnt, without resorting to MCMC-based inference.

This was the human disease-symptom network that expresses correlation between the i-th and j-th diseases based on the (rank) correlation of the vector of symptoms, where the symptoms for the i-th disease is ranked by relevance, $\forall\ i, j = 1, \ldots, N_{dis}; i \neq j$. The ranking of a given vector of symptoms was performed based on the frequency of co-occurrences of the text for that symptom, with the i-th disease, in a documented set of titles and abstracts of medical science articles. This existing text-mined information was converted into a vector of ranks for a given set of symptoms. The Spearman rank correlation is then computed between the (symptom) vector of ranks for the i-th and j-th diseases, to thereafter compute the partial correlation structure of this disease-symptom dataset. The likelihood of the ij-th edge of the sought disease-symptom graphical model was defined as Normal density with mean given by the partial correlation ρ_{ij} computed from the rank correlation between the ij-th disease pair, and variance that we fixed to $\rho_{ij}(1-\rho_{ij})$. (In contrast, in the MCMC chain we ran, this variance was treated as an unknown, and learnt). Bernoulli priors with rate 0.5 was imposed on the edge between these diseases, and the posterior probability of each edge in this graphical model was computed. Only edges with posterior in excess of 0.9 were retained in the final graphical model.

This is a very useful and practical way of learning very large networks, in real time, as long as the correlation structure is empirically known. This is often possible when the problem of learning the correlation can be cast into a semantic context–as was done in the example we consider, in learning the disease-disease correlation in terms of the associated symptoms, ordered by relevance. Other situations also admit such possibilities, for example, the product-to-product, or service-to-service correlation in terms of associated emotion, (or some other response parameter), can be semantically gleaned from the corpus of customer reviews uploaded to a chosen internet facility, and the same used to learn the

network of products/services. Importantly, this method of probabilistic learning of small to large networks, is useful for the construction of networks that evolve with time, i.e. of dynamic networks.

Chapter 4

Conclusions

4.1 Conclusions summarised

My doctoral work that is reported in this thesis is aimed at making methodological advances in the area of learning correlation structures in demanding information paradigms, using Bayesian inference. The ulterior aim is to present generic methodologies that can then be tuned to address real-world problems, and with this in mind, I illustrate the developed methodologies that I have discussed in previous chapters, on applications that use real data. thus, to summarise, My doctoral thesis results in multiple new methodologies, which are subsequently illustrated using simulated and real datasets. First, I enumerate the new methods that I have put forward in my doctoral work:

1. Modelling a discontinuous, tensor-valued function as a realisation from tensor-variate Gaussian Process that is compounded with multiple scalar-variate GPs.

2. Learning correlation matrices of general tensor-Normal likelihoods in three different ways.

3. Learning graphical model of a multivariate, rectangular-shaped data, with uncertainties.

4. Learning the graphical model of noisy,multivariate data, while acknowledging measurement noise.

5. Computing the distance between 2 learnt graphical models (given their respective dataset), using a new metric that is similar to a Hellinger metric that is normalised by the learnt uncertainty in the graphical model.

I illustrated these newly developed methodologies, on making the following applications, given real and simulated data.

1. Further work: application made to learn the location of the Sun in the Milky Way disc, after modelling the relationship between Galactic parameters (i.e. solar location), and matrix-valued measurements of velocity vectors that land in the solar neighbourhood – either in real life, as manifest in the data recorded by the *Hipparcos* satellite, or in astronomical simulations in which a bunch of stars were allowed to evolve in the Milky Way disk from a primordial time, and velocity vectors of the stars that land at a design solar location, are collated into the velocity matrix for that (designed) value of the solar location.

2. Application made to learn the with-uncertainty graphical model of a small, toy dataset, where the graphical model learning takes measurement uncertainty into account.

3. Application made to learn the graphical models of vino-chemical datasets of red and white Portuguese wine samples. Distance between the learnt graphical models is computed.

4. Application made to learn the large human disease-disease network, in phenotype space.

There are multiple areas of my doctoral work that could be improved upon. As always, my work motivates multiple other strands that I would like to explore in the future.

1. Firstly, I would like to illustrate our very potent methodology for learning challenging correlation structure of high-dimensional data, on a higher dimensional application than the one I worked on in my doctoral thesis. I hope to address an application that is marked by a diverse correlations between the multiple k-th ordered tensor-valued slices of the $k+1$-th ordered tensor-valued observable (measurements of which comprise the hyper-cuboidally shaped dataset). Such an application would be a bigger challenge.

2. The method can be implemented to learn correlations in high-dimensional time series data, with temporally evolving correlation structure, to then undertake forecasting.

3. I would very much like to undertake the learning of the correlation between the sought pair of graphical models of respective rectangularly-shaped data, where the 2 datasets in question have been realised at tow different time points. Thus, we are considering the case of a multivariate, rectangularly-shaped dataset that is temporally evolving, and I learn the temporally evolving graphical models of such data realised at any 2 time points. The distance between the of such graphical models realised at the current, and current-but-one times, then tracks the evolving nature of the (partial) correlation of the data.

4. Ultimately, such graphical models learning can be extended to with-uncertainty learning of directed graphs. This could be pursued using relative causal effects, in addition to invoking the simultaneously learnt partial correlation matrix that results in

the Bayesian learning of with-uncertainty undirected graphical model, that I have undertaken here.

Bibliography

Abramowitz, M. and Stegun, I. A. [1964], *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, Vol. 55, Courier Corporation.

Airoldi, E. M. [2007], 'Getting started in probabilistic graphical models', *PLoS Computational Biology* **3**(12), e252.

Albert, J. H. and Chib, S. [1993], 'Bayesian analysis of binary and polychotomous response data', *Journal of the American statistical Association* **88**(422), 669–679.

Aston, J. A. D. and Kirch, C. [2012], 'Evaluating stationarity via change-point alternatives with applications to fmri data', *Annals of Applied Statistics* **6**(4), 1906–1948.

Bandyopadhyay, D. and Canale, A. [2016], 'Sparse multi-dimensional graphical models: A unified bayesian framework', *Journal of Rotyal Statistical society Series C* **65**(4), 619–640.

Banerjee, S., Basu, A., Bhattacharya, S., Bose, S., Chakrabarty, D. and Mukherjee, S. [2015], 'Minimum distance estimation of milky way model parameters and related inference', *SIAM/ASA Journal on Uncertainty Quantification* **3**(1), 91–115.

Barton, T. A. and Fuhrmann, D. R. [1993], 'Covariance structures for multidimensional data', *Multidimensional Systems and Signal Processing* **4**(2), 111–123.

Basser, P. J. and Pajevic, S. [2003], 'A normal distribution for tensor-valued random variables: applications to diffusion tensor mri', *IEEE transactions on medical imaging* **22**(7), 785–794.

Ben-Israel, A. and Greville, T. N. [2003], *Generalized inverses: theory and applications*, Vol. 15, Springer Science & Business Media.

Benner, P., Findeisen, R., Flockerzi, D., Reichl, U. and Sundmacher, K. [2014], *Large-Scale Networks in Engineering and Life Sciences*, Modeling and Simulation in Science, Engineering and Technology, Springer, Switzerland.

Berger, J. O., De Oliveira, V. and Sansó, B. [2001], 'Objective bayesian analysis of spatially correlated data', *Journal of the American Statistical Association* **96**(456), 1361–1374.

Bhattacharyya, A. [1943], 'On a measure of divergence between two statistical populations defined by their probability distributions', *Bull. Calcutta Math. Soc.* **35**, 99–109.

Biegler, L., Biros, G., Ghattas, O., Heinkenschloss, M., Keyes, D., Mallick, B., Tenorio, L., van Bloemen Waanders, B., Willcox, K. and Marzouk, Y. [2011], *Large-scale inverse problems and quantification of uncertainty*, Vol. 712, John Wiley & Sons.

Bijma, F., De Munck, J. C. and Heethaar, R. M. [2005], 'The spatiotemporal meg covariance matrix modeled as a sum of kronecker products', *NeuroImage* **27**(2), 402–415.

Binney, J. and Merrifield, M. [1998], *Galactic astronomy*, Princeton University Press.

Box, G. E. P. [1973], *Bayesian Inference in Statistical Analysis*, Wiley.

Carvalho, C. M. and West, M. [2007], 'Dynamic matrix-variate graphical models', *Bayesian Analysis* **2**(1), 69–97.

Chakrabarty, D. [2004], 'Patterns in the outer parts of galactic disks', *Monthly Notices of the Royal Astronomical Society* **352**, 427.

Chakrabarty, D. [2007], 'Phase space structure in the solar neighbourhood', *Astronomy & Astrophysics* **467**(1), 145–162.

Chakrabarty, D., Biswas, M., Bhattacharya, S. et al. [2015], 'Bayesian nonparametric estimation of milky way parameters using matrix-variate data, in a new gaussian process based method', *Electronic Journal of Statistics* **9**(1), 1378–1403.

Chari, R., Coe, B. P., Vucic, E. A., Lockwood, W. W. and Lam, W. L. [2010], 'An integrative multi-dimensional genetic and epigenetic strategy to identify aberrant genes and pathways in cancer', *BMC systems biology* **4**(1), 67.

Chari, R., Thu, K. L., Wilson, I. M., Lockwood, W. W., Lonergan, K. M., Coe, B. P., Malloff, C. A., Gazdar, A. F. et al. [2010], 'Integrating the multiple dimensions of genomic and epigenomic landscapes of cancer', *Cancer and Metastasis Reviews* **29**(1), 73–93.

Christian, R. [1994], *An overview of robust Bayesian analysis*, Springer-Verlag.

Cios, K. J., Swiniarski, R. W., Pedrycz, W. and Kurgan, L. A. [2007], Unsupervised learning: clustering, *in* 'Data Mining', Springer, pp. 257–288.

Clarke, R., Ressom, H. W., Wang, A., Xuan, J., Liu, M. C., Gehan, E. A. and Wang, Y. [2008], 'The properties of high-dimensional data spaces: implications for exploring gene and protein expression data', *Nature Reviews Cancer* **8**(1), 37.

Coates, A., Ng, A. and Lee, H. [2011], An analysis of single-layer networks in unsupervised feature learning, *in* 'Proceedings of the fourteenth international conference on artificial intelligence and statistics', pp. 215–223.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T. and Reis, J. [1998], 'Modeling wine preferences by data mining from physicochemical properties', *Decision Support Systems* **47**(4), 547–553.

Dawid, A. P. and Lauritzen, S. L. [1993], 'Hyper-Markov laws in the statistical analysis of decomposable graphical models', *Ann. Statist.* **21**(3), 1272–1317.

De Lathauwer, L., De Moor, B. and Vandewalle, J. [2000], 'A multilinear singular value decomposition', *SIAM journal on Matrix Analysis and Applications* **21**(4), 1253–1278.

Demuth, H. B., Beale, M. H., De Jess, O. and Hagan, M. T. [2014], *Neural network design*, Martin Hagan.

Dereniowski, D. and Kubale, M. [2003], Cholesky factorization of matrices in parallel and ranking of graphs, *in* 'International Conference on Parallel Processing and Applied Mathematics', Springer, pp. 985–992.

Diack, C. A. [1999], *Measures of deviation for nonparametric tests of regression*, Eurandom.

Dryden, I. L., Bai, L., Brignell, C. J. and Shen, L. [2009], 'Factored principal components analysis, with applications to face recognition', *Statistics and Computing* **19**(3), 229–238.

Duff, G. F. D. and Naylor, D. [1966], *Differential equations of applied mathematics*, John Wiley& Sons, Inc., New York-London-Sydney.

Dunstan, P. K., Foster, S. D., Hui, F. K. and Warton, D. I. [2013], 'Finite mixture of regression modeling for high-dimensional count and biomass data in ecology', *Journal of agricultural, biological, and environmental statistics* **18**(3), 357–375.

Fan, Y. [2017], Statistical Learning with Applications in High Dimensional Data and Health Care Analytics, PhD thesis, University of Maryland.

Frieze, A. and Karonski, M. [2016], *Introduction to Random Graphs*, Cambridge University Press, Cambridge.

Fu, X. [2016], Exploring geometrical structures in high-dimensional computer vision data, PhD thesis, University of Otago.

Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. [2014], *Bayesian data analysis*, Vol. 2, Chapman & Hall/CRC Boca Raton, FL, USA.

Goldstein, M. [2006], *Subjectivism principles and practice*, Bayesian Analysis.

Goodman, L. A. [1970], 'The multivariate analysis of qualitative data: Interaction among multiple classifications', *Journal of the American Statistical Association* **65**, 226–256.

Gramacy, R. B. [2005], Bayesian treed Gaussian process models, PhD thesis, University of California.

Gregory, C. [2015], 'The computational complexity of probabilistic inference using bayesian belief networks', *Electronic Journal of Statistics* .

Gruber, L. and West, M. [2016], 'Gpu-accelerated bayesian learning and forecasting in simultaneous graphical dynamic linear models', *Bayesian Analysis* **11**(1), 125–149.

Guilleminot, J. and Soize, C. [2010], 'A stochastic model for elasticity tensors with uncertain material symmetries', *International Journal of Solids and Structures* **47**(22), 3121–3130.

Guinness, J., Fuentes, M., Hesterberg, D. and Polizzotto, M. [2014], 'Multivariate spatial modeling of conditional dependence in microscale soil elemental composition data', *Spatial Statistics* **9**, 93–108.

Hajian-Tilaki, K. [2013], 'Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation', *Caspian Journal of Internal Medicine* **4**(2), 627–635.

Hartigan, J. A. and Hartigan, J. [1975], *Clustering algorithms*, Vol. 209, Wiley New York.

Hastie, T. J. and Tibshirani, R. J. [1990], *Generalized additive models*, Vol. 43, CRC press.

Hastie, T., Tibshirani, R. and Friedman, J. [2009], Unsupervised learning, *in* 'The elements of statistical learning', Springer, pp. 485–585.

Heinonen, M., Mannerström, H., Rousu, J., Kaski, S. and Lähdesmäki, H. [2016], Non-stationary gaussian process regression with hamiltonian monte carlo, *in* 'Artificial Intelligence and Statistics', pp. 732–740.

Hensman, J., Fusi, N. and Lawrence, N. D. [2013], 'Gaussian processes for big data', *arXiv preprint arXiv:1309.6835* .

Higham, N. J. [1990], *Analysis of the Cholesky decomposition of a semi-definite matrix*, Oxford University Press.

Hoehndorf, R., Schofield, P. N. and Gkoutos, G. V. [2015], 'Analysis of the human diseasome using phenotype similarity between common, genetic, and infectious diseases', *Scientific Reports* **5**(10888).

Hoff, K. [1997], 'Bayesian learning in an infant industry model', *Journal of International Economics* **43**(3-4), 409–436.

Hoff, P. D. et al. [2011], 'Separable covariance arrays via the tucker product, with applications to multivariate relational data', *Bayesian Analysis* **6**(2), 179–196.

Hofmann, T., Schölkopf, B. and Smola, A. J. [2008], 'Kernel methods in machine learning', *The annals of statistics* pp. 1171–1220.

Jacobsen, M. et al. [1996], 'Laplace and the origin of the ornstein-uhlenbeck process', *Bernoulli* **2**(3), 271–286.

Kiiveri, H., Speed, T. P. and Carlin, J. B. [1984], 'Recursive causal models', *Journal of the Australian Mathematical Society* **36**(Ser.A), 30–52.

Knuth, D. [1997], *The Art of Computer Programming: Seminumerical Algorithms*, Addison-Wesley Longman Publishing Co., Boston, MA, USA.

Koistinen, P. and Holmström, L. [1992], Kernel regression and backpropagation training with noise, *in* 'Advances in Neural Information Processing Systems', pp. 1033–1039.

Kolda, T. G. and Bader, B. W. [2009], 'Tensor decompositions and applications', *SIAM Review* **51**(3), 455–500.

Krishna B. Athreya, S. N. L. [2006], *Measure Theory and Probability Theory*, Springer, New York.

Krishnamoorthy, A. and Menon, D. [2013], Matrix inversion using cholesky decomposition, *in* 'Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2013', IEEE, pp. 70–72.

Lauritzen, S. L. [1996], *Graphical Models*, Oxford University Press, Oxford, UK.

Lawrence, N. D. [2003], Gaussian process latent variable models for visualisation of high dimensional data., *in* 'Nips', Vol. 2, p. 5.

Leitao, P. J., Schwieder, M., Suess, S., Catry, I., Milton, E. J., Moreira, F., Osborne, P. E., Pinto, M. J., van der Linden, S. and Hostert, P. [2015], 'Mapping beta diversity from space: Sparse generalised dissimilarity modelling (sgdm) for analysing high-dimensional data', *Methods in Ecology and Evolution* **6**(7), 764–771.

Ley, C. P. [2016], 'Gaussian process regression', *Training* **2**, 16.

Liu, K. [1988], 'Measurement error and its impact on partial correlation and multiple linear regression analyses', *Americal Jl.of Epidemiology* **127**(4), 864–874.

MacKay, D. J. [1998], 'Introduction to gaussian processes', *NATO ASI Series F Computer and Systems Sciences* **168**, 133–166.

Madigan, D. and Raftery, A. E. [1994], 'Model selection and accounting for model uncertainty in graphical models using occam's window', *Journal of the American Statistical Association* **89**(428), 1535–1546.

Manceur, A. M. and Dutilleul, P. [2013], 'Maximum likelihood estimation for the tensor normal distribution: Algorithm, minimum sample size, and empirical bias and dispersion', *Journal of Computational and Applied Mathematics* **239**, 37 – 49.

Mardia, K. V. and Goodall, C. R. [1993], *Spatial-temporal analysis of multivariate environmental monitoring data*, Vol. 6, Elsevier New York.

Mathai, A. M. and G.Pederzoli [1997], *Some Properties of Matrix-Variate Laplace Transforms and Matrix-Variate Whittaker Functions*, number 253, Elsevier Science, New York.

Matusita, K. [1953], 'On the estimation by the minimum distance method', *Annals of the Institute of Statistical Mathematics* **5**(1), 59–65.

McCullagh, P. [1987], *Tensor Methods in Statistics*, Chapman and Hall.

Mitchell, T. J. and Beauchamp, J. J. [1988], 'Bayesian variable selection in linear regression', *Journal of the American Statistical Association* **83**(404), 1023–1032.

Nasrabadi, N. M. [2007], 'Pattern recognition and machine learning', *Journal of electronic imaging* **16**(4), 049901.

Neal, H. and Nayfeh, A. [1990], 'Response of a single-degree-of-freedom system to a non-stationary principal parametric excitation', *International Journal of Non-Linear Mechanics* **25**(2-3), 275–284.

Neal, R. M. [2012], *Bayesian learning for neural networks*, Vol. 118, Springer Science & Business Media.

Ni, Y., Stingo, F. C. and Baladandayuthapani, V. [2017], 'Sparse multi-dimensional graphical models: A unified bayesian framework', *Journal of the American Statistical Association* **112**(518), 779–793.

Oberg, A. L., McKinney, B. A., Schaid, D. J., Pankratz, V. S., Kennedy, R. B. and Poland, G. A. [2015], 'Lessons learned in the analysis of high-dimensional data in vaccinomics', *Vaccine* **33**(40), 5262–5270.

Oseledets, I. V. [2011], 'Tensor-train decomposition', *SIAM Journal on Scientific Computing* **33**(5), 2295–2317.

Paciorek, C. J. and Schervish, M. J. [2004], Nonstationary covariance functions for gaussian process regression, *in* 'Advances in neural information processing systems', pp. 273–280.

Pang, Y. H., Khor, E. Y. and Ooi, S. Y. [2016], Biometric access control with high dimensional facial features, *in* 'Australasian Conference on Information Security and Privacy', Springer, pp. 437–445.

Park, S. and Choi, S. [2010], Hierarchical gaussian process regression, *in* 'Proceedings of 2nd Asian Conference on Machine Learning', pp. 95–110.

Qiang, Q. and Fei, Z. [2011], Generation of facial gesture and expression in high-dimensional space, *in* '2011 International Conference on Internet Technology and Applications', pp. 1–5.

Raftery, A. E., Madigan, D. and Hoeting, J. A. [1997], 'Bayesian model averaging for linear regression models', *Journal of the American Statistical Association* **92**(437), 179–191.

Rasmussen, C. E. and Williams, C. K. I. [2006], *Gaussian Processes for Machine Learning*, The MIT Press, MIT.

Revuz, D. and Yor, M. [2013], *Continuous martingales and Brownian motion*, Vol. 293, Springer Science & Business Media.

Richter, A., Salmi, J. and Koivunen, V. [2008], Ml estimation of covariance matrix for tensor valued signals in noise, *in* 'Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on', IEEE, pp. 2349–2352.

Robert, C. P. and Casella, G. [2004], *Monte Carlo Statistical Methods*, Springer-Verlag, New York.

Rowley, H. A., Baluja, S. and Kanade, T. [1998], 'Neural network-based face detection', *IEEE Transactions on pattern analysis and machine intelligence* **20**(1), 23–38.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A. and Nolan, G. P. [2005], 'Causal protein-signaling networks derived from multiparameter single-cell data', *Science* **308**(5721), 523–529.

Sampson, P. D. and Guttorp, P. [1992], 'Nonparametric estimation of nonstationary spatial covariance structure', *Journal of the American Statistical Association* **87**(417), 108–119.

Sarkar, C. [2015], Improving Predictive Modeling in High Dimensional, Heterogeneous and Sparse Health Care Data, PhD thesis, University of Minnesota.

Schmidt, A. M. and O'Hagan, A. [2003], 'Bayesian inference for non-stationary spatial covariance structure via spatial deformations', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **65**(3), 743–758.

Sebah, P. and Gourdon, X. [2002], 'Introduction to the gamma function', *American Journal of Scientific Research* .

Seber, G. A. and Lee, A. J. [2012], *Linear regression analysis*, Vol. 936, John Wiley & Sons.

Seo, S., Wallat, M., Graepel, T. and Obermayer, K. [2000], Gaussian process regression: Active data selection and test point rejection, *in* 'Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on', Vol. 3, IEEE, pp. 241–246.

Sheskin, D. [2004], *Handbok of parametric and nonparametric statistical procedures*, Chapman & Hall/CRC, Boca Raton, Florida.

Shinozuka, M. and Deodatis, G. [1991], 'Simulation of stochastic processes by spectral representation', *Applied Mechanics Reviews* **44**(4), 191–204.

Signoretto, M., De Lathauwer, L. and Suykens, J. A. [2011], 'A kernel-based framework to tensorial data analysis', *Neural networks* **24**(8), 861–874.

Snoek, J., Swersky, K., Zemel, R. and Adams, R. [2014], Input warping for bayesian optimization of non-stationary functions, *in* 'International Conference on Machine Learning', pp. 1674–1682.

Specht, D. F. [1991], 'A general regression neural network', *IEEE transactions on neural networks* **2**(6), 568–576.

Theobald, D. L. and Wuttke, D. S. [2008], 'Accurate structural correlations from maximum likelihood superpositions', *PLoS computational biology* **4**(2), e43.

Tibshirani, R. [2014], *An Introduction to Statistical Learning*, Springer Publishing Company, Incorporated.

Tolvanen, V., Jylänki, P. and Vehtari, A. [2014], Expectation propagation for nonstationary heteroscedastic gaussian process regression, *in* 'Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on', IEEE, pp. 1–6.

Vaughn, B. K. [2008], 'Data analysis using regression and multilevel/hierarchical models, by gelman, a., & hill, j.', *Journal of Educational Measurement* **45**(1), 94–97.

Veraart, J., Sijbers, J., Sunaert, S., Leemans, A. and Jeurissen, B. [2013], 'Weighted linear least squares estimation of diffusion mri parameters: strengths, limitations, and pitfalls', *NeuroImage* **81**, 335–346.

Vovk, V. [2013], Kernel ridge regression, *in* 'Empirical inference', Springer, pp. 105–116.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S. et al. [2001], Constrained k-means clustering with background knowledge, *in* 'ICML', Vol. 1, pp. 577–584.

Wang, H. and West, M. [2009], 'Bayesian analysis of matrix normal graphical models', *Biometrika* **96**, 821–834.

Wang, J. [2011], *Geometric structure of high-dimensional data and dimensionality reduction*, Springer.

Wang, W., Chen, L. and Zhang, Q. [2015], 'Outsourcing high-dimensional healthcare data to cloud with personalized privacy preservation', *Computer Networks* **88**, 136–148.

Warton, D. I. [2011], 'Regularized sandwich estimators for analysis of high-dimensional data using generalized estimating equations', *Biometrics* **67**(1), 116–123.

Weber, M., Welling, M. and Perona, P. [2000], 'Unsupervised learning of models for recognition', *Computer Vision-ECCV 2000* pp. 18–32.

Werner, K., Jansson, M. and Stoica, P. [2008], 'On estimation of covariance matrices with kronecker product structure', *IEEE Transactions on Signal Processing* **56**(2), 478–491.

Whittaker, J. [2008], *Graphical Models in Applied Multivariate Statistics*, Wiley, Switzerland.

Wilson, A. G. and Ghahramani, Z. [2010], 'Generalised wishart processes', *arXiv preprint arXiv:1101.0240* .

Wothke, W. [1993], *Nonpositive definite matrices in structural modeling*, Sage, Newbury Park, CA.

Xu, Z. and Yan, F. [2015], 'Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis', *arXiv:1108.6296* .

Zhao, Q., Zhou, G., Zhang, L. and Cichocki, A. [2014], Tensor-variate gaussian processes regression and its application to video surveillance, *in* 'Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on', IEEE, pp. 1265–1269.

Appendix A

Splitting methods

As part of an initial numerical warm-up exercise, I undertook the writing of a programme (in C++) that serves as the main code that implements the method that we have written about and submitted a paper on arxiv (Chakrabartty, Wang & Chakrabarty, 2015). In this exercise, we try to develop a method to find two parallel sub-groups of a given data set that is multivariate and big. By "parallel" sub-groups, we mean subsets of data that have the same mean and nearly the same variance as each other. I describe the context of these "groups" in the following subsection. In one of our implementations, the observed data set is a 912×50-dimensional matrix and we allot the elements of each row to either one sub-group or another, so that at the end of splitting the whole observed data set, we end up with two sub-groups with the same mean and nearly same variance.

Introduction to the underlying problem

Examinee ability is measured by the scores obtained by an examinee in a test designed to assess such ability. As with all other measurements, this ability measurement too is fundamentally uncertain or inclusive of errors. In Classical Test Theory, quantification of the complementary certainty, or reliability of a test, is defined as the ratio of the true score

variance and the observed score variance i.e. reliability is defined as the proportion of observed test score variance that is attributable to true score. Here, the observed score is treated as inclusive of the measurement error. This theoretical definition notwithstanding, there are different methods of obtaining reliability in practice, and problems arise from the implementation of these different techniques even for the same test. Importantly, it is to be noted that the different methods of estimating reliability coefficients differently from the aforementioned classical or theoretical definition of reliability. This can potentially result in different estimates of reliability of a particular test even for the same examinee sample. As a result, a method of obtaining reliability of a test under its theoretical definition is needed.

Calculating Reliability

According to the classical definition, the test score X is the true score (T) added to the error (e). As a result, the reliability of a test (r_{tt}) is defined classically as the ratio of the variance of true test score (S_T^2) and the variance of observed test score (S_X^2), i.e.

$$r_{tt} = \frac{S_T^2}{S_X^2}$$

This ratio also can be calculated by the variance of error score(S_e^2) and the variance of observed data set:

$$r_{tt} = \frac{S_T^2}{S_X^2} = 1 - \frac{(S_e)^2}{S_X^2} \tag{2.1}$$

The reliability measurement is then very difficult to calculate since we do not know either S_e or S_T. In our method, we focus on finding the variance of the error (S_e^2). We split the "test" into two "parallel" halves, where "parallel" implies equal sum of scores, which results in nearly equal variances of scores. By "test" above, we imply the matrix of scores

(either 1 or 0) obtained by each of the N examinees in each of the n multiple-choice questions of the test. The data matrix is then $N \times n$. We choose half the scores of the i-th examinee to be the i-th row of a $N \times (n/2)$ data matrix and other scores obtained by the same examinee to be the i-th row of another $N \times (n/2)$ data matrix. We refer to these data matrices as the g-th and h-th sub-tests. The observed score (X_i) for i-th examinee is the sum of the scores $X_i^{(g)}$ and $X_i^{(h)}$ in i-th rows of the two sub-tests. Thus, when splitting into two subgroups, $X_i^{(g)}$ and $X_i^{(h)}$ imply the observed scores for i-th examinee in the two subgroups(g and h).

The observed score($\{X_i\}_{i=1}^N$) can be regarded as the sum of true score($\{T_i\}_{i=1}^N$) and error score($\{E_i\}_{i=1}^N$). In each parallel subgroup, we can get equation below for any examinee:

$$X_i^{(g)} = T_i^{(g)} + E_i^{(g)}, X_i^{(h)} = T_i^{(h)} + E_i^{(h)}$$

The true score in each group is a function for the exam ability. The same examinee will maintain the same exam ability for all questions (or items) of the test (assumed). When split into two parallel subgroups ($\{X_i^{(g)}\}_{i=1}^N$ and $\{X_i^{(h)}\}_{i=1}^N$), the true score of any examinees in each subgroup will be same($T_i^{(g)} = T_i^{(h)}$). Thus, we can get $X_i^{(h)} - X_i^{(g)} = E_i^{(h)} - E_i^{(g)}$. We can write down the variance of error as:

$$\| \boldsymbol{X}_g \|^2 + \| \boldsymbol{X}_h \|^2 - 2 \| \boldsymbol{X}_g \| \| \boldsymbol{X}_h \| \cos\theta_{gh}$$

$$= \| \boldsymbol{E}_g \|^2 + \| \boldsymbol{E}_h \|^2 = N(S_e^{(g)})^2 + N(S_e^{(h)})^2 = 2N(S_e^{(g)})^2 \tag{2.2}$$

Where $\boldsymbol{X}_g$ and $\boldsymbol{X}_h$ is the vector includes all the observed scores for subgroup g and h. θ_{gh} is the angle between vectors $\boldsymbol{X}_h$ and $\boldsymbol{X}_g$. $\boldsymbol{E}_h$ and $\boldsymbol{E}_g$ is the vector for error scores in subgroup g and h. We assume the error scores for examinees are independent of each other. Then the covariance for error score will be zero which implies the angle for the two error vectors is

90 degrees. We ensure that each subgroup contains the same number of questions, and we assume the error distribution for each subgroup is the same. Thus, the variance for error scores in two subgroups is equal.

Then the term $2 \parallel \boldsymbol{X}_g \parallel \parallel \boldsymbol{X}_h \parallel \cos\theta_{gh}$ can be re-written as:

$$2 \parallel \boldsymbol{X}_g \parallel \parallel \boldsymbol{X}_h \parallel \cos\theta_{gh} = 2\sum_{i=1}^{N} X_i^{(g)} X_i^{(h)} \tag{2.3}$$

Using the equation (2.2) and (2.3), we can easily calculate the variance of error score by splitting the observed data set into two parallel groups:

$$S_e^2 = \parallel \boldsymbol{X}_g \parallel^2 + \parallel \boldsymbol{X}_h \parallel^2 - 2\sum_{i=1}^{N} X_i^{(g)} X_i^{(h)}$$

As a result, the reliability of a test can be calculated using equation (2.1).

Splitting method

Chakrabartty (2011) gave a method for splitting a test into 2 parallel halves. Here we give a novel method of splitting a test into 2 parallel halves–g and h–that have nearly equal means and variances of the observed scores. The splitting is initiated by the determination of the item-wise total score for each item. So let the j-th item in the test have the item-wise score $\tau_j := \sum_{i=1}^{N} X_i^{(j)}$, where $X_i^{(j)}$ is the i-th examinee's score in the j-th item, $j = 1, \ldots, n$. Our method of splitting is as follows.

Step-I The item-wise scores are sorted in an ascending order resulting in the ordered sequence $\tau_1, \tau_2, \ldots, \tau_n$. Following this, the item with the highest total score is identified and allocated to the g-th sub-test. The item with second highest total score is then allocated to the h-th test, while the item with the third highest score is assigned to h-th test and the fourth highest to the g-th test, and so on. In other words, allocation of items is performed to ensure realisation of the following structure.

sub-test g	sub-test h	difference in item-wise scores of 2 sub-test
τ_1	τ_2	$\tau_1 - \tau_2 \geq 0$
τ_4	τ_3	$\tau_4 - \tau_3 \leq 0$
$\vdots$	$\vdots$	$\vdots$

where we assume n to be even; for tests with an odd number of items, we ignore the last item for the purposes of dichotomisation. The sub-tests obtained after the very first slotting of the sequence $\{\tau_k\}_{k=1}^{n}$ into the sub-tests, following this suggested method of distribution, is referred to as the "seed sub-tests".

Step-II Next, the difference of item-wise scores in every item of the g-th and h-th sub-tests is recorded and the sum $\mathcal{S}$ of these differences is computed (total of column 3 in the above table). If the value of $\mathcal{S}$ is zero, we terminate the process of distribution of items across the 2 sub-tests, otherwise we proceed to the next step.

Step-III We identify rows in the above table, the swapping of the entries of columns 1 and 2 of which, results in the reduction of $|\mathcal{S}|$, where $|\cdot|$ denotes absolute value. Let the row numbers of such rows be $\rho^{(\star_\ell)}$, $\ell = 1, 2, \ldots, n^{(\star)}$ where $n^{(\star)} \leq n/2$. We swap the $\rho^{(\star_\ell)}$-th item of the g-th sub-test with the $\rho^{(\star_\ell)}$-th item of the h-th sub-test and re-calculate sum of the entries of the revised g-th sub-test and h-th sub-test. If the revised value of $|\mathcal{S}|$ is zero or a number close to zero that does not reduce upon further iterations, we stop the iteration; otherwise we return to the identification of the row numbers $\rho^{(\star_\ell)}$ and proceed therefrom again.

Empirical illustration of the splitting method

Since a large test usually involves thousands of examinees, the splitting algorithm should be efficient. The dimensionality of the test score matrix, which is $N \times n$, shows that the number of columns which presents the number of questions in exam usually much less than the number of rows which presents the number of examinees in the test. As a result, we focus on finding the equal-mean and nearly equal variance groups by using column swap method.

We conducted a number of experiments with finding reliabilities of larger test data sets that were simulated. The simulations were performed such that the test score variable has a Bernoulli distribution with parameter p. In these simulations, we chose p_i as fixed for the i-th examinee, with p_i randomly sampled from a chosen Gaussian pdf, i.e. $p_i \sim \mathcal{N}(0.5, 0.2)$ by choice, $i = 1, \ldots, N$. We simulated different test score data sets in this way, including

- a test data set for 5×10^5 examinees taking a 50-item test.
- a test data set for 5×10^4 examinees taking a 50-item test.
- a test data set for 1000 examinees taking a 100-item test.
- a test data set for 1000 examinees taking a 1000-item test.

In each case, the test data was split using our method and reliability of the test was computed as per the classical definition. The 4 simulated test data sets mentioned above, yielded reliabilities of 0.96, 0.98, 0.93, 0.85, in order of the above enumeration. Histograms of the sub-tests obtained by splitting each test data were over-plotted to confirm their concurrence.

Importantly, the run-time of reliability computation of these large cohorts of examinees

(N=500,000 and 50,000), who take the 50-item long test, is very short–from about 0.8 seconds for the 50,000 cohort to about 6.2 seconds for the 500,000 cohort. On the other hand the order of our splitting algorithm being $O((n/2)^2)$, the run-times increase rapidly for the 1000-item test, from the 100-item one, with the fixed examinee number. These experiments indicate that the computation of reliabilities for very large cohorts of examinees, in a test with a realistic number of items, is rendered very fast indeed, using our method.

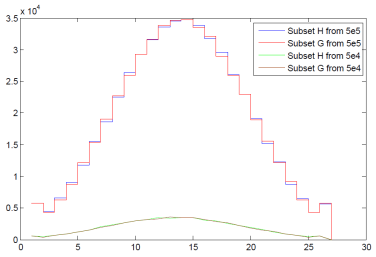

Figure A.1: Histogram of scores obtained by 5e5 and 5e4 number of examinees

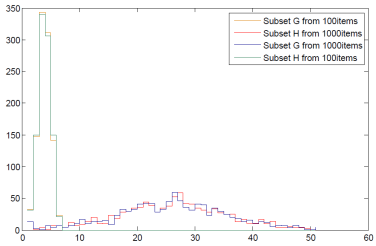

Figure A.2: Histogram of scores obtained by 100 and 1000 questions

Figure 1 shows the histogram obtained by 5×10^5 and 5×10^4 number of examinees from 50 questions. Figure 2 shows the histogram from 100 questions and 1000 questions by 1000 examinees. The nearly coincident histogram for subgroups in same exam indicates that our algorithm works well. As mentioned, the order of our splitting algorithm being $O((n/2)^2)$. All this indicates the algorithm solves the large data set efficiently.

Code for the splitting method

In this section, I present the code used to implement the splitting method. The array "h[]" and "g[]" stores the split item-wise scores from 100 items and the C++ function "find_parallel()" finds the parallel splitting for the two group.

```
void find_parallel(long h[],long g[],long sum_h,long sum_g,
   long difference[])
{
      long i,j,temp,k,m;

      j=sum_h-sum_g;
      m=1;
      while (m==1)
      {
         m=0;
         for(i=1;i<51;i++)
         {
            temp=sum_h-sum_g-2*difference[i];
            if (abs(temp)<abs(j))
            {
                  j=temp;
                  k=i;
                  m=1;
            }
         }
         if(m==1)
         {
            sum_h=sum_h-difference[k];
```

```
            sum_g=sum_g+difference[k];
            swap(h[k],g[k]);
            difference[k]=-difference[k];
            temp=sum_h-sum_g;
        }
    }
}


int main()
{
    ...
    for(j=1;j<51;j++)
    {
        difference[j]=0;
        h[j]=score_no[j*2-1];
        g[j]=score_no[j*2];
        if(j%2==0)
        swap(h[j],g[j]);
        sum_h=sum_h+score[h[j]];
        sum_g=sum_g+score[g[j]];
        difference[j]=score[h[j]]-score[g[j]];
```

```
        }

        find_parallel(h,g,sum_h,sum_g,difference);
            ...

}
```

Appendix B

Code for Tensor variate MCMC

In this section, I present the numerical implementation of the MCMC algorithm for the Tensor variate GP, as a code in C++:

1. sampling parameters (stored in array Q1 and Sigma_c) from random walk, where the gaussrand() function is a random function for sampling random variables distributed according to the standard Normal, i.e. $\mathcal{N}(0,1)$; array a[][2] stores the jump scale for the random walk sampling:

```
...
Q1[1][1]=q1[1][1]+a[1][2]*gaussrand();
Q1[2][2]=q1[2][2]+a[2][2]*gaussrand();

Sigma_c[1][1]=Q3[1][1]+a[3][2]*gaussrand();
Sigma_c[2][2]=Q3[2][2]+a[6][2]*gaussrand();
Q3[3][1]=Q3[3][2]+a[5][2]*gaussrand();
Sigma_c[1][2]=Q3[3][1]*pow((long double)Sigma_c
    [1][1],0.5)*pow((long double)Sigma_c[2][2],0.5);
```

```
Sigma_c[2][1]=Sigma_c[1][2];
...
```

2. Calculate the kernel parameterised covariance matrix and the tensor variate the 3rd-order Tensor Normal likelihood. The three covariance matrices of the 3-rd order Tensor Normal density, is stored in array Sigma_a, Sigma_b and Sigma_c. Function cholsl() calculates the factors of the relevant matrix, using Cholesky Decomposition. Function inverse() calculates the inverse of an input matrix. Function MatMultt(),MatMultt2(),MatMultt3() calculates the multiplication of 3rd-order tensor and matrix, in 3 different dimensions:

```
cholsl(3,Sigma_c,sigma_c);

if ((Sigma_c[1][1]<0)||(Sigma_c[2][2]<0)) continue;
if ((Q1[1][1]<0)||(Q1[2][2]<0)) continue;
if ((Q2[1][1]<0)||(Q2[2][2]<0)) continue;

for(i=1;i<217;i++)
    for(j=1;j<217;j++)
    {
        Sigma_a[i][j]=(long double)exp(-((s[1][i]-s[1][j])
            *Q1[1][1]*(s[1][i]-s[1][j])+(s[2][i]-s[2][j])*
            Q1[2][2]*(s[2][i]-s[2][j])));
    }
```

```
cholsl(217,Sigma_a,sigma_a);

for(i=1;i<217;i++)
for(j=1;j<217;j++)
{
     Inv_a[i][j]=0;
     if(i==j)  Inv_a[i][j]=1;
     temp[i][j]=sigma_a[i][j];
}
inverse(217,temp,Inv_a);
for(i=1;i<217;i++)
for(j=i+1;j<217;j++)
{
   Inv_a[i][j]=Inv_a[j][i];
   Inv_a[i][j]=0;
}

cholsl(51,Sigma_b,sigma_b);

for(i=1;i<51;i++)
for(j=1;j<51;j++)
{
```

```
        Inv_b[i][j]=0;
        if(i==j)  Inv_b[i][j]=1;
        temp[i][j]=sigma_b[i][j];
    }
    inverse(51,temp,Inv_b);
    for(i=1;i<51;i++)
    for(j=i+1;j<51;j++)
    {
        Inv_b[i][j]=Inv_b[j][i];
        Inv_b[i][j]=0;
    }

for(i=1;i<3;i++)
        for(j=1;j<3;j++)
        {
            Inv_c[i][j]=0;
            temp[i][j]=sigma_c[i][j];
        }
    Inv_c[1][1]=1;
    Inv_c[2][2]=1;

    e=temp[1][1];
    for(j=1;j<=2;j++)
```

```
{
      temp[1][j]=temp[1][j]/e;
      Inv_c[1][j]=Inv_c[1][j]/e;
}
Inv_c[2][1]=-Inv_c[1][1]*temp[2][1];
temp[2][2]=temp[2][2]-temp[2][1]*temp[1][2];

e=temp[2][2];
for(j=1;j<=2;j++)
{
      temp[2][j]=temp[2][j]/e;
      Inv_c[2][j]=Inv_c[2][j]/e;
      Inv_c[1][j]=Inv_c[1][j]-temp[1][2]*Inv_c[2][j];
}
for(i=1;i<3;i++)
for(j=i+1;j<3;j++)
{
    Inv_c[i][j]=Inv_c[j][i];
    Inv_c[i][j]=0;
}

MatMultt(51,217,3,Inv_b,vv,tem);
MatMultt2(51,3,217,Inv_c,tem,tem2);
```

```
MatMultt3(217,51,3,Inv_a,tem2,tem);

sum=0;

for(i=1;i<217;i++)
    for(j=1;j<3;j++)
        for(k=1;k<51;k++)
            sum=sum+tem[i][j][k]*tem[i][j][k];

det_3=Sigma_c[1][1]*Sigma_c[2][2]-Sigma_c[1][2]*Sigma_c
    [2][1];
det_1=0;
for(i=1;i<20;i++)
{
      det_1=det_1+log(sigma_a[i][i]);
}

det_2=0;
for(i=1;i<20;i++)
{
      det_2=det_2+log(sigma_b[i][i]);
}
```

```
prob=-sum/2-216*2*50*(det_1/(216)+log(det_3)/4+det_2
    /(50));
```

3. Calculate the logarithm of the acceptance ratio and determine whether to accept the new sampled value or not.

```
u=(long double)rand()/RAND_MAX;
uu=(long double)prob-pre_prob;
u=log(u);


if(u<uu)
{


    pre_prob=prob;
    for(i=1;i<3;i++)
        for(j=1;j<3;j++)
        {
            Q3[i][j]=Sigma_c[i][j];
            q1[i][j]=Q1[i][j];
            q2[i][j]=Q2[i][j];
        }
    Q3[1][2]=Q3[3][1];
 }
```

Appendix C

Code for graphical model

In this section I present the MCMC algorithm for the graphical model of the real wine datasets:

1. Calculate the likelihood of the between-columns correlation matrix given the data, using the closed-form likelihood we present in the text (Chapter 3). Function cholsl() calculates the factors for the relevant matrix using Cholesky Decomposition. Function inverse() calculates the inverse of input matrix. Function MatMult() calculates the matrix multiplication:

```
...
for(i=1;i<param;i++)
for(j=i+1;j<param;j++)
{
    Sigma_c[i][j]=prop[i][j][0]+gaussrand()*0.005;
    Sigma_c[j][i]=Sigma_c[i][j];
}
```

```
        for(i=1;i<param;i++)
        {
        Sigma_c[i][i]=1;
        for(j=1;j<param;j++)
                temp[i][j]=Sigma_c[i][j];
                    }
        error=cholsl(param,temp,sigma_c);

        det_c=0;

        for(i=1;i<param;i++)
               det_c=det_c+log(sigma_c[i][i]);

        for(i=1;i<param;i++)
        for(j=1;j<param;j++)
                temp[i][j]=Sigma_c[i][j];

        error=inverse(param,temp,Inv_C);

//calculate    data*Sigma^-1*data'
        MatMult(line,param,param,data_s,Inv_C,tem2);
        MatMult(line,param,line,tem2,data_sT,tem);
```

```
        error=cholsl(line ,tem ,tem3);



//calculate the det of data*Sigma^-1*data'
        sum=0;
        for(i=1;i<line;i++)
        {
             sum=sum+log(tem3[i][i]);
    }
        prob=0;
        t=param;
//calculate the marginal likelihood
        prob=prob-det_c*(t-(long double)1.0)-sum*t*(long
           double)2.0;
        prior=0;
            for(i=1;i<param;i++)
           for(j=i+1;j<param;j++)
                    {
                              prior=prior-(Sigma_c[i][j]-M[i][j
                                 ])*(Sigma_c[i][j]-M[i][j])/(2*
                                 pri_var[i][j]);
                                  }
```

```
        prob=prob+prior;

            uu=prob-pre_prob;//accept ratio
            u=(long double)rand() / RAND_MAX;

    if(log(u)<uu)
        {
            for(i=1;i<param;i++)
                for(j=i+1;j<param;j++)
                    prop[i][j][0]=Sigma_c[i][j];
            pre_prob=prob;
            ct_sum=sum;
            ct_det=det_c;
        }
```

2. Calculates the particle correlation from the correlation matrix. The correlation matrix is stored in array [covariance] and the particle correlation is stored in array [p_corr]

```
        error=inverse2(param,covariance,inv_co);

        for(i=1;i<13;i++)
        for(j=1;j<13;j++)
        {
          p_corr[i][j]=-inv_co[i][j]/sqrt(inv_co[i][i]*inv_co
```

```
        [j][j]);
    if (i==j)
    {
        p_corr[i][j]=sqrt(1-1/inv_co[i][j]);
    }
}
```

3. Update the graph using the particle correlation matrix.

```
...
    likelihood=0;
    for(i=1;i<13;i++)
     for(j=i+1;j<13;j++)
     {
        g[i][j]=biornd(p_corr[i][j]);
        g[j][i]=g[i][j];
        if (g[i][j]==1)  prob_graph=prob_graph+g[i][j]*
           log(p_corr[i][j]);
        if (g[i][j]==0)  prob_graph=prob_graph+(1-g[i][j
           ])*log(1-p_corr[i][j]);
        t5=length_scale[i][j];
        t2=(long double)exp(-pow(g[i][j]-p_corr2[i][j
           ],2)/(t5))+exp(-pow(g[i][j]+p_corr2[i][j],2)
           /(t5));
```

```
            likelihood=(long double) likelihood+log(t2)-log(
                sqrt(t5));
        }

        u=(long double)rand()/(long double)RAND_MAX;
        uu=likelihood-prob_graph-pre_prob_graph;
        u=log(u);
        if(u<uu)
       {
           pre_prob_graph=likelihood-prob_graph;
           pre_likelihood=likelihood;
           for(i=1;i<13;i++)
               for(j=1;j<13;j++)
                   G[i][j]=g[i][j];
       }
...
```