Clustering Time Series Data by Analysing Graphical Models of Connectivity and the Application to Diagnosis of Brain Disorders

a thesis presented for the degree of

Doctor of Philosophy of Imperial College London

and the

Diploma of Imperial College

by

Robert Wolstenholme

Department of Mathematics

Imperial College

180 Queen's Gate, London SW7 2BZ

December 2016

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Signed: Robert Wolstenholme

Copyright

Thesis advisor: Professor Andrew Walden Robert Wolstenholme

Clustering Time Series Data by Analysing Graphical Models of Connectivity and the Application to Diagnosis of Brain Disorders

Abstract

In this thesis we investigate clustering and classification techniques applied to time series data from multivariate stochastic processes. In particular we focus on extracting features in the form of graphical models of conditional dependence between the process components. The motivation is to use the techniques on brain EEG data measured from multiple patients and investigate whether it can be used in areas such as medical diagnosis. We look at both the case where the graphical model is estimated based on time series recorded on the scalp and also where the graphical model is estimated based on source signals within the brain. In the first case we use a multiple hypothesis testing approach to build the graphical models and a learning algorithm based on random forests to find patterns within multiple graphical models. In the second case we use independent component analysis (ICA) to extract the source time series and estimate the conditional dependence graphs using partial mutual information. It is of particular note that in this case due to the indeterminacy issues associated with ICA we only know the conditional dependence graphs up to some unknown permutation of the nodes. To solve this issue we use novel methods based on an extension of graph matching to multiple inputs in order to develop a new clustering algorithm. Finally, we show how this algorithm can be combined with further information obtained during the ICA phase contained in columns of the unmixing matrix, to create a more powerful method.

Acknowledgments

Thank you to my supervisor Andrew Walden for his support throughout my PhD.

List of figures

List of tables

List of Publications

R. Wolstenholme and A. Walden, "An efficient approach to graphical modelling of time series," *IEEE Transactions on Signal Processing,* vol. 63, pp. 3266–3276, 2015.

R. Wolstenholme and A. Walden, "A sampling strategy for projecting to permutations in the graph matching problem," *arXiv,* eprint. 1604.04235, 2016.

Material from above is included with permission from IEEE (appendix A)

Contents

0

Introduction

The main contributions of this thesis are a novel conditional dependence graph extraction method for time series based on multiple hypothesis testing, a projection strategy to be used to improve current inexact graph matching techniques, a novel multiple graph matching framework that generalises the idea of inexact graph matching to more than two graphs and an algorithm to be used to cluster time series of EEG data obtained from human brains with a particular focus on projecting the observed data back to sources within the brain.

The brain is the centre of the human nervous system and the most complex organ in the human body. Understanding how it works is a key area in Biological and Medical Science as this provides us with an important insight into the causes of human behaviour and this can be used in the diagnosis and treatment of psychological disorders. A human brain can be though of as a complex network made up of a vast number of neurons. The connections between these neurons form anatomical circuits shaping the basis of all neural computations. To fundamentally understand the brain, it is necessary to focus our attention to identifying them. Due to the complex nature of the brain, trying to map it on a neuron by neuron basis is impractical.

We instead concentrate on studying regions (collections of neurons) in the brain. Through EEG and fMRI scans, we can measure electrical activity in different regions of the brains. When the activity is sampled at regular time intervals, we can apply a variety of time series analysis techniques to extract information from it.

This thesis is concerned with ways in which to cluster and classify multivariate time series and focuses particularly on methods based on the underlying conditional independence graphical models. Some of the key problems to overcome are efficiently estimating these models from time series data, modifying these techniques for when the observed time series are a mixture of latent signals, clustering based on graphical models (particularly with unknown node positions) and extending all techniques to be computationally feasible for high dimensional time series.

The motivation behind the techniques is to aid with diagnosis of neurological disorders in the brain. Through EEG scans we can measure electrical activity at the scalp emitted from different sources within the brain. The measurements form a time series and it is these time series that we use as our data. We have both labelled and unlabelled data. In the case of the labelled data (type I and II schizophrenia), we aim to train a classifier that given unseen data would be able to 'diagnose' it. In the case of the unlabelled data we aim to find clusters of patients with similar neurological activity.

The structure of the thesis is as follows, chapter 1 covers the main preliminary results necessary for the project.

Chapter 2 contains the new multiple hypothesis testing framework approach from [85] for estimating graphical models of time series connectivity measured on the scalp. While it closely follows the original paper, there are also new remarks on how the framework can be extended in future.

Chapter 3 introduces the idea of inexact graph matching and covers standard as well as state of the art algorithms such as FAQ [88] for solving it. The second half of the chapter closely follows [86] in which a projection scheme was developed to send solutions of relaxed versions of the inexact graph matching problem back to permutation matrices.

Chapter 4 once again looks at extracting graphical models from time series but in this case it is concerned with the case where the observed time series is a linear mixing of some latent source time series. The method used to model the process is similar to [33] which uses the idea of ICA applied to the residuals of a vector autoregressive process fit to the observed time series. We use however the partial mutual information rate (PMIR) to extract the conditional dependence graphs. We note that due to inherent properties of the ICA we can only extract source time series up to some unknown scaling and permutation. The unknown scaling issue is dealt with in the chapter by using the PMIR.

Chapter 5 investigates various algorithms that can be used to cluster and classify graphical models with known node position i.e. we could not use it for the ICA extracted graphical models. The chapter is not the focus of the research but an algorithm based on a modified version of random forests [10] was shown to have promising performance on simulated data.

Chapter 6 contains the novel techniques based on extending the idea of graph matching to multiple graphs. Most of the chapter is dedicated to the theory required to generalise the graph matching methods. At the end a clustering algorithm is presented that can for instance be used with our ICA extracted graphical models.

The final chapter 7 shows how we can combine multiple graph matching with information contained in the unmixing matrix after application of ICA to obtain more powerful clustering algorithms. The idea is essentially to perform multiple graph matching with labelled graphs, where the labels come from the columns of the unmixing matrix.

1

Preliminaries

Basics

We begin with some basic definitions for real valued random variables. We assume random variables are real valued with absolutely continuous distributions and that multivariate random variables are p-dimensional unless stated otherwise. We differentiate between univariate and multivariate random variables using bold type i.e. $X \in \mathbb{R}$ and $\boldsymbol{X} \in \mathbb{R}^p$.

Definition 1. *The cumulative probability distribution function (cdf) of a univariate random variable X is defined as*

$$F_X(x) = \Pr(X \leq x).$$

Definition 2. *The joint cumulative probability distribution function (cdf) of univariate random variables X, Y is defined as*

$$F_{X,Y}(x, y) = \Pr(X \leq x \cap Y \leq y).$$

Definition 3. *The probability density function (pdf) of univariate random*

variable X with absolutely continuous distribution is defined

$$f_X(x_0) = \frac{\partial}{\partial x} F_X(x)|_{x=x_0}.$$

Definition 4. *The joint probability density function (pdf) of univariate random variables X, Y with absolutely continuous distributions is defined*

$$f_{X,Y}(x_0, y_0) = \frac{\partial^2}{\partial x \partial y} F_{X,Y}(x, y)|_{(x,y)=(x_0,y_0)}.$$

Definition 5. *The expectation (first central moment) of a univariate random variable X is*

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x dF_X(x),$$

where F_X is the cdf of X.

Remark 1. For a p-valued random variable $\boldsymbol{X}$, we define $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_p]^T$ as

$$\mu_i = \mathbb{E}(X_i)$$

for $i = 1 \ldots p$.

Definition 6. *The variance (second central moment) of a univariate random variable X is*

$$\text{var}(X) = \int_{-\infty}^{\infty} (x - \mu)^2 dF_X(x),$$

where F_X is the cdf of X and $\mu = \mathbb{E}(X)$.

Definition 7. *Assuming the joint pdf exists, the covariance of univariate random variables X and Y is*

$$\text{cov}(X, Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \mu_X)(y - \mu_Y) f_{X,Y}(x, y) dx dy,$$

where $f_{X,Y}$ is the joint pdf of X and Y, $\mu_X = \mathbb{E}(X)$ and $\mu_Y = \mathbb{E}(Y)$.

Remark 2. 1. For a random variable X, $\text{cov}(X, X) = \text{var}(X)$.

2. For a p-valued random variable $\boldsymbol{X}$, we define matrix Σ as

$$\Sigma_{ij} = \text{cov}(X_i, X_j)$$

for $i, j = 1 \dots p$.

Remark 3. The integrals used in definitions 5 and 6 are Riemann-Stieltjes integrals. A stochastic version of the integral will be used later when considering the spectral representation theorem.

Stochastic Processes

Stochastic processes are collections of random variables. We provide some basic definitions and results necessary for the research in this thesis. The material in this section closely follows [64].

Definition 8. *A stochastic process $\{\boldsymbol{X}(t)|t \in T\}$ is a collection of p-valued random variables, indexed by t belonging to an index set T, where $\boldsymbol{X}_t = [X_{1,t}, \dots, X_{p,t}]$.*

Remark 4.

1. In definition 8, t commonly represents time.
2. If t is a discrete parameter, we write $\{\boldsymbol{X}_t|t \in T\}$ as the stochastic process such that $\boldsymbol{X}_t$ is the tth component.
3. If T is not explicitly stated when discussing a discrete parameter stochastic process, we assume it is the set of all integers and we write this $\{\boldsymbol{X}_t\}$.

Definition 9. *We call a sequence of realisations from a stochastic process a time series.*

Definition 10. *The cumulative probability distribution function (cdf) of random variable $\boldsymbol{X}_t \in \mathbb{R}^p$ is written as*

$$F_t(\boldsymbol{a}_0) = \Pr(\bigcap_{i=1}^{p}\{X_{i,t} \leq a_{i,0}\}),$$

where $X_{i,t}$ represents the ith component of $\boldsymbol{X}_t$ and $a_{i,0}$ represents the ith component of $\boldsymbol{a}_0$.

Definition 11. *The joint cumulative probability distribution function of random variables $\boldsymbol{X}_{t_1}, \ldots, \boldsymbol{X}_{t_n} \in \mathbb{R}^p$ is written as*

$$F_{t_1,\ldots,t_n}(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n) = \Pr(\bigcap_{i=1}^{p}\{X_{i,t_1} \leq a_{i,1}\}, \ldots, \bigcap_{i=1}^{p}\{X_{i,t_n} \leq a_{i,n}\}).$$

Definition 12. *The process $\{\boldsymbol{X}_t \,|t \in T\}$ is complete (strongly) stationary if for any $\{t_1, \ldots, t_n\} \subset T$ and τ such that $\{t_1 + \tau, \ldots, t_n + \tau\} \subset T$, then*

$$F_{t_1,\ldots,t_n}(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n) = F_{t_1+\tau,\ldots,t_n+\tau}(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n).$$

Strong stationarity can be too strict an assumption for stochastic processes in many real world applications. Second-order stationarity, a weaker form, is often used in practice. We give the definition as in [64, p. 36].

Definition 13. *The process $\{\boldsymbol{X}_t \,|t \in T\}$ is second-order (weakly) stationary if for any $\{t_1, \ldots, t_n\} \subset T$ and τ such that $\{t_1 + \tau, \ldots, t_n + \tau\} \subset T$, then all joint moments of orders 1 and 2 from $\boldsymbol{X}_{t_1}, \ldots, \boldsymbol{X}_{t_n}$, exist, are finite and equal to the corresponding moments from $\boldsymbol{X}_{t_1+\tau}, \ldots, \boldsymbol{X}_{t_n+\tau}$.*

Definition 14. *The process $\{\boldsymbol{X}_t \,|t \in T\}$ is Gaussian if for any $\{t_1, \ldots, t_n\} \subset T$, the variable $(\boldsymbol{X}_{t_1}^T, \ldots, \boldsymbol{X}_{t_n}^T)$ follows a multivariate Gaussian distribution.*

Definition 15. *The process $\{\boldsymbol{X}_t \,|t \in T\}$ is Gaussian stationary if it is a Gaussian process and second-order stationary.*

Remark 5. Gaussian stationary processes are completely defined by their second order statistics and are therefore completely stationary.

Definition 16. *The cross-covariance sequence (ccvs) of stochastic process $\{\boldsymbol{X}_t\}$ is given by*

$$C_{ij}(s,t) = \operatorname{cov}(X_{i,s}, X_{j,t})$$

where we write $X_{i,s} = (\boldsymbol{X}_s)_i$ and $X_{j,t} = (\boldsymbol{X}_t)_j$.

Remark 6. 1. If $\{X_t\}$ is second order stationary we can write the ccvs simply in terms of time lag τ i.e. we can define matrix $\boldsymbol{s}_\tau$ such that

$$s_{ij,\tau} = C_{ij}(t, t+\tau) = \text{cov}(X_{i,t}, X_{j,t+\tau}) = \text{cov}(X_{i,0}, X_{j,\tau})$$

where we write $(\boldsymbol{s}_\tau)_{ij} = s_{ij,\tau}$.

2. The value $s_{ii,\tau}$ is referred to as the autocovariance sequence (acvs) of stochastic process $\{X_{i,t}\}$.

Spectral Density Function and Cross-Spectra

We introduce the idea of the spectral density function for stochastic processes. Again this section closely follows [64].

Definition 17. *A stochastic process $\{Z(f)\}$, not necessarily real valued, is called an orthogonal process and said to have orthogonal increments if the increments are uncorrelated over disjoint intervals i.e.*

$$\text{cov}(Z(f_4) - Z(f_3), Z(f_2) - Z(f_1)) = \\ \mathbb{E}[(Z(f_4) - Z(f_3))^*(Z(f_2) - Z(f_1))] = 0$$

for $f_1 < f_2 < f_3 < f_4$.

The spectral representation theorem for discrete parameters as stated in [64, p. 36] is as follows

Theorem 1. *Let $\{X_t\}$ be a univariate real-valued discrete parameter stationary process such that $\mathbb{E}(X_t) = 0$. Then, there exists an orthogonal process $\{Z(f)\}$ on $[-1/2, 1/2]$ such that*

$$X_t = \int_{-1/2}^{1/2} e^{i2\pi ft} dZ(f),$$

where the equality is in the mean square sense.

Proof. See [65] p. 251. □

Remark 7. 1. We say that two random variables $\boldsymbol{X}, \boldsymbol{Y}$ are equal in the mean square sense if and only if

$$\mathbb{E}\left[|\boldsymbol{X} - \boldsymbol{Y}|^2\right] = 0.$$

2. If $\mathbb{E}[|dZ(f)|^2]$ is differentiable everywhere, we can write it as

$$\mathbb{E}[|dZ(f)|^2] = S(f)df$$

and we call $S(f)$ the spectral density function.

3. By using the spectral representation of X_t and $X_{t+\tau}$, we can write the acvs as

$$s_\tau = \mathbb{E}(X_t X_{t+\tau}) = \int_{-1/2}^{1/2} S(f) e^{i2\pi f\tau} df.$$

4. As s_τ is deterministic, the above equation indicates it is the inverse Fourier transform of S. Under the assumption that S is square integrable (implying s_τ is square summable by Parseval's theorem) then S is the Fourier transform of s_τ i.e.

$$S(f) = \sum_{\tau=-\infty}^{\infty} s_\tau e^{-i2\pi f\tau}.$$

The transformation into the frequency domain can be extended to multivariate processes using cross-spectra.

Definition 18. *Given p-valued stationary multivariate process $\{\boldsymbol{X}(t)\}$, the cross-spectra between $\{X_{j,t}\}$ and $\{X_{k,t}\}$ are defined as*

$$S_{jk}(f) = \sum_{\tau=-\infty}^{\infty} s_{jk,\tau} e^{-i2\pi f\tau}$$

and

$$S_{kj}(f) = \sum_{\tau=-\infty}^{\infty} s_{kj,\tau} e^{-i2\pi f\tau},$$

where $s_{jk,\tau}$ and $s_{kj,\tau}$ are square summable cross-covariance sequences.

Definition 19. *The spectral density matrix (spectral matrix) of* $\{\boldsymbol{X}_t\}$ *is defined as*

$$\boldsymbol{S}(f) = \begin{pmatrix} S_{11}(f) & S_{12}(f) & \dots & S_{1p}(f) \\ S_{21}(f) & S_{22}(f) & \dots & S_{2p}(f) \\ \vdots & \vdots & \ddots & \vdots \\ S_{p1}(f) & S_{p2}(f) & \dots & S_{pp}(f) \end{pmatrix}.$$

The matrix $\boldsymbol{S}(f)$ holds many important properties about the process $\{\boldsymbol{X}_t\}$ and we will make extensive use of it. One such property from [22] links the spectral matrix to the partial correlation of the univariate processes. This is given below.

Proposition 1. *Given p-valued stochastic process* $\{\boldsymbol{X}_t\}$*, two univariate processes* $\{X_{j,t}\}$ *and* $\{X_{k,t}\}$ *from* $\{\boldsymbol{X}_t\}$ *are partially uncorrelated given all other processes if*

$$S^{jk}(f) = 0 \quad (-1/2 \leq f < 1/2).$$

Proof. See [22]. □

The Periodogram

Given some observations of a stochastic process, we investigate the periodogram, an estimator for the spectral matrix defined in definition 19. Once again this section closely follows [64]. We assume our stochastic process is indexed by time and the difference between successive observations is $\Delta t = 1$.

Definition 20. *For p-valued vector random variables* $\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$*, the periodogram is defined as*

$$\hat{\boldsymbol{S}}^{(p)}(f) = \frac{1}{N} \left| \sum_{t=1}^{N} \boldsymbol{X}_t e^{-i2\pi ft} \right|^2.$$

Remark 8. 1. The periodogram as an estimator in its own right has many undesirable properties, in particular the estimates can have very

high variance. Instead it can be used as a basis for building better estimators e.g. by smoothing or tapering.

2. We can write the discrete Fourier transform of our data as

$$\boldsymbol{W}(f) = \frac{1}{N^{1/2}} \sum_{t=1}^{N} \boldsymbol{X}_t e^{i2\pi ft}.$$

The periodogram is then

$$\hat{\boldsymbol{S}}^{(p)}(f) = \boldsymbol{W}(f)\boldsymbol{W}^*(f).$$

This means we can use fast computational techniques used for calculating Fourier transforms, to calculate $\hat{\boldsymbol{S}}^{(p)}(f)$.

Definition 21. *Let $f_j = j/n$, the jth Fourier frequency and given a weight sequence w_k for $k = -M \dots M$ where M is a parameter to be specified, then the weighted (frequency smoothed) periodogram is defined as*

$$\hat{\boldsymbol{S}}(f_j) = \left(\sum_{k=-M}^{M} w_k \right)^{-1} \sum_{k=-M}^{M} w_k \hat{\boldsymbol{S}}^{(p)}(f_{j+k}). \tag{1.1}$$

If we also require $\sum_{k=-M}^{M} w_k = 1$, we can write the weighted periodogram more concisely as

$$\hat{\boldsymbol{S}}(f_j) = \sum_{k=-M}^{M} w_k \hat{\boldsymbol{S}}^{(p)}(f_{j+k}).$$

Proposition 2. *The weighted periodogram, is non-singular if and only if $2M + 1 \geq p$.*

Proof. See [26] p. 3007. □

Remark 9.

1. An example weight sequence and one that we use later is $w_k = \cos(\pi k/m)$.

2. The weighted periodogram is a consistent estimator of the spectral matrix if $M, N \to \infty$ and $M/N \to 0$ [59].

3. For the samples we consider, we expect to have $M >> p$ and certainly $2M + 1 \geq p$. Hence by proposition 2 we will have a non-singular weighted periodogram that can be inverted.

4. The bandwidth $2M$ is an important parameter to specify as it controls the level of smoothing. A larger M implies more smoothing, less variance but more bias towards the overall mean. The parameter can be chosen visually by simply observing plots of the estimator for different M. However for an automatic way, we recommend using methods such as SURE [51] or PURE [52].

Vector Autoregressive Processes

A commonly studied class of stochastic processes are vector autoregressive (VAR) processes. Below we define VAR processes and some useful conditional independence results when using them. The section closely follows [85].

Definition 22. *A p-valued stochastic process $\{\boldsymbol{\epsilon}_t\}$ is called a white noise process if*

1. $\mathbb{E}[\epsilon_{i,t}] = 0$ *for all* $i = 1, \ldots, p$ *and* $t \in T$.
2. $|\text{cov}(\epsilon_{i,t}, \epsilon_{j,t})| < \infty$ *for all* $i, j = 1, \ldots, p$ *and* $t \in T$.
3. $\text{cov}(\epsilon_{i,t}, \epsilon_{j,s}) = 0$ *for all* $i, j = 1, \ldots, p$ *and* $s, t \in T$ *such that* $s \neq t$.

Definition 23. *A p-valued vector autoregressive process $\{\boldsymbol{X}_t\}$ of order l ($VAR_p(l)$) satisfies the relationship*

$$\boldsymbol{X}_t = \sum_{u=1}^{l} \Phi_u \boldsymbol{X}_{t-u} + \boldsymbol{\epsilon}_t$$

where the $\boldsymbol{\Phi}_u \in \mathbb{R}^{p \times p}$ are coefficient matrices and $\boldsymbol{\epsilon}_t \in \mathbb{R}^p$ is a zero mean white noise process with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} \in \mathbb{R}^{p \times p}$.

Proposition 3. *Let* $\{\boldsymbol{X}_t\}$ *follow a* $VAR_p(l)$ *model and satisfy*

$$\det[\boldsymbol{I}_p - \sum_{u=1}^{l} \boldsymbol{\Phi}_u z^u] \neq 0$$

for all $z \in \mathbb{C}$ *such that* $|z| \leq 1$. *Then* $\{\boldsymbol{X}_t\}$ *is a second-order stationary process.*

Proof. See [57] p. 25. □

Remark 10. For notational ease we write $\boldsymbol{\Phi}_0 = \boldsymbol{I}_p$.

The following proposition can be extracted from [85] to help us see which constituent univariate stochastic processes (channels) of a VAR process are partially uncorrelated. Two univariate stochastic processes from a multivariate stochastic process are partially uncorrelated if they are uncorrelated when the effect from all other processes in the multivariate process is removed.

Proposition 4. *Let* $\{\boldsymbol{X}_t\}$ *be a stationary stochastic process following a* $VAR_p(l)$ *model such that* $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$ *is a diagonal matrix. Then* $\{X_{j,t}\}$ *and* $\{X_{k,t}\}$ *are partially uncorrelated if and only if*

$$\boldsymbol{\Phi}_i(f)^H \boldsymbol{\Phi}_j(f) = 0$$

for all values $-1/2 \leq f < 1/2$, *where*

$$\boldsymbol{\Phi}(f) = -\sum_{u=0}^{l} \boldsymbol{\Phi}_u e^{-i2\pi f u}$$

and we write

$$\boldsymbol{\Phi}(f) = [\boldsymbol{\Phi}_1(f), \ldots, \boldsymbol{\Phi}_p(f)].$$

Proof. For a VAR process, it is a common result that the spectral matrix can be written

$$\boldsymbol{S}(f) = \boldsymbol{\Phi}^{-1}(f) \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} [\boldsymbol{\Phi}^{-1}(f)]^H$$

for $-1/2 \leq f < 1/2$. Hence

$$\boldsymbol{S}^{-1}(f) = \boldsymbol{\Phi}^H(f)\boldsymbol{\Sigma_\epsilon}^{-1}\boldsymbol{\Phi}(f).$$

As $\boldsymbol{\Sigma_\epsilon}$ is diagonal, it follows that $S^{ij}(f) = 0$ if and only if

$$\boldsymbol{\Phi}_i(f)^H\boldsymbol{\Phi}_j(f) = 0.$$

Hence the result follows from proposition 1. □

Remark 11. For $l = 1$, proposition 4 is satisfied for $\{X_{j,t}\}$ and $\{X_{k,t}\}$ if all the following hold

1. $A_{jk} = 0$
2. $A_{kj} = 0$
3. For $m = 1, \ldots, p$, $A_{mj} = 0$ or $A_{mk} = 0$

VAR Models

We use the following $\text{VAR}_5(1)$ models in this section for future testing as in [85].

The $\text{VAR}_5(1)$ model with residuals distributed to a multivariate normal distribution is written

$$\boldsymbol{X}_t = \boldsymbol{\Phi}_1\boldsymbol{X}_{t-1} + \boldsymbol{\epsilon}_t \tag{1.2}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma_\epsilon})$. Hence a model is specified by $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Sigma_\epsilon}$.

Model A:

1. $\boldsymbol{\Sigma_\epsilon} = \boldsymbol{I}_5$

2.
$$\boldsymbol{\Phi}_1 = \begin{bmatrix} 0.2 & 0 & -0.1 & 0 & -0.5 \\ 0.4 & -0.2 & 0 & 0.2 & 0 \\ -0.2 & 0 & 0.3 & 0 & 0.1 \\ 0.3 & 0.1 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.5 & 0.2 \end{bmatrix}. \tag{1.3}$$

3. By inspection of $\boldsymbol{\Phi}_1$ and use of proposition 4, we see that the set of missing edges is $\{(2,3),(2,5),(3,4)\}$.

Model B (Matsuda [59]):

1. $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} = \boldsymbol{I}_5$
2.
$$\Phi_1 = \begin{bmatrix} 0.2 & 0 & 0.3 & 0 & 0.3 \\ 0.3 & -0.2 & x & 0 & 0 \\ 0.2 & x & 0.3 & 0 & 0 \\ 0.2 & 0.3 & 0 & 0.3 & 0 \\ 0.2 & 0 & 0.2 & 0.2 & 0.2 \end{bmatrix}, \tag{1.4}$$

3. Again using proposition 4, we see that for $x = 0$, the set of missing edges is $\{(2,3),(2,5)\}$ and for $x = 0.1$ the set of missing edges is $\{(2,5)\}$.

Model C:

1. $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{-1} = \boldsymbol{I}_5$ except for entries $(1,2)$ and $(2,1)$ which are equal to 0.5.
2. Φ_1 of the same as (1.4) with $x = 0$
3. By considering in proposition 4 the form of the spectral matrix
$$\boldsymbol{S}^{-1}(f) = \boldsymbol{\Phi}^H(f)\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}{}^{-1}\boldsymbol{\Phi}(f)$$
we see only edge $(2,3)$ is missing as opposed to $\{(2,3),(2,5)\}$ when $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$ was diagonal.

Random VAR Model Construction

We also make use of the random $\text{VAR}_p(1)$ model construction specified in the appendix of [85].

> For a given p value a $p \times p$ matrix $\mathbf{\Phi}_1$ was constructed with null entries. All diagonal elements and non-diagonal elements in position (i, j) for which $(i+j)_{\bmod k} = 1$ were populated by random values sampled from the $\mathcal{N}(0, 1)$ distribution. The matrix was then subject to spectral decomposition and any eigenvalues with modulus greater than unity were replaced by their reciprocals and $\mathbf{\Phi}_1$ reconstructed using the modified eigenvalues. For such a $\mathbf{\Phi}_1$ we know $\det\{\boldsymbol{I}_p - \mathbf{\Phi}_1 z\} \neq 0$ for all $|z| \leq 1$, [57, pp. 15 & 653] and so a stationary process results. The choice of k controls the sparsity; our default choice $k = 5$ makes approximately 64% of the $\mathbf{\Phi}_1$ matrix entries zero for $p = 10 : 50$.

Graphical Models

In this section we include some basic definitions for graphical models.

Definition 24. *A graph $G = (V, E)$ is an ordered pair where V is a set of vertices/nodes and E is a multiset (allows multiple instances of a set element) of edges. An edge is an ordered pair of elements from V.*

Remark 12. We can think of an edge $(v_1, v_2) \in E$ as representing some directed relationship between vertices v_1 and v_2.

Definition 25. *For graphical models $G = (V, E)$ and $G' = (V, E')$, we say G' is a subgraph of G if $E' \subset E$ and we write this as $G' \subset G$.*

Definition 26. *A graph is simple and undirected if it contains no multiple edges, loops and all edges are undirected i.e*

1. *E is a set*

2. $\nexists v \in V$ *such that* $(v, v) \in E$

3. $(v_1, v_2) \in E \Leftrightarrow (v_2, v_1) \in E$.

Remark 13. When working with undirected graphs, we can treat edges as undirected pairs of vertices i.e. instead of writing $(v_1, v_2) \in E$ and $(v_2, v_1) \in E$, we can write $\{v_1, v_2\} \in E$.

Definition 27. *A graph is called a weighted graph if it has a weight/number assigned to each edge.*

Definition 28. *For graph* $G = (V, E) \in \mathbb{G}$*, the associated adjacency matrix* $A \in [0, 1]^{p \times p}$ *satisfies*

$$A_{ij} = 0 \Leftrightarrow (i, j) \notin E$$

and

$$A_{ij} = 1 \Leftrightarrow (i, j) \in E.$$

Remark 14.
1. We write the space of all simple directed graphs with p vertices as $\mathbb{G}_p$.
2. We write the adjacency matrix of $G \in \mathbb{G}_p$ as $A(G) \in \mathbb{R}^{p \times p}$.
3. If $P \in \mathbb{P}$ is a permutation matrix, we write $P^T G P$ as the graph with adjacency matrix $P^T A(G) P$.

Definition 29. *We define the set of edges containing all information about a simple undirected graph in* $\mathbb{G}_p$ *as*

$$ES(p) = \{\{i, j\} | 1 \leq i, j \leq p \text{ and } i < j\}$$

Random Bernoulli Graphs

Definition 30. *Let* $p \in \mathbb{Z}^+$ *and* $Q \in \mathbb{R}^{p \times p}$ *such that* $0 \leq Q_{ij} \leq 1$ *for* $1 \leq i, j \leq p$*. Define the random Bernoulli graph* $G^{RBG}(Q)$ *to be a probability distribution over* $\mathbb{G}_p$ *such that for* $G = G^{RBG}(Q)$*,*

$$Pr[(i, j) \in E(G)] = Q_{ij}$$

for all edges $(i,j) \in \{(i,j)|1 \leq i,j \leq p\}$ *and the events (edges existing) are mutually independent.*

Remark 15. Given a graph $G_0 \in \mathbb{G}_p$, then for $G = G^{RBG}(Q)$,

$$Pr(G = G_0) = \prod_{1 \leq i,j \leq p} Q_{ij}^{I[(i,j) \in E(G_0)]} \cdot (1 - Q_{ij})^{I[(i,j) \notin E(G_0)]}.$$

Graphical Models for Stochastic Processes

In this section we define what it means for a stochastic process to have an associated graphical model.

Definition 31. *Given p-valued stochastic process* $\{\boldsymbol{X}_t\}$*, we say two univariate processes* $\{X_{j,t}\}$ *and* $\{X_{k,t}\}$ *from* $\{\boldsymbol{X}_t\}$ *are conditionally independent given all other processes, if* $X_{j,s}$ *and* $X_{k,t}$ *are independent for all* $s,t \in \mathbb{Z}$*, given the random variables in stochastic process* $\{\boldsymbol{X}_{\setminus jk,t}\}$*. We write this as*

$$\{X_{j,t}\} \perp \{X_{k,t}\}|\{\boldsymbol{X}_{\setminus jk,t}\}. \tag{1.5}$$

We now define a graphical model for a multivariate stochastic process as in [22].

Definition 32. *Given a p-valued multivariate stochastic process* $\{\boldsymbol{X}_t\}$*, let* $V = \{1 \dots p\}$ *be a set of vertices and* $E \subset V \times V$ *a set of edges between these vertices. Then, if for* $j \neq k$

$$(j,k) \notin E \Leftrightarrow \{X_{j,t}\} \perp \{X_{k,t}\}|\boldsymbol{X}_{\setminus jk,t}$$

and for $j = k$

$$(j,k) \notin E,$$

we say (V,E) *is a graphical model for* $\{\boldsymbol{X}_t\}$*.*

Remark 16. Using definition 32, the graphical model of $\{\boldsymbol{X}_t\}$ is a simple undirected graph.

The spectral matrix of a stationary Gaussian process is related to its graphical model by the following result in [22].

Proposition 5. *For a multivariate Gaussian stationary process* $\{\boldsymbol{X}_t\}$*, the following are equivalent*

$$\{X_{j,t}\} \perp \{X_{k,t}\} | \{\boldsymbol{X}_{\setminus jk,t}\}$$

and

$$S^{jk}(f) = 0 \quad (-1/2 \leq f < 1/2).$$

Proof. Follows from proposition 1 and the fact that partial correlation for Gaussian stationary processes is equivalent to conditional independence. □

Definition 33. *Consider a p-dimensional graph* $G = (V, E)$ *such that each vertex represents a set of random variables. Let* Z_i *represent the set of random variables for node* i*. Also, let* $N(i)$ *represent the set of nodes that node* i *shares an edge with. This then forms a Markov random field if the following is satisfied,*

$$Z_i \perp Z_j | Z_{N(i)}$$

for all $i \in V$ *and* $j \in \{k | k \in V \text{ and } k \notin N(i) \text{ and } k \neq i\}$*.*

Remark 17. Definition 32 doesn't use any notion of direction or causality and defines an undirected graphical model. In this case, the graph and univariate stochastic processes form a Markov random field [47].

Example 1. Consider the $\text{VAR}_4(1)$ model

$$\boldsymbol{X}_t = A\boldsymbol{X}_{t-1} + \boldsymbol{\epsilon}_t \tag{1.6}$$

where $\boldsymbol{X}_t = (X_{1,t} \dots X_{4,t})^T$, $\boldsymbol{\epsilon}_t \sim N(0, \boldsymbol{I}_4)$ and

$$A = \begin{pmatrix} 0.3 & 0 & 0.1 & 0.2 \\ 0.4 & -0.2 & 0 & 0.2 \\ 0 & 0 & 0.1 & 0 \\ 0.1 & 0.2 & 0 & -0.3 \end{pmatrix}.$$

By inspection of matrix A and the placement of the zeros, we can see using proposition 4, that the conditionally independent processes are $\{X_{2,t}\}$ and $\{X_{3,t}\}$, as they don't have any influence on each other except through other variables. The graphical model of $\{\boldsymbol{X}_t\}$ is represented by the graph below.

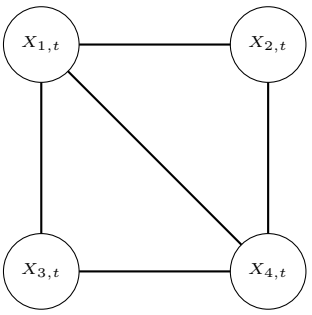

Figure 1.1: Graphical Model For (1.6)

Matrix Norms

Definition 34. *The Frobenius norm between $A, B \in \mathbb{R}^{p\times p}$ is defined as*

$$||A - B||_F = \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{p}(A_{ij} - B_{ij})^2}.$$

Multiple Hypothesis Tests

When a number of hypotheses are being tested simultaneously, we refer to this as a multiple hypothesis test. With an individual test, we look to control the level at which we incorrectly reject a null hypothesis e.g. only 5 % of the time. This can be generalised when working with multiple hypothesis tests. We will look at two measures of control, the familywise error rate (FWER) and the false discovery rate (FDR).

Definition 35. *If Y is the number of true null hypotheses that are falsely rejected, then the FWER is defined as*

$$FWER = P(Y \geq 1).$$

Remark 18. 1. The FWER is essentially the probability of making at least one incorrect rejection.

2. If we were to individually test n independent hypotheses at a significance level α, then we would have

$$FWER = 1 - (1 - \alpha)^n.$$

For a large n this value can be very high for standard choices of α (e.g. 5 %). If we are looking to control using FWER, this is may not be what we want.

Definition 36. *If Y is the number of true null hypotheses that are falsely rejected and R is the total number of hypotheses rejected, then the FDR is defined as*

$$\mathbb{E}\left(\frac{Y}{R}\right).$$

Remark 19. When we have a large number of hypotheses, controlling the FWER at the 5 % level say, it can result in a very unpowerful test. This is because controlling the probability of making at least one incorrect rejection must be very conservative causing many tests to be incorrectly accepted (of course this depends on the difference in distribution between the null and alternative and the proportion of true to false null hypotheses but the argument is made qualitatively). In this case it can make more sense to use the FDR as the controlling procedure.

Maximin Stepdown Procedure

Whether controlling using the FDR or the FWER, the maximin stepdown procedure [53, Sec. 9.2] can be used to perform multiple hypothesis testing. The algorithm is as follows

Algorithm 1 Maximin Stepdown Procedure

Require: Ordered test statistics $Z^1 \leq Z^2 \leq \cdots \leq Z^L$ and monotonic critical levels $C_1, \ldots, C_L$

```
accept_index ← 0
for i = L : 1 do
    if Z^i < C_i then
        accept_index ← i
        break
    end if
end for
accept_set ← {j|1 ≤ j ≤ accept_index}
return accept_set
```

Remark 20.

1. If $H_1, \ldots, H_L$ are the corresponding hypotheses to statistics $Z^1, \ldots, Z^L$ in algorithm 1, then we accept all H_i where $i \in$ `accept_set` and reject the rest.

2. In order to modify the procedure for different control procedures (FWER or FDR), we change the values of the critical levels that we input.

3. The way we have given algorithm 1 is in terms of test statistics. That is because later we use it for performing multiple hypothesis tests where the individual tests are one-sided. For a more general approach, the algorithm only requires a small modification so that instead of taking test statistics it takes corresponding p-values.

FWER

Proposition 6. *If we choose the C_l in algorithm 1 such that*

$$\Pr(Z^l < C_l | H_1, \ldots, H_l) = 1 - \alpha \tag{1.7}$$

then $FWER \leq \alpha$.

Proof. See [53] theorem 9.1.3. □

Remark 21. Hence to control the FWER we need to solve equations (1.7) for C_l. Different methods are needed for when the statistics are dependent and independent.

Proposition 7. *To control the FWER with independent identically distributed statistics, use critical levels that solve*

$$\Pr(Z^* < C_l) = (1-\alpha)^{\frac{1}{l}},$$

where Z^* is the distribution of the statistics under the null hypothesis.

Proof. Using first the ordered nature of the statistics and secondly the identical independent distribution of the statistics, we can write (1.7) as

$$\begin{aligned}
&\Pr(Z^l < C_l | H_1, \ldots, H_l) \\
&= \prod_{i=1}^{l} \Pr(Z^i < C_l | H_1, \ldots, H_l) \\
&= \Pr(Z^* < C_l)^l.
\end{aligned}$$

Using the ordering $Z^1 \leq Z^2 \leq \cdots \leq Z^L$. Hence,

$$\Pr(Z^* < C_l) = (1-\alpha)^{\frac{1}{l}}.$$

□

Remark 22. For Z^* a standard normal distribution, we have

$$C_l = \Phi^{-1}((1-\alpha)^{\frac{1}{l}}).$$

In this case we would be performing a one-sided test, only rejecting values if they are too large (not too small).

When the statistics are dependent, we can use the Holm approach [53, p. 363] to choose the critical levels. This comes from the following proposition.

Proposition 8. *To control the FWER for dependent test statistics, use critical levels that solve*

$$\Pr(Z^* < C_l) = 1 - \frac{\alpha}{l}.$$

Proof. See [53] p. 363. □

Remark 23. If Z^* is standard normal distributed, we have

$$C_l = \Phi^{-1}\left(1 - \frac{\alpha}{l}\right).$$

FDR

Once again, we can use different approaches for controlling the FDR when the statistics are independent and dependent. The first is from [7].

Proposition 9. *To control the FDR with independent identically distributed statistics, use critical levels that solve*

$$\Pr(Z^* < C_l) = 1 - \frac{\alpha l}{L}.$$

Proof. See [7]. □

The paper [8] showed how to control the FDR for dependent statistics.

Proposition 10. *To control the FDR for positive dependent test statistics (i.e.* $\Pr(X \cap Y) > \Pr(X)\Pr(Y)$*), use critical levels that solve*

$$\Pr(Z^* < C_l) = 1 - \frac{\alpha l}{L\sum_{i=1}^{L} 1/i}.$$

Proof. See [8]. □

Remark 24. The $\sum_{i=1}^{L} \frac{1}{i}$ term in proposition 10 can cause the test to be very conservative. Finding alternative critical values for more powerful tests is a current area of active research.

Independent Component Analysis

Independent component analysis (ICA) is a form of blind source separation (BSS). Consider some stochastic process $\{\boldsymbol{X}_t \in \mathbb{R}^{\tilde{p}}\}$ and some unobserved source stochastic process $\{\boldsymbol{S}_t \in \mathbb{R}^p\}$ such that

$$\boldsymbol{X}_t = f(\{\boldsymbol{S}_t\}), \tag{1.8}$$

where f is some transformation of the source process. Blind source separation is concerned with retrieving the source process $\{\boldsymbol{S}_t\}$ from the observation process $\{\boldsymbol{X}_t\}$ [19, Ch. 1.2].

Depending on the form of transformation f, we can get different mixing models e.g.

1. Instantaneous: f is an instantaneous linear transformation, represented by a matrix A i.e. $\boldsymbol{X}_t = A\boldsymbol{S}_t$.
2. Convolutive: f is a finite impulse response filter represented by coefficient matrices $A_0, \ldots, A_l$ i.e. $\boldsymbol{X}_t = \sum_{i=0}^{l} A_i \boldsymbol{S}_{t-i}$.

If we are to determine f, we need some assumptions to overcome the clear unidentifiability inherent to (1.8). These generally fall under the umbrella of assuming diversity between the sources which gets reduced when they are mixed together. Hence we aim to choose an unmixing transformation that maximises this diversity between the recovered sources. Note also that it is only possible to recover the sources from the observations if the transformation f is invertible.

Some common types of diversity are:

1. Independent sources with non-Gaussian distributions (classical Independent Component Analysis).
2. Sources with different temporal dependency structures.
3. Non-circularity where the sources are complex valued [30].

We now look at some common ICA methods.

Classical ICA

The classical approach to ICA considers observations as an instantaneous linear mixing of sources [18] i.e.

$$\boldsymbol{X}_t = A\boldsymbol{S}_t \tag{1.9}$$

for $A \in \mathbb{R}^{\tilde{p}\times p}$.

The sources are assumed spatially independent, with non-Gaussian distributions (except at most one) and $\tilde{p} = p$. In this case the BSS model in (1.9) is identifiable. Note we do not use any temporal properties of the data and the time indexing is essentially unimportant i.e. we could re-order the observations in any way we wanted and it would not affect the model.

A popular measure of dependence between sources is mutual information and it is common to separate the sources in (1.9) by minimising the mutual information of the recovered signals [41]. We write the mutual information for process $\{\boldsymbol{S}_t\}$ as

$$I(\{\boldsymbol{S}_t\}) = \sum_{i=1}^{p} H(\{S_{i,t}\}) - H(\{\boldsymbol{S}_t\}), \tag{1.10}$$

where $H(\{S_{i,t}\})$ is the entropy of process $\{S_{i,t})\}$ and $H(\{\boldsymbol{S}_t\}$ is the joint entropy of the multivariate process $\{\boldsymbol{S}_t\}$. Note this is not the entropy *rate* and does not take into account temporal dependency.

For example the deterministic process $a_t = 1 - a_{t-1}$ where $a_{-1} = 0$ has zero entropy rate as it is simply a sequence of alternate 0 and 1. The entropy of the sequence however is log(2) because while the entropy rate doesn't consider there to be any randomness in the sequence, the entropy doesn't take the temporal information into account and simply views a value from the sequence as being 0 with probability 1/2 and 1 with probability 1/2.

We also have the relation [19, Lemma 2.1], if g is an invertible linear transformation such that $\boldsymbol{S}_t = g(\boldsymbol{X}_t)$ and g' is its Jacobian, then

$$H(\{\boldsymbol{S}_t\}) = H(\{\boldsymbol{X}_t\}) + \mathbb{E}(\log|\det g'(\boldsymbol{X}_t)|). \tag{1.11}$$

Note for all linear transformations satisfying $g(\mathbf{x}) = B\mathbf{x}$, $g'(\mathbf{x}) = B$ and from (1.10) and (1.11) this leads to the following contrast function that we aim to minimise

$$C(B) = \sum_{i=1}^{p} H(\{S_{i,t}\}) - \log|\det B|, \tag{1.12}$$

where $\boldsymbol{S}_t = B\boldsymbol{X}_t$ and $B \in \mathbb{R}^{p\times p}$. This measures how independent the recovered sources are when B is used as the unmixing matrix.

Note for a finite number of observations, the entropies $H(\{S_{i,t}\})$ must be replaced by an estimator. A common method is to estimate the distribution of a component using a kernel estimate [5] from which $H(\{S_{i,t}\})$ can be calculated. Some popular algorithms for performing classical ICA are FastICA [40], JADE [16] and Infomax [6].

Remark 25. When performing ICA, we want the recovered unmixing matrix estimate $\hat{B}$ from minimising (1.12) to be close to A^{-1} from (1.9). Note however the columns of A and the source components can be permuted, or the former multiplied by a constant and the latter by the inverse of the constant and (1.9) is still satisfied. Hence there is an inherent indeterminability for ICA of an unknown scaling and permutation.

ICA with Temporal Dependence

A further measure of diversity we can use is that of temporal dependence between the sources. A nice consequence of this is that we can separate sources with Gaussian distributions if they have different second order temporal dependence structures [81].

The separation can be seen as a consequence of minimising the mutual information rate [19, Ch. 2.3],

$$I_r(\{\boldsymbol{S}_t\}) = \sum_{i=1}^{p} H_r(\{S_{i,t}\}) - H_r(\{\boldsymbol{S}_t\}),$$

where

$$H_r(\{\boldsymbol{S}_t\}) = \lim_{T\to\infty} H(\boldsymbol{S}_1, \ldots, \boldsymbol{S}_T)/T$$

is the entropy rate. This leads to a contrast function analogous to before

$$C(B) = \sum_{i=1}^{p} H_r(\{S_{i,t}\}) - \log|\det B|. \tag{1.13}$$

As mentioned in [1], minimisation of entropy rate works in two ways. Firstly it minimises entropy which can be thought of as minimising redundant information but it also promotes temporal dependence as this means new observations are easier to predict given past observations, therefore reducing entropy rate.

Minimising (1.13) means that now (1.9) is separable if no two sources are both Gaussian distributed and have proportional autocovariance matrices [1]. Alternatively, what this is saying is that it is only inseparable if we have two Gaussian source components $\{S_{i,t}\}$ and $\{S_{j,t}\}$ that have proportional autocovariance matrices i.e. $\boldsymbol{R}^i \neq c\boldsymbol{R}^j$ for some $c \in \mathbb{R}$ where $\boldsymbol{R}^k_{t_1t_2} = \mathbb{E}[S_{kt_1}S_{kt_2}]$, the autocovariance matrix of $\{S_{k,t}\}$.

Innovation Process ICA

Another possibility to take into account temporal information is to assume a model as in [39] where we define the innovation process $\tilde{S}_t$ as the difference between $\boldsymbol{S}_t$ and the best prediction of it from past data

$$\tilde{\boldsymbol{S}}_t = \boldsymbol{S}_t - \mathbb{E}(\boldsymbol{S}_t|t, \boldsymbol{S}_{t-1}, \ldots \boldsymbol{S}_1). \tag{1.14}$$

Determining $\mathbb{E}(\boldsymbol{S}_t|t, \boldsymbol{S}_{t-1}, \ldots \boldsymbol{S}_1)$ is a regression problem which can be solved using anything from a VAR model to a neural network, depending on the

underlying structure of the stochastic process $\{\boldsymbol{S}_t\}$.

Multiplying both sides of (1.14) by A and combining with (1.9) gives

$$A\tilde{\boldsymbol{S}}_t = \boldsymbol{X}_t - \mathbb{E}(\boldsymbol{X}_t|t, \boldsymbol{S}_{t-1}, \ldots \boldsymbol{S}_1)$$

and as A is invertible, the information contained in $\boldsymbol{X}_{t-1}, \ldots \boldsymbol{X}_1$ is equivalent to that in $\boldsymbol{S}_{t-1}, \ldots \boldsymbol{S}_1$. So,

$$A\tilde{\boldsymbol{S}}_t = \boldsymbol{X}_t - \mathbb{E}(\boldsymbol{X}_t|t, \boldsymbol{X}_{t-1}, \ldots \boldsymbol{X}_1) = \tilde{\boldsymbol{X}}_t.$$

Therefore, if we work with the assumption the $\tilde{\boldsymbol{S}}_t$ are mutually independent and non-Gaussian, we can recover the mixing matrix A by applying the classical ICA techniques to $\tilde{\boldsymbol{X}}_t$, recovered by fitting a regression model to $\boldsymbol{X}$. This is the approach we adopt in later chapters.

2

Building Graphical Models from Time Series at the Scalp Level Using a Multiple Hypothesis Test

Introduction

In this chapter, we look at a method for estimating the graphical model of a p-valued stochastic process $\{\boldsymbol{X}_t\}$ given some observed time series from the process. In particular, we concentrate on Gaussian stationary processes. The graph has p vertices and therefore $p(p-1)/2$ potential undirected edges. This means there are $2^{p(p-1)/2}$ possible graphical models to choose from. We look to develop a non-parametric technique that works for large dimensional values p, without having to assume sparsity of the edges in the graphical model. A prominent issue we then have to overcome is that of being able to build a computationally tractable algorithm. We aim to solve this by designing an efficient and readily parallelisable algorithm that can easily

scale with advance in high performance computation hardware. The work in this chapter closely follows that of [85].

Early work on extracting graphical models from time series was carried out by Brillinger [12]. Dalhaus [22] proposed a non-parametric testing approach which the techniques in this chapter are closely related to. Unfortunately his initial approach required a test that the partial coherence between two process components is zero at every computed frequency. This has many complications, so an alternative test based on the maximum partial coherence across all frequencies was proposed. However the asymptotic null distribution is not known and must be approximated.

In [59], Matsuda used a non-parametric approach based on the Kullback-Leibler (KL) divergence between estimated spectral matrices for an observed time series, obeying certain constraints. The approach tests, based on a statistic derived from the KL divergence, whether a spectral matrix constrained to a certain graphical model is significantly different from the spectral matrix constrained to an alternative graphical model where it is assumed the alternative graphical model is a correct graph (defined in section 2.1.1). The asymptotic null distribution of the statistic is normal and fairly easy to calculate.

Matsuda used the test in an iterative procedure by starting with a fully connected graphical model (which is guaranteed to be correct) and testing whether alternative subgraphs with one edge removed do not have a significantly different constrained spectral matrix. Intuitively, if this is the case for a subgraph, the constraint imposed on our spectral matrix by the graphical model with an edge removed had little effect, implying the graphical model is also correct. Matsuda then repeats this procedure but now testing using the newly found correct graphical model in place of the fully connected graphical model. Iterating in this way, once we reach a point where no more edges can be removed without resulting in a likely incorrect graphical model, it means we are left with a good estimate for the true graphical model.

Unfortunately the approach is computationally intractable even for time series with only moderately sized p (i.e. $p = 10$) and it doesn't have a global

type I error rate control, instead only using a control at each step of the iteration. The relationship between this and the global rate is unclear. Moreover, the calculation of the test statistic past the first iteration requires the iterative procedure of section 2.1.2 as the exact formulation cannot be used which further increases the computational burden.

Wolstenholme and Walden [85] proposed a highly parallel framework based on multiple hypothesis testing (MHT) that solves many of the issues with Matsuda's approach. In particular, presenting an algorithm that is computationally tractable for far larger dimensional time series. The main points as presented in the paper are

1. The tests are only performed between the fully connected graph and graphs with one edge missing. This means that the test statistics can be calculated using the exact approach in section 2.1.2 and do not require iteration.

2. The MHT framework allows control of the FWER, a global measure of the type I error rate. While not mentioned in the paper, only a small modification to the critical levels used, allows the algorithm to be adjusted to control the FDR, another global type I error measure. This may be more appropriate in many practical applications.

3. The MHT method does not require the iterative testing that Matsuda uses. This significantly reduces the computational burden of the algorithm, resulting in a complexity of $O(p^4)$ in comparison to $O(p^6)$.

4. Simulations in [85] showed that for similar type I error rates, the MHT algorithm achieved a power 'at least as good' as Matsuda's algorithm. These results are included at the end of this chapter.

5. As their is no iterative test procedure, there is also no dependency between the outcome of individual tests. This allows the algorithm to be almost perfectly parallelisable. Once again this was demonstrated empirically and the results are included at the end of this chapter.

The structure of this chapter is similar to [85] and is as follows, section 2.1 contains some necessary preliminary results. In section 2.2 we introduce the test statistic used by Matsuda and the iterative procedure he used for selecting a graphical model. Section 2.3 is concerned with the multiple hypothesis testing approach of Wolstenholme and Walden. The complexity of the algorithm is formally investigated in section 2.4. The speed and accuracy of both algorithms is compared empirically in section 2.5. A demonstration of the parallelisability of the MHT approach along with some tests on larger dimension models is included in section 2.6. Finally, section 2.7 contains the results of the MHT algorithm on real EEG data, as used in [85] and the chapter is ended with some concluding remarks and ideas for future work in section 2.8.

Remark 26.

1. One can argue that the iterative approach of Matsuda may lead to a more parsimonious model being fitted at each step, resulting in a more accurate final graphical model than the MHT. [85] show this does not appear to be the case for the small dimension models it was computationally feasible to run a comparison for.

2. Another issue, not investigated in much detail, with the Matsuda iterative approach is that if a mistake is made i.e. an edge is removed that shouldn't be, we are no longer testing against a correct model on the next iteration. This is a vital assumption for the test to have any meaning.

3. The MHT framework can be used with other statistics that measure whether an edge should be included in a graphical model, and alternative error measures. Only the Matsuda test statistic and FWER were considered by [85].

Preliminaries

Correct Graphs

The concept of correct graphs was introduced in [59] and also used in [85], in order to test whether the true graphical model for Gaussian stationary stochastic process $\{\boldsymbol{X}_t\}$ given by definition 32, is a subgraph of a proposed graphical model.

Definition 37. *A graph $G^{'} = (V, E^{'})$ is correct for Gaussian stationary stochastic process $\{\boldsymbol{X}_t\}$ with true graphical model $G = (V, E)$, if $\forall (j,k) \notin E^{'}$*

$$S^{jk}(f) = 0 \quad (-1/2 \leq f < 1/2) \tag{2.1}$$

where $\boldsymbol{S}$ is the spectral matrix for $\{\boldsymbol{X}_t\}$.

Remark 27. By proposition 5, we see that (2.1) is equivalent to saying $G \subset G^{'}$. Clearly then the graphical model with no edges missing, G^{*} is correct for any $\{\boldsymbol{X}_t\}$. We later use the idea of missing edges imposing constraints on the original time series conditional independence structure. When thinking in this way, G^{*} imposes no constraints, so is always correct.

Example 2. We illustrate the idea of correct graphs with an example from [85].

In figure 2.1, G is the true graphical model. The graphs G_0 and G_1 are correct as $G \subset G_0$ and $G \subset G_1$. However $G \not\subset G_2$ as the edge $\{2, 4\}$ is present in G but not in G_2.

Fitting a Matrix Subject to Modelling Constraints in its Inverse

We make use of the following technique employed in [59]. We first define the problem to be solved and then give an algorithm for it.

Definition 38. *Consider a matrix $\boldsymbol{X} \in \mathbb{R}^{p \times p}$ and a set $F \subset \{(i,j) | i = 1 \ldots p, j = 1 \ldots p\}$ such that*

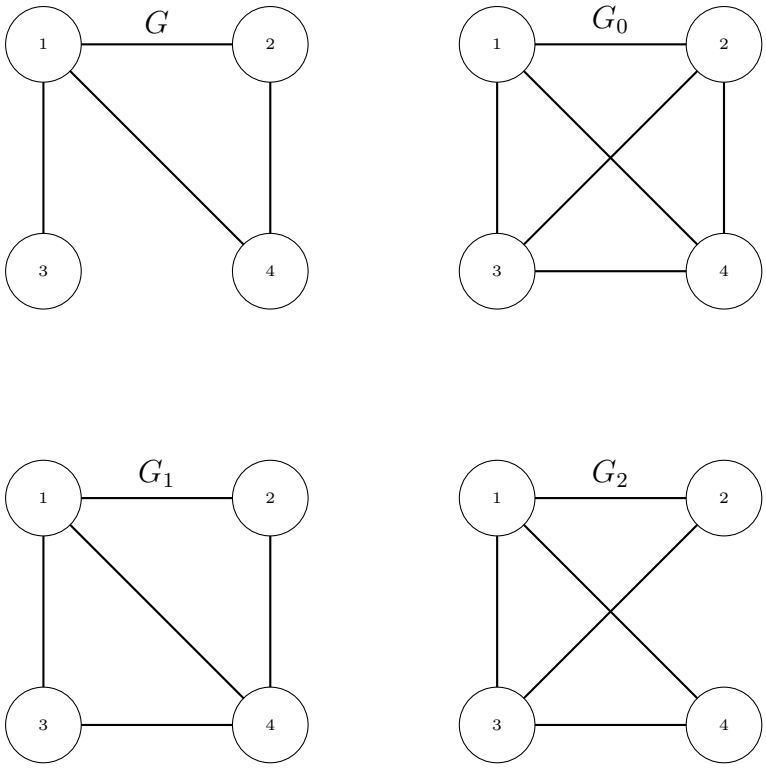

Figure 2.1: G_0 and G_1 are correct for G, but G_2 is not correct for G

1. *If* $(i, j) \in F$, *then* $(j, i) \in F$.

2. $(i, i) \notin F$ *for all* i.

The fitting a matrix subject to modelling constraints in its inverse problem (FMMCIP) is concerned with finding $\boldsymbol{Y} \in \mathbb{R}^{p \times p}$ *satisfying*

$$\begin{array}{cc} Y_{jk} = X_{jk} & (j,k) \in F, \\ Y^{jk} = 0 & (j,k) \notin F. \end{array} \tag{2.2}$$

Proposition 11. *If* $|F| = 2$, *then the FMMCIP is solved by*

$$\begin{array}{cc} Y_{jk} = X_{jk} & (j,k) \in F, \\ Y_{jk} = X_{jk} + \frac{X^{jk}}{X^{jj}X^{kk} - X^{jk}X^{kj}} & (j,k) \notin F. \end{array}$$

Proof. See [84]. □

For any sized set F, the following algorithm from [84] can be used to solve the FMMCIP.

Algorithm 2 Fitting a Matrix Subject to Modelling Constraints in its Inverse

Require: $\boldsymbol{X}$, F, convergence condition

Split F into m mutually exclusive sets F_i such that $\cup F_i = F$, $|F_i| = 2$ and $(i,j) \in F_i \Leftrightarrow (j,i) \in F_i$
$Y_0 \leftarrow X$
$n \leftarrow 0$
while convergence not true **do**
    $n \leftarrow n + 1$
    $n' \leftarrow n \mod m + 1$
    for $j = 1 : p$ **do**
        for $k = 1 : p$ **do**
            if $(j,k) \in F_{n'}$ **then**
                $(Y_n)_{jk} \leftarrow (Y_{n-1})_{jk} + \frac{Y_{n-1}^{jk}}{Y_{n-1}^{jj} Y_{n-1}^{kk} - Y_{n-1}^{jk} Y_{n-1}^{kj}}$
            else
                $(Y_n)_{jk} \leftarrow (Y_{n-1})_{jk}$
            end if
        end for
    end for
end while
return $\boldsymbol{Y}_n$

Proposition 12. *In algorithm 2, the sequence of values $\boldsymbol{Y}_n$ pointwise converges to the solution of the FMMCIP i.e.*

$$\lim_{n\to\infty} \boldsymbol{Y}_n = \boldsymbol{Y}$$

where $\boldsymbol{Y}$ solves the FMMCIP.

Proof. See [78] proposition 3. □

Remark 28. 1. Given an edge set of an undirected graph E, we can write $F = \{(i,j)|\{i,j\} \in E, i > j\} \cup \{(i,j)|\{i,j\} \in E, i < j\}$. Then, solving the FMMCIP for F and the spectral matrix of a stochastic process

$\{\boldsymbol{X}_t\}$ can be thought of as creating a new spectral estimator 'close' to the old one but whose associated graphical model has zero edges at least for all edges not in E (using proposition 5). Hence the graph associated with the edge set E can be thought of as a constraint and the new spectral estimator is a constrained version of the old.

2. If in the above we think of F as a set of directed edges, the statement $|F| = 2$ in proposition 11 in is equivalent to saying E contains one undirected edge.

Matsuda's Work

In this section we give an overview of the work from [59]. We describe how the test statistic was constructed and the iterative algorithm used for estimating the graphical model of a Gaussian stationary stochastic process given some observed time series.

Graphical Constraints

The following result from [60] is important when it comes to calculating constrained estimated spectral matrices.

Proposition 13. *Given graph (V, E) and (estimated) spectral matrix $\hat{\boldsymbol{S}}(f) \in \mathbb{R}^{p\times p}$. Then given another graph (V, E'), there exists $\hat{\boldsymbol{T}}(f) \in \mathbb{R}^{p\times p}$ such that*

$$\begin{array}{ll} \hat{T}_{jk}(f) = \hat{S}_{jk}(f) & (j,k) \in E', \\ \hat{T}^{jk}(f) = 0 & (j,k) \notin E'. \end{array} \tag{2.3}$$

Proof. See [60] lemma 7. □

Remark 29. As mentioned in [85], proposition 13 suggests a method for determining whether a graph $G_2 = (V, E_2)$ is correct given we know graph $G_1 = (V, E_1)$ is correct and $G_2 \subset G_1$. Writing $\boldsymbol{T}_1(f)$ as the unique matrix solving (2.3) for true spectral matrix $\boldsymbol{S}(f)$ and graphical model G_1, then clearly by proposition 5, $\boldsymbol{T}_1(f) = \boldsymbol{S}(f)$ as G_1 is correct. Hence we can

calculate estimate of true spectral matrix $\hat{\boldsymbol{S}}(f)$ and find $\hat{\boldsymbol{T}}_1(f)$ and $\hat{\boldsymbol{T}}_2(f)$ using (2.3) with $\hat{\boldsymbol{S}}(f)$ and graphs G_1 and G_2 respectively. Intuitively, if we find frequency values such that there is a big difference between $\hat{\boldsymbol{T}}_2(f)$ and $\hat{\boldsymbol{T}}_1(f) \approx \boldsymbol{S}(f)$, then it suggests G_2 is not correct.

Testing Problem

Using the ideas in the above remark, we want to be able to perform the following test (using the same notation as [85]),

$$H_0 : (V, E_2) \text{ correct} \quad \text{vs} \quad H_A : (V, E_2) \text{ incorrect}$$

To perform this test we want to use some comparison between $\hat{\boldsymbol{T}}_1(f)$, the estimated spectral matrix constrained by (V, E_1) and $\hat{\boldsymbol{T}}_2(f)$, the estimated spectral matrix constrained by (V, E_2), where (V, E_1) is known to be correct. A large difference provides more evidence in favour of the alternative hypothesis H_A.

Example 3. The 2D example from [85] helps illustrate this idea. Let $\hat{\boldsymbol{S}}(f)$ be the estimated spectral matrix for some observed time series. Let (V, E_1) be the fully connected graphical model (certainly correct) and (V, E_2) be the graphical model with both vertices unconnected. Then by (2.3),

$$\hat{\boldsymbol{T}}_1(f) = \begin{pmatrix} \hat{S}_{11}(f) & \hat{S}_{12}(f) \\ \hat{S}_{21}(f) & \hat{S}_{22}(f) \end{pmatrix}$$

and

$$\hat{\boldsymbol{T}}_2(f) = \begin{pmatrix} \hat{S}_{11}(f) & 0 \\ 0 & \hat{S}_{22}(f) \end{pmatrix}.$$

Hence in this case, if $\hat{S}_{12}(f)$ and $\hat{S}_{21}(f) = \hat{S}_{12}(f)^*$ are far from zero for some frequency f, this provides evidence that (V, E_2) is incorrect and the vertices are connected. This also makes intuitive sense in the 2D case where conditional independence (and partial uncorrelation) are equivalent to independence (and uncorrelation).

Test Statistic

Matsuda used the estimated KL divergence to create his test statistic. This is defined as follows

Definition 39. *The estimated KL divergence between estimated spectral matrices $\hat{\boldsymbol{T}}_1(\cdot)$ and $\hat{\boldsymbol{T}}_2(\cdot)$ defined at the Fourier frequencies $f_j = j/N$ for $j = 1 \dots N/2$ is written*

$$eKL(\hat{\boldsymbol{T}}_1, \hat{\boldsymbol{T}}_2) = \frac{1}{N} \sum_{j=1}^{N/2} \left[\mathrm{tr}(\hat{\boldsymbol{T}}_1(f_j)\hat{\boldsymbol{T}}_2^{-1}(f_j)) - \log\det(\hat{\boldsymbol{T}}_1(f_j)\hat{\boldsymbol{T}}_2^{-1}(f_j)) - p \right]$$

where N is assumed even.

The test statistic is then defined as follows.

Definition 40. *Given observed time series $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$, let $\hat{\boldsymbol{S}}$ be the weighted periodogram, calculated with weight sequence $\{w_k\}$ such that $w_k = u\left(\frac{k}{2M}\right)$ for $k = -M \dots M$ and $u(\cdot)$ an even function continuous on $[-1/2, 1/2]$. Also, let $\hat{\boldsymbol{T}}_1(f)$ and $\hat{\boldsymbol{T}}_2(f)$ be the solutions of (2.3) where $\hat{\boldsymbol{S}}(f)$ is the estimated spectral matrix and (V, E_1) and (V, E_2) are the respective graphical models, with (V, E_1) assumed correct.*

Then the Matsuda test statistic is written

$$Z_N(\hat{\boldsymbol{T}}_1, \hat{\boldsymbol{T}}_2) = \left[\frac{2MN}{D_u(m_2 - m_1)} \right]^{1/2} \left[eKL(\hat{\boldsymbol{T}}_1, \hat{\boldsymbol{T}}_2) - \frac{C_u(m_2 - m_1)}{2M} \right] \quad (2.4)$$

where $m_i = \#\{\{j,k\} | \{j,k\} \notin E_i\}$ (total missing edges in (V, E_i)) and C_u, D_u are constants determined by $u(\cdot)$ whose exact formulation can be found in [59].

The following result from [59] shows the asymptotic distribution of the above statistic under certain conditions.

Proposition 14. *Given the following assumptions,*

1. *The underlying stochastic process $\{\boldsymbol{X}_t\}$ is Gaussian stationary.*

2. *The true spectral matrix* $\boldsymbol{S}(f)$ *is positive definite for* $|f| \leq 1/2$.

3. $S_{jk}(f)$ *is twice differentiable for* $j, k = 1, \ldots, p$ *and* $-1/2 \leq f < 1/2$.

4. $M = O(N^{\beta})$ *for* $-1/2 < \beta < 3/4$.

Then the following holds:

1. *Under* H_0*:*

$$Z_N(\hat{\boldsymbol{T}}_1, \hat{\boldsymbol{T}}_2) \to \mathcal{N}(0,1) \quad as \quad N \to \infty \ .$$

2. *Under* H_A*:*

$$Z_N(\hat{\boldsymbol{T}}_1, \hat{\boldsymbol{T}}_2) = \left[\frac{2MN}{D_u(m_2 - m_1)}\right]^{1/2} KL(\boldsymbol{S}, \boldsymbol{T}_2) + o_p([MN]^{1/2}),$$

where $\boldsymbol{S}$ *represents the true spectral matrix and* $KL(\cdot, \cdot)$ *is the true KL divergence.*

Proof. See [59]. □

Remark 30. As noted in [59], the dominant term under H_A is positive, resulting in a one-sided critical region.

Algorithm

We now provide the algorithm from [59], making use of the test statistic defined in definition 40.

Algorithm 3 Matsuda's algorithm for estimating the graphical model of a stochastic process

Require: Significance level α, observed time series $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^p$ from process, bandwidth parameter M, even continuous function $u(\cdot)$ on $[-1/2, 1/2]$

Ensure: $p \leq 2M + 1$

Calculate weight sequence $w_k = u\left(\frac{k}{2M}\right)$ for $-M, \ldots, M$
Calculate constants C_u, D_u using weight function u and formulation in [59]
Calculate $\hat{\boldsymbol{S}}(f)$ the weighted periodogram for observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ with bandwidth M and weight sequence $\{w_k\}$
Set (V, E_0) as the p vertex graph with no missing edges
$k \leftarrow 0$
$\hat{\boldsymbol{T}}_0 \leftarrow \hat{\boldsymbol{S}}$
while True **do**
    Let $(V, E_{k+1}^1), \ldots, (V, E_{k+1}^{L_k})$ be the distinct subgraphs of (V, E_k) with exactly one more edge missing
    for $i = 1 : L_k$ **do**
        $\hat{\boldsymbol{T}}_{k+1}^{i} \leftarrow$ solution of (2.3) for graph (V, E_{k+1}^i) using algorithm 2
        $Z_N^i \leftarrow Z_N(\hat{\boldsymbol{T}}_k, \hat{\boldsymbol{T}}_{k+1}^{i})$
    end for
    $C_k(\alpha) \leftarrow \Phi^{-1}((1-\alpha)^{1/L_k})$
    `accept_set` $\leftarrow \{i | Z_N^i \leq C_k(\alpha)\}$
    if `accept_set` $= \emptyset$ **then**
        break
    else
        $j \leftarrow$ `index_of_min`$(\{Z_N^i | i \in$ `accept_set`$\})$
        $(V, E_{k+1}) \leftarrow (V, E_{k+1}^j)$
        $\hat{\boldsymbol{T}}_{k+1} \leftarrow \hat{\boldsymbol{T}}_{k+1}^{j}$
        $k \leftarrow k + 1$
    end if
end while
Return (V, E_k)

Remark 31.
1. The algorithm ends at step k when all the test statistics are below $C_k(\alpha)$.
2. The probability all statistics are below the critical level, assuming all

statistics are Gaussian is

$$\Pr\left(\bigcap_{i=1}^{L_k}(Z_N^i \leq C_k(\alpha))\right) \geq \prod_{i=1}^{L_k} \Pr\left(Z_N^i \leq C_k(\alpha)\right) = 1 - \alpha.$$

Hence if we specify a test at step k being the hypothesis all (V, E_{k+1}^i) are correct, we should only make an error if we find a test statistic greater than $C_k(\alpha)$, which from above occurs with probability $\leq \alpha$.

3. Note that the test in the point above that is performed at each iteration of the algorithm is rather strange, as certainly if all (V, E_{k+1}^i) are correct, then the true graphical model should be the one with no edges. As this is rarely the case, the error measure does not seem entirely relevant. We instead see a more intuitive measure of the error being controlled in the next section.

Worked Example

We now provide a worked example for algorithm 3, that was first shown in [85]. The inputs are as follows,

1. $\alpha = 0.5$
2. $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{1024}$ simulated time series from model A of section 1.5.1
3. $M = 64$
4. $u(x) = \cos(\pi x)$

The true graph for the model used has missing edges $\{\{2,3\},\{2,5\},\{3,4\}\}$. The constant values are evaluated as $C_u = 0.617$ and $D_u = 0.446$. The weight sequence is easily calculated as $w_k = u(k/2M)$ for $k = -M, \ldots, M$.

Setting (V, E_0) as the fully connected graphical model and looking at the main loop in algorithm 3, we observe the following, where values of test statistics and critical levels are given in table 2.1,

Edge	$k=0$	$k=1$	$k=2$	$k=3$
(1,2)	53.71	54.03	57.02	67.63
(1,3)	12.72	14.62	17.55	17.54
(1,4)	22.25	24.14	24.14	23.71
(1,5)	67.92	68.96	70.12	79.62
(2,3)	0.54	**0.20**	—	—
(2,4)	18.16	17.82	17.82	22.41
(2,5)	1.89	1.90	**1.94**	—
(3,4)	**0.21**	—	—	—
(3,5)	5.86	5.29	5.50	5.49
(4,5)	73.17	72.60	72.60	77.23
$C_k(0.05)$	2.53	2.49	2.44	2.39

Table 2.1: Test statistics $Z_N(T_k, T^i_{k+1})$ and critical levels $C_k(0.05)$ for Matsuda's algorithm

$k = 0$: Edge $\{3, 4\}$ corresponds to the lowest test statistic and it is below the critical level, so set (V, E_1) to the graph with it missing.

$k = 1$: As above, although this time for edge $\{2, 3\}$. Set (V, E_2) to the graph with edges $\{\{3, 4\}, \{2, 3\}\}$ missing.

$k = 2$: As above, this time for edge $\{2, 5\}$. Set (V, E_3) to the graph with edges $\{\{3, 4\}, \{2, 3\}, \{2, 5\}\}$ missing.

$k = 3$: All test statistics are above the critical level. Break the loop and return (V, E_3).

Note that (V, E_3) is in fact the true graphical model for the process defined in model A and the algorithm has been successful.

An Efficient Testing Framework

In this section we look at the efficient testing framework used in [85] along with Matsuda's test statistic. The idea is based on multiple hypothesis testing where the critical levels are chosen in order to control the FWER.

Multiple Hypothesis Testing

The multiple hypothesis testing approach in [85] is very similar to the $k = 0$ step of algorithm 3. The comparison against critical values is however slightly different. The reason why only the $k = 0$ step is required is explained in their proposition 15.

Proposition 15. *Let $f : \{1, \ldots, p(p-1)/2\} \to E_0$ be a bijective mapping from the integers to a fully saturated edge set E_0. Let $E_1^i = E_0 - f(i)$, the edge set containing all possible edges except for $f(i)$. If the graph (V, E_1^i) is correct (with respect to some stochastic process $\{\boldsymbol{X}_t\}$)for $i = i_1, \ldots, i_s$ and incorrect for all others, then the graphical model (V, E) for $\{\boldsymbol{X}_t\}$ is the graph with only edges $\{f(i_1), \ldots, f(i_s)\}$ missing.*

Proof. If graph (V, E_1^i) is correct and i corresponds to the edge $\{j, k\}$, then by definition $S^{jk}(f) = S^{kj}(f) = 0$ for $-1/2 \leq f < 1/2$ where $\boldsymbol{S}(f)$ is the spectral matrix of the true graphical model. This means that edge $\{j, k\}$ must also be missing in (V, E) and this is the case for all $i = i_1, \ldots, i_s$. Conversely, if (V, E_1^i) is incorrect, $S^{jk}(f) \neq 0$ and $\{j, k\}$ must necessarily be in (V, E), hence the result. □

Remark 32. Proposition 15 means that in fact knowing whether every (V, E_1^i) is correct or incorrect is enough information to infer the true graphical model. Hence we can restrict ourselves to just performing these tests.

Algorithm

We can now give the general algorithm for performing the multiple hypothesis testing, where we leave the critical levels to be specified. First list the $L = p(p-1)/2$ hypotheses as in [85] to make notation easier,

$$
\begin{aligned}
H_1 : &\quad (V, E_1^1) \text{ is correct; } (1,2) \notin E \\
&\quad \vdots \\
H_{p-1} : &\quad (V, E_1^{p-1}) \text{ is correct; } (1,p) \notin E \\
H_p : &\quad (V, E_1^p) \text{ is correct; } (2,3) \notin E \\
&\quad \vdots \\
H_L : &\quad (V, E_1^L) \text{ is correct; } (p-1,p) \notin E.
\end{aligned}
$$

The algorithm is then given in algorithm 4.

Algorithm 4 MHT algorithm for estimating the graphical model of a stochastic process

Require: Significance level α, observed time series $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^p$ from process, bandwidth parameter M, even continuous function $u(\cdot)$ on $[-1/2, 1/2]$, critical level function $C_i(\cdot) : [0,1] \to \mathbb{R}$

Ensure: $p \leq 2M + 1$

1: Calculate weight sequence $w_k = u\left(\frac{k}{2M}\right)$ for $-M, \ldots, M$
2: Calculate constants C_u, D_u using weight function u and formulation in [59]
3: Calculate $\hat{\boldsymbol{S}}(f)$ the weighted periodogram for observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ with bandwidth parameter M and weight sequence $\{w_k\}$
4: Calculate $\hat{\boldsymbol{T}}_1^i$ from $\hat{\boldsymbol{T}}_0 = \hat{\boldsymbol{S}}$ for all (V, E_1^i) using the exact formulation in proposition 11 as there is only one missing edge
5: Calculate test statistics $Z_N^i = Z_N(\hat{\boldsymbol{T}}_0, \hat{\boldsymbol{T}}_1^i)$ for all (V, E_1^i)
6: Order test statistics by permutation σ i.e. $Z_N^{\sigma(1)} \leq \cdots \leq Z_N^{\sigma(L)}$
7: Calculate critical levels $C_i = C_i(\alpha)$ for $i = 1 \ldots L$
8: `sorted_accept_set` $\leftarrow$ output of maximin stepdown procedure (algorithm 1) with the ordered statistics and calculated critical levels
9: `accept_set` $\leftarrow \{\sigma(i) | i \in$ `sorted_accept_set`$\}$
10: $(V, E) \leftarrow$ Graph missing only edges corresponding to hypothesis H_i where $i \in$ `accept_set`
11: Return (V, E)

Remark 33. 1. In [85], critical levels were chosen using the Holm ap-

i	Missing Edge	$Z_N^{(i)}$	$C_i(0.05)$
10	(4,5)	73.17	2.58
9	(1,5)	67.92	2.54
8	(1,2)	53.71	2.50
7	(1,4)	22.25	2.45
6	(2,4)	18.16	2.39
5	(1,3)	12.72	2.33
4	(3,5)	5.86	2.24
3	(2,5)	1.89	2.13
2	(2,3)	0.54	1.96
1	(3,4)	0.21	1.64

Table 2.2: Ordered statistics $Z_N^{(i)}$ and critical levels $C_i(0.05)$ for MHT

proach to control the FWER at level α i.e.

$$C_i(\alpha) = \Phi^{-1}(1 - \frac{\alpha}{i}).$$

2. We note that choosing critical levels according to other methods can control different error measures e.g. using the FDR.

Worked Example

Once again, we use an example from [85] to demonstrate algorithm 4. Calculating the test statistics and critical levels for each edge leads to the values in table 2.2, ordered by test statistic column, where we use the same inputs as in the previous worked example (i.e. observations based on VAR model A) and critical levels defined by the Holm approach above.

Note that as $Z_N^{(i)} \geq C_i(\alpha)$ for $i = 10, \ldots, 4$ and $Z_N(3) < C_3(\alpha)$, we accept hypotheses corresponding to $Z_N^{(3)}, Z_N^{(2)}$ and $Z_N^{(1)}$ and reject the rest. This is equivalent to returning an estimated graphical model with edges $\{\{2,3\},\{2,5\},\{3,4\}\}$ missing, which is in fact the true graphical model for model A.

Theoretical Complexity

The theoretical complexity of both algorithms is given in the following propositions from [85].

Proposition 16. *The number of test statistics calculated in the Matsuda algorithm is $O(p^4)$ and in the MHT is $O(p^2)$.*

Proof. For Matsuda's algorithm, assuming the final output is the true graphical model with k missing edges,

$$\begin{aligned} \frac{p(p-1)}{2} &+ \left[\frac{p(p-1)}{2} - 1\right] + \cdots + \left[\frac{p(p-1)}{2} - k\right] \\ &= (k+1)\frac{p(p-1)}{2} - \frac{k(k-1)}{2} \end{aligned} \tag{2.5}$$

test statistics are calculated, where $k \in \{0, \ldots, p(p-1)/2\}$. Setting the ratio of non-edges to total possible edges to a, we can write $k = a\frac{p(p-1)}{2}$ for $0 \leq a \leq 1$. Then substituting into (2.5), the total number of test statistics needing to be calculated, satisfies

$$p^4\left[\frac{a}{4} - \frac{a^2}{8}\right] + o(p^4),$$

where $o(p^4)$ denotes terms of smaller order than p^4.

For the MHT, regardless of the number of missing edges in the model, we always calculate $p(p-1)/2$ statistics, which is $O(p^2)$. □

Remark 34. The value $\frac{a}{4} - \frac{a^2}{8}$ is increasing for $a \in [0, 1]$. Hence the sparser the graph, the larger the value of a and the more test statistics need to be calculated in the Matsuda algorithm (as it takes more iterations to remove the necessary edges).

We now consider the complexity of actually computing the test statistics. Given we have already computed the estimated weighted periodogram, a necessary preprocessing step in both algorithms, we have the following result.

Proposition 17. *Given estimated weighted periodogram $\hat{\boldsymbol{S}}(f) \in \mathbb{R}^{p\times p}$ at the Fourier frequencies, constrained estimated weighted periodogram $\hat{\boldsymbol{T}}_1(f)$, constants C_u, D_u and total iterations l_k in calculating (2.3), then given $\hat{T}_1(f)$ and $\hat{T}_2(f)$, we can calculate the eKL summand*

$$\operatorname{tr}(\hat{T}_1(f)\hat{T}_2^{-1}(f)) - \log\det(\hat{T}_1(f)\hat{T}_2^{-1}(f))$$

in time complexity $O(1)$.

Furthermore, the complexity of calculating a test statistic $Z_N(\hat{\boldsymbol{T}}_1, \hat{\boldsymbol{T}}_2)$ where $\hat{\boldsymbol{T}}_2$ is a constrained weighted estimated periodogram to be calculated given some constraining graphical model (V, E_k), is $O(Np^2)$.

Proof. By considering algorithm 2 with l_k iterations, we see that calculation of $\hat{\boldsymbol{T}}_2$ for all Fourier frequencies is $O(Np^2 l_k)$. We also see the calculation of the eKL under our assumption is $O(N)$. Hence, treating l_k as constant at each step, calculation of the test statistics is $O(Np^2)$. □

Remark 35.

1. In reality, the calculation of the eKL summand is $O(p^3)$ with the determinant being the dominant function in terms of complexity (calculated using LU decomposition). However, the p^3 term has a very small constant relative to the Np^2 term and the time taken only starts to behave like p^3 for very large values of p.

2. Combining propositions 16 and 17, implies the main loop of both algorithms for fixed N and under the eKL $O(1)$ assumption, has complexity

$$T = \begin{cases} O(p^6) & \text{for Matsuda's algorithm;} \\ O(p^4) & \text{for the MHT.} \end{cases} \tag{2.6}$$

3. The overheads for calculating the initial weighted periodogram and performing the multiple hypothesis tests are $o(p^4)$, so do not factor into either of the big O complexities above.

4. We denoted the total iterations by l_k to signify the fact that in Matsuda's algorithm, more iterations are needed as k increases in order to

get a good estimate from algorithm 2. In the MHT case we treat $k = 0$ and here $l_0 = 1$ as there is an exact solution in this case.

Small Dimension Comparisons

For small values of p in which it was possible to run Matsuda's algorithm in a reasonable time, [85] provided a comparison of the timings and the accuracy of the Matsuda and MHT algorithm. We look at these comparisons in this section.

Timing

Figure 2.2 from [85] illustrates the comparison between the time taken to run each algorithm. The details are as follows, where T_1 represents the time taken for the Matsuda algorithm and T_2 the time taken for the MHT algorithm

(a) A plot of $T_1^{1/6}$ vs p for Matsuda's algorithm with $N = 1024$, $M = 32$. $T_1^{1/6}$ is close to linear with respect to p.

(b) A plot of $T_2^{1/4}$ vs p for the MHT algorithm with $N = 1024$, $M = 32$. $T_2^{1/4}$ is close to linear with respect to p.

(c) The ratio T_1/T_2 vs p (where T_1 and T_2 are from the above figures) showing the increase in computation time for Matsuda's algorithm compared to the MHT, with respect to p.

(d) A plot of T_2 vs MN for the MHT algorithm, where $p = 5$, $M = N/32$ and N is increased from 200 to 9000. The plot shows that as n increases, the plot looks fairly linear.

In all the above, when specified with p and N, observations were simulated from a $\text{VAR}_p(1)$ model created according to the scheme in section 1.5.2 and the time to compute each algorithm was recorded using $\alpha = 0.05$, the M provided, $w_k = \cos\left(\frac{\pi k}{2M}\right)$ and Holm critical levels for the MHT algorithm.

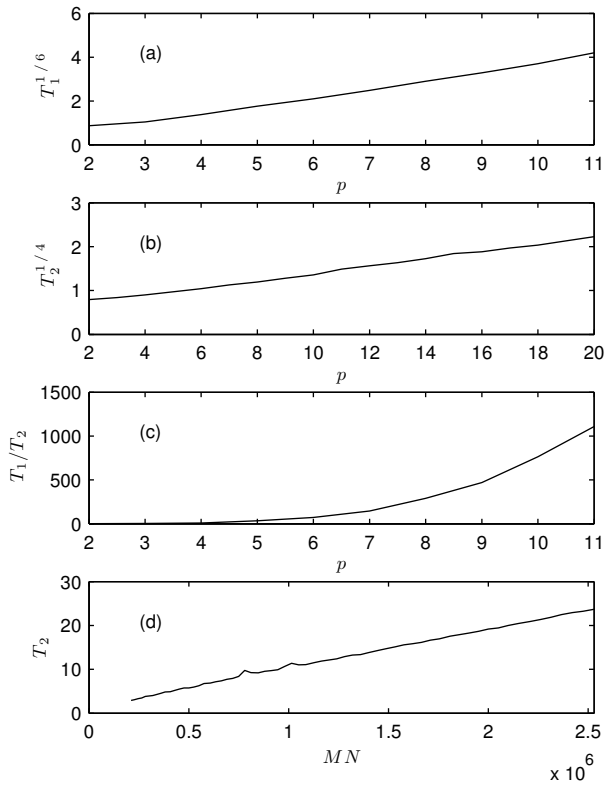

Figure 2.2: Calculation timings in seconds: (a) T_1 for Matsuda's algorithm, to the one-sixth power, versus p, (b) T_2 for the MHT, to the one-quarter power, versus p, (c) the ratio of computation times T_1/T_2 versus p, and (d) T_2 for the MHT versus MN. Here $N = 1024, M = 32$.

Power

We now look at the comparison between the power of both algorithms as reported in [85]. They empirically show that the stepwise procedure in [59] does not improve the power of the testing procedure over the single step multiple hypothesis test.

Model 1

The first model considered is the VAR process specified in model B. The following combinations of (N, M) were used $(512, 16)$, $(1024, 32)$, $(2048, 64)$ for both cases $x = 0$ and $x = 0.1$. All the results were based on 600 separate

Edge	Average	Standard Error
(1,2)	26.93	4.57
(1,3)	37.94	5.25
(1,4)	12.55	3.10
(1,5)	41.39	5.63
(2,3)	**0.25**	**1.08**
(2,4)	33.21	5.03
(2,5)	**0.34**	**1.05**
(3,4)	**1.00**	**1.21**
(3,5)	13.40	3.39
(4,5)	15.39	3.68

Table 2.3: Average and standard error of values of the Model B ($x = 0$) test statistic Z_N^i for each edge test with $N = 2048, M = 64$.

time series generated according to model B. The only edges considered were $\{2, 3\}$, $\{2, 5\}$ and $\{3, 4\}$ as other edges being rejected occurred so infrequently it wasn't worth considering them (illustrated in table 2.3). As noted by the authors, test statistics from the other edges are essentially treated as infinite.

Figure 2.3 from [85] shows the comparison between both algorithms. To get the results for the MHT, the α value was varied between 0 and 0.5 in steps of 0.00125. To get the results for Matsuda's algorithm, a parameter β as varied between 0 and 0.5 in steps of 0.00125.The transformation $\alpha = \beta^5$ was then used as this allowed more values to be concentrated around zero which gave a better grid for the empirical FWER. The FWER and effective power in figure 2.3 were calculated as follows

1. FWER is the proportion of replications in which a true null hypothesis was rejected i.e. an edge was included that shouldn't have been.

2. Effective power was the proportion of replications in which the hypothesis edge $\{3, 4\}$ should not be included in the model, was correctly rejected in the $x = 0$ case. In the $x = 0.1$ case it was the proportion of replications in which hypotheses edge $\{2, 3\}$ and $\{3, 4\}$ not being edges were both correctly rejected.

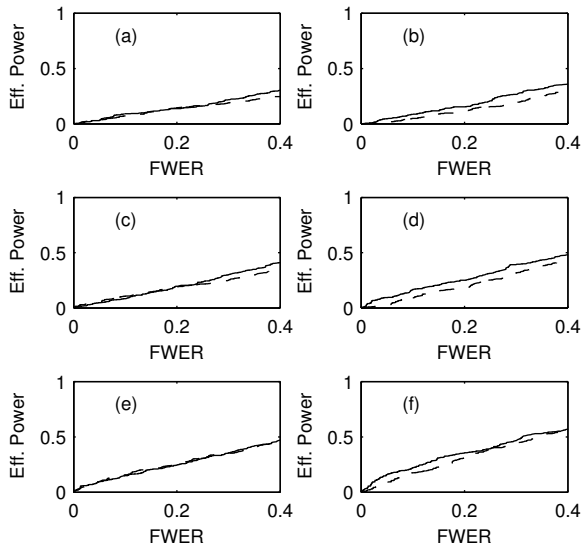

Figure 2.3: FWER versus effective power for the MHT (solid lines) and Matsuda's algorithm (dashed line) for Model B, (1.4), with (a) $N = 512, M = 16, x = 0$ (b) $N = 512, M = 16, x = 0.1$, (c) $N = 1024, M = 32, x = 0$ and (d) $N = 1024, M = 32, x = 0.1$, (e) $N = 2048, M = 64, x = 0$ and (f) $N = 2048, M = 64, x = 0.1$.

The results for $x = 0$ are in figure 2.3 (a), (c) and (e), while the results for $x = 0.1$ are in figure 2.3 (b), (d) and (f). The MHT is represented by a solid line and Matsuda's algorithm by a dashed line. We can see in all cases the effective power is very similar and that the MHT performs at least as well as Matsuda's algorithm.

Model 2

The second empirical test in [85] was based on model A. The missing edges are $\{\{2,3\},\{2,5\},\{3,4\}\}$ and $\{3,5\}$ is a borderline edge as demonstrated in table 2.4. Using the previous method for generating empirical FWER and effective power values, where this time the effective power was the proportion of 600 replications where the hypothesis $\{3,5\}$ is not an edge in the model was correctly rejected. Simulations were run for (N, M) pairs $(512, 16), (1024, 32), (2048, 64)$ and once again we see in figure 2.4 that the effective power is very similar.

Edge	Average	Standard Error
(1,2)	50.00	5.93
(1,3)	15.74	3.52
(1,4)	22.02	4.34
(1,5)	64.12	6.79
(2,3)	**0.29**	**1.06**
(2,4)	15.66	3.38
(2,5)	**0.27**	**1.06**
(3,4)	**0.32**	**1.05**
(3,5)	**3.86**	**1.95**
(4,5)	66.09	6.61

Table 2.4: Average and standard error of values of the model 2 test statistic Z_N^i for each edge test with $N = 2048$ and $M = 64$.

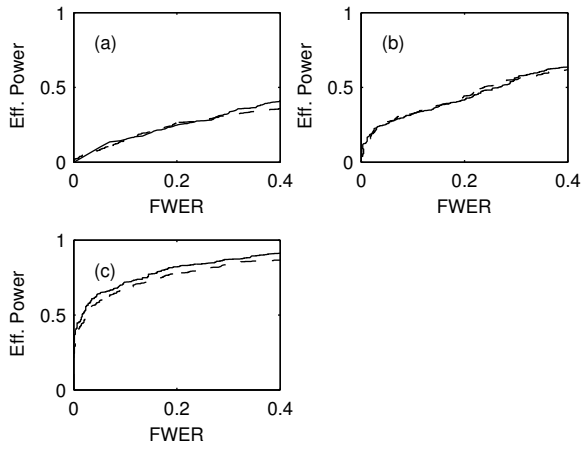

Figure 2.4: FWER versus effective power for the MHT (solid lines) and Matsuda's algorithm (dashed line) for Model A, (1.3), and (a) $N = 512, M = 16$ (b) $N = 1024, M = 32$ and (c) $N = 2048, M = 64$.

MHT Algorithm for Larger Dimensions

One of the key benefits of the MHT approach is its ability to run in a reasonable time for larger values of p. While it is no longer possible to compare with Matsuda's algorithm for the larger dimensions, we can instead report timings and the type I and II errors for individual tests. It is also possible

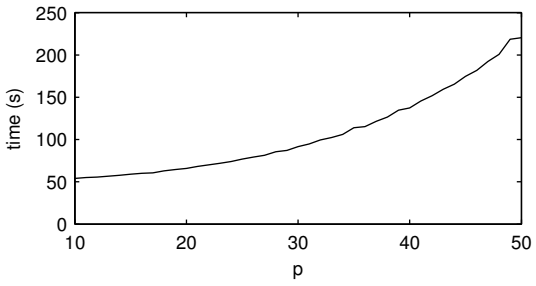

Figure 2.5: Calculation timings in seconds for the MHT algorithm as p varies from 10 to 50. Here $N = 2048$ and $M = 128$.

to demonstrate that the MHT algorithm can be parallelised for even better computational efficiency. All results were based on a $\text{VAR}_p(1)$ model generated according to the scheme in 1.5.2, resulting in true graphical models with around 36% of potential edges being present.

Timings

Figure 2.5 from [85] shows the timings for the MHT algorithm ranging from $p = 10 : 50$. In all cases, parameter values of $N = 2048$ and $M = 128$ were used. As noted by the authors, a crude scaling up of the $p = 50$ timing of 220s for the MHT by multiplying by p^2, gives an estimated time of over 6 days for Matsuda's algorithm.

Accuracy

In the higher dimension case we investigate the accuracy of the MHT algorithm by considering type I and type II errors when estimating a graphical model. A type I error is produced when an edge is included that shouldn't be and a type II error is an edge not included that should be. The type I percentage error is then (number of true missing edges estimated as not missing)/(number of true missing edges) and the type II percentage error is (number of true non-missing edges estimated as missing)/(number of true non-missing edges).

	$p = 10:29$	$p = 30$	$p = 30:50$
Type I	2.2	3.0	4.1
Type II	1.3	2.4	2.9

Table 2.5: Average type I and II percentage errors

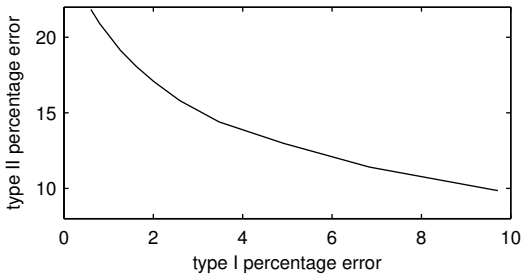

Figure 2.6: Type I and II percentage errors for $p = 150$ as α is varied. Here $N = 2048, M = 512$.

Table 2.5 from [85] shows the average type I and II percentage errors using $\alpha = 0.05$ where the average was taken over

1. Each model for $p = 10:29$.
2. 100 repeats for $p = 30$.
3. Each model for $p = 30:50$.

Remark 36. The increasing type I and II errors with respect to increasing p is likely due to the relatively worse weighted periodogram estimate for fixed N, M values.

Figure 2.6 from [85] shows the relationship between type I and II errors as α is varies for $p = 150$, $N = 2048$ and $M = 128$.

Parallelisability

As noted in [85], the MHT algorithm can be split into 3 steps

1. Computing the weighted periodogram and its inverse
2. Calculating the $p(p-1)/2$ test statistics
3. Performing the multiple hypothesis test

Step 3 is negligible in terms of time taken. Step 1 can be calculated fairly efficiently as it consists of a fast Fourier transform (FFT) and a subsequent filtering with a weight sequence. The time taken does however depend on p, N and M. It can be parallelised in the sense that the above common functions have been implemented (by others) to take advantage of GPUs and multicore CPUs.

Step 2 however, the most compute intensive step, can be almost perfectly parallelised as there is no dependency between test statistic calculation. Figure 2.7 from [85] illustrates this by assigning the calculation of each test statistic to a different core of a multicore CPU. The result is an almost linear relationship between time and 1/(number of cores). This indicates that besides some constant overheads, the algorithm as expected is almost perfectly parallelisable and can be easily scaled in terms of efficiency with advances in parallel computing hardware.

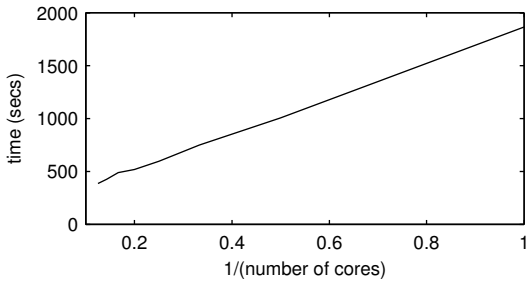

Figure 2.7: Calculation timings in seconds for the MHT algorithm for $p = 60, N = 2048, M = 512$ against the reciprocal number of cores, as the number of cores varies from 1 (right of plot) to 8 (left).

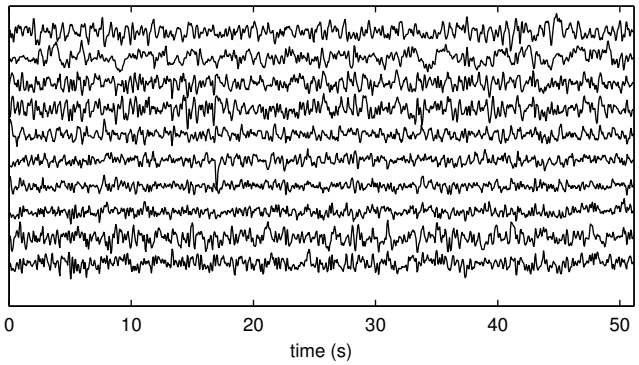

Figure 2.8: Ten channel EEG time series for one of the negative-syndrome patients.

Application to EEG Data

We now look at the results from [85] when applying the MHT algorithm to real EEG data. The dataset is explained in [61]. Essentially there are 33 male patients, 19 of which are diagnosed with negative syndrome schizophrenia and 24 are used as controls. The EEG data is collected when the subjects are resting with eyes closed. The dimension of the time series of EEG data for each patient is 10. Figure 2.8 from [85] shows an example time series for a patient.

Figure 2.9 from [85] shows the percentage of negative syndrome schizophrenia patients (heavy line) and controls (thin line) who were found to have an edge existing in their graphical model, where the edge is indexed by the same connection index as [61]. It is noted that 3/4 of the connections appear with lower percentage for the controls, a result consistent with [61].

Conclusion and Future Work

Matsuda introduced a stepwise algorithm for estimating the graphical model of a stochastic process given an observed time series from the process. The approach however is very computationally expensive and only practical for small dimensional time series.

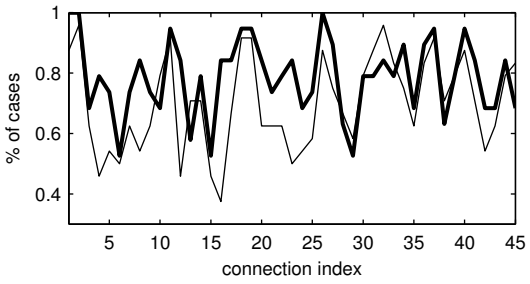

Figure 2.9: Percentage of negative-syndrome patients (heavy line) and controls (thin line) exhibiting a specified connection. ($N = 1024, M = 20, \alpha = 0.01$).

The approach in [85] was based on a multiple hypothesis testing procedure using Matsuda's statistic. The computational complexity of the MHT algorithm was $O(p^4)$ as opposed to $O(p^6)$ for Matsuda's. The algorithm was shown to lend itself very well to parallel computing. Also, as it is based on multiple hypothesis testing, the measure of error rate being controlled by the algorithm is arguably more relevant.

Mentioned in [85] is the fact that it may be more intuitive for the type I error to be deleting an edge that should be in the true graphical model. This makes sense if we treat lack of edges as constraints, so the null hypothesis would normally be that no constraint exists and we need to see evidence to overturn this. However this would require the distribution of Matsuda's test statistic being accurately known under the alternate hypothesis which is not the case.

Also noted in [85] is that the Holm approach used to select critical levels in the multiple hypothesis test can be improved using an adaptive approach as in [34] to be more powerful. However when implemented, the authors found only small difference but commented it is worth further investigation.

We note that while Wolstenholme and Walden only focused on controlling the FWER using Matsuda's statistic, the multiple hypothesis testing framework can be easily modified to control other error rate measures such as the FDR and to use other, potentially better statistics.

Finally the parallelisation results only came from parallelising on multicore

CPUs. Much larger speedups may be possible if specific high performance computing hardware such as GPUs or FPGAs were used.

3

Graph Matching

Introduction

We see in chapter 4 how to convert time series into undirected graphs with labelled nodes up to some unknown permutation of the nodes. Given two of these graphs, an interesting question is, fixing the first graph, what permutation of the second graph makes it as similar as possible to the first. This of course depends on how you measure similarity.

An intuitive method of measuring this similarity is to use the Frobenius norm between adjacency matrices. Minimising this value over all possible permutations of the nodes of the second graph is known as the graph matching problem (GMP) [55]. This problem is very well studied in the literature over many different fields. It is particularly well studied in the field of computer vision but has many other applications, ranging from circuit design to social network analysis. The problem itself can be split into exact graph matching, where one tries to find an exact isomorphism from one graph to another and inexact graph matching, where one aims to find how 'similar' two graphs are to one another.

For graphs with the same number of nodes, finding the eigenvalues of the

adjacency matrices is proven to be the optimal way to solve the exact graph matching problem [82]. However, exact graph matching is not of much use for noisy real world problems. In this chapter we concentrate solely on inexact graph matching and refer to it simply as graph matching from now on.

A paramount issue with graph matching is the fact the number of fixed node arrangements for a graph $G = (V, E)$ is $|V!|$. Therefore computation time for most optimal accuracy algorithms (that require the analysis of all possible permutations) is exponential in the number of nodes. These become infeasible as dimension increases and instead most methods are developed to find a balance between speed and accuracy.

In fact, it is not hard to show that solving the GMP is equivalent to solving a quadratic assignment problem (QAP) which is known to be NP-hard in the general case. Hence exact solutions to the GMP can be computationally intractable even when dealing with graphs with moderately sized dimensions. Note also that the GMP in its basic sense does not take the node labels into account. It is however alluded to in [90] and we show how it can be done in chapter 7.

Spectral methods based on the graphs' adjacency or Laplacian matrix are very popular when aiming to find a good solution to the GMP. Xu and King [89] presented a PCA approach as an approximation to the minimum Frobenius norm over all vertex permutations, between two graphs adjacency matrices. Knussow et al [48] used a method based on a projection onto suitably selected eigenvectors of the graph Laplacian. Both [90] and [88] solve relaxed versions of the GMP and then look to project the approximate solution back onto the space of permutation matrices. This chapter covers the main ideas behind these approaches and presents a graph matching algorithm based on [86].

The layout of the chapter is as follows, section 3.1 contains some preliminary results. Section 3.2 covers the basics of graph matching and section 3.3 presents an initial relaxation approach one can use. Section 3.4 gives an overview of the method used in [88]. We first look at a simple method greedy optimisation for improving the relaxed GMP solution in section 3.5 and a

more advance technique in 3.6 which uses the ideas from [86]. We finish with section 3.7 which contains results from [86] comparing against the method in [90] and end with concluding remarks and ideas for further work.

Preliminaries

Frank-Wolf Algorithm

The Frank-Wolfe algorithm [28] is used for finding global minima of convex quadratic functions with linear constraints on the variables. Note it can be used on non-convex functions but the minimum it converges to cannot be guaranteed to be a global minimum. We present the algorithm where the linear constraints are that our matrix solution must be a doubly stochastic matrix.

Algorithm 5 Frank-Wolfe algorithm for doubly stochastic solutions

Require: Objective function $f : \mathbb{R}^{n\times n} \to \mathbb{R}$, a stopping criterion and initialisation method
Ensure: $\hat{Q}$ is a doubly stochastic matrix

1: Initialise $Q^{(0)}$ according to the initialisation method
2: $i \leftarrow 0$
3: **while** Stopping criteria not met **do**
4: Compute gradient $\nabla f(Q^i)$
5: Compute search direction $W^{(i)} = \arg\min_{Q\in\mathbb{D}} \operatorname{tr}(\nabla f(Q^i)Q)$ by the Hungarian algorithm
6: Compute the step size $\alpha^{(i)} = \arg\min_{\alpha\in[0,1]} f(Q^{(i)} + \alpha W^{(i)})$ (exactly solvable as the objective is a quadratic function of α
7: $Q^{(i+1)} \leftarrow Q^{(i)} + \alpha^{(i)} W^{(i)}$
8: $i \leftarrow i + 1$
9: **end while**
10: $\hat{Q} \leftarrow Q^{(i)}$
11: **return** $\hat{Q}$

Variance Adaption

This is a variance adaption scheme for sampling points in $\mathbb{R}^p$ as given in [86]. Given $\boldsymbol{x}_t \in \mathbb{R}^p$, consider sampling $\boldsymbol{x}_{t+1}$ from $N(\boldsymbol{x}_t, \sigma_t I_p)$ where we have control over the σ_t^2 value. After sampling $\boldsymbol{x}_{t+1}$, project back onto the unit hypersphere i.e. $\boldsymbol{x}_{t+1} \leftarrow \frac{\boldsymbol{x}_{t+1}}{||\boldsymbol{x}_{t+1}||}$. Now consider permutation matrices $P(x_t)$ and $P(x_{t+1})$ that are some functions of x_t and x_{t+1}. Define the distance between them as

$$\Delta t = ||P(\boldsymbol{x}_t) - P(\boldsymbol{x}_{t+1})||_F^2. \tag{3.1}$$

Our goal is to choose σ_t^2 such that Δt follows some given target function. Mathematically we write this as follows. Assume

$$\Delta_t = \bar{\Delta}_t(\sigma_t^2) + \epsilon_t, \tag{3.2}$$

where $\{\epsilon_t\}$ is some zero mean noise process and $\bar{\Delta}(\sigma_t^2)$ is a deterministic function i.e. $\mathbb{E}(\Delta_t) = \bar{\Delta}(\sigma_t^2)$. Now let f_t be the indexed values of the target function we want Δ_t to follow. Given we know $\bar{\Delta}(\cdot)$, we choose our σ_t^2 at time t by solving

$$\sigma_t^2 = \arg\min_{\tilde{\sigma}_t^2 > 0} |f_t - \bar{\Delta}(\tilde{\sigma}_t^2)|. \tag{3.3}$$

Fitting $\bar{\Delta}$

Given some previous observations of Δ_t and σ_t^2, we must learn an appropriate function to approximate $\bar{\Delta}$. Firstly write $y_t = \log(\sigma_t^2)$, as $\sigma_t^2 > 0$. Instead of learning an estimate of $\bar{\Delta}$ directly, we aim to learn $\tilde{\Delta}(y_t)$, an estimator of $\bar{\Delta}(\exp(y_t))/\bar{\Delta}_{\max}$, where $\bar{\Delta}_{\max}$ is the maximum value of $\bar{\Delta}(\cdot)$. To do this, we learn $\tilde{\Delta}(y_t)$ via logistic regression using previous values of $\Delta_t/\bar{\Delta}_{\max}$ and y_t.

To approximate $\bar{\Delta}_{\max}$, we sample uniformly on the unit hypersphere $\boldsymbol{z}_1, \ldots \boldsymbol{z}_M$ and set

$$\Delta_{\max} = \frac{1}{M} \sum_{i=1}^{M} ||P(\boldsymbol{x}_0) - P(\boldsymbol{z}_i)||_F^2, \tag{3.4}$$

where $\boldsymbol{x}_0$ is some initial point in the variance adaption method. Then we use $\Delta_{\max}$ as an estimate for $\bar{\Delta}_{\max}$. Note that this works well if points far away in $\mathbb{R}^p$ result in permutation matrices $P(\cdot)$ that also have a large distance between them. This holds true in our main application, although if it didn't we would suggest finding a different way to estimate $\bar{\Delta}_{\max}$.

Pre-Samples

When we first begin, we have no observations to use to learn $\tilde{\Delta}$. Therefore we generate an initial set of pre-samples $y_{-L}, \ldots, y_{-1}$. To generate these pre-samples we use the following idea. Choose some small $\epsilon > 0$ and sample y_i such that $\Delta_i / \Delta_{\max} \in [\epsilon, 1 - \epsilon]$. The reason behind this is to avoid having all samples falling within $[0, \epsilon]$ or $[1 - \epsilon, 1]$ which would be inadequate for fitting a logistic regression model. We use algorithm 6 to generate the pre-samples.

Algorithm 6 Generate pre-samples for variance adaption algorithm

Require: $\Delta_{\max}$, initial point $\boldsymbol{x}_0$, $\epsilon > 0$ and $\delta > 1$

$s \leftarrow 1$
\# Get first bound y_a
while True **do**
    Sample $\boldsymbol{y}_a$ from $N(0, s)$
    Sample $\boldsymbol{x}$ from $N(\boldsymbol{x}_0, \sqrt{\exp(y_a)}I))$
    $\boldsymbol{x} \leftarrow \boldsymbol{x}/||\boldsymbol{x}||$
    $\Delta_a \leftarrow ||P(\boldsymbol{x}_0) - P(\boldsymbol{x})||_F^2$
    if $\Delta_a/\Delta_{\max} \notin [\epsilon, 1-\epsilon]$ **then**
        break
    else
        $s \leftarrow \delta s$
    end if
end while
\# Get second bound y_b
while True **do**
    Sample $\boldsymbol{y}_b$ from $N(0, s)$
    Sample $\boldsymbol{x}$ from $N(\boldsymbol{x}_0, \sqrt{\exp(y_b)}I))$
    $\boldsymbol{x} \leftarrow \boldsymbol{x}/||\boldsymbol{x}||$
    $\Delta_b \leftarrow ||P(\boldsymbol{x}_0) - P(\boldsymbol{x})||_F^2$
    if $[\min(\Delta_a, \Delta_b)/\Delta_{\max}, \max(\Delta_a, \Delta_b)/\Delta_{\max}] \supset [\epsilon, 1-\epsilon]$ **then**
        break
    else
        $s \leftarrow \delta s$
    end if
end while
\# Order bounds
$\texttt{temp} \leftarrow \{y_a, y_b\}$
$y_a \leftarrow \min(\texttt{temp})$
$y_b \leftarrow \max(\texttt{temp})$
\# Sample pre-samples
for $i = 1 : L$ **do**
    Sample y_{-i} from $U[y_a, y_b]$
    Sample $\boldsymbol{x}$ from $N(\boldsymbol{x}_0, \sqrt{\exp(y_{-i})}I)$
    $\Delta_i \leftarrow ||P(\boldsymbol{x}_0) - P(\boldsymbol{x})||_F^2$
    if $\Delta_i/\Delta_{\max} < \epsilon$ **then**
        $y_a \leftarrow y_{-i}$
    end if
    if $\Delta_i/\Delta_{\max} > 1 - \epsilon$ **then**
        $y_b \leftarrow y_{-i}$
    end if
end for
Return $y_{-1}, \ldots, y_{-L}$

Learning $\tilde{\Delta}$

Given initial pre-samples $y_{-1}, \ldots, y_{-L}$ we sample $\boldsymbol{x}_{-i}$ from $N(\boldsymbol{x}_0, \sqrt{\exp(y_{-i})}I)$ and calculate

$$\Delta_{-i} = ||P(\boldsymbol{x}_0) - P(\boldsymbol{x}_{-i})||_F^2.$$

This gives independent variables

$$y_{-1}, \ldots, y_{-L}$$

and dependent variables

$$\Delta_{-1}, \ldots, \Delta_{-L}$$

which we use to find $\tilde{\Delta}$ using a logistic regression. Note that as we continue to run our algorithm we receive more information from σ_t and Δ_t. To continue to make use of this, we have a parameter T such that when $t \mod T = 0$, we re-learn $\tilde{\Delta}$ but now using $y_{-L}, \ldots, y_{-1}, y_0, \ldots, y_t$ and $\Delta_{-L}, \ldots, \Delta_{-1}, \Delta_0, \ldots, \Delta_t$.

Figures 3.1 and 3.2 from [86] show an example of our variance adaption logistic regression learning of $\tilde{\Delta}$ (which we can see is a reasonably good fit) and the tracking of the target function. The target function used was $f_t = \Delta_{\max}[1-(t/1000)^{0.6}]$ with re-learning parameter $T = 100$. We point out the importance of the incremental updating using T as in this case, initially Δ_t moves away from f_t but is quickly corrected when new samples are added to the regression. In this case we used a function for the permutation matrices that we will see later, for $Q \in \mathbb{R}^{p \times p}$,

$$P(\boldsymbol{x}) = \arg\min_{P \in \mathbb{P}} ||Q\boldsymbol{x} - P\boldsymbol{x}||_F^2.$$

Graph Matching

Given graphs $G_A, G_B \in \mathbb{G}_p$ with matrices $A, B \in \mathbb{R}^{p \times p}$ such that $A = A(G_A)$ and $B = A(G_B)$, then the aim of graph matching based on adjacency matrices

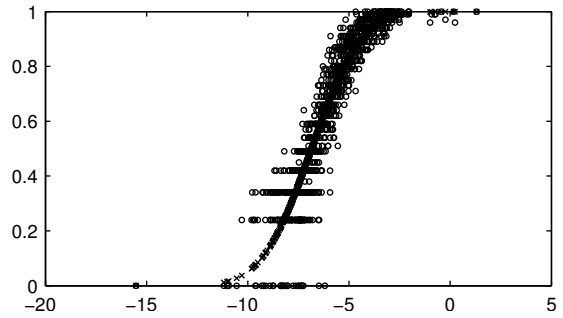

Figure 3.1: Illustration of learning $\tilde{\Delta}(y_t)$. The circles show $\Delta_t/\Delta_{\max}$ (vertical) against y_t (horizontal). The crosses give the fitted logistic curve for $\tilde{\Delta}(y_t)$.

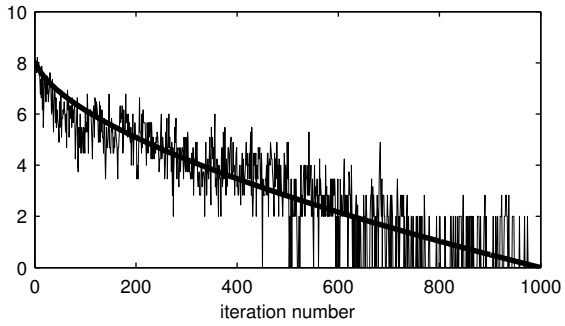

Figure 3.2: Δ_t (hashed line) versus f_t (thick curve), against t.

in a general form is to find

$$P^* = \arg\min_{P \in \mathbb{P}} ||A, P^T B P||, \tag{3.5}$$

where $\mathbb{P}$ is the set of $p \times p$ permutation matrices and $||\cdot, \cdot||$ is a matrix distance measure. The distance is often chosen to be the Frobenius norm of the difference of the two adjacency matrices, giving the equation

$$P^* = \arg\min_{P \in \mathbb{P}} ||A - P^T B P||_F^2. \tag{3.6}$$

Relaxation

Solving (3.6) is usually a very computationally expensive operation and the time taking becomes prohibitive even for moderately sized p. Current methods such as [90] and [88], solve a relaxed version of (3.6) by optimising over a superset of the permutation matrices $X \supset \mathbb{P}$ and solve

$$Q^* = \arg\min_{Q \in X} f_{A,B}(Q) \tag{3.7}$$

such that

$$\arg\min_{Q \in \mathbb{P}} ||A - Q^T B Q||_F^2 = \arg\min_{Q \in \mathbb{P}} f_{A,B}(Q),$$

Remark 37. Some common choices of $f_{A,B}$ are

1. The standard GMP formulation

$$f_{A,B}(Q) = ||A - Q^T B Q||_F^2,$$

2. The formulation used in [90]

$$f_{A,B}(Q) = ||QA - BQ||_F^2,$$

3. The formulation used in [88]

$$f_{A,B}(Q) = -\operatorname{tr}(AQB^TQ^T).$$

Another interesting choice that we haven't yet seen studied would be to extend the matrices to non-linear transformations using,

$$f_{A,B}(Q) = ||B - Q\{Q\{A^T\}^T\}||_F^2,$$

for some general transformation

$$Q : \mathbb{R}^p \to \mathbb{R}^p$$

whose extension to

$$Q : \mathbb{R}^{p\times p} \to \mathbb{R}^{p\times p}$$

is defined as

$$Q\{X\} = [Q\{X_1\}, Q\{X_2\}, \ldots, Q\{X_p\}],$$

where X_i is the ith column of X. Clearly taking Q as a matrix turns this into standard matrix multiplication and means that our condition

$$\arg\min_{Q\in\mathbb{P}} ||A - Q^T BQ||_F^2 = \arg\min_{Q\in\mathbb{P}} f_{A,B}(Q)$$

is satisfied.

The most common choice of X, the set to relax over, is the set of doubly stochastic matrices $\mathbb{D} = \{Q \in \mathbb{R}^{p\times p} : Q\mathbf{1} = Q^T\mathbf{1} = \mathbf{1} \text{ and } Q \succeq 0\}$. For $f_{A,B}(Q) = ||A - Q^T BQ||_F^2$ this results in (3.7) being a convex optimisation and can be efficiently solved by the Frank-Wolfe algorithm. Note that if $f_{A,B}(Q) = -\operatorname{tr}(AQB^TQ^T)$, the optimisation is no longer convex but Frank-Wolfe can still be used to find a local optima. This will be discussed in more detail later.

We could also try relaxing over the orthogonal matrices $\mathbb{Q} = \{Q \in \mathbb{R}^{n\times n} : QQ^T = I\}$. Once again (3.7) can be efficiently solved, this time using the singular value decomposition. However as mentioned in [86] this does not have a unique solution and in fact has at least 2^p solutions. If the eigenvalues of either A or B are not distinct, it has over 2^p solutions.

Projection

The intuition behind the relaxation is that because $X \supset \mathbb{P}$, a permutation that is close to the solution Q^* of (3.7), should be a good candidate as an approximate solution to the GMP.

A commonly used projection (i.e. in [88]) is

$$P^* = \arg\min_{P\in\mathbb{P}} ||Q^* - P||_F = \arg\max_{P\in\mathbb{P}} \operatorname{tr}(Q^T P) \tag{3.8}$$

the closest permutation matrix to Q^* with respect to Frobenius norm. This can be efficiently solved by the Hungarian algorithm in $O(p^3)$ as

$$\max_{P \in \mathbb{P}} \operatorname{tr}(Q^T P)$$

is a linear assignment problem [15].

Another method of projecting back to the permutation matrices, used by Wolstenholme and Walden [86], is to use the following projection

$$P^* = \arg\min_{P \in \mathbb{P}} ||Q^* x - Px||_F \tag{3.9}$$

for some $x \in \mathbb{R}^p$. By sampling the x from a suitably chosen probability distribution and applying (3.9) to each, you obtain an empirical distribution over the permutation matrices. This can then be explored to find potential solutions of (3.6) using an optimisation technique such as simulated annealing. We look at this approach in more detail later.

Convex Relaxation

An initial idea to solve a relaxed form of the GMP was QCV [68], a convex relaxation. This is defined by setting

$$f_{A,B}(Q) = ||QA - BQ||_F^2 \tag{3.10}$$

and relaxing over the set of double stochastic matrices $\mathbb{D}$. For a given $A, B \in \mathbb{R}^{p \times p}$, we can solve

$$\arg\min_{Q \in \mathbb{D}} f_{A,B}(Q) \tag{3.11}$$

using the Frank-Wolfe algorithm. Note that (3.10) is a convex relaxation so has a unique global minimum. However it is not guaranteed that projecting this minimum back onto the set of permutation matrices will lead to a

solution close to the actual minimum of the non-relaxed GMP. In fact as commented on in [90] and [88], the method often performs rather poorly.

Algorithm

The algorithm including both the Frank-Wolfe optimisation and projection step is as follows, writing $f_{A,B}$ as f for convenience, is given in algorithm 7.

Algorithm 7 QCV for finding an approximate solution to the GMP

Require: Adjacency matrices A, B, a stopping criterion and an initialisation method (see below)
Ensure: $\hat{P}$ is a permutation matrix

1: Initialise $Q^{(0)}$ according to the initialisation method
2: $i \leftarrow 0$
3: **while** Stopping criteria not met **do**
4: Compute gradient $\nabla f(Q^i) = -2BQ^{(i)}A^T - 2B^TQ^{(i)}A + 2B^TQ^{(i)}(Q^{(i)})^TBQ^{(i)} + 2BQ^{(i)}(Q^{(i)})^TB^TQ^{(i)}$
5: Compute search direction $W^{(i)} = \arg\min_{Q\in\mathbb{D}} \operatorname{tr}(\nabla f(Q^i)Q)$ by the Hungarian algorithm
6: Compute the step size $\alpha^{(i)} = \arg\min_{\alpha\in[0,1]} f(Q^{(i)} + \alpha W^{(i)})$ (exactly solvable as the objective is a quadratic function of α)
7: $Q^{(i+1)} \leftarrow Q^{(i)} + \alpha^{(i)}W^{(i)}$
8: $i \leftarrow i + 1$
9: **end while**
10: Compute $\hat{P} = \arg\min_{P\in\mathbb{P}} -\operatorname{tr}(Q^{(i)}P^T)$ by the Hungarian algorithm
11: **return** $\hat{P}$

Fast Approximate Quadratic Programming for Graph Matching

In [88] they cast the graph matching problem as a quadratic assignment problem. They then solved a relaxed version of the QAP to get an approximate solution to the GMP and named the algorithm fast approximate quadratic programming for graph matching (FAQ). In particular they use

$$f_{A,B}(Q) = -\operatorname{tr}(AQB^TQ^T) \tag{3.12}$$

relaxed over the set doubly stochastic matrices.

The below proposition shows the relationship between the GMP and the QAP.

Proposition 18. *The solution to the graph matching problem with adjacency matrices $A, B \in \mathbb{R}^n$ is equivalent to the solution of the quadratic assignment problem with matrices $-A$ and B.*

Proof. This results from expanding the GMP equation as follows

$$\arg\min_{P\in\mathbb{P}} ||A - P^T BP||_F = \arg\min_{P\in\mathbb{P}} \operatorname{tr}[(A - P^T BP)^T (A - P^T BP)]$$

$$= \arg\min_{P\in\mathbb{P}} \operatorname{tr}(A^T A) - 2\operatorname{tr}(AP^T B^T P) + tr(P^T B^T PP^T BP)$$

$$= \arg\min_{P\in\mathbb{P}} -\operatorname{tr}(AP^T B^T P)$$

We note that $\operatorname{tr}(P^T B^T PP^T BP) = \operatorname{tr}(P^T B^T BP) = \operatorname{tr}(B^T B)$ as firstly permutation matrices are orthogonal and secondly the affect of the transformation $P^T XP$ for $X \in \mathbb{R}^n$ on the leading diagonal is simply to swap the values, hence leaving the trace unaffected. □

Local Minima

We can write the problem when relaxed over the set of doubly stochastic matrices as

$$\arg\min_{Q\in\mathbb{D}} -\operatorname{tr}(AQB^T Q^T). \tag{3.13}$$

Unlike QCV, this is not longer necessarily a convex optimisation. While it is still then NP-hard to find a global optimum, the relaxation allows the use of continuous optimisation to find a local optima which can then be projected onto $\mathbb{P}$, giving an approximate solution to the GMP. Multiple initial points can be used when solving (3.13) to help to alleviate this issue. Note that despite the non-unique solutions, FAQ tends to outperform QCV in almost all difficult problems in QAPLIB [14], a widely used benchmark for QAP solvers.

Initial Points

As mentioned previously, the algorithm output is sensitive to the choice of initial point. A number of different methods can be used:

1. An intuitive point to use if we are only going to use a single run of the algorithm would be
$$J = \mathbf{1}\mathbf{1}^T/p$$
the completely flat non-informative doubly stochastic matrix.

2. If we are going to use multiple starts, it is necessary to add some randomness to the choice of the initial point. In this case [88] suggest using Sinkhorn balancing [74] to sample a random double stochastic matrix K and use as initial point
$$(J + K)/2.$$

3. A final option if we are just using a single start is to use the doubly stochastic matrix that is the pre-projected solution of QCV.

Algorithm

We can now present the full FAQ algorithm for finding an approximate solution to the GMP, writing $f_{A,B}$ as f for convenience, in algorithm 8.

Algorithm 8 FAQ for finding an approximate solution to the GMP

Require: Adjacency matrices A, B, a stopping criterion and an initialisation method (see below)
Ensure: $\hat{P}$ is a permutation matrix

1: Initialise $Q^{(0)}$ according to the initialisation method
2: $i \leftarrow 0$
3: **while** Stopping criteria not met **do**
4: Compute gradient $\nabla f(Q^i) = -AQ^{(i)}B^T - A^TQ^{(i)}B$
5: Compute search direction $W^{(i)} = \arg\min_{Q \in \mathbb{D}} \operatorname{tr}(\nabla f(Q^i)Q)$ by the Hungarian algorithm
6: Compute the step size $\alpha^{(i)} = \arg\min_{\alpha \in [0,1]} f(Q^{(i)} + \alpha W^{(i)})$ (exactly solvable as the objective is a quadratic function of α)
7: $Q^{(i+1)} \leftarrow Q^{(i)} + \alpha^{(i)}W^{(i)}$
8: $i \leftarrow i + 1$
9: **end while**
10: Compute $\hat{P} = \arg\min_{P \in \mathbb{P}} -\operatorname{tr}(Q^{(i)}P^T)$ by the Hungarian algorithm
11: **return** $\hat{P}$

Remark 38. It may be possible in certain cases to modify the projection step when performing relaxed graph matching to improve upon the output permutation matrix when simply using a direct projection (3.8). We investigate two methods, a 2-opt strategy applied to the permutation matrix after projection and a simulated annealing approach as a substitute for the direct projection step. Both bear some similarities, the former being a search directly in the permutation space and the latter being a transformation of the search into Euclidean space $\mathbb{R}^p$.

2-opt

The 2-opt strategy is a greedy method for improving an output permutation matrix with respect to some objective function by considering all possible swaps of pairs of its rows and updating it to the permutation matrix with the minimum objective value over the swaps. The algorithm is alluded to in [44] which was referenced by [69] as the source of the 2-opt algorithm. The actual algorithm is not explicitly included in [44], so we provide it here in

algorithm 9.

Algorithm 9 2-opt strategy for greedy optimisation

Require: Permutation matrix $P_0 \in \mathbb{R}^{p \times p}$, objective function $f : \mathbb{P} \to \mathbb{R}$

while True **do**
  for $i = 1 : p - 1$ **do**
    for $j = i + 1 : p$ **do**
      $P(i, j) \leftarrow$ Permutation matrix swapping rows i and j
      $C_{ij} \leftarrow f(P_{ij} P_0)$
    end for
  end for
  $\bar{i}, \bar{j} \leftarrow \arg\min_{i,j} C_{ij}$
  if $C_{\bar{i}\bar{j}} < f(P_0)$ **then**
    $P_0 \leftarrow P_{\bar{i}\bar{j}} P_0$
  else
    break
  end if
end while
return P_0

Dynamic Calculation

In the graph matching case, we do not have to re-calculate $f(P_{ij}P_0)$ from scratch for each i and j value. Let $P_0 \in \mathbb{P}$ be the output from a relaxed graph matching approach e.g. QCV. The objective function we know is written

$$f(P) = ||A - P^T B P||_F^2.$$

Given $f(P_0)$, we want to find the most efficient way of calculating $f(P_{ij}P_0)$. Well,

$$f(P_0) = \sum_{r=1}^{p} \sum_{s=1}^{p} (A_{rs} - (P^T B P)_{rs})^2.$$

This can be written as

$$f(P_0) = \sum_{r=1}^{p}\sum_{s=1}^{p}(A_{rs} - (P_0^T B P_0)_{rs})^2 I(r \notin \{i,j\}, s \notin \{i,j\}) + \sum_{r\in\{i,j\}}\sum_{s=1}^{p}(A_{rs} - (P_0^T B P_0)_{rs})^2 + \sum_{s\in\{i,j\}}\sum_{r=1}^{p}(A_{rs} - (P_0^T B P_0)_{rs})^2 - \sum_{r\in\{i,j\}}\sum_{s\in\{i,j\}}(A_{rs} - (P_0^T B P_0)_{rs})^2.$$

Therefore

$$\begin{aligned} f(P_0) - f(P_{ij}P_0) = & \\ & \sum_{r\in\{i,j\}}\sum_{s=1}^{p}(A_{rs} - (P_0^T B P_0)_{rs})^2 - (A_{rs} - (P_0^T P_{ij}^T B P_{ij} P_0)_{rs})^2 \\ & + \sum_{s\in\{i,j\}}\sum_{r=1}^{p}(A_{rs} - (P_0^T B P_0)_{rs})^2 - (A_{rs} - (P_0^T P_{ij}^T B P_{ij} P_0)_{rs})^2 \\ & + \sum_{r\in\{i,j\}}\sum_{s\in\{i,j\}}(A_{rs} - (P_0^T P_{ij}^T B P_{ij} P_0)_{rs})^2 - (A_{rs} - (P_0^T B P_0)_{rs})^2. \quad (3.14) \end{aligned}$$

Hence, we can efficiently evaluate $f(P_{ij}P_0)$ using $f(P_0)$ and summing $4p+4$ terms in (3.14)as opposed to p^2 terms if calculating the value from scratch.

A Sampling Strategy for Projecting to Permutation Matrices

Similarly to the 2-opt strategy, in this section we wish to improve our output relaxed graph matching permutation matrix. However in this case our improvement is a substitute for the direct projection step of graph matching as opposed to working with the permutation matrix after direct projection as the 2-opt strategy does. The rest of this section closely follows the work in [86].

Matrix Rounding

Given a matrix $Q \in X \supset \mathbb{P}$ we look to round/project it to some close $P \in \mathbb{P}$. As we saw previously this is normally done using the direct projection step solving the linear assignment problem

$$\arg\max_{P \in \mathbb{P}} \operatorname{tr}(Q^T P)$$

using the Hungarian algorithm in $O(p^3)$ time.

We instead consider an alternative rounding. For a given $\boldsymbol{x} \in \mathbb{R}^p$, solve

$$\arg\min_{P \in \mathbb{P}} ||Q\boldsymbol{x} - P\boldsymbol{x}||_F^2. \tag{3.15}$$

It is shown in theorem 2 this is a sorting problem it can be solved in $O(p \log p)$ time. Clearly (3.15) provides a different solution for different values of $\boldsymbol{x}$ and the method is based on exactly which $\boldsymbol{x}$ to choose to try to get a good solution to the GMP i.e. a solution close to the optimal solution.

Barvinok's Method

Barvinok showed in [4] a method for rounding an orthogonal Q to a permutation matrix by considering the action of Q on a vector $\boldsymbol{x} \in \mathbb{R}^p$ sampled from a Gaussian distribution.

Definition 41. *Let $\boldsymbol{r}(\boldsymbol{x}) \in \mathbb{Z}_+^p$ represent the ordering of vector $\boldsymbol{x}$ such that $\boldsymbol{r}(\boldsymbol{x})_i = j$ where x_j is the jth smallest value of $\boldsymbol{x}$.*

Example 4. If we let $\boldsymbol{x} = [3.1, 7.3, 2.4, 8.7]^T$ then we have $\boldsymbol{r}(\boldsymbol{x}) = [2, 3, 1, 4]^T$.

Representing Barvinok's idea using this notation, Q rounds to $P \in \mathbb{P}$ (given $\boldsymbol{x}$) if they both transform $\boldsymbol{x}$ so the sorting is in the same order i.e.

$$P\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{r}(Q\boldsymbol{x}). \tag{3.16}$$

We also note that permutation matrices always round to themselves as

$$P\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{r}(P\boldsymbol{x}). \tag{3.17}$$

If we consider the fact as Barvinok did, that the $\boldsymbol{x} \in \mathbb{R}^p$ are from a Gaussian distribution, then the above provides a way of projecting Q to a distribution of permutation matrices. Clearly the distribution can be sampled from by first sampling $\boldsymbol{x}$ and then solving (3.16).

Remark 39.
1. Nothing in the process requires Q to be orthogonal and it can be used for other classes of matrices, in particular doubly stochastic matrices.
2. The distribution that the vector $\boldsymbol{x}$ is sampled from does not have to be Gaussian. Of course the distribution chosen for $\boldsymbol{x}$ directly effects the implied distribution over the permutation matrices.

We now show as in [86], that this method from Barvinok is equivalent to the alternative rounding method (3.15).

Theorem 2. *Given $\boldsymbol{x} \in \mathbb{R}^p$ and $Q \in \mathbb{R}^{p \times p}$, any permutation matrix P solving (3.16) is also a solution to (3.15).*

Proof. We first show that

$$F(P) \stackrel{\text{def}}{=} ||\boldsymbol{a} - P\boldsymbol{b}||_F^2$$

is minimised when permutation matrix P sorts the vector $\boldsymbol{b}$ such that $a_i \leq a_j \Rightarrow (P\boldsymbol{b})_i \leq (P\boldsymbol{b})_j$ i.e., $\boldsymbol{r}(\boldsymbol{a}) = \boldsymbol{r}(P\boldsymbol{b})$.

The contribution to F at indices i and j is

$$(a_i - c_i)^2 + (a_j - c_j)^2,$$

where $\boldsymbol{c} \stackrel{\text{def}}{=} P\boldsymbol{b}$. Now,

$$\begin{aligned}(a_i - c_i)^2 &= (a_i - c_j + c_j - c_i)^2 \\ &= (a_i - c_j)^2 + (c_j - c_i)(2a_i - c_j - c_i).\end{aligned}$$

Similarly,

$$(a_j - c_j)^2 = (a_j - c_i)^2 + (c_i - c_j)(2a_j - c_j - c_i).$$

Therefore,

$$\begin{aligned}(a_i - c_i)^2 + (a_j - c_j)^2 &= (a_i - c_j)^2 + (a_j - c_i)^2 \\ &+ (c_j - c_i)(2a_i - 2a_j). \end{aligned} \tag{3.18}$$

We also know that if P is to be an optimal transformation, we must have

$$(a_i - c_i)^2 + (a_j - c_j)^2 \leq (a_i - c_j)^2 + (a_j - c_i)^2, \tag{3.19}$$

otherwise we can define $P^{'}$ such that $(P^{'}\boldsymbol{b})_k = (P\boldsymbol{b})_k$ for $k \neq i, j$ but $(P^{'}\boldsymbol{b})_i = (P\boldsymbol{b})_j$ and $(P^{'}\boldsymbol{b})_j = (P\boldsymbol{b})_i$. Clearly if (3.19) did not hold, $F(P^{'}) < F(P)$, contradictory to P being optimal.

Combining (3.18) and (3.19) gives $(c_j - c_i)(a_i - a_j) \leq 0$. Hence if $a_i \leq a_j$ then we must have $c_i \leq c_j$. Thus $F(P)$ is minimised when P sorts $\boldsymbol{b}$ to the same ordering as $\boldsymbol{a}$. Letting $\boldsymbol{a} = Q\boldsymbol{x}$ and $\boldsymbol{b} = \boldsymbol{x}$ in theorem 2 gives the result. □

We also have the following proposition that allows us for any distribution of $\boldsymbol{x}$ in $\mathbb{R}^p$, to convert this to a distribution on the unit hypersphere. This is useful in that it makes sampling $\boldsymbol{x}$ easier and it provides more geometric insight into exactly what is happening in our process.

Proposition 19. *The solution of (3.15) is invariant to the norm of $\boldsymbol{x}$, i.e., if $P(Q, \boldsymbol{x})$ is the solution and we write $\boldsymbol{x} = (r, \boldsymbol{\theta})$ in polar coordinates then we can equally write $P(Q, \boldsymbol{\theta})$ as the solution.*

Proof. Consider $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^p$ such that $\boldsymbol{x}_1 = (r_1, \boldsymbol{\theta})$ and $\boldsymbol{x}_2 = (r_2, \boldsymbol{\theta})$. Then as both the sorting of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is unchanged by multiplication by some constant $k > 0$ and also $Q(k\boldsymbol{x}) = kQ\boldsymbol{x}$ is also unchanged with respect to sorting, if P solves (3.15) for a given Q and $\boldsymbol{x}_1$, it also solves it for $(r_2/r_1)\boldsymbol{x}_1 = \boldsymbol{x}_2$, a rescaled version of $\boldsymbol{x}_1$. □

Permutation Distribution

We can now formally define the permutation distribution elicit by $Q \in \mathbb{R}^{p\times p}$ and cumulative distribution function μ for random variable $\boldsymbol{X}$ that our $\boldsymbol{x}$ are drawn from.

Definition 42. *We define the set of points in $\mathbb{R}^p$ that round Q to $P^* \in \mathbb{P}$ as*

$$S_Q(P^*) = \{\boldsymbol{x}_0 \in \mathbb{R}^p | P^* = \arg\min_{P\in\mathbb{P}} ||Q\boldsymbol{x}_0 - P\boldsymbol{x}_0||_F^2\}.$$

Definition 43. *The random variable representing the distribution over the permutation matrices elicit by Q and μ is written $P_{Q,\mu}$ and satisfies*

$$\Pr(P_{Q,\mu} = P) \stackrel{\text{def}}{=} \Pr(\boldsymbol{x} \in S_Q(P)).$$

Remark 40.
1. For different permutation matrices P_1 and P_2, the intersection $S_Q(P_1) \cap S_Q(P_2)$ is a set of measure zero. We prove this later.
2. Unless it is unclear the Q matrix being referred to, we drop the Q subscript from the notation $S_Q(\cdot)$ in future.

Link to Direct Rounding

Interestingly, for appropriately chosen values fo the distribution μ, we can show that minimising the expected Frobenius norm $\mathbb{E}_X[||Q\boldsymbol{X} - P\boldsymbol{X}||_F^2]$ is equivalent to projecting Q directly onto $\mathbb{P}$ using (3.8) as shown in [86].

Proposition 20. *For $\boldsymbol{X}$ uniformly distributed on the unit hypersphere*

$$S^{p-1} \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^p : ||\boldsymbol{x}|| = 1\},$$

we have that

$$\arg\min_{P\in\mathbb{P}} ||Q - P||_{\text{F}}^2 = \arg\min_{P\in\mathbb{P}} E(||Q\boldsymbol{X} - P\boldsymbol{X}||_{\text{F}}^2). \tag{3.20}$$

Proof. We show this in 3 steps:

1. Firstly,

$$\min_{P\in\mathbb{P}} E(||Q\boldsymbol{X} - P\boldsymbol{X}||_{\mathrm{F}}^2) = \max_{P\in\mathbb{P}} \int_{S^{p-1}} \boldsymbol{x}^T Q^T P \boldsymbol{x}\, \mathrm{d}\boldsymbol{x}.$$

2. Considering the quantity $\int_{S^{p-1}} \boldsymbol{x}^T A \boldsymbol{x}\, \mathrm{d}\boldsymbol{x}$ for some $A \in \mathbb{R}^{p\times p}$,
   (a) all off-diagonal terms, i.e., those of the form $A_{ij}x_i x_j$ for $i \neq j$, integrate to 0,
   (b) all diagonal elements $A_{ii}x_i^2$ integrate to $A_{ii}\beta$ for some constant β.
3. Hence $\max_{P\in\mathbb{P}} \int_{S^{p-1}} \boldsymbol{x}^T Q^T P \boldsymbol{x}\, \mathrm{d}\boldsymbol{x}$ is equivalent to maximising $\mathrm{tr}(Q^T P)$ which is equivalent to $\min_{P\in\mathbb{P}} ||Q - P||_{\mathrm{F}}$, so the result in (3.20) follows.

Step 1

$$\begin{aligned} ||Q\boldsymbol{x} - P\boldsymbol{x}||_{\mathrm{F}}^2 &= \mathrm{tr}\{(Q\boldsymbol{x} - P\boldsymbol{x})^T(Q\boldsymbol{x} - P\boldsymbol{x})\} \\ &= \mathrm{tr}\{(Q\boldsymbol{x})^T(Q\boldsymbol{x})\} + \mathrm{tr}\{(P\boldsymbol{x})^T(P\boldsymbol{x})\} \\ &- 2\boldsymbol{x}^T Q^T P\boldsymbol{x} \\ &= \mathrm{tr}\{(Q\boldsymbol{x})^T(Q\boldsymbol{x})\} + \boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{x}^T Q^T P\boldsymbol{x}, \end{aligned}$$

using the fact that P is an orthogonal matrix.

Therefore,

$$\begin{aligned} \min_{P\in\mathbb{P}} E(||Q\boldsymbol{X} - P\boldsymbol{X}||_{\mathrm{F}}^2) &= \max_{P\in\mathbb{P}} \int_{S^{p-1}} \boldsymbol{x}^T Q^T P\boldsymbol{x}\, \mathrm{d}\mu \\ &= \max_{P\in\mathbb{P}} \int_{S^{p-1}} \boldsymbol{x}^T Q^T P\boldsymbol{x}\, \mathrm{d}\boldsymbol{x}, \end{aligned}$$

where the final equality is a result of μ being a uniform distribution.

Step 2 Now consider $I \stackrel{\mathrm{def}}{=} \int_{S^{p-1}} \boldsymbol{x}^T A\boldsymbol{x}\mathrm{d}\boldsymbol{x}$. Writing $\boldsymbol{x}$ in terms of hyperspherical coordinates, we have on the unit hypersphere that the volume element is

$$\sin^{p-2}(\theta_1)\sin^{p-3}(\theta_2)\ldots\sin(\theta_{p-2})\mathrm{d}\theta_1\ldots\mathrm{d}\theta_{p-1},$$

and

$$\begin{aligned} x_1 &= \cos(\theta_1) \\ x_2 &= \sin(\theta_1)\cos(\theta_2) \\ &\vdots \\ x_{p-1} &= \sin(\theta_1)\ldots\sin(\theta_{p-2})\cos(\theta_{p-1}) \\ x_p &= \sin(\theta_1)\ldots\sin(\theta_{p-2})\sin(\theta_{p-1}). \end{aligned}$$

Consider off-diagonal elements of I of the form $A_{ij}x_ix_j$. For $i \neq j$, we see that x_ix_j contains at least one term of the form $\sin^L(\theta_k)\cos(\theta_k)$ for $L \geq 0$, i.e., when $k = i$ or $k = j$ as we cannot have both $i = p$ and $j = p$ as they cannot be equal. Hence

$$\int_{\mathbb{X}} x_ix_j \, \mathrm{d}\boldsymbol{x} = 2\int_{\mathcal{I}}\int_0^{\pi} \sin^L(\theta_k)\cos(\theta_k)\mathrm{d}\theta_k f(\boldsymbol{\theta}_{/k})\mathrm{d}\boldsymbol{\theta}_{/k},$$

where f is some function, $\boldsymbol{\theta}_{/k}$ is a vector of all θ_l without θ_k and $\mathcal{I}$ is the region over which we are integrating $\boldsymbol{\theta}_{/k}$. But,

$$\int_0^{\pi} \sin^L(\theta_k)\cos(\theta_k)\mathrm{d}\theta_k = \left[\frac{\sin^{L+1}(\theta_k)}{L+1}\right]_0^{\pi} = 0,$$

so all off-diagonal elements of I integrate to 0.

Now consider diagonal elements of I, of the form $A_{ii}x_i^2$. We now require two identities. Firstly,

$$\int_0^{\pi} \sin^p(\theta)\cos^2(\theta)\mathrm{d}\theta = \int_0^{\pi} \frac{\sin^{p+2}(\theta)}{p+1}\mathrm{d}\theta, \tag{3.21}$$

found from integrating by parts with $\mathrm{d}v = \sin^p(\theta)\cos(\theta)$ and $u = \cos(\theta)$. Secondly,

$$\int_0^{\pi} \sin^p(\theta)\mathrm{d}\theta = \frac{p-1}{p}\int_0^{\pi} \sin^{p-2}(\theta)\mathrm{d}\theta \tag{3.22}$$

integrating by parts with $\mathrm{d}v = \sin(\theta)$ and $u = \sin^{p-1}(\theta)$.

For $i \neq p$, we see that $\int_{S^{p-1}} x_i^2 \mathrm{d}\boldsymbol{x}$ can be written

$$\int_{S^{p-1}} x_i^2 \mathrm{d}\boldsymbol{x} = 2 \int_0^\pi [\sin^2(\theta_1) \ldots \sin^2(\theta_{i-1}) \cos^2(\theta_i)] \times \sin^{p-2}(\theta_1) \sin^{p-3}(\theta_2) \ldots \sin(\theta_{p-2}) \mathrm{d}\boldsymbol{\theta}, \tag{3.23}$$

where $\int_0^\pi$ represents the fact all θ_l are to be integrated between these bounds.

Now consider θ-index j and define $\int_0^\pi \sin^k(\theta) \mathrm{d}\theta \stackrel{\text{def}}{=} I_k$.

Case 1: $j < i$ The relevant integral in (3.23) is

$$\int_0^\pi \sin^{p-j-1}(\theta_j) \sin^2(\theta_j) \, \mathrm{d}\theta_j = I_{p-j+1} = \frac{p-j}{p-j+1} I_{p-j-1},$$

using (3.22).

Case 2: $j > i$ The relevant integral in (3.23) is

$$\int_0^\pi \sin^{p-j-1}(\theta_j) \mathrm{d}\theta_j = I_{p-j-1}.$$

Case 3: $j = i$

The relevant integral in (3.23) is

$$\begin{aligned} \int_0^\pi \cos^2(\theta_j) \sin^{p-j-1}(\theta_j) \, \mathrm{d}\theta_j &= \frac{1}{p-j} I_{p-j+1} \\ &= \frac{1}{p-j+1} I_{p-j-1}. \end{aligned}$$

using both (3.21) and (3.22).

Putting together all these cases we see that

$$\begin{aligned}\int_{S^{p-1}} x_i^2 \,\mathrm{d}\boldsymbol{x} &= 2\left(\prod_{j=1}^{i-1}\frac{p-j}{p-j+1}I_{p-j-1}\right)\left(\prod_{j=i+1}^{p-1} I_{p-j-1}\right)\\ &\times \quad \left(\frac{1}{p-i+1}I_{p-i-1}\right)\\ &= \quad 2\left(\frac{p-1}{p}\frac{p-2}{p-1}\cdots\frac{p-i+1}{p-i+2}\frac{1}{p-i+1}\right)\\ &\times \quad \prod_{j=1}^{p-1} I_{p-j-1}\\ &= \quad \frac{2}{p}\alpha,\end{aligned}$$

where $\alpha = \prod_{j=1}^{p-1} I_{p-j-1} = \prod_{j=0}^{p-2} I_j$.

Finally we look at $i = p$, for which $\int_{S^{p-1}} x_p^2 \,\mathrm{d}\boldsymbol{x}$ is

$$\begin{aligned}&2\int_0^{\pi} \sin^p(\theta_1)\sin^{p-1}(\theta_2)\ldots\sin^3(\theta_{p-2})\sin^2(\theta_{p-1})\,\mathrm{d}\boldsymbol{\theta}\\ &\qquad =2\prod_{j=2}^{p} I_j = 2\prod_{j=2}^{p}\frac{j-1}{j}I_{j-2} = \frac{2}{p}\prod_{j=0}^{p-2} I_j = \frac{2}{p}\alpha.\end{aligned}$$

Hence,

$$\int_{S^{p-1}} \boldsymbol{x}^T A\boldsymbol{x}\,\mathrm{d}\boldsymbol{x} = \int_{S^{p-1}}\sum_{i=1}^{p} A_{ii}x_i^2\,\mathrm{d}\boldsymbol{x} = \frac{2}{p}\alpha\,\mathrm{tr}\{A\}.$$

Step 3 Now we see that,

$$\begin{aligned}\arg\min_{P\in\mathbb{P}} E(||Q\boldsymbol{X}-P\boldsymbol{X}||_{\mathrm{F}}^2) &= \arg\max_{P\in\mathbb{P}}\int_{\mathbb{X}} \boldsymbol{x}^T Q^T P\boldsymbol{x}\,\mathrm{d}\boldsymbol{x}\\ &= \arg\max_{P\in\mathbb{P}}\mathrm{tr}\{Q^T P\}\\ &= \arg\min_{P\in\mathbb{P}}||Q-P||_{\mathrm{F}}^2,\end{aligned}$$

when $\boldsymbol{X}$ is uniformly distributed on the unit hypersphere. □

Proposition 21. *For $\boldsymbol{x}\in\mathbb{R}^p$ with $\boldsymbol{X}$ uniformly distributed in the unit hy-*

percube $H = [0,1]^p$ *for which* $0 \leq x_i \leq 1$,

$$\arg\min_{P\in\mathbb{P}} ||Q-P||_{\mathrm{F}}^2 = \arg\min_{P\in\mathbb{P}} E(||Q\boldsymbol{X} - P\boldsymbol{X}||_{\mathrm{F}}^2). \tag{3.24}$$

Proof. In this case we have for $A = Q^T P$,

$$\begin{aligned}\int_H \boldsymbol{x}^T A\boldsymbol{x}\,\mathrm{d}\boldsymbol{x} &= \int_H \left(\sum_{i=1}^p A_{ii}x_i^2 + \sum_{i\neq j} A_{ij}x_i x_j\right)\,\mathrm{d}\boldsymbol{x} \\ &= \left[\frac{1}{3}\sum_{i=1}^p A_{ii}x_i^3\frac{V}{x_i} + \frac{1}{4}\sum_{i\neq j} A_{ij}x_i^2x_j^2\frac{V}{x_ix_j}\right]_H\end{aligned}$$

where $V = \prod_{i=1}^p x_i$. Plugging in the limits for H the integral is

$$\frac{1}{3}\operatorname{tr}\{A\} + \frac{1}{4}\mathbf{1}^T A\mathbf{1} - \frac{1}{4}\operatorname{tr}\{A\} = \frac{1}{12}\operatorname{tr}\{A\} + \frac{1}{4}\mathbf{1}^T A\mathbf{1}.$$

Noting that $\mathbf{1}^T Q^T P\mathbf{1} = \mathbf{1}^T Q^T\mathbf{1}$ is invariant for all permutation matrices as they simply permute the columns of Q^T, we see that

$$\arg\min_{P\in\mathbb{P}} E(||Q\boldsymbol{X} - P\boldsymbol{X}||_{\mathrm{F}}^2) = \arg\max_{P\in\mathbb{P}} \operatorname{tr}\{Q^T P\}$$

and the result follows. □

Partitioning of $\mathbb{R}^p$

We can now delve deeper into exactly what the equation (3.15) does by considering the transformation into $\mathbb{R}^p$. We investigate the $S(P)$ and what it means if two sets $S(P_1)$ and $S(P_2)$ are close to one another in $\mathbb{P}^p$. Looking at the 3D case, we can represent our rounding sets as partitions of a unit sphere and look at a way to convert back from a permutation matrix to a point in $\mathbb{R}^p$. Finally we see this does not work for certain classes of matrices, doubly stochastic being one of them and introduce an adjustment to fix this issue.

Continuity

We first show as in [86], that for any point in $S(P)$ not on its boundary, we can find other points close to it that are also in $S(P)$.

Proposition 22. *For $\boldsymbol{x} \in S(P)$ such that $x_i \neq x_j$ and $(Q\boldsymbol{x})_i \neq (Q\boldsymbol{x})_j$ for $i \neq j$, we can find $\epsilon > 0$ such that $\mathbb{B}_\epsilon(\boldsymbol{x}) \subset S(P)$, whenever $\mathbb{B}_\epsilon(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^n : ||\boldsymbol{y} - \boldsymbol{x}||_{\mathrm{F}}^2 < \epsilon\}$.*

Proof. Using theorem 2, we know that for the permutation to be the same for $\boldsymbol{x}$ and $\boldsymbol{y}$ it is sufficient, (from (3.15)), that

$$\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{r}(\boldsymbol{y}) \text{ and } \boldsymbol{r}(Q\boldsymbol{x}) = \boldsymbol{r}(Q\boldsymbol{y}).$$

We have $y \in \mathbb{B}_\epsilon(\boldsymbol{x})$ such that by definition $||\boldsymbol{y}-\boldsymbol{x}||_{\mathrm{F}}^2 < \epsilon$. Also, $||Q(\boldsymbol{y}-\boldsymbol{x})||_{\mathrm{F}}^2 < \delta\epsilon$, where $\delta = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } Q^TQ\}$, [36, p. 296].

By definition no $(Q\boldsymbol{x})_i = (Q\boldsymbol{x})_j$, so for the sorting order to remain the same for $Q\boldsymbol{y}$ and $Q\boldsymbol{x}$ we require that, if $(Q\boldsymbol{x})_i < (Q\boldsymbol{x})_j$, then $(Q\boldsymbol{y})_i < (Q\boldsymbol{y})_j$.

Also note that if, $x_i < x_j$ then for the sorting to remain the same, we require $y_i < y_j$.

Now, $|y_i - x_i| < \epsilon^{1/2}$ so $y_i \in (x_i - \epsilon^{1/2}, x_i + \epsilon^{1/2})$, so

$$y_i < x_i + \epsilon^{1/2}$$

and similarly,

$$y_j > x_j - \epsilon^{1/2}.$$

Then taking for example

$$\epsilon^{1/2} = \epsilon_1 < \frac{1}{2} \min_{u,v} |x_u - x_v|$$

ensures that

$$y_i < x_i + \epsilon_1 < x_j - \epsilon_1 < y_j.$$

Similarly

$$(Q\boldsymbol{y})_i < (Q\boldsymbol{x})_i + (\delta\epsilon)^{1/2}$$

and

$$(Q\boldsymbol{y})_j > (Q\boldsymbol{x})_j - (\delta\epsilon)^{1/2}.$$

Then taking

$$\epsilon_2 = \epsilon^{1/2} < \frac{1}{2\delta^{1/2}} \min_{u,v} |(Q\boldsymbol{x})_u - (Q\boldsymbol{x})_v|$$

ensures that

$$\begin{aligned}(Q\boldsymbol{y})_i &< (Q\boldsymbol{x})_i + \delta^{1/2}\epsilon_2 < (Q\boldsymbol{x})_j - \delta^{1/2}\epsilon_2 \\ &< (Q\boldsymbol{y})_j.\end{aligned}$$

Hence we can choose $\epsilon = \min(\epsilon_1^2, \epsilon_2^2)$, completing the proof. □

Remark 41. Points $\boldsymbol{x} \in \mathbb{R}^p$ are on the boundary of $S(P)$ if firstly it is in $S(P)$ and $x_i = x_j$ or $(Q\boldsymbol{x})_i = (Q\boldsymbol{x})_j$ for some $i \neq j$. In this case there are multiple solutions to (3.15) so $\boldsymbol{x}$ belongs to multiple rounding sets. In the case $\boldsymbol{x} = \boldsymbol{0}$, $\boldsymbol{x} \in S(P)$ for all permutation matrices P. Of course we normally only consider $\boldsymbol{x}$ on the unit hypersphere.

It is now possible to show as mentioned before that when sampling points on the unit hypersphere based on a Gaussian distribution, we select a point on the boundary of two rounding sets with probability 0 if Q is a doubly stochastic or orthogonal matrix. The result comes from [86].

Proposition 23. *If we are sampling from a purely continuous distribution with $\boldsymbol{x} \in \mathbb{R}^p$ defined by random variable $\boldsymbol{X}$, then* $\Pr((\boldsymbol{X} = \boldsymbol{x}) \cap (x_i = x_j : i \neq j)) = 0$ *and* $\Pr((\boldsymbol{X} = \boldsymbol{x}) \cap ((Q\boldsymbol{x})_i = (Q\boldsymbol{x})_j : i \neq j, Q \in \mathbb{D} \cup \mathbb{Q})) = 0$.

Proof. In both cases the sets are of measure zero in $\mathbb{R}^p$ and hence correspond to zero probability. □

Distance in $\mathbb{R}^p$ vs $\mathbb{P}$

When moving small distances in $\mathbb{R}^p$ i.e. between $\boldsymbol{x}^0$ and $\boldsymbol{x}^1$ such that $\boldsymbol{x}^0 \in S(P_0)$ and $\boldsymbol{x}^1 \in S(P_1)$ then normally the Frobenius norm between P_0 and P_1 is small (although this is not always the case). In the most common scenario

there exist some i and j such that $x_i^0 < x_j^0$ but $x_i^1 < x_j^1$ with no other entries flipping. This causes a similar flipping in the entries of P_0 and P_1.

Using the example in [86], if P_0 sends in particular $1 \to 2$ and $3 \to 4$, P_1 may now send $1 \to 4$ and $3 \to 2$ while agreeing with P_0 on all other element permutations. This leads to $||P_0 - P_1||_F^2 = 4$ which is the minimum possible squared Frobenius norm between two different permutation matrices. Note however, the distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ can be small with more indices being switched than simply i and j, resulting in a larger difference between P_0 and P_1.

Put another way, there may exist P_0 and P_1 such that the boundary between them corresponds to some hyperplane $x_{i_1} = x_{i_2} = \cdots = x_{i_k}$ for some large integer k. The distance between two points in $\mathbb{R}^p$ either side of this hyperplane may be small, however P_0 and P_1 would have a large difference with respect to Frobenius norm.

Remark 42. In the above arguments we just concentrated on flips occurring due to crossing hyperplanes of the form $x_{i_1} = x_{i_2} = \cdots = x_{i_k}$. Note that the same arguments can be applied when crossing hyperplanes $(Q\boldsymbol{x})_{i_1} = (Q\boldsymbol{x})_{i_2} = \cdots = (Q\boldsymbol{x})_{i_k}$.

This shows there is a degree of continuity between points in $\mathbb{R}^p$ and the permutation matrices they map to given some $Q \in \mathbb{R}^{p \times p}$. We look to leverage this fact in the next section by using an algorithm that searches in $\mathbb{R}^p$ as opposed to $\mathbb{P}$ (like the 2-opt strategy).

3D Visualisation

As we mentioned previously, the boundaries between rounding sets are hyperplanes such that $x_i = x_j$ or $(Q\boldsymbol{x})_i = (Q\boldsymbol{x})_j$. In the 3D case we can visualise the partitioning of the unit sphere into these rounding sets. The top image of figure 3.3 from [86] shows for random symmetric matrices $A, B \in \mathbb{R}^{3\times3}$, the partitions of the unit sphere induced by doubly stochastic matrix $Q \in \mathbb{R}^{3\times3}$ solving (3.11). The bottom image shows the partitioning induced by a random orthogonal matrix Q with an eigenvector of $\mathbf{1}$. In both cases, the thin

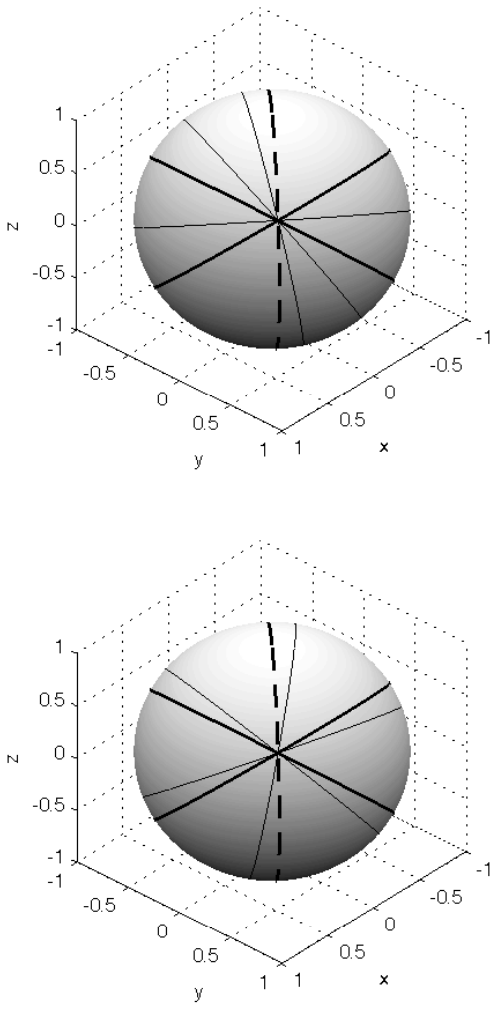

Figure 3.3: 3D sphere showing the partition boundaries for a doubly stochastic matrix Q (top figure), and for an orthogonal matrix (bottom figure). Here $x = x_1, y = x_2, z = x_3$. See text for further details.

lines correspond to the boundaries caused by $(Q\boldsymbol{x})_i = (Q\boldsymbol{x})_j$ and the thick lines by $x_i = x_j$. The dotted line represents $x_1 = x_2$ and is used simply to aid in orientation.

Remark 43. 1. In both figures, the point $x_1 = x_2 = x_3$ is in the same position as $(Q\boldsymbol{x})_1 = (Q\boldsymbol{x})_2 = (Q\boldsymbol{x})_3$ and can be represented by $\boldsymbol{a} = 3^{-1/2}\mathbf{1}$. This reason this is, is because both in the doubly stochastic

and orthogonal case, Q has an eigenvector $\mathbf{1}$ (this is a property of all doubly stochastic matrices and we specified it in the orthogonal case). Hence in both cases $Q\mathbf{1} = c\mathbf{1}$ for some $c \in \mathbb{R}$ and as $\boldsymbol{a}$ is on the unit sphere, the boundaries of all permutation sets intersect at it. In effect, it is a significant point of discontinuity as we spoke about previously. We will see how this causes some issues when using doubly stochastic matrices in future.

2. We can calculate the distribution induced by Q over the permutation matrices by calculating the angles between each plane $(Q\boldsymbol{x})_i = (Q\boldsymbol{x})_j$ and $x_i = x_j$ at the point $\boldsymbol{a}$. Given a boundary angle θ for a rounding set, the probability of sampling a point in it is $\frac{\theta}{2\pi}$. Doing this for all rounding sets (and calculating the permutation the set rounds to) gives us our distribution over $\mathbb{P}$ induced by Q where μ is uniform on the unit sphere. Note this only works in the case $Q\mathbf{1} = c\mathbf{1}$, otherwise the boundary of all rounding sets does not meet at a single point $\boldsymbol{a}$. Note that we can find θ by considering normal vectors to the planes $(Q\boldsymbol{x})_i = (Q\boldsymbol{x})_j$ and $x_i = x_j$ in $\mathbb{R}^3$.

3. In the orthogonal case, the thin lines have a constant angle between them and orthogonal matrix Q can be thought of as rotating the thick lines to the thin lines abut the line $\overrightarrow{\mathbf{1}}$. In the doubly stochastic case however the thin lines don't have a constant able between them and it highlights the fact doubly stochastic matrices can induce more 'varied' permutation distributions than orthogonal matrices.

Mapping Permutations to Points

Given a matrix $Q \in \mathbb{R}^{p\times p}$, then for any point $\boldsymbol{x} \in \mathbb{R}^p$ we can map it to a permutation matrix P using (3.15). It is now interesting and of use to see if given a permutation matrix, can we find a point in $\mathbb{R}^p$ that maps to it i.e. a point in one of the rounding sets of P. This is in fact possible given certain properties of Q and is described in the following theorem and remarks from [86].

Theorem 3. *Consider $Q \in \mathbb{R}^{p\times p}$ and let $\boldsymbol{a} = p^{-1/2}\mathbf{1}$. If $\boldsymbol{b} = Q^{-1}\boldsymbol{a}$ is such that $b_i = b_j \Rightarrow i = j$, then for any $P^* \in \mathbb{P}$, we can find $\boldsymbol{x} \in \mathbb{R}^p$ such that*

$$P^* = \arg\min_{P\in\mathbb{P}} ||Q\boldsymbol{x} - P\boldsymbol{x}||_{\mathrm{F}}^2$$

and it is given by $\boldsymbol{x} = \boldsymbol{b} + Q^{-1}P^*P_b^T\boldsymbol{\epsilon}$, where P_b orders $\boldsymbol{b}$ in ascending order (i.e. $\boldsymbol{r}((P_b\boldsymbol{b}))_i = i$) and $\boldsymbol{\epsilon} = \delta[1, 2, \ldots, p]^T$ for some $\delta > 0$.

Proof. Let $\boldsymbol{x} = \boldsymbol{b}+A\boldsymbol{\epsilon}$ for some $A \in \mathbb{R}^{p\times p}$. Now, $Q\boldsymbol{x} = Q(\boldsymbol{b}+A\boldsymbol{\epsilon}) = \boldsymbol{a}+QA\boldsymbol{\epsilon}$.

Step 1. What is $\boldsymbol{r}(\boldsymbol{x})$? Since $b_i = b_j \Rightarrow i = j$, we can choose $\delta > 0$ sufficiently small such that

$$\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{r}(\boldsymbol{b}), \tag{3.25}$$

i.e., the ordering of $\boldsymbol{x}$ is the same as $\boldsymbol{b}$. Note that $\boldsymbol{r}(\boldsymbol{\epsilon}) = \boldsymbol{r}(P_b\boldsymbol{b})$ as both are in ascending order.

Step 2. What is $\boldsymbol{r}(Q\boldsymbol{x})$? Now $\boldsymbol{r}(Q\boldsymbol{x}) = \boldsymbol{r}(\boldsymbol{a} + QA\boldsymbol{\epsilon}) = \boldsymbol{r}(QA\boldsymbol{\epsilon})$ as $\boldsymbol{a}$ is a constant vector. We can therefore choose $A = Q^{-1}P^*P_b^T$ to get

$$\boldsymbol{r}(Q\boldsymbol{x}) = \boldsymbol{r}(P^*P_b^T\boldsymbol{\epsilon}) = \boldsymbol{r}(P^*\boldsymbol{b}).$$

Then we use (3.17) which says that $\boldsymbol{r}(P^*\boldsymbol{b}) = P^*\boldsymbol{r}(\boldsymbol{b})$. So

$$\boldsymbol{r}(Q\boldsymbol{x}) = P^*\boldsymbol{r}(\boldsymbol{b}) = P^*\boldsymbol{r}(\boldsymbol{x}),$$

where the last step uses (3.25).

Step 3. Therefore by theorem 2, for $\boldsymbol{x} = \boldsymbol{b}+Q^{-1}P^*P_b^T\boldsymbol{\epsilon}$, $P^* = \arg\min_{P\in\mathbb{P}} ||Q\boldsymbol{x}-P\boldsymbol{x}||_F^2$, and the proof is complete. □

Remark 44.

1. As the above theorem is constructive, we can use it later in our algorithm. In particular, it can be used for finding initial points i.e. if P_0 is the direct projection of Q onto $\mathbb{P}$, we can use theorem 3 to find $\boldsymbol{x}$ in the rounding set of P_0 and use this value as a starting point for the search algorithm.

2. The value $\delta > 0$ in practice, is up to us to choose. We generally want to choose the largest value possible such that the result of the theorem

still holds, when finding starting points. This is to move $\boldsymbol{x}$ as far from $\boldsymbol{b}$ as possible, which is a significant point of discontinuity. Being close to it neutralises the benefits of a well chosen starting point.

3. The theorem does not hold for a doubly stochastic matrix Q, as

$$Q\boldsymbol{a} = \boldsymbol{a} \Rightarrow \boldsymbol{a} = \boldsymbol{b}.$$

Therefore

$$b_i = b_j \not\Rightarrow i = j$$

as all elements of $\boldsymbol{a}$ and hence $\boldsymbol{b}$ are equal.

4. If $\boldsymbol{r}(\boldsymbol{b})$ has no distinct ordering i.e. there exists $i \neq j$ such that $b_i = b_j$, then at least two permutations sort $\boldsymbol{b}$ in the same way. Hence $\boldsymbol{b}$ is on the boundary between permutation sets.

Doubly Stochastic Matrices

If Q is doubly stochastic than as seen above, we cannot apply theorem 3 to get a starting point for our algorithm. What we can instead do is slightly perturb Q to $Q^{'}$ i.e.

$$Q^{'} = Q + \lambda U,$$

where $\lambda \in \mathbb{R}$ is some small constant and $U \in \mathbb{R}^{p \times p}$ is a uniform random matrix i.e. all entries satisfy $U_{ij} \sim Unif[0, 1]$. With probability 1, $Q^{'}$ satisfies the conditions in theorem 3 and hence for any permutation we can find a point mapping to it when using $Q^{'}$ as the partitioning matrix.

Figure 3.4 from [86] shows an example partitioning in 3D when using $Q^{'}$ as opposed to Q. We see the boundaries of all permutation sets no longer intersect at $\boldsymbol{a}$.

Sampling Strategy with Variance Adaption

We now propose an algorithm for rounding a solution to the relaxed graph matching problem to a permutation matrix as presented in [86]. It bears

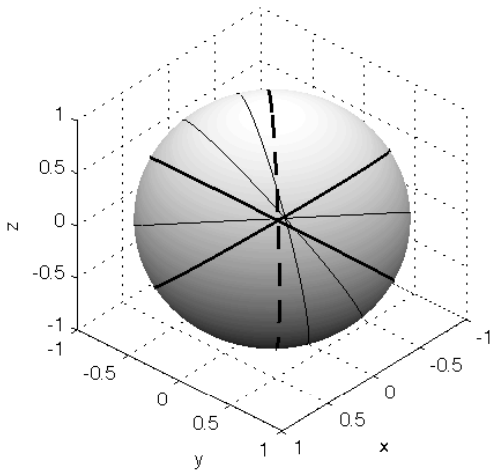

Figure 3.4: 3D sphere showing the partition boundaries for a perturbed doubly stochastic matrix ($\lambda = 0.05$).

many similarities to simulated annealing where broadly, we update our current optimal point if a proposal point results in a better objective function value and as we progress our proposal points are chosen closer and closer to the current optimal point. As with simulated annealing, we can also randomly accept worse proposal points to aid in exploration of the algorithm.

Algorithm 10 Sampling strategy to round a matrix to a permutation matrix given an objective function to minimise

Require: `total_iterations`, $\Lambda_{\max}$ estimating samples M, pre-samples L, variance update T, target function f_t, acceptance probability function $\mathcal{A}(E, E_*, t)$, matrix to round to permutations $Q \in \mathbb{R}^{p\times p}$, objective function to minimise $g(\cdot)$, initial point $\boldsymbol{x}_0 \in \mathbb{R}^p$

$P_0 \leftarrow \arg\min_{P\in\mathbb{P}} ||Q\boldsymbol{x}_0 - P\boldsymbol{x}_0||_F^2$
$E_0 \leftarrow g(P_0)$
Sample $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$ on the unit hypersphere and calculate $\Delta_{\max}$ using (3.4).
Learn $\tilde{\Delta}(\sigma^2)$ and hence an estimate for $\bar{\Delta}(\sigma^2)/\Delta_{\max}$ by generating pre-samples $y_{-L}, \ldots, y_{-1}$ and corresponding $\Delta_{-L}, \ldots, \Delta_{-1}$ as in algorithm 6.
$t \leftarrow 0$
while $t <$ `total_iterations` **do**
    if $t \mod T = 0$ **then**
        Re-learn $\tilde{\Delta}(\sigma^2)$ using $y_{-L}, \ldots, y_{t-1}$ and $\Delta_{-L}, \ldots, \Delta_{t-1}$.
    end if
    $\sigma_t^2 \leftarrow \arg\min_{\tilde{\sigma}^2} |\tilde{\Delta}(\tilde{\sigma}^2)\Delta_{\max} - f_t|$
    Sample proposal $\boldsymbol{x}_*$ from $N(\boldsymbol{x}_{t-1}, \sigma_t^2$
    $\boldsymbol{x}_* \leftarrow \boldsymbol{x}_*/||\boldsymbol{x}_*||$
    $P_* \leftarrow \arg\min_{P\in\mathbb{P}} ||Q\boldsymbol{x} - P\boldsymbol{x}_0||_F^2$
    $E_* \leftarrow g(P_*)$
    Sample u from $Unif[0, 1]$
    if $u \leq \mathcal{A}(E_{t-1}, E_*, t)$ **then**
        $(\boldsymbol{x}_t, P_t, E_t) \leftarrow (\boldsymbol{x}_*, P_*, E_*)$
    else
        $(\boldsymbol{x}_t, P_t, E_t) \leftarrow (\boldsymbol{x}_{t-1}, P_{t-1}, E_{t-1})$
    end if
    $t \leftarrow t + 1$
end while
Return $(\boldsymbol{x}_t, P_t, E_t)$

Remark 45. 1. In the graph matching case, we want to input into the algorithm $Q \in \mathbb{R}^{p\times p}$ solving some relaxed version of the GMP e.g. the result of QCV or FAQ before projection.

2. Once again, in the graph matching case we use the objective function

$$g(P) = ||A - P^T BP||_F^2$$

for some matrices $A, B \in \mathbb{R}^{p \times p}$.

3. The initial point $\boldsymbol{x}_0$ can be randomly chosen on the unit hypersphere. A better method however is to find $P_0 = \arg\min_{P \in \mathbb{P}} ||Q - P||_F^2$ using the Hungarian algorithm. We can then reverse this to an initial point $\boldsymbol{x}_0$ using theorem 3 and perturbing the matrix Q if necessary (as discussed above for doubly stochastic matrices).

4. A common choice of $\mathcal{A}$ is the indicator function (i.e. no randomness) such that

$$\mathcal{A}(E, E_*, t) = I(E_* \leq E).$$

Notice we use a $\leq$ sign to allow our algorithm to move around within permutation sets and not get stuck in the middle of a large one.

5. The choice of f_t is important to the algorithm. Of course it must decay towards 0, but too sharply and we don't explore the permutation sets well enough, too slowly and we don't spend enough time 'fine tuning' to improve the permutation estimate.

6. Due to the algorithm similarity with simulated annealing, we can parallelise using similar techniques. Work in [63] investigates a number of approaches covering both asynchronous approaches where no information is shared between threads and synchronous approaches, where the threads share information at given iterations to help each other out. An obvious extension to our algorithm would be to incorporate one of these techniques.

Results

We now include the results from [86] when testing the sampling strategy on the QAPLIB benchmarks. They use the QCV method to obtain a doubly

stochastic relaxed solution to the graph matching problem and project this onto the permutation matrices using the sampling strategy. They refer to this method as SSQCV. The parameters used were as follows

> In the results which follow we use λ to perturb Q such that $Q \leftarrow Q + \lambda U$ where U is a matrix of uniform random numbers between $[0, 1]$. We take $\texttt{totalIterations} = 100000; M = 100, L = 1000, \lambda = 0.1, T = \texttt{totalIterations}/10$. We use the pure strategy choice of $\mathcal{A}$ and set $f_t = \Delta_{\max}[1 - (t/\texttt{totalIterations})^{0.6}]$.

The table of results is given in 3.1 where

1. QAP: The name of the benchmark in QAPLIB.
2. Min: The true minimum trace value of the benchmark.
3. PATH: The minimum trace value found by the PATH algorithm.
4. SSQCV Mean: Over 20 runs of the algorithm, the mean minimum trace value found.
5. SSQCV Best: Over 20 runs of the algorithm, the best minimum trace value found.
6. SSQCV Time: Over 20 runs of the algorithm, the mean execution time taken.

While the sampling strategy algorithm was able to outperform the PATH algorithm on many of the QAPLIB problems as seen in table 3.1, when we tested the FAQ algorithm on QABLIB, it outperformed the SSQCV on most benchmarks.

Remark 46. When using our projection step (instead of the common projection (3.8)) with the FAQ algorithm, we were not able to achieve any improvement on the high dimension QAPLIB problems. It suggests that the FAQ solution is a local minimum with respect to our transformed Euclidean space in these cases. We note however that in [58], for correlated Bernoulli

QAP	Min	PATH	SSQCV Mean	SSQCV Best	SSQCV Time (s)
chr12c	11156	18048	**13088**	11414	15.98
chr15a	9896	19086	**14247**	11168	20.07
chr15c	9504	16206	**15199**	11200	19.07
chr20b	2298	5560	**3960**	3054	16.73
chr22b	6194	8500	**7574**	7196	17.50
exc16b	292	300	**292**	292	16.54
rou12	235528	256320	**246063**	240598	16.31
rou15	354210	391270	**380746**	365264	16.49
rou20	725522	**778284**	778709	760874	16.99
tai15a	388214	419224	**409769**	395714	16.94
tai17a	491812	530978	**525815**	514496	16.76
tai20a	703482	**753712**	766274	751414	17.03
tai30a	1818146	**1903872**	1979579	1946888	18.37
tai35a	2422002	**2555110**	2659594	2613758	22.40
tai40a	3139370	**3281830**	3459139	3407476	24.16

Table 3.1: Experimental results for QAPLIB benchmark data sets

random graphs with $\rho < 0.75$, the FAQ solution is empirically found to far from the anticipated global minimum of the GMP and it would make sense doing more comprehensive tests on whether our modified projection could outperform the standard projection for varying correlation and dimension.

Conclusion and Future Work

Graph matching is useful in our brain diagnosis application as it allows us to define some distance between two graphs extracted from the patients' observed time series. This can then be used as part of a larger clustering algorithm to aid in diagnosis of mental disease. As the GMP can be computationally intractable to find an exact solution in many real life applications a typical approach is to solve a relaxed form of the GMP to obtain an approximate solution. This gives a computation time versus accuracy trade off. In this chapter we present current methods used for finding approximate solutions to the GMP and we present our own projection method that projects

matrices solving relaxed versions of the GMP to permutation matrices.

While initial tests of the method were promising, they were unable to beat the FAQ algorithm on the QAPLIB benchmark. It was shown in [58] that for a specific class of problem, the non-convex relaxation is provably better than the convex relaxation. They also showed this held empirically for problems outside this specific class and hypothesised that the theoretical result would extend to a much broader class of problem. We tend to agree and as such use the FAQ algorithm in our multiple graph matching framework to be introduced in chapter 6.

There may however be cases as mentioned in [58], where the optimisation fails to find a good estimate and fine-tuning either the convex or non-convex optimum may provide better results.

4

Building Graphical Models From Time Series at the Source Level

Introduction

In chapter 2 we showed given a time series recorded on the scalp, how to extract a conditional dependence graph from it. In this chapter we make the assumption that the recordings on the scalp are actually a linear mixing of signals being emitted from sources within the brain. We aim to build conditional dependence graphs based on these source signals.

The idea of working with source signals is not new. In fact our methodology of extracting the sources closely follows [33]. They assume the sources follow a VAR process and unmix them by estimating this process and applying ICA to the residuals. It is then possible to build a conditional dependence graph with the extracted source VAR process. We note that ICA leaves us with sources only up to some unknown scaling and permutation. Thus we choose to build our graphs using PMIR which is unaffected by the unknown scaling

(the unknown permutation we deal with in later chapters).

The outline of the chapter is as follows, section 4.1 contains some preliminary results. Section 4.2 defines the model we assume for our data and section 4.3 shows how we can estimate the model from time series observations. In section 4.4 we look at how to build conditional independence graphs from the estimated model and give the algorithm. Section 4.5 shows some initial experiments using the model on simulated data. We end the chapter with some concluding remarks and ideas for future work.

Preliminaries

Principal Component Analysis

Uncorrelated Components

Let $\tilde{X} \in \mathbb{R}^{p \times n}$, then principal component analysis (PCA) is concerned with finding matrix $C \in \mathbb{R}^{p \times p}$ such that for

$$X = C\tilde{X}$$

then

$$\text{cov}(X)_{ij} = 0 \tag{4.1}$$

for all $i \neq j$.

From the basic rules of covariances,

$$\text{cov}(C\tilde{X}) = C\text{cov}(\tilde{X})C^{-1}.$$

Hence we can find a C such that (4.1) is satisfied by solving the eigendecomposition for $\text{cov}(\tilde{X})$. In this case C will be the matrix of eigenvectors of $\text{cov}(\tilde{X})$ and $\text{cov}(C\tilde{X})$ will be the diagonal matrix of eigenvalues.

Remark 47.

As $\text{cov}(\tilde{X})$ is a symmetric matrix, it is certainly diagonalisable and in this

case C will be an orthogonal matrix.

For high dimensional p, calculating the covariance directly can be costly. Instead the eigenvectors of the covariance can be found more efficiently by calculating the singular value decomposition of $\tilde{X}$.

Dimensionality Reduction

We can also use PCA for dimensionality reduction. Each row of X (a principal component) is uncorrelated and explains a certain amount of the variance associated with $\tilde{X}$. If a principal component explains very little of the overall variance, it may be worth reducing the dimension of the data by removing that row where the idea is that most of the signal in the data will be captured by the principal components explaining the most variance.

If we know a priori how many rows we want matrix X to have i.e. L, then we simply keep the L rows with the largest variance. Often this is not the case and we may instead keep as many rows as necessary, starting with the ones corresponding to the largest variance, to explain some ratio γ of the total variance.

VAR Model Fitting

Given some observations $\boldsymbol{x}_1, \dots \boldsymbol{x}_n \in \mathbb{R}^p$ from a VAR process, we show how we can estimate the parameter matrix of the VAR process by minimising the least squares error.

Consider the $\text{VAR}_p(L)$ model

$$\boldsymbol{X}_t = \sum_{\tau=1}^{L} A_\tau \boldsymbol{X}_{t-\tau} + \boldsymbol{\epsilon}_t$$

for some spatially independent, zero mean, white noise process $\{\boldsymbol{\epsilon}_t\}$. We now aim to estimate the parameters by solving

$$\arg \min_{A_1,\ldots,A_L} \sum_{\tau=L+1}^{n} ||\boldsymbol{x}_t - \sum_{\tau=1}^{L} A_\tau \boldsymbol{x}_{t-\tau}||^2$$

i.e. minimising the sum of all the squared estimated residuals (least squares). Note we only have enough observations to calculate the estimated residuals for $t = L+1, \ldots, n$.

This essentially gives the multivariate regression

$$Y = BZ + U,$$

where

$$\begin{aligned} Y &= [\boldsymbol{x}_{L+1}, \ldots, \boldsymbol{x}_n] \\ &= \begin{bmatrix} (\boldsymbol{x}_{L+1})_1 & (\boldsymbol{x}_{L+2})_1 & \ldots & (\boldsymbol{x}_n)_1 \\ \vdots & \vdots & & \vdots \\ (\boldsymbol{x}_{L+1})_p & (\boldsymbol{x}_{L+2})_p & \ldots & (\boldsymbol{x}_n)_p \end{bmatrix}, \end{aligned}$$

$$\begin{aligned} B &= [A_1, \ldots, A_L] \\ &= \begin{bmatrix} (A_1)_{11} & \ldots & (A_1)_{1p} & (A_2)_{11} & \ldots & (A_L)_{11} & \ldots & (A_L)_{1p} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ (A_1)_{p1} & \ldots & (A_1)_{pp} & (A_2)_{p1} & \ldots & (A_L)_{p1} & \ldots & (A_L)_{pp} \end{bmatrix}, \end{aligned}$$

$$Z = \begin{bmatrix} \boldsymbol{x}_L & \boldsymbol{x}_{L+1} & \dots & \boldsymbol{x}_{n-1} \\ \boldsymbol{x}_{L-1} & \boldsymbol{x}_L & \dots & \boldsymbol{x}_{n-2} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_{n-1-p} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{x}_L)_1 & (\boldsymbol{x}_{L+1})_1 & \dots & (\boldsymbol{x}_{n-1})_1 \\ \vdots & \vdots & & \vdots \\ (\boldsymbol{x}_L)_p & (\boldsymbol{x}_{L+1})_p & \dots & (\boldsymbol{x}_{n-1})_p \\ (\boldsymbol{x}_{L-1})_1 & (\boldsymbol{x}_L)_1 & \dots & (\boldsymbol{x}_{n-2})_1 \\ \vdots & \vdots & & \vdots \\ (\boldsymbol{x}_1)_1 & (\boldsymbol{x}_2)_1 & \dots & (\boldsymbol{x}_{n-1-p})_1 \\ \vdots & \vdots & & \vdots \\ (\boldsymbol{x}_1)_p & (\boldsymbol{x}_2)_p & \dots & (\boldsymbol{x}_{n-1-p})_p \end{bmatrix},$$

$$\begin{aligned} U &= [\boldsymbol{\epsilon}_{L+1}, \dots, \boldsymbol{\epsilon}_n] \\ &= \begin{bmatrix} (\boldsymbol{\epsilon}_{L+1})_1 & (\boldsymbol{\epsilon}_{L+2})_1 & \dots & (\boldsymbol{\epsilon}_n)_1 \\ \vdots & \vdots & & \vdots \\ (\boldsymbol{\epsilon}_{L+1})_p & (\boldsymbol{\epsilon}_{L+2})_p & \dots & (\boldsymbol{\epsilon}_n)_p \end{bmatrix}. \end{aligned}$$

The least squares estimate of this regression model is given by

$$\hat{B} = YZ^T(ZZ^T)^{-1},$$

and extracting the parameter matrices from $\hat{B}$ gives the estimates $\hat{A}_1, \dots, \hat{A}_L$.

Remark 48. Some other techniques to estimate the parameters of a VAR process include ridge regression (often provides a better bias vs variance tradeoff) or LASSO (when searching for sparse solutions) and these may be necessary to look into when running the graph extraction on real data.

Swartz's Bayesian Criterion

In the VAR model fitting, we had to pre-specify the order of the process L. If this is unknown, we can use Swartz's Bayesian criterion, a model selection

method. This essentially says to choose a model such that the value

$$SBC = -2logL + M\log(n)$$

is minimised, where M is the number of free parameters in the model to be estimated, n is the number of observations and $logL$ is the log likelihood of the model.

In the case of the multivariate linear regression $Y = BZ + U$ elicit when fitting a VAR process using least squares, if we assume Gaussian residuals with unknown variance σ^2 to be estimated, then

$$logL = \frac{-n}{2}(\log(2\pi) + \log(\sigma^2)) - \frac{1}{2\sigma^2}||Y - BZ||_F^2,$$

where the number of estimated parameters is $p^2L + 1$, as we estimate p^2L parameters in $\hat{B} = YZ^T(ZZ^T)^{-1}$ and one parameter in $\hat{\sigma}^2 = \frac{1}{n}||Y - BZ||_F^2$.

Using the fact that the residuals are Gaussian makes the SBC easy to calculate as the log likelihood function is known. Therefore even when this is not the case, it can still be a useful approximation.

Entropy Rate

To analyse information when dealing with time series, we use the notion of entropy rate [21] previously encountered in section 1.11.2.

If $\{\mathbf{S}_t\}$ is a Gaussian stationary process i.e. a VAR model with Gaussian innovations, the entropy rate is equivalent to

$$H_r(\{\boldsymbol{S}_t\}) = \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(2\pi e S_{\boldsymbol{S}}(f))df, \tag{4.2}$$

where $\boldsymbol{S}_{\boldsymbol{S}}(f)$ is the spectral matrix of $\{\mathbf{S}_t\}$.

Partial Mutual Information Rate

The conditional information rate of arbitrary random processes $\{\boldsymbol{X}_t\}$, $\{\boldsymbol{Y}_t\}$ given $\{\boldsymbol{Z}_t\}$ is defined in terms of entropy rates as

$$I_r(\{\boldsymbol{X}_t\},\{\boldsymbol{Y}_t\}|\{\boldsymbol{Z}_t\}) = H_r(\{\boldsymbol{X}_t\},\{\boldsymbol{Z}_t\}) + H_r(\{\boldsymbol{Y}_t\},\{\boldsymbol{Z}_t\}) \\ - H_r(\{\boldsymbol{Z}_t\}) - H_r(\{\boldsymbol{X}_t\},\{\boldsymbol{Y}_t\},\{\boldsymbol{Z}_t\}).$$

The partial mutual information rate (PMIR) between components i and j of a multivariate stochastic process $\{\boldsymbol{S}_t\}$, is then given as

$$PMIR_{ij} = I_r(\{S_{i,t}\},\{S_{j,t}\}|\{\boldsymbol{S}_{/ij,t}\}) = \\ H(\{\boldsymbol{S}_{/i,t}\}) + H(\{\boldsymbol{S}_{/j,t}\}) - H(\{\boldsymbol{S}_{/ij,t}\}) - H(\{\boldsymbol{S}_t\}),$$

where $\{\boldsymbol{S}_{/i,t}\}$ represents the process $\{\boldsymbol{S}_t\}$ with the series $\{S_{i,t}\}$ removed. Note that $PMIR_{ij} = 0$ implies components i and j are conditionally independent.

From (4.2), we see the entropy rate for Gaussian processes only depends on the spectral matrix of the arguments.

Invariance of Partial Mutual Information Rate

We show the PMIR for components of p-valued Gaussian stationary VAR process $\{\boldsymbol{S}_t\}$ and p-valued stationary process $\{\bar{\boldsymbol{S}}_t\}$ is invariant up to some unknown permutation where

$$\bar{\boldsymbol{S}}_t = Q^{-1}\boldsymbol{S}_t$$

and

$$Q = PD,$$

where $P \in \mathbb{P}$ and D is a diagonal matrix.

We first show that $\{\bar{\boldsymbol{S}}_t\}$ is also a VAR process.

Lemma 1. *If process* $\{\boldsymbol{S}_t\}$ *is a VAR process satisfying*

$$\boldsymbol{S}_t = \sum_{\tau=1}^{l} A_\tau \boldsymbol{S}_{t-\tau} + \boldsymbol{\epsilon}_t$$

and $\{\bar{\boldsymbol{S}}_t\}$ *satisfies*

$$\bar{\boldsymbol{S}}_t = Q^{-1}\boldsymbol{S}_t,$$

then $\{\bar{\boldsymbol{S}}_t\}$ *is a VAR process satisfying*

$$\bar{\boldsymbol{S}}_t = \sum_{\tau=1}^{l} \bar{A}_\tau \bar{\boldsymbol{S}}_{t-\tau} + \bar{\boldsymbol{\epsilon}}_t,$$

where $\bar{A}_\tau = Q^{-1}A_\tau Q$ *and* $\bar{\boldsymbol{\epsilon}} = Q^{-1}\boldsymbol{\epsilon}$.

Proof. We have

$$Q^{-1}\boldsymbol{S}_t = \sum_{\tau=1}^{l} Q^{-1}A_\tau \boldsymbol{S}_{t-\tau} + Q^{-1}\boldsymbol{\epsilon}_t$$

and therefore using $\bar{\boldsymbol{S}}_t = Q^{-1}\boldsymbol{S}_t$,

$$\bar{\boldsymbol{S}}_t = \sum_{\tau=1}^{l} Q^{-1}A_\tau Q \bar{\boldsymbol{S}}_{t-\tau} + Q^{-1}\boldsymbol{\epsilon}_t$$

□

Next we show the relationship between the spectral matrices of $\{\boldsymbol{S}_t\}$ and $\{\bar{\boldsymbol{S}}_t\}$.

Lemma 2. *Let* $\{\boldsymbol{S}_t\}$ *and* $\{\bar{\boldsymbol{S}}_t\}$ *be p-valued VAR processes satisfying*

$$\boldsymbol{S}_t = \sum_{\tau=1}^{l} A_\tau \boldsymbol{S}_{t-\tau} + \boldsymbol{\epsilon}_t$$

and

$$\bar{\boldsymbol{S}}_t = \sum_{\tau=1}^{l} \bar{A}_\tau \bar{\boldsymbol{S}}_{t-\tau} + \bar{\boldsymbol{\epsilon}}_t,$$

where $\bar{A}_\tau = Q^{-1}A_\tau Q$ *and* $\bar{\boldsymbol{\epsilon}} = Q^{-1}\boldsymbol{\epsilon}$*, then we have*

$$S_{\bar{\boldsymbol{S}}}(f) = Q^{-1}S_{\boldsymbol{S}}(f)(Q^{-1})^H,$$

where $S_{\bar{\boldsymbol{S}}}(f)$ *is the spectral matrix of* $\{\bar{\boldsymbol{S}}_t\}$ *and* $S_{\boldsymbol{S}}(f)$ *is the spectral matrix of* $\{\boldsymbol{S}_t\}$*.*

Proof. It is a well known that the spectral matrix of VAR processes in the form of $\{\boldsymbol{S}_t\}$ can be written

$$S_{\boldsymbol{S}}(f) = \Gamma_{\boldsymbol{S}}^{-1}(f)\Sigma_{\boldsymbol{\epsilon}}(\Gamma_{\boldsymbol{S}}^{-1}(f))^H,$$

where

$$\Gamma_{\boldsymbol{S}}(f) = I - \sum_{\tau=1}^{L} A_\tau e^{-i2\pi f\tau}$$

and $\Sigma_{\boldsymbol{\epsilon}}$ is the covariance matrix of $\boldsymbol{\epsilon}_t$.

Now, applying similar reasoning to the VAR process $\{\bar{\boldsymbol{S}}_t\}$ we see that

$$\begin{aligned}\Gamma_{\bar{\boldsymbol{S}}}(f) &= I - \sum_{\tau=1}^{L} \bar{A}_\tau e^{-i2\pi f\tau} \\ &= Q^{-1}(I - \sum_{\tau=1}^{L} A_\tau e^{-i2\pi f\tau})Q \\ &= Q^{-1}\Gamma_{\boldsymbol{S}}(f)Q,\end{aligned}$$

where we used the fact $\bar{A}_\tau = Q^{-1}A_\tau Q$.

We can write the covariance matrix of $\bar{\boldsymbol{\epsilon}}_t = Q^{-1}\boldsymbol{\epsilon}_t$ as

$$\Sigma_{\bar{\boldsymbol{\epsilon}}} = Q^{-1}\Sigma_{\boldsymbol{\epsilon}}(Q^{-1})^H.$$

Hence the spectral matrix of $\{\bar{\boldsymbol{S}}_t\}$ can be written

$$\begin{aligned}
S_{\bar{\boldsymbol{S}}}(f) &= \Gamma_{\bar{\boldsymbol{S}}}^{-1}(f)\Sigma_{\bar{\boldsymbol{\epsilon}}}(\Gamma_{\bar{\boldsymbol{S}}}^{-1}(f))^H \\
&= (Q^{-1}\Gamma_{\boldsymbol{S}}(f)Q)^{-1}Q^{-1}\Sigma_{\boldsymbol{\epsilon}}(Q^{-1})^H((Q^{-1}\Gamma_{\boldsymbol{S}}(f)Q)^{-1})^H \\
&= Q^{-1}\Gamma_{\boldsymbol{S}}^{-1}(f)QQ^{-1}\Sigma_{\boldsymbol{\epsilon}}(Q^{-1})^HQ^H(\Gamma_{\boldsymbol{S}}^{-1}(f))^H(Q^{-1})^H \\
&= Q^{-1}\Gamma_{\boldsymbol{S}}^{-1}(f)\Sigma_{\boldsymbol{\epsilon}}(\Gamma_{\boldsymbol{S}}^{-1}(f))^H(Q^{-1})^H \\
&= Q^{-1}S_{\boldsymbol{S}}(f)(Q^{-1})^H.
\end{aligned}$$

□

We now show the relationship between the entropy rates of $\{\boldsymbol{S}_t\}$ and $\{\bar{\boldsymbol{S}}_t\}$.

Lemma 3. *For Gaussian stationary processes $\{\boldsymbol{S}_t\}$ and $\{\bar{\boldsymbol{S}}_t\}$, with the following relationship between the spectral matrices*

$$S_{\bar{\boldsymbol{S}}}(f) = Q^{-1}S_{\boldsymbol{S}}(f)(Q^{-1})^H,$$

where

$$Q = PD$$

for some $P \in \mathbb{P}$ and D a diagonal matrix, then the entropy rates satisfy

(i)

$$H_r(\{\boldsymbol{S}_t\}) = H_r(\{\bar{\boldsymbol{S}}_t\}) + \log\det D,$$

(ii)

$$H_r(\{\boldsymbol{S}_{/i,t}\}) = H_r(\{\bar{\boldsymbol{S}}_{/\sigma(i),t}\}) + \log\det D_{/\sigma(i)},$$

where $\sigma : \{1,\ldots,p\} \to \{1,\ldots,p\}$ satisfies

$$\sigma(i) = \left[P\begin{pmatrix}1\\ \vdots \\ p\end{pmatrix}\right]_i.$$

$\{\boldsymbol{S}_{/i,t}\}$ represents the multivariate process $\{\boldsymbol{S}_t\}$ with component i removed and $D_{/i}$ represents the matrix D with row i and column i missing.

Proof. Case (i)

For a Gaussian stationary process, we have

$$\begin{aligned} H_r(\{\boldsymbol{S}_t\}) &= \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(2\pi e S_{\boldsymbol{S}}(f))df \\ &= \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(2\pi e)df + \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(S_{\boldsymbol{S}}(f))df. \end{aligned}$$

Now,

$$\begin{aligned} \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(S_{\boldsymbol{S}}(f))df &= \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(PDS_{\bar{\boldsymbol{S}}}(f)DP^{-1})df \\ &= \frac{1}{4\pi}\int_{-\pi}^{\pi} [\log\det(S_{\bar{\boldsymbol{S}}}(f)) + 2\log\det(D)]df \\ &= \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(S_{\bar{\boldsymbol{S}}}(f))df + \log\det(D), \end{aligned}$$

where we used the fact that the permutation matrix P satisfies

$$\det(P) = \det(P^{-1}) = 1.$$

Putting this back into the entropy rate equation gives

$$\begin{aligned} H_r(\{\boldsymbol{S}_t\}) &= \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(2\pi e S_{\bar{\boldsymbol{S}}}(f))df + \log\det(D) \\ &= H_r(\{\bar{\boldsymbol{S}}_t\}) + \log\det(D). \end{aligned}$$

Case (ii)

We need two results on matrices with rows and columns removed for a general matrix $X \in \mathbb{R}^{p\times p}$.

(a)
$$(DXD)_{/i} = D_{/i}X_{/i}D_{/i}$$
where $D \in \mathbb{R}^{p\times p}$ is a diagonal matrix.

(b)

$$(PXP^{-1})_{/i} = P_{/(i,\sigma(i))} X_{/\sigma(i)} P^{-1}_{/(\sigma(i),i)}$$

where $P \in \mathbb{P}$,

$$\sigma(i) = \left[P \begin{pmatrix} 1 \\ \vdots \\ p \end{pmatrix} \right]_i$$

and we use the following notation

(1) $Y_{/(i,j)}$ represents matrix Y with ith row and jth column removed

(2) $Y_{/i}$ represents matrix Y with ith row and ith column removed

(3) $Y_{/ij}$ represents matrix Y with ith and jth rows and columns removed removed

The first result (a) is obvious. The second result (b) follows from the fact the redundant rows and columns in X when calculating $(PXP^{-1})_{/i}$ are the ones that get sent to i, i.e. the $\sigma(i)$th row and column, so the relevant matrix is $X_{/\sigma(i)}$. The redundant row and column in the row permutation matrix P is the one that sends row $\sigma(i)$ to i i.e. the relevant matrix is $P_{/(i,\sigma(i))}$. Finally, the redundant row and column in the column permutation matrix P^{-1} is the one that sends column $\sigma(i)$ to i i.e. the relevant matrix is $P^{-1}_{/(\sigma(i),i)}$.

Now,

$$H_r(\{\boldsymbol{S}_{/i,t}\}) = \frac{1}{4\pi} \int_{-\pi}^{\pi} \log\det(2\pi e) df + \frac{1}{4\pi} \int_{-\pi}^{\pi} \log\det(S_{\boldsymbol{S}_{/i}}(f)) df,$$

where $S_{\boldsymbol{S}_{/i}}(f)$ is spectral matrix of $\{\boldsymbol{S}_t\}$ with ith row and column removed.

We also have

$$\begin{aligned} S_{\boldsymbol{S}}(f) &= Q S_{\bar{\boldsymbol{S}}}(f) Q^H \\ &= (PD) S_{\bar{\boldsymbol{S}}}(f) (PD)^H \\ &= PD S_{\bar{\boldsymbol{S}}}(f) D P^{-1}. \end{aligned}$$

Using our above results we have

$$S_{\boldsymbol{S}_{/i}}(f) = P_{/(i,\sigma(i))} D_{/\sigma(i)} S_{\bar{\boldsymbol{S}}_{/\sigma(i)}}(f) D_{/\sigma(i)} P^{-1}_{/(\sigma(i),i)}.$$

Now,

$$\begin{aligned}
\frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(S_{\boldsymbol{S}_{/i}}(f))df &= \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(P_{/(i,\sigma(i))} D_{/\sigma(i)} S_{\bar{\boldsymbol{S}}_{/\sigma(i)}}(f) D_{/\sigma(i)} P^{-1}_{/(\sigma(i),i)})df \\
&= \frac{1}{4\pi}\int_{-\pi}^{\pi} [\log\det(S_{\bar{\boldsymbol{S}}_{/\sigma(i)}}(f)) + 2\log\det(D_{/\sigma(i)})]df \\
&= \frac{1}{4\pi}\int_{-\pi}^{\pi} \log\det(S_{\bar{\boldsymbol{S}}_{/\sigma(i)}}(f))df + \log\det(D_{/\sigma(i)})
\end{aligned}$$

using the fact that by definition of σ, $P_{i\sigma(i)} = P^{-1}_{\sigma(i)i} = 1$. Therefore the matrices $P_{/(i,\sigma(i))}$ and $P^{-1}_{/(\sigma(i),i)}$ are still valid permutation matrices and have determinant equal to 1.

Hence, putting this back into the entropy rate for Gaussian processes equation,

$$H_r(\{\boldsymbol{S}_{/i,t}\}) = H_r(\{\bar{\boldsymbol{S}}_{/\sigma(i),t}\}) + \log\det D_{/\sigma(i)}.$$

□

We are now in a position to prove the invariance of the partial mutual information rate up to an unknown permutation.

Proposition 24. *Let $\{\boldsymbol{S}_t\}$ be a Gaussian stationary VAR process satisfying*

$$\boldsymbol{S}_t = \sum_{\tau=1}^{l} A_\tau \boldsymbol{S}_{t-\tau} + \boldsymbol{\epsilon}_t$$

and $\{\bar{\boldsymbol{S}}_t\}$ be a stochastic process satisfying

$$\bar{\boldsymbol{S}}_t = Q^{-1}\boldsymbol{S}_t,$$

where

$$Q = PD$$

for some $P \in \mathbb{P}$ *and* D *a diagonal matrix. Let the PMIR between components* i *and* j *of* $\{\boldsymbol{S}_t\}$ *be written* $PMIr_{ij}$ *and the PMIR between components* i *and* j *of* $\{\bar{\boldsymbol{S}}_t\}$ *be written* $P\bar{M}Ir_{ij}$*. Then,*

$$PMIr_{ij} = P\bar{M}Ir_{\sigma(i)\sigma(j)}$$

where $\sigma : \{1, \ldots, p\} \to \{1, \ldots, p\}$ *satisfies*

$$\sigma(i) = \left[P \begin{pmatrix} 1 \\ \vdots \\ p \end{pmatrix} \right]_i .$$

Proof. By lemma 1, $\{\boldsymbol{S}_t\}$ and $\{\bar{\boldsymbol{S}}_t\}$ are both Gaussian stationary VAR processes satisfying

$$\boldsymbol{S}_t = \sum_{\tau=1}^{l} A_\tau \boldsymbol{S}_{t-\tau} + \boldsymbol{\epsilon}_t$$

and

$$\bar{\boldsymbol{S}}_t = \sum_{\tau=1}^{l} \bar{A}_\tau \bar{\boldsymbol{S}}_{t-\tau} + \bar{\boldsymbol{\epsilon}}_t,$$

where $\bar{A}_\tau = Q^{-1} A_\tau Q$ and $\bar{\boldsymbol{\epsilon}} = Q^{-1}\boldsymbol{\epsilon}$.

Hence by lemma 2, we can write the spectral matrices of the processes as

$$S_{\bar{\boldsymbol{S}}}(f) = Q^{-1} S_{\boldsymbol{S}}(f) (Q^{-1})^H,$$

where $S_{\bar{\boldsymbol{S}}}(f)$ is the spectral matrix of $\{\bar{\boldsymbol{S}}_t\}$ and $S_{\boldsymbol{S}}(f)$ is the spectral matrix of $\{\boldsymbol{S}_t\}$.

By lemma 3 the entropy rate of both processes satisfies

$$H_r(\{\boldsymbol{S}_{/i,t}\}) = H_r(\{\bar{\boldsymbol{S}}_{/\sigma(i),t}\}) + \log\det D_{/\sigma(i)}.$$

All that's left is to show the equivalence of the PMIR up to an unknown

permutation. By definition of PMIR,

$$\begin{aligned}
PMIr_{ij} &= H_r(\{\boldsymbol{S}_{/i,t}\}) + H_r(\{\boldsymbol{S}_{/j,t}\}) - H_r(\{\boldsymbol{S}_{/ij,t}\}) - H_r(\{\boldsymbol{S}_t\}) \\
&= P\bar{M}Ir_{\sigma(i)\sigma(j)} + \log\det(D_{/\sigma(i)}) + \log\det(D_{/\sigma(j)}) \\
&- \log\det(D_{/\sigma(i)\sigma(j)}) - \log\det(D) \\
&= P\bar{M}Ir_{\sigma(i)\sigma(j)} + \sum_{k=1,k\neq\sigma(i)}^{p} \log(D_{kk}) + \sum_{k=1,k\neq\sigma(j)}^{p} \log(D_{kk}) \\
&- \sum_{k=1,k\neq\sigma(i),k\neq\sigma(j)}^{p} \log(D_{kk}) - \sum_{k=1}^{p} \log(D_{kk}) \\
&= P\bar{M}Ir_{\sigma(i)\sigma(j)}.
\end{aligned}$$

□

MODEL

We adopt the model as mentioned in [39] and [33], that the source signals are governed by a VAR process with non-Gaussian spatially independent residuals as they are separated in space and come from resting patients (spatially independent residuals as separated in space and assumed no underlying driving process e.g. finger tapping). We observe signals on the scalp that are assumed to be an instantaneous mixing of the source signals. This mixing is caused by volume conduction and the instantaneous assumption from the fact the electromagnetic signal travels from the source to the scalp at the speed of light. We also make the assumption that the number of sources is equal to the number of significant factors after applying PCA to the scalp signals.

Remark 49.

1. If the number of source signals is greater than the number of scalp signals, the problem is underdetermined and additional assumptions would be necessary to recover a model. For statistical analysis when using EEG data, it is a common assumption to make that we have more signals than relevant sources [33].

2. In practice, the brain is a very complex object and it is far from obvious

to know a priori the number of sources it contains that are significant to our application. Instead, we proceed as in [33] where we estimate the number of sources by applying PCA to the observed scalp signals and reducing the dimension so that the transformed series adequately explains a certain proportion of the variance in the data.

Model Equations

Mathematically, we can write our model using the following equations

$$\boldsymbol{X}_t = C\tilde{\boldsymbol{X}}_t, \tag{4.3}$$

$$\tilde{\boldsymbol{X}}_t = W\boldsymbol{S}_t, \tag{4.4}$$

$$\boldsymbol{S}_t = \sum_{\tau=1}^{L} A_\tau \boldsymbol{S}_{t-\tau} + \boldsymbol{\epsilon}_t, \tag{4.5}$$

where

1. $\{\tilde{\boldsymbol{X}}_t\}$ is a $\tilde{p}$ dimensional stochastic process corresponding to signals observed on the scalp (potentially processed to remove any unwanted artefacts).
2. $C \in \mathbb{R}^{p\times\tilde{p}}$ is a PCA transformation matrix.
3. $\{\boldsymbol{X}_t\}$ is a p dimensional stochastic process corresponding to factors that when linearly transformed explain a certain percentage of the variance in our scalp observations.
4. $\{\boldsymbol{S}_t\}$ is a p dimensional stochastic process corresponding to source signals.
5. $W \in \mathbb{R}^{\tilde{p}\times p}$ is a mixing matrix (determined by physical volume conduction effects).

6. $\{A_\tau \in \mathbb{R}^{p\times p} | \tau = 1 \dots L\}$ is a set of VAR co-efficients.

7. $\{\boldsymbol{\epsilon}_t\}$ is a spatially and temporally independent innovation process with a non-Gaussian distribution.

Remark 50. The above model contains all information about the relationships between sources in the VAR co-efficients.

Combining equations (4.3), (4.4) and (4.5) we obtain

$$\boldsymbol{X}_t = \sum_{\tau=1}^{L} CWA_\tau (CW)^{-1} X_{t-\tau} + (CW)\boldsymbol{\epsilon}_t. \tag{4.6}$$

Letting $F_\tau = CWA_\tau(CW)^{-1}$ and $\boldsymbol{\nu}_t = CW\boldsymbol{\epsilon}_t$, we have

$$\boldsymbol{\nu}_t = \boldsymbol{X}_t - \sum_{\tau=1}^{L} F_\tau X_{t-\tau}. \tag{4.7}$$

We can therefore estimate the mixing matrix W (up to scale and permutation reference or preliminaries for BSS) by applying PCA to calculate $\boldsymbol{X}_t$ from $\tilde{\boldsymbol{X}}_t$, fitting a VAR process to $\boldsymbol{X}_t$ and then applying ICA to the residuals $\boldsymbol{\nu}_t = CW\boldsymbol{\epsilon}_t$.

Remark 51. This is not the only source separation method for time series data. We chose the model due to its speed, simplicity and proven performance [33]. It also makes intuitive sense in our application for the brain, where the sources are physically separated in space (spatial independence) but interact by passing signals to one another (temporal dependence). Note however that if we fail to include a source that temporally drives two or more other sources at a certain time lag, the driven sources will appear to be spatially dependent.

Model Parameter Estimation

Pre Processing

The pre-processing of the raw observed data to obtain the scalp signals $\tilde{\boldsymbol{X}}_t$ is very application dependent. Raw time series data often contains artefacts that must be removed before many methods of analysis can be applied and there are different methods to achieve this. As this is not the focus of the research, we do not go into detail on this topic.

PCA

Our first step in the algorithm as in [33] is to apply PCA to processed EEG data, $\tilde{\boldsymbol{X}}_t$ to recover our factored scalp signal time series $\boldsymbol{X}_t$. This is done by only including as many principal components as necessary to account for a certain fraction of the variance in the observed signal $\tilde{\boldsymbol{X}}_t$. This fraction is a parameter in the model and we denote it γ (a common value chosen for it may be $\gamma = 0.99$).

Application of PCA allows a more robust VAR parameter estimation as it avoids a badly conditioned covariance matrix. It also has the added benefit that the VAR estimation is faster due to the dimension being reduced.

PCA is therefore used to estimate C and $\boldsymbol{X}_t$ in equation (4.3).

VAR Fitting

Fitting a VAR model to (4.7) can be done efficiently using L2 regression. This provides a consistent estimator of the VAR coefficient matrices even if the innovations $\boldsymbol{\nu}_t$ are not spatially uncorrelated [43].

Remark 52. In [33] they used the algorithm ARfit [70] to fit the VAR coefficient matrices which is claimed to be more efficient than L2 regression for large dimensions. We found a standard least square L2 regression to be sufficient as the bottleneck for larger dimensions occurred further along in our algorithm.

Order Selection

Order selection of VAR processes is a complex field in its own right. We note that [33] select the order l of their VAR process choosing the minimum $l \in \mathbb{L}$ that achieves 90% of the reduction of $SBC(l_{\min})$ to $SBC(l_{\max})$ (this is the maximum reduction over $\mathbb{L}$ of the SBC, where $\mathbb{L}$ is some set of integers to search over and $SBC(l)$ represents the SBC of the model with order l. The reasoning behind this is that the SBC was usually monotonic decreasing on $\mathbb{L}$ for EEG data i.e. there was no obvious minimum not at a boundary point.

We did not find this was the case when working with simulated data (as expected) so were always able to find a clear minimum for the SBC. If we apply the method to EEG data however we may have to undertake this step as above.

Note that in [33] they do not state how they obtained the log likelihood which is necessary to calculate the SBC. The innovations are assumed non-Gaussian but the exact distribution is unspecified. For our purpose we use an approximation to the true log likelihood, the log likelihood given the innovations are Gaussian but note there may be a more accurate way to do this.

Combining both the order selection step and the VAR fitting step allows us to estimate the order of the VAR process l and thus $\boldsymbol{\nu}_t$ and the matrices F_τ in equation (4.7).

Remark 53. This selection procedure could certainly be improved upon by other methods but this is outside the scope of the focus of this research.

ICA

Once the innovation process $\boldsymbol{\nu}_t = CW\boldsymbol{\epsilon}_t$ has been recovered, we can apply ICA to recover CW up to an unknown scaling and permutation of the columns. This is because the $\boldsymbol{\epsilon}_t$ are assumed independent, non-Gaussian and $CW \in \mathbb{R}^{p\times p}$ is square and non-singular. In order to perform this, we use the FastICA algorithm [40].

We can now recover an estimate of the sources

$$\bar{\boldsymbol{S}}_t = (C\bar{W})^{-1}\boldsymbol{X}_t$$

where $C\bar{W} = CWQ$ such that $Q = PD$ for some $P \in \mathbb{P}$ and D a diagonal matrix. Hence,

$$\bar{\boldsymbol{S}}_t = Q^{-1}(CW)^{-1}\boldsymbol{X}_t = Q^{-1}\boldsymbol{S}_t$$

and the recovered source VAR coefficients are

$$A_\tau = C\bar{W}F_\tau(C\bar{W})^{-1}.$$

Remark 54. Our true sources and recovered sources satisfy the relationship in proposition 24.

Building Conditional Independence Graphs

We now look to build graphical models from our estimated parameters as a concise representation of the connectivity between sources. The idea is that we can then make inference based on these graphical models more efficiently than using the whole time series. It is also commonly assumed that brain function is a product of the connectivity between sources but not the actual time series themselves. For example someone with schizophrenia may have two sources within their brain that do not communicate whereas these sources communicate for a non-schizophrenia patient. However the actual communication between these areas may be vastly different for alternative non-schizophrenia patients. This implies that classification and clustering (for diagnosing mental disorders) should be carried out on graphical models of connectivity as opposed to the time series themselves. We therefore think of the graphical models as an efficient dimension reduction of the original time series, capturing the relevant information.

We show how we can extract labelled graphical models from the estimated source process. The nodes represent a channel of our source process. The node labels are based on the columns of the mixing matrix (which can be

used to get a rough estimate of the location of the sources within a patients brain) and the connections between nodes come from the estimated source VAR coefficients. Combining both of these noisy estimates we aim to achieve a more powerful statistical diagnosis tool than using either one on its own. We shall see this later in chapter 7.

We base our conditional independence graphs on the partial mutual information rate (PMIR) between components of our extracted source series. Estimating the PMIR for non-Gaussian stationary VAR processes is relatively difficult and requires estimating the distribution of the residuals. Instead we approximate it by using the PMIR given the sources are Gaussian (which is clearly not the case as we assumed they are non-Gaussian for the ICA step). We justify the approximation as it preserves the conditional independence structure between the process components as seen in proposition 25. It is also justified empirically in the results section.

Proposition 25. *If* $\boldsymbol{X}_t = \sum_{\tau=1}^{L} A_\tau \boldsymbol{X}_{t-\tau} + \boldsymbol{\epsilon}_t$ *and* $\boldsymbol{Y}_t = \sum_{\tau=1}^{L} A_\tau \boldsymbol{Y}_{t-\tau} + \boldsymbol{\eta}_t$ *where* $\boldsymbol{\epsilon}_t$ *is Gaussian white noise and* $\boldsymbol{\eta}_t$ *is non-Gaussian independent white noise, then series* $\{X_{i,t}\}$ *and* $\{X_{j,t}\}$ *conditionally independent* $\iff$ $\{Y_{i,t}\}$ *and* $\{Y_{j,t}\}$ *conditionally independent.*

Proof. For independent residuals (whether Gaussian or not), all dependency information between components is contained in the parameters of the VAR process. □

Remark 55. After BSS we can only recover our unmixing matrix up to some unknown scaling and permutation of the columns. Proposition 24 says that when using the partial mutual information rate, the corresponding graph based on the recovered unmixing matrix is equivalent to the graph based on the true unmixing matrix up to some permutation of the nodes i.e. the unknown scaling is now irrelevant. This can be used because as we showed earlier,

$$\bar{\boldsymbol{S}}_t = Q^{-1}(CW)^{-1}\boldsymbol{X}_t = Q^{-1}\boldsymbol{S}_t,$$

where $Q = PD$ for some $P \in \mathbb{P}$ and D a diagonal matrix.

Edge Values

We now look at how to infer if there is an edge between two nodes in our graph i.e. if two of our sources are conditionally dependent. We use a simple thresholding method to determine whether to include an edge in our graph based on the PMIR. Ideally the approach would be extended to a multiple hypothesis test as in the prior chapter alleviating the need to specify a thresholding parameter.

Algorithm 11 Thresholding algorithm for determining edges in conditional independence graph

Require: Recovered VAR coefficient matrices $\bar{A}_\tau$, recovered innovations $\bar{\boldsymbol{\epsilon}}_t$, threshold $\delta > 0$

Calculate diagonal $\Sigma_{\bar{\boldsymbol{\epsilon}}}$ such that $\Sigma_{\bar{\boldsymbol{\epsilon}},ii}$ is the sample variance of $\bar{\epsilon}_{i,t}$
Calculate $\bar{\Gamma}(f) = I - \sum_{\tau=1}^{L} \bar{A}_\tau e^{-i2\pi f\tau}$
Calculate $\bar{S}(f) = \bar{\Gamma}^{-1}(f)\Sigma_{\bar{\boldsymbol{\epsilon}}}(\bar{\Gamma}^{-1}(f))^H$
Initialise graph $G = (V, E)$ such that $V = \{1, \ldots, p\}$ and $E = \emptyset$
for $i = 1 : p - 1$ **do**
    for $j = i + 1 : p$ **do**
        Calculate $x = P\bar{M}Ir_{ij}$ using $\bar{S}(f)$
        if $x > \delta$ **then**
            $E \leftarrow E \cup \{\{i, j\}\}$
        end if
    end for
end for
return G

Remark 56. Ideally we can choose the threshold value a priori by considering statistical properties of the PMIR. Alternatively, it may be chosen retroactively as part of a scheme to maximise inter-cluster and minimise intra-cluster entropy, if we are extracting many graphs from different patients.

Node Labels

In [33], the mixing matrix W is used to produce an intra-cranial localisation of each source using the swLoretta algorithm [75] i.e. a 3D point in space. Intuitively in our application, the position of sources would make a natural label for the nodes in our graph. When comparing graphs from different patients we can then use both the dependence structure represented by the edges and the model labels.

In order to determine the labels, we recall

$$\tilde{\boldsymbol{X}}_t = W\boldsymbol{S}_t.$$

This is exactly the form of the source and signal relationship for swLoretta where the mixing matrix W is referred to as the leadfield matrix and there is a further noise process added to the transformed sources. Hence we could use this method as in [33] to get an estimated 3D localisation for our sources within the brain.

We note however that the information necessary for transforming a source into 3D space is contained in the columns of W. We assume the recovered source channels have unit variance (to allow for a more determinable model), if one patient has recovered mixing matrix column i equal to $\boldsymbol{u}$ and another has recovered mixing matrix column j equal to $\boldsymbol{v}$, then an intuitive measure of distance between the localisation of the first patient's source i and the second's source j is then the Euclidean distance

$$||\boldsymbol{u} - \boldsymbol{v}||_2.$$

We therefore choose to use the mixing matrix columns as labels for our graph nodes as opposed to the transformed 3D point in space.

Remark 57. An obvious extension would be to apply swLoretta and use the 3D points as the node labels. However as mentioned previously, the necessary information is contained in the mixing matrix columns so we chose the above approach due to restriction on both processing and research time. Whether

using 3D locations as the labels for nodes works better is a topic for further study.

Algorithm

Given an observed time series from the scalp of a patient we now present an algorithm for converting this to a labelled graphical model up to some unknown permutation of the nodes.

Algorithm 12 Labelled graph extraction from time series data

Require: $\tilde{X} \in \mathbb{R}^{\tilde{p} \times N}$ time series data, γ PCA variance ratio, $l_{\min}, l_{\max}$ bounds on the VAR process order and l_{step} the size of the steps between candidate orders

1: Apply PCA with variance ratio γ to $\tilde{X}$ to get $C \in \mathbb{R}^{p \times \tilde{p}}$ and $X = C\tilde{X}$
2: Choose order l that attains the minimum BIC value over $l_min : l_step : l_max$
3: Fit VAR process of order l to X and obtain residuals $\boldsymbol{\nu}_t$ and coefficient matrices F_τ
4: Apply FastICA to $\boldsymbol{\nu}_t$ to recover $C\bar{W}$
5: Calculate $\bar{\boldsymbol{S}}_t = (C\bar{W})^{-1}\boldsymbol{X}_t$
6: Calculate $\bar{A}_\tau = (C\bar{W})^{-1}F_\tau(C\bar{W})$
7: Calculate $\bar{\boldsymbol{\epsilon}}_t = (C\bar{W})^{-1}\boldsymbol{\nu}_t$
8: Calculate graph G using the thresholding algorithm 11 with $\bar{A}_\tau$ and $\bar{\boldsymbol{\epsilon}}_t$ (the conditional independence graph for $\boldsymbol{S}_t$ up to some unknown permutation)
9: Calculate $\bar{W} = C^+(C\bar{W})$
10: Label the vertices in G using vector $L \in \mathbb{R}^{\frac{p(p-1)}{2} \times \tilde{p}}$ (a vector of row vectors) by setting $L_i = \bar{W}_i$
11: **return** (G, L)

Remark 58. Note that in general it is not possible to find the solution to $\bar{W} = C^{-1}(C\bar{W})$ as C is not necessarily a square matrix. We therefore use the Moore-Penrose pseudoinverse C^+ and calculate $\bar{W} = C^+(C\bar{W})$ instead.

Entry	PMIR
(1,1)	1.2944
(1,2)	0.0661
(1,3)	0.0208
(1,4)	0.0280
(1,5)	0.0833
(2,2)	1.4064
(2,3)	0.0000
(2,4)	0.0209
(2,5)	0.0000
(3,3)	1.4263
(3,4)	0.0000
(3,5)	0.0045
(4,4)	1.2931
(4,5)	0.0868
(5,5)	1.3066

Table 4.1: True approximate PMIR values based on model A VAR parameter matrix with Gaussian innovations

Results

The main result we wish to empirically justify is that the estimated approximate PMIR of our extracted source process is close to a permutation of the approximate PMIR of the true source process. To do this, we use the VAR model A, which has approximate PMIR under independent Gaussian innovations for each series component given in table 4.1 (note as the PMIR is symmetrical we only report the upper triangular entries).

Note that entries corresponding to series $(2,3),(2,5)$ and $(3,4)$ are zero as expected, as these are the conditionally independent series in the model for independent innovations. We now generate 100 length 1024 time series using model A but with independent generalised Gaussian (i.e. non-Gaussian) innovations with pdf

$$f(x) = \frac{\beta^{1/2}}{2\Gamma(1+1/\rho)} \exp(-\beta^{\rho/2}|x-\mu|^{\rho}),$$

Entry	Mean	SD
(1,1)	1.2935	0.0190
(1,2)	0.0655	0.0180
(1,3)	0.0222	0.0119
(1,4)	0.0310	0.0093
(1,5)	0.0796	0.0110
(2,2)	1.3971	0.0117
(2,3)	0.0047	0.0055
(2,4)	0.0235	0.0158
(2,5)	0.0055	0.0059
(3,3)	1.4188	0.0079
(3,4)	0.0038	0.0052
(3,5)	0.0078	0.0062
(4,4)	1.2938	0.0179
(4,5)	0.0851	0.0138
(5,5)	1.3091	0.0181

Table 4.2: Estimated approximate PMIR values based on model A VAR parameter matrix with Gaussian innovations

where we choose $\mu = 0$, $\beta = 1$ and $\rho = 3$.

For each of the generated time series (from a non-Gaussian stationary stochastic process), we extract the estimated VAR parameter matrices using the method in this chapter. We then calculate the approximate PMIR given the estimated VAR parameter matrices and assuming Gaussian innovations. Finally we use graph matching to align the estimated approximate PMIR with the true approximate PMIR. Table 4.2 is the mean estimated approximate PMIR matrix and standard deviation over the 100 iterations.

As we can see, the distribution of the estimated approximate PMIR is very close to the true approximate PMIR, indicating the method works and can be used to extract graphical models of source signals from mixed signals recorded on the scalp. Also note that encouragingly, the only entries that contain 0 within 1 standard deviation are in fact the missing edges in the model. The $(3, 5)$ entry however is not missing but is also very close to 0. This means it would require a fairly accurately picked threshold in order to

extract (a graph isomorphic to) the true graphical model using our method.

Conclusion and Future Work

In this chapter we developed a graph extraction algorithm that allows us to build graphical models based on source signals when we only observed a linear mixing of these signals. The algorithm closely follows the work in [33] to extract the source signals. It uses a different measure of conditional dependence, namely the partial mutual information rate.

It was shown that an approximation to the true PMIR, the PMIR using extracted VAR parameter matrix and assumed Gaussian innovations, contained the necessary information to create a conditional independence graph for the source signals. This was also justified empirically. The graph however is only known up to some permutation of the nodes and further analysis must be done to use it in a clustering algorithm by taking into consideration multiple patients. In [33] they looked to find the position of the nodes by analysing the unmixing matrix obtained through ICA. We however believe that by also taking into consideration the structure of the extracted graphs, a more powerful algorithm can be developed. This is investigated further in later chapters.

The main area of future work is in developing an automated threshold. Currently it must be explicitly specified. Ideally we could have a multiple hypothesis test as in the earlier chapter where all that needs to be specified is an error rate control parameter and the threshold would be automatically calculated from this. Currently it is hard to link the threshold to anything tangible.

Furthermore, we would like to more extensively test the graph extraction algorithm for larger dimensional models and also on real EEG data.

5

Clustering and Classification with Known Node Position

For completeness we include a chapter on clustering and classification of scalp signal graphs. In the previous chapter we saw how to reduce a very high dimensional time series object to a graphical model. This is far more tangible to apply classification and clustering techniques to. Of course this is based on the assumption the graph captures enough of the key discriminatory information from the time series. In this chapter we look at how to apply these techniques to a number of observed graphs with known node position e.g. vertex 2 of graph 1 matches with vertex 2 of graph 2. If the node position were not known, it is unknown which vertex of graph 2 the vertex 2 of graph 1 matches with. We will look at this second case in later chapters.

Introduction

We initially look at some common classification and clustering techniques when applied to features extracted from our graphical models. In particular, we consider logistic regression and k-means. Both are widely used and

provide a benchmark for testing more advanced techniques. Their effectiveness as with all learning algorithms, is highly dependent on the features we extract from our graphs that act as inputs. A naive extraction would be to use each potential edge in a graph as a variable. We can then create a vector containing a one in entry i if the corresponding edge indexed by i exists in the graph and a zero otherwise. This however can lead to a very large number of (low entropy) variables. A better method would be to reduce these down to a smaller number of more informative variables. This method also doesn't take into account dependency between edges.

An initial idea to improve our graph feature extraction is to use graph centrality [9] to encapsulate the edge information in the nodes. Essentially, if a node is connected to other important nodes, it itself has high importance. This leads to a balancing equation that can be solved by singular value decomposition (SVD). This technique reduces the feature dimension from $p(p-1)/2$ to p and to a degree takes into account dependency between edges. The vector of centralities can then be used in a learning algorithm as opposed to the vector of edges. Once again however, the number of features p may still be very high for large graphs with a low number of dimensions which can lead to issues such as overfitting.

A further idea is that the important features are actually contained in subgraphs of the original graphs. This not only reduces dimension by considering smaller subgraphs, but the exact dependency information of the edges within the subgraph can be taken into account. We do this by modifying the random forest [10] algorithm such that at each split we choose a random subset of vertices (as opposed to a random subset of edges which we treat as features). The split is then based on the edge structure of this subgraph. To extend this to clustering, we can use the work from [73] that extended the random forest algorithm to clustering.

The chapter is set out as follows, section 5.1 contains preliminary results. Section 5.2 and 5.3 contains classification and clustering overviews respectively, along with descriptions of logistic regression and k-means. Section 5.4 then looks at how to extract features from graphical models to be used in learning algorithms. In particular we look at the naive approach of using a

vector indicating whether edges exist and the idea of graph centrality. The idea of using random forests is then discussed in section 5.5 where we introduce modified algorithms based on them for classification and clustering. Finally we compare the above methods in a small dimension and large dimension study in section 5.6. While we do not include tests on real data, we bear in mind when discussing the algorithms the application to graphical models based on EEG data from a set of patients labelled with negative syndrome schizophrenia and controls. The chapter is ended with some concluding remarks and ideas for further work.

Remark 59. The modified random forest algorithm is heavily dependent on the choice of the 'randomness' when creating the subgraphs. For example we select subgraphs by choosing nodes uniformly at random. However it may make sense that if we already have a certain number of nodes then we should add other nodes to the subgraphs that many of the nodes are connected to, with higher probability than a node none are connected to. The size can also be fixed beforehand or in a similar vein to above we may have an 'algorithm' that searches for random subgraphs at each step based on some measure of how useful (discriminatory) they may be. The size could therefore vary between splits.

Preliminaries

Multidimensional Scaling

Multidimensional scaling provides a way of transforming points into some best fitting Euclidean space where we are provided only with a dissimilarity matrix representing the distances between the points. There are many different versions but in this chapter we use classical multidimensional scaling (CMDS) [71].

Classification

Overview

A version of the classification problem we consider is, given some labelled observations from C classes

$$y_1, \ldots, y_n \in \{1, \ldots, C\}$$

and some associated features for each observation

$$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^m,$$

then we want to find for some class of functions sending the features to a probability vector over the classes

$$\boldsymbol{f}_{\boldsymbol{\theta}} : \mathbb{R}^m \to [0,1]^C$$

optimal parameters $\boldsymbol{\theta}$ solving

$$\arg\min_{\boldsymbol{\theta}} d(\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)_{y_i} | i = 1, \ldots, n\})$$

where d is some aggregated measure of distance. A commonly used distance is the likelihood of the observations

$$d(\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)_{y_i} | i = 1, \ldots, n\}) = \prod_{i=1}^{n} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)_{y_i}.$$

Remark 60. In our case, the observed features must come from graphs. We investigate how to process these into a vector of real numbers in a later section.

Logistic Regression

We briefly discuss one of the most common classification techniques, logistic regression. Note we only look at binary observation classes (as in traditional

logistic regression) as this is what relates to our application of negative syndrome schizophrenia and controls.

Method

Let $Y \in \{1, 2\}$ be a random variable representing our observation labels such that given some observed features $\boldsymbol{x} \in \mathbb{R}^m$ and parameter vector $\boldsymbol{\beta}$, it is conditionally Bernoulli distributed such that

$$\Pr(Y = 1|\boldsymbol{x}, \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}\cdot\boldsymbol{x}}}{1 + e^{\boldsymbol{\beta}\cdot\boldsymbol{x}}}$$

and of course

$$\Pr(Y = 2|\boldsymbol{x}, \boldsymbol{\beta}) = 1 - \Pr(Y = 1|\boldsymbol{x}, \boldsymbol{\beta}).$$

Note that in this case we have classification function

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \begin{bmatrix} \Pr(Y = 1|\boldsymbol{x}, \boldsymbol{\theta}) \\ \Pr(Y = 2|\boldsymbol{x}, \boldsymbol{\theta}) \end{bmatrix}.$$

Estimation

Given observation labels $y_1, \ldots, y_n$ and features $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, the above model gives the likelihood function for $\boldsymbol{\beta}$

$$\mathcal{L}(\boldsymbol{\beta}|y_1, \ldots, y_n, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \prod_{i=1}^{n} \left(\frac{e^{\boldsymbol{\beta}\cdot\boldsymbol{x}}}{1 + e^{\boldsymbol{\beta}\cdot\boldsymbol{x}}} \right)^{I(y_i=1)} \left(\frac{1}{1 + e^{\boldsymbol{\beta}\cdot\boldsymbol{x}}} \right)^{I(y_i=2)}. \quad (5.1)$$

Maximising the log transform of (5.1) can be readily solved computationally to find a maximum likelihood value for $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}$. This gives our optimal classification function $\boldsymbol{f}_{\hat{\boldsymbol{\beta}}}$.

Remark 61. For an in depth review of logistic regression see [37].

Clustering

Overview

A version of the clustering problem we consider is, given some observed real features

$$\boldsymbol{x}_1, \ldots, x_n \in \mathbb{R}^m$$

finding k partitions of the observations maximising intra-cluster similarity.

For example if we consider the case $m = 1$ and our intra-cluster similarity measure as Euclidean distance between a point and the mean of the cluster, we may write the clustering problem as

$$\arg \min_{I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} |x_j - \mu_i|,$$

where $I_1, \ldots, I_k$ form a partition of $\{1, \ldots, n\}$ and μ_i is the mean of the points in cluster i represented by partition I_i.

Remark 62. Once again in our application, the observed features must come from graphs and we investigate how to process them into vectors of real numbers later.

k-means

One of the most common clustering techniques is k-means and can form the basis for more complex clustering techniques.

Model

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^m$ be some observed features. Then given total clusters k, we want to find partition $I_1, \ldots, I_k$ of $\{1, \ldots, n\}$ satisfying

$$\arg\min_{I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} ||\boldsymbol{x}_j - \boldsymbol{\mu}_i||^2, \tag{5.2}$$

where $\boldsymbol{\mu}_i = \frac{1}{|I_i|} \sum_{j \in I_i} \boldsymbol{x}_j$ and $|| \cdot ||$ is Euclidean distance.

Estimation

Finding the solution to (5.2) is in general NP-hard. The most common algorithm used that is guaranteed to find a local minimum is Lloyd's algorithm given in algorithm 13.

Algorithm 13 k-means clustering

Require: Observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^m$, total clusters k, initialisation method

Ensure: $p \leq 2M + 1$

$\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \leftarrow$ initialise cluster means according to initialisation method
while convergence criteria not met **do**
    $\mathbb{K}_1, \ldots, \mathbb{K}_k \leftarrow$ set centroid sets to $\emptyset$
    for $i = 1 : n$ **do**
        $k_i \leftarrow \arg\min_{j \in \{1,\ldots,k\}} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2$
        Add $\boldsymbol{x}_i$ to $\mathbb{K}_{k_i}$
    end for
    Recalculate $\boldsymbol{\mu}_j = \frac{1}{|k_j|} \sum_{i \in \mathbb{K}_j} \boldsymbol{x}_i$
end while
Return $\mathbb{K}_1, \ldots, \mathbb{K}_k$

Remark 63.

1. As k-means is guaranteed to converge to a local minimum, our convergence criteria is simply when the objective function (5.2) no longer improves between while iterations.

2. There are many different initialisation methods possible. For example the Forgy method chooses k observations at random from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and uses these as the initial means. Another, the Random Partition method gives each observation an initial cluster uniformly randomly

chosen from $\{1, \ldots, k\}$. The means are then initialised as the means of these assigned clusters.

3. For a detailed overview of k-means see [87].

Graph Feature Extraction

In this section we look at how we can extract features from graphical models to use in classification and clustering algorithms.

Edges

The most basic feature extraction is to simply vectorise an indicator of whether the edges in the graph exist i.e. if we have a graph $G \in \mathbb{G}_p$, transform this to a feature vector $\boldsymbol{x} \in \{0,1\}^{p(p-1)/2}$ such that

$$x = \begin{cases} 1 & \text{if edge indexed by } i \text{ exists in } G \\ 0 & \text{o/w} \end{cases}.$$

Certainly then in this case $\boldsymbol{x} \in \mathbb{R}^{p(p-1)/2}$ and it can be used in both a logistic regression classification and a k-means clustering.

Remark 64. This method has a number of issues as mentioned in the introduction. In particular, the number of features can become very large for high dimensional graphs, something that may cause classification algorithms to overfit. Secondly, treating the edges as independent features doesn't take into account dependency/interaction between them which may be important.

Centrality

A way to overcome some of the issues in the previous approach is to use values assigned to the nodes of the graph based on their adjoining edges. This reduces the dimension to p features (one for each node) and also takes into account some degree of dependency between the edges. To accomplish

this we use the idea of graph centrality and in particular eigenvector centrality [9].

We assign a centrality c_k to node k to represent its importance within the network. The centralities satisfy

$$c_k = \frac{1}{\alpha} \sum_{q \in N_k} c_q,$$

where N_k is the set of adjacent nodes to k and α is a constant. The interpretation of this formula is that the centrality/importance of a node in a graph is a linear sum of the importance of all the nodes it is directly connected to. This can be reformulated as

$$\alpha \boldsymbol{c} = A\boldsymbol{c},$$

where $A \in \mathbb{R}^{p \times p}$ is the adjacency matrix of the graph and $\boldsymbol{c}$ the p dimensional vector of centralities. Thus $\boldsymbol{c}$ satisfies our conditions if it is an eigenvector of A and we choose it to be the principal eigenvector so that it has non-negative entries.

Remark 65.

1. As we are dealing with undirected graphs, our adjacency matrices are symmetric and hence can be decomposed into $A = QDQ^T$ where Q is an orthogonal matrix of eigenvectors and D is a diagonal matrix of eigenvalues.

2. If we use a weighting on our edges, we can define centrality as

$$c_k = \frac{1}{\alpha} \sum_{q=1}^{p} W_{qk} c_q,$$

where W_{qk} is the weight assigned to edge $\{q, k\}$. This gives the resulting expression

$$\alpha \boldsymbol{c} = \boldsymbol{W}\boldsymbol{c}$$

where $\boldsymbol{W} \in \mathbb{R}^{p \times p}$ is the weighted adjacency matrix.

3. The vector of centralities $\boldsymbol{c} \in \mathbb{R}^p$ can be used as features in our classification and clustering algorithms. We note that for large p, further dimension reduction may be necessary to achieve good results.

4. Interestingly, if we were to only know the position of the nodes of each graph up to some permutation, the centrality values would simply be reordered. Hence if this were the case it may be possible to use some ordered graph centrality values to discriminate between graphs with unknown permutations. We explore these ideas in depth in later chapters.

Random Forests

We investigate both clustering and classification algorithms based on random forests [10]. This is a supervised learning method that generates an ensemble of decision trees built from the dataset, that then vote on the class an input should belong to. Clearly without any inherent changes in the generation of the trees from our dataset, they will all be the same which is of no use when using them in an ensemble. Instead, to reduce correlation between the individual trees the first step to building one is to bootstrap the training data by sampling with replacement a new dataset. The tree is then built using this bootstrapped dataset. Note that this technique of bootstrapping many new datasets and aggregating an ensemble of learners built from the individual sets is known as bagging [11].

The next concept in random forests that reduces the correlation between individual trees is that at each possible split of a tree, we use a random subset of the overall features available. This is known as a random subspace method or attribute bagging [35]. A typical subspace size of $\sqrt{L}$ where L is the total number of features is most often used. Note that due to the very large number of edges (which we will use as features) for high dimensional graphs, using all of them at each split in a decision tree leads to overfitting, a problem that is mitigated by using random forests.

Classification

We demonstrate how the ideas used in the original random forest algorithm can be extended for classification of graphs (where the existence of edges are

used as features). The key difference from the original algorithm is that our random subspace method does not consist of a uniform sampling from the set of potential edges but instead sampling uniformly L vertices and using the $L(L-1)/2$ edges from this subgraph as the features. This method is given in algorithm 14.

Algorithm 14 Random forest for graph classification

Require: Observation labels $y_1, \ldots, y_n$ and graphs $G_1, \ldots, G_n \in \mathbb{G}_p$, bootstrap size B, minimum leaf size $l_{\min}$, total trees T, subgraph dimension L, split rule

1: **for** $i = 1 : T$ **do**
2: $Z \leftarrow$ draw bootstrap sample from $(G_1, y_1), \ldots, (G_n, y_n)$ by sampling B times with replacement
3: $t_i \leftarrow$ learn decision tree from sample Z where at each leaf, if total members $> l_{\min}$, sample L vertices without replacement and split according to split rule using edges of the dimension L induced subgraphs as features
4: **end for**
5: Return $t_1, \ldots, t_T$

Remark 66. A common split rule for classification trees is the normalised information gain [66]).

For an unseen graph $G \in \mathbb{G}_p$ we can output the probability it is in a certain class y using the output decision trees as

$$\frac{1}{T}\sum_{i=1}^{T} I[t_i(G) = y]$$

i.e. the proportion of decision trees that classify G into class y.

Clustering

This idea can be extended to clustering graphs so we can analyse data without any doctor's diagnosis, or if we are not confident the diagnosis is correct. Shi

and Horvath [73] proposed a way to do this using random forests. The idea is to create synthetic data based on the original observed data. In our case, as we have a sample of graphs, we estimate the MLE binomial probability for each edge that would have produced the observed graphs and then simulate from this estimated distribution.

The original data is then labelled as one class and the synthetic data as another. After training a random forest on this aggregated data, we measure how close graphs in the original data were, based on the number of trees in which they were both classified into the same leaf node. This gives rise to a similarity matrix which can be transformed into a dissimilarity matrix representing the distance between graphs.

By using multidimensional scaling, the dissimilarity matrix can be transformed into a lower dimensional Euclidean space. Finally, we can use a point clustering technique such as k-means in order to extract our clusters from the original data set.

The modified version of the Shi and Horvath algorithm for graphical models is given in algorithm 15.

Algorithm 15 Random forest for graph clustering

Require: Observed graphs $G_1, \ldots, G_n \in \mathbb{G}_p$, bootstrap size B, minimum leaf size $l_{\min}$, total trees T, subgraph dimension L, split rule, method for generating synthetic data, total clusters k

1: Create n synthetic observations similar to $G_1, \ldots, G_n$ using synthetic data generation method
2: Train a random forest for graph classification using algorithm 14 on the synthetic data with labels 1, the observed data with labels 2 and parameters B, $l_{\min}$, T, L and split rule
3: $S \leftarrow$ proximity matrix such that S_{ij} is the proportion of decision trees that G_i and G_j ended in the same leaf node
4: $D \leftarrow$ dissimilarity matrix such that $D_{ij} = \sqrt{1 - S_{ij}}$
5: Use classical multidimensional scaling to convert graphs represented by D into points in Euclidean space
6: Run k-means to assign the points to k clusters
7: Return output clusters for graphs $G_1, \ldots, G_n$

Algorithm	class 1 correct (%)	class 2 correct (%)
(i)	92.5	92.5
(ii)	97.5	75
(iii)	100	92.5

Table 5.1: Results of the classification on $p = 5$

Remark 67. One way to generate synthetic data is to estimate the maximum likelihood random Bernoulli graph for the observations $G_1, \ldots, G_n$. Synthetic data can then be sampled from this distribution.

Results

Classification

In the following tests, we compare the classification algorithms of

i Logistic regression using edges as features

ii Logistic regression using centrality as features

iii Modified random forest

Given a dimension value p, we generate two matrices A_1 and A_2 according to the method in section 1.5.2. We then use these as the parameter matrices of a $\text{VAR}_p(1)$ process.

Next, we generate 50 length 512 time series from each VAR process to be used as training data and 40 length 512 time series to be used as testing data. The graphical model for each time series is extracted using the MHT technique in algorithm 4 with $\alpha = 0.05$ and $M = 32$.

Finally, we train the three classification models on the training data and report the performance of each on the test data. The results for $p = 5$ are given in table 5.1 and $p = 20$ in table 5.2. In the case of the modified random forest algorithm, we train 500 trees.

Algorithm	class 1 correct (%)	class 2 correct (%)
(i)	87.5	57.5
(ii)	57.5	60
(iii)	100	80

Table 5.2: Results of the classification on $p = 20$

We see that the modified random forest performs well on the models for both dimensions however both the logistic regression on edges and logistic regression on centrality do not perform well for $p = 20$.

Conclusion and Future Work

This chapter briefly covered classification and clustering of graphs with known node position. We introduced a modification to the well known random forest algorithm which outperformed classical logistic regression over a variety of classification experiments. It was also shown how the algorithm can be adapted for clustering but this requires further tests. We note that this is not the main focus of the thesis and the techniques used could be vastly improved.

For future consideration, we note the similarity between edge features and pixels of black and white image data (a large number of binary features). Variants of neural networks have proved very successful when working with the latter data e.g. on the MNIST dataset of handwriting. As such they may be another approach to look into if we have enough data to adequately train them.

6

Multiple Graph Matching and Graph Clustering

Introduction

In chapter 3 we looked at how to align two graphs over one another to make them as similar as possible. In this chapter we look at a related but less studied problem, that of finding the best way to align multiple graphs over one another so they are all as similar as possible. We call this multiple graph matching.

Multiple graph matching can be further extended into a clustering method where we look to split n graphs into k clusters such that within each cluster all the graphs align well over one another. It turns out that we can use pairwise graph matching between individual graphs and some aggregated object of the graphs within a cluster in a k-means type algorithm to perform this clustering.

This clustering approach then allows us in our brain diagnosis application, to potentially find common connectivity patterns within the brain which may

describe for example, certain terms of schizophrenia.

The chapter is organised as follows, section 6.1 contains some necessary preliminary results. Section 6.2 provides a general way for measuring how well multiple graphs fit over one another. This is then used in section 6.3 where we define the multiple graph matching problem. Section 6.4 investigates certain types of fit evaluation functions that arise from heuristic measures of graph similarity and section 6.5 looks at ways of normalising these functions to work in a clustering algorithm. We then present the clustering algorithm in section 6.6. We end the chapter with some concluding remarks and ideas for future work.

Preliminaries

Frobenius Norm Extension

We extend the definition of the Frobenius norm to a set of graphs.

Definition 44. *Given a set of graphs $\boldsymbol{G} = \{G_1, \ldots, G_m\}$, then we extend the Frobenius norm to be the average pairwise Frobenius norm of the individual adjacency matrices,*

$$||\cdot||_F : \mathbb{G}_n^m \to \mathbb{R}$$

as

$$||\boldsymbol{G}||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} ||A(G_i) - A(G_j)||_F^2}.$$

Sampling from Correlated Bernoulli Distributions

Some of our tests require us to be able to sample $Y_1, \ldots, Y_k \in \{0, 1\}$ such that for some $\theta \in [0, 1]$ and $\rho \in [0, 1]$,

$$Y_i \sim Bernoulli(\theta)$$

and

$$corr(Y_i, Y_j) = \rho$$

for $i \neq j$.

In order to do this we can use the technique in [25] which involves first sampling correlated multivariate Gaussians and then transforming these to Bernoullis.

Remark 68.

1. In the general case, the Y_i can have unique Bernoulli parameter and pairwise correlations i.e. $Y_i \sim Bernoulli(\theta_i)$ and $corr(Y_i, Y_j) = \rho_{ij}$.

2. In [25] the following bounds must be satisfied

$$\max(-(\frac{\theta_j\theta_k}{(1-\theta_j)(1-\theta_k)})^{1/2}, -(\frac{(1-\theta_j)(1-\theta_k)}{\theta_j\theta_k})^{1/2}) \leq \rho_{ij}$$

and

$$\min(-(\frac{\theta_j(1-\theta_k)}{(1-\theta_j)\theta_k})^{1/2}, -(\frac{(1-\theta_j)\theta_k}{\theta_j(1-\theta_k)})^{1/2}) \geq \rho_{ij}.$$

In our case we have $\theta_j = \theta_k$ so the bounds reduce to

$$0 \leq \rho \leq 1$$

and as we only consider non-negative correlations, these bounds are then irrelevant.

The algorithm can be written as in algorithm 16.

Algorithm 16 Sample correlated Bernoulli variables

Require: Bernoulli parameter $\theta \in [0,1]$, correlation $\rho \in [0,1]$, total number of samples n

Solve $\Phi(z(\theta), z(\theta), \delta) = \rho\theta^2 + \theta^2$ for δ
Sample an n-dimensional normal random variable $\boldsymbol{x} \sim MVN(\mathbf{0}, \delta I)$
for $i = 1 : n$ **do**
    if $x_i < z(\theta)$ **then**
        $Y_i \leftarrow 0$
    else
        $Y_i \leftarrow 1$
    end if
end for
return $Y_1, \ldots, Y_n$

where $\Phi(a, b, \delta)$ is a standard bivariate normal cdf with correlation coefficient δ i.e. $MVN(\mathbf{0}, \begin{pmatrix} 1 & \delta \\ \delta & 1 \end{pmatrix}$ and $z(a)$ is the ath quantile of a standard normal distribution.

Remark 69. Solving $\Phi(z(\theta), z(\theta), \delta) = \rho\theta^2 + \theta^2$ can be done using numerical techniques and only has to be done once in the algorithm. They suggest in [25] a simple bisection method.

Correlated Random Bernoulli Graphs

Correlated random Bernoulli graphs are a generalisation of random Bernoulli graphs we define them as in [58].

Definition 45. *Consider symmetric parameter matrix $Q \in [0,1]^{p \times p}$ with $p \in \mathbb{Z}$. Then for some correlation coefficient $\rho \in [0,1]$, the simple undirected graphs $G_1, \ldots, G_n$ are ρ correlated random Bernoulli graphs with parameter matrix Q if*

1. *$A(G_k)_{ij} = A(G_k)_{ji} \sim Bernoulli(Q_{ij})$ for $1 \leq i < j \leq p$ and $1 \leq k \leq n$.*
2. *$corr(A(G_k)_{ij}, A(G_l)_{ij}) = \rho$ for $i \neq j$ and $k \neq l$.*

3. *$A(G_k)_{ij}$ and $A(G_l)_{ij}$ are independent at all higher order moments for $k \neq l$.*

Remark 70. If ρ is 1, all realisations from a ρ correlated random Bernoulli graph are isomorphic to one another. If ρ is 0, the distribution is simply a random Bernoulli graph.

In [58] they used correlated random Bernoulli graphs to obtain their theoretical results for the FAQ graph matching algorithm. They found empirically that for $\rho < 0.6$ they were almost never successful at finding the optimal matching between two realisations from a ρ correlated random Bernoulli graph of dimension 150 with one of the realisations transformed by some permutation matrix. For $\rho > 0.75$ they were however almost always successful at recovering the permutation matrix, which is very likely to be the optimal matching. This is an interesting point that we will see is later relevant to our graph clustering algorithm.

In order to generate a pair of realisations from a ρ correlated random Bernoulli graph with parameter matrix Q by sampling the two adjacency matrices $A \in \mathbb{R}^{p\times p}$ and $B \in \mathbb{R}^{p\times p}$ as follows. For $i \leq 1 < j \leq p$ sample $B_{ij} \sim Bernoulli(Q_{ij})$ and then $A_{ij} \sim Bernoulli((1-\rho)Q_{ij} + \rho B_{ij})$. Finally set $B_{ji} = B_{ij}$ and $A_{ji} = A_{ij}$.

As we have to sample an arbitrary number of graphs, this formula based on the conditional distributions becomes increasingly complex. Instead we make use of the method from algorithm 16. The sampling of random Bernoulli graphs is then given in algorithm 17.

Algorithm 17 Sample simple undirected graphs with a ρ correlated random Bernoulli graph and return their adjacency matrices

Require: Parameter matrix $Q \in \mathbb{R}^{p \times p}$, correlation $\rho \in [0, 1]$, total number of samples n

Initialise adjacency matrices $A_1, \ldots, A_n = \mathbf{0} \in \mathbb{R}^{p \times p}$
for $i = 1 : p - 1$ **do**
    for $j = i + 1 : p$ **do**
        Sample $\boldsymbol{x} \in \mathbb{R}^n$ from a ρ correlated Bernoulli distribution with parameter Q_{ij} using algorithm 16
        $k \leftarrow 1$
        for $k = 1 : n$ **do**
            $(A_k)_{ij} \leftarrow x_k$
            $(A_k)_{ji} \leftarrow x_k$
        end for
    end for
end for
return $A_1, \ldots, A_n$

Clustering Evaluation

The main objective measure we use to analyse the performance of our clustering algorithm is the adjusted Rand index). We define the adjusted Rand index as in [67].

Consider a set of n objects $S = \{s_1, \ldots, s_n\}$ and two clusterings represented by the sets of sets of elements of S, $U = \{u_1, \ldots, u_r\}$ and $V = \{v_1, \ldots, v_q\}$ such that

$$\cup_{i=1}^{r} u_i = \cup_{i=1}^{q} v_i = S$$

and

$$u_i \cap u_j = v_i \cap v_j = \emptyset$$

for $i \neq j$.

For each pair of elements in S, (s_i, s_j) where $i \neq j$, let

$$f_U((s_i, s_j)) = \begin{cases} 1 & \text{if } (s_i, s_j)\text{are in the same cluster in } U \\ 0 & \text{otherwise} \end{cases},$$

$$f_V((s_i, s_j)) = \begin{cases} 1 & \text{if } (s_i, s_j)\text{are in the same cluster in } V \\ 0 & \text{otherwise} \end{cases}.$$

Now define

$$a = \sum_{i \neq j} f_U((s_i, s_j)) f_V((s_i, s_j))$$

total pairs of elements in the same cluster in U and the same cluster in V.

$$b = \sum_{i \neq j} f_U((s_i, s_j))(1 - f_V((s_i, s_j)))$$

total pairs of elements in the same cluster in U and different cluster in V.

$$c = \sum_{i \neq j} (1 - f_U((s_i, s_j))) f_V((s_i, s_j))$$

total pairs of elements in different cluster in U and the same cluster in V.

$$d = \sum_{i \neq j} (1 - f_U((s_i, s_j)))(1 - f_V((s_i, s_j)))$$

total pairs of elements in different cluster in U and different cluster in V.

Then from [38] we have the following definition.

Definition 46. *The adjusted Rand index is defined as*

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]}.$$

We can measure how well our clustering performs if we know the true clustering labels for each element by assigning U as our calculated clustering and

V as the true clustering and calculating the adjusted Rand index.

Generating Random Bernoulli Parameter Matrices

In order to generate parameter matrices for random Bernoulli graphs, we would like to be able to control the expected number of edges a realisation from the distribution has and the entropy of the distribution. Note that in layman's terms, for two random Bernoulli graphs with the same number of expected edges, the one with the lower entropy will have more values in the parameter matrix closer to 0 and 1 whereas the higher entropy distribution will have more values closer to 0.5. In order to generate these distributions for a given number of expected edges TEE and some value relating to the distribution entropy EC, we use the following heuristic method given in algorithm 18.

Algorithm 18 Generate a random Bernoulli graph parameter matrix

Require: Total expected edges TEE, entropy adjustment EC, dimension p

Initialise $U = \mathbf{0} \in \mathbb{R}^{p \times p}$
while $\sum_{ij} U_{ij} < 2TEE$ **do**
    Sample continuous uniform random variable x from $U[0, 1]$
    Sample discrete uniform random variable y_1 on the set $\{1, p-1\}$
    Sample discrete uniform random variable y_2 on the set $\{y_1 + 1, p\}$
    $U_{y_1 y_2} \leftarrow x$
    $U_{y_2 y_1} \leftarrow x$
end while
for $i = 1 : p$ **do**
    for $j = 1 : p$ **do**
        if $U_{ij} \leq 0.5$ **then**
            $U_{ij} \leftarrow 2^{EC-1} U_{ij}^{EC}$
        else
            $U_{ij} \leftarrow (1 - 2^{EC-1})(1 - U_{ij}^{EC})$
        end if
    end for
end for
return U

Note that the first step of the algorithm repeatedly adds uniform random variables to random entries in U until we are past the TEE limit. Then we push the values in U towards (away from) 0 and 1 when $EC > 1$ ($EC < 1$) by using an update based on the functions

$$f(x) = \frac{(2x)^n}{2}$$

for $x \leq 0.5$ and

$$g(x) = 1 - \frac{(2(1-x))^n}{2}$$

for $x > 0.5$.

Remark 71.
1. This algorithm is heuristic and doesn't ensure that the total expected edges of a realisation from our random Bernoulli graph is in fact equal to TEE but only that it is relatively close.
2. The first step of the algorithm is very inefficient if TEE is close to $p(p-1)/2$ so we should only use relatively small values for TEE.

Fit Evaluations

We look to extend the previous graph matching ideas to work with multiple graphs and refer to this as multiple graph matching. Given a set of graphs, we want to find some permutations applied to each graph so that they 'fit' together well. We begin by defining how well a set of unpermuted graphs fit together.

Let $\boldsymbol{G} = \{G_1, ..., G_m\}$ be a set of simple undirected graphs.

Definition 47. *For an edge e, define the edge count as*

$$m_e(1) = \sum_{i=1}^{m} I[e \in E(G_i)],$$

where $I[\cdot]$ is the indicator function and $E(G_i)$ is the set of edges for graph G_i.

We denote the number of non-edges as

$$m_e(0) = \sum_{i=1}^{m} I[e \notin E(G_i)] = m - m_e(1).$$

Definition 48. *The edge count function*

$$eC : \mathbb{G}_p^m \to [0, m]^{p(p-1)/2}$$

is defined as

$$eC(\boldsymbol{G}) = \begin{pmatrix} m_{e_1}(1) \\ \vdots \\ m_{e_{p(p-1)/2}}(1) \end{pmatrix},$$

where the e_j are the distinct edges from the set $ES(n)$. Note that as our graphs are simple and undirected, the edges in ES are all we need to capture all information from them.

Remark 72. The edge count function counts the number of graphs in $\boldsymbol{G}$ that contain a specific edge, for all edges.

Definition 49. *A fit evaluation function*

$$H_{l_m} : \mathbb{G}_p^m \to \mathbb{R}$$

is defined as

$$H_l = L_m \circ eC,$$

where eC is the edge count function and

$$L_m : [0, m]^{n(n-1)/2} \to \mathbb{R}$$

is defined as

$$L_m(\boldsymbol{x}) = \sum_{i=1}^{p(p-1)/2} l_m(x_i),$$

where $l_m : [0, m] \to \mathbb{R}$ is a symmetric convex function.

Remark 73. 1. A set of graphs with a high fit evaluation intuitively can

be placed on top of one another and have a similar edge structure.

2. As l_m is symmetric, having a lot of edges fit over one another is equally beneficial to having a lot of non-edges fit over one another.

3. As l_m is convex, it means if we have the option to permute one of the graphs in our set, then a permutation that causes its edges (and non-edges) to fall where most other graphs have edges (and non-edges) is preferred to a permutation causing its edges (and non-edges) to fall where just over half of the other graphs have edges (and non-edges).

4. As l_m is symmetric and convex on $[0, m]$, it has a minimum at $m/2$. Therefore edges that half the graphs contain and half don't is the worst possible contribution of that edge to the fit evaluation function. As the fit evaluation function is a sum over all edges, it may still be maximised with respect to some permutation of the graphs when some edges have very low contribution to it, implying this is outweighed by the contribution of other better fitting edges.

Definition 50. *We can naturally extend the concept of the edge count function and fit evaluation functions to act on the adjacency matrices of graphs (as opposed to the graphs themselves) and so also write the extension of eC and H_l as*

$$eC : (\mathbb{R}^{p\times p})^m \to [0, m]^{p(p-1)/2}$$

and

$$H_l : (\mathbb{R}^{p\times p})^m \to \mathbb{R}.$$

Multiple Graph Matching Problem

We previously considered solving the graph matching problem (GMP). We define the multiple graph matching problem (MGMP) for a given fit evaluation function H_{l_m} and graphs $G_1, \ldots, G_m$ as

$$\arg \max_{P_1,\ldots,P_m\in\mathbb{P}} H_{l_m}(\{P_1^T A(G_1)P_1, \ldots, P_m^T A(G_m)P_m\}). \tag{6.1}$$

Proposition 26. *The MGMP is equivalent to the GMP when we only have 2 simple undirected non-weighted graphs G_1 and G_2, for ANY fit evaluation function.*

Proof. If we only consider two graphs G_1 and G_2, we can write the MGMP as

$$\begin{aligned}
&\arg\max_{P_1,P_2\in\mathbb{P}} H_{l_2}(\{P_1^T A(G_1)P_1, P_2^T A(G_2)P_2\}) \\
&= \arg\max_{P_1,P_2\in\mathbb{P}} H_{l_2}(\{A(G_1), P_1P_2^T A(G_2)P_2P_1^T\}) \\
&= \arg\max_{P\in\mathbb{P}} H_{l_2}(\{A(G_1), P^T A(G_2)P\}).
\end{aligned}$$

Note also that for only two graphs, our fit evaluation function must be

$$l_2(x) = \begin{cases} a \text{ if } x = 0, 2 \\ b \text{ if } x = 1 \end{cases} \tag{6.2}$$

where $a, b \in \mathbb{R}$ and $a > b$ by symmetry and convexity of the l function.

Therefore

$$\begin{aligned}
&\arg\max_{P\in\mathbb{P}} H_{l_2}(\{A(G_1), P^T A(G_2)P\}) \\
&\qquad = \arg\max_{P\in\mathbb{P}} \sum_{i=1}^{p-1}\sum_{j=i+1}^{p} l_2(A(G_1)_{ij} + (P^T A(G_2)P)_{ij}).
\end{aligned} \tag{6.3}$$

Note also

$$l_2(y+z) = \begin{cases} a \text{ if } y = z \\ b \text{ if } y \neq z \end{cases} \tag{6.4}$$

where $y, z \in \{0, 1\}$.

Also, as our graphs are simple and undirected, the adjacency matrices have zero entries on the diagonal and are symmetric. Hence we can write (6.3) as

$$\arg\max_{P\in\mathbb{P}} \sum_{i=1}^{p}\sum_{j=1}^{p}(a-b)((A(G_1)_{ij} - (P^T A(G_2)P)_{ij})^2 + b$$

$$= \arg\max_{P \in \mathbb{P}} \sum_{i=1}^{p} \sum_{j=1}^{p} ((A(G_1)_{ij} - (P^T A(G_2) P)_{ij})^2$$

$$= \arg\max_{P \in \mathbb{P}} ||A(G_1) - P^T A(G_2) P||_F^2.$$

Note the max didn't change to a min as we previously assumed $a > b$. Therefore for any fit evaluation function, under the assumptions in the proposition, the MGMP is equivalent to the GMP. □

Types of Fit Evaluation Functions

While we have discussed fit evaluation functions and the properties they must satisfy, we can now show how they arise when applying heuristic methods to fit multiple graphs together.

Frobenius Norm

An intuitive method for measuring how well a set of graphs fit together is to calculate the pairwise Frobenius norm between all the graphs' adjacency matrices. Maximising this with respect to some chosen permutations can be shown to lead to an optimisation in the form of a MGMP.

Proposition 27. *Minimising the Frobenius norm of a set of graphs* $\boldsymbol{G} = \{G_1, \ldots, G_m\}$ *(as defined in (44)) is equivalent to maximising a fit evaluation function*

Proof. From our definition,

$$||\boldsymbol{G}||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{m} ||A(G_i) - A(G_j)||_F^2} \qquad (\star)$$

$$= \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{m} 2 \sum_{r=1}^{p(p-1)/2} (I[e_r \in E(G_i)] - I[e_r \in E(G_j)])^2}.$$

where e_r are the distinct edges from $ES(p)$. We used the fact our graphs are undirected and simple, so are symmetric and have zeros on the leading diagonal.

By considering the values the inner term can take, $(\star)$ can be written as

$$\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} 2 \sum_{r=1}^{p(p-1)/2} (I[e_r \in E(G_i)] \cdot I[e_r \notin E(G_j)] + I[e_r \notin E(G_i)] \cdot I[e_r \in E(G_j)])}$$

$$= \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} 4 \sum_{r=1}^{p(p-1)/2} I[e_r \in E(G_i)] \cdot I[e_r \notin E(G_j)]}$$

$$= \sqrt{4 \sum_{r=1}^{p(p-1)/2} (\sum_{i=1}^{m} I[e_r \in E(G_i)])(\sum_{j=1}^{m} I[e_r \notin E(G_j)])}.$$

Recalling the edge count $m_{e_r}(1) = \sum_{i=1}^{m} I[e_r \in E(G_i)]$ and $m_{e_r}(0) = \sum_{i=1}^{m} I[e_r \notin E(G_i)] = m - m_{e_r}(1)$.

Then we can re-write $(\star)$ as

$$\sqrt{4 \sum_{r=1}^{p(p-1)/2} m_{e_r}(1) m_{e_r}(0)}$$

$$= \sqrt{4m^2 \sum_{r=1}^{p(p-1)/2} \frac{m_{e_r}(1)}{m} \frac{m_{e_r}(0)}{m}}.$$

Now let $l_m(x) = -4x(m-x)$ and $L_m(\boldsymbol{x}) = \sum_{i=1}^{n(n-1)/2} l_m(x_i)$, then

$$H_{l_m} = L_m \circ eC(\boldsymbol{G}) = \sum_{r=1}^{p(p-1)/2} -4 m_{e_r}(1) m_{e_r}(0)$$

$$= -4m^2 \sum_{r=1}^{p(p-1)/2} \frac{m_{e_r}(1)}{m} \frac{m_{e_r}(0)}{m}$$

$$= -||\boldsymbol{G}||_F^2.$$

We also see that H_l is a fit evaluation function by (49) as l is symmetric and convex on $[0, m]$.

Finally, as $||\boldsymbol{G}||_F > 0$, if we introduce optimising over permutation matrices we conclude

$$\arg \max_{P_1,\ldots,P_m \in Pset} H_{l_m}(\{P_1^T G_1 P_1, \ldots, P_m^T G_m P_m\})$$

$$= \arg \max_{P_1,\ldots,P_m \in Pset} ||\{P_1^T G_1 P_1, \ldots, P_m^T G_m P_m\}||_F^2.$$

□

Entropy

We now investigate an entropy based method for performing the clustering. The idea behind this is that if we assume our observations come from an unknown distribution in a certain general class of distributions, then permutations of the observations that imply the distribution they came from has low entropy will then tend to make the transformed observations less random and in some sense 'fit' together better.

Proposition 28. *For a random Bernoulli graph $G = G^{RBG}(Q)$ the entropy of G satisfies*

$$\exp[-H(G)] = \prod_{1 \leq i,j \leq p} Q_{ij}^{Q_{ij}} \cdot (1 - Q_{ij})^{(1-Q_{ij})}.$$

Proof. As the probability of edges occurring in realisations from G are independent, we can write the entropy as

$$H(G) = \mathbb{E}[-\log(f(G))] = \sum_{1 \leq i,j \leq p} \mathbb{E}[-\log(f_{ij}(G))],$$

where $f(G)$ is the probability mass function for G and $f_{ij}(G)$ is the probability mass function for edge (i, j) of G.

Now the f_{ij} are independent Bernoulli trials, so we have

$$\mathbb{E}[-\log(f_{ij}(G))] = -Q_{ij}\log(Q_{ij}) - (1-Q_{ij})\log(1-Q_{ij})$$

and thus combining with the above equation

$$\exp[-H(G)] = \prod_{1\leq i,j\leq p} Q_{ij}^{Q_{ij}} \cdot (1-Q_{ij})^{(1-Q_{ij})}.$$

□

Definition 51. *Given observations $P_1G_1P_1^T, \ldots, P_mG_mP_m^T \in \mathbb{G}_n$ such that the G_i are realisations from $G^{DEP}(n,Q)$ and the P_i are realisations from a uniform distribution over $\mathbb{P}$, we define the combined likelihood of the observations as*

$$f(G_1, \ldots, G_m | Q, P_1, \ldots, P_m).$$

Proposition 29. *Consider observations $P_iG_iP_i^T$ as in definition (51). Then estimating $P_1, \ldots, P_m$ by minimising the entropy of the maximum likelihood estimate of Q where the assumed distribution is $G^{RBG}(Q)$, is equivalent to maximising the combined likelihood.*

Proof. Consider permutation matrix estimates $\hat{P}_1, \ldots, \hat{P}_m$, then as each edge from a $G^{RBG}(Q)$ distribution corresponds to a Bernoulli trial with probability Q_{ij}, the maximum likelihood estimate of Q can be written as

$$\hat{Q}_{ij} = m_{(i,j)}(1)/m$$

where

$$m_{(i,j)}(1) = \sum_{r=1}^{m} I[(i,j) \in E(\hat{P}_r^T P_r G_r P_r^T \hat{P}_r)]$$

is the number of transformed observations that contain edge (i,j).

Hence the entropy of the $G^{RBG}(\hat{Q})$ distribution with the maximum likelihood estimate of Q, can be written by proposition 28 as

$$\exp[-\hat{H}(G)] = \prod_{1\leq i,j\leq p} \hat{Q}_{ij}^{\hat{Q}_{ij}} \cdot (1-\hat{Q}_{ij})^{(1-\hat{Q}_{ij})}. \qquad (\star)$$

Now consider the combined likelihood

$$f(G_1, \ldots, G_m | \hat{Q}, \hat{P}_1, \ldots, \hat{P}_m)$$

$$= \prod_{r=1}^{m} f(G_r | \hat{Q}, \hat{P}_1, \ldots, \hat{P}_m).$$

Now,

$$f(G_r | \hat{Q}, \hat{P}_1, \ldots, \hat{P}_m) = \prod_{1 \leq i,j \leq p} \hat{Q}_{ij}^{I[(i,j) \in E(\hat{P}_r^T P_r G_r P_r^T \hat{P}_r)]} \cdot (1 - \hat{Q}_{ij})^{I[(i,j) \notin E(\hat{P}_r^T P_r G_r P_r^T \hat{P}_r)]}.$$

Therefore, by combining the above equations

$$f(G_1, \ldots, G_m | \hat{Q}, \hat{P}_1, \ldots, \hat{P}_m) = \prod_{1 \leq i,j \leq p} \hat{Q}_{ij}^{m_{(i,j)}(1)} \cdot (1 - \hat{Q}_{ij})^{m_{(i,j)}(0)}.$$

Note now that for $f(x) = x^a \cdot (1 - x)^{m-a}$, for arbitrary $0 \leq a \leq m$,

$$\frac{\partial \log f}{\partial x} = \frac{a}{x} - \frac{m - a}{1 - x}.$$

Finding the value of x that maximises f can be obtained by setting the above to 0. This gives

$$a(1 - x) = x(m - a)$$

$$\Rightarrow x = \frac{a}{m}.$$

Therefore $f(G_1, \ldots, G_m | \hat{Q}, \hat{P}_1, \ldots, \hat{P}_m)$ is maximised when $\hat{Q}_{ij} = \frac{m_{(i,j)}(1)}{m}$. Thus as the $m_{(i,j)}(1)$ are simply functions of our observations and estimated permutations, we do not have to consider the $\hat{Q}$ values when maximising the combined likelihood.

So,

$$f(G_1, \ldots, G_m | \hat{Q}, \hat{P}_1, \ldots, \hat{P}_m) = \prod_{1 \leq i,j \leq p} \left(\frac{m_{(i,j)}(1)}{m} \right)^{m_{(i,j)}(1)} \left(\frac{m_{(i,j)}(0)}{m} \right)^{m_{(i,j)}(0)}$$

and therefore from $(\star)$,

$$f(G_1, \ldots, G_m|\hat{Q}, \hat{P}_1, \ldots, \hat{P}_m)^{\frac{1}{m}} = \exp[-\hat{H}(G)]$$

as $\frac{m_{(i,j)}(1)}{m} = \hat{Q}_{ij}$.

Hence as $(\cdot)^{1/m}$and exp are increasing functions, minimising the estimated entropy is equivalent to maximising the combined likelihood. □

Remark 74.

The maximum likelihood graph obtained by maximising the fit evaluation function is not equivalent to the maximum likelihood graph obtained by maximising the likelihood based on just the observations themselves

$$f(G_1, \ldots, G_m|Q) \propto \sum_{\hat{P_1}, \ldots, \hat{P_m} \in \mathbb{P}} f(G_1, \ldots, G_m|Q, \hat{P}_1, \ldots, \hat{P_m}), \tag{6.5}$$

where the permutations are assumed uniformly distributed on $\mathbb{P}$. See [54] for an efficient way to solve this.

Further, the multiple graph matching does not intend to find a good/consistent solution of Q which is obtained when maximising (6.5). It is simply a way to connect the entropy of a Bernoulli random graph to fit evaluation functions.

Proposition 30. *Maximising the combined likelihood is equivalent to maximising a fit evaluation function.*

Proof. Consider the function

$$l_m(x) = x \log x/m + (1 - x) \log[(m - x)/m]$$

and

$$L_m(\boldsymbol{x}) = \sum_{r=1}^{p(p-1)/2} l_m(x_i).$$

Then $H_l = L_m \circ eC$ is a fit evaluation function as l is symmetric and convex

on $[0, m]$. Note also that

$$H_l(\{\hat{P}_1^T P_1 G_1 P_1^T \hat{P}_1, \ldots, \hat{P}_m^T P_m G_m P_m^T \hat{P}_m\}) = \sum_{e_r \in ES(p)} m_{e_r}(1) \log \frac{m_{e_r}(1)}{m} + m_{e_r}(0) \log \frac{m_{e_r}(0)}{m}$$

$$= m \sum_{e_r \in ES(p)} \frac{m_{e_r}(1)}{m} \log \frac{m_{e_r}(1)}{m} + \frac{m_{e_r}(0)}{m} \log \frac{m_{e_r}(0)}{m}$$

$$= m \sum_{e_r \in ES(p)} \hat{Q}_{e_r} \log \hat{Q}_{e_r} + (1 - \hat{Q}_{e_r}) \log[1 - \hat{Q}_{e_r}]$$

$$= -m\hat{H}(G),$$

where $G = G^{RBG}(\hat{Q})$ and

$$\hat{Q}_{ij} = m_{(i,j)}(1)/m.$$

Now, minimising the estimated entropy $\Leftrightarrow$ maximising the combined likelihood $\Leftrightarrow$ maximising a fit evaluation function. □

Fit Evaluation Norms

When we look to extend our multiple graph matching to clustering, we need the idea of fit evaluation norms. This is to ensure fit evaluation functions behave appropriately on different sized sets.

Definition 52. *We define the norm of l-function l_m as*

$$||l_m|| = \frac{l_m(x)}{l_1(x/m)}.$$

Note that for any fit evaluation function used for clustering, we need $l_m(x)/l_1(x/m)$ to be independent of x. This will be shown to be true for both the fit evaluation functions derived from the Frobenius norm and entropy.

Proposition 31. *Given two sets of graphs $\boldsymbol{G}_1, \boldsymbol{G}_2$ such that*

$$\frac{eC(\boldsymbol{G}_1)}{|\boldsymbol{G}_1|} = \frac{eC(\boldsymbol{G}_2)}{|\boldsymbol{G}_2|}$$

i.e. the 'proportion' of each edge is equivalent. Then,

$$\frac{H_{l_{m_1}}(\boldsymbol{G}_1)}{||l_{m_1}||} = \frac{H_{l_{m_2}}(\boldsymbol{G}_2)}{||l_{m_2}||}$$

where $m_1 = |\boldsymbol{G}_1|$ *and* $m_2 = |\boldsymbol{G}_2|$.

Proof. Let $m_1 = m$ and $m_2 = km$ for some constant k. Now,

$$H_{l_m}(\boldsymbol{G}_1) = L_m(eC(\boldsymbol{G}_1)) = L_m(m\boldsymbol{v})$$

where

$$\boldsymbol{v} = \frac{eC(\boldsymbol{G}_1)}{|\boldsymbol{G}_1|}.$$

Therefore the ith entry of the above value is as follows

$$l_m(mv_i) = ||l_m|| \cdot l_1(v_i)$$

from definition 52.

We also have

$$l_{km}(kmv_i) = ||l_{km}|| \cdot l_1(v_i)$$

from the same definition. Therefore

$$\begin{aligned}
\frac{l_m(mv_i)}{||l_m||} &= \frac{l_{km}(kmv_i)}{||l_{km}||} \\
\Rightarrow \frac{L_m(m\boldsymbol{v})}{||l_m||} &= \frac{L_{km}(km\boldsymbol{v})}{||l_{km}||} \\
\Rightarrow \frac{L_m(eC(\boldsymbol{G}_1))}{||l_m||} &= \frac{L_{km}(eC(\boldsymbol{G}_2))}{||l_{km}||} \\
\Rightarrow \frac{H_{l_{m_1}}(\boldsymbol{G}_1)}{||l_{m_1}||} &= \frac{H_{l_{m_2}}(\boldsymbol{G}_2)}{||l_{m_2}||}.
\end{aligned}$$

□

Example 5. Consider the fit evaluation function

$$l_m(x) = |\frac{m}{2} - x|^n$$

for $n > 1$. This is symmetric about $m/2$ and convex so is a valid fit evaluation function. Now,

$$||l_m|| = \frac{|m/2 - x|^n}{|1/2 - x/m|^n} = m^n.$$

Hence for 2 sets of graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ satisfying the properties in proposition 31,

$$\frac{H_{l_{m_1}}(\boldsymbol{G}_1)}{||l_{m_1}||} = \frac{H_{l_{m_2}}(\boldsymbol{G}_2)}{||l_{m_2}||}.$$

Therefore

$$\begin{aligned} H_{l_m}(\boldsymbol{G}_1) &= \frac{m^n}{(km)^n} H_{l_{km}}(\boldsymbol{G}_2) \\ &= k^{-n} H_{l_{km}}(\boldsymbol{G}_2), \end{aligned}$$

again letting $m_1 = m$ and $m_2 = km$.

We can now see why we must use norms when clustering. The clustering algorithm would be affected by the choice of n and the cluster size if we weren't to normalise. This can be seen be considering $k > 1$, then for large n, $H_{l_m}(\boldsymbol{G}_1)$ would be negligible compared to $H_{l_m}(\boldsymbol{G}_2)$. Instead we simply wanted to use n to control the shape of our fit evaluation l function but keep the cluster size influence on our clustering constant as seen in (6.6).

Current Norms

We showed 2 fit evaluation functions we could use, motivated by intuitive measures of how well a set of graphs fit together. These can be defined by their l function as,

1. Frobenius norm:

$$l_m(x) = -x(m-x)$$

(dropping multiplicative constant)

We can calculate the norm of this l-function using definition (52) to get

$$||l_m|| = \frac{-x(m-x)}{-x/m(1-x/m)} = m^2.$$

2. Entropy :

$$l_m(x) = x \log \frac{x}{m} + (m-x) \log \left[\frac{m-x}{m} \right].$$

We can calculate the norm of this l-function to get

$$||l_m|| = \frac{x \log \frac{x}{m} + (m-x) \log \left[\frac{m-x}{m}\right]}{\frac{x}{m} \log \frac{x}{m} + (1-\frac{x}{m}) \log \left[1-\frac{x}{m}\right]} = m.$$

Clustering Algorithm

We showed in the previous section how to build objective functions for evaluating how well a set of graphs fit together. This is now extended to measure how well clusters of graphs fit together. Maximising the objective function over permutations and clusters then gives us our multiple graph clustering algorithm.

For observations $G_1, \ldots, G_m \in \mathbb{G}_p$, a fit evaluation function H_l, k clusters and index sets $I_1, \ldots, I_k$ that form a partition of $\{1, \ldots, m\}$ we can write our objective to maximise as

$$\arg \max_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} |I_i| \frac{H_l(\{P_j^T G_j P_j | j \in I_i\})}{||l_{|I_i|}||}. \tag{6.6}$$

Remark 75.

The size of each cluster is accounted for only in the $|I_i|$ term, as the fit evaluation function is normalised. This allows us to intuitively think of the

normalised fit evaluation function as the average contribution (to the objective function) of each graph in a specific cluster. Multiplying by the total graphs in the cluster gives the total contribution of that cluster.

If we choose total clusters $k = 1$, we are simply left with a multiple graph matching optimisation.

We now show that for the Frobenius norm and entropy fit evaluation functions, (6.6) can be written in the form of a k-means objective function and can be solved using similar techniques. However we do not show this for general fit evaluation functions.

Frobenius Norm

Proposition 32. *Let $l_m(x) = -4x(m-x)$, the l-function derived from the pairwise Frobenius norm. Then for the associated fit evaluation function H_l, solving (6.6) is equivalent to solving*

$$\arg \min_{P_1,\dots,P_m,I_1,\dots,I_k} \sum_{i=1}^{k} \sum_{j\in I_i} ||P_j^T A(G_j) P_j - \mu_i||_F^2,$$

where $\mu_i = \frac{1}{|I_i|} \sum_{j\in I_i} P_j^T A(G_j) P_j$.

Proof. In the proof of proposition 27 we saw that $H_l(\boldsymbol{G}) = -||\boldsymbol{G}||_F^2$ for the given l-function and some $\boldsymbol{G} \in \mathbb{G}_n^m$. We also know that $||l_m|| = m^2$. Therefore, we can write (6.6) as

$$\arg \min_{P_1,\dots,P_m,I_1,\dots,I_k} \sum_{i=1}^{k} \frac{1}{|I_i|} \sum_{j_1\in I_i} \sum_{j_2\in I_i} ||P_{j_1}^T A(G_{j_1}) P_{j_1} - P_{j_2}^T A(G_{j_2}) P_{j_2}||_F^2$$

$$= \arg \min_{P_1,\dots,P_m,I_1,\dots,I_k} \sum_{i=1}^{k} \frac{1}{|I_i|} \sum_{j_1\in I_i} \sum_{j_2\in I_i} \operatorname{tr}(A(G_{j_1}))^2 + \operatorname{tr}(A(G_{j_2}))^2 - 2\operatorname{tr}[P_{j_1}^T A(G_{j_1}) P_{j_1} P_{j_2}^T A(G_{j_2}) P_{j_2}]$$

$$= \arg \min_{P_1,\dots,P_m,I_1,\dots,I_k} \left[2\sum_{i=1}^{m} \operatorname{tr}(A(G_i))^2 \right] - 2\sum_{i=1}^{k} \frac{1}{|I_i|} \sum_{j_1\in I_i} \sum_{j_2\in I_i} \operatorname{tr}[P_{j_1}^T A(G_{j_1}) P_{j_1} P_{j_2}^T A(G_{j_2}) P_{j_2}]$$

$$= \arg \max_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j_1 \in I_i} \operatorname{tr}[P_{j_1}^T A(G_{j_1}) P_{j_1} \frac{1}{|I_i|} \sum_{j_2 \in I_i} P_{j_2}^T A(G_{j_2}) P_{j_2}]$$

$$= \arg \max_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} \operatorname{tr}[P_j^T A(G_j) P_j \mu_i]. \qquad (\star)$$

Now consider

$$\arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} ||P_j^T A(G_j) P_j - \mu_i||_F^2$$

$$= \arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} \operatorname{tr}[(P_j^T A(G_j) P_j - \mu_i)^T (P_j^T A(G_j) P_j - \mu_i)]$$

$$= \arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} \operatorname{tr}[P_j^T A(G_j)^T A(G_j) P_j] - 2 \operatorname{tr}[P_j^T A(G_j) P_j \mu_i] + \operatorname{tr}[\mu_i^T \mu_i]$$

$$= \arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} -2 \operatorname{tr}[P_j^T A(G_j) P_j \mu_i] + \operatorname{tr}[\mu_i^T \mu_i] \qquad (\star\star)$$

as the first term of the sum does not depend on P_j. Now,

$$\operatorname{tr}[\mu_i^T \mu_i] = \frac{1}{|I_i|} \operatorname{tr}[\sum_{r \in I_i} P_r^T A(G_r)^T P_r \mu_i]$$

$$= \frac{1}{|I_i|} \operatorname{tr}[\sum_{r \in I_i} P_r^T A(G_r) P_r \mu_i] \qquad (\star\star\star)$$

as $A(G_r)$ is symmetric.

Combining $(\star\star)$ and $(\star\star\star)$ gives

$$\arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} ||P_j^T A(G_j) P_j - \mu_i||_F^2$$

$$= \arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} (-2 \operatorname{tr}[P_j^T A(G_j) P_j \mu_i] + \frac{1}{|I_i|} \operatorname{tr}[\sum_{r \in I_i} P_r^T A(G_r) P_r \mu_i])$$

$$= \arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} - \operatorname{tr}[P_j^T A(G_j) P_j \mu_i]$$

$$= \arg \max_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} \operatorname{tr}[P_j^T A(G_j) P_j \mu_i]$$

which is equivalent to $(\star)$. □

Hence, for a Frobenius norm fit evaluation function, the optimisation can be written

$$\arg \min_{I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} \min_{P_j} ||P_j^T A(G_j) P_j - \mu_i||_F^2. \tag{6.7}$$

This is in the form of a k-means optimisation with the distance between a graph and a centroid being

$$\arg \min_{P} ||P^T A(G_j) P - \mu_i||_F^2.$$

This distance is simply a graph matching problem and we can use techniques from chapter 3 to solve it.

Entropy

Proposition 33. *Let $l_m(x) = x \log \frac{x}{m} + (m-x) \log \frac{m-x}{m}$, the l-function derived from minimising the estimated entropy of a G^{RBG} distribution. Then for the associated fit evaluation function H_{l_m}, solving (6.6) is equivalent to solving*

$$\min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} -\mathcal{S}(P_j, G_j, \mu_i), \tag{6.8}$$

where

$$\mu_i = \frac{1}{|I_i|} \sum_{j \in I_i} P_j^T A(G_j) P_j$$

and

$$\mathcal{S}(P,G,\mu) = \sum_{r=1}^{p}\sum_{s=1}^{p}\left[(P^TA(G)P)_{rs}\log(\mu)_{rs} + (1-(P^TA(G)P)_{rs})\log(1-(\mu)_{rs})\right].$$

Proof. In the proof of proposition 30, we saw that

$$H_l(\{P_j^TG_jP_j|j\in I_i\})$$

$$= |I_i|\sum_{1\le r<s\le p}(\mu_i)_{rs}\log(\mu_i)_{rs} + (1-(\mu_i)_{rs})\log(1-(\mu_i)_{rs})$$

$$= |I_i|\sum_{1\le r<s\le p}\frac{1}{|I_i|}\sum_{j\in I_i}(P_j^TA(G_j)P_j)_{rs}\log(\mu_i)_{rs} + \frac{1}{|I_i|}\sum_{j\in I_i}(1-(P_j^TA(G_j)P_j)_{rs})\log(1-(\mu_i)_{rs})$$

$$= \sum_{1\le r<s\le p}\sum_{j\in I_i}(P_j^TA(G_j)P_j)_{rs}\log(\mu_i)_{rs} + \sum_{j\in I_i}(1-(P_j^TA(G_j)P_j)_{rs})\log(1-(\mu_i)_{rs})$$

$$= L_1 - L_2,$$

where

$$L_1 = \sum_{r=1}^{p}\sum_{s=1}^{p}[\frac{1}{2}\sum_{j\in I_i}(P_j^TA(G_j)P_j)_{rs}\log(\mu_i)_{rs} + (1-(P_j^TA(G_j)P_j)_{rs})\log(1-(\mu_i)_{rs})]$$

$$= \frac{1}{2}\sum_{j\in I_i}\mathcal{S}(P_j,G_j,\mu_i)$$

and

$$L_2 = \sum_{r=1}^{p}[\frac{1}{2}\sum_{j\in I_i}(P_j^TA(G_j)P_j)_{rr}\log(\mu_i)_{rr} + (1-(P_j^TA(G_j)P_j)_{rr})\log(1-(\mu_i)_{rr}].$$

As we are working with simple graphs (no nodes connected to themselves), L_2 in the above equation is 0. We also saw from before the norm

$$||l_{|I_i|}|| = |I_i|.$$

Hence

$$\arg \max_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} |I_i| \frac{H_l(\{P_j^T G_j P_j | j \in I_i\})}{||l_{|I_i|}||}$$

$$= \arg \min_{P_1,\ldots,P_m,I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} -\mathcal{S}(P_j, G_j, \mu_i).$$

□

As before, we can write the fit evaluation optimisation using the entropy fit evaluation functions as

$$\arg \min_{I_1,\ldots,I_k} \sum_{i=1}^{k} \sum_{j \in I_i} \min_{P_j} -\mathcal{S}(P_j, G_j, \mu_i) \tag{6.9}$$

which is once again a k-means optimisation with the distance between graphs and centroids

$$\min_{P \in \mathbb{P}} -\mathcal{S}(P, G_j, \mu_i).$$

Proposition 34. *We can write* $\min_{P \in \mathbb{P}} -\mathcal{S}(P, G_j, \mu_i)$ *in the form of a GMP*

$$\min_{P \in \mathbb{P}} -\mathcal{S}(P, G_j, \mu_i) = \min_{P \in \mathbb{P}} ||\text{logit}(\mu_i) - P^T A(G_j) P||_F^2.$$

Proof. We can see that

$$\min_{P \in \mathbb{P}} -\mathcal{S}(P, G_j, \mu_i)$$

$$= \min_{P \in \mathbb{P}} - \sum_{r=1}^{p} \sum_{s=1}^{p} (P^T A(G_j) P)_{rs} \log(\mu_i)_{rs} + (1 - (P^T A(G_j) P)_{rs}) \log(1 - (\mu_i)_{rs})$$

$$= \min_{P \in \mathbb{P}} - \sum_{r=1}^{p} \sum_{s=1}^{p} (P^T A(G_j) P)_{rs} (\log(\mu_i)_{rs} - \log(1 - (\mu_i)_{rs}))$$

$$= \min_{P \in \mathbb{P}} - \text{tr}(\text{logit}(\mu_i) P A(G_j)^T P^T)$$

where logit : $\mathbb{R}^{p \times p} \to \mathbb{R}^{p \times p}$ such that if

$$\text{logit}(\mu_i) = Z$$

then

$$Z_{rs} = \log\left(\frac{(\mu_i)_{rs}}{1-(\mu_i)_{rs}}\right).$$

Note the equivalence of the GMP and the QAP, hence

$$\begin{aligned}\min_{P\in\mathbb{P}} &-\mathcal{S}(P, G_j, \mu_i) \\ &= \min_{P\in\mathbb{P}} -\operatorname{tr}(\operatorname{logit}(\mu_i) P A(G_j)^T P^T) \\ &= \min_{P\in\mathbb{P}} ||\operatorname{logit}(\mu_i) - P^T A(G_j) P||_F^2. \quad (6.10)\end{aligned}$$

□

Algorithm

Solving (6.9) is tricky because of the graph matching formulation in (6.10). In particular the k-means intermediate centroids will often contain 0 or 1 values, in which case the logit transforms are $-\infty$ and ∞. This however means we can never assign a graph with an edge where the centroid has a 0 or a non-edge where the centroid has a 1 without the objective function 'blowing up'. Thus these intermediate 0 or 1 centroid values can never change. As k-means requires calculating non-optimal intermediate centroids, this property may be an issue. Hence we only use the Frobenius norm fit evaluation function in our clustering algorithm.

Remark 76.

Using a different clustering technique to k-means, without the non-optimal intermediate centroids, may allow us to use the entropy fit evaluation functions.

In standard graph matching between 2 graphs, it may be interesting to see how the entropy based distance compares to the commonly used Frobenius norm.

Definition 53. *Given a set of graphs* $\boldsymbol{G} = \{G_1, \ldots, G_m\} \in \mathbb{G}_p^m$*, we define*

the centroid of $\boldsymbol{G}$ *as*

$$\mu : \mathbb{G}_p^m \to \mathbb{R}^{p \times p}$$

such that

$$\mu(\boldsymbol{G}) = \frac{1}{m} \sum_{i=1}^{m} A(G_i).$$

Algorithm 19 Multiple graph clustering

Require: Observed graphs $\boldsymbol{G} = \{G_1, \ldots, G_m\} \in \mathbb{G}_p^m$ and total clusters k

Sample k graphs uniformly from $\boldsymbol{G}$ such that none are isomorphic
Initialise centroids $C_1, \ldots, C_k$ as the adjacency matrices of the sampled graphs
while True **do**
    Set centroid sets $\mathbb{K}_1, \ldots \mathbb{K}_k = \emptyset$
    $i \leftarrow 1$
    while $i \leq m$ **do**
        $P_i, k_i \leftarrow \arg\min_{P \in \mathbb{P}, j \in 1, \ldots, k} ||C_j - P^T A(G_i) P||_F^2$
        Add $P_i^T G_i P_i$ to $\mathbb{K}_{k_i}$
        $i \leftarrow i + 1$
    end while
    Recalculate centroids $C_j = \mu(\mathbb{K}_j)$ for $j = 1, \ldots, k$
    if Convergence criteria met **then**
        break
    end if
end while
Return permutation matrices $P_1, \ldots, P_m$ and clusters of graphs $\mathbb{K}_1, \ldots, \mathbb{K}_k$

Remark 77. In Euclidean space, k-means is guaranteed to find a local optima. This result requires extending to our problem as the unknown permutations introduce a new degree of complexity. If it holds (which is suggested by empirical results), then when we use an exhaustive approach to solve the GMP in algorithm 19, we simply converge when the objective function no longer improves between epochs. However if we are only using an approximate graph matching solver (e.g. FAQ), this is not necessarily the case. Here we recommend using a n-epoch non-improvement policy. This essentially means that if after n-epochs our current best objective function has not

improved, we break the loop and return the partitions and clusters associated with that best objective function.

Conclusion and Future Work

In this chapter we generalised graph matching to include more than two graphs. It was then shown how intuitive heuristic measures of graph similarity fall under the umbrella of our multiple graph matching framework. The framework was then extended to a clustering algorithm.

The work in this chapter is to the best of our knowledge novel. In future we would like to test the results on simulated data to both verify them and test the performance of the clustering algorithm.

7

Application to Neuroscience

Introduction

In this chapter we look to bring together the ideas from extracting source graphs and multiple graph matching to develop a clustering algorithm for time series data. It is not quite as simple as combining both approaches as we note the multiple graph matching doesn't use the information contained in the unmixing matrix that we use to label the nodes of our graphs

We show however that a small modification can take this into account in the FAQ graph matching which then naturally fits into the multiple graph matching framework. This creates an algorithm that takes into account both the structural information contained in the input conditional dependence graphs and the labelling of the nodes to create a more powerful clustering algorithm.

In section 7.1 we show how we modify the FAQ algorithm to work with labelled graphs and finish the chapter with some concluding remarks and ideas for future work.

Labelled Vertices

When performing ICA, it can be possible to use reverse methods to get a rough location of the source signals in the brain [33]. This can provide us with a priori information for a graph matching scheme. For example if the estimated location of two nodes for separate people are close together, we would like to increase the weight that our graph matching gives to mapping these nodes onto one another and vice versa. This can be achieved using labelled graph matching alluded to in [90]. Mathematically, our graph matching optimisation now has an additional term

$$\min_{P\in\mathbb{P}}(1-\alpha)||A-P^TBP||_F+\alpha\,\mathrm{tr}(C^TP) \tag{7.1}$$

where $A, B \in \mathbb{R}^{p\times p}$ are adjacency matrices, $\alpha \in [0,1]$ is a constant controlling the weight we give to the structural Frobenius norm term and the labelling term and $C \in \mathbb{R}^{p\times p}$ is a labelling distance such that

$$C_{ij} = d(G_A(i), G_B(j))$$

the distance based mapping node i of graph G_A to node j of graph G_B. In the ICA case, this distance could be for example the Euclidean distance between estimated source locations [33] or alternatively it could be calculated directly from the column of the unmixing matrix associated with a given node.

The labelling term in (7.1) is simply a linear term, so the FAQ approach of relaxing and using the Frank-Wolfe algorithm only needs a slight modification. Writing (7.1) in a quadratic programming form relaxed over the doubly stochastic matrices, we get the optimisation

$$\min_{Q\in\mathbb{D}} -2(1-\alpha)\,\mathrm{tr}(AQB^TQ^T)+\alpha\,\mathrm{tr}(C^TQ). \tag{7.2}$$

Algorithm 20 lFAQ for finding an approximate solution to the labelled GMP

Require: Adjacency matrices A, B, a labelling distance matrix C, a weighting constant α, a stopping criterion and an initialisation method (see below)

Ensure: $\hat{P}$ is a permutation matrix

Initialise $Q^{(0)}$ according to the initialisation method
$i \leftarrow 0$
while Stopping criteria not met **do**
    Compute gradient $\nabla f(Q^i) = -2(1-\alpha)(AQ^{(i)}B^T - A^TQ^{(i)}B) + \alpha C$
    Compute search direction $W^{(i)} = \arg\min_{Q \in \mathbb{D}} \operatorname{tr}(\nabla f(Q^i)Q)$ by the Hungarian algorithm
    Compute the step size $\alpha^{(i)} = \arg\min_{\alpha \in [0,1]} f(Q^{(i)} + \alpha W^{(i)})$ (exactly solvable as the objective is a quadratic function of α)
    $Q^{(i+1)} \leftarrow Q^{(i)} + \alpha^{(i)} W^{(i)}$
    $i \leftarrow i + 1$
end while
Compute $\hat{P} = \arg\min_{P \in \mathbb{P}} -\operatorname{tr}(Q^{(i)}P^T)$ by the Hungarian algorithm
return $\hat{P}$

This labelled graph matching algorithm naturally fits into the multiple graph matching framework in place of the original FAQ algorithm. This allows us to cluster on both graph structure and graph labels.

Remark 78. Note that as we are considering multiple different people with different shaped heads and brains just using the labels is not enough to perform our clustering. Similarly just using the graph dependency structure with no labels results in a huge number of possibilities to consider. However combining both can result in a powerful technique for clustering patients.

Future Work

In this chapter we showed how we could join the methods of [33] who simply used locations based on their unmixing matrices to align multiple graphs and the multiple graph matching framework which only used the conditional dependence structure. This allows us to develop a more powerful clustering

algorithm that does not waste information.

The next step would be to test the algorithm on real EEG data measured from multiple patients' brain scans and investigate the affect of the parameter α on the performance of the algorithm using computer tests with varying levels of added noise. Finally it would be interesting to see a comparison of the algorithm versus simply using the [33] and a comparison against simply using multiple graph matching to see how much added value there is in combining the approaches. Note that setting $\alpha = 0$ results in the algorithm being pure multiple graph matching and $\alpha = 1$ results in it just using labels as in [33].

8

Conclusion

Presented in this thesis were multiple frameworks for finding patterns within time series. In particular, we have two focuses. Firstly on observations that are not assumed a linear mixing of some latent time series. In this case we presented a novel graph extraction framework based on multiple hypothesis testing and analysed how common classification and clustering methods performed as well as a new method based on a modified random forest algorithm.

We also looked at the case where the observations were assumed to be a linear mixture of latent time series. It was shown how we could extract graphical models in this case and how we could cluster based on graphs with an unknown permutation of their nodes, using a novel multiple graph matching framework. Finally it was shown how this could be further extended by including meta information about nodes in a graph i.e. position.

The motivation behind the thesis was to develop techniques that would work with EEG recorded from human brains such that we could cluster groups of patients. The hope is that this can aid in the diagnosis of mental illness. We note however that the methods provided in almost every chapter are general to either time series or graphical models and so will hopefully have a wide

variety of other applications. The only chapter specific to the application was the final one.

There are many areas for future work. The multiple hypothesis testing framework was only considered using Matsuda's test statistic and the FWER error rate measure. It is definitely worth investigating alternate statistics and other potentially more appropriate error measures such as the FDR. The graph matching projection proved to be very promising, outperforming many existing algorithms on the QAPLIB benchmarks. It wasn't however able to outperform the FAQ algorithm. It was noted in [58] that the FAQ algorithm does not achieve optimal performance on certain graph matching problems concerning correlated Bernoulli graphs. We would like to investigate if in this case the projection is able to improve the FAQ.

The building graphs from source signals chapter requires a thresholding parameter to be specified. It would be much better if as in the multiple hypothesis testing chapter we were able to perform a test as to the existence of an edge in the recovered graphical model where the control was something more tangible such as the FDR. This requires a test statistic to be created based on the PMIR.

While much of the theory has been done for multiple graph matching, it requires more extensive testing of the key results and its clustering performance. Finally, the algorithm combining the source graph extraction, multiple graph matching and labelled graph matching needs to be tested on real EEG data.

Many of the algorithms presented require a large amount of computation. For time series with obvious differences in the conditional dependence structure (that aren't significantly deformed due to linear mixing of the sources), simple time series based clustering at the signal level would be far more efficient. Similarly the same could be said for graphs with clear differences in structure e.g. if separate clusters contained graphs with significantly different number of edges.

Another point to note is that while we explored graph matching in a suitable way for our application, for many other applications there may be a more

suitable distance metric between graphs other than the Frobenius norm between the adjacency matrices. One such distance metric that is commonly used is the graph edit distance.

A final point is that we stick to a very general domain for our inputs. Vast improvements in computation time can be achieved if we know our data is in a specific form. For example we may be able to assume there is a sparse conditional dependence structure for our time series a Kronecker representation for our graphical models. In order to make any sort of inference on very large dimensional models, often these further assumptions are needed.

A

Permission to use IEEE Copyrighted material

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Imperial College London's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to:

http://www.ieee.org/publications_standards/publications/rights/rights_link.html

to learn how to obtain a License from RightsLink.

References

[1] T. Adali, M. Anderson, G. Fu, "Diversity in Independent Component and Vector Analyses: Identifiability, algorithms, and applications in medical imaging", *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 18–33, 2014.

[2] H. A. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 15, pp. 522–525, 1993.

[3] F. R. Bach and M. I. Jordan, "Learning graphical models for stationary time series," *IEEE Trans. Signal Process.*, vol. 52, pp. 2189–99, 2004.

[4] A. Barvinok, "Approximating orthogonal matrices by permutation matrices," *Pure and Applied Mathematics Quarterly,* vol. 2, pp. 943–961, 2006.

[5] J. Beirlant, E. J. Dudewicz, L. Györfi and E. C. Van Der Meulen, "Nonparametric entropy estimation: An overview", *International Journal of Mathematical and Statistical Sciences*, vol. 6, pp. 17–39, 1997.

[6] A. J. Bell and T. J. Sejnowski, "An Information-maximization Approach to Blind Separation and Blind Deconvolution", *Neural Comput.*, vol. 7, no. 6, pp. 1129–1159, 1995.

[7] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal Stastistical Society Series B,* vol. 57, pp. 289–300, 1995.

[8] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *The Annals of Stastistics,* vol. 29, pp. 1165–1188, 2001.

[9] A. Bertrand and M. Moonen, "Seeing the bigger picture: How nodes can learn their place within a complex ad hoc network topology", *IEEE Signal Processing Magazine.*, vol. 30, iss. 3, pp. 71–82, 2013.

[10] L. Breiman, "Random Forests", *Machine Learning*, vol. 45, pp. 5–32, 2001.

[11] L. Breiman, "Bagging Predictors", *Machine Learning*, vol. 24, pp. 123–140, 1996.

[12] D. R. Brillinger, "Remarks concerning graphical models for time series and point processes," *Revista de Econometria (Brazilian Review of Econometrics),* vol. 16, pp. 1–23, 1996.

[13] H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," *Pattern Recognition Letters,* vol. 1, pp. 245–253, 1983.

[14] R. Burkard, S. Karisch and F. Rendl, "Qaplib – a quadratic assignment problem library." *J. Global Optimization,* vol. 10, pp. 391–403, 1997.

[15] R. Burkard, M. Dell'Amico and S. Martello, *Assignment Problems.*, Philadelphia, PA: SIAM, 2009.

[16] J. F. Cardoso and A. Souloumiac, "Blind beamforming for non Gaussian signals", *IEE Proceedings-F*, vol. 140, no. 6, pp. 362–370, 1993.

[17] J-F. Cardoso, "Multidimensional independent component analysis", *In Proc. Int. Workshop on Higher-Order Stat.*, pp. 111–120, 1998.

[18] P. Comon, "Independent component analysis: a new concept?", *Signal Process.,* vol. 36, pp. 287–314, 1994.

[19] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent Component Analysis and Applications.*, Academic Press, 2010.

[20] D. Conte, P. Foggia, C. Sansone and M. Vento,"Thirty years of graph matching in pattern recognition," *International J. Pattern Recognition and Artificial Intelligence,* vol. 18, pp. 265–298, 2004.

[21] T. Cover and J. Thomas, *Elements of Information Theory.* John Wiley and Sons, 1991.

[22] R. Dahlhaus, "Graphical interaction models for multivariate time series," *Metrika*, vol. 51, pp. 157-172, 2000.

[23] D. M. Edwards, *Introduction to Graphical Modelling, 2nd Ed.*, New York: Springer, 2000.

[24] M. Eichler, "Fitting graphical interaction models to multivariate time series," in *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence.*, Arlington, VA: AUAI Press, pp. 147–154, 2006.

[25] L. Emrich and M. Piedmonte, "A Method for Generating High-Dimensional Multivariate Binary Variates", *The American Statistician* vol. 45(4), pp. 302–304, 1991.

[26] M. Fiecas, H. Ombao, C. Linkletter, W. Thompson & J. Sanes, "Functional connectivity: shrinkage estimation and randomization test," *NeuroImage*, vol. 49, pp. 3005–3014, 2010.

[27] F. Fogel, R. Jenatton, F. Bach and A. d'Aspremont, "Convex relaxations for permutation problems," In *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Weinberger (Eds), pp. 1016–1024, Curran Associates, Inc., 2013

[28] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly,* vol. 3, pp. 95–110, 1956.

[29] R. Fried and V. Didelez, "Decomposability and selection of graphical models for time series," *Biometrika*, vol. 90, pp. 251–67, 2003. 251267.

[30] G. Fu, R. Phlypo, M. Anderson and T. Adali, "Complex Independent Component Analysis Using Three Types of Diversity: Non-Gaussianity, Nonwhiteness, and Noncircularity", *IEEE Transactions on Signal Processing*, vol. 63, pp. 794–805, 2015.

[31] U. Gather, M. Imhoff and R. Fried, "Graphical models for multivariate time series from intensive care monitoring," *Statist. Med.*, vol. 21, pp. 2685-701, 2002.

[32] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for graph matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, pp. 377–388, 1996.

[33] G. Gómez-Herrero, M. Atienza, K. Egiazarian and J. Cantero, "Measuring directional coupling between EEG sources", *NeuroImage*, vol. 43, pp. 497–508, 2008.

[34] W. Guo, "A note on adaptive Bonferroni and Holm procedures under dependence," *Biometrika*, vol. 96, pp. 1012–1018, 2009.

[35] T. K.Ho, "The Random Subspace Method for Constructing Decision Forests", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 832–844, 1998.

[36] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge UK: Cambridge University Press, 1985.

[37] D. W. Hosmer and S. Lemeshow, *Applied logistic regression*, John Wiley and Sons, 1989.
@

[38] L. Hubert and P. Arabie, "Comparing partitions", *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[39] A. Hyvärinen, "Independent component analysis for time-dependent stochastic processes", *Proc. Int. Conf. on Artificial Neural Networks (ICANN'98)*, Sweden, pp. 135–1402242, 1998.

[40] A. Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis", *IEEE Trans. Neural Netw.*, vol. 10, pp. 626–634, 1999.

[41] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications", *Neural Netw.*, vol. 13, pp. 411–430, 2000.

[42] A. Hyvärinen, J. Hurri and PO. Hoyer, *Natural Image Statistics.*, Berlin: Springer, 2009.

[43] A. Hyvärinen, K. Zhang, S. Shimizu and PO. Hoyer, "Estimation of a Structural Vector Autoregression Model Using Non-Gaussianity", *Journal of Machine Learning Research*, vol. 11, pp. 1709–1731, 2010.

[44] S. Ishii and M. Sato, "Doubly constrained network for combinatorial optimization", *Neurocomputing*, vol. 43, pp. 239–257, 2002.

[45] C. Jutten and J. Hérault, "Blind Seperation of sources, part I: an adaptive algorithm based on neuromimetic architecture," *Signal Process.,* vol. 24, pp. 1–10, 1991.

[46] D. Kazakos and P. Papantoni-Kazakos, "Spectral distance measures between Gaussian Processes" *IEEE Trans. on Automatic Control*, vol. 25, pp. 950-59.

[47] R. Kinderman and J. L. Snell, *Markov Random Fields and Their Applications*, American Mathematical Society: Contempory Mathematics, 1980.

[48] D. Knossow, A. Sharma, D. Mateus and R. Horaud, "Inexact matching of large and sparse graphs using Laplacian eigenvectors," In *Graph-Based Representations in Pattern Recognition*, A. Torsello, F. Escolano and L. Brun (Eds.). Volume 5534 of the series Lecture Notes in Computer Science, pp. 144–153, Springer, 2009.

[49] F. Kohl, G. Wübbeler, D. Kolossa, C. Elster, M. Bär and R. Orglmeister, "Non-independent BSS: A model for evoked MEG signals with controllable dependencies," *Independent Component Analysis and Signal Seperation, Lecture Notes in Computer Science,* vol. 5441, pp. 443–450, 2009.

[50] S. Lauritzen, *Graphical Models.*, Oxford Univ. Press, 1996.

[51] T. C. M. Lee, "A simple span selector for periodogram smoothing," *Biometrika*, vol. 84, pp. 965–969, 1997.

[52] T. C. M. Lee, "A stabilized bandwidth selection method for kernel smoothing of the periodogram," *Signal Processing*, vol. 81, pp. 419–430, 2001.

[53] E. L. Lehmann and J. P. Romano, *Testing Statistical Hypotheses, 3rd Ed.* New York: Springer, 2005.

[54] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos and Z. Ghahramani, "Kronecker Graphs: An Approach to Modeling Networks", *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, 2010.

[55] L. Livi and A. Rizzi, "The graph matching problem", *Pattern Analysis and Applications*, vol. 16, pp. 253–283, 2013.

[56] J. Long, *Regression Models for Categorical and Limited Dependent Variables.*, SAGE, 1997.

[57] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*, Berlin: Springer, 2006.

[58] V. Lyzinski, D. E. Fishkind, M. Fiori, J. T. Vogelstein, C. E. Priebe and G. Sapiro, "Graph Matching: Relax at Your Own Risk', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 60–73, 2016.

[59] Y. Matsuda, "A test statistic for graphical modelling of multivariate time series," *Biometrika*, vol. 93, pp. 399–409, 2006.

[60] Y. Matsuda, Y. Yajima & H. Tong, "Selecting models with different spectral density matrix structures by the cross-validated log likelihood criterion,"

[61] T. Medkour, A. T. Walden, A. P. Burgess & V. B. Strelets, "Brain connectivity in positive and negative syndrome schizophrenia," *Neuroscience*, vol. 169, pp. 1779–88.

[62] M. Newman, *Networks: An Introduction.*, London: Oxford Univ. Press, 2010.

[63] E. Onbaşoğlu and L. Özdamar, Linet, "Parallel Simulated Annealing Algorithms in Global Optimization", *Journal of Global Optimization*, vol. 19(1), pp. 27–50, 2001.

[64] D. B. Percival and A. T. Walden, *Spectral analysis for physical applications*, Cambridge UK: Cambridge University Press, 1993.

[65] M. B. Priestley, *Spectral Analysis and time series*, Academic Press, 1981.

[66] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., 1993.

[67] J. M. Santos and M. Embrechts, On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification, *Artificial Neural Networks – ICANN 2009: 19th International Conference*, Springer Berlin Heidelberg, pp. 175–184, 2009.

[68] C. Schellewald, S. Roth and C. Schnörr, "Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision," In *Pattern Recognition: 23rd DAGM Symposium Proceedings*, B. Radig and S. Florczyk (Eds.). Volume 2191 of the series Lecture Notes in Computer Science, pp. 361–368, Springer, 2001.

[69] C. Schellewald, S. Roth and C. Schnörr, "Evaluation of a convex relaxation to a quadratic assignment matching approach for relational object views," *Image and Vision Computing*, vol. 25, pp. 1301–1314, 2007.

[70] T. Schneider and A. Neumaier, "Algorithm 808: ARFITa matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models", *ACM T. Math. Soft.*, vol. 27(1), pp. 58-65, 2001.

[71] G. A. F. Seber, *Multivariate observations*, John Wiley and Sons, 1984.

[72] J. P. Shaffer, "Multiple hypothesis testing," *Annual Review of Psychology*, vol. 46, pp. 561–584.

[73] T. Shi and S. Horvath, "Unsupervised Learning with Random Forest Predictors", *JJournal of Computational and Graphical Statistics*, vol. 15, pp. 118–138, 2006.

[74] R. Sinkhorn, "A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices", *The Annals of Mathematical Statistics*, vol. 35(2), pp. 876–879, 1964.

[75] E. Palmero Soler, K. Dolan, V. Hadamschek, P. A. Tass, "swLORETA: a novel approach to robust source localization and synchronization tomography", *Phys. Med. Biol.*, vol. 52, pp. 1783-1800, 2007.

[76] J. Songsiri, J. Dahl & L. Vandenberghe, "Graphical models of autoregressive processes," in *Convex Optimization in Signal Processing and Communications,* D. P. Palomar and Y. C. Eldar, Eds. Cambridge UK: Cambridge University Press, 2010.

[77] J. Songsiri, J. Dahl & L. Vandenberghe, "Topology selection in graphical models of autoregressive processes," *Journal of Machine Learning Research*, vol. 11, pp. 2671–2705, 2010.

[78] T. P. Speed and H. Kiiveri, "Gaussian Markov distributions over finite graphs," *Annals of Statistics*, vol. 14, pp. 138–150, 1986.

[79] Z. Szabo, B. Poczos, and A. Lorincz, "Separation theorem for independent subspace analysis and its consequences," *Pattern Recognition,* vol. 45, pp. 1782–1791, 2012.

[80] J. Timmer, M. Lauk, S. Häußler, V. Radt, B. Köster, B. Hellwig, B. Guschlbauer, C.H. Lücking, M. Eichler and G. Deuschl, "Cross-spectral analysis of tremor time series," *Internat. J. Bifurcation and Chaos*, vol. 10, pp. 2595–2610, 2000.

[81] L. Tong, R. Liu, V. C. Soon and Y. F. Huang, "Indeterminacy and identifiability of blind identification", *Circuits and Systems, IEEE Transactions on*, vol. 38, no. 5, pp. 499–509, 1991.

[82] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 10, pp. 695–703, 1988.

[83] R. Vanderbei, *Linear Programming: Foundations and Extensions.*, Springer, 2007.

[84] N. Wermuth and E. Scheidt, "Fitting a covariance selection model to a matrix," *Applied Statistics*, vol. 26, pp. 88–92, 1977.

[85] R. Wolstenholme and A. Walden, "An efficient approach to graphical modelling of time series," *IEEE Transactions on Signal Processing,* vol. 63, pp. 3266–3276, 2015.

[86] R. Wolstenholme and A. Walden, "A sampling strategy for projecting to permutations in the graph matching problem," *arXiv,* eprint. 1604.04235, 2016.

[87] J. Wu, *Advances in K-means clustering*, Springer, 2012.

[88] J. T. Vogelstein, J. M. Conroy, V. Lyzinski, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein and C. E. Priebe, "Fast approximate quadratic programming for graph matching", *PLOS One*, vol. 10, no. e0121002, 2015.

[89] L. Xu and I. King, "A PCA approach for fast retrieval of structural patterns in attributed graphs", *IEEE Trans. Systems, Man and Cybernetics*, vol. 31, pp. 812–817, 2001.

[90] M. Zaslavskiy, F. Bach and J-P. Vert, "A path following algorithm for the graph matching problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, pp. 2227–2242, 2009.