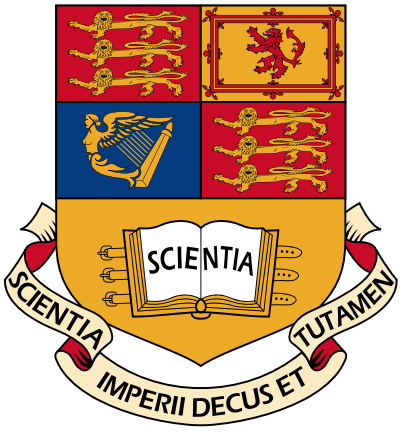
Bayesian Learning for Nonlinear System Identification



Wei Pan

Department of Bioengineering
Imperial College London

This thesis is submitted for the degree of
Doctor of Philosophy

March 2017

I would like to dedicate this thesis to my loving parents.

Declaration of Originality

I hereby declare that this thesis was entirely my own work and that any additional sources of information have been duly cited.

I hereby declare that any internet sources, published or unpublished works from which I have quoted or drawn reference have been reference fully in the text and in the contents list. I understand that failure to do this will result in failure of this project due to Plagiarism.

I understand I may be called for a viva and if so must attend. I acknowledge that was my responsibility to check whether I am required to attend and that I will be available during the viva period.

Wei Pan
March 2017

Copyright Declaration

Wei Pan
March 2017

Acknowledgements

First, I want to deeply thank my supervisor, Dr. Guy-Bart Stan, for his encouragement and guidance and support for my research all these years, as well as for providing a unique free and open environment that allows me pursue my own interest. I appreciate all his contributions of ideas, time, funding to make my Ph.D. study an open eye journey.

I would like to thank Dr. Ye Yuan, my long term collaborator. I can always find spark when we discuss in London, Cambridge, Luxembourg and Wuhan. Thanks, Ye, for everything. I have been so fortunate to collaborate and benefit from many great minds: Prof. Jorge Gonçalves, Prof. Mauricio Barahona, Dr. Aivar Sootla, Dr. Wei Dai, Prof. Lennart Ljung, Prof. Henrik Sandberg, Dr. Neil Dalchau, Dr. Andrew Philips and Prof. Yike Guo. Every tiny detail on meetings, discussions, presentations and paper writing suddenly came out of my mind. I apologise not to include all the aspects I learn from you simply because there are too many. Thank you all.

I am indebted to my undergraduate supervisor, Prof. Huijun Gao, for the invaluable advice and support he has been giving me all these years. I would also to thank Prof. Zidong Wang, my role model through years. I felt lucky to reunion with you in London.

Outside the lab, I like to thank all the friends I have met here in London, for their friendship and the many wonderful moments they have shared with me, and for their kind help and support during various stages of my stay here.

I also would like to thank two companies I've worked for during my Ph.D. study: Active Securities and Cardwell Investment Technologies, who showed me the possibilities of my work in financial industry.

Last but not least, I want to acknowledge and thank the support of my research. This research was generously supported by Microsoft Research, Dorothy-Hodgkin Postgraduate Award, Department of Bioengineering.

Abstract

Prediction and control of behaviour and abnormalities in any complex dynamical systems, and in particular those encountered in biology, physics, engineering require the development of multivariate mechanistic and predictive models that integrate large datasets from different sources. Although, a large amount of data are being collected on a daily basis, very few methods allow the automatic creation from these data of nonlinear dynamical models for understanding and (re-)design/control, and an inordinate amount of time is still being spent on the manual aggregation of information and development of models that explains these data.

In particular, this thesis considers sparse modelling and estimation for a selection of nonlinear dynamical systems classes. There are two key features of modern time series data, i.e., high dimensionality and large scale. The dimensionality, or the complexity, grew with the sample size, and "ultra-high" refers to the case where the dimensionality increased at a non-polynomial rate. Scale, or the size, refers to the dimension of the system, i.e., the number of state variables. This work aims to design a framework and associated algorithms for the identification of a variety of nonlinear dynamical systems encountered in practice from high-dimensional and large-scale time series data.

In the first part of the thesis, we introduce the type of time series data and the class of nonlinear dynamical system considered in this thesis. Both a selection of time-invariant and time-varying nonlinear dynamical systems are covered. For time-invariant system, the classic nonlinear system identification problem from single dataset is addressed in the beginning. Then we move to a more practical and significant yet complicated scenario where heterogeneous datasets are used simultaneously. Such datasets typically contain (a) data from several replicates of an experiment performed on a biological system of interest and/or (b) data measured from a biochemical system subjected to different experimental conditions, for example, changes/perturbations in biological inductions, temperature, gene knock-out, gene over-expression, etc. For time-varying systems, the regime-switch system identification problem is considered, i.e., the problem of identifying both

the switching points and the nonlinear model structure within each regime. Then the abrupt change point detection problem is considered. Using these, the classic trending filtering and fault diagnosis problems are revisited. All the identification problems are formulated as various ℓ_0 type optimisation problems. In the end, we discuss some technical issues on data processing arising from practical applications.

In the second part of the thesis, a repository of algorithms are derived respectively for each identification problem formulated in the first part. These algorithms are not distinct and can be formulated in a unified way using Bayesian Learning with structural sparse prior. Furthermore, we suggest a series of iterative reweighted convex relaxation schemes for connecting these algorithms to popular algorithms including Lasso, Group-Lasso, Generalised-Lasso, Fused-Lasso and Graphical-Lasso. In this part, we go beyond from simple nonlinear model class to more general class; from data likelihood in Gaussian distribution to the more general exponential family. The estimation of the stochastic term also discussed including ARMA and ARCH. Many optimisation framework, such as (stochastic) gradient descent, Newton method, Quasi-Newton method, alternating direction method of multiplier can be seamlessly integrated into our formulation as either centralised or distributed optimisation strategy to address high dimensionality and large scale problems. These algorithms largely enrich not only the family of time series modelling algorithms but also sparse signal recovery/modelling/estimation algorithms in various communities.

In the third part of the thesis, several time series modelling applications from systems biology, complex networks and power systems are given to illustrate the effectiveness of our modelling framework.

Last but not least, two future research directions based on the output of this thesis are pointed out, both related to "brains". The first is focusing on theory and algorithm about modelling/identification/learning on deep neural networks. The second is focusing applications in neuroscience: understanding the neural basis of decision making using mathematical modelling from big data.

Abbreviations and Symbols

Roman Symbols

ADMM **A**lternative **D**irection **M**ethod of **M**ultiplier

AIC **A**kaike **I**nformation **C**riterion

ARCH **A**utoreg**r**essive **C**onditional **H**eteroskedasticity

ARMA **A**uto**r**egressive **M**oving **A**verage

ARMAX **A**uto**r**egressive **M**oving **A**verage with E**x**ternal Input

ARX **A**uto**r**egressive with E**x**ternal Input

BIC **B**ayesian **I**nformation **C**riterion

CCCP **C**onvex-**C**on**c**ave **P**rocedure or **C**on**c**ave-**C**onvex **P**rocedure

DC **D**ifference of **C**onvex Functions

DNN **D**eep **N**eural **N**etworks

DL **D**eep **L**earning

GRN **G**enetic of **R**egulatory textbfNetworks

KKT **K**arush-**K**uhn-**T**ucker

LS **L**east **S**quare

MAP **M**aximum **a** **P**osteriori

MIMO **M**ultiple **I**nput **M**ultiple **O**utput

ML **M**aximum **L**ikelihood

MLE **M**aximum **L**ikelihood **E**stimate

MM **M**ajorisation-**M**inimisation

NARX **N**onlinear **A**uto**r**egressive with E**x**ternal Input

NN **N**eural **N**etworks

NP **N**on-deterministic **P**olynomial-time

PLS **P**enalised **L**east **S**quare

RIP **R**estricted **I**sometry **P**roperty

SBL **S**parse **B**ayesian **L**earning

SISO **S**ingle **I**nput **S**ingle **O**utput

SYSID **Sys**tem **Id**entfication

w.r.t. **w**ith **r**epect **t**o

Subscripts

$\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ a matrix in $\mathbb{R}^{M \times N}$

$\boldsymbol{\Phi}_{i,j} \in \mathbb{R}$ element in the i^{th} row and j^{th} column of a matrix

$\boldsymbol{\Phi}_{i,:}$ the i^{th} row of a matrix

$\boldsymbol{\Phi}_{:,j}$ the j^{th} column of a matrix

$\boldsymbol{\alpha} \in \mathbb{R}^{N \times 1}$ a column vector in $\mathbb{R}^N$

α_i the i^{th} element of a vector

$\boldsymbol{I}_L$ a identity matrix of size $L \times L$, we simply use $\boldsymbol{I}$ when the dimension is obvious from context

$\|\boldsymbol{\beta}\|_p, \|\boldsymbol{\beta}\|_{\ell_p}$ ℓ_p norm of a vector $\boldsymbol{\beta} \in \mathbb{R}^N$, that is $\|\boldsymbol{\beta}\|_{\ell_p} \triangleq \sqrt[p]{\sum_{i=1}^{N} \boldsymbol{\beta}_i^p}$

$\|\boldsymbol{\beta}\|_0, \|\boldsymbol{\beta}\|_{\ell_0}$ ℓ_0-quasinorm is the number of non-zero elements in a vector

$\text{diag}[\gamma_1, \ldots, \gamma_N]$ a diagonal matrix with principal diagonal elements being $\gamma_1, \ldots, \gamma_N$

$\mathbb{E}(\boldsymbol{\alpha})$ the expectation of the stochastic variable $\boldsymbol{\alpha}$

$\propto$ proportional to

$\triangleq$ defined as

blkdiag$[\boldsymbol{\Phi}^{[1]}, \ldots, \boldsymbol{\Phi}^{[C]}]$ a block diagonal matrix with principal diagonal blocks bei
$\boldsymbol{\Phi}^{[1]}, \ldots, \boldsymbol{\Phi}^{[C]}$ in turn

Tr$(\boldsymbol{\Phi})$ the trace of a matrix$\boldsymbol{\Phi}$

$\boldsymbol{\Phi} \succeq \mathbf{0}$ the matrix $\boldsymbol{\Phi}$ is positive semidefinite

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 System Identification

During the thesis author's visit to Professor Lennart Ljung's group at Linköping University, who is the authority in the field of *system identification* (SYSID), a series of ideas were shared on SYSID. I would like to quote and summarise these ideas in what follows.

1.1.1 The Omni-present Model

It is clear to everyone in science and engineering that mathematical models are playing increasingly important roles. Today, model-based design and optimization is the dominant engineering paradigm to systematic design and maintenance of engineering systems. It has proven very successful and is widely used in basically all engineering disciplines. Concerning control applications, the aerospace industry is the earliest example on a grand-scale of this paradigm. This industry was very quick to adopt the theory for model based optimal control that emerged in the 1960s, and is spending great efforts and resources on developing models. In the process industry, Model Predictive Control (MPC) has during the last 25 years become the dominant method to optimize production on an intermediate level. MPC uses dynamical models to predict future process behaviour and to optimize the manipulated variables subject to process constraints.

Increasing demands on performance, efficiency, safety and environmental aspects are pushing engineering systems to become increasingly complex. Advances in (wireless) communications systems and micro-electronics are key enablers for this rapid development; allowing systems to be efficiently inter-connected in networks, reducing costs and size and paving the way for new sensors and actuators.

Model-based techniques are also gaining importance outside engineering applications. Let us just mention systems biology and health care. In the latter case it is expected that personalised health systems will become more and more important in the future.

Common to the examples given above are the requirements of integrating sensing actuation, communication and computation abilities of the engineering systems, in many cases in distributed architectures. It is also clear that these systems should be able to operate in a reliable way in an uncertain and temporally and spatially changing environment. In many applications, cognitive abilities and abilities to adapt will be important. With systems being decentralized and typically containing many actuators, sensors, states and non-linearities, but with limited access to sensor

information, model building that delivers models of sufficient fidelity becomes very challenging.

1.1.2 System Identification: Data Driven Modelling

Construction of models requires access to observed data. It could be that the model is developed entirely from information in signals from the system ("black box models"), or it could be that physical/engineering insights are combined with such information ("grey box models"). In any case, verification (validation) of a model must be done in the light of measured data. Theories and methodologies for such model construction have been developed in many different research communities (to some extent independently). *System Identification* is the term used in the Control Community for the area of constructing mathematical models of dynamical systems from measured input/output signals. Other communities use other terms for often very similar techniques. The term *Machine Learning* has become very common in recent years.

System identification has a history of more than 50 years since the term was coined by Lotfi Zadeh, [227]. It is a mature research field with numerous publications, text books, conference series, and software packages. It is often used as an example in the control field of an area with good interaction between theory and industrial practice. The backbone of the theory relies upon statistical grounds, with maximum likelihood and asymptotic analysis (in the number of observed data). The goal of the SYSID field is to find a model of the plant in question as well as of its disturbance and also to find a characterisation of the uncertainty bound of the description.

1.1.3 The State-of-the-Art Identification Setup

To approach a SYSID problem, a number of questions need to be answered, like

- what model type should be used?
- how should the parameters in the model be adjusted?
- what inputs should be applied to obtain a good model?
- how do we assess the quality of the model?
- how do we gain confidence in an estimated model?

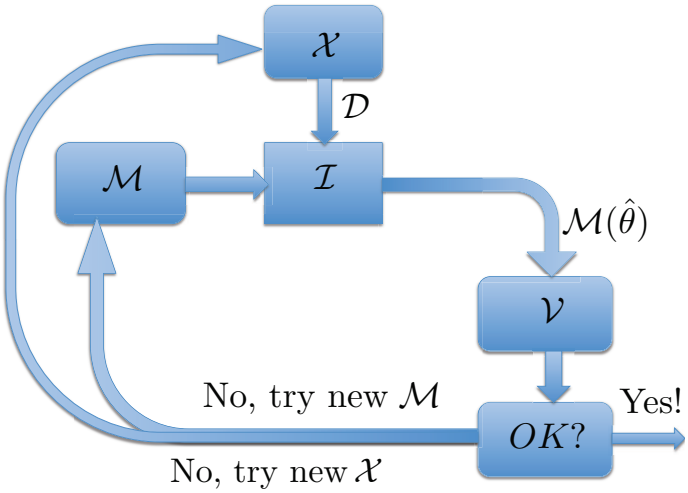


Fig. 1.1 The identification work loop (Coutersey of Professor Lennart Ljung).

There is a very extensive literature on the subject, with many text books [117, 175, 227]. The SYSID problem is usually formulated mathematically by introducing the following notations

- $\mathcal{X}$: The experimental conditions under which the data is generated
- $\mathcal{D}$: The data
- $\mathcal{M}$: The Model Structure and its parameters β
- $\mathcal{I}$: The identification method by which a parameter value $\hat{\beta}$ in the model structure $\mathcal{M}(\beta)$ is determined based on the data $\mathcal{D}$
- $\mathcal{V}$: The validation process that scrutinises the identified model.

The identification of a model is typically an iterative process that involves passing through the non-invalidation test step ("the model is not falsified (so far)"), and also involving various revisions of the modelling choices. that passes through the validation test ("is not falsified"), involving revisions of the necessary choices. For several of the steps in this loop helpful support tools have been developed. It is not quite possible or desirable to fully automate the choices, since subjective perspectives, related to the intended use of the model are very important.

1.2 Convex Optimisation

1.2.1 Convex Relaxation

Convex optimisation is a special class of mathematical optimisation problems for which there are efficient polynomial time algorithms that are guaranteed to converge to a global minimum [28]. Least square, linear programming, semidefinite programming, second order cone programming are all convex optimisation problems. If a problem can be formulated as a convex problem, it can be reliably and efficiently solved, for instance, using interior point methods. There are free optimisation packages (i.e., solvers) such as `SDPT3` [193] and `SeDuMi` [183] for solving convex programs. Moreover, user-friendly modelling languages such as `CVX` and `YALMIP` [119] make these solvers accessible for a wide range of users.

On the contrary to convex problems, non-convex problems usually have lots of local minima where optimisation algorithms are likely to get trapped if not initialised properly. Convex relaxation can be used to approximate (in some cases, even exactly solve for) the global optimum of a non-convex problem in a principled way. In some cases, they can also be used to find bounds on the global optimum or to provide a good point for gradient search.

In this thesis, all the SYSID problems are recast into appropriate optimisation problems. Unfortunately, these problems are usually non-convex and NP-hard. Nevertheless, resorting to convex relaxations, we obtain efficient solutions. In what follows some important convex relaxations, which play a central role in deriving our main results, are summarised. It is important to note that this Section is by no means a comprehensive treatment of the subject. Rather, it aims at giving a basic intuition behind each relaxation together with readily usable, relaxed, convex formulations that will help the reader understand the development in the proceeding Chapters.

1.2.2 Convex Concave Procedure

The *Convex Concave Procedure* (CCCP) [226] is a majorisation-minimisation (MM) algorithm [92] that is popularly used to solve DC (difference of convex functions). Its main idea is to yield an iterative scheme for

$$\min_{x \in C} f(x)$$

with $C \subseteq \mathbb{R}^p$ where each iteration consists of minimising a so-called majorisation function $\bar{f}(x, x^{(k)})$ of $f(x)$ at $x^{(k)} \in C$

$$x^{(k+1)} = \underset{x \in C}{\operatorname{argmin}} \, \bar{f}\left(x, x^{(k)}\right) \tag{1.1}$$

where $\bar{f} : C \times C \to \mathbb{R}$ satisfies $\bar{f}(x, x) = f(x)$ for $x \in C$ and $f(x) \leq \bar{f}(x, z)$ for $x, z \in C$. Clearly, (1.1) yields a descent algorithm. Construction of a suitable majorisation function is a key step for MM algorithm. For difference of convex programming problems $\min_{x \in C} f(x)$ where

$$f(x) = u(x) - v(x),$$

$u, v : C \to \mathbb{R}$ are convex and differentiable functions with C being a convex set in $\mathbb{R}^p$, there are many ways to construct the majorisation function [92]. The simplest one is the so-called linear majorisation via "supporting hyperplane" [92], i.e.,

$$\bar{f}\left(x, x^{(k)}\right) = u(x) - v\left(x^{(k)}\right) - \nabla v\left(x^{(k)}\right)^{\top}\left(x - x^{(k)}\right). \tag{1.2}$$

For this particular choice of majorisation function, it is also referred to as "sequential convex optimisation".

The convergence of MM algorithm to a stationary point (the point satisfies the Karush-Kuhn-Tucker (KKT) conditions, see e.g., [28]) has been discussed in e.g., [179].

1.3 Sparse Signal Recovery

In this Section, we present the background results on the problem of sparse signal recovery [31, 196, 50] that motivates the approach pursued in this thesis.

The *Sparse signal recovery* problem can be stated as: given some linear measurements $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$ of a discrete signal $\boldsymbol{\beta} \in \mathbb{R}^N$ where $\mathbf{X} \in \mathbb{R}^{M \times N}$, $M \ll N$, find the sparsest signal $\boldsymbol{\beta}^*$ consistent with the measurements. In terms of the ℓ_0 quasi-norm (i.e. $\|\cdot\|_{\ell_0}$ or $\|\cdot\|_0$ satisfies all of the norm axioms except homogeneity since $\|c\mathbf{x}\|_0 = \|\boldsymbol{\beta}\|_0$ for all non-zero scalars c), this problem can be recast into the following optimisation form:

$$\min \|\boldsymbol{\beta}\|_{\ell_0} \quad \text{subject to:} \quad \mathbf{y} = \mathbf{X}\boldsymbol{\beta}. \tag{1.3}$$

When the measurements are contaminated with some noise, the optimisation problem can be formulated as a regularised linear regression form:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_0} . \tag{1.4}$$

where λ is a tradeoff parameter or the regularisation parameter. It is well known that the optimisation problems above are at least generically non-deterministic polynomial-time (NP)-complete. Two fundamental questions in sparse signal recovery are: (i) the uniqueness of the sparse solution, (ii) the existence of efficient algorithms for finding such a solution. In the past few years it has been shown that if the matrix A satisfies the so-called *restricted isometry property* (RIP), the solution is unique and can be recovered efficiently by several algorithms. These algorithms fall into two main categories: greedy algorithms (e.g. orthogonal matching pursuit [195, 197, 129, 44]) and ℓ_1-based convex relaxation (also known as basis pursuit [31, 196, 50]).

As shown in [31], the RIP condition is a sufficient condition for exact reconstruction based on ℓ_1-minimisation. It was shown in [31, 44, 30] that both convex ℓ_1-minimisations and greedy algorithms lead to exact reconstruction of S-sparse signals if the matrix $\mathbf{X}$ satisfies the RIP condition. The ℓ_1 relaxation of the optimisation problem in (1.4) is

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_1} \tag{1.5}$$

The idea behind this relaxation is the fact that the ℓ_1 norm is the *convex envelope* or *tightest convex relaxation* of the ℓ_0 norm, and thus, in a sense, minimizing the former yields the best convex relaxation to the (non-convex) problem of minimising the latter. Moreover, as shown in [31, 196, 50], this relaxation is stable and robust to noise. That is, even when only noisy linear measurements are available, if RIP holds for $\mathbf{X}$, which is true with high probability for random matrices, recovery of the correct support of the original signal and approximating the true value within a factor of the noise is always possible. This formulation arises naturally in many engineering applications such as magnetic resonance imaging, radar signal processing and image processing. Moreover, existence of efficient algorithms to solve this problem led to the *compressive sensing* framework which enabled speeding up signal acquisition considerably since the original sparse signal can be reconstructed using relatively few measurements.

One major drawback of the RIP condition is that it can be very difficult to check (combinatorial search). Another related and easier-to-check property is the coherence property.

Definition 1 (mutual coherence [52]) *For any matrix* $\mathbf{X} = [\mathbf{X}_{:,1}, \ldots, \mathbf{X}_{:,N}] \in \mathbb{R}^{M \times N}$, *the* coherence *of a matrix* $\mathbf{X}$ *is defined as*

$$\mu(\mathbf{X}) = \max_{1 \leq j,k \leq N, j \neq k} \frac{|\langle \mathbf{X}_{:,j}, \mathbf{X}_{:,k} \rangle|}{\|\mathbf{X}_{:,j}\|_2 \|\mathbf{X}_{:,k}\|_2}. \tag{1.6}$$

As shown in [51], for a column normalised matrix $\mathbf{X}$, i.e., $\|\mathbf{X}_{:,i}\|_2 = 1$, ℓ_1-minimisation solutions are equivalent to ℓ_0-minimisation solutions if, for a solution, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$, the following condition is satisfied:

$$\|\boldsymbol{\beta}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{X})}\right) \tag{1.7}$$

It was also shown that RIP guarantees *incoherence* of $\mathbf{X}$, i.e. $\mu(\mathbf{X}) \approx 0$, [31]. This means one is guaranteed that ℓ_1-minimisation solutions are equivalent to the true solution only when $\mathbf{X}$ is near orthogonal, i.e. when the columns of $\mathbf{X}$ are strongly uncorrelated.

Revisit the background of SYSID problems in Section 1.1 for a moment. As we will show in Part II of the thesis, SYSID problems can be formulated as ℓ_0 norm regularised optimisation problems such as (1.4) and its variations. The dictionary matrix $\mathbf{X}$ is constructed directly from time series data or certain transformation of the data. Quite often, correlation between the columns of $\boldsymbol{\Phi}$ is typically high (close to 1) and the RIP condition and low coherence condition are hardly guaranteed. In practice, ℓ_1 relaxation based algorithms do not yield the optimal solution.

1.4 Machine Learning

The *Machine Learning* community is huge. The success of machine learning applications has boosted and been boosting the development of our modern societies. An exhaustive literature review on machine learning is not necessary in this thesis. Several excellent textbooks will give an overview, technique explanations and applications of machine learning, e.g., [22, 79, 127], to just name a few.

In this Section we would like to offer an insight into the links that exist between machine learning and the system identification problem that we introduced earlier.

We do this as inference tools from machine learning since they are superior in certain aspects to the conventional approaches in SYSID. Until very recently, there has been little contact between these concepts and system identification.

Recent research has shown that model selection problems can be successfully dealt with using a different approach to SYSID that leads to an interesting cross fertilization with the machine learning field [151]. Rather than postulating finite-dimensional hypothesis spaces, e.g. using ARX, ARMAX or Laguerre models, the new paradigm formulates the problem as function estimation possibly in an infinite-dimensional space. In the context of linear system identification, the elements of such space are all possible impulse responses. The intrinsical ill-posedness of the problem is circumvented using regularization methods that also admit a Bayesian interpretation [158]. In particular, the impulse response is modelled as a zero-mean Gaussian process. In this way, prior information is introduced in the identification process just assigning a covariance, named also kernel in the machine learning literature [166]. In view of the increasing importance of these kernel methods also in the general SYSID scenario, some of the key mathematical tools and concept of these learning techniques should be made accessible to the control community, e.g. reproducing kernel Hilbert spaces [8, 156], kernel methods and regularisation networks [58, 148, 202] and the connection with the theory of Gaussian processes [81, 158]. It is also worth noting that a straight application of these techniques in the control field is doomed to fail unless some key features of the system identification problem are taken into account. First, as already recalled, the relationship between the unknown function and the measurements is not direct, as typically assumed in the machine learning setting, but instead indirect, through the convolution with the system input. Furthermore, in system identification it is essential that the estimation process be informed on the stability of the impulse response. In this regard, a recent major advance has been the introduction of new kernels which include information on impulse response exponential stability [36, 151]. These kernels depend on some hyperparameters which can be estimated from data e.g. using marginal likelihood maximization. This procedure is interpretable as the counterpart of model order selection in the classical PEM paradigm but, as it will be shown, it turns out to be much more robust, appearing to be the real reason of success of these new procedures. Other research directions recently developed have been the justification of the new kernels in terms of Maximum Entropy arguments [152], the analysis of these new approaches in a classical deterministic framework leading to the derivation of the

optimal kernel [36], as well as the extension of these new techniques to the estimation of optimal predictors [150].

Next we would like to introduce two topics or concepts in machine learning which will be mainly applied in this thesis.

1.4.1 Why Choose Marginal Likelihood

The benefit of marginal likelihood can be quoted and summarised from [208] and Professor Michael Jordan's lectures on "Bayesian Modelling and Inference" at UC Berkeley. If one were to plot the "classical" likelihood with respect to the "complexity" of the model, one would find that as the complexity of the model increased, so would the likelihood. In this setting, the likelihood is obtained after estimating the parameters using some statistical methods (e.g. maximum likelihood), and then plotting the value of $\mathbf{P}(\mathbf{y}|\boldsymbol{\beta})$. However, the problem of using this method of comparison is that more "complex" models will always do better than simpler models. Because more complex models have more parameters, whatever can be done in a simple model can be done in a complex model, and thus, a more complex model will lead to a beeter fit. The downside is that this leads to **overfitting**, which is particularly damaging in the setting of *prediction*. In frequentist statistics, many methods have been developed to penalise the likelihood with respect to increasing complexity of the model. However, these methods are typically only justified after proposing the idea and then performing a series of analyses to show that it has desirable properties, rather than being will-motivated from the start.

In the Bayesian framework, marginal likelihoods have a natural built-in penalty for more complex modes. At a certain point, the marginal likelihood will begin to decrease with increasing complexity, and hence, does not directly suffer from the overfitting problem that occurs when considering only likelihoods. The intuition for why the marginal likelihood will begin to decrease with increasing complexity is that as the complexity of the model increases, the prior will be spread out more thinly across both the "good" models and the "bad" models. Because the marginal likelihood is the likelihood integrated with respect to the prior, spreading the prior across too many models will place too little prior mass on the good models, and as a result, cause the marginal likelihood to decrease.

Another way to look at it is as follows. Suppose $\mathbf{y}$ is the data, $\boldsymbol{\beta}$ are the parameters and $\mathcal{M}$ is the models

$$\mathbf{P}(\boldsymbol{\beta}|\mathbf{y},\mathcal{M}) = \frac{\mathbf{P}(\mathbf{y}|\boldsymbol{\beta},\mathcal{M})\mathbf{P}(\boldsymbol{\beta}|\mathcal{M})}{\mathbf{P}(\mathbf{y}|\mathcal{M})}. \tag{1.8}$$

Solving for the marginal likelihood $\mathbf{P}(\mathbf{y}|M)$ in (1.8), we obtain

$$\mathbf{P}(\mathbf{y}|\mathcal{M}) = \frac{\mathbf{P}(\mathbf{y}|\boldsymbol{\beta},\mathcal{M})\mathbf{P}(\boldsymbol{\beta}|\mathcal{M})}{\mathbf{P}(\boldsymbol{\beta}|\mathbf{y},\mathcal{M})}. \tag{1.9}$$

Taking the $\log$ of (1.9) results in

$$\log \mathbf{P}(\mathbf{y}|\mathcal{M}) = \underbrace{\log \mathbf{P}(\mathbf{y}|\boldsymbol{\beta},\mathcal{M})}_{\text{log likelihood}} + \underbrace{\log \mathbf{P}(\boldsymbol{\beta}|\mathcal{M}) - \log \mathbf{P}(\boldsymbol{\beta}|\mathbf{y},\mathcal{M})}_{\text{penalty}} \tag{1.10}$$

This is true for any choice of $\boldsymbol{\beta}$, and in particular, the maximum likelihood estimate of $\boldsymbol{\beta}$. The $\log \mathbf{P}(\mathbf{y}|\boldsymbol{\beta},\mathcal{M})$ term in (1.10) is the $\log$ likelihood, and only increases with increasing model complexity. On the other hand, the $\log \mathbf{P}(\boldsymbol{\beta}|\mathcal{M}) - \log \mathbf{P}(\boldsymbol{\beta}|\mathbf{y},\mathcal{M})$ term can be viewed as a "penalty" that penalises against complex models. The overall sign of this penalty is negative because $\mathbf{P}(\boldsymbol{\beta}|\mathbf{y},\mathcal{M})$, the posterior, is generally larger than $\mathbf{P}(\boldsymbol{\beta}|\mathcal{M})$, the prior, assuming that given the data, the posterior "sharpens" up with respect to the prior. Hence, the penalty term balances out the increase in likelihood as the model complexity increases.

1.4.2 Why Choose Sparse Bayesian Learning

Referencing several excellent Ph.D. thesis relevant to SBL [215, 10, 231], the advantage of choosing SBL for proposing solutions to the SYSID problem can be summarised as follows

1. It is known that SBL algorithms have achieved top performance in many practical problems, or even solved some bottlenecks which other sparse signal recovery algorithms cannot solve [232, 233]. Besides, it is interesting to see that SBL has connections to Lasso-type algorithms, therefore, one can modify existing Lasso-type algorithms or design new Lasso-type algorithms to exploit a special, application-specific structure for better performance.

2. Its recovery performance is robust to the characteristics of the matrix $\mathbf{X}$, while other algorithms are not. For example, it has been shown that when columns

of $\mathbf{X}$ are highly coherent, SBL still maintains good performance, while other algorithms such as Lasso or other algorithms based on convex relaxations have seriously degraded performance [216]. Experiments also showed that when $\mathbf{X}$ is a non-random matrix or a sparse matrix, SBL algorithms maintain excellent performance. This advantage is very attractive to feature selection in bioinformatics, source localisation, and other applications, since in these application $\mathbf{X}$ is not a random matrix and its column are highly correlated. Such situation is very often encountered in SYSID problems, or general time series modelling problems, where $\mathbf{X}$ is constructed from time series data.

3. SBL has a number of desired advantages over many popular algorithms in terms of local and global convergence. It can be shown that SBL provides a sparser solution than Lasso-type algorithms [212]. In particular, in noiseless situations and under certain conditions, the global minimum of the SBL cost function is unique and corresponds to the true sparsest solution, while the global minimum of the cost function of Lassotype algorithms is not necessarily the true sparsest solution [213, 218]. Besides, it can be shown [214] that in certain settings, Lasso-type algorithms and ℓ_p ($p < 1$) minimisation algorithms always fail, while SBL succeed, regardless of $\mathbf{X}$ and the sparsity of $\boldsymbol{\beta}$. These advantages imply that SBL is a better choice in spare signal recovery and SYSID problems.

4. SBL provides scale-invariant solutions, while Lasso-type algorithms cannot [214]. Let $\boldsymbol{\beta}_{\text{SBL}}$ be the optimal solution provided by SBL with the sensing matrix $\mathbf{X}$. With a diagonal matrix $\mathbf{D}$, i.e., $\mathbf{X} \to \mathbf{XD}$, the optimal solution provided by SBL becomes $\mathbf{D}\boldsymbol{\beta}_{\text{SBL}}$. In contrast, for Lasso-type algorithms, there is no such linear relationship between the solutions. For example, the solution to the problem $\min \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$ has no such linear relationship with the solution to the rescaled problem $\min \|\mathbf{y} - \mathbf{XD}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_{\ell_1}$. This warns that rescaling $\mathbf{X}$ may be problematic in SYSID problems when time series data are normalised and the re-weighting procedure is used for each iteration of the iterative algorithms.

Admittedly, SBL is not perfect. The main drawback is that SBL generally involves large computational loads. Some strategies have been used to speed up SBL. For example, using the marginalized likelihood method [192], several fast algorithms have been derived [95, 9]. Using the connection between SBL and Lasso-type algorithms [212, 235], one can obtain optimal SBL solutions by iteratively performing

Lasso-type algorithms several times. Since Lasso-type algorithms become more efficient year by year, using this iteration strategy also greatly benefits SBL. However, SBL is still slower than some efficient algorithms, such as greedy algorithms or message passing algorithms. Thus, new strategies are required to speed up SBL, and more efficient SBL algorithms are needed.

Another drawback of SBL is that the estimation of noise variance is not reliable. Learning rules for the noise variance in most SBL algorithms are not effective in noisy environments. Thus, most SBL algorithms [213, 218, 95, 217] use some fixed sub-optimal values, or require users or other algorithms to provide the value, instead of learning it. Recently, an effective empirical strategy to enhance these learning rules has been proposed [235, 234], which helps SBL achieve satisfactory solutions. However, this strategy does not completely solve this problem. More effective methods and theoretical guidance are called for.

1.5 The Big Picture and Contributions

The main content of thesis is based on this thesis author's publications with co-authors. All of these publications are peer reviewed and the thesis author is the first author of all of them [138, 141, 143, 142, 136, 137, 139, 140].

1.5.1 A Story on Healthcare

The treatment of some disease relies on the development and evaluation of multivariate predictive models, to give "the right treatment to the right patient at the right time" by integrating large dataset from different sources. One important question from the patient is "what is my individual prognosis?" This question cannot always be easily answered by doctors. As an aid to diagnostics and treatment design a model can be developed to identify different patterns of progression based on sensor data from wearable devices for example.

Although the elapsed time between disease onset and the collection of data samples may be unknown, the samples are normally classified with staging phases (e.g. prognosis stages) that characterise the clinical or pathological status of disease progression. For each patient, a personalised and tailored dynamical model needs to be developed to explain the observations at each stage, see Figure 1.2.

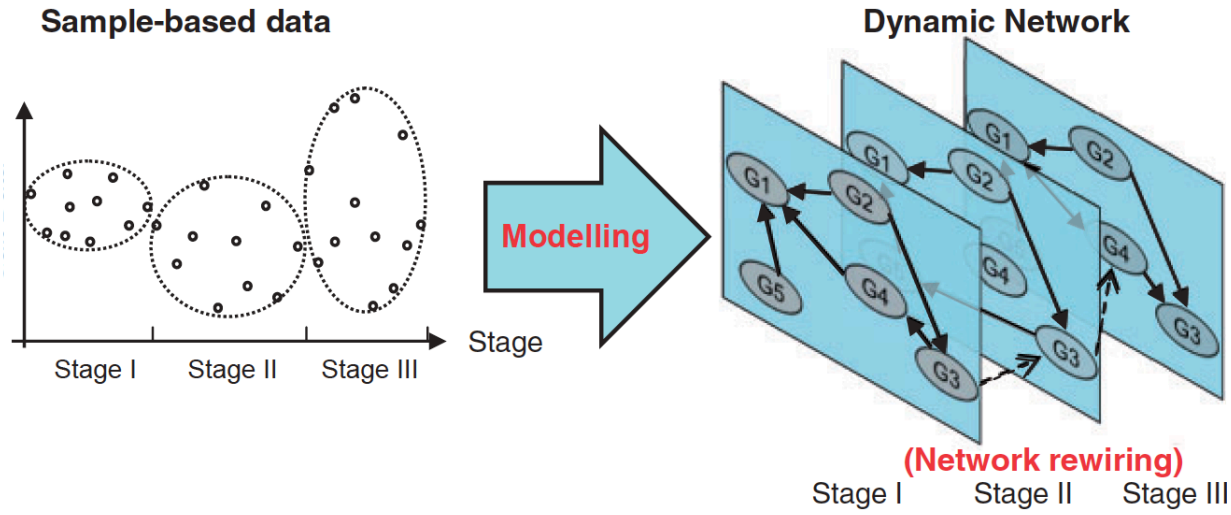


Fig. 1.2 Schematic illustration of the evolution of dynamical model duirng disease progression

For each individual patient k, $k = 1, \ldots,$ K, the dynamic model can be specified by differential/difference equations

$$\delta \mathbf{x}_k(t) = \begin{cases} \mathbf{f}_{k1}(\mathbf{x}_k(t), \mathbf{p}_{k1}), & \text{if } t \in [t_{k1}, t_{k2}), & \text{Stage 1} \\ \mathbf{f}_{k2}(\mathbf{x}_k(t), \mathbf{p}_{k2}), & \text{if } t \in [t_{k2}, t_{k3}), & \text{Stage 2} \\ \vdots & \vdots & \vdots \\ \mathbf{f}_{ks}(\mathbf{x}_k(t), \mathbf{p}_{ks}), & \text{if } t \in [t_{k,\text{start}}, t_{k,\text{end}}], & \text{Stage s} \end{cases}. \tag{1.11}$$

δ denotes the differentiation operator for continuous-time systems, or the shift operator for discrete-time system. $\mathbf{x}_k$ is the quantity of interested observed for the k-th individual. $\mathbf{f}_{ks}$ describes the dynamics of $\mathbf{x}_k$ and $\mathbf{p}_{ks}$ are the corresponding parameters of $\mathbf{f}_{ks}$. It should be noted that $\mathbf{f}_{ks}$ may be nonlinear. t_{ks} denotes the starting time of stage s, start $= 1, \ldots,$ s (s may be unknown). Then the model identification problem of interest can be summarised as follows:

Problem 1 *For each patient k, given the observed the heterogeneous sample-based dataset $\mathbf{x}_k(t)$, we are interested in identifying the onset time $t_{k,start}$ of stage s; the form of $\mathbf{f}_{ks}$ and the associated parameters $\mathbf{p}_{ks}$ at each stage s.*

1.5.2 Strategy

At time instant t, the system in (1.11) for the patient k, can be expressed as a linear combination of several dictionary functions by letting $y_k(t) \triangleq \delta \mathbf{x}_k(t)$ as the output,

$$y_k(t) = \sum_{n=1}^{N} \mathbf{X}_{kn}(t) \boldsymbol{\beta}_{kn}(t), \tag{1.12}$$

$\mathbf{X}_{kn}$ encodes the type of functions $\mathbf{f}(\mathbf{x}_k(t), \mathbf{p}_k)$ that are used to describe the system, $\boldsymbol{\beta}_{kn}$ encodes the unknown parameter $\mathbf{p}_k$. For nonlinear systems, more details on such expansion can be found in [138, 136, 143, 139, 140].

Problem formulation

If there is no switch (s $= 1$ in (1.11) and $\boldsymbol{\beta}_{kn}(t) = \boldsymbol{\beta}_{kn}, \forall t$) and only one patient's data is collected, the model identification problem can be formulated as the following regularised regression problem (see [138, 136, 139])

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \sum_{t=1}^{T} \|y(t) - \sum_{n=1}^{N} \mathbf{X}_n(t) \boldsymbol{\beta}_n\|_2^2 + \lambda \sum_{n=1}^{N} \|\boldsymbol{\beta}_n\|_{\ell_0}. \tag{1.13}$$

In particular when the data is contaminated by measurement noise, the problem is discussed in [137]. If all the K patients' data are taken into account, the model identification problem can be formulated as (see [140])

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \sum_{k=1}^{\mathrm{K}} \sum_{t=1}^{T} \|y_k(t) - \sum_{n=1}^{N} \mathbf{X}_{kn}(t)\boldsymbol{\beta}_{kn}\|_2^2 + \lambda \sum_{n=1}^{N} \left\| \sqrt{\textstyle\sum_{k=1}^{\mathrm{K}} w_{kn}^2} \right\|_{\ell_0}. \tag{1.14}$$

If there are switches (s > 1 in (1.11)) and only one patient's data is collected, the model identification problem can be formulated as (see [143]),

$$\begin{aligned} \min_{\boldsymbol{\beta}} & \frac{1}{2} \sum_{t=1}^{T} \|y(t) - \sum_{n=1}^{N} \mathbf{X}_{kn}(t)\boldsymbol{\beta}_{kn}(t)\|_2^2 \\ & \lambda_1 \sum_{t=1}^{T} \sum_{n=1}^{N} \|\boldsymbol{\beta}_{kn}(t)\|_{\ell_0} + \lambda_2 \sum_{t=1}^{T-1} + \sum_{n=1}^{N} \|\boldsymbol{\beta}_{kn}(t+1) - \boldsymbol{\beta}_{kn}(t)\|_{\ell_0}. \end{aligned} \tag{1.15}$$

Algorithm Development

The regularised regression problems in Eqs. (1.13)-(1.15) are NP-hard. To be able to offer some time-efficient solution to this optimisation problem, one typically considers empirical relaxations to this optimisation problem, e.g. replacing ℓ_0 with ℓ_1 minimisation, which is known to give the tightest convex relaxation to ℓ_0 minimisation problems. This gives rise to the well-known Lasso type algorithms [188, 189, 62, 190]. However, such empirical relaxations sometimes yield poor performance mainly due to the high coherence of the dictionary matrix [52]. The variational Bayesian framework often yields better performance by integrating the marginal likelihood maximisation and usage of hyperparameter to control the sparsity. One of the drawbacks of the Bayesian framework is the associated computational cost, which is higher in comparison with currently existing state-of-art algorithms. In [136, 140, 137], distributed convex optimisation algorithms are proposed to identify large-scale nonlinear state-space systems from high-dimensional time series data, i.e., large K, T, N in Eqs. (1.13)-(1.15). This algorithm can exploit multiple computation units in parallel which makes big time series data modelling possible.

Real-time Monitoring

Another important problem is to monitor the status of the patient and give early diagnostic and warning signals before before his health state starts to deteriorate. This can be formulated as a problem of detecting the change point of $\boldsymbol{\beta}_{kn}(t)$ when

$\boldsymbol{\beta}_{kn}(t+1) - \boldsymbol{\beta}_{kn}(t) \neq 0$. A similar mathematical problem for large-scale nonlinear power systems has been investigated in [141, 142], i.e. the problem of fault diagnosis on transmission lines. The strategy can be potentially applied to diseases with different patterns of progression.

Pharmacokinetics

Last but not least, prediction of human pharmacokinetics, dose and drug interactions is also of importance. If some measured quantity of interest is treated as output and the dosage of certain drugs is treated as input, a personalised transfer function from input to output should be calibrated for each patient. The modelling techniques in [138, 136, 140, 137] can be potentially used to this effect.

1.5.3 Contributions and Outlines

The central hypothesis of the thesis is that a suite of learning and optimisation techniques based on exhibiting structures and sparseness can and will enable more accurate reliable solutions to larger and richer time series modelling across a wide variety of domains. In particular, this thesis focuses on addressing both statistical and computational aspects of learning from high-dimensional large-scale structured time series data in the following three different aspects:

Statistical modelling of nonlinear dynamical systems from time series data: In Part I of the thesis, we propose a series of sparse statistical modelling formulations by exploiting the structural information in both the time series data and dynamical systems. In Chapter 2, we deal with the identification of time-invariant nonlinear dynamical system from a single dataset [138, 136, 139]. In Chapter 3, we consider the identification of time-invariant nonlinear dynamical system, but from heterogeneous datasets [140]. In Chapter 3, we focus on time-varying dynamical systems. In the first part of Chapter 3, we investigate regime-switch systems modelling from "static" time series data [143]; in the second part, we investigate the identification of "abrupt change" models from "streaming" time series data and in particular the trend filtering and fault diagnosis problems [141, 142]. At the end of Part I, we discuss some technical issues related to dynamical systems identification, model structure selections, and data preprocessing in practice.

Learning and optimisation framework for large-scale and high-dimensional time series data: In Part II of the thesis, a scalable, general algorithm framework based on machine learning and convex optimisation is proposed for the inference of models presented in Part I. A schematic of the work flow for algorithm derivations in Chapters 6, 7 of Part II can be summarised in Table 1.1.

1. Specify the likelihood of the data;
2. Specify the structural sparse prior controlled by structured hyperparameters for various penalties in the original ℓ_0 problem;
3. Formulate the nonconvex optimisation problem joint in both parameter to be estimated and hyperparameters using a marginal likelihood maximisation framework;
4. Apply the convex concave procedure to convexify the nonconvex optimisation problem;
5. Derive the iterative re-weighed ℓ_1 or ℓ_2 type algorithm (the first iteration usually start with the empirical Lasso/Ridge regression type algorithms);
6. (Optionally) Formulate the distributed version of the algorithm for "big data" analysis purposes.

Table 1.1 The unified work flow for algorithm derivations in Chapters 6, 7 of Part II

Applications In Part III, we apply the proposed methods to a broad class of problems in systems biology, physics and engineering [137].

A schematic of the structure of this thesis is shown in Figure 1.3. It covers the nonlinear systems class in this thesis and two technical foundations for the algorithms developed in this thesis is illustrated, i.e., machine learning and convex optimisation. In particular, we emphasise the importance of closing the loop (the red arrow on the left of Figure 1.3) from Part III to Part I. Modelling time series data requires deep understanding and insight into the underlying dynamical systems of specific application domains.

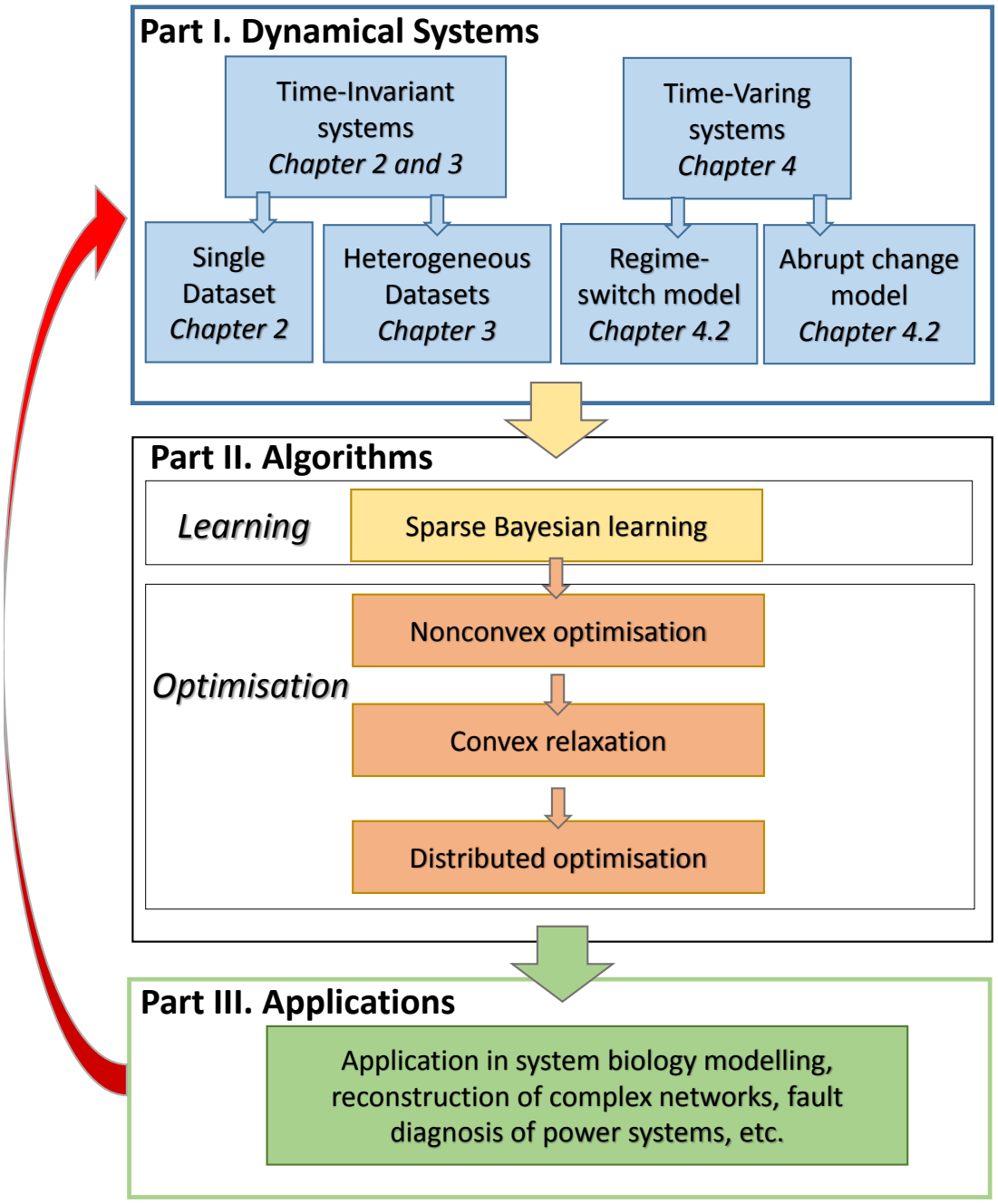


Fig. 1.3 The outline of this thesis.

Part I

Dynamical Systems

Chapter 2

Nonlinear Dynamical Systems

2.1 Introduction

Identification of nonlinear dynamical systems from noisy time-series data is relevant to many different fields such as systems/synthetic biology, econometrics, finance, chemical engineering, social networks, etc. Yet, the development of general reconstruction techniques remains challenging, especially due to the difficulty of adequately identifying nonlinear systems [117]. Nonlinear dynamical system reconstruction aims at recovering the set of nonlinear equations associated with the system from noisy time-series observations. The importance of nonlinear structure identification and its associated difficulties have been widely recognised [118, 173].

Since, typically, nonlinear functional forms can be expanded as sums of terms belonging to a family of parameterised functions (see Sec. 5.4, [117]), an usual approach to identify nonlinear black-box models is to search amongst a set of possible nonlinear terms (e.g., basis functions) for a parsimonious description coherent with the available data [72]. A few choices for basis functions are provided by classical functional decomposition methods such as Volterra expansion, Taylor polynomial expansion or Fourier series [117, 15]. This is typically used to model systems such as those described by Wiener and Volterra series [210, 15], neural networks [128], nonlinear auto-regressive with exogenous inputs (NARX) models [112], and Hammerstein-Wiener [12] structures, to name just a few examples.

Graphical models have been proposed to capture the structure of nonlinear dynamical networks. In the standard graphical models where each state variable represents a node in the graph and is treated as a random variable, the nonlinear relation among nodes can be characterised by factorising the joint probability distribution according to a certain directed graph [104, 147, 177]. However, standard graphical models are often not adequate for dealing with times series directly. This is mainly due to two aspects inherent to the construction of graphical models. The first aspect pertains to the efficiency of graphical models built using time series data. In this case, the building of graphical models requires the estimation of conditional distributions with a large number of random variables [16] (each time series is modelled as a finite sequence of random variables), which is typically not efficient. The second aspect pertains to the estimation of the moments of conditional distribution, which is very hard to do with a limited amount of data especially when the system to reconstruct is nonlinear. In the case of linear dynamical systems, the first two moments can sometimes be estimated from limited amount of data [11, 123]. However,

higher moments typically need to be estimated if the system under consideration is nonlinear.

In this Chapter, we will start with an introduction to linear time-invariant systems in Section 2.2; then we will extend this introduction to nonlinear time-invariant systems in Section 2.3; in Section 2.4, we will give the regression problem formulation for the nonlinear SYSID problem from *single* experiment. The regression problem is further formulated as a ℓ_0 type optimisation problem in Section 2.4.2 and a tentative empirical ℓ_1 relaxation is proposed in Section 2.4.3. At the end of the Chapter, we discuss the uniqueness of the solution and the selection of the basis functions. The latter discussion also applies to other Chapters throughout this thesis. The algorithms for this Chapter will be provided in Section 6.6 of Chapter 6.

2.2 Linear Time-Invariant Systems

2.2.1 Impulse Response and Transfer Function

Impulse Response

As in [117], we first introduce some preliminary for impulse response and transfer function. Consider a system with a scalar input signal $u(t)$ and a scalar output signal $y(t)$. The system is said to be *time invariant* if its response to a certain input signal does not depend on absolute time. It is said to be *linear* if its output responses to a linear combination of inputs is the same linear combination of the output responses of the individual inputss. Furthermore, it is said to be *causal* if the output at a certain time depends on the input up to that time only.

It is well known that a linear, time-invariant, causal system can be described by its *impulse response* (or *weighting functions*) $g(\tau)$ as follows:

$$y(t) = \int_{\tau=0}^{\infty} g(\tau)u(t-\tau)d\tau \tag{2.1}$$

Knowing $\{g(\tau)\}_{\tau=0}^{\infty}$ and knowing $u(s)$ for $s \leq t$, we can consequently compute the corresponding output $y(s)$, $s \leq t$ for any inputs. The impulse response is thus a complete characterisation of the system.

In practice, 1). inputs and outputs are observed in discrete time due to the typical data-acquisition mode; 2). disturbance is ubiquitous due to measurement noise and/or uncontrollable inputs. We thus assume $y(t)$ to be observed at the *sampling instants* $t_k = kT$, $k = 1, 2, \ldots$: $y(kT) = \int_{\tau=0}^{\infty} g(\tau)u(kT-\tau)d\tau$. The interval T will be

called the *sampling interval*. It is, of course, also possible to consider the situation where the sampling instants are not equally spread. By defining the *impulse response* of the system $\{g(k)\}_{k=1}^{\infty}$, we have

$$y(t) = \sum_{k=1}^{\infty} g(k)u(t-k) + v(t), \quad t = 0, 1, 2, \ldots \tag{2.2}$$

where $v(t)$ is the disturbance. However, the probability distribution of the disturbance is *not known a priori*. Typically, the disturbance is assumed to be

$$v(t) = \sum_{k=1}^{\infty} h(k)e(t-k).$$

where $\{e(t)\}$ is *white noise*, i.e., a sequence of independent (identical distributed) random variables with a certain probability density function. Although this description does not allow completely general characterisations of all possible probabilistic disturbance, it is versatile enough for most practical purposes. Then we have

$$y(t) = \sum_{k=1}^{\infty} g(k)u(t-k) + \sum_{k=1}^{\infty} h(k)e(t-k), \quad t = 0, 1, 2, \ldots \tag{2.3}$$

Transfer Function

It will be convenient to introduce a shorthand notation for (2.3) we introduce the *forward shift operator* q by

$$qu(t) = u(t+1)$$

and the *backward shift operator* q^{-1}:

$$q^{-1}u(t) = u(t-1).$$

This is exactly equivalent to the lag operator as introduced in time series literature (see [73, eq.(2.1.3)] for example), i.e.,

$$Lu(t) = u(t-1).$$

We introduced the notation

$$G(q) = \sum_{k=1}^{\infty} g(k)q^{-k} \tag{2.4}$$

which is called the *transfer operator* or the *transfer function* of the linear system. It should be noted that, the term *transfer function* should be reserved for the z-transform of $\{g(k)\}_{k=1}^{\infty}$, that is

$$G(z) = \sum_{k=1}^{\infty} g(k) z^{-k}. \tag{2.5}$$

And similarly with

$$H(q) = \sum_{k=1}^{\infty} h(k) q^{-k} \tag{2.6}$$

2.2.2 Linear Models and Sets of Linear Models

A linear time-invariant model is specified by the impulse response $\{g(k)\}_1^{\infty}$, the spectrum $\Phi_v(\omega) = \lambda |H(e^{j\omega})|^2$ of the additive disturbance, and, possibly, the probability density function (PDF) of the distrubance $e(t)$. A complete model is thus given by

$$\begin{aligned} y(t) =& g(1)u(t-1) + g(2)u(t-2) + \ldots + g(\infty)u(t-\infty) \\ &+ h(1)e(t-1) + h(2)e(t-2) + \ldots + h(\infty)e(t-\infty) \\ =& G(q)u(t) + H(q)e(t) \\ p_e(\cdot),& \quad \text{the PDF of } e \end{aligned} \tag{2.7}$$

with

$$G(q) = \sum_{k=1}^{\infty} g(k) q^{-k}, \qquad H(q) = 1 + \sum_{k=1}^{\infty} h(k) q^{-k}. \tag{2.8}$$

A particular model thus corresponds to the specification of the three functions G, H, and p_e. It is in most cases impractical to make this specification by enumerating the infinite sequences $\{g(k)\}$, $\{h(k)\}$ together with the function $p_e(\cdot)$. Instead one chooses to work with structures that permit the specification of G and H in terms of a finite number of numerical values. Rational transfer functions and finite-dimensional state-space descriptions are typical examples of this. Also, most often the PDF p_e is not specified as a function, but described in terms of a few numerical characteristics, typically the first and second moments

$$\begin{aligned} \mathbb{E}(e(t)) &= \int x p_e(x) dx = 0, \\ \mathbb{E}(e^2(t)) &= \int x^2 p_e(x) dx = \lambda. \end{aligned} \tag{2.9}$$

It is also common to assume that $e(t)$ is Gaussian, in which the PDF is entirely specified by (2.9). The specification of (2.7) in terms of finite number of numerical values,

or coefficients, has another and most important consequence for the purposes of system identification. Quite often it is not possible to determine these coefficients *a priori* from knowledge of the physical mechanisms that govern the system's behaviour. Instead the determination of all or some of them must be left to estimation procedures. This means that the coefficients in question enter the model (2.7) as *parameters to be determined*. We shall generally denote such parameters by the vector $\boldsymbol{\theta}$, and thus have a model description

$$\begin{aligned} &y(t) = G(q, \boldsymbol{\theta})u(t) + H(q, \boldsymbol{\theta})e(t) \\ &p_e(\cdot), \quad \text{the PDF of e} \end{aligned} \tag{2.10}$$

The parameters vector $\boldsymbol{\theta}$ then ranges over a subset of $\mathbb{R}^N$, where N is the dimension of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} \in D_{\mathcal{M}} \subset \mathbb{R}^N \tag{2.11}$$

Notice that (2.10) and (2.11) no longer is *a* model; it is a *set of models*, and it is for the estimation procedure to select that member in the set that appears to be most suitable for the purpose in question.

Perhaps the most immediate way of parametrising G and H is to represent them as rational functions and let the parameters be the numerator and denominator coefficients.

2.2.3 ARX Model Structure

Probably the most simple input-output relationship is obtained by describing it as linear difference equation:

$$y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \ldots + b_{n_b} u(t-n_b) + e(t). \tag{2.12}$$

Since the white-niose term $e(t)$ here enters as a direct error in the difference equation, the model (2.12) is often called as an *equaiton error model* (structure). The adjustable parameters are in this case

$$\boldsymbol{\theta} = [a_1 \; a_2 \; \ldots \; a_{n_a} \; b_1 \; b_2 \; \ldots \; b_{n_b}]^\top . \tag{2.13}$$

If we introduce

$$A(q) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a},$$

and

$$B(q) = b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b}.$$

We see that (2.12) corresponds to (2.10) with

$$G(q,\theta) = \frac{B(q)}{A(q)}, \qquad H(q,\theta) = \frac{1}{A(q)}. \tag{2.14}$$

Remark 1 *It may seem annoying to use q as an argument of $A(q)$, being a polynomial in q^{-1}. The reason for this is, however, simply to be consistent with the conventional definition of the* **z***-transform.*

2.2.4 ARMAX Model Structure

The basic disadvantage with the simple model (2.12) is the lack of adequate freedom in describing the properties of the disturbance term. We could add flexibility to that by describing the equation as a moving average of white noise. This gives the model

$$\begin{aligned} &y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) \\ =&b_1 u(t-1) + \ldots + b_{n_b} u(t-n_b) + e(t) + c_1 e(t-1) + \ldots + c_{n_e} e(t-n_c), \end{aligned} \tag{2.15}$$

with

$$C(q) = 1 + c_1 q^{-1} + \ldots + c_{n_c} q^{-n_c}.$$

It can be rewritten as

$$A(q)y(t) = B(q)u(t) + C(q)e(t) \tag{2.16}$$

and clearly corresponds to (2.7) with

$$G(q,\boldsymbol{\theta}) = \frac{B(q)}{A(q)}, \quad H(q,\boldsymbol{\theta}) = \frac{C(q)}{A(q)}, \tag{2.17}$$

where now

$$\boldsymbol{\theta} = \begin{bmatrix} a_1 & \ldots & a_{n_a} & b_1 & \ldots & b_{n_b} & c_1 & \ldots & c_{n_c} \end{bmatrix}^\top.$$

In view of the moving average (MA) part $C(q)e(t)$, the model (2.16) will be called ARMAX. The ARMAX model has become a standard tool in control and econometrics for both system description and control design. A version with an enforced integration in the noise description is the ARIMA(X) model (I for integration, with or without the X-variable u), which is useful to describe systems with slow disturbance:

see the book by Box and Jenkins [25]. It is obtained by replacing $y(t)$ and $u(t)$ in (2.16) by their differences $\Delta y(t) = y(t) - y(t-1)$.

2.2.5 Linear Regression Model

Let us compute the predictor for (2.12), which gives

$$\hat{y}(t|\boldsymbol{\theta}) = B(q)u(t) + [1 - a(q)]\, y(t) \tag{2.18}$$

Clearly, this expression could have more easily been derived from (2.12). Without a stochastic framework, the predictor (2.18) is a natural choice if the the term $e(t)$ in (2.12) is considered to be "insignificant" or "difficult to guess." It is thus perfect natural to work with the expression (2.18) also for the "deterministic" models.

Now we introduce the vector

$$\phi(t) = \begin{bmatrix} -y(t-1) & \dots & -y(t-n_a) & u(t-1) & \dots & u(t-n_b) \end{bmatrix}^\top. \tag{2.19}$$

Then (2.18) can be rewritten as

$$\hat{y}(t|\boldsymbol{\theta}) = \phi(t)^\top \boldsymbol{\theta}. \tag{2.20}$$

This is the important property of (2.12) that we alluded to previously. The predictor is a scalar product between a known data vector $\phi(t)$ and the parameter vector θ. Such a model is called a *linear regression* in statistics, and the vector $\phi(t)$ is known as the *regression vector*. It is of importance since powerful and simple estimation method can be applied for the determination of θ.

In case some coefficient of the polynomials A and B are known, we arrive at the linear regression of the form

$$\hat{y}(t|\theta) = \phi^\top(t)\theta + \mu(t) \tag{2.21}$$

where $\mu(t)$ is a known term.

2.3 Nonlinear Time-Invariant Systems

In Eq. (2.21), we defined a linear regression model structure where the prediction is linear in the parameters:

$$\hat{y}(t|\theta) = \phi(t)^\top \boldsymbol{\theta}. \tag{2.22}$$

To describe a linear difference equation, the components of the vector $\phi(t)$ (i.e., the regressors), were chosen as lagged input and output values: see (2.19). When using (2.22) it is, however, immaterial how $\phi(t)$ is formed: what matters is that it is a known quantity at time t. We can thus let it contain arbitrary transformations of measured data. Let, as usual, y^t and u^t denote the input and output sequences up to time t. Then we could write

$$\hat{y}(t|\boldsymbol{\theta}) = \theta_1\phi_1(u^t, y^{t-1}) + \ldots + \theta_d\phi_d(u^t, y^{t-1}) = \phi(t)^\top\boldsymbol{\theta} \tag{2.23}$$

with arbitrary functions $\phi(t)$ of past data. The structure (2.23) could be regarded as a finite-dimensional parametrisation of a general, unknown, nonlinear predictor. The key is how to choose the functions $\phi_i(u^t, y^{t-1})$. Next we will discuss on how to choose $\phi_i(u^t, y^{t-1})$ using polynomial terms.

Example 1 *As an example of the above, consider the following model of polynomial terms single-input single-output (SISO) nonlinear autoregressive system with exogenous inputs (NARX model) [112]*

$$x(t+1) = 0.7x^5(t)x(t-1) - 0.5x(t-2) + 0.6u^4(t-2) - 0.7x(t-2)u^2(t-1) + \xi(t). \tag{2.24}$$

with $x \in \mathbb{R}$, $u \in \mathbb{R}$, and $\xi \in \mathbb{R}$. The maximal state and input memory orders are $m_x = 2$ and $m_u = 2$ respectively.

Suppose we collect $M + 2$ data samples satisfying (2.24). We can then write the corresponding dynamics as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\Xi}$$

with

$$\begin{aligned}
\mathbf{y} &\triangleq \left[\; x(4), \ldots, x(M) \;\right]^T \in \mathbb{R}^{(M-3)\times 1}, \\
\mathbf{X} &\triangleq \begin{bmatrix} x^5(3)x(2) & x(1) & u^4(1) & x(1)u^2(2) \\ \vdots & \vdots & \vdots & \vdots \\ x^5(M-1)x(M-2) & x(M-3) & u^4(M-3) & x(M-3)u^2(M-2) \end{bmatrix} \in \mathbb{R}^{(M-3)\times 4}, \\
\boldsymbol{\theta} &\triangleq [0.7, -0.5, 0.6, 0.7]^T \in \mathbb{R}^{4\times 1} \\
\boldsymbol{\Xi} &\triangleq \left[\; \xi(3), \ldots, \xi(M-1) \;\right]^T \in \mathbb{R}^{(M-3)\times 1}.
\end{aligned} \tag{2.25}$$

2.3.1 Nonlinear Time-Invariant Systems

Now we give a formal introduction to the nonlinear time-invariant systems considered in this and next Chapters. With appropriately chosen embedding dimension or memory of the system κ_i (resp. ν_j) for state (resp. input) variable x_i (resp. u_j), for $i = 1, \ldots, n_{\mathbf{x}}$, $j = 1, \ldots, n_{\mathbf{u}}$ and $k = 1, \ldots, n_{\boldsymbol{\xi}}$ one can obtain delayed coordinate and $\mathcal{N}_{\mathbf{x}}$-dimensional stacked state vector

$$\begin{aligned} \mathbf{x}_i(t) &= [x_i(t), x_i(t-1), \ldots, x_i(t-(\kappa_i-1))]^\top \in \mathbb{R}^{\kappa_i}, \\ \mathbf{X}(t) &= \left[(\mathbf{x}_1(t))^\top, \cdots, (\mathbf{x}_{n_{\mathbf{x}}}(t))^\top\right]^\top \in \mathbb{R}^{\mathcal{N}_{\mathbf{x}}}, \quad \mathcal{N}_{\mathbf{x}} = \textstyle\sum_{i=1}^{n_{\mathbf{x}}} \kappa_i; \end{aligned} \tag{2.26}$$

and delayed coordinate and $\mathcal{N}_{\mathbf{u}}$-dimensional stacked input vector

$$\begin{aligned} \mathbf{u}_j(t) &= [u_j(t), u_j(t-1), \ldots, u_j(t-(\nu_j-1))]^\top \in \mathbb{R}^{\nu_j}, \\ \mathbf{U}(t) &= [(\mathbf{u}_1(t))^\top, \cdots, (\mathbf{u}_{n_{\mathbf{u}}}(t))^\top]^\top \in \mathbb{R}^{\mathcal{N}_{\mathbf{u}}}, \quad \mathcal{N}_{\mathbf{u}} = \textstyle\sum_{j=1}^{n_{\mathbf{u}}} \nu_j. \end{aligned} \tag{2.27}$$

and delayed coordinate and $\mathcal{N}_{\boldsymbol{\xi}}$-dimensional stacked noise vector

$$\begin{aligned} \boldsymbol{\xi}_k(t) &= [\xi_k(t), \xi_k(t-1), \ldots, \xi_k(t-(\upsilon_k-1))]^\top \in \mathbb{R}^{\upsilon_k}, \\ \boldsymbol{\Xi}(t) &= [(\boldsymbol{\xi}_1(t))^\top, \cdots, (\boldsymbol{\xi}_{n_{\boldsymbol{\xi}}}(t))^\top]^\top \in \mathbb{R}^{\mathcal{N}_{\boldsymbol{\xi}}}, \quad \mathcal{N}_{\boldsymbol{\xi}} = \textstyle\sum_{k=1}^{n_{\boldsymbol{\xi}}} \upsilon_k. \end{aligned} \tag{2.28}$$

Suppose τ is a natural number, the *delayed* element in $\mathbf{x}_i(t)$, $\mathbf{u}_j(t)$ and $\boldsymbol{\xi}_k(t)$ can be indexed as $x_i(t-\tau)$, $u_j(t-\tau)$ and $\xi_k(t-\tau)$ respectively. τ can be interpreted as temporal index of the data, as well as the sampling interval and delay which can be an arbitrary positive real number. To ease notations and efficient temporal index of the data, we assume $\tau = 1$ throughout the thesis. The system dynamics are typically written in the Langevin form and referred to as "the Langevin equation" and the corresponding observation equations, $i = 1, \ldots, n_{\mathbf{x}}$:

$$x_i(t+1) = \mathbf{F}_i\left(\mathbf{X}(t), \mathbf{U}(t)\right) + \mathbf{G}_i\left(\mathbf{X}(t), \mathbf{U}(t)\right) \cdot \boldsymbol{\Xi}(t), \tag{2.29}$$

$$z_i(t) = x_i(t) + \epsilon_i(t), \tag{2.30}$$

$x_i(t)$ is the system or state variable, the observation variable $z_i(t)$ is the collected data contaminated by the ubiquitous observational noise. Similarly, we can define the delayed coordinate and $\mathcal{N}_{\mathbf{z}}$-dimensional stacked observation vector $\mathbf{z}_i(t)$ and $\boldsymbol{\mathcal{Z}}(t)$. The underlying assumption here is *all* state variables are observable, i.e., $\mathcal{N}_{\mathbf{z}} = \mathcal{N}_{\mathbf{x}}$. In probability and mathematical finance community, $\mathbf{F}_i$ is the drift coefficient and $\mathbf{G}_i$ is the diffusion coefficient.

Remark 2 *On the left hand side of* (2.29)*, we replace* $x_i(t+1)$ *with some other quantities, i.e.,* $\delta(t, \mathbf{x}_i(t+1))$*.* $\delta(t, \mathbf{x}_i(t))$ *can be defined depending whether there is state variable in the expression. In the state-independent scenario, it can be expressed as a polynomial function of time* t*, for example,*

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq t \text{ or } t^2 \text{ etc}, \tag{2.31a}$$

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq \log t, \quad t > 0, \tag{2.31b}$$

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq \text{constant}. \tag{2.31c}$$

Applications include factor asset pricing models [38], conservation laws in physics and chemistry.

In the state-dependent scenario, to give a few examples in the following, it can be expressed as

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq x_i(t+1), \tag{2.32a}$$

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq x_i(t+1) - x_i(t), \tag{2.32b}$$

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq \log x_i(t+1), \quad x_i(t) > 0, \tag{2.32c}$$

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq \log x_i(t+1) - \log x_i(t), \quad x_i(t), x_i(t-\tau) > 0. \tag{2.32d}$$

In particular, when the system can be characterised by differential equations, we have

$$\delta(t, \mathbf{x}_i(t+1)) \triangleq \frac{dx_i(t)}{dt} = \dot{x}_i(t). \tag{2.33}$$

Remark 3 *Here we discuss the identification of continuous system. It should be noted that continuous system can be covered if* $\dot{x}_i(t)$ *can be obtained or assumed to estimated with confidence. Otherwise, continuous system identification is not covered in the thesis. In practice, estimation of the first derivative data matrix is not a trivial task. After trying various techniques, we decided to use the techniques proposed in [48]. The details will be given later in Section 5.6. The other issue that needs to be pointed out is how to deal measurement noise* $\epsilon_i(t)$ *in Eq.* (2.30)*. This issue will be discussed later in Section 5.5.*

2.3.2 Some Key Assumptions

In all its generality, the formulation in Eq. (2.29) and other variations in Remark 2 encompasses a wide variety of networked models in physics, chemistry, biology, engineering, finance and economics.

The objective is thus to identify the possibly nonlinear functions $\mathbf{F}_i$ (and their associated parameters) in (2.29) given measured noisy time series data z_i in (2.30) of the state variables x_i. However, it's too ambitious to accomplish the objective without any assumptions. Before proposing our method, some (hopefully) mild assumptions need to be made on the structure form of $\mathbf{F}_i$; estimation or calculation of left hand side quantity $\delta(\cdot,\cdot)$; distribution and form of dynamic and/or observation noise.

We focus on systems where our *a priori* knowledge about the underlying principles allows us to assume that the model will adopt a concise description in terms of a set of candidate *basis* (or dictionary) functions, as is the case in many of the applications of interest in physics, chemistry, biology, engineering, finance and economics. In particular, we make the following key assumption.

Assumption 1 *$x_i(t)$ in* (2.26)*, $\mathbf{u}_j(t)$ in* (2.27)*, $\boldsymbol{\xi}_k(t)$ in* (2.28) *and $\epsilon_i(t)$ in* (2.30) *are assumed to be bounded and independent with each other for $i = 1, \ldots, n_{\mathbf{x}}$, $j = 1, \ldots, n_{\mathbf{u}}$ and $k = 1, \ldots, n_{\boldsymbol{\xi}}$.*

Assumption 2 *The function $\mathbf{F}_i(\mathbf{X}(t), \mathbf{U}(t))$ can be represented as linear combinations of dictionary functions:*

$$\boldsymbol{\varphi}_i(\mathbf{X}(t), \mathbf{U}(t)) = \left\{\varphi_{in} : \mathbb{R}^{n_{\mathbf{x}}+n_{\mathbf{u}}} \to \mathbb{R}\right\}_{n=1}^{N_i} \tag{2.34}$$

such that

$$\mathbf{F}_i\left(\mathbf{X}(t), \mathbf{U}(t)\right) = \sum_{n=1}^{N_i} \beta_{in}\varphi_{in}(\mathbf{X}(t), \mathbf{U}(t)) := \boldsymbol{\Phi}_i^{\top}(t)\boldsymbol{\beta}_i, \tag{2.35}$$

where $\boldsymbol{\Phi}_i^{\top}(t) : \mathbb{R}^{\mathcal{N}_{\mathbf{x}}+\mathcal{N}_{\mathbf{u}}} \to \mathbb{R}^{N_i}$. It should be noted that some φ_{in} may be redundant in this representation, i.e. $\beta_{in} = 0$ accordingly. The set of $n_{\mathbf{x}}$ such description vectors

$$\{\boldsymbol{\beta}_i\}_{i=1,\ldots,n_{\mathbf{x}}} \tag{2.36}$$

encodes the representation of the full system in terms of the dictionary functions $\{\boldsymbol{\Phi}_i\}$.

Remark 4 *In Assumption 2, the basis function is assumed to be "linear in parameters". Though this covers a large class of system, it is still limited for some systems such as (Deep) Neural Networks. Thereafter, we next propose Assumption 3 which is more general than Assumption 2. However, throughout the thesis, the dynamical systems considered are assumed as stated in Assumption 2 except those of Chapter 7.1 of Part II and future work in*

14. This Assumption is applied in Chapter 7 where the likelihood with exponential family distribution.

Now we go beyond the "linear in parameters" restriction and propose the more general Assumption 2.

Assumption 3 *The function* $\mathbf{F}_i(\mathbf{X}(t), \mathbf{U}(t))$ *can be represented as linear combinations of dictionary functions:*

$$\boldsymbol{\varphi}_i(\mathbf{X}(t), \mathbf{U}(t), \boldsymbol{\beta}_i) = \left\{\varphi_{in} : \mathbb{R}^{n_{\mathbf{x}}+n_{\mathbf{u}}+n_{\beta_{in}}} \to \mathbb{R}\right\}_{n=1}^{N_i} \tag{2.37}$$

such that

$$\mathbf{F}_i\left(\mathbf{X}(t), \mathbf{U}(t)\right) = \sum_{n=1}^{N_i} \varphi_{in}(\mathbf{X}(t), \mathbf{U}(t), \boldsymbol{\beta}_{in}) \tag{2.38}$$

where $\boldsymbol{\beta}_{in} \in \mathbb{R}^{n_{\beta_{in}}}$*. The set of* $\sum_{i=1}^{n_{\mathbf{x}}} N_i$ *such description vectors*

$$\{\boldsymbol{\beta}_{in}\}_{i=1,\ldots,n_{\mathbf{x}},n=1,\ldots,N_i} \tag{2.39}$$

encodes the representation of the full system in terms of the dictionary functions $\{\boldsymbol{\varphi}_i\}$*.*

Next we discuss assumptions on the form of the diffusion term in (2.29).

Assumption 4 *Given* (2.28)*, the diffusion term in* (2.29) *can be expressed as*

$$\mathbf{G}_i\left(\mathbf{X}(t), \mathbf{U}(t)\right) \cdot \boldsymbol{\Xi}(t) = \sum_{k=1}^{n_{\boldsymbol{\xi}}} \sum_{\tau=0}^{v_k-1} g_{ik\tau}\left(\mathbf{X}(t), \mathbf{U}(t)\right) \xi_k(t-\tau) \tag{2.40}$$

where $g_{ik\tau} : \mathbb{R}^{\mathcal{N}_{\mathbf{x}}+\mathcal{N}_{\mathbf{u}}} \to \mathbb{R}$ *is an arbitrary bounded function, e.g., zero, monomial, etc. And* $\mathbf{G}_i$ *is bounded as well.*

Follow the linear combination representation assumption both in (2.35) and (2.38), another key assumption is on "sparse representation".

Assumption 5 *The sets in* (2.36) *and* (2.39) *are sparse (*$\boldsymbol{\beta}$ *has many zero elements), and the representation in* (2.35) *and* (2.38) *are sparse as well.*

Remark 5 (Assumption 5) *This assumption is concise but fundamental in this thesis. It is relevant to model selection criteria such as Akaike information criterion (AIC) [3] and Bayesian information criterion (BIC) [167].*

We make another key assumption on "full measurement"

Assumption 6 *The system* (2.29) *is fully measurable, i.e., all the state variables x_i can be measured and there are no hidden variables in the nonlinear system.*

Remark 6 (Assumption 6) *Typically, only part of the state is measured [224, 223], and, in particular, the number of hidden/unobservable nodes and their position in the network are usually unknown. Fortunately, this may be problematic only in **nonlinear** system identification but not in **linear** system identification. However, in economics and financial time series modelling [122, 203, 198], asset pricing modelling [38] and the seminal paper on Granger causality [68], the model only consist of the measured variables. Meanwhile, in (deep) neural network systems, the model also employs complicated nonlinear embedding from input to output.*

To incorporate prior knowledge into the identification problem, it is often important to be able to impose constraints on $\boldsymbol{\beta}$. In biological systems, positivity of the parameters constituting $\boldsymbol{\beta}$ is an example of such constraints. The other motivation for constrained optimisation comes from stability considerations. Typically, the underlying system is known *a priori* to be stable, especially if this system is a biological or physical system. A lot of stability conditions can be formulated as convex optimisation problems, e.g. Lyapunov stability conditions expressed as Linear Matrix Inequalities [26], Gershgorin Theorem for linear systems [89], etc. Only few contributions are available in the literature that address the problem of how to consider *a priori* information on system stability during system identification [34, 229]. To be able to integrate constraints on $\boldsymbol{\beta}$ into the problem formulation, we consider the following assumption on $\boldsymbol{\beta}$.

Assumption 7 *Constraints on the weights $\boldsymbol{\beta}$ can be described by a set of convex functions:*

$$\begin{aligned} H_i^{[I]}(\boldsymbol{\beta}) &\leq 0, \;\; i = 1, \ldots, m_I, \\ H_j^{[E]}(\boldsymbol{\beta}) &= 0, \;\; j = 1, \ldots, m_E. \end{aligned} \tag{2.41}$$

where the convex functions $H_i^{[I]} : \mathbb{R}^N \to \mathbb{R}$ are used to define inequality constraints, whereas the convex functions $H_j^{[E]} : \mathbb{R}^N \to \mathbb{R}$ are used to define equality constraints.

2.3.3 Linear Regression Model

We first start with a simple model where the next step states only depends on the current step states and/or inputs, i.e. discrete-time MIMO nonlinear systems with

additive noise:

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\xi}(t), \tag{2.42}$$

$$z_i(t) = x_i(t), \tag{2.43}$$

and

$$\mathbf{x} = [x_1, \ldots, x_{n_\mathbf{x}}]^\top \in \mathbb{R}^{n_\mathbf{x}}$$

denotes the state vector;

$$\mathbf{u} = [u_1, \ldots, u_{n_\mathbf{u}}]^\top \in \mathbb{R}^{n_\mathbf{u}}$$

denotes the input vector;

$$\mathbf{f}(\cdot) \triangleq [\mathbf{f}_1(\cdot), \ldots, \mathbf{f}_{n_\mathbf{x}}(\cdot)]^\top : \mathbb{R}^{n_\mathbf{x}+n_\mathbf{u}} \to \mathbb{R}^{n_\mathbf{x}}$$

denotes the nonlinear functions;

$$\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_{n_\mathbf{x}}]^\top \in \mathbb{R}^{n_\mathbf{x}}$$

denotes the dynamic noise vector, where $\xi_i(t)$ is assumed to be Gaussian i.i.d. :

$$\xi_i(t) \sim \mathcal{N}(0, \sigma_i^2)$$

with

$$\begin{aligned}
&\mathbb{E}(\xi_i(p)) = 0, \\
&\mathbb{E}(\xi_i(p)\xi_i(q)) = \sigma_i^2 \delta_{pq}, \\
&\delta_{pq} = \begin{cases} 1, & p = q, \\ 0, & p \neq q. \end{cases}
\end{aligned}$$

Under Assumptions 2 and 6, the system in (2.42) can be rewritten into the linear regression form:

$$x_i(t+1) = \mathbf{f}_i^\top(\mathbf{x}(t), \mathbf{u}(t))\mathbf{v}_i + \xi_i(t),\ i = 1, \ldots, n_\mathbf{x}, \tag{2.44}$$

where

$$\mathbf{v}_i = [v_{i1}, \ldots, v_{iN_i}]^\top \in \mathbb{R}^{N_i}$$

is the weight vector to be identified and

$$\mathbf{f}_i^\top(\mathbf{x}(t),\mathbf{u}(t)) = [f_{i1}(\mathbf{x}(t),\mathbf{u}(t)),\ldots,f_{iN_i}(\mathbf{x}(t),\mathbf{u}(t))] \in \mathbb{R}^{N_i}$$

is the vector of considered dictionary functions (which does not contain unknown parameters).

If data samples satisfying (2.44) can be obtained from the system of interest. When data are sampled, we assume the data matrix and first derivative/difference data matrix satisfying (2.44) can be obtained as

$$\begin{bmatrix} x_1(1) & \ldots & x_{n_\mathbf{x}}(1) \\ \vdots & \ddots & \vdots \\ x_1(M) & \ldots & x_{n_\mathbf{x}}(M) \end{bmatrix} \tag{2.45}$$

and

$$\begin{bmatrix} x_1(2) & \ldots & x_{n_\mathbf{x}}(2) \\ \vdots & \ddots & \vdots \\ x_1(M+1) & \ldots & x_{n_\mathbf{x}}(M+1) \end{bmatrix} \tag{2.46}$$

respectively. Now we let

$$y_i(t) \triangleq x_i(t+1),$$

model (2.42) can be written as

$$\mathbf{y}_i = \mathbf{\Psi}_i\mathbf{v}_i + \mathbf{\Xi}_i,\ i = 1,\ldots,n_\mathbf{x}, \tag{2.47}$$

with

$$\begin{aligned} \mathbf{y}_i &\triangleq [y_i(1),\ldots,y_i(M)]^\top \in \mathbb{R}^{M\times 1}, \\ \mathbf{\Psi}_i &\triangleq \begin{bmatrix} f_{i1}(\mathbf{x}(1),\mathbf{u}(1)) & \ldots & f_{iN_i}(\mathbf{x}(1),\mathbf{u}(1)) \\ \vdots & \ddots & \vdots \\ f_{i1}(\mathbf{x}(M),\mathbf{u}(M)) & \ldots & f_{iN_i}(\mathbf{x}(M),\mathbf{u}(M)) \end{bmatrix} \in \mathbb{R}^{M\times N_i}, \\ \mathbf{v}_i &\triangleq [v_{i1},\ldots,v_{iN_i}]^T \in \mathbb{R}^{N_i\times 1}, \\ \mathbf{\Xi}_i &\triangleq [\xi_i(1),\ldots,\xi_i(M)]^\top \in \mathbb{R}^{M\times 1}. \end{aligned} \tag{2.48}$$

We want to find $\mathbf{v}_i$ given the measured noisy data stored in $\mathbf{y}_i$. This is a typical linear regression problem that can be solved using standard least square approaches, provided that the structure of the nonlinearities in the model are known, i.e., provided that $\mathbf{\Psi}_i$ is known.

In what follows we gather in a matrix $\mathbf{X}_i$ similar to $\mathbf{\Psi}_i$ the set of *all* candidate/possible basis functions. This is an equivalent augmentation of (2.44) by introducing "redundant" terms with corresponding parameters be zero in the representation. Due to such expansion of the dictionary matrix, $\mathbf{v}_i$ will be substituted by a new parameter vector $\boldsymbol{\beta}_i$. The issues on the selection of basis functions will be discussed later in Section 5.2. Now we consider:

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta}_i + \mathbf{\Xi}_i,\ i = 1, \ldots, n_{\mathbf{x}}. \tag{2.49}$$

The solution $\hat{\boldsymbol{\beta}}_i$ to (2.49) is typically going to be sparse due to the potential introduction of non-relevant and/or non-independent dictionary functions in $\mathbf{X}_i$.

Example 2 *As an illustrative example, we consider the following model of polynomial terms for a single-input single-output (SISO) nonlinear autoregressive system with exogenous input (NARX model) [112]:*

$$x(t+1) = 0.7x^5(t)x(t-1) - 0.5x(t-2) + 0.6u^4(t-2) - 0.7x(t-2)u^2(t-1) + \xi(t), \tag{2.50}$$

with $x, u, \xi \in \mathbb{R}$. Then we can write an extended form

$$\begin{aligned} x(t+1) &= w_1 + w_2x(t) + \ldots + w_{m_x+2}x(t-m_x) + \ldots + w_Nx^{d_x}(t-m_x)u^{d_u}(t-m_u) + \xi(t) \\ &= \boldsymbol{\beta}^\top\mathbf{f}(x(t), \ldots, x(t-m_x), u(t), \ldots, u(t-m_u)) + \xi(t), \end{aligned}$$

where d_x (resp. d_u) is the degree of the output (resp. input); m_x (resp. m_u) is the maximal memory order of the output (resp. input); $\boldsymbol{\beta}^\top = [w_1, \ldots, w_N] \in \mathbb{R}^N$ is the weight vector; and

$$\mathbf{f}(\mathbf{x}(t), \ldots, \mathbf{x}(t-m_{\mathbf{x}}), \mathbf{u}(t), \ldots, \mathbf{u}(t-m_{\mathbf{u}})) = [f_1(\cdot), \ldots, f_N(\cdot)]^\top \in \mathbb{R}^N$$

is the dictionary functions vector. Then for the NARX model (2.50), we can find $d_x = 5$, $d_u = 4$, $m_x = 2$, $m_u = 2$. To define the dictionary matrix, we consider all possible monomials up to degree $d_x = 5$ (resp. $d_u = 4$) and up to memory order $m_x = 5$ (resp. $m_u = 2$) in x (resp. u). which gives $\mathbf{f}(\cdot) \in \mathbb{R}^{1960}$ thus $\boldsymbol{\beta} \in \mathbb{R}^{1960}$. Since $\mathbf{v} \in \mathbb{R}^4$, only 4 out of the 1960 associated weights w_i are nonzero.

Since the $n_{\mathbf{x}}$ linear regression problems in (2.49) are independent, for simplicity of notation, we omit the subscript i in (2.49) and write

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{\Xi}. \tag{2.51}$$

As indicated in Example 2, a large number of candidate basis functions can be considered to construct $\mathbf{X}$. This means that the number of columns N of $\mathbf{X}$ tends to be very large. Meanwhile, the time-series measurements collected are quite limited. To find the sparsest solution is desirable but NP-hard and numerically unstable.

2.3.4 Additional Experiment Designs

During the identification process, additional experiments can be performed to make the data set and the data-dependent dictionary matrix more informative for identification (see Sec. 13 of [117]). To ease notation, we assume the same amount of measurements are sampled for each new experiment. The equation with superscript $[0]$ denotes the initial experiment. For simplicity, we consider in particular on the discrete-time dynamical system

$$x_i(t+1) = \mathbf{f}_i^\top(\mathbf{x}(t), \mathbf{u}(t))\mathbf{v}_i + \xi_i(t),\ i = 1, \ldots, n_{\mathbf{x}}, \tag{2.52}$$

Additional Excitation Signals Applied to the Inputs of the System Suppose there are n_d different excitation signals $\mathbf{u}^{[k]}$, $k = 0, \ldots, n_d$, applied to the inputs of the system. We can then expand (2.52) as:

$$\begin{cases} x_i^{[0]}(t+1) = \mathbf{f}_i^\top(\mathbf{x}^{[0]}(t), \mathbf{u}^{[0]}(t))\mathbf{v}_i + \xi_i^{[0]}(t), \\ x_i^{[1]}(t+1) = \mathbf{f}_i^\top(\mathbf{x}^{[1]}(t), \mathbf{u}^{[1]}(t))\mathbf{v}_i + \xi_i^{[1]}(t), \\ \qquad\qquad \vdots \qquad\qquad \vdots \\ x_i^{[n_d]}(t+1) = \mathbf{f}_i^\top(\mathbf{x}^{[n_d]}(t), \mathbf{u}^{[n_d]}(t))\mathbf{v}_i + \xi_i^{[n_d]}(t). \end{cases} \tag{2.53}$$

We stack all input and output data together and define

$$\begin{aligned} \mathbf{y}_i &\triangleq \begin{bmatrix} \mathbf{y}_i^{[0]} \\ \mathbf{y}_i^{[1]} \\ \vdots \\ \mathbf{y}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times 1}, \quad \boldsymbol{\Psi}_i \triangleq \begin{bmatrix} \boldsymbol{\Psi}_i^{[0]} \\ \boldsymbol{\Psi}_i^{[1]} \\ \vdots \\ \boldsymbol{\Psi}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times N_i}, \\ \mathbf{v}_i &\triangleq \begin{bmatrix} v_{i1} \\ \vdots \\ v_{iN_i} \end{bmatrix} \in \mathbb{R}^{N_i\times 1}, \qquad \boldsymbol{\Xi}_i \triangleq \begin{bmatrix} \boldsymbol{\Xi}_i^{[0]} \\ \boldsymbol{\Xi}_i^{[1]} \\ \vdots \\ \boldsymbol{\Xi}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times 1}, \end{aligned} \tag{2.54}$$

where

$$\begin{aligned}
\mathbf{y}_i^{[j]} &\triangleq \left[x_i^{[j]}(1), \ldots, x_i^{[j]}(M)\right]^T \in \mathbb{R}^{M\times 1},\\
\mathbf{\Psi}_i^{[j]} &\triangleq \begin{bmatrix} f_{i1}(\mathbf{x}^{[j]}(0), \mathbf{u}^{[j]}(0)) & \ldots & f_{iN_i}(\mathbf{x}^{[j]}(0), \mathbf{u}^{[j]}(0)) \\ \vdots & \vdots & \vdots \\ f_{i1}(\mathbf{x}^{[j]}(M-1), \mathbf{u}^{[j]}(M-1)) & \ldots & f_{iN_i}(\mathbf{x}^{[j]}(M-1), \mathbf{u}^{[j]}(M-1)) \end{bmatrix}\\
&\quad \in \mathbb{R}^{M\times N_i},\\
\mathbf{\Xi}_i^{[j]} &\triangleq \left[\xi_i^{[j]}(0), \ldots, \xi_i^{[j]}(M-1)\right]^T \in \mathbb{R}^{M\times 1}.
\end{aligned}$$

Using the notation above, system (2.53) can be written as $\mathbf{y}_i = \mathbf{\Psi}_i \mathbf{v}_i + \mathbf{\Xi}_i,\ i = 1, \ldots, n_{\mathbf{x}}$, which has the same form as (2.47).

Perturbation Experiments This case refers to perturbations of the weights $\mathbf{v}_i$ that leads to variations in the outputs of the system.[1] In this situation, we can write the following equations

$$\begin{cases}
x_i^{[0]}(t+1) = \mathbf{f}_i^\top(\mathbf{x}^{[0]}(t), \mathbf{u}(t))\mathbf{v}_i + \xi_i^{[0]}(t),\\
x_i^{[1]}(t+1) = \mathbf{f}_i^\top(\mathbf{x}^{[1]}(t), \mathbf{u}(t))(\mathbf{v}_i + \Delta\mathbf{v}_i^{[1]}) + \xi_i^{[1]}(t),\\
\qquad\vdots \qquad\qquad\vdots\\
x_i^{[n_d]}(t+1) = \mathbf{f}_i^\top(\mathbf{x}^{[n_d]}(t), \mathbf{u}(t))(\mathbf{v}_i + \Delta\mathbf{v}_i^{[n_d]}) + \xi_i^{[n_d]}(t).
\end{cases} \tag{2.55}$$

Similar to the formulation for additional excitation signals applied to the inputs in (2.54), we stack all input and output data and define

$$\begin{aligned}
\mathbf{y}_i &\triangleq \begin{bmatrix} \mathbf{y}_i^{[0]} \\ \mathbf{y}_i^{[1]} \\ \vdots \\ \mathbf{y}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times 1}, \quad \mathbf{\Psi}_i \triangleq \begin{bmatrix} \mathbf{\Psi}_i^{[0]} & \mathbf{0}_{M\times N_i} & \ldots & \mathbf{0}_{M\times N_i} \\ \mathbf{\Psi}_i^{[1]} & \mathbf{\Psi}_i^{[1]} & \ldots & \mathbf{0}_{M\times N_i} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Psi}_i^{[n_d]} & \mathbf{0}_{M\times N_i} & \ldots & \mathbf{\Psi}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times N_i},\\
\mathbf{v}_i &\triangleq \begin{bmatrix} \mathbf{v}_i^{[0]} \\ \mathbf{\Delta v}_i^{[1]} \\ \vdots \\ \mathbf{\Delta v}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{N_i\times 1}, \qquad \mathbf{\Xi}_i \triangleq \begin{bmatrix} \mathbf{\Xi}_i^{[0]} \\ \mathbf{\Xi}_i^{[1]} \\ \vdots \\ \mathbf{\Xi}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times 1},
\end{aligned} \tag{2.56}$$

[1] In molecular biology experiments, this corresponds to the situation where the expression of one or more of an organism's genes is reduced through genetic manipulations.

where

$$
\begin{aligned}
\mathbf{y}_i^{[j]} &\triangleq \left[x_i^{[j]}(1), \ldots, x_i^{[j]}(M)\right]^T \in \mathbb{R}^{M\times 1}, \\
\mathbf{\Psi}_i^{[j]} &\triangleq \begin{bmatrix} f_{i1}(\mathbf{x}^{[j]}(0), \mathbf{u}(0)) & \ldots & f_{iN_i}(\mathbf{x}^{[j]}(0), \mathbf{u}(0)) \\ \vdots & \vdots & \vdots \\ f_{i1}(\mathbf{x}^{[j]}(M-1), \mathbf{u}(M-1)) & \ldots & f_{iN_i}(\mathbf{x}^{[j]}(M-1), \mathbf{u}(M-1)) \end{bmatrix} \in \mathbb{R}^{M\times N_i}, \\
\mathbf{\Delta v}_i^{[j]} &\triangleq \left[\Delta v_{i1}^{[j]}, \ldots, \Delta v_{iN_i}^{[j]}\right]^\top \in \mathbb{R}^{N_i\times 1}, \\
\mathbf{\Xi}_i^{[j]} &\triangleq \left[\xi_i^{[j]}(0), \ldots, \xi_i^{[j]}(M-1)\right]^T \in \mathbb{R}^{M\times 1},
\end{aligned}
$$

and let $\mathbf{\Delta v}_i^{[0]} = \mathbf{0}_{N_i}$. Using the notation above, we can again write the formulation as $\mathbf{y}_i = \mathbf{\Psi}_i \mathbf{v}_i + \mathbf{\Xi}_i,\ i = 1, \ldots, n_{\mathbf{x}}$, which is similar to (2.47).

Knockout Experiment The term "Knockout" comes from "gene knockout" in molecular biology when certain organism's genes are knocked out of the organism. If one state variable x_j is knocked out, all the dictionary functions that involved x_j will be excluded from the dictionary matrix or set to zero. The model structure and connections among other state variables will remain unchanged. This also applies to the deletion of multiple state variables. Recall the definition of the dictionary function vector: $\mathbf{f}_i(\mathbf{x}(t), \mathbf{u}(t)) = [f_{i1}(\mathbf{x}(t), \mathbf{u}(t)), \ldots, f_{iN_i}(\mathbf{x}(t), \mathbf{u}(t))]^\top \in \mathbb{R}^{N_i}$, and further define the subset index $I_j \subset \{1, 2, \ldots, n\}$ containing the indices of the knocked out variables during knockout experiment j. $\mathbf{f}_i^{[I_j^-]}(\cdot)$ denotes the vector/matrix formed by the elements/columns of $\mathbf{f}_i(\cdot)$ with indices in I_j set to zeros. The data collected through knockout experiments can then be seen to satisfy the following equations

$$
\begin{cases}
x_i^{[0]}(t+1) = {\mathbf{f}_i^{[I_0^-]}}^\top(\mathbf{x}^{[0]}(t), \mathbf{u}^{[0]}(t))\mathbf{v}_i + \xi_i^{[0]}(t), \\
x_i^{[1]}(t+1) = {\mathbf{f}_i^{[I_1^-]}}^\top(\mathbf{x}^{[1]}(t), \mathbf{u}^{[1]}(t))\mathbf{v}_i + \xi_i^{[1]}(t), \\
\qquad\vdots \qquad\qquad\qquad \vdots \\
x_i^{[n_d]}(t+1) = {\mathbf{f}_i^{[I_{n_d}^-]}}^\top(\mathbf{x}^{[n_d]}(t), \mathbf{u}^{[n_d]}(t))\mathbf{v}_i + \xi_i^{[n_d]}(t).
\end{cases}
\tag{2.57}
$$

Similar to the formulation for additional excitation signals in (2.54) and perturbation experiments in (2.56), we stack all input and output data and define

$$\mathbf{y}_i \triangleq \begin{bmatrix} \mathbf{y}_i^{[0]} \\ \mathbf{y}_i^{[1]} \\ \vdots \\ \mathbf{y}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times 1}, \quad \mathbf{\Psi}_i \triangleq \begin{bmatrix} \mathbf{\Psi}_i^{[I_0^-]} \\ \mathbf{\Psi}_i^{[I_1^-]} \\ \vdots \\ \mathbf{\Psi}_i^{[I_{n_d}^-]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times N_i},$$

$$\mathbf{v}_i \triangleq \begin{bmatrix} v_{i1} \\ \vdots \\ v_{iN_i} \end{bmatrix} \in \mathbb{R}^{N_i\times 1}, \qquad \mathbf{\Xi}_i \triangleq \begin{bmatrix} \mathbf{\Xi}_i^{[0]} \\ \mathbf{\Xi}_i^{[1]} \\ \vdots \\ \mathbf{\Xi}_i^{[n_d]} \end{bmatrix} \in \mathbb{R}^{M\cdot(n_d+1)\times 1}, \tag{2.58}$$

where

$$\begin{aligned}
\mathbf{y}_i^{[j]} &\triangleq \left[x_i^{[j]}(1), \ldots, x_i^{[j]}(M)\right]^T \in \mathbb{R}^{M\times 1}, \\
\mathbf{\Psi}_i^{[I_j^-]} &\triangleq \begin{bmatrix} f_{i1}(\mathbf{x}^{[j]}(0), \mathbf{u}^{[j]}(0)) & \ldots & f_{iN_i}(\mathbf{x}^{[j]}(0), \mathbf{u}^{[j]}(0)) \\ \vdots & \vdots & \vdots \\ f_{i1}(\mathbf{x}^{[j]}(M-1), \mathbf{u}^{[j]}(M-1)) & \ldots & f_{iN_i}(\mathbf{x}^{[j]}(M-1), \mathbf{u}^{[j]}(M-1)) \end{bmatrix} \\
&\quad \in \mathbb{R}^{M\times N_i}, \\
\mathbf{\Xi}_i^{[j]} &\triangleq \left[\xi_i^{[j]}(0), \ldots, \xi_i^{[j]}(M-1)\right]^T \in \mathbb{R}^{M\times 1}.
\end{aligned}$$

It should be noted that certain columns of $\mathbf{\Psi}_i^{[I_j^-]}$ indexed by I_j would be zero columns. Again, using the above notation, we obtain the formulation $\mathbf{y}_i = \mathbf{\Psi}_i\mathbf{v}_i + \mathbf{\Xi}_i,\ i = 1, \ldots, n_{\mathbf{x}}$, which has the same form as (2.47).

Remark 7 *From the above analysis, we can re-formulate the data coming from the additional experiments (replicates, perturbation, knock-out) to the standard form $\mathbf{y}_i = \mathbf{\Psi}_i\mathbf{v}_i + \mathbf{\Xi}_i$ (with different matrix dimensions). In the next Section, we shall propose a solution to such problems.*

2.4 Linear Regression Problem

2.4.1 Regression Problem Statement

Since the $n_{\mathbf{x}}$ linear regression models in Eq. (2.49) are independent, for simplicity of notation, we omit the subscript i used to index the state variable and simply write:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\Xi}. \tag{2.59}$$

We assume the stochastic term $\boldsymbol{\Xi}$ is Gaussian i.i.d. Further discussion on the distribution on $\boldsymbol{\Xi}$ can be found in Section 5.4. The identification problem can be formally stated as follows:

Problem 2 (Consistency) *Given* $\mathbf{y}$ *and* $\mathbf{X}$ *over the interval* $[1, M]$*, find a coefficient vector* $\boldsymbol{\beta}$ *such that* (2.51) *holds for all* $t \in [1, M]$*.*

Since the stochastic term $\boldsymbol{\Xi}$ is under the Gaussian i.i.d assumption, the basic consistency result is almost immediate under quadratic criteria, i.e. using ordinary least square (OLS)

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2. \tag{2.60}$$

However, it is clear that this problem is not well-posed and has infinitely many solutions. For instance, one can always find a trivial model with a large order or/and a large number of (nonlinear) basis functions that perfectly fits the data, which is known to be *overfitting*. To avoid the overfitting issue, $\mathbf{f}(\mathbf{x}(t))$ is usually favoured to be sparsely represented . Then Problem 2 can be modified with the following formal statement:

Problem 3 (A minimum number of basis functions) *Given* $\mathbf{y}$ *and* $\mathbf{X}$ *over the interval* $[1, M]$ *and* a prior *selection of a set of* N *dictionary functions, find a coefficient vector* $\boldsymbol{\beta}$ *with a minimum number of nonzero entries such that* (2.51) *holds for all* $t \in [1, M]$*.*

2.4.2 Nonconvex Optimisation Problem

To enforce a minimum number of nonzero entries in $\boldsymbol{\beta}$, we can arrive at the following famous *sparse signal recovery* problem as we discussed in Section 1.3 by adding the penalty term $\lambda\|\boldsymbol{\beta}\|_{\ell_0}$

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_{\ell_0}, \tag{2.61}$$

where λ is the regularisation parameter.

2.4.3 Convex Relaxation

Again, as we discussed in Chapter 1.3, we replace $\|\boldsymbol{\beta}\|_{\ell_0}$ in the optimisation above by $\|\boldsymbol{\beta}\|_{\ell_1}$. The idea behind this relaxation is the fact that the ℓ_1 norm is the convex envelope of the ℓ_0 norm, and thus, in a sense, minimising the former yields the best convex relaxation to the (non-convex) problem of minimizing the latter. It gives

$$\min_{\beta} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_{\ell_1}. \tag{2.62}$$

Remark 8 *We notice that there is a hyperparameter λ which needs to be specified a priori or tuned manually. λ is known as regularisation parameter. The selection of optimal regularisation parameter is an open problem in machine leaning and statistics and has been discussed in any machine learning textbook. Without otherwise specified, we use cross-validation for choice of the optimal λ [22, 79, 127].*

Chapter 3

Nonlinear Dynamical System with Heterogeneous Datasets

3.1 Introduction

The problem of identifying biological networks from experimental time series data is of fundamental interest in systems and synthetic biology. For example, such information can aid in the design of drugs or of synthetic biology genetic controllers. Tools from system identification [117] can be applied for such purposes. However, most system identification methods produce estimates of model structures based on data coming from a single experiment.

The interest in identification methods able to handle several datasets simultaneously is twofold. Firstly, with the increasing availability of "big data" obtained from sophisticated biological instruments, e.g. large 'omics' datasets, attention has turned to the efficient and effective integration of these data and to the maximum extraction of information from them. Such datasets can contain (a) data from replicates of an experiment performed on a biological system of interest under identical experimental conditions; (b) data measured from a biochemical network subjected to different experimental conditions, for example, different biological inducers, temperature, stress, optical input, gene knock-out and over-expression, etc. The challenges for simultaneously considering heterogeneous datasets during system identification are: (a) the system itself is unknown, i.e. neither the structure nor the corresponding parameters are known; (b) it is unclear how heterogeneous datasets collected under different experimental conditions influence the "quality" of the identified system.

Secondly, in control or synthetic biology applications the systems to be controlled typically need to be modelled first. Highly detailed or complex models are typically difficult to handle using rigorous control design methods. Therefore, one typically prefers to use simple or sparse models that capture at best the dynamics expressed in the collected data. The identification and use of simple or sparse models inevitably introduces model class uncertainties and parameter uncertainties [98, 201]. To assess these uncertainties replicates of multiple experiments is typically necessary.

Our approach is based on the concept of sparse Bayesian learning [191, 214] and on the definition of a unified optimisation problem allowing the consideration of different parameter values for different experimental conditions, and whose solution is a model consistent with all datasets available for identification. The ability to consider various datasets simultaneously can potentially avoid non-identifiability issues arising when a single dataset is used [93]. Furthermore, by comparing the identified parameter values associated with different conditions, we can pinpoint the influence specific experimental changes have on system parameters. In should

be noted that there are similarities between our approach to nonlinear random effects models and recent work in the area of generalized linear models (GLIM) for longitudinal data (e.g., Stiratelli et al., 1984; Liang and Zeger, 1986).

In this Chapter, we will extend the regression model in Chapter 2, to deal with multiple datasets. In Section 3.2, we will give the regression problem formulation for the nonlinear SYSID problem from *multiple* experiments. The regression problem is further formulated as a ℓ_0 type optimisation problem in Section 3.3.2 and a tentative empirical ℓ_1 relaxation is proposed in Section 3.3.3. The algorithms for this Chapter will be provided in Section 6.7 of Chapter 6.

3.2 Linear Regression Model

In Section 2.3.3, we introduced the linear regression model in (2.49) for single dataset modelling. Suppose a total number of C datasets are collected from C independent experiments, we put a subscript $[c]$ to index the identification problem associated with the specific dataset obtained from experiment $[c]$. The linear regression problem (2.49) for single dataset can be rewritten as

$$\mathbf{y}^{[c]} = \mathbf{X}^{[c]}\boldsymbol{\beta}^{[c]} + \boldsymbol{\xi}^{[c]}, c = 1, \ldots, C, \tag{3.1}$$

in which,

$$\begin{aligned}
\mathbf{X}^{[c]} &\triangleq \left[\mathbf{X}^{[c]}_{:,1}, \ldots, \mathbf{X}^{[c]}_{:,N}\right] \\
&= \begin{bmatrix} f_1(\mathbf{x}^{[c]}(1)) & \ldots & f_N(\mathbf{x}^{[c]}(1)) \\ \vdots & & \vdots \\ f_1(\mathbf{x}^{[c]}(M^{[c]})) & \ldots & f_N(\mathbf{x}^{[c]}(M^{[c]})) \end{bmatrix} \\
&\in \mathbb{R}^{M^{[c]} \times N}, \\
\boldsymbol{\beta}^{[c]} &\triangleq \left[w_1^{[c]}, \ldots, w_N^{[c]}\right]^\top \in \mathbb{R}^N, \\
\boldsymbol{\xi}^{[c]} &\triangleq \left[\xi^{[c]}(1), \ldots, \xi^{[c]}(M^{[c]})\right]^\top \in \mathbb{R}^{M^{[c]}},
\end{aligned} \tag{3.2}$$

where $\mathbf{x}^{[c]}(t) = \left[x_1^{[c]}(t), \ldots, x_{n_\mathbf{x}}^{[c]}(t)\right] \in \mathbb{R}^{n_\mathbf{x}}$ is the state vector at time instant t. It should be noted that N, the number of basis functions or number of columns of the dictionary matrix $\mathbf{X}^{[c]} \in \mathbb{R}^{M^{[c]} \times N}$, can be very large. Without loss of generality, we assume $M^{[1]} = \cdots = M^{[C]} = M$. The solution to $\boldsymbol{\beta}^{[c]}$ to (3.1) is typically going to be sparse, which is mainly due to the potential introduction of non-relevant and/or non-independent basis functions in $\mathbf{X}^{[c]}$.

$$
\begin{aligned}
\begin{bmatrix} \mathbf{y}^{[1]} \\ \vdots \\ \mathbf{y}^{[C]} \end{bmatrix} &= \underbrace{\left[\begin{array}{ccc|c|ccc} \mathbf{X}_{:,1}^{[1]} & \dots & \mathbf{X}_{:,N}^{[1]} & & & & \\ & & & \ddots & & & \\ & & & & \mathbf{X}_{:,1}^{[C]} & \dots & \mathbf{X}_{:,N}^{[C]} \end{array}\right]}_{C \textbf{ Blocks}} \left[\begin{array}{c} \boldsymbol{\beta}^{[1]} \\ \hline \vdots \\ \hline \boldsymbol{\beta}^{[C]} \end{array}\right] + \left[\begin{array}{c} \boldsymbol{\xi}^{[1]} \\ \hline \vdots \\ \hline \boldsymbol{\xi}^{[C]} \end{array}\right] \\
&= \underbrace{\left[\begin{array}{ccc|c|ccc} \mathbf{X}_{:,1}^{[1]} & & & & \mathbf{X}_{:,N}^{[1]} & & \\ & \ddots & & \ddots & & \ddots & \\ & & \mathbf{X}_{:,1}^{[C]} & & & & \mathbf{X}_{:,N}^{[C]} \end{array}\right]}_{N \textbf{ Blocks}} \left[\begin{array}{c} w_1^{[1]} \\ \vdots \\ \hline w_1^{[C]} \\ \hline \vdots \\ \hline w_N^{[1]} \\ \vdots \\ w_N^{[C]} \end{array}\right] + \left[\begin{array}{c} \boldsymbol{\xi}^{[1]} \\ \hline \vdots \\ \hline \boldsymbol{\xi}^{[C]} \end{array}\right] \\
&= \left[\begin{array}{c|c|c} \mathbf{X}_1 & \cdots & \mathbf{X}_N \end{array}\right] \left[\begin{array}{c} \boldsymbol{\beta}_1 \\ \hline \vdots \\ \hline \boldsymbol{\beta}_N \end{array}\right] + \left[\begin{array}{c} \boldsymbol{\xi}^{[1]} \\ \hline \vdots \\ \hline \boldsymbol{\xi}^{[C]} \end{array}\right].
\end{aligned}
\tag{3.3}
$$

To ensure reproducibility, experimentalists repeat their experiments under the same conditions, and the collected data are then called "replicates". Typically, only the average value over these replicates is used for modelling or identification purposes In this case, however, only the first moment is used and information provided by higher order moments is lost. Moreover, when data originate from different experimental conditions, it is usually very hard to combine the datasets into a single identification problem. This Section will address these issues by showing how several datasets can be combined to define a unified optimisation problem whose solution is an identified model consistent with the various datasets available for identification.

To consider heterogeneous datasets in one single formulation, we stack the various equations in (3.1) (see Eq. (3.3)). Each stacked equation in Eq. (3.3) corresponds to a replicate or an experiment performed by changing the experimental conditions on the same system.

In Eq. (3.3), for $n = 1, \dots, N$,

$$
\mathbf{X}_n = \text{blkdiag}[\mathbf{X}_{:,n}^{[1]}, \dots, \mathbf{X}_{:,n}^{[C]}], \quad \boldsymbol{\beta}_n = [w_n^{[1]}, \dots, w_n^{[C]}]^\top.
$$

Based on the stacked formulation given in Eq. (3.3) we further define

$$\begin{aligned} \mathbf{y} &= \begin{bmatrix} \mathbf{y}^{[1]} \\ \vdots \\ \mathbf{y}^{[C]} \end{bmatrix}, \mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \,\middle|\, \cdots \,\middle|\, \mathbf{X}_N \end{bmatrix}, \\ \boldsymbol{\beta} &= \left[\begin{array}{c} \boldsymbol{\beta}_1 \\ \hline \vdots \\ \hline \boldsymbol{\beta}_N \end{array}\right], \boldsymbol{\xi} = \left[\begin{array}{c} \boldsymbol{\xi}^{[1]} \\ \hline \vdots \\ \hline \boldsymbol{\xi}^{[C]} \end{array}\right], \end{aligned} \tag{3.4}$$

which gives

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\xi}. \tag{3.5}$$

This yields a formulation very similar to that presented previously for a single linear regression problem. However, in the multi-experiment formulation (9.8), there is now a special block structure for $\mathbf{y}$, $\mathbf{X}$ and $\boldsymbol{\beta}$.

Remark 9 *When $\boldsymbol{\beta}^{[c]}$ is fixed to be $\boldsymbol{\beta}$ for all the experiments, i.e. $\boldsymbol{\beta}^{[1]} = \cdots = \boldsymbol{\beta}^{[C]} = \boldsymbol{\beta}$, we can formulate the identification problem as a single linear regression problem by concatenation:*

$$\begin{bmatrix} \mathbf{y}^{[1]} \\ \vdots \\ \mathbf{y}^{[C]} \end{bmatrix} = \begin{bmatrix} \mathbf{X}^{[1]} \\ \vdots \\ \mathbf{X}^{[C]} \end{bmatrix} \boldsymbol{\beta} + \begin{bmatrix} \boldsymbol{\xi}^{[1]} \\ \vdots \\ \boldsymbol{\xi}^{[C]} \end{bmatrix}. \tag{3.6}$$

3.3 Linear Regression Problem

3.3.1 Regression Problem Statement

Since the $n_{\mathbf{x}}$ linear regression models in Eq. (3.3) are independent, for simplicity of notation, we omit the subscript i used to index the state variable and simply write:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\Xi}. \tag{3.7}$$

We assume the stochastic term $\boldsymbol{\Xi}$ is Gaussian i.i.d. Further discussion on the distribution on $\boldsymbol{\Xi}$ can be found in Section 5.4. The identification problem can be formally stated as follows which is similar to the one in Section 2.4:

Problem 4 (Consistency) *Given $\mathbf{y}$ and $\mathbf{X}$ over the interval $[1, M]$, find a coefficient vector $\boldsymbol{\beta}$ such that* (3.3) *holds for all $t \in [1, M]$.*

Since the stochastic term $\boldsymbol{\Xi}$ is under the Gaussian i.i.d assumption, the basic consistency result is almost immediate under quadratic criteria, i.e. using ordinary least square (OLS)

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2. \tag{3.8}$$

However, it is clear that this problem is not well-posed and has infinitely many solutions. For instance, one can always find a trivial model with a large order or/and a large number of (nonlinear) basis functions that perfectly fits the data, which is known to be *overfitting*. The "sparse" idea will be again used to avoid overfitting issue. However, a slightly differently from the strategy proposed in Section 2.4, the Problem 4 can be modified with the following formal statement:

Problem 5 (A minimum number of basis functions) *Given* $\mathbf{y}$ *and* $\mathbf{X}$ *over the interval* $[1, M]$ *and a* prior *selection of a set of* N *dictionary functions, find a coefficient vector* $\boldsymbol{\beta}$ *with a minimum number of basis functions such that* (3.3) *holds for all* $t \in [1, M]$.

3.3.2 Nonconvex Optimisation Problem

To achieve a minimum number of basis functions, we will try to find a coefficient vector $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N]^\top$ with a minimum number of nonzero blocks $\boldsymbol{\beta}_i$, $\forall i = 1, \ldots, N$ such that (3.3) holds for all $t \in [1, M]$. We can arrive at the following regularised regression problem

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{n=1}^{N} \|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0}, \tag{3.9}$$

where λ is the regularisation parameter.

The penalty term $\|\|\boldsymbol{\beta}_i\|_{\ell_2}\|_{\ell_0}$ seems a bit complicated, we will give some explanations. Since $\boldsymbol{\beta}_n = [w_i^{[1]}, \ldots, w_n^{[C]}]^\top$, for $n = 1, \ldots, N$, we have

$$\|\boldsymbol{\beta}_n\|_{\ell_2} = \sqrt{(w_n^{[1]})^2 + \ldots + (w_n^{[C]})^2}, \tag{3.10}$$

it means $\|\boldsymbol{\beta}_i\|_{\ell_2} = 0$ if and only if $w_i^{[c]} = 0$, $\forall c = 1, \ldots, C$. Therefore, $\sum_{i=1}^{N} \|\|\boldsymbol{\beta}_i\|_{\ell_2}\|_{\ell_0}$ counts the number of basis functions.

3.3.3 Convex Relaxation

Again, similar as we discussed in Section 2.4.3, we replace $\|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0}$ in the optimisation above by $\|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_1} = \|\boldsymbol{\beta}_i\|_{\ell_2}$. The idea behind this relaxation is the fact that the ℓ_1 norm is the convex envelope of the ℓ_0 norm, and thus, in a sense, minimising the former yields the best convex relaxation to the (non-convex) problem of minimizing the latter. It gives

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{n=1}^{N} \|\boldsymbol{\beta}_n\|_{\ell_2}, \tag{3.11}$$

This minimisation problem is known as Group Lasso.

Chapter 4

Time-Varying Dynamical System

4.1 Introduction

Identification of switched systems, which are characterised by the interaction of both continuous and discrete dynamics, is widely used in many different fields such as systems/synthetic biology, econometrics, finance, biochemical engineering, social networks, etc. In this Section, we are interested in the identification of regime switch dynamical systems. Biochemical processes can go through different phases in time; for example, a cell cycle in bacteria or diurnal alternations in plants. These switches are typically triggered by time dependent processes or by some external force. Therefore, the dynamics of biochemical reactions can be modelled as a collection of submodels amongst which switches occur over time. For biochemical reaction networks, the submodels are typically nonlinear due to mass action kinetics.

In classical switched system identification, the submodels are typically assumed to be linear or of the switch affine type [144], which is often used to approximate nonlinear dynamics. In [132], the structure of submodels is fixed and a minimal number of switches between submodels is inferred. However, these techniques are not generally applicable to biochemical kinetics due to highly nonlinear terms and model complexity. In the nonlinear case, there is an additional problem of nonlinear basis selection, which is fixed and predefined in the linear case. Unlike the linear case, the number of nonlinear basis functions can be infinite and one might have to use complicated nonlinear functions to model the dynamics without any switches. In practice, if one is interested in obtaining the least number of switches, the number of nonlinear basis functions will typically grow, and vice versa, a small number of nonlinear basis functions will result in many switches. Hence there are two different and competing minimisation criteria: the number of switches between submodels and the number of basis functions in each submodel.

In the first part of this Chapter, we will extend the regression model in Chapter 2 and 3, to deal with time-varying systems. In Section 4.3, we will give the regression problem formulation for the nonlinear SYSID problem of time-varying systems. The regression problem is further formulated as a ℓ_0 type optimisation problem by minimising the number of parameters and the number of switches in Section 4.4.2. A tentative empirical ℓ_1 relaxation is proposed in Section 4.4.3. The algorithms for this Chapter will be provided in Section 6.8 of Chapter 6.

In the second part of the Chapter, we revisit the problem of trend filtering

4.2 Regime-Switch Dynamical System

4.2.1 Scalar Linear Regime-Switch Systems

Without loss of generality and to ease notations, we will consider switched autoregressive exogenous (SARX) hybrid affine models with single input and single output (SISO-SARX) of the form:

$$y(t) = \sum_{i=1}^{n_a} a_i(\alpha_t) y(t-i) + \sum_{i=1}^{n_b} b_i(\alpha_t) u(t-i) + f(\alpha_t) + \eta(t) \tag{4.1}$$

where u, y and η denote the input, output, and noise, respectively, and where $t \in [t_0, T]$. The discrete variable $\alpha_t \in \{1, \ldots, s\}$ - the mode of the system - indicates which of the *s submodels* is active at time t. The time instants where the value of α_t changes are called *discrete transitions* or *switches*. These switches partition the interval $[0, M]$ into a *discrete hybrid time set* [121], $\tau = \{I_i\}_{i=0}^{k}$, such that α_t is constant within each subinterval $I_i = [\tau_i, \tau_i']$ and different in consecutive intervals. In the sequel, we denote by τ_i and τ_i' the beginning and ending times of the i-th interval, respectively. Clearly, τ satisfies

- $\tau_0 = 0$[1], and $\tau_k' = M$,
- $\tau_i \leq \tau_i' = \tau_{i+1} - 1$,

and the number of switches is equal to k.

An equivalent representation of is in the following linear regression form

$$y(t) = \phi^\top(t)\boldsymbol{\theta}(\alpha_t) + \eta(t) \tag{4.2}$$

where the regressor vector is

$$\phi^\top = \begin{bmatrix} y(t-1), & \ldots, & y(t-n_a), & u(t-1), & \ldots, & u(t-n_b) & 1 \end{bmatrix}$$

and the unknown coefficient vector at time t is

$$\boldsymbol{\theta}(\alpha_t) = \begin{bmatrix} a_1(\alpha_t), & \ldots, & a_{n_a}(\alpha_t), & b_1(\alpha_t), & \ldots, & b_{n_b}(\alpha_t), & f(\alpha_t) \end{bmatrix}$$

[1]Since it is not possible to deduce information for $t < \max(n_a, n_b)$ when the initial condition are unknown, in the identification problem we take $t_0 = \max(n_a, n_b)$

4.2.2 Multivariate Regime-Switch Nonlinear Systems

As we have done in Chapter 2, for nonlinear multivariate hybrid systems, subsystems are modelled by nonlinear function for the i-th state variable.

$$\delta(x_i(t)) = \mathbf{f}^{\alpha_t}\left(\mathbf{x}(t), \ldots, \mathbf{x}(t-n_a), \mathbf{u}(t), \ldots, \mathbf{u}(t-n_b)\right) + \xi(t), \tag{4.3}$$

where $t \in [t_0, M]$. $\delta(\cdot)$ is defined as in (2.29) The discrete variable $\alpha_t \in \{1, \ldots, s\}$ - the mode of the system - indicates which of the s submodels is active at time instant t. The time instants where the value of α_t changes are call *discrete transitions* or *switches*. These switches partition the interval $[0, M]$ into a *discrete hybrid time set* [121], $\tau = \{I_i\}_{i=0}^{k}$, such that α_t is constant within each subinterval $I_i = [\tau_i, \tau_i']$ and different in consecutive intervals. In the sequel, we denote by τ_i and τ_i' the beginning and ending times of the i-th interval, respectively. Clearly, τ satisfies

- $\tau_0 = 0$, and $\tau_k' = M$,
- $\tau_i \leq \tau_i' = \tau_{i+1} - 1$,

and the number of switches is equal to k.

We assume $\mathbf{f}^{\alpha_t}(\cdot)$ can be expanded as a linear combination of finite basis functions. Formally, similar as Assumption 2, we have

Assumption 8 *The function terms $\mathbf{f}^{\alpha_t}(\cdot)$ in* (4.3) *can be represented as a linear combinations of several dictionary functions:*

$$\mathbf{f}^{\alpha_t}(\cdot) = \sum_{i=1}^{N} \theta_i(\alpha_t) f_i(\cdot), \tag{4.4}$$

where $\theta_i(\alpha_t) \in \mathbb{R}$ and $f_i(\mathbf{x}(t)) : \mathbb{R}^{n_a + \sum_{i=1}^{m} n_{b_i}} \to \mathbb{R}^{n_\mathbf{x}}$ are smooth functions that are assumed to govern the dynamics.

For simplicity, we let $n_a = 0$ and input $u = 0$. By letting

$$y(t) = \delta(x_i(t))$$

and under Assumption 8, an equivalent linear regression model of (4.3) is

$$y(t) = \begin{pmatrix} f_1(\mathbf{x}(t)) & \ldots & f_N(\mathbf{x}(t)) \end{pmatrix} \cdot \boldsymbol{\beta}^{\alpha_t} + \xi(t), \tag{4.5}$$

where $f_j(\mathbf{x}(t) : \mathbb{R}^{n_a + \sum_{i=1}^{m} n_{b_i}} \to \mathbb{R}^{n_\mathbf{x}}$.

Denote $\mathcal{I}_i$ is a subset of $\tau = \{I_i\}_{i=0}^{k}$. And $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$, $\forall i \neq j$ and $\mathcal{I}_1 \cup \ldots \cup \mathcal{I}_s = \tau$. The *discrete transition* α_t depends on the time instant t, i.e.

$$\mathbf{f}^{\alpha_t}(\mathbf{x}) = \begin{cases} \mathbf{f}_1(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}(t)) & \ldots & f_N(\mathbf{x}(t)) \end{pmatrix} \cdot \boldsymbol{\beta}^1, & \text{if } t \in \mathcal{I}_1 \\ \vdots & & \vdots \\ \mathbf{f}_s(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}(t)) & \ldots & f_N(\mathbf{x}(t)) \end{pmatrix} \cdot \boldsymbol{\beta}^s, & \text{if } t \in \mathcal{I}_s \end{cases}. \tag{4.6}$$

In each mode, the submodel $\mathbf{f}_j$ is a smooth function. Therefore, the underlying assumption on models with the number of switches s is relatively small

$$s \leq k \ll M. \tag{4.7}$$

4.3 Linear Regression Model

Define the following block matrix and vectors, for $i = 1, \ldots, n_{\mathbf{x}}$

$$\begin{aligned} \mathbf{y}_i &\triangleq [y(1), \ldots, y(M)]^\top, \\ \mathbf{X}_i &\triangleq \left[\begin{array}{ccc|c|ccc} f_1(\mathbf{x}(1)) & & & \cdot & & f_N(\mathbf{x}(1)) & \\ & \ddots & & \ddots & & \ddots & \\ & & f_1(\mathbf{x}(M)) & \cdot & & & f_N(\mathbf{x}(M)) \end{array}\right] \\ &= \left[\begin{array}{c|c|c} A_1 & \ldots & A_N \end{array}\right] \in \mathbb{R}^{M \times MN}, \\ \boldsymbol{\beta}_i &\triangleq \left[\begin{array}{ccc|c|ccc} \beta_{i1}(1), & \ldots, & \beta_{i1}(M) & \ldots & \beta_{iN}(1), & \ldots, & \beta_{iN}(M) \end{array}\right]^\top \\ &= \left[\begin{array}{c|c|c} \boldsymbol{\beta}_{i1}^\top & \ldots & \boldsymbol{\beta}_{iN}^\top \end{array}\right]^\top \in \mathbb{R}^{MN}, \\ \boldsymbol{\Xi}_i &\triangleq [\xi(1), \ldots, \xi(M)]^\top \in \mathbb{R}^M. \end{aligned} \tag{4.8}$$

It should be noted that the stochastic term is Gaussian i.i.d with covariance matrix $\sigma^2\mathbf{I}$. We can then formulate the following linear regression problem

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta}_i + \boldsymbol{\Xi}_i. \tag{4.9}$$

There are two issues that need to be considered at this stage. First, each block $\boldsymbol{\beta}_{in} = [w_{in}(1), \ldots, w_{in}(M)]$ is associated only with certain dictionary function. The solution $\hat{\boldsymbol{\beta}}_i$ to (4.9) is therefore typically going to be block sparse, which is mainly due to the potential introduction of non-relevant and/or non-independent basis functions in $\mathbf{X}$. Second, in the switched case, we have to penalise the number of

switches from t_1 to t_M and/or the number of modes N_{modes}, which can be fixed in advance or set equal to M. Clearly such a problem has an infinite number of solutions, especially in the noisy setting. Therefore, we refine the problem statement to identify the system (4.9) with the least number of non-zero entries in $\boldsymbol{\beta}_i$ and the least number of switches in the sequence α_t.

These are actually two different and competing criteria: if we want the least number of switches, the number of non-zero parameters in $\boldsymbol{\beta}_i$ will grow, and vice versa, a small number of non-zero parameters in $\boldsymbol{\beta}_i$ will result in many switches.

4.4 Linear Regression Problem

4.4.1 Regression Problem Statement

Since the $n_{\mathbf{x}}$ linear regression models in Eq. (4.8) are independent, for simplicity of notation, we omit the subscript i used to index the state variable and simply write:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\Xi}. \tag{4.10}$$

We assume the stochastic term $\boldsymbol{\Xi}$ is Gaussian i.i.d. Further discussion on the distribution on $\boldsymbol{\Xi}$ can be found in Section 5.4. The identification problem can be formally stated as follows which is similar to the one in Section 2.4 and 3.3:

Problem 6 (Consistency) *Given* $\mathbf{y}$ *and* $\mathbf{X}$ *over the interval* $[1, M]$*, find a coefficient vector* $\boldsymbol{\beta}$ *such that* (4.9) *holds for all* $t \in [1, M]$.

Since the stochastic term $\boldsymbol{\Xi}$ is under the Gaussian i.i.d assumption, the basic consistency result is almost immediate under quadratic criteria, i.e. using ordinary least square (OLS)

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2. \tag{4.11}$$

It is clear that this problem is not well-posed and has infinitely many solutions. For instance, one can always find a trivial piecewise affine model with M submodels or on model with a large order or/and a large number of (nonlinear) dictionary functions that perfectly fits the data, which is known to be ***overfitting***. This situation can be partially avoided by imposing upper bounds n_y and n_{u_i}, $i = 1, \ldots, m$, on the order of each of the terms on the right hand side of (4.3), e.g., $n_a \leq n_y$, $n_{c_i} \leq n_{u_i}$. Still, even in this case the problem admits multiple solutions, especially when n_y

and n_{u_i} are very large. More interesting problems can be posed by using the existing degrees of freedom to optimise suitable performance criteria. One such criterion is to minimise the number of switches (i.e., the minimum number of k), subject to consistency. Under this criterion, the hybrid system is better characterised by (4.6), where $s \leq k \ll M.$. The formal statement of the identification problem with this criterion is as follows:

Problem 7 (A minimum number of switches) *Given* $\mathbf{y}$ *and* $\mathbf{X}$ *over the interval* $[1, M]$ *and* a prior *selection of a set of* N *basis functions, find a coefficient vector* $\boldsymbol{\beta}$ *as in* (4.8) *with a minimum number of switches, such that* (4.9) *holds for all* $t \in [1, M]$.

To further avoid the overfitting issue, $\mathbf{f}^{\alpha_t}(\mathbf{x}(t))$ is usually favoured to be sparsely represented, like we discussed in the previous Chapters. Thus other criterion is to minimise number of same type of dictionary functions. Then Problem 6 can be modified with the following formal statement:

Problem 8 (A minimum number of basis functions) *Given* $\mathbf{y}$ *and* $\mathbf{X}$ *over the interval* $[1, M]$ *and* a prior *selection of a set of* N *basis functions, find a coefficient vector* $\boldsymbol{\beta}$ *as in* (4.8) *with minimum number of basis functions, such that* (4.9) *holds for all* $t \in [1, M]$.

Now we can consider Problem 7 and Problem 8 simultaneously and state that

Problem 9 *Given* $\mathbf{y}$ *and* $\mathbf{X}$ *and the block partitions formulated in (4.9), find a* $\boldsymbol{\beta}$ *that can explain the data with the minimal number of switches and the minimal number of non-zero blocks in* $\boldsymbol{\beta}$.

4.4.2 Nonconvex Optimisation Problem

If we index the vector $\boldsymbol{\beta}$ appropriately, the problem of minimising the number of switches can be formulated by enforcing $\mathbf{D}_n\boldsymbol{\beta}_n$ sparse, where the matrix $\mathbf{D}_n$ is defined as follows:

$$\mathbf{D}_n \triangleq \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(M-1)\times M}. \tag{4.12}$$

Problem 7 can be formulated as follows

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{n=1}^{N} \|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_0}, \tag{4.13}$$

where λ_1 in (4.13) is the regularisation parameter in this penalised linear regression problems.

Considering $\boldsymbol{\beta} = \left[\begin{array}{c|c|c} \boldsymbol{\beta}_1^\top & \dots & \boldsymbol{\beta}_N^\top \end{array} \right]^\top$ as in (4.8) is block-wise defined, the problem of minimising the number of basis functions can be formulated by enforcing $\boldsymbol{\beta}$ with a minimum number of nonzero blocks $\boldsymbol{\beta}_n$, $n = 1, \dots, N$, Problem 8 can be formulated as follows

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_2 \sum_{n=1}^{N} \|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0}, \tag{4.14}$$

Since

$$\|\boldsymbol{\beta}_n\|_{\ell_2} \leq \|\boldsymbol{\beta}_n\|_{\ell_1},$$

we can have a bit more ambitious target

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_2 \sum_{n=1}^{N} \|\|\boldsymbol{\beta}_n\|_{\ell_1}\|_{\ell_0}, \tag{4.15}$$

Here we may prefer (4.14) to (4.15) since the former has clearer interpretation in the dynamical model and less computation challenge than (4.15).

Based on the formulation in (4.13) and (4.14) , we can have the following regression problem for Problem 9

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{n=1}^{N} \|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_0} + \lambda_2 \sum_{n=1}^{N} \|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0}, \tag{4.16}$$

The first regulariser $\lambda_1 \sum_{n=1}^{N} \|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_0}$ penalise the number of switches that occur; and the second regulariser $\lambda_2 \sum_{n=1}^{N} \|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0}$ the number of non-zero element in every identified model.

4.4.3 Convex Relaxation

Again, similar as we discussed in Section 2.4.3 and 3.3.3, we replace the ℓ_0 with ℓ_1 in the optimisation problem (4.13), (4.14) and (4.25), we can reach to the following empirical convex relaxations. For (4.13), the relaxation is

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{n=1}^{N} \|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_1}. \tag{4.17}$$

or we can write in a more compact form by defining

$$\mathbf{D} = \text{blkdiag}\left(\mathbf{D}_1, \ldots, \mathbf{D}_N\right), \tag{4.18}$$

as

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\mathbf{D}\boldsymbol{\beta}\|_{\ell_1}. \tag{4.19}$$

For (4.14), the relaxation is

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_2 \sum_{n=1}^{N} \|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_1}, \tag{4.20}$$

or

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_2 \sum_{n=1}^{N} \|\boldsymbol{\beta}_n\|_{\ell_2}. \tag{4.21}$$

For (4.16), the relaxation is

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{n=1}^{N} \|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_1} + \lambda_2 \sum_{n=1}^{N} \|\boldsymbol{\beta}_n\|_{\ell_2} \tag{4.22}$$

Algorithms minimising the number of non-zero elements and the number of switches, i.e. (4.22), belong to the class of so-called *fused Lasso algorithms* [189]. For general $\mathbf{D}$ matrices, the problem defined in (4.19) and (4.22) would be solved using *generalised Lasso algorithms* [190].

4.5 Models with Abrupt Change

4.5.1 Trend Filtering

We are given a scalar time series $y(t)$, $t = 1, \ldots, M$, assumed to consist of an underlying slowly varying trend $\beta(t)$. The problem we want to solve is to estimate the random component $z(t) = y(t) - \beta(t)$. This can be considered as an optimisation problem with two competing objectives: we want $\beta(t)$ to be smooth, and we want $z(t)$ (our estimate of the random component, sometimes called the *residual*), to be small. In some contexts, estimating $\beta(t)$ is called *smoothing* or *filtering*.

In [101], an ℓ_1 trending filter is proposed and a comprehensive review on the state-of-art of trend filtering is introduced. Trend filtering comes up in several applications and settings including macroeconomics (e.g., [87, 172]), geophysics (e.g., [13, 23, 24]), financial time series analysis (e.g., [198]), social sciences (e.g., [114]),

revenue management (e.g., [186]), and biological and medical sciences (e.g., [71, 115]). Many trend filtering methods have been proposed, including Hodrick-Prescott (H-P) filtering [87, 113], moving average filtering [131], exponential smoothing [120], bandpass filtering [37, 18], smoothing splines [160], de-trending via rational square-wave filters [157], a jump process approach [236], median filtering [206], a linear programming (LP) approach with fixed kink points [126], and wavelet transform analysis [42]. (All these methods except for the jump process approach, the LP approach, and median filtering are linear filtering methods; see [18] for a survey of linear filtering methods in trend estimation.) The most widely used methods are moving average filtering, exponential smoothing, and H-P filtering, which is especially popular in economics and related disciplines since its application to business cycle theory [87]. The idea behind H-P filtering can be found in several fields, and can be traced back at least to work in 1961 by Leser [113] in statistics.

Let's revisit the optimisation problem in (4.13), when $\mathbf{X}$ is the identity matrix $\mathbf{I}_N$, and $\mathbf{D}$ is like second difference. Hereafter, we introduce the "sparsest" version of the trend filter problem, i.e.,

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\boldsymbol{\beta}\|_2^2 + \lambda\|\mathbf{D}\boldsymbol{\beta}\|_{\ell_0}, \tag{4.23}$$

where

$$\mathbf{D} \triangleq \begin{bmatrix} 1 & -2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{(M-2)\times M}. \tag{4.24}$$

For what follows, it is useful and clearer to rewrite (4.23) in the following equivalent form:

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\sum_{t=1}^{M}\|y(t)-\beta(t)\|_2^2 + \lambda\sum_{t=2}^{M-1}\|\beta(t-1)-2\beta(t)+\beta(t+1)\|_{\ell_0}, \tag{4.25}$$

where $\beta(t)$ is the trend estimate; $\lambda \geq$ is the regularisation parameter used to control the trade-off between the smoothness of $\beta(t)$ and the size of residual $y(t)-\beta(t)$. The first term in the objective function measures the size of the residual; the second term measures the smoothness of the estimated trend. The argument appearing in the second term, $\beta(t-1)-2\beta(t)+\beta(t+1)$, is the second difference of the time series at time t; it is zero when and only when the three points $\beta(t-1)$, $\beta(t)$ and $\beta(t+1)$ are on a line.In other words, the secondterm in the objective is zero if and only if $\beta(t)$ is

affine, i.e., has the form $\beta(t) = a + bt$ for some constants a and b. (In other words, the graph of $\beta(t)$) is a straight line.)

For the problem (4.25), two convex relaxation strategies can be employed. The first one is to replace ℓ_0 norm penalty with a "sum of square" norm term

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \boldsymbol{\beta}\|_2^2 + \lambda\|\mathbf{D}\boldsymbol{\beta}\|_{\ell_2}^2, \tag{4.26}$$

This optimisation problem (4.26) is strictly convex and coercive in β, and so has a unique minimiser. The ℓ_2 relaxation is known as the Hodrick-Prescott (H-P) filter. The H-P filter is supported in several standard software packages for statistical data analysis, e.g., `SAS`, `R`, and `Stata`.

The other relaxation strategy is to replace the ℓ_0 norm penalty with ℓ_1 norm term

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \boldsymbol{\beta}\|_2^2 + \lambda\|\mathbf{D}\boldsymbol{\beta}\|_{\ell_1}, \tag{4.27}$$

which is known as an ℓ_1 trend filter [101]. The principal difference is that the ℓ_1 trend filter produces trend estimates that are smooth in the sense of being piecewise linear. The ℓ_1 trend filter is thus well suited to analysing time series with an underlying piecewise linear trend. The kinks, knots, or changes in slope of the estimated trend can be interpreted as abrupt changes or events in the underlying dynamics of the time series; the ℓ_1 trend filter can be interpreted as detecting or estimating changes in an underlying linear trend.

4.5.2 Fault Diagnosis Problem

In the previous Sections in this Chapter, we were trying to find the abrupt change point from *static* time series. The motivation for this Section is to pose the problem when time series data are streaming in and out *on line* or in *real time*. This problem is known as the fault diagnosis problem in the system and control context. Regarding the recommendations of the IFAC Technical Committee *SAFEPROCESS*, this work proposes a method to: 1) decide whether there is an occurrence of a fault and the time of this occurrence (*detection*), 2) establish the location of the detected fault (*isolation*), and 3) determine the size and time-varying behaviour of the detected fault (*identification*).

Re-consider the linear regression model in (2.47)

$$\mathbf{y}_i = \boldsymbol{\Psi}_i \mathbf{v}_i + \boldsymbol{\Xi}_i, \ i = 1, \ldots, n_{\mathbf{x}}. \tag{4.28}$$

with

$$\begin{aligned}
\mathbf{y}_i &\triangleq [y_i(1), \ldots, y_i(M)]^\top \in \mathbb{R}^{M\times 1}, \\
\mathbf{\Psi}_i &\triangleq \begin{bmatrix} f_{i1}(\mathbf{x}(1), \mathbf{u}(1)) & \ldots & f_{iN_i}(\mathbf{x}(1), \mathbf{u}(1)) \\ \vdots & \ddots & \vdots \\ f_{i1}(\mathbf{x}(M), \mathbf{u}(M)) & \ldots & f_{iN_i}(\mathbf{x}(M), \mathbf{u}(M)) \end{bmatrix} \in \mathbb{R}^{M\times N_i}, \\
\mathbf{v}_i &\triangleq [v_{i1}, \ldots, v_{iN_i}]^T \in \mathbb{R}^{N_i\times 1}, \\
\mathbf{\Xi}_i &\triangleq [\xi_i(1), \ldots, \xi_i(M)]^\top \in \mathbb{R}^{M\times 1}.
\end{aligned} \tag{4.29}$$

We will have the following problem statement for dynamic change point detection problem

Definition 2 *If a system can be described by* (4.28)*, the system is faulty when v_{ij} changes to a new scalar $v_{ij}^{[\mathbf{f}]}$.*

Based on the considerations above and Definition 2, the problem that we are interested in solving is the following:

Problem 10 *Having access to the measurements and the distribution of the noise, how can we detect the occurrence and magnitude of a fault, namely, how can we estimate the magnitude of the errors $v_{ij} - w_{ij}^{[\mathbf{f}]}$, $\forall i, j$, using the smallest possible number of samples.*

Chapter 5

Technical Issues Related to Dynamical System Identification

In this Chapter, we discuss some technical issues related to dynamical systems, model structure selections, and data preprocessing in practice.

5.1 Uniquesness of Solutions in Chapter 2

First, we introduce a very important concept in SYSID, namely, *identifiability*. As pointed out in [117], identifiability is a concept that is central in SYSID problem. Loosely speaking, the problem is whether the identification procedure will yield a unique value of the parameter $\boldsymbol{\theta}$, and/or whether the resulting model is equal to the true system. This issue, regrettably to say, is beyond the scope of the thesis. The issue involves aspects on whether the data set (the experimental conditions) is informative enough to distinguish between different models as well as properties of the model structure itself: If the data are informative enough to distinguish between nonequal models, then the question is whether different values of $\boldsymbol{\theta}$ can give equal models. With the terminology mentioned in Figure 1.1, the latter problem concerns the *invertibility of the model structure* $\mathcal{M}$ (i.e., whether $\mathcal{M}$ is injective). The definition of *global identifiability* is

Definition 3 *A model structure* $\mathcal{M}$ *is* globally identifiable at $\boldsymbol{\theta}^*$ *if*

$$\mathcal{M}(\boldsymbol{\theta}) = \mathcal{M}(\boldsymbol{\theta})^*, \qquad \boldsymbol{\theta} \in D_{\mathcal{M}} \Rightarrow \boldsymbol{\theta} = \boldsymbol{\theta}^*. \tag{5.1}$$

Thus the identification condition for general nonlinear system is hardly given. It will be dangerous to assume the system under identification is identifiable a prior. But under the Assumption 5, we can have another way to approach identifiability condition. If the system under consideration is identifiable, we cannot get a sparser solution than the true one, as this would contradict the identifiability assumption, i.e., more than one model can equivalently explain the data. In order to search for the sparsest solution $\boldsymbol{\beta}$, we impose a penalty on the ℓ_0 norm of $\boldsymbol{\beta}$, $\|\boldsymbol{\beta}\|_0$, i.e. on the number of nonzero elements in $\boldsymbol{\beta}$. With the addition of this ℓ_0 norm penalty, the linear regression problem (9.8) can be formulated into the following regularised regression problem, which is also known as an ℓ_0-minimisation problem [31, 196, 50]:

$$\hat{\mathbf{w}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_0}. \tag{5.2}$$

In (1.4), $\mathbf{y}$ is the the vector observations, $\mathbf{X}$ is a known regressor matrix, $\boldsymbol{\beta}$ is the vector of unknown coefficients and λ is a tradeoff parameter. Subsequently, one may

wonder what the gap between the solution to this ℓ_0-minimisation problem and the true solution is.

To characterise this gap, we shall firstly introduce the following definition.

Definition 4 *[Definition 1 of [51]] The spark of a given matrix A is the smallest number of columns of A that are linearly dependent.*

Proposition 1 *[Corollary 1 of [51]] In the noiseless case where $\boldsymbol{\eta} = \mathbf{0}$ for any vector $\mathbf{y} \in \mathbb{R}^M$, there exists one unique signal $\boldsymbol{\beta}$, such that $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$ with $\|\mathbf{w}\|_0 = S$ if and only if Spark$(\mathbf{X}) > 2S$.*

Remark 10 *It is easy to see that Spark$(\mathbf{X}) \in [2, M+1]$. Therefore, in order to get the unique S-sparse solution $\boldsymbol{\beta}$ to $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$, Proposition 1 imposes that $M \geq 2S$.*

Corollary 1 *If the number of samples M is greater or equal to 2 times the number of nonzero elements S in the "true" value of $\mathbf{w}$, then the ℓ_0-minimisation solution $\mathbf{w}$ to the equation $\mathbf{y} = \mathbf{X}\mathbf{w}$ will be consistent with the "true" value.*

Proof *Since the sparsest solution can be obtained through ℓ_0-minimisation in (5.2), this Corollary is straightforward from Proposition 1 and Remark 10.*

Remark 11 *This Corollary bridges the gap between the "true" solution and that obtained by ℓ_0-minimisation provided the assumptions of Corollary 1 hold. If these assumptions do not hold, then prior knowledge, additional experiments and/or data points might be required.*

As introduced in Chapter 1.3, the *restricted isometry property* (RIP) condition is a sufficient condition for exact reconstruction based on ℓ_1-minimisation. It was shown in [31, 44, 30] that both convex ℓ_1-minimisations and greedy algorithms lead to exact reconstruction of S-sparse signals if the matrix $\mathbf{X}$ satisfies the RIP condition. The ℓ_1 relaxation of the optimisation problem in (5.2) is

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_1} \tag{5.3}$$

Unfortunately, either RIP condition or incoherence condition is hardly satisfied in system identification problems since the dictionary matrix is constructed from time series data and the candidate basis functions can be selected arbitrarily in principle. ℓ_1 relaxation can hardly be expected to show high performance. Therefore, the Bayesian approach can be potentially alleviate these issues.

5.2 Selection of Candidate Basis Functions

It can be found that the success of modelling highly depends on the selection of the dictionary function. However, the selection procedure is a piece of art which is field specific and requires subtle domain knowledge. For example, $\sin$ function can be hardly found in the description of biochemical reaction dynamics. Fortunately, some general basis functions may exist and this setup is familiar in a number of fields, including standard expansions in terms of orthogonal polynomials in classical physics [1] as well as in approximation theory [165] and neural networks [85]. An exhaustive introduction and review of the commonly used basis functions can be found in some classic text books from different communities, e.g., system identification by Ljung [117, Chapter 5.4], statistical learning by Hastie, Tibshirani and Friedman [79, Chapter 5], or kernels named in machine learning By Murphy [127, Chapter 14], Bishop [22, Chapter 6].

The candidate dictionary functions $\boldsymbol{\varphi}_i$ can assume any of a variety of forms, including polynomials, rational, exponentials, sinusoidals, or others, based on the underlying knowledge of the system. Classical functional decomposition methods, e.g., polynomial Volterra and Taylor expansions, orthogonal polynomials or Fourier series [15, 117, 165], are sometimes used to approximate the functional behaviour even if the number of dictionary functions could, in theory, be infinite. Here, we focus on systems relevant in many modelling and experimental setups where we can use our *a priori* knowledge about the system to propose an informed set of dictionary functions based on the fundamental physical or biological laws expected to be at play. In many physical systems, the models include particular classes of functional couplings that emerge directly from the physical field interactions, e.g., laser arrays [211], arrays of antennas [185], mechanical couplings [82], power systems [2]. In chemical reaction networks only polynomial terms typically need to be considered [7], whereas biochemical networks relevant in Systems and Synthetic Biology of the cell, typically involves nonlinearities that capture fundamental biochemical kinetic laws, e.g., first-order degradation functions, mass-action kinetics, Hill and Michaelis-Menten functions, which are confined to either polynomial or rational functions [5]. In neuroscience, sparse representations for neural coding are extracted [130]. The appropriate selection of dictionary functions is an important area of current interest in system identification in other areas of systems engineering [117].

Try Black Box Expansions We could construct the regressors as typical (polynomial) combinations of the past inputs and outputs and see if the model is able to describe the data. It normally gives a large number of possible regressors. It is somewhat simpler for the Hammerstein model, where we may approximate the static nonlinearity by a polynomial expansion:

$$f(u) = \alpha_1 u + \alpha_2 u^2 + \ldots + \alpha_m u^m. \tag{5.4}$$

Each power of u could then pass different numerator dynamics:

$$A(q)y(t) = B_1(q)u(t) + B_2(q)u^2(t) + \ldots + B_m(q)u^m(t). \tag{5.5}$$

This is clearly a linear regression model structure.

Use Physical Insight A few moments reflection, using college physics, often may reveal which are the essential nonlinearities in a system. This will suggest which regressor to try in (2.23). We call this *semi-physical modelling*. It could be as simple as taking the product of voltage and current measurements to create a power signal.

Polynomial and Rational Functions Polynomial and rational representations are probably the most widely used, at least in this thesis, in the context of system and synthetic biology modelling and control engineering.

If $\mathbf{f}_i(\cdot)$ can be represented by polynomial functions then it can be decomposed into sums of monomial terms. A monomial m_d in n variables is a function defined as $m_d(\mathbf{x}) \triangleq x_1^{d_1} x_2^{d_2} \ldots x_n^{d_n}$ for $d_i \in \mathbb{Z}_+$. The degree of a monomial is defined as $\deg m_d := \sum_{i=1}^n d_i$. Polynomials being decomposable into sums of monomial terms, the elements $\mathbf{f}_{is}(\mathbf{x})$ appearing in the dictionary function matrix $\mathbf{X}_i$ can be represented as monomials of the form:

$$\mathbf{f}_{is}(\mathbf{x}) = x_1^{d_1^{[is]}} x_2^{d_2^{[is]}} \ldots x_n^{d_n^{[is]}}. \tag{5.6}$$

In this example of a system with states and inputs memories, we will show how to construct the expanded dictionary matrix by adding candidate nonlinear functions.

Example 3 *To illustrate how to expand $\mathbf{\Psi}_i$ to $\mathbf{X}_i$, we consider the following general SISO NARX model with polynomial terms:*

$$\begin{aligned} x(t+1) &= w_1 + w_2 x(k) + \ldots + w_{m_x+2} x(k-m_x) + w_{m_x+3} x(t)x(k-1) + \ldots \\ &\quad + w_N x^{d_x}(k-m_x)u^{d_u}(k-m_u) + \xi(k) \\ &= \boldsymbol{\beta}^T \mathbf{f}(x(t), \ldots, x(k-m_x), u(t), \ldots, u(k-m_u)) + \xi_i(t), \end{aligned} \tag{5.7}$$

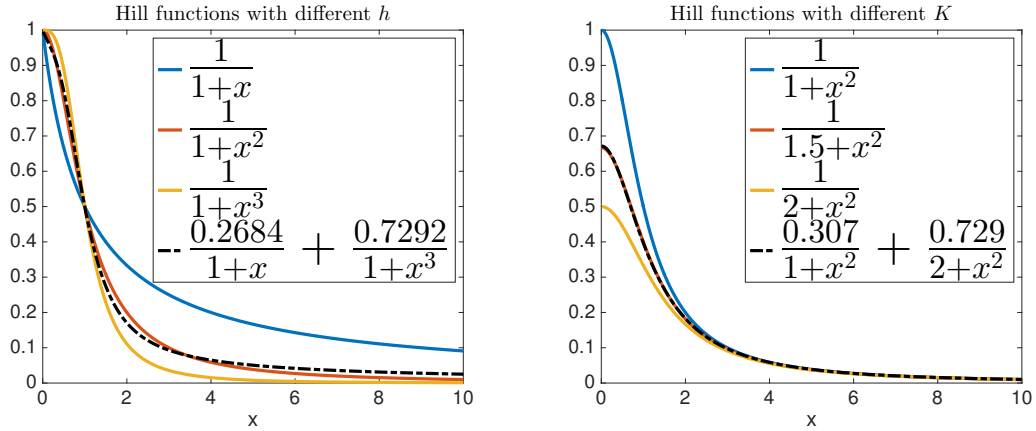
where d_x (resp. d_u) is the degree of the output (resp. input); m_x (resp. m_u) is the maximal memory order of the output (resp. input); $\boldsymbol{\beta}^\top = [w_1, \ldots, w_N] \in \mathbb{R}^N$ is the weight vector; and $\mathbf{f}(x(t), \ldots, x(k-m_x), u(t), \ldots, u(k-m_\mathbf{u})) = [f_1(\cdot), \ldots, f_N(\cdot)]^\top \in \mathbb{R}^N$ is the dictionary functions vector. By identification of (5.7) *with the NARX model (2.24), we can easily see that $d_x = 5$, $d_u = 4$, $m_x = 2$, $m_u = 2$. To define the dictionary matrix, we consider all possible monomials up to degree $d_x = 5$ (resp. $d_u = 4$) and up to memory order $m_x = 5$ (resp. $m_u = 2$) in x (resp. u). This yields $\mathbf{f}(\cdot) \in \mathbb{R}^{1960}$ and thus $\boldsymbol{\beta} \in \mathbb{R}^{1960}$. Since $\mathbf{v} \in \mathbb{R}^4$, only 4 out of the 1960 associated weights w_i are nonzero.*

5.3 Dealing with Basis Function Nonlinearity

The *a priori* selection of a good set of dictionary functions $\mathbf{f}_{is}(\mathbf{x})$ is key to the identification process. Some *a priori* knowledge of the provenance of the data and the field for which the models are developed can be particularly helpful for this. For example, the typical nonlinearities used to create nonlinear ODE models of gene regulatory networks can be restricted to those known to capture fundamental biochemical kinetic laws, e.g. first-order functions $f(x) = \alpha x$, mass action functions $f([x_1, x_2]) = \beta x_1 \cdot x_2$, Michaelis-Menten functions $f(x) = \frac{V_{\max}}{K+x}$, or Hill functions $f(x) = \frac{V_{\max}}{K+x^h}$. Using our framework, h and K are assumed to be known *a priori*, whereas $\alpha, \beta, V_{\max}$ can be identified through the process described in the previous sections.

Thus, to find practical solutions to the identification of the parameters embedded nonlinearly in the dictionary functions is challenging, e.g. the parameters h and K of the Hill functions.

A naive solution to the estimation of the Hill coefficient, h, is to introduce more nonlinear terms in the set of dictionary functions, each with a different Hill coefficient. Since $h \in \mathbb{Z}^+$ and very few biological systems are characterised by Hill coefficients larger than 8, the number of such terms is typically relatively low. On the basis of this, the set of Hill functions $\frac{V_{\max}}{K+x^h}$ with $h = 1, 2, \cdots, 8$ is a good candidate subset to



(a) Hill functions $\frac{1}{1+x^h}$ characterised by different Hill coefficients, h, with $h = \{1, 2, 3\}$. The Hill function $\frac{1}{1+x^2}$ is tightly approximated by the linear combination $\frac{0.2684}{1+x} + \frac{0.7292}{1+x^3}$.

(b) Hill functions $\frac{1}{K+x^2}$ characterised by different Hill thresholds, K, with $K = \{1, 1.5, 2\}$. The Hill function $\frac{1}{1.5+x^2}$ is tightly approximated by the linear combination $\frac{0.307}{1+x^2} + \frac{0.729}{2+x^2}$.

Fig. 5.1 Hill functions can be approximated by linear combinations of other Hill functions.

be included in the set of dictionary functions. Furthermore, even if the true function is not a member of the considered set of dictionary functions, it is often the case that the true dictionary function can be approximated by a linear combination of the other members of the set of dictionary functions. For example, suppose the true function to be identified is $\frac{1}{1+x^2}$ and that the set of dictionary functions is $\{\frac{1}{1+x}, \frac{1}{1+x^3}, \frac{1}{1+x^4}\}$. The true function $\frac{1}{1+x^2}$ can be approximated as a linear combination of the other rational functions present in the set of dictionary functions: $\frac{1}{1+x^2} \approx a \cdot \frac{1}{1+x} + b \cdot \frac{1}{1+x^3} + 0 \cdot \frac{1}{1+x^4}$, where a and b are some real numbers that can be identified using our framework, see Figure 5.1(a).

The estimation of the Hill threshold parameter K can be dealt with in a similar manner as for the Hill coefficient h. For example, the nonlinear Hill function $\frac{1}{1.5+x^2}$ can be approximated by a linear combination of Hill functions with different values of K: $\frac{1}{1.5+x^2} \approx \frac{a}{1+x^2} + \frac{b}{2+x^2}$, where a and b are some real numbers that can be identified using our framework, see Figure 5.1(b).

5.4 Gaussian Assumption

In the linear regression models of the previous Chapters, i.e., Eq. (2.59), Eq. (3.7), and Eq. (4.10), we assumed the stochastic term Ξ to be Gaussian i.i.d. The stochastic term can be treated as dynamic noise in the dynamical systems' setting. Unfortunately,

noise is not always Gaussian in practice. However, here we will still use a Gaussian assumption to use the Gaussian assumption for a couple of reasons.

First of all, the underlying assumption for ordinary least square (OLS) is the Gaussian noise assumption. However, noise is called "residual" in the usual OLS literature. The nonconvex optimisation problems in Sections 2.4.2, 3.3.2 and 4.4.2 are essentially regularised least square problems. Therefore, these formulations should share the asymptotic convergence and consistency property regardless of the distribution of the residual.

Secondly, if the distribution of the noise belongs to an exponential family, it can be approximated by a series of Gaussian distributions. This is also the idea of Laplace approximation [127]. As summarised from Michael Jordan's lectures on "Bayesian Modelling and Inference" at UC Berkeley, the Gaussian assumption is naturally linked to Laplace approximation. The Laplace approximation is a general way to approach marginalisation problems. The basic idea is to approximate an integral of the following form:

$$I(t) = \int e^{-Mh(x)} dx \tag{5.8}$$

where M is typically the number of data points. After performing a Taylor series expansion of both $h(x)$ and the exponential function and evaluating some elementary integrals, we show that the following approximation of $I(t)$ can be derived.

$$I(M) = e^{-Nh(\hat{x})}\sqrt{2\pi}\sigma M^{-1/2}\left(1 - \frac{h_4(\hat{x})\sigma^4}{8N} + \frac{5h_3^2(\hat{x})\sigma^6}{24N}\right), \tag{5.9}$$

where $\sigma^2 = 1/h_2(\hat{x})$ and $\hat{x} = \operatorname{argmin}_x h(x)$. If $\hat{x}$ can not be determined analytically, it is typically approximated with some value $\tilde{x}$ such that the approximation error of $\hat{x} - \tilde{x}$ is within a factor of $\mathcal{O}(1/M)$. For example, in a Bayesian application $-h(x)$ can be the likelihood times a prior and $\hat{x}$ is then the maximum a posteriori probability (MAP). As M gets large, the MAP approaches the maximum likelihood estimate (MLE), so we can approximate $\hat{x}$ with the MLE and still obtain a rigorous accuracy bound.

The multivariate case is derived in exactly the same way s the univariate case. The only difference being that we perform a multivariate Taylor series expansion and get a multivariate Gaussian integral. Letting x denote a d-dimensional vector

and $h(x)$ a scalar function of x, we obtain:

$$\int e^{-Nh(x)}dx \approx e^{-Nh(\hat{x})}(2\pi)^{d/2}|\Sigma|^{1/2}M^{-d/2}, \tag{5.10}$$

where $\Sigma = (D^2h(\hat{x}))^{-1}$ is the inverse of the Hessian of h evaluated at $\hat{x}$. This expansion is accurate to order $\mathcal{O}(1/N)$, since we only consider the firs order terms of the Laplace approximation. However, as in Eq. (5.9) the expansion can be continued to obtain an accuracy of order $\mathcal{O}(1/N^2)$.

One application of the Laplace approximation is to compute the marginal likelihood. Letting $\mathfrak{M}$ be the marginal likelihood we have

$$\mathfrak{M} = \int P(X|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{5.11}$$

$$= \int \exp\left(-M\left(-\frac{1}{M}\log P(X|\boldsymbol{\theta}) - \frac{1}{M}\log \pi(\boldsymbol{\theta})\right)\right)d\boldsymbol{\theta} \tag{5.12}$$

where, $h(\boldsymbol{\theta}) = -\frac{1}{M}\log P(X|\boldsymbol{\theta}) - \frac{1}{M}\log \pi(\boldsymbol{\theta})$, Using the Laplace approximation up to the first order as in Eq. (5.10) we get

$$\mathfrak{M} \approx P(X|(\hat{\boldsymbol{\theta}}))\pi((\hat{\boldsymbol{\theta}}))(2\pi)^{d/2}|\Sigma|^{1/2}M^{-d/2} \tag{5.13}$$

This approximation is used for example in model selection, where computing the marginal likelihood analytically can be hard unless there is conjugacy. Computing the Laplace approximation requires finding the maximum a posteriori probability $\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}} -h(x)$, which can be done using a standard method such as gradient search. It also requires computing the second derivative matrix and inverting it to obtainΣ. This is usually the harder quantity to calculate.

5.5 Dealing with Measurement Noise

The model class considered in (2.49) can be enlarged in various ways. First, measurement noise, which is ubiquitous in practice, can be accounted for using the following linear measurement equation:

$$z_t = x_t + \epsilon_t, \tag{5.14}$$

where the measurement noise ϵ_t is assumed i.i.d. Gaussian. Under this formulation, the noise-contaminated data z_t represents the collected data rather than x_t. Second, the additive stochastic term ξ_t is often used to model dynamic noise or diffusion. In many practical application, however, it is necessary to account for multiplica-

tive noise instead of additive noise. Multiplicative noise can be accounted for by replacing the system equation with

$$\dot{x}_t = f(\mathbf{x}_t, \mathbf{u}_t)\mathbf{v} + h(\mathbf{x}_t, \mathbf{u}_t)\xi_t.$$

We show how the framework presented here can be modified to encompass these extensions. Without loss of generality and to ease notation, we consider the scalar case. In the scalar case, the system equation is written:

$$\dot{x}(t) = g(x(t)) + \eta_x(t), \tag{5.15}$$

while the measurement equation is given as:

$$y(t) = x(t) + \epsilon(t), \tag{5.16}$$

where $\epsilon(t)$ is the measurement noise which is assumed to be Gaussian i.i.d. We can simply use Taylor series expansion to expand $g(x(t))$ from $y(t)$:

$$\begin{aligned} g(x(t)) &= g(y(t) - \epsilon(t)) \\ &= g(y(t)) \underbrace{- g'(x(t))|_{x(t)=y(t)}\epsilon(t) + \mathcal{O}(\epsilon^2(t))}_{\textbf{Correlated Gaussian noise}} \\ &= g(y(t)) + \eta_y(t). \end{aligned} \tag{5.17}$$

Therefore, if we can estimate $\dot{x}$ from y properly, we can write the following

$$\begin{aligned} \dot{x}_{\textbf{estimate}}(t) &= g(y(t)) \underbrace{+ \eta_x(t) + \eta_y(t)}_{\textbf{new noise}} \\ &= g(y(t)) + \eta(t). \end{aligned} \tag{5.18}$$

Clearly, $\eta(t)$ is not independent and identically distributed anymore.

In stochastic differential equations used to describe biochemical reactions or in Langevin equations, the diffusion term is typically described as a multiplicative noise, e.g. $\eta_x(t)$ in (5.15) can be expressed as $\eta_x(t) = h(x(t))\epsilon(t)$ where $h(x(t))$ is unknown bounded nonlinear function and $\epsilon(t)$ is Gaussian i.i.d. If the form of $h(x(t))$ is not of particular interest, $\eta_x(t)$ can be absorbed by $\eta(t)$ in (5.18).

5.6 Estimation of the Derivative

Estimating time derivatives from noisy data in continuous-time systems can either be achieved using a measurement equipment with a sufficiently high sampling rate, or by using state-of-the-art mathematical approaches [48]. Estimation of derivatives is key to the identification procedure [48]. As pointed out in [145], the identification problem is generally solved through discretisation of the proposed model. Assuming that samples are taken at sufficiently short time intervals, various discretisation methods can be applied. Typically, a forward Euler discretisation is used to approximate first derivatives, i.e., $\mathbf{y}_i$ can be defined as

$$\mathbf{y}_i \triangleq \left[\frac{x_i(t_2) - x_i(t_1)}{t_2 - t_1}, \ldots, \frac{x_i(t_{M+1}) - x_i(t_M)}{t_{M+1} - t_M}\right]^\top \in \mathbb{R}^{M\times 1}.$$

In this thesis, the local polynomial regression framework in [48] is applied to estimate $\dot{\mathbf{x}}(t)$. Forward Euler discretisation and central difference discretisation are special cases of the local polynomial regression framework.

Proposition 2 (Proposition 1 in [48]) *Consider the bivariate data $(t_1, Y_1), \ldots, (t_M, Y_M)$. Assume data are equispace-sampled and let $\sum_{j=1}^{k} w_j = 1$. For $k+1 \leq i \leq n-k$, the weights w_j are chosen as:*

$$w_j = \frac{6j^2}{k(k+1)(k+2)}, \quad j = 1, \ldots, k.$$

Based on these weights, the first derivative can be approximated as:

$$Y_i^{'} = \sum_{j=1}^{k} w_j \cdot \left(\frac{Y_{i+j} - Y_{i-j}}{t_{i+j} - t_{i-j}}\right).$$

Part II

Algorithms

Chapter 6

Algorithms for Likelihood in Gaussian

We first revisit the nonlinear system identification problems defined in Chapters 2, 3 and 4 respectively.

Chapter 2: Linear/Nonlinear Time-Invariant Systems Revisit the nonconvex optimisation problem (2.61) and its convex relaxation in (2.62):

$$\begin{aligned}\text{Nonconvex Problem: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_{\ell_0},\\ \text{Convex Relaxation: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_{\ell_1}.\end{aligned}$$

Chapter 3: Nonlinear Dynamical System with Heterogeneous Datasets Revisit the nonconvex optimisation problem (3.9) and its convex relaxation in (3.11):

$$\begin{aligned}\text{Nonconvex Problem: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\sum_{n=1}^{N}\|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0},\\ \text{Convex Relaxation: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\sum_{n=1}^{N}\|\boldsymbol{\beta}_n\|_{\ell_2}.\end{aligned}$$

Chapter 4: Time-Varying Dynamical System Revisit the nonconvex optimisation problem (4.13) and its convex relaxation in (4.17):

$$\begin{aligned}\text{Nonconvex Problem: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\sum_{n=1}^{N}\|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_0},\\ \text{Convex Relaxation: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda\sum_{n=1}^{N}\|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_1}.\end{aligned}$$

Then for the complicated case, revisit the nonconvex optimisation problem (4.16) and its convex relaxation in (4.22):

$$\begin{aligned}\text{Nonconvex Problem: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda_1\sum_{n=1}^{N}\|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_0}+\lambda_2\sum_{n=1}^{N}\|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0},\\ \text{Convex Relaxation: } &\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda_1\sum_{n=1}^{N}\|\mathbf{D}_n\boldsymbol{\beta}_n\|_{\ell_1}+\lambda_2\sum_{n=1}^{N}\|\boldsymbol{\beta}_n\|_{\ell_2}.\end{aligned}$$

From the above problem formulations, it is found that these minimisation problems belonged to a penalised/regularised maximum a posteriori estimation form: data fitting term $\frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2$ plus some spare penalties/regularisations term. The underlying assumption behind the quadratic form of the data fitting term is that the

data likelihood is Gaussian distributed. In this Chapter, we will propose a general framework to address these above problems where data likelihood is Gaussian distributed. The structure of this Chapter is organised as follows. In Section 6.1, we revisit shortly about the data likelihood with Gaussian distribution. In Section 6.2, the sparse prior will be introduced as a controller for the structural sparsity. Next in Section 6.3, the optimisation problem from a Bayesian perspective will be defined. Subsequently in Section 6.4, some principles for solving the optimisation problem are discussed. Then in Section 6.5, the general optimisation algorithms are proposed. In the last three Sections, i.e., 6.6, 6.7 and 6.8, optimisation algorithms will be proposed for Chapters 2, 3 and 4 respectively.

6.1 Gaussian Likelihood

To get an estimate of $\boldsymbol{\beta}$, we use Bayesian modelling to treat all unknowns as stochastic variables with certain probability distributions. For

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\Xi},$$

it is assumed that the stochastic variables in the vector $\boldsymbol{\Xi}$ are Gaussian distributed with

$$\boldsymbol{\Xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Pi}).$$

In what follows we consider the following variable substitution for the inverse of covariance matrix or precision matrix:

$$\boldsymbol{\Pi}^{-1} \triangleq \boldsymbol{\Theta}.$$

In such case, the likelihood of the data given $\boldsymbol{\beta}$ is

$$\begin{aligned} p(\mathbf{y}|\boldsymbol{\beta}) &= \mathcal{N}(\mathbf{y}|\mathbf{X}\boldsymbol{\beta}, \boldsymbol{\Pi}) \\ &= \frac{1}{(2\pi)^{M/2}\,|\boldsymbol{\Pi}|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top}\boldsymbol{\Pi}^{-1}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)\right] \\ &= \frac{1}{(2\pi)^{M/2}\,|\boldsymbol{\Theta}|^{-1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top}\boldsymbol{\Theta}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)\right]. \end{aligned} \tag{6.1}$$

For simplicity, we typically define

$$\boldsymbol{\Pi} \triangleq \sigma^2\mathbf{I},$$

we have

$$\begin{aligned} p(\mathbf{y}|\boldsymbol{\beta}) &= \mathcal{N}(\mathbf{y}|\mathbf{X}\boldsymbol{\beta}, \boldsymbol{\Pi}) \\ &= \frac{1}{(2\pi)^{M/2}\,\sigma^N} \exp\left[-\frac{1}{2\sigma}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^\top \boldsymbol{\Theta}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)\right] \end{aligned} \tag{6.2}$$

with $\boldsymbol{\beta} \in \mathbb{R}^N$.

6.2 Sparse Prior

Based on [134], we apply the convex representation criteria for variational representations of non-Gaussian latent variables. In [134], it is showed the general equivalence between the convex variational representation and the integral type scale mixture representation. Moreover, they show general equivalence between the variational convex approximate maximum a posterior estimate of hyperparameters and the ensemble learning or variational Bayesian method.

We define a prior distribution $p(\boldsymbol{\beta})$ as

$$p(\boldsymbol{\beta}) \propto \exp\left[-\frac{1}{2}\sum_i g(\beta_i)\right] = \prod_{i=1}^N \exp\left[-\frac{1}{2}g(\beta_i)\right] = \prod_{i=1}^N p(\beta_i), \tag{6.3}$$

where $g(\beta_i)$ is a given function of β_i. To enforce sparsity on $\boldsymbol{\beta}$, the function $g(\cdot)$ is usually chosen as a concave, non-decreasing function of $|\beta_i|$. Examples of such functions $g(\cdot)$ include Generalised Gaussian priors and Student's *t* priors (see [134] for details).

Computing the posterior mean $\mathbb{E}(\boldsymbol{\beta}|\mathbf{y})$ is typically intractable because the posterior $p(\boldsymbol{\beta}|\mathbf{y})$ is highly coupled and non-Gaussian. To alleviate this problem, ideally one would like to approximate $p(\boldsymbol{\beta}|\mathbf{y})$ as a Gaussian distribution for which efficient algorithms to compute the posterior exist [22]. Another approach consists in considering *super-Gaussian* priors, which yield a lower bound for the priors $p(\beta_i)$ [134]. The sparsity inducing priors mentioned above are *super-Gaussian*. More specifically, if we define $\boldsymbol{\gamma} \triangleq [\gamma_1, \ldots, \gamma_N]^\top \in \mathbb{R}_+^N$, we can represent the priors in the following relaxed (variational) form:

$$\begin{aligned} p(\boldsymbol{\beta}) &= \prod_{i=1}^N p(\beta_i), \\ p(\beta_i) &= \max_{\gamma_i > 0} \mathcal{N}(\beta_i|0,\gamma_i)\varphi(\gamma_i), \end{aligned} \tag{6.4}$$

where $\varphi(\gamma_i)$ is a nonnegative function which is treated as a hyperprior with γ_i being its associated hyperparameters. Throughout, we call $\varphi(\gamma_i)$ the "*potential function*". This Gaussian relaxation is possible if and only if $\log p(\sqrt{\beta_i})$ is concave on $(0, \infty)$. The following proposition provides a justification for the above:

Proposition 3 *[134] A probability density $p(\beta_i) \equiv \exp(-g({\beta_i}^2))$ can be represented in the convex variational form: $p(\beta_i) = \max_{\gamma_i>0} \mathcal{N}(\beta_i|0, \gamma_i)\varphi(\gamma_i)$ if and only if $-\log p(\sqrt{\beta_i}) = g(\beta_i)$ is concave on $(0, \infty)$. In this case the potential function takes the following expression: $\varphi(\gamma_i) = \sqrt{2\pi/\gamma_i} \exp\left(g^*\left(\gamma_i/2\right)\right)$ where $g^*(\cdot)$ is the concave conjugate of $g(\cdot)$. A symmetric probability density $p(\beta_i)$ is said to be super-Gaussian if $p(\sqrt{\beta_i})$ is log-convex on $(0, \infty)$.*

Remark 12 *For the* Laplace *prior $p(\beta_i) \propto \exp(-\lambda \sum_j |w_i|)$, one can have a* Laplace *potential function $\varphi(\gamma_i) = \exp\left(-\frac{1}{2}|\gamma_i|\right)\sqrt{2\pi|\gamma_i|}$. For the* Student's t *prior $p(\beta_i) \propto (b + \frac{{\beta_i}^2}{2})^{-(a+\frac{1}{2})}$, one can have a* Student's t *potential function $\varphi(\gamma) = 1$, when $a, b \to 0$.*

For a fixed $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_N]$, we define a relaxed prior which is a joint probability distribution over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as

$$p(\boldsymbol{\beta}; \boldsymbol{\gamma}) = \prod_{i=1}^{N} \mathcal{N}(\beta_i|0, \gamma_i)\varphi(\gamma_i) = p(\boldsymbol{\beta}|\boldsymbol{\gamma})p(\boldsymbol{\gamma}) \leq p(\boldsymbol{\beta}) \tag{6.5}$$

where

$$p(\boldsymbol{\beta}|\boldsymbol{\gamma}) \triangleq \prod_{i=1}^{N} \mathcal{N}(\beta_i|0, \gamma_i), \quad p(\boldsymbol{\gamma}) \triangleq \prod_{i=1}^{N} \varphi(\gamma_i). \tag{6.6}$$

Now, we make a slight modification on the prior by introducing a linear transformation, i.e., $\mathbf{B}\boldsymbol{\beta}$. With $\mathbf{B} \in \mathbb{R}^{\aleph \times N}$, we can define

$$p(\mathbf{B}\boldsymbol{\beta}) = \prod_{i=1}^{\aleph} p(\mathbf{B}_{i,:}\boldsymbol{\beta}) \propto \prod_{i=1}^{\aleph} \exp\left[-\frac{1}{2} g(\mathbf{B}_{i,:}\boldsymbol{\beta})\right] \tag{6.7}$$

with $g(\mathbf{B}_{i,:}\boldsymbol{\beta})$ being a given function of $\mathbf{B}_{i,:}\boldsymbol{\beta}$. Generally, $\mathbf{B}\boldsymbol{\beta}$ in (6.7) is sparse, and therefore certain sparsity properties should be enforced on $\boldsymbol{\beta}$. To this effect, the function $g(\cdot)$ is usually chosen to be a concave, non-decreasing function of $|\mathbf{B}_{i,:}\boldsymbol{\beta}|$ [214]. Similarly, we introduce *super-Gaussian* priors $p(\mathbf{B}_{i,:}\boldsymbol{\beta})$, i.e.,

$$p(\mathbf{B}_{i,:}\boldsymbol{\beta}) = \max_{\gamma_i>0} \mathcal{N}(\mathbf{B}_{i,:}\boldsymbol{\beta}|0, \gamma_i)\varphi(\gamma_i), \tag{6.8}$$

or in the equivalent compact form

$$\begin{aligned} p(\mathbf{B}\boldsymbol{\beta}) &= \max_{\gamma_i>0} \prod_{i=1}^{\aleph} \mathcal{N}(\mathbf{B}_{i,:}\boldsymbol{\beta}|0,\gamma_i)\varphi(\gamma_i) \\ &= \max_{\boldsymbol{\gamma}\succ\mathbf{0}} \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|0,\boldsymbol{\Gamma})\varphi(\boldsymbol{\gamma}) \end{aligned} \tag{6.9}$$

where

$$\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_{\aleph}] \in \mathbb{R}^{\aleph},\ \boldsymbol{\Gamma} = \operatorname{diag}[\boldsymbol{\gamma}]\,. \tag{6.10}$$

6.3 Optimisation Problem Definition

Once we introduce the Gaussian likelihood in (6.1) and the variational prior in (6.9), the target is to maximise the marginal likelihood as

$$\int \mathcal{N}(\mathbf{y}|\mathbf{X}\boldsymbol{\beta},\boldsymbol{\Pi})\mathcal{N}(\mathbf{B}\boldsymbol{\beta}|\mathbf{0},\boldsymbol{\Gamma}) \prod_{i=1}^{\aleph} \varphi(\gamma_i) d\boldsymbol{\beta}. \tag{6.11}$$

We can get the following optimisation problem jointly on $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\Pi}$.

It is a bit abrupt to introduce the target and Proposition 4 below. Too many motivations are missing here, for example, why marginal likelihood maximisation is our target, why B is introduced, etc. In the subsequent Sections of this Chapter, these issues will be addressed. Many results are direct consequence of the following Proposition which will be directly referred.

Proposition 4 *The unknowns $\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Pi}$ can be obtained by solving the following optimisation problem*

$$\min_{\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}} \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}) \tag{6.12}$$

where

$$\begin{aligned} \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}) = (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\top} \boldsymbol{\Pi}^{-1} (\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \boldsymbol{\beta}^{\top}\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} \\ + \log|\boldsymbol{\Pi}| + \log|\boldsymbol{\Gamma}| + \log|\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X}| - \sum_{i=1}^{\aleph} \log\varphi(\gamma_i). \end{aligned} \tag{6.13}$$

Proof *To derive the cost function in* (6.13)*, we first introduce the posterior mean and variance*

$$\mathbf{m}_{\boldsymbol{\beta}} = \boldsymbol{\Sigma}_{\boldsymbol{\beta}} \mathbf{X}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{y}, \tag{6.14a}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\beta}} = (\mathbf{X}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{X} + \mathbf{B}^{\top} \boldsymbol{\Gamma}^{-1} \mathbf{B})^{-1}. \tag{6.14b}$$

Since the data likelihood $p(\mathbf{y}|\boldsymbol{\beta})$ is Gaussian, i.e.,

$$\mathcal{N}(\mathbf{y}|\mathbf{X}\boldsymbol{\beta}, \boldsymbol{\Pi}) = \frac{1}{(2\pi)^{M/2} |\boldsymbol{\Pi}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\top} \boldsymbol{\Pi}^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right], \tag{6.15}$$

we can write the marginal likelihood as

$$\begin{aligned} &\int \mathcal{N}(\mathbf{y}|\mathbf{X}\boldsymbol{\beta}, \boldsymbol{\Pi}) \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|\mathbf{0}, \boldsymbol{\Gamma}) \prod_{i=1}^{\aleph} \varphi(\gamma_i) d\boldsymbol{\beta} \\ =&\frac{1}{(2\pi)^{M/2} |\boldsymbol{\Pi}|^{1/2}} \frac{1}{(2\pi)^{\aleph}} \int \exp\{-E(\boldsymbol{\beta})\} d\boldsymbol{\beta} \prod_{i=1}^{\aleph} \varphi(\gamma_i), \end{aligned} \tag{6.16}$$

where

$$E(\boldsymbol{\beta}) = \frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\top} \boldsymbol{\Pi}^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{1}{2} \boldsymbol{\beta}^{\top} \mathbf{B}^{\top} \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta}. \tag{6.17}$$

Equivalently, we get

$$E(\boldsymbol{\beta}) = \frac{1}{2} (\boldsymbol{\beta} - \mathbf{m}_{\boldsymbol{\beta}})^{\top} \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{-1} (\boldsymbol{\beta} - \mathbf{m}_{\boldsymbol{\beta}}) + E(\mathbf{y}) \tag{6.18}$$

where $\mathbf{m}_{\boldsymbol{\beta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$ are given above.

We first show the data-dependent term is convex in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. From (6.14a) *and* (6.14b)*, the data-dependent term can be re-expressed as*[1]

$$\begin{aligned} E(\mathbf{y}) =&\frac{1}{2} \left(\mathbf{y}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{y} - \mathbf{y}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{X} \boldsymbol{\Sigma}_{\boldsymbol{\beta}} \mathbf{X}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{y}\right) \\ =&\frac{1}{2} \left(\mathbf{y}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{y} - \mathbf{y}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{X} \boldsymbol{\Sigma}_{\boldsymbol{\beta}} \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{\beta}} \mathbf{X}^{\top} \boldsymbol{\Pi}^{-1} \mathbf{y}\right) \\ =&\frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{m}_{\boldsymbol{\beta}})^{\top} \boldsymbol{\Pi}^{-1} (\mathbf{y} - \mathbf{X}\mathbf{m}_{\boldsymbol{\beta}}) + \frac{1}{2} \mathbf{m}_{\boldsymbol{\beta}}^{\top} \boldsymbol{\Gamma}^{-1} \mathbf{m}_{\boldsymbol{\beta}} \\ =&\min_{\boldsymbol{\beta}} \left[\frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\top} \boldsymbol{\Pi}^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{1}{2} \boldsymbol{\beta}^{\top} \mathbf{B}^{\top} \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta}\right]. \end{aligned} \tag{6.19}$$

[1]Woodbury inversion lemma does not apply here since $\mathbf{B}^{\top} \boldsymbol{\Gamma}^{-1} \mathbf{B}$ in $\boldsymbol{\Pi}$ may be not invertible.

Using (6.18)*, we can evaluate the integral in* (6.16)*) to obtain*

$$\int \exp\{-E(\boldsymbol{\beta})\}d\boldsymbol{\beta} = \exp\{-E(\mathbf{y})\}(2\pi)^{\aleph}|\boldsymbol{\Sigma}_{\beta}|^{1/2}. \tag{6.20}$$

Applying a $-2\log(\cdot)$ *transformation to* (6.16)*, we have*

$$\begin{aligned}
\hat{\mathcal{L}}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}) &= -2\log\left[\frac{1}{(2\pi)^{M/2}\,|\boldsymbol{\Pi}|^{1/2}}\frac{1}{(2\pi)^{\aleph}}\int \exp\{-E(\boldsymbol{\beta})\}d\boldsymbol{\beta}\prod_{i=1}^{\aleph}\varphi(\gamma_i)\right] \\
&= \mathbf{y}^{\top}\left[\boldsymbol{\Pi}^{-1} - \boldsymbol{\Pi}^{-1}\mathbf{X}(\mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X} + \mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B})^{-1}\mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\right]\mathbf{y} \\
&\quad + \log|\boldsymbol{\Pi}| + \log|\boldsymbol{\Gamma}| + \log|\mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X} + \mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}| \\
&\quad - \textstyle\sum_{i=1}^{\aleph}\log\varphi(\gamma_i) + (M + 2\aleph)\log 2\pi \\
&= \min_{\boldsymbol{\beta}}\left[(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\top}\boldsymbol{\Pi}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \boldsymbol{\beta}^{\top}\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}\right] \\
&\quad + \log|\boldsymbol{\Pi}| + \log|\boldsymbol{\Gamma}| + \log|\mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X} + \mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}| \\
&\quad - \textstyle\sum_{i=1}^{\aleph}\log\varphi(\gamma_i) + (M + 2\aleph)\log 2\pi.
\end{aligned} \tag{6.21}$$

Therefore we get the following cost function to be minimised in (6.13) *over* $\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}$

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}) &= (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\top}\boldsymbol{\Pi}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \boldsymbol{\beta}^{\top}\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} \\
&\quad + \log|\boldsymbol{\Pi}| + \log|\boldsymbol{\Gamma}| + \log|\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X}| - \sum_{i=1}^{\aleph}\log\varphi(\gamma_i).
\end{aligned}$$

Then we have

$$\min_{\boldsymbol{\beta},\boldsymbol{\Pi},\boldsymbol{\Gamma}}\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}) = \min_{\boldsymbol{\beta},\boldsymbol{\Pi},\boldsymbol{\Gamma}}\hat{\mathcal{L}}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}). \tag{6.22}$$

It should be notice that the explicit form of $\varphi(\gamma_i)$ *is typically not available, thus it is usually chosen as non-informative as* $\varphi(\gamma_i) = 1$ *which is a consequence of specifying* Student's t *prior. Then we have*

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\Pi}) &= (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\top}\boldsymbol{\Pi}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \boldsymbol{\beta}^{\top}\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} \\
&\quad + \log|\boldsymbol{\Pi}| + \log|\boldsymbol{\Gamma}| + \log|\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X}|.
\end{aligned} \tag{6.23}$$

■

Next, we the show that the stated program can be formulated as a convex-concave procedure (CCCP).

Proposition 5 *The following programme*

$$\min_{\boldsymbol{\beta},\boldsymbol{\Pi},\boldsymbol{\Gamma}} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top}\boldsymbol{\Pi}^{-1}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)+\boldsymbol{\beta}^{\top}\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} + \log|\boldsymbol{\Pi}|+\log|\boldsymbol{\Gamma}|+\log|\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}+\mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X}|. \quad (6.24)$$

can be formulated as a convex-concave procedure (CCCP).

Proof Fact on convexity: *the function*

$$u(\boldsymbol{\beta},\boldsymbol{\Pi},\boldsymbol{\Gamma})=\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top}\boldsymbol{\Pi}^{-1}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)+\boldsymbol{\beta}^{\top}\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} \quad (6.25)$$

is convex jointly in $\boldsymbol{\beta}$, $\boldsymbol{\Pi}$, $\boldsymbol{\Gamma}$ due to the fact that $f(\boldsymbol{\beta},Y)=\boldsymbol{\beta}^{\top}Y^{-1}\boldsymbol{\beta}$ is jointly convex in $\boldsymbol{\beta}$, Y (see, [28, p.76]). Hence u as a sum of convex functions is convex.

Fact on concavity: *the function*

$$v(\boldsymbol{\Pi},\boldsymbol{\Gamma})=\log|\boldsymbol{\Pi}|+\log|\boldsymbol{\Gamma}|+\log|\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}+\mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X}| \quad (6.26)$$

is jointly concave in $\boldsymbol{\Gamma}$, $\boldsymbol{\Pi}$. We exploit the properties of the determinant of a matrix

$$|A_{22}||A_{11}-A_{12}A_{22}^{-1}A_{21}|=\left|\begin{pmatrix}A_{11}&A_{12}\\A_{21}&A_{22}\end{pmatrix}\right|=|A_{11}||A_{22}-A_{21}A_{11}^{-1}A_{12}|.$$

Then we have

$$\begin{aligned}
v(\boldsymbol{\Pi},\boldsymbol{\Gamma}) &= \log|\boldsymbol{\Pi}|+\log|\boldsymbol{\Gamma}|+\log|\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B}+\mathbf{X}^{\top}\boldsymbol{\Pi}^{-1}\mathbf{X}| \\
&= \log|\boldsymbol{\Gamma}|+\log\left|\begin{pmatrix}\boldsymbol{\Gamma}&\mathbf{0}\\\mathbf{0}&\boldsymbol{\Pi}\end{pmatrix}\right|+\log\left|\begin{pmatrix}\mathbf{B}^{\top}&\mathbf{X}^{\top}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Gamma}&\mathbf{0}\\\mathbf{0}&\boldsymbol{\Pi}\end{pmatrix}^{-1}\begin{pmatrix}\mathbf{B}\\\mathbf{X}\end{pmatrix}\right| \\
&= \log|\boldsymbol{\Gamma}|+\log\left(\left|\begin{pmatrix}\boldsymbol{\Gamma}&\mathbf{0}\\\mathbf{0}&\boldsymbol{\Pi}\end{pmatrix}\right|\left|\begin{pmatrix}\mathbf{B}^{\top}&\mathbf{X}^{\top}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Gamma}&\mathbf{0}\\\mathbf{0}&\boldsymbol{\Pi}\end{pmatrix}^{-1}\begin{pmatrix}\mathbf{B}\\\mathbf{X}\end{pmatrix}\right|\right) \\
&= \log|\boldsymbol{\Gamma}|+\log\left(\left|\begin{pmatrix}\boldsymbol{\Gamma}&\mathbf{0}\\\mathbf{0}&\boldsymbol{\Pi}\end{pmatrix}+\begin{pmatrix}\mathbf{B}\\\mathbf{X}\end{pmatrix}\boldsymbol{\Gamma}\begin{pmatrix}\mathbf{B}^{\top}&\mathbf{X}^{\top}\end{pmatrix}\right|\right) \\
&= \log\left(\left|\begin{pmatrix}\boldsymbol{\Gamma}&\mathbf{0}\\\mathbf{0}&\boldsymbol{\Pi}\end{pmatrix}+\begin{pmatrix}\mathbf{B}\\\mathbf{X}\end{pmatrix}\boldsymbol{\Gamma}\begin{pmatrix}\mathbf{B}^{\top}&\mathbf{X}^{\top}\end{pmatrix}\right|\right),
\end{aligned} \quad (6.27)$$

which is a log*-determinant of an affine function of semidefinite matrices $\boldsymbol{\Pi}$, $\boldsymbol{\Gamma}$ and hence concave.* ■

Remark 13 *It can be found that in* (6.24) *of Proposition 5, we drop the term* $-2\sum_{i=1}^{\aleph}\log\varphi(\gamma_i)$ *from* (6.13) *in Proposition 4. From Proposition 3,* $\varphi(\gamma_i)$ *can be expressed as* $\varphi(\gamma_i) = \sqrt{2\pi/\gamma_i}\exp\left(g^*\left(\gamma_i/2\right)\right)$. *Therefore*

$$-2\sum_{i=1}^{\aleph}\log\varphi(\gamma_i) = \sum_{i=1}^{\aleph}(-\log 2\pi + \log\gamma_i - 2g^*\left(\gamma_i/2\right)) \propto \log|\boldsymbol{\Gamma}| - 2\sum_{i=1}^{\aleph} g^*\left(\gamma_i/2\right)$$

Then we can write $\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta})$ *in* (7.16) *as*

$$\hat{\mathcal{L}}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) = \hat{u}\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right) + \hat{v}(\boldsymbol{\Gamma}) \tag{6.28}$$

where

$$\begin{aligned}\hat{u}\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right) &= u\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right) - 2\sum_{i=1}^{\aleph} g^*\left(\gamma_i/2\right)\\ \hat{v}(\boldsymbol{\Gamma}) &= v(\boldsymbol{\Gamma}) + \log|\boldsymbol{\Gamma}|\end{aligned} \tag{6.29}$$

Since $g^*\left(\gamma_i/2\right)$ *is the concave conjugate of* $g(\cdot)$, $-2\sum_{i=1}^{\aleph} g^*\left(\gamma_i/2\right)$ *is convex in* γ_i *[28, pp.91]. Also, nonnegative weighted sums of convex functions preserve convexity [28, pp.79]. Therefore,* $\hat{u}\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right)$ *is convex in* $\boldsymbol{\gamma}$ *as well. Similarly, since* $\log|\boldsymbol{\Gamma}|$ *is concave in* $\boldsymbol{\gamma}$, $\hat{v}(\boldsymbol{\Gamma})$ *is concave in* $\boldsymbol{\gamma}$.

Like we mentioned in Remark 12, for the Laplace *prior* $p(\beta_i) \propto \exp(-\lambda\sum_j|w_i|)$ *where one possible* Laplace *potential function* $\varphi(\gamma_i) = \exp\left(-\frac{1}{2}|\gamma_i|\right)\sqrt{2\pi|\gamma_i|}$. *Therefore,*

$$\begin{aligned}\hat{u}\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right) &= u\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right) + \sum_{i=1}^{\aleph}|\gamma_i|,\\ \hat{v}(\boldsymbol{\Gamma}) &= v(\boldsymbol{\Gamma}) + \log|\boldsymbol{\Gamma}|.\end{aligned} \tag{6.30}$$

However, for the Student's t *prior* $p(\beta_i) \propto (b + \frac{{\beta_i}^2}{2})^{-(a+\frac{1}{2})}$, *one can have a* Student's t *potential function* $\varphi(\gamma) = 1$, *when* $a, b \to 0$. *We have*

$$\hat{u}\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right) = u\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right),\ \hat{v}(\boldsymbol{\Gamma}) = v(\boldsymbol{\Gamma}).$$

6.4 Optimisation Principle

Given Proposition 5, we can derive the iterative algorithm solving the CCCP. We have the following iterative convex optimisation program by calculating the gradient

of concave part.

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}, \boldsymbol{\gamma}^k, \boldsymbol{\Pi}), \tag{6.31}$$

$$\boldsymbol{\gamma}^{k+1} = \underset{\boldsymbol{\gamma}\succeq\mathbf{0}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}^k, \boldsymbol{\gamma}, \boldsymbol{\Pi}) + \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k, \boldsymbol{\Pi})^\top \boldsymbol{\gamma}, \tag{6.32}$$

Suppose $\boldsymbol{\beta}^*$ and $\boldsymbol{\gamma}^*$ are the estimate to $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ while certain convergence criteria is met or k reaches to the pre-defined maximum number, $\boldsymbol{\Pi}$ will then be estimated.

$$\boldsymbol{\Pi}^* = \underset{\boldsymbol{\Pi}\succeq\mathbf{0}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\Pi}) + v(\boldsymbol{\gamma}^*, \boldsymbol{\Pi}). \tag{6.33}$$

Inspired by implementation of Generalised Method of Moment (GMM) [76, 77], a work winning Nobel Prize in Economics by Lars Hansen, we propose two ways to optimise for $\boldsymbol{\theta}$.

The first one is inspired by two-step feasible GMM, where after the find estimation of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, i.e., at the iteration $k = k_{\text{end}}$ of the CCCP (6.31) and (6.32)

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}, \boldsymbol{\gamma}^k, \boldsymbol{\Pi}),\\
\boldsymbol{\gamma}^{k+1} &= \underset{\boldsymbol{\gamma}\succeq\mathbf{0}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}^k, \boldsymbol{\gamma}, \boldsymbol{\Pi}) + \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k, \boldsymbol{\Pi})^\top \boldsymbol{\gamma},
\end{aligned}$$

we have

$$\boldsymbol{\Pi}^* = \underset{\boldsymbol{\Pi}\succeq\mathbf{0}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\Pi}) + v(\boldsymbol{\gamma}^*, \boldsymbol{\Pi}) \tag{6.34}$$

The second on is inspired by iterated GMM, where at each iteration k of the CCCP (6.31) and (6.32), we perform

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}, \boldsymbol{\gamma}^k, \boldsymbol{\Pi}),\\
\boldsymbol{\gamma}^{k+1} &= \underset{\boldsymbol{\gamma}\succeq\mathbf{0}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}^k, \boldsymbol{\gamma}, \boldsymbol{\Pi}) + \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k, \boldsymbol{\Pi})^\top \boldsymbol{\gamma},\\
\boldsymbol{\Pi}^{k+1} &= \underset{\boldsymbol{\Pi}\succeq\mathbf{0}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}, \boldsymbol{\Pi}) + v(\boldsymbol{\gamma}^{k+1}, \boldsymbol{\Pi}).
\end{aligned} \tag{6.35}$$

Remark 14 *Revisit Proposition 4. At first instance, we may argue if it is necessary to come up with Proposition 5 and subsequent alternative optimisation using CCCP. One can actually optimise for $\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Pi}$ simultaneously. For example using Monte-Carlo methods, if computation resource allowed, this method will most likely demonstrate a better performance than the traditional iterative optimisation method.*

6.5 Optimisation Algorithm

6.5.1 Iterative Reweighted ℓ_1 Algorithm

First, we fix/give the known inverse covariance matrix $\mathbf{\Pi}^{-1} = \mathbf{\Theta} = \mathbf{\Theta}^*$. Using basic principles in convex analysis, we then obtain the following analytic form for the negative gradient of $v(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}$ is (using chain rule):

$$\begin{aligned}\boldsymbol{\alpha}^k \triangleq & \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}, \mathbf{\Pi})^\top|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k} \\ = & \nabla_{\boldsymbol{\gamma}} \left(-\log|\mathbf{\Theta}^*| + \log|\mathbf{\Gamma}| + \log|\mathbf{B}^\top\mathbf{\Gamma}^{-1}\mathbf{B} + \mathbf{X}^\top\mathbf{\Theta}^*\mathbf{X}|\right)^\top|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k} \\ = & -\operatorname{diag}\{(\mathbf{\Gamma}^k)^{-1}\} \circ \operatorname{diag}\{\mathbf{B}(\mathbf{B}^\top(\mathbf{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{X}^\top\mathbf{\Theta}^*\mathbf{X})^{-1}\mathbf{B}^\top\} \circ \operatorname{diag}\{(\mathbf{\Gamma}^k)^{-1}\} \\ & + \operatorname{diag}\{(\mathbf{\Gamma}^k)^{-1}\}, \\ = & \begin{bmatrix} \alpha_1^k & \cdots & \alpha_N^k \end{bmatrix}.\end{aligned} \tag{6.36}$$

where $\circ$ denote the Hadamard product operation (entrywise multiplication) and diag denote the operation to get diagonal elements of matrix. Then equivalently, we have

$$\alpha_i^k = -\frac{\mathbf{B}_{i,:}(\mathbf{B}^\top(\mathbf{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{X}^\top\mathbf{\Theta}^*\mathbf{X})^{-1}\mathbf{B}_{i,:}^\top}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}. \tag{6.37}$$

Therefore, the iterative procedures (6.31) and (6.32) for $\boldsymbol{\beta}^{k+1}$ and $\boldsymbol{\gamma}^{k+1}$ can be formulated as

$$\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] = \underset{\boldsymbol{\gamma}\succeq\mathbf{0},\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \mathbf{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \sum_{i=1}^{\aleph} \left(\frac{\boldsymbol{\beta}^\top\mathbf{B}_{i,:}^\top\mathbf{B}_{i,:}\boldsymbol{\beta}}{\gamma_i} + \alpha_i^k\gamma_i\right). \tag{6.38}$$

Or in the compact form

$$\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] = \underset{\boldsymbol{\gamma}\succeq\mathbf{0},\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \mathbf{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \boldsymbol{\beta}^\top\mathbf{B}^\top\mathbf{\Gamma}^{-1}\mathbf{B} + \sum_{i=1}^{\aleph} \alpha_i^k\gamma_i. \tag{6.39}$$

Since

$$\frac{\boldsymbol{\beta}^\top\mathbf{B}_{i,:}^\top\mathbf{B}_{i,:}\boldsymbol{\beta}}{\gamma_i} + \alpha_i^k\gamma_i \geq 2\left|\sqrt{\alpha_i^k}\cdot\mathbf{B}_{i,:}\boldsymbol{\beta}\right|,$$

the optimal $\boldsymbol{\gamma}$ can be obtained as:

$$\gamma_i = \frac{|\mathbf{B}_{i,:}\boldsymbol{\beta}|}{\sqrt{\alpha_i^k}}, \forall i. \tag{6.40}$$

From (6.36), it is found that α_i^k is a function of γ_i^k. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}$ first to calculate $\boldsymbol{\gamma}^{k+1}$. If we define

$$w_i^k \triangleq \sqrt{\alpha_i^k} = \sqrt{-\frac{\mathbf{B}_{i,:}(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X})^{-1}\mathbf{B}_{i,:}^\top}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}}, \tag{6.41}$$

$\boldsymbol{\beta}^{k+1}$ can be obtained as follows

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \boldsymbol{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \sum_{i=1}^{\aleph} \|w_i^k \cdot \mathbf{B}_{i,:}\boldsymbol{\beta}\|_{\ell_1}. \tag{6.42}$$

We can then inject this into (6.40), which yields

$$\gamma_i^{k+1} = \frac{|\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}|}{w_i^k}, \forall i. \tag{6.43}$$

As we found in the expression for α_i in (6.37), α_i^k is function of γ^k, therefore γ^{k+1} is function of γ^k and $\boldsymbol{\beta}^{k+1}$. We notice that the update for $\boldsymbol{\beta}^{k+1}$ is to use ℓ_1-*regularised* regression type optimisation. The pseudo code is summarised in Algorithm 1.

Algorithm 1 Reweighted ℓ_1 type algorithm for Gaussian likelihood

Initialise the unknown $\mathbf{w}$ as a unit vector;
Fix/given the known inverse covariance matrix $\boldsymbol{\Pi}^{-1} = \boldsymbol{\Theta} = \boldsymbol{\Theta}^*$;
Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
for $k = 1, \ldots, k_{\max}$ **do**

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \boldsymbol{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \lambda \sum_{i=1}^{\aleph} \|w_i^k \cdot \mathbf{B}_{i,:}\boldsymbol{\beta}\|_{\ell_1}; \tag{6.44}$$

  $\gamma_i^{k+1} = \left|\frac{\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}}{w_i^k}\right|$;
  $\mathbf{C}^{k+1} = \left(\mathbf{B}^\top(\boldsymbol{\Gamma}^{k+1})^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X}\right)^{-1}$;
  ${\alpha_i}^{k+1} = -\frac{\mathbf{B}_{i,:}\mathbf{C}^{k+1}\mathbf{B}_{i,:}^\top}{(\gamma_i^{k+1})^2} + \frac{1}{\gamma_i^{k+1}}$;
  ${w_i}^{k+1} = \sqrt{{\alpha_i}^{k+1}}$;
  if a stopping criterion is satisfied **then**
    Break.
  end if
end for

Remark 15 *The above derivation is essentially to employ Student-t distribution. Now we consider the Laplace distribution to see the difference. As in* (6.30)

$$\hat{u}\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right)=u\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right)+\sum_{i=1}^{\aleph}|\gamma_i|,$$
$$\hat{v}(\boldsymbol{\Gamma})=v(\boldsymbol{\Gamma})+\log|\boldsymbol{\Gamma}|.$$

We derive a new $\boldsymbol{\alpha}^k$

$$\begin{aligned}\boldsymbol{\alpha}^k \triangleq&\nabla_{\boldsymbol{\gamma}}\hat{v}(\boldsymbol{\gamma},\boldsymbol{\Pi})^\top|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k}\\ =&\nabla_{\boldsymbol{\gamma}}\left(-\log|\boldsymbol{\Theta}^*|+2\log|\boldsymbol{\Gamma}|+\log|\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}+\mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X}|\right)^\top|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k}\\ =&-\operatorname{diag}\{(\boldsymbol{\Gamma}^k)^{-1}\}\circ\operatorname{diag}\{\mathbf{B}(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B}+\mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X})^{-1}\mathbf{B}^\top\}\circ\operatorname{diag}\{(\boldsymbol{\Gamma}^k)^{-1}\}\\ &+2\cdot\operatorname{diag}\{(\boldsymbol{\Gamma}^k)^{-1}\},\\ =&\left[\begin{array}{ccc}\alpha_1^k&\cdots&\alpha_N^k\end{array}\right].\end{aligned}\tag{6.45}$$

Then joint optimisation problem (6.39) *can be reformulated as*

$$\left[\boldsymbol{\beta}^{k+1},\boldsymbol{\gamma}^{k+1}\right]=\underset{\boldsymbol{\gamma}\succeq\mathbf{0},\boldsymbol{\beta}}{\operatorname{argmin}}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^\top\boldsymbol{\Theta}^*\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)+\boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}+\sum_{i=1}^{\aleph}|\gamma_i|+\sum_{i=1}^{\aleph}\alpha_i^k\gamma_i.$$

Since $\boldsymbol{\gamma}\succeq\mathbf{0}$*, we remove* $|\cdot|$ *of* $|\gamma_i|$ *and get*

$$\left[\boldsymbol{\beta}^{k+1},\boldsymbol{\gamma}^{k+1}\right]=\underset{\boldsymbol{\gamma}\succeq\mathbf{0},\boldsymbol{\beta}}{\operatorname{argmin}}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^\top\boldsymbol{\Theta}^*\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)+\boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}+\sum_{i=1}^{\aleph}(\alpha_i^k+1)\gamma_i.$$

Then w_i^k *in* (6.41) *becomes*

$$w_i^k=\sqrt{\alpha_i^k+1}.$$

It can be found that Laplace prior regularised more than Student's-t prior by adding more weight.

6.5.2 Iterative Reweighted ℓ_2 Algorithm

Again, we first fix/give the known inverse covariance matrix $\boldsymbol{\Pi}^{-1}=\boldsymbol{\Theta}=\boldsymbol{\Theta}^*$. In (6.38), instead of formulating a convex program for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ jointly, they are

optimised respectively:

$$\begin{aligned}\boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top} \boldsymbol{\Pi}^{-1} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right) + \sum_{i=1}^{\aleph} \left\| \frac{\mathbf{B}_{i,:}\boldsymbol{\beta}}{\sqrt{\gamma_i^k}} \right\|_{\ell_2}^2 \\ &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top} \boldsymbol{\Pi}^{-1} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right) + \boldsymbol{\beta}^{\top}\mathbf{B}^{\top}(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B}\boldsymbol{\beta}, \qquad (6.46)\end{aligned}$$

$$\gamma_i^{k+1} = \underset{\gamma_i \geq 0}{\operatorname{argmin}} \frac{(\boldsymbol{\beta}^{k+1})^{\top}\mathbf{B}_{i,:}^{\top}\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}}{\gamma_i} + \alpha_i^k\gamma_i, \forall i. \qquad (6.47)$$

Once $\boldsymbol{\beta}^{k+1}$ is obtained, $\boldsymbol{\gamma}^{k+1}$ has a closed form solution to (6.47), i.e.,

$$\gamma_i^{k+1} = \sqrt{\frac{(\boldsymbol{\beta}^{k+1})^{\top}\mathbf{B}_{i,:}^{\top}\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}}{\alpha_i^k}},$$

where α_i^k is the same as (6.37)

$$\alpha_i^k = -\frac{\mathbf{B}_{i,:}(\mathbf{B}^{\top}(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{X}^{\top}\boldsymbol{\Theta}^*\mathbf{X})^{-1}\mathbf{B}_{i,:}^{\top}}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}.$$

But unlike (6.41), define

$$w_i^k \triangleq \frac{1}{\sqrt{\gamma_i^k}}.$$

The pseudo code is summarised in Algorithm 2.

Remark 16 *Similar to Remark 15 for Laplace prior, in ℓ_2 regularised algorithm, we can only change* (6.47) *as*

$$\gamma_i^{k+1} = \underset{\gamma_i \geq 0}{\operatorname{argmin}} \frac{(\boldsymbol{\beta}^{k+1})^{\top}\mathbf{B}_{i,:}^{\top}\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}}{\gamma_i} + (\alpha_i^k + 1)\gamma_i, \forall i.$$

for Laplace prior.

6.5.3 Inverse Covariance Matrix Estimation

Remind the statement in Proposition 4. Once unknowns $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are obtained as $\boldsymbol{\beta}^*$ and $\boldsymbol{\gamma}^*$, we can proceed with the optimisation for the covariance matrix $\boldsymbol{\Pi}$

$$\min_{\boldsymbol{\Pi}} \mathcal{L}(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\Pi}) \qquad (6.49)$$

Algorithm 2 Reweighted ℓ_2 type algorithm for Gaussian likelihood

Initialise the unknown hyperparameter $\boldsymbol{\gamma}$ as a unit vector;
Fix/given the known inverse covariance matrix $\boldsymbol{\Pi}^{-1} = \boldsymbol{\Theta} = \boldsymbol{\Theta}^*$;
Initialise $w_i^1 = 1,\ \forall i$;
Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
for $k = 1, \ldots, k_{\max}$ **do**

$$\begin{aligned}\boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \boldsymbol{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \lambda \boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1} \mathbf{B}\boldsymbol{\beta} \\ &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \boldsymbol{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \lambda \sum_{i=1}^{\aleph} \|w_i^k \cdot \mathbf{B}_{i,:}\boldsymbol{\beta}\|_{\ell_2}^2;\end{aligned} \tag{6.48}$$

  $\mathbf{C}^k = \left(\mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1} \mathbf{B} + \mathbf{X}^\top \boldsymbol{\Theta}^* \mathbf{X}\right)^{-1}$;
  $\alpha_i^{k+1} = -\frac{\mathbf{B}_{i,:}\mathbf{C}^k\mathbf{B}_{i,:}^\top}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}$;
  $\gamma_i^{k+1} = \frac{\left|\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}\right|}{\sqrt{\alpha_i^{k+1}}}$;
  $w_i^{k+1} = \frac{1}{\sqrt{\gamma_i^{k+1}}}$;
  if a stopping criterion is satisfied **then**
    Break.
  end if
end for

where

$$\begin{aligned}\mathcal{L}(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\Pi}) = &\left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*\right)^\top \boldsymbol{\Pi}^{-1} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*\right) + (\boldsymbol{\beta}^*)^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^*)^{-1} \mathbf{B}\boldsymbol{\beta}^* \\ &+ \log|\boldsymbol{\Pi}| + \log|\boldsymbol{\Gamma}^*| + \log|\mathbf{B}^\top (\boldsymbol{\Gamma}^*)^{-1}\mathbf{B} + \mathbf{X}^\top \boldsymbol{\Pi}^{-1}\mathbf{X}| - \sum_{i=1}^{\aleph} \log \varphi(\gamma_i^*).\end{aligned} \tag{6.50}$$

Since

$$\boldsymbol{\Theta} = \boldsymbol{\Pi}^{-1},$$

we can re-write the optimisation problem with a new cost function over $\boldsymbol{\Theta}$ and remove the constant terms in (6.50)

$$\min_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\Theta}) \tag{6.51}$$

where

$$\mathcal{L}(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\Theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*)^\top \boldsymbol{\Theta} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*) - \log|\boldsymbol{\Theta}| + \log|\mathbf{B}^\top(\boldsymbol{\Gamma}^*)^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}\mathbf{X}|. \quad (6.52)$$

By letting

$$\mathbf{Y}^* = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*) \cdot (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*)^\top,$$

we can further re-write (6.52) as

$$\mathcal{L}(\boldsymbol{\Theta}) = \mathrm{Tr}\,(\boldsymbol{\Theta}\mathbf{Y}^*) - \log|\boldsymbol{\Theta}| + \log|\mathbf{B}^\top(\boldsymbol{\Gamma}^*)^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}\mathbf{X}|. \quad (6.53)$$

We find that $\mathrm{Tr}\,(\boldsymbol{\Theta}\mathbf{Y}^*) - \log|\boldsymbol{\Theta}|$ is convex over the semidefinite matrix $\boldsymbol{\Theta}$ and $\log|\mathbf{B}^\top(\boldsymbol{\Gamma}^*)^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}\mathbf{X}|$ is concave over the semidefinite matrices $\boldsymbol{\Theta}$. As the CCCP procedure over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, we can also design an iterative CCCP over $\boldsymbol{\Theta}$. At the k-th iteration, we get the gradient of $\log|\mathbf{B}^\top(\boldsymbol{\Gamma}^*)^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}\mathbf{X}|$ as Λ^k

$$\begin{aligned}\Lambda^k &= \nabla_{\boldsymbol{\Theta}} \left(\log\det\left(\mathbf{B}^\top\boldsymbol{\Gamma}^{-k}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}^k\mathbf{X}\right)\right) \\ &= \mathbf{X}(\mathbf{B}^\top\boldsymbol{\Gamma}^{-k}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}^k\mathbf{X})^{-1}\mathbf{X}^\top.\end{aligned} \quad (6.54)$$

Then we have the following iterative algorithm to estimate the inverse covariance matrix

$$\boldsymbol{\Theta}^{k+1} = \underset{\boldsymbol{\Theta}\succeq\mathbf{0}}{\mathrm{argmin}}\, \mathrm{Tr}\left(\boldsymbol{\Theta}\mathbf{Y}^k\right) - \log|\boldsymbol{\Theta}| + \mathrm{Tr}\left(\Lambda^k\boldsymbol{\Theta}\right). \quad (6.55)$$

6.5.4 Volatility Estimation

In this Section, we discuss the estimation of the parameters associated with the "noise" term. Two classic model class will be considered, i.e., autoregressive moving average with exogenous input (ARMAX) and autoregressive conditional heteroskedasticity (ARCH).

Discussion on ARMAX Model

Recall the ARMAX model structure defined in (2.15), i.e.,

$$\begin{aligned}&y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) \\ =&b_1 u(t-1) + \ldots + b_{n_b} u(t-n_b) + e(t) + c_1 e(t-1) + \ldots + c_{n_e} e(t-n_c).\end{aligned}$$

Once $\boldsymbol{\beta}$ is estimated as $\hat{a}_i$ and $\hat{b}_j$, where $i = 1, \ldots, n_a$, $j = 1, \ldots, n_b$, one can define the quantity

$$\hat{y}(t) \triangleq y(t) + \hat{a}_1 y(t-1) + \ldots + \hat{a}_{n_a} y(t-n_a) - \hat{b}_1 u(t-1) - \ldots - \hat{b}_{n_b} u(t-n_b)$$

and have

$$\hat{y}(t) = e(t) + c_1 e(t-1) + \ldots + c_{n_e} e(t-n_c). \tag{6.56}$$

Obviously, (6.56) is a moving average process. Suppose $\mathbf{c} = [c_1, \ldots, c_{n_e}]$, the exact likelihood function is given by

$$f(\hat{\mathbf{y}}, \mathbf{c}) = (2\pi)^{-T/2} |\boldsymbol{\Pi}|^{-1/2} \exp\left[-\frac{1}{2}\hat{\mathbf{y}}^\top \boldsymbol{\Pi}^{-1} \hat{\mathbf{y}}\right] \tag{6.57}$$

where as before $\hat{\mathbf{y}} = [\hat{y}(1), \ldots, \hat{y}(T)]$. Here $\boldsymbol{\Pi}$ represents the variance-covariance matrix of T consecutive draws from an moving average process

$$\boldsymbol{\Pi} = \mathbf{C}^\top \mathbf{C}$$

where

$$\mathbf{C} = \begin{bmatrix} 1 & c_1 & c_2 & \cdots & c_{n_e} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & c_1 & \cdots & c_{n_e-1} & c_{n_e} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & c_1 & \cdots & c_{n_e} \end{bmatrix} \in \mathbb{R}^{T \times (T+n_e)} \tag{6.58}$$

This is the same as eq.(5.5.6) in [73, pp. 130]. The row i, column j element of $\boldsymbol{\Pi}$ is given by $\gamma_{|i-j|}$, where γ_k is the kth autocovariance of an moving average process, i.e., $MA(q)$ process:

$$\gamma_k = \begin{cases} \sigma^2(c_k + c_{k+1}c_1 + c_{k+2}c_2 + \cdots + c_q c_{q-k}) & \text{for } k = 0, 1, \ldots, q \\ 0 & \text{for } k > q, \end{cases}$$

where $c_0 = 1$.

This is consistent with the the optimisation program in (6.55) but without the regularisation term $\mathrm{Tr}\left(\Lambda^k \boldsymbol{\Theta}\right)$, which is *Maximum Likelihood Estimation*

$$\boldsymbol{\Theta}^{k+1} = \underset{\boldsymbol{\Theta} \succeq \mathbf{0}}{\operatorname{argmin}} \, \mathrm{Tr}\left(\boldsymbol{\Theta}\mathbf{Y}^k\right) - \log|\boldsymbol{\Theta}|. \tag{6.59}$$

Discussion on ARCH Model

To model a time series using an ARCH process, let ϵ_t denote the error terms (return residuals, with respect to a mean process), i.e. the series terms. These ϵ_t are split into a stochastic piece z_t and a time-dependent standard deviation σ_t characterizing the typical size of the terms so that

$$\epsilon_t = \sigma_t z_t$$

The random variable z_t is a strong white noise process. The series σ_t^2 is modelled by

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2$$

where $\alpha_0 > 0$ and $\alpha_i \geq 0,\ i > 0$.

An ARCH(q) model can be estimated using ordinary least squares. A methodology to test for the lag length of ARCH errors using the Lagrange multiplier test was proposed by Robert Engle [56]. This procedure is as follows:

1. Estimate the best fitting autoregressive model AR(q)

$$y_t = a_0 + a_1 y_{t-1} + \cdots + a_q y_{t-q} + \epsilon_t = a_0 + \sum_{i=1}^{q} a_i y_{t-i} + \epsilon_t.$$

2. Obtain the squares of the error $\hat{\epsilon}^2$ and regress them on a constant and q lagged values:

$$\hat{\epsilon}_t^2 = \hat{\alpha}_0 + \sum_{i=1}^{q} \hat{\alpha}_i \hat{\epsilon}_{t-i}^2$$

   where q is the length of ARCH lags.

3. The null hypothesis is that, in the absence of ARCH components, we have $\alpha_i = 0$ for all $i = 1, \cdots, q$. The alternative hypothesis is that, in the presence of ARCH components, at least one of the estimated α_i coefficients must be significant. In a sample of T residuals under the null hypothesis of no ARCH errors, the test statistic $T'R^2$ follows χ^2 distribution with q degrees of freedom, where T' is the number of equations in the model which fits the residuals vs the lags (i.e. $T' = T - q$). If $T'R^2$ is greater than the Chi-square table value, we "reject" the null hypothesis and conclude there is an ARCH effect in the ARMA model. If $T'R^2$ is smaller than the Chi-square table value, we do not reject the null hypothesis.

In particular for Step 2, algorithms proposed in Sections 6.5.1 and 6.5.2 can be applied to estimate α_i, $i = 1, \ldots, q$ and select the order q which is favourable to be sparse.

6.6 Algorithms for Chapter 2

Let's revisit the nonconvex optimisation problem in Section 2.4.2

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_{\ell_0},$$

where λ is the regularisation parameter.

This is the simplest case in this thesis. To begin with, the motivations and details will be gone through step by step. Several different ways of derivation will be introduced to get more insights. It should be noted that some contents in this Section may be *partially* overlapping with previous Sections in this Chapter, e.g., Section 6.6.2 with Section 6.3, Section 6.6.3 with Section 6.5. To be coherent and provide more insights, these overlapped contents will remain as they are. In the end, a distributed optimisation framework is proposed based on ADMM technique [27].

6.6.1 Sparse Prior for Chapter 2

By setting $\mathbf{B}$ in (6.9) to identity matrix, the sparse prior is simply

$$p(\boldsymbol{\beta}) = \max_{\gamma_j > 0} \prod_{j=1}^{N} \mathcal{N}(\beta_j|0, \gamma_j)\varphi(\gamma_j). \tag{6.60}$$

For a fixed $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_N]$, we define a relaxed prior which is a joint probability distribution over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as

$$p(\boldsymbol{\beta}; \boldsymbol{\gamma}) = \prod_j \mathcal{N}(\beta_j|0, \gamma_j)\varphi(\gamma_j) = p(\boldsymbol{\beta}|\boldsymbol{\gamma})p(\boldsymbol{\gamma}) \leq p(\boldsymbol{\beta}) \tag{6.61}$$

where

$$p(\boldsymbol{\beta}|\boldsymbol{\gamma}) \triangleq \prod_j \mathcal{N}(\beta_j|0, \gamma_j), \quad p(\boldsymbol{\gamma}) \triangleq \prod_j \varphi(\gamma_j). \tag{6.62}$$

6.6.2 Optimisation Problem Derivation

Since the likelihood is $p(\mathbf{y}|\boldsymbol{\beta})$ is Gaussian, we can get a relaxed posterior which is also Gaussian

$$p(\boldsymbol{\beta}|\mathbf{y},\boldsymbol{\gamma}) = \frac{p(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta};\boldsymbol{\gamma})}{\int p(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta};\boldsymbol{\gamma})d\boldsymbol{\beta}} = \mathcal{N}(\mathbf{m}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}). \tag{6.63}$$

Defining $\boldsymbol{\Gamma} \triangleq \mathrm{diag}[\boldsymbol{\gamma}]$, the posterior mean and covariance are given by:

$$\mathbf{m}_{\boldsymbol{\beta}} = \boldsymbol{\Gamma}\mathbf{X}^\top(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{y}, \tag{6.64}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\beta}} = \boldsymbol{\Gamma} - \boldsymbol{\Gamma}\mathbf{X}^\top(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{X}. \tag{6.65}$$

Now the key question is how to choose the most appropriate $\boldsymbol{\gamma} = \hat{\boldsymbol{\gamma}} = [\hat{\gamma}_1, \ldots, \hat{\gamma}_N]$ to maximise $\prod_j \mathcal{N}(\beta_j|0,\gamma_j)\varphi(\gamma_j)$ such that $p(\boldsymbol{\beta}|\mathbf{y},\hat{\boldsymbol{\gamma}})$ can be a "good" relaxation to $p(\boldsymbol{\beta}|\mathbf{y})$. Using the product rule for probabilities, we can write the full posterior as:

$$\begin{aligned} p(\boldsymbol{\beta},\boldsymbol{\gamma}|\mathbf{y}) &\propto p(\boldsymbol{\beta}|\mathbf{y},\boldsymbol{\gamma})p(\boldsymbol{\gamma}|\mathbf{y}) \\ &= \mathcal{N}(\mathbf{m}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}) \times \frac{p(\mathbf{y}|\boldsymbol{\gamma})p(\boldsymbol{\gamma})}{p(\mathbf{y})}. \end{aligned} \tag{6.66}$$

Since $p(\mathbf{y})$ is independent of $\boldsymbol{\gamma}$, the quantity $p(\mathbf{y}|\boldsymbol{\gamma})p(\boldsymbol{\gamma}) = \int p(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}|\boldsymbol{\gamma})p(\boldsymbol{\gamma})d\boldsymbol{\beta}$ is the prime target for variational methods [208]. This quantity is known as evidence or marginal likelihood. A good way of selecting $\hat{\boldsymbol{\gamma}}$ is to choose it as the minimiser of the sum of the misaligned probability mass, e.g.,

$$\begin{aligned} \hat{\boldsymbol{\gamma}} &= \underset{\boldsymbol{\gamma}\geq\mathbf{0}}{\mathrm{argmin}} \int p(\mathbf{y}|\boldsymbol{\beta})\,|p(\boldsymbol{\beta}) - p(\boldsymbol{\beta};\boldsymbol{\gamma})|\,d\boldsymbol{\beta} \\ &= \underset{\boldsymbol{\gamma}\geq\mathbf{0}}{\mathrm{argmax}} \int p(\mathbf{y}|\boldsymbol{\beta}) \prod_{j=1}^{n} \mathcal{N}(\beta_j|0,\gamma_j)\varphi(\gamma_j)d\boldsymbol{\beta}. \end{aligned} \tag{6.67}$$

The second equality is a consequence of $p(\boldsymbol{\beta};\boldsymbol{\gamma}) \leq p(\boldsymbol{\beta})$. The procedure in (6.67) is referred to as evidence maximisation or type-II maximum likelihood [191]. It means that the marginal likelihood can be maximised by selecting the most probable hyperparameters able to explain the observed data. Once $\hat{\boldsymbol{\gamma}}$ is computed, an estimate of the unknown weights can be obtained by setting $\hat{\boldsymbol{\beta}}$ to the posterior mean (6.64) as

$$\hat{\boldsymbol{\beta}} = \mathbb{E}(\boldsymbol{\beta}|\mathbf{y};\hat{\boldsymbol{\gamma}}) = \hat{\boldsymbol{\Gamma}}\mathbf{X}^\top(\sigma^2\mathbf{I} + \mathbf{X}\hat{\boldsymbol{\Gamma}}\mathbf{X}^\top)^{-1}\mathbf{y}, \tag{6.68}$$

with $\hat{\boldsymbol{\Gamma}} \triangleq \mathrm{diag}[\hat{\boldsymbol{\gamma}}]$. If an algorithm can be proposed to compute $\hat{\boldsymbol{\gamma}}$ in (6.67), we can obtain an estimation of the posterior mean $\hat{\boldsymbol{\beta}}$.

Remark 17 *By using a Laplace prior (see Remark 12) and the maximum a posterior (MAP) formulation, one can easily obtain the ℓ_1 minimiser, which is a penalised least square (PLS) estimate. Therefore, it might be tempting to assume that the Bayesian framework is simply a probabilistic re-interpretation of classical methods since we have just seen that the MAP and PLS estimates are equivalent. However, this is not the case. It is sometimes overlooked that the distinguishing element of Bayesian methods is really marginalisation, where instead of seeking to "estimate" all "nuisance" variables in our models, we attempt to integrate them out. In the Bayesian framework, marginal likelihoods have a natural built-in penalty for more complex models. At a certain point, the marginal likelihood will begin to decrease with increasing complexity, and hence, does not intrinsically suffer from the overfitting problems that occur when considering only likelihoods. An intuitive explanation about why the marginal likelihood will begin to decrease with increasing complexity is that, as the complexity of the model increases, the prior will be spread out more thinly across both the "good" models and the "bad" models. Because the marginal likelihood is the likelihood integrated with respect to the prior, spreading the prior across too many models will place too little prior mass on the "good" models, and as a result, cause the marginal likelihood to decrease.*

6.6.3 Centralised Optimisation Algorithm

Algorithm Derivation I: Duality Perspective

Proposition 6 *The optimal hyperparameters $\hat{\boldsymbol{\gamma}}$ in (6.67) can be obtained by minimising the following objective function*

$$\mathcal{L}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}) = \log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \mathbf{y}^\top(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{y} + \sum_{j=1}^{N} p(\gamma_j), \tag{6.69}$$

where $p(\gamma_j) = -2\log\varphi(\gamma_j)$. The posterior mean is then given by

$$\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\Gamma}}\mathbf{X}^\top(\sigma^2\mathbf{I} + \mathbf{X}\hat{\boldsymbol{\Gamma}}\mathbf{X}^\top)^{-1}\mathbf{y},$$

where $\hat{\boldsymbol{\Gamma}} = \mathrm{diag}[\hat{\boldsymbol{\gamma}}]$.

Proof *We first re-express $\mathbf{m}_\beta$ and $\boldsymbol{\Sigma}_\beta$ in (6.64) and (6.65) using the Woodbury inversion identity:*

$$\mathbf{m}_\beta = \boldsymbol{\Gamma}\mathbf{X}^\top(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{y} = \sigma^{-2}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top\mathbf{y}, \tag{6.70}$$

$$\boldsymbol{\Sigma}_\beta = \boldsymbol{\Gamma} - \boldsymbol{\Gamma}\mathbf{X}^\top(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{X}\boldsymbol{\Gamma} = (\boldsymbol{\Gamma}^{-1} + \sigma^{-2}\mathbf{X}^\top\mathbf{X})^{-1}. \tag{6.71}$$

Since the data likelihood $p(\mathbf{y}|\boldsymbol{\beta})$ is Gaussian, we can write the integral for the marginal likelihood in (6.67), as

$$\begin{aligned}&\int \mathcal{N}(\mathbf{y}|\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}) \prod_{j=1}^{N} \mathcal{N}(\beta_j|0, \gamma_j)\varphi(\gamma_j)d\boldsymbol{\beta} \\ &= \left(\frac{1}{2\pi\sigma^2}\right)^{M/2} \left(\frac{1}{2\pi}\right)^{N/2} \int \exp\left(-E(\boldsymbol{\beta})\right) d\boldsymbol{\beta} \prod_{j=1}^{N} \frac{\varphi(\gamma_j)}{\sqrt{\gamma_j}},\end{aligned} \tag{6.72}$$

where

$$\begin{aligned}E(\boldsymbol{\beta}) &= \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \frac{1}{2}\boldsymbol{\beta}^\top\boldsymbol{\Gamma}^{-1}\boldsymbol{\beta}, \\ \boldsymbol{\Gamma} &= \mathrm{diag}(\boldsymbol{\gamma}).\end{aligned}$$

Equivalently, we get

$$E(\boldsymbol{\beta}) = \frac{1}{2}(\boldsymbol{\beta} - \mathbf{m}_\beta)^\top\boldsymbol{\Sigma}_\beta^{-1}(\boldsymbol{\beta} - \mathbf{m}_\beta) + E(\mathbf{y}), \tag{6.73}$$

where $\mathbf{m}_\beta$ and $\boldsymbol{\Sigma}_\beta$ are given by (6.70) and (6.71). Using the Woodbury inversion identity, we obtain:

$$\begin{aligned}E(\mathbf{y}) &= \frac{1}{2}\left(\sigma^{-2}\mathbf{y}^\top\mathbf{y} - \sigma^{-2}\mathbf{y}^\top\mathbf{X}\boldsymbol{\Sigma}_\beta\boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top\mathbf{y}\sigma^{-2}\right) \\ &= \frac{1}{2}\mathbf{y}^\top(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{y}.\end{aligned} \tag{6.74}$$

Using (6.73), we can evaluate the integral in (6.72) to obtain

$$\int \exp\left(-E(\boldsymbol{\beta})\right) d\boldsymbol{\beta} = \exp\left(-E(\mathbf{y})\right) (2\pi)^{N/2}|\boldsymbol{\Sigma}_\beta|^{1/2}.$$

Exploiting the determinant identity, we have

$$|\boldsymbol{\Gamma}^{-1}||\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| = |\sigma^2\mathbf{I}||\boldsymbol{\Gamma}^{-1} + \sigma^{-2}\mathbf{X}^\top\mathbf{X}|,$$

from which we can compute the first term in (6.69) as

$$\log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| = -\log|\boldsymbol{\Sigma}_\beta| + M\log\sigma^2 + \log|\boldsymbol{\Gamma}|.$$

Then applying a $-2\log(\cdot)$ *transformation to (6.72), we have*

$$\begin{aligned}
&-2\log\int p(\mathbf{y}|\boldsymbol{\beta})\prod_{j=1}^{N}\mathcal{N}(\beta_j|0,\gamma_j)\varphi(\gamma_j)d\boldsymbol{\beta}\\
=&-2\log\left[\left(\frac{1}{2\pi\sigma^2}\right)^{M/2}\left(\frac{1}{2\pi}\right)^{N/2}\exp\left(-E(\mathbf{y})\right)(2\pi)^{N/2}|\boldsymbol{\Sigma}_\beta|^{1/2}d\boldsymbol{\beta}\prod_{j=1}^{N}\frac{\varphi(\gamma_j)}{\sqrt{\gamma_j}}\right]\\
=&M\log 2\pi\sigma^2-\log|\boldsymbol{\Sigma}_\beta|+2E(\mathbf{y})+\log|\boldsymbol{\Gamma}|+\textstyle\sum_{j=1}^{N}\log\varphi(\gamma_j)\\
=&-\log|\boldsymbol{\Sigma}_\beta|+M\log 2\pi\sigma^2+\log|\boldsymbol{\Gamma}|+\mathbf{y}^\top(\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{y}+\textstyle\sum_{j=1}^{N}p(\gamma_j)\\
=&\log|\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top|+M\log 2\pi+\mathbf{y}^\top(\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{y}+\textstyle\sum_{j=1}^{N}p(\gamma_j).
\end{aligned}$$

From (6.67), we then obtain

$$\hat{\boldsymbol{\gamma}}=\underset{\boldsymbol{\gamma}\geq\mathbf{0}}{\operatorname{argmin}}\log|\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top|+\mathbf{y}^\top(\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top)^{-1}\mathbf{y}+\textstyle\sum_{j=1}^{N}p(\gamma_j).$$

We compute the posterior mean to get an estimate of $\boldsymbol{\beta}$*:*

$$\hat{\boldsymbol{\beta}}=\mathbb{E}(\boldsymbol{\beta}|\mathbf{y};\hat{\boldsymbol{\gamma}})=\hat{\boldsymbol{\Gamma}}\mathbf{X}^\top(\sigma^2\mathbf{I}+\mathbf{X}\hat{\boldsymbol{\Gamma}}\mathbf{X}^\top)^{-1}\mathbf{y},$$

where $\hat{\boldsymbol{\Gamma}}=\operatorname{diag}[\hat{\boldsymbol{\gamma}}]$.

Lemma 1 *The cost function* $\mathcal{L}_\gamma(\boldsymbol{\gamma})$ *in* (6.69) *is a nonconvex function with respect to* $\boldsymbol{\gamma}$.

Proof *We first show that the data-dependent term in (6.69) is convex in* $\boldsymbol{\beta}$ *and* $\boldsymbol{\gamma}$*. From (6.70), (6.71) and (6.74), the data-dependent term can be re-expressed as*

$$\begin{aligned}
&\mathbf{y}^\top\left(\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top\right)^{-1}\mathbf{y}\\
=&\sigma^{-2}\mathbf{y}^\top\mathbf{y}-\sigma^{-2}\mathbf{y}^\top\mathbf{X}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top\sigma^{-2}\mathbf{y}\\
=&\sigma^{-2}\|\mathbf{y}-\mathbf{X}\mathbf{m}_\beta\|_2^2+\mathbf{m}_\beta^\top\boldsymbol{\Gamma}^{-1}\mathbf{m}_\beta\\
=&\min_{\boldsymbol{\beta}}\{\sigma^{-2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\boldsymbol{\beta}^\top\boldsymbol{\Gamma}^{-1}\boldsymbol{\beta}\},
\end{aligned}\tag{6.75}$$

where $\mathbf{m}_\beta$ *is the posterior mean defined in (6.64). It can easily be shown that the minimisation problem is convex in* $\boldsymbol{\beta}$ *and* $\boldsymbol{\gamma}$*, where* $\boldsymbol{\Gamma}\triangleq\operatorname{diag}[\boldsymbol{\gamma}]$.

Next we define

$$h(\boldsymbol{\gamma})\triangleq\log|\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top|+\textstyle\sum_{j=1}^{N}p(\gamma_j),\tag{6.76}$$

and show $h(\boldsymbol{\gamma})$ *is a concave function with respect to* $\boldsymbol{\gamma}$*.* $\log|\cdot|$ *is concave in the space of positive semi-definite matrices. Moreover,* $\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top$ *is an affine function of* $\boldsymbol{\gamma}$ *and*

is positive semidefinite for any $\boldsymbol{\gamma} \geq 0$*. This implies that* $\log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top|$ *is a concave, nondecreasing function of* $\boldsymbol{\gamma}$*. Since we adopt a super-Gaussian prior with potential function* $\varphi(\gamma_j), \forall j$*, as described in Proposition 3, a direct consequence is that* $p(\gamma_j) = -\log\varphi(\gamma_j)$ *is concave.*

Before presenting the main results of this Section, we introduce an important duality lemma (see Sec. 4.2 in [97]) which is deeply rooted in convex analysis [162]. This duality lemma will be useful for the development of the convex optimisation algorithm in this and the next Sections.

Lemma 2 *It is a general fact of convex analysis that a concave function* $f(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$ *can be represented via a* conjugate *or* dual *function as follows* $f(\mathbf{x}) = \min_{\mathbf{x}^*}\left[\langle\mathbf{x}^*, \mathbf{x}\rangle - f^*(\mathbf{x}^*)\right]$*, where the conjugate function* f^* *can be obtained from the following dual expression:* $f^*(\mathbf{x}^*) = \min_{\mathbf{x}}\left[\langle\mathbf{x}^*, \mathbf{x}\rangle - f(\mathbf{x})\right]$.

We can express an nonconvex function $h(\boldsymbol{\gamma})$ as $h(\boldsymbol{\gamma}) = \min_{\boldsymbol{\gamma}^*\geq 0}\langle\boldsymbol{\gamma}^*, \boldsymbol{\gamma}\rangle - h^*(\boldsymbol{\gamma}^*)$, where $h^*(\boldsymbol{\gamma}^*)$ is defined as the concave conjugate of $h(\boldsymbol{\gamma})$ and is given by $h^*(\boldsymbol{\gamma}^*) = \min_{\boldsymbol{\gamma}\geq 0}\langle\boldsymbol{\gamma}^*, \boldsymbol{\gamma}\rangle - h(\boldsymbol{\gamma})$.

Let $h(\boldsymbol{\gamma}) = \log\left|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top\right| + \sum_{j=1}^N p(\gamma_j)$, and assume that $p(\gamma_j)$ is concave with respect to ${\gamma_j}^2$. Using Lemma 2, we can create a strict upper bounding auxiliary function $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta})$ of $\mathcal{L}(\boldsymbol{\gamma})$ in (6.69),

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta}) \\
&\triangleq \langle\boldsymbol{\gamma}^*, \boldsymbol{\gamma}\rangle - h^*(\boldsymbol{\gamma}^*) + \mathbf{y}^\top\left(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top\right)^{-1}\mathbf{y} \\
&= \frac{1}{\sigma^2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum_{j=1}^N\left(\frac{{\beta_j}^2}{\gamma_j} + \gamma_j^*\gamma_j\right) - h^*(\boldsymbol{\gamma}^*).
\end{aligned} \tag{6.77}$$

For a fixed $\boldsymbol{\gamma}^*$, we notice that $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta})$ is jointly convex in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ and can be globally minimised by solving over $\boldsymbol{\gamma}$ and then $\boldsymbol{\beta}$. Since ${\beta_j}^2/\gamma_j + \gamma_j^*\gamma_j \geq 2\beta_j\sqrt{\gamma_j^*}$, for any $\boldsymbol{\beta}$, $\gamma_j = |w_j|/\sqrt{\gamma_j^*}$ minimises $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta})$.

The next step is to find a $\hat{\boldsymbol{\beta}}$ that minimises $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta})$. When $\gamma_j = |w_j|/\sqrt{\gamma_j^*}$ is substituted into $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta})$, $\hat{\boldsymbol{\beta}}$ can be obtained by solving the following weighted convex ℓ_1-minimisation problem

$$\hat{\mathbf{w}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + 2\sigma^2\sum_{j=1}^N\sqrt{\boldsymbol{\gamma}_j^*}|\beta_j|\}, \tag{6.78}$$

[2]This is not a strong assumption since all distributions in Remark 7 satisfy it.

where $\sqrt{\gamma_j^*}$ are the weights.

We can then set

$$\gamma_j = \frac{|\hat{w}_j|}{\sqrt{\gamma_j^*}}, \ \forall j, \tag{6.79}$$

and, as a consequence, $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta})$ will be minimised for any fixed $\boldsymbol{\gamma}^*$.

Now, consider again $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\gamma}^*, \boldsymbol{\beta})$ in (6.77). For any fixed $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, the tightest bound can be obtained by minimising over $\boldsymbol{\gamma}^*$. From the definition of $\boldsymbol{\gamma}^*$, the tightest value of $\boldsymbol{\gamma}^* = \hat{\boldsymbol{\gamma}^*}$ equals the slope at the current $\boldsymbol{\gamma}$ of the function $h(\boldsymbol{\gamma}) \triangleq \log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \sum_j p(\gamma_j)$. Using basic principles in convex analysis, we then obtain the following analytic form for the optimiser $\boldsymbol{\gamma}^*$:

$$\begin{aligned} \hat{\boldsymbol{\gamma}}^* &= \nabla_\gamma \left(\log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \sum_j p(\gamma_j) \right) \\ &= \operatorname{diag}\left[\mathbf{X}^\top \left(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top \right)^{-1} \mathbf{X} \right] + p'(\boldsymbol{\gamma}), \end{aligned} \tag{6.80}$$

where $p'(\boldsymbol{\gamma}) = [p'(\gamma_1), \ldots, p'(\gamma_N)]^{\mathrm{T}}$.

The algorithm is then based on successive iterations of (6.78), (6.99) and (6.101) until convergence to $\hat{\boldsymbol{\gamma}}$. We then compute the posterior mean and covariance for the faults as follows

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \mathbb{E}(\boldsymbol{\beta}|\mathbf{y}; \hat{\boldsymbol{\gamma}}) = \hat{\boldsymbol{\Gamma}}\mathbf{X}^{\mathrm{T}}(\lambda\mathbf{I} + \mathbf{X}\hat{\boldsymbol{\Gamma}}\mathbf{X}^\top)^{-1}\mathbf{y}, \\ \boldsymbol{\Sigma}_{\hat{\boldsymbol{\beta}}} &= \hat{\boldsymbol{\Gamma}} - \hat{\boldsymbol{\Gamma}}\mathbf{X}^{\mathrm{T}}(\lambda\mathbf{I} + \mathbf{X}\hat{\boldsymbol{\Gamma}}\mathbf{X}^\top)^{-1}\mathbf{X}, \end{aligned} \tag{6.81}$$

where $\hat{\boldsymbol{\Gamma}} = \operatorname{diag}[\hat{\boldsymbol{\gamma}}]$. The above described procedure is summarised in Algorithm 3.

Algorithm Derivation II: Convex Constraints Perpective

One problem in getting $\hat{\boldsymbol{\beta}}$ by computing $\hat{\boldsymbol{\gamma}}$ in (6.67) is that it is difficult to enforce additional constraints on the parameters $\boldsymbol{\beta}$ of the system, such as positivity. Another motivation for constrained optimisation comes from stability considerations. Typically, the underlying system is known *a priori* to be stable. A lot of stability conditions can be formulated as convex optimisation problems (see [26, 89], etc). Only few contributions are available in the literature that address the problem of how to take into account *a priori* information on system stability [34, 229]. Based on the terminology in [28], we consider the following assumption on $\boldsymbol{\beta}$.

Algorithm 3 Reweighted ℓ_1-minimisation on hyperparameter $\boldsymbol{\gamma}$

Data: Successive observations of $\mathbf{y}$ from t_0 to t_M.
Result: Posterior mean for $\boldsymbol{\beta}$.

Step 1 *Set iteration count k to zero and initialise each $w_j^1 = \sqrt{\gamma_j^*}$, with randomly chosen initial values for $\gamma_j^*,\ \forall j$, e.g. with $\gamma_j^* = 1,\ \forall j$;*

Step 2 *Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;*

Step 3 *At the k^{th} iteration, solve the reweighted ℓ_1-minimisation problem*

$$\boldsymbol{\beta}^k = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_j |w_j^k \beta_j|,\ \forall j;$$

Step 4 *Compute*

$$\gamma_j^k = \frac{|\beta_j^k|}{w_j^k}, \forall j;$$

Step 5 *Update $\hat{\boldsymbol{\gamma}^*}^{k+1}$ using* (6.99)

$$\hat{\boldsymbol{\gamma}^*}^{k+1} = \operatorname{diag}\left[\mathbf{X}^\top \left(\lambda \mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}^k \mathbf{X}^\top\right)^{-1} \mathbf{X}\right] + p'(\boldsymbol{\gamma}^k);$$

Step 6 *Update weights w_j^k for the ℓ_1-minimisation at the next iteration $w_j^{k+1} = \sqrt{\hat{\boldsymbol{\gamma}}_j^{*\,k+1}}$;*

Step 7 *$k \to k+1$ and iterate Steps 2 to 5 until convergence to some $\hat{\boldsymbol{\gamma}}$ and $\hat{\boldsymbol{\Gamma}} = \operatorname{diag}(\hat{\boldsymbol{\gamma}})$;*

Step 8 *Compute $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\Gamma}}\mathbf{X}^{\mathrm{T}}(\lambda \mathbf{I} + \mathbf{X}\hat{\boldsymbol{\Gamma}}\mathbf{X}^\top)^{-1}\mathbf{y}$.*

Assumption 9 *Constraints on the weights $\boldsymbol{\beta}$ can be described by a set of convex functions*

$$\begin{aligned} H_p^{[I]}(\boldsymbol{\beta}) &\leq 0,\ \ p = 1, \ldots, m_I, \\ H_q^{[E]}(\boldsymbol{\beta}) &= 0,\ \ q = 1, \ldots, m_E. \end{aligned} \tag{6.82}$$

The convex functions $H_p^{[I]} : \mathbb{R}^N \to \mathbb{R}$ represent inequality constraint functions. The convex functions $H_q^{[E]} : \mathbb{R}^N \to \mathbb{R}$ represent equality constraint functions and $H_q^{[E]}(\boldsymbol{\beta}) = a_q^\top \boldsymbol{\beta} - b_q$ are affine functions where $a_q, b_q \in \mathbb{R}^N$.

Based on the analysis in Section 6.6.3, we first derive a dual objective function in the $\boldsymbol{\beta}$-space with convex constraints by considering the equivalent objective function

of (6.69) in the $\boldsymbol{\gamma}$-space. We then show that this equivalent objective function is also nonconvex.

Proposition 7 *The estimate for $\boldsymbol{\beta}$ with constraints can be obtained by solving the optimisation problem*

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sigma^2 g_{sbl}(\boldsymbol{\beta}), \quad \text{subject to} \quad (6.82), \tag{6.83}$$

where

$$g_{sbl}(\boldsymbol{\beta}) = \min_{\boldsymbol{\gamma} \geq \mathbf{0}} \left(\boldsymbol{\beta}^\top \boldsymbol{\Gamma}^{-1} \boldsymbol{\beta} + \log |\sigma^2 \mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \sum_{j=1}^N p(\gamma_j) \right)$$

and the estimate of the stochastic variable $\boldsymbol{\beta}$ is given by the poseterior mean $\mathbf{m}_\beta$ defined in (6.64).

Proof *Using the data-dependent term in* (6.75)*, together with $\mathcal{L}_\gamma(\boldsymbol{\gamma})$ in (6.69), we can create a strict upper bounding auxiliary function on $\mathcal{L}_\gamma(\boldsymbol{\gamma})$ as*

$$\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \boldsymbol{\beta}^\top \boldsymbol{\Gamma}^{-1} \boldsymbol{\beta} + \log |\sigma^2 \mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \sum_{j=1}^N p(\gamma_j).$$

When we minimise over $\boldsymbol{\gamma}$ instead of $\boldsymbol{\beta}$, we obtain

$$\begin{aligned} \mathcal{L}_\beta(\boldsymbol{\beta}) &\triangleq \min_{\boldsymbol{\gamma} \geq \mathbf{0}} \mathcal{L}_{\gamma,\beta}(\boldsymbol{\gamma}, \boldsymbol{\beta}) \\ &= \frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \min_{\boldsymbol{\gamma} \geq \mathbf{0}} \left(\boldsymbol{\beta}^\top \boldsymbol{\Gamma}^{-1} \boldsymbol{\beta} + \log |\sigma^2 \mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \sum_{j=1}^N p(\gamma_j) \right). \end{aligned} \tag{6.84}$$

Then for $\boldsymbol{\beta}$ with convex constraints as described in Assumption 9, we can obtain the formulation in Proposition 7.

From the derivations in (6.75), we can clearly see that the estimate of the stochastic variable $\boldsymbol{\beta}$ is the poseterior mean $\mathbf{m}_\beta$ defined in (6.64).

Although all the constraint functions are convex in Proposition 7, we show in the following Lemma that the objective function in (6.83) is nonconvex since it is the sum of convex and concave functions.

Lemma 3 *The penalty function $g_{sbl}(\boldsymbol{\beta})$ in Theorem 7 is a non-decreasing, concave function of $|\boldsymbol{\beta}|$ which promotes sparsity on the weights $\boldsymbol{\beta}$.*

Proof *It is shown in Lemma 1 that $h(\boldsymbol{\gamma})$ is concave with respect to $\boldsymbol{\gamma} \geq 0$. According to the duality lemma 2, we can express the concave function $h(\boldsymbol{\gamma})$ as*

$$h(\boldsymbol{\gamma}) = \min_{\boldsymbol{\gamma}^* \geq 0} \langle \boldsymbol{\gamma}^*, \boldsymbol{\gamma} \rangle - h^*(\boldsymbol{\gamma}^*),$$

where $h^*(\boldsymbol{\gamma}^*)$ is defined as the concave conjugate of $h(\boldsymbol{\gamma})$ and is given by

$$h^*(\boldsymbol{\gamma}^*) = \min_{\gamma \geq 0} \langle \boldsymbol{\gamma}^*, \boldsymbol{\gamma} \rangle - h(\boldsymbol{\gamma}).$$

From the proof of Lemma 1, the data-dependent term $\mathbf{y}^\top \left(\sigma^2 \mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top\right)^{-1} \mathbf{y}$ can be re-expressed as

$$\min_{\boldsymbol{\beta}} \left(\frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \boldsymbol{\beta}^\top \boldsymbol{\Gamma}^{-1} \boldsymbol{\beta} \right).$$

Therefore we can create a strict upper bounding auxiliary function $\mathcal{L}_{\gamma,\beta}(\boldsymbol{\gamma}, \boldsymbol{\beta})$ on $\mathcal{L}_\gamma(\boldsymbol{\gamma})$ in (6.69) by considering the fact that, in the dual expression,

$$\begin{aligned} \mathcal{L}_{\gamma,\beta}(\boldsymbol{\gamma}, \boldsymbol{\beta}) &\triangleq \langle \boldsymbol{\gamma}^*, \boldsymbol{\gamma} \rangle - h^*(\boldsymbol{\gamma}^*) + \mathbf{y}^\top \left(\sigma^2 \mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top\right)^{-1} \mathbf{y} \\ &= \frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum\nolimits_j \left(\frac{{\beta_j}^2}{\gamma_j} + \gamma_j^* \gamma_j \right) - h^*(\boldsymbol{\gamma}^*). \end{aligned} \tag{6.85}$$

We can then re-express $g_{sbl}(\boldsymbol{\beta})$ as

$$g_{sbl}(\boldsymbol{\beta}) = \min_{\boldsymbol{\gamma}, \boldsymbol{\gamma}^* \geq \mathbf{0}} \left(\sum\nolimits_j \left(\beta_j^2 / \gamma_j + \gamma_j^* \gamma_j \right) - h^*(\boldsymbol{\gamma}^*) \right). \tag{6.86}$$

$g_{sbl}(\boldsymbol{\beta})$ is minimised over $\boldsymbol{\gamma}$ when $\gamma_j = |\beta_j| / \sqrt{\gamma_j^*}$, $\forall j$. Substituting this expression into $g_{sbl}(\boldsymbol{\beta})$, we get

$$g_{sbl}(\boldsymbol{\beta}) = \min_{\boldsymbol{\gamma}^* \geq \mathbf{0}} \left(\sum\nolimits_j 2 |\sqrt{\gamma_j^*} \beta_j| - h^*(\boldsymbol{\gamma}^*) \right). \tag{6.87}$$

This indicates that $g_{sbl}(\boldsymbol{\beta})$ can be represented as a minimum over upper-bounding hyperplanes in $\|\boldsymbol{\beta}\|_1$, and thus must be concave. $g_{sbl}(\boldsymbol{\beta})$ thus promotes sparsity. Moreover, $g_{sbl}(\boldsymbol{\beta})$ must be non-decreasing since $\boldsymbol{\gamma}^* \geq \mathbf{0}$.

We define the terms excluding $h^*(\boldsymbol{\gamma}^*)$ as

$$\mathcal{L}_{\gamma^*}(\boldsymbol{\gamma}, \boldsymbol{\beta}) \triangleq \frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum\nolimits_j \left({\beta_j}^2 / \gamma_j + \gamma_j^* \gamma_j \right). \tag{6.88}$$

For a fixed $\boldsymbol{\gamma}^*$, we notice that $\mathcal{L}_{\gamma^*}(\boldsymbol{\gamma}, \boldsymbol{\beta})$ is jointly convex in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ and can be globally minimised by solving over $\boldsymbol{\gamma}$ and then $\boldsymbol{\beta}$. Since ${\beta_j}^2 / \gamma_j + \gamma_j^* \gamma_j \geq 2\beta_j \sqrt{\gamma_j^*}$, for any $\boldsymbol{\beta}$, $\gamma_j = |w_j| / \sqrt{\gamma_j^*}$ minimises $\mathcal{L}_{\gamma^*}(\boldsymbol{\gamma}, \boldsymbol{\beta})$. When $\gamma_j = |w_j| / \sqrt{\gamma_j^*}$ is substituted into $\mathcal{L}_{\gamma^*}(\boldsymbol{\gamma}, \boldsymbol{\beta})$, $\hat{\boldsymbol{\beta}}$ can be obtained by solving the following weighted convex ℓ_1-

minimisation procedure

$$\hat{\mathbf{w}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + 2\sigma^2 \sum_{j=1}^{N} |\sqrt{\gamma_j^*}\beta_j| \right\}. \tag{6.89}$$

We can then set $\gamma_j = |\hat{w}_j|/\sqrt{\gamma_j^*}$, $\forall j$. As a consequence, $\mathcal{L}_{\gamma^*}(\boldsymbol{\gamma}, \boldsymbol{\beta})$ will be minimised for any fixed $\boldsymbol{\gamma}^*$. Due to the concavity of $g_{sbl}(\boldsymbol{\beta})$, the objective function in (6.83) can be optimised using a reweighted ℓ_1-minimisation in a similar way as was considered in (6.89). The updated weight at the k^{th} iteration is then given by

$$u_j^{(k)} \triangleq \left. \frac{\partial g_{sbl}(\boldsymbol{\beta})}{2\partial|\beta_j|} \right|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(k)}} = \sqrt{\gamma_j^*}. \tag{6.90}$$

We can now explain how the update of the parameters can be performed based on the above. We start by setting the iteration count k to zero and $u_j^{(0)} = 1$, $\forall j$. At this stage, the solution is a typical ℓ_1-minimisation solution.Then at the k^{th} iteration, we initialise $u_j^{(k)} = \sqrt{\boldsymbol{\gamma}_j^{*(k)}}$, $\forall j$ and then minimise over $\boldsymbol{\gamma}$ using $\gamma_j = |\beta_j|/\sqrt{\gamma_j^*}$, $\forall j$. Consider again $\mathcal{L}_{\gamma,\beta}(\boldsymbol{\gamma}, \boldsymbol{\beta})$. For any fixed $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, the tightest bound can be obtained by minimising over $\boldsymbol{\gamma}^*$. The tightest value of $\boldsymbol{\gamma}^* = \hat{\boldsymbol{\gamma}^*}$ equals the gradient of the function

$$h(\boldsymbol{\gamma}) \triangleq \log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \sum_{j=1}^{N} p(\gamma_j)$$

defined in Lemma 1 at the current $\boldsymbol{\gamma}$. $\boldsymbol{\gamma}^*$ has the following analytical expression:

$$\begin{aligned} \hat{\boldsymbol{\gamma}}^* &= \nabla_{\boldsymbol{\gamma}} \left(\log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top| + \sum_{j=1}^{N} p(\gamma_j) \right) \\ &= \operatorname{diag}\left[\mathbf{X}^\top \left(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top \right)^{-1} \mathbf{X} \right] + p'(\boldsymbol{\gamma}), \end{aligned} \tag{6.91}$$

where $p'(\boldsymbol{\gamma}) = [p'(\gamma_1), \ldots, p'(\gamma_N)]^\top$. The optimal $\boldsymbol{\gamma}^{*(k+1)}$ can then be obtained as

$$\boldsymbol{\gamma}^{*(k+1)} = \operatorname{diag}\left[\mathbf{X}^\top \left(\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}^{(k)}\mathbf{X}^\top \right)^{-1} \mathbf{X} \right] + p'(\boldsymbol{\gamma}^{(k)}). \tag{6.92}$$

After computing the estimation of ${\gamma_j}^{(k)} = \frac{|\beta_j^{(k)}|}{\sqrt{\gamma_j^{*(k)}}}$, we can compute $\boldsymbol{\gamma}^{*(k+1)}$, which gives

$$\gamma_j^{*(k+1)} = \mathbf{X}_j^\top \left(\sigma^2\mathbf{I} + \mathbf{X}\mathbf{U}^{(k)}\mathbf{W}^{(k)}\mathbf{X}^\top \right)^{-1} \mathbf{X}_j + p'(\gamma_j^{(k)}),$$

where

$$\begin{aligned}\boldsymbol{\Gamma}^{(k)} &\triangleq \operatorname{diag}\left[\boldsymbol{\gamma}^{(k)}\right],\\ \mathbf{U}^{(k)} &\triangleq \operatorname{diag}\left[\mathbf{u}^{(k)}\right]^{-1} = \operatorname{diag}\left[\sqrt{\boldsymbol{\gamma}^{*(k)}}\right]^{-1},\\ \mathbf{W}^{(k)} &\triangleq \operatorname{diag}\left[|\boldsymbol{\beta}^{(k)}|\right].\end{aligned}$$

We can then define

$$u_j^{(k+1)} \triangleq \sqrt{\gamma_j^{*(k+1)}}$$

for the next iteration of the weighted ℓ_1-minimisation.

Algorithm Derivation III: Convex Concave Procedure Perspective

In this Section, we derive the algorithm form the perspective of convex concave procedure. According to Proposition 7, the optimisation problem can be cast as the following nonconvex optimisation problem

$$\min_{\boldsymbol{\gamma}>\mathbf{0},\boldsymbol{\beta}} \frac{1}{\sigma^2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \boldsymbol{\beta}^\top\boldsymbol{\Gamma}^{-1}\boldsymbol{\beta} + \log|\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top|, \tag{6.93}$$

where $\boldsymbol{\Gamma}$ is a diagonal matrix with diagonal entries γ_j. Note that the structure of this optimisation programme is similar to the one set up in [32]. However, in our case the logarithmic penalty is on a full matrix, while in [32] the logarithmic penalty is on a diagonal matrix. This results in a better performance of (6.93) in terms of sparsity of solutions [212]. However, finding a solution to the optimisation problem (6.93) is more computationally expensive.

We first show that the stated program can be formulated as a convex-concave procedure (CCCP). Again, to be more general, $\boldsymbol{\beta}$ is substituted with $\mathbf{B}\boldsymbol{\beta}$. We have the following Proposition.

Proposition 8 *The following programme*

$$\min_{\boldsymbol{\beta},\boldsymbol{\Gamma}} \frac{1}{\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^\top\left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right) + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} + \log|\boldsymbol{\Gamma}| + \log\left|\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B} + \frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{X}\right|.$$

can be formulated as a convex-concave procedure (CCCP).

Proof *In the first half part of the proof, we show that the stated program can be formulated as a convex-concave procedure (CCCP).*

Fact on convexity: *the function*

$$\begin{aligned} u(\boldsymbol{\beta}, \boldsymbol{\Gamma}) &= \frac{1}{\sigma^2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta} \\ &\propto \frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{\sigma^2}{2} \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta} \end{aligned} \tag{6.94}$$

is convex jointly in $\boldsymbol{\beta}, \boldsymbol{\Gamma}$. The function $f(\boldsymbol{\beta}, Y) = \boldsymbol{\beta}^\top Y^{-1} \boldsymbol{\beta}$ is jointly convex in $\boldsymbol{\beta}, Y$ (see, [28, p.76]), hence u as a sum of convex functions is convex.

Fact on concavity: *the function*

$$v(\boldsymbol{\Gamma}) = \log |\boldsymbol{\Gamma}| + \log |\mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} + \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X}|$$

is concave in $\boldsymbol{\Gamma}$, if $\mathbf{X}^\top \mathbf{X}$ is invertible. We exploit the properties of the determinant of a matrix

$$|A_{22}||A_{11} - A_{12} A_{22}^{-1} A_{21}| = \left| \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \right| = |A_{11}||A_{22} - A_{21} A_{11}^{-1} A_{12}|.$$

We have

$$\begin{aligned} v(\boldsymbol{\Gamma}) &= \log |\boldsymbol{\Gamma}| + \log |\mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} + \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X}| \\ &= \log \left(|-\boldsymbol{\Gamma}||\mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} + \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X}| \right) \\ &= \log \left| \begin{pmatrix} \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X} & \mathbf{B}^\top \\ \mathbf{B} & -\boldsymbol{\Gamma} \end{pmatrix} \right| \\ &= \log \left| \mathbf{X}^\top \mathbf{X} \right| + \log \left| \boldsymbol{\Gamma} + \sigma^2 \mathbf{B} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{B}^\top \right|, \end{aligned} \tag{6.95}$$

which is a log*-determinant of an affine function of a semidefinite matrix $\boldsymbol{\Gamma}$ and hence concave. If $\mathbf{X}^\top \mathbf{X}$ is not invertible, then the proof is a bit longer, but it is also possible to show that $v(\boldsymbol{\Gamma})$ is concave.* ■

The cost function in (6.93) is convex in $\boldsymbol{\beta}$ but nonconvex in $\boldsymbol{\Gamma}$. This nonconvex optimisation problem can be formulated as a convex convave procedure (CCCP). It can be shown that solving this CCCP is equivalent to solving an iterative convex optimisation programme, which converges to a stationary point [179]. Hereafter, we provide another derivation of the algorithm presented in [212]. From our point of view this derivation is much simpler and hence we present it here. In Section 6.6.4, we will derive a decentralised version of this iterative algorithm using ADMM.

Defining the following two expressions

$$\begin{aligned} u(\boldsymbol{\beta}, \boldsymbol{\gamma}) &\triangleq \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_j \frac{\beta_j^2}{\gamma_j}, \\ v(\boldsymbol{\gamma}) &\triangleq -\log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top|. \end{aligned} \tag{6.96}$$

Note that $u(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is jointly convex in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, and $v(\boldsymbol{\gamma})$ is convex in $\boldsymbol{\gamma}$. As a consequence the minimisation of the cost function (6.93) can be formulated as a concave-convex procedure:

$$\min_{\boldsymbol{\gamma} \geq \mathbf{0}, \boldsymbol{\beta}} u(\boldsymbol{\beta}, \boldsymbol{\gamma}) - v(\boldsymbol{\gamma}). \tag{6.97}$$

Since $v(\boldsymbol{\gamma})$ is differentiable over $\boldsymbol{\gamma}$, the problem in (6.97) can be transformed into the following iterative convex optimisation problem

$$\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] = \operatorname*{argmin}_{\boldsymbol{\gamma} \geq \mathbf{0}, \boldsymbol{\beta}} u(\boldsymbol{\beta}, \boldsymbol{\gamma}) - \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k)^\top \boldsymbol{\gamma}. \tag{6.98}$$

Using basic principles in convex analysis, we then obtain the following analytic form for the negative gradient of $v(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}$:

$$\begin{aligned} \boldsymbol{\alpha}^k &= -\nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k)^\top \\ &= -\nabla_{\boldsymbol{\gamma}} \left(-\log|\sigma^2\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top|\right)|_{\gamma=\boldsymbol{\gamma}^k} \\ &= \operatorname{diag}\left[\mathbf{X}^{\mathrm{T}}\left(\lambda\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}^k\mathbf{X}^\top\right)^{-1}\mathbf{X}\right]. \end{aligned}$$

The iterative procedure (6.98) can then be formulated as

$$\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] = \operatorname*{argmin}_{\boldsymbol{\gamma} \geq \mathbf{0}, \boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_j \left(\frac{\beta_j^2}{\gamma_j} + \alpha_j^k \gamma_j\right). \tag{6.99}$$

The objective function in (6.99) is jointly convex in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ and can be globally minimised by solving over $\boldsymbol{\gamma}$ and then $\boldsymbol{\beta}$. If $\boldsymbol{\beta}$ is fixed, this gives

$$\boldsymbol{\gamma}^{k+1} = \operatorname*{argmin}_{\boldsymbol{\gamma} \geq \mathbf{0}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_j \left(\frac{\beta_j^2}{\gamma_j} + \alpha_j^k \gamma_j\right). \tag{6.100}$$

We notice that in (6.100), $\boldsymbol{\gamma}^{k+1}$ has a closed form solution $\gamma_j^{k+1} = |\beta_j|/\sqrt{\alpha_j^k}$. If $\gamma_j^{k+1} = |w_j|/\sqrt{\alpha_j^k}$ is substituted into (6.100), we obtain

$$\begin{aligned}\boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_j \left(\frac{\beta_j^2}{\gamma_j^{k+1}} + \alpha_j^k \gamma_j^{k+1} \right) \\ &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{N} |\sqrt{\alpha_j^k}\beta_j|.\end{aligned}$$

After computing $\boldsymbol{\beta}^{k+1}$, we can set

$$\gamma_j^{k+1} = \frac{|\beta_j^{k+1}|}{\sqrt{\alpha_j^k}}, \quad \forall j, \tag{6.101}$$

and then update $\boldsymbol{\alpha}^{k+1}$ by (6.99).

The iterative procedure can be initialised by setting γ_j^1 equal to any positive real scalar. However, some additional insight can be obtained by initialising $\alpha_j^1 = 1, \forall j$ instead. In that case, the first iteration becomes a linear regression problem with ℓ_1 penalty on the parameters $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}^1 = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_1}.$$

We can then update γ_j^1 using $\gamma_j^1 = \boldsymbol{\beta}^1/\sqrt{\alpha_j^1}$. Using this initialisation, we provably get results at least not worse than the generalised Lasso algorithm. The approach is summarised as Algorithm 4.

Implementation, Computational Complexity and Convergence

There is a heuristic option which can be used to speed up the algorithm. At every iteration of the algorithm, we update the parameters $\boldsymbol{\gamma}$ by means of equations (6.103) and (6.104), which involve the inversion of a matrix of large dimensions. One can speed up this part of the algorithm by pruning out the hyperparameter space of parameters $\boldsymbol{\gamma}_j$ (and respectively β_j) that are close to zero within some small threshold. According to the initialisation above, the first iteration of the algorithm (6.102) is actually a Lasso problem, which tends to yield a sparse solution, that is a solution with many $\boldsymbol{\gamma}_j$ close to zero within some small threshold. Consequently, many $\boldsymbol{\gamma}_j$

Algorithm 4 Reweighted ℓ_1-minimisation on parameter $\boldsymbol{\beta}$

1: Set $\boldsymbol{\Theta}^1$ be equal to the identity matrix;
2: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
3: **for** $k = 1, \ldots, k_{\max}$ **do**
4: Update the parameters as follows

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\Theta}^k\boldsymbol{\beta}\|_{\ell_1} \quad \text{subject to (6.82)}; \tag{6.102}$$

$$\boldsymbol{\gamma}_j^{k+1} = \frac{\boldsymbol{\beta}^{k+1}}{\boldsymbol{\Theta}_{jj}^k}; \tag{6.103}$$

$$\boldsymbol{\Theta}_{jj}^{k+1} = \left(\mathbf{X}_{j,:}^\top \left(\lambda\mathbf{I} + \mathbf{X}\boldsymbol{\Gamma}^{k+1}\mathbf{X}^\top\right)^{-1}\mathbf{X}_{j,:}\right)^{1/2}; \tag{6.104}$$

5: **if** A stopping criterion is satisfied **then**
6: Break;
7: **end if**
8: **end for**

(therefore β_j) can be pruned out already after the first iteration. Pruning at every iteration is a heuristic, which can be used to speed up the algorithm.

There are two important aspects of the reweighted ℓ_1-minimisation algorithm. First, for convex optimisation, there will be no exact zeros during the iterations and strictly speaking, we will always get a solution without any zero entry even when the RIP condition holds. However, some of the estimated weights will have very small magnitudes compared to those of other weights, e.g., $\pm 10^{-5}$ compared to 1, or the "energy" some of the estimated weights will be several orders of magnitude lower than the average "energy", e.g., $\|\beta_j\|_2^2 \ll \|\boldsymbol{\beta}\|_2^2$. Thus a threshold needs to be defined *a priori* to prune "small" weights at each iteration. The second aspect concerns the computational complexity of this approach. The repeated execution of Algorithm 4 is very cheap computationally since it scales as $\mathcal{O}(MN\|\boldsymbol{\beta}^{(k)}\|_0)$ (see [32, 212]). Since at each iteration certain weights are estimated to be zero, certain dictionary functions spanning the corresponding columns of $\mathbf{X}$ can be pruned out for the next iteration.

It is natural to investigate the convergence properties of this iterative reweighted ℓ_1-minimisation procedure. Let $\mathcal{A}(\cdot)$ denote a mapping that assigns to every point in $\mathbb{R}_+^N$ the subset of $\mathbb{R}_+^N$ which satisfies Steps 3 and 4 in Algorithm 3. Then the convergence property can be established as follows:

Proposition 9 *Given the initial point $\boldsymbol{\gamma}^{(0)} \in \mathbb{R}_+^n$ the sequence $\{\boldsymbol{\gamma}^{(k)}\}_{k=0}^{\infty}$ is generated satisfying $\boldsymbol{\gamma}^{(k+1)} \in \mathcal{A}(\boldsymbol{\gamma}^{(k)})$. This sequence is guaranteed to converge to a local minimum (or saddle point) of $\mathcal{L}_\gamma$ in (6.69).*

Proof *The proof is in one-to-one correspondence with that of the Global Convergence Theorem [228].*

1. *The mapping $\mathcal{A}(\cdot)$ is compact. Since any element of $\boldsymbol{\gamma}$ is bounded, $\mathcal{L}(\boldsymbol{\gamma})$ will not diverge to infinity. In fact, for any fixed $\mathbf{y}$, $\mathbf{X}$ and $\boldsymbol{\gamma}$, there will always exist a radius r such that for any $\|\boldsymbol{\gamma}^{(0)}\| \leq 0$, $\|\boldsymbol{\gamma}^{(k)}\| \leq 0$.*

2. *We denote $\boldsymbol{\gamma}'$ as the non-minimising point of $\mathcal{L}(\boldsymbol{\gamma}'') < \mathcal{L}(\boldsymbol{\gamma}')$, $\forall\, \boldsymbol{\gamma}'' \in \mathcal{A}(\boldsymbol{\gamma}')$. At any non-minimising $\boldsymbol{\gamma}'$ the auxiliary objective function $\mathcal{L}_{(\gamma^*)'}$ obtained from $\boldsymbol{\gamma}^*_{\sigma^2\mathbf{I}+\mathbf{X}\boldsymbol{\Gamma}\mathbf{X}^\top}$ will be strictly tangent to $\mathcal{L}(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}'$. It will therefore necessarily have a minimum elsewhere since the slope at $\boldsymbol{\gamma}'$ is nonzero by definition. Moreover, because the $\log|\cdot|$ function is strictly concave, at this minimum the actual const function will be reduced still further. Consequently, the proposed updates represent a valid descent function [228].*

3. *$\mathcal{A}(\cdot)$ is closed at all non-stationary points.*

6.6.4 Distributed Optimisation Algorithm

Motivation

We are living in the era of "big data". The answer to the question "What is big data?" provided by the research and advisory company Gartner is now widely accepted. According to Gartner, the Five 'V's of big data are: volume, velocity, value, veracity, and variety. One important category of data is time series data. Time series data satisfy Gartner's Five 'V's definition in many contexts, from complex physics simulations to biological and environmental research, to finance, business, healthcare, meteorology, and genomics. Fuelled by recently emerging areas such as Cloud Computing, Internet of Things, and Cyber Physical Systems, big data is increasingly playing an important role in modern society [222]. In these areas, massive time series data are generated. Based on these time series data a model of the dynamical system generating these data is sought after for analysis and/or control purposes. For large-scale industrial processes, the corresponding models are inevitably complex with various nonlinearities and noise terms. Modelling such nonlinear systems directly from time series data is known as nonlinear system

identification [117, 21]. Therefore, either the number of basis function is very large or the number of observations is very large

We will extend the centralised framework into a distributed nonlinear identification framework, thereby offering a means to improve on the computational performance of the algorithm. For this, the centralised identification problem is split into several sub-problems solved independently and in parallel. Parallelisation is performed by means of the Alternating Direction Method of Multipliers or ADMM (cf. [27]), which is a powerful algorithm for solving structured convex optimisation problems. ADMM was introduced in optimisation in the 1970's and is closely related to many other optimisation algorithms including Bregman iterative algorithms, Douglas-Rachford splitting, and proximal point methods (cf. [27] and references therein). ADMM has been shown to have strong convergence properties and to be useful for solving, by decomposition, large optimisation problems, which cannot be handled by generic optimisation solvers. ADMM has been applied in many areas, such as filtering [207], image processing [61] as well as large-scale problems in statistics and machine learning [27]. This approach has the advantage that memory and computational requirements can be both reduced in comparison to generic centralised solvers.

Alternating Direction Method of Multipliers

Recall that, in fact, we have $n_{\mathbf{x}}$ independent regression problems (see (6.102)) Moreover, the number of dictionary functions can be very large and at each iteration a nonsmooth ℓ_1 optimisation problem is solved, which is computationally expensive. Hereafter, we show how an approach based on the so-called ADMM can be used to significantly speed up the computation.

ADMM can be used to obtain solutions to problems of the following form:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & f(\boldsymbol{\beta}) + g(\mathbf{z}), \\ \text{subject to} \quad & P\boldsymbol{\beta} + Q\mathbf{z} = \mathbf{c}, \end{aligned} \tag{6.105}$$

where $\boldsymbol{\beta} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, $P \in \mathbb{R}^{p\times n}$, $Q \in \mathbb{R}^{p\times m}$, and $\mathbf{c} \in \mathbb{R}^p$. The functions $f(\cdot)$ and $g(\cdot)$ are convex, but can be nonsmooth, e.g. weighted ℓ_1 norm. The first step of the method consists in forming the augmented Lagrangian

$$\begin{aligned} L_\rho = & f(\boldsymbol{\beta}) + g(\mathbf{z}) + \mathbf{u}^\top (P\boldsymbol{\beta} + Q\mathbf{z} - c) + \\ & \rho/2 \|P\boldsymbol{\beta} + Q\mathbf{z} - c\|_2^2. \end{aligned} \tag{6.106}$$

After that optimisation programmes with respect to different variables can be solved separately as follows:

$$\begin{aligned}
\boldsymbol{\beta}^{\tau+1} &:= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(f(\boldsymbol{\beta}) + \frac{\rho}{2} \|P\boldsymbol{\beta} + Q\mathbf{z}^{\tau} - \mathbf{c} + \mathbf{u}^{\tau}\|_2^2 \right) \\
\mathbf{z}^{\tau+1} &:= \underset{\mathbf{z}}{\operatorname{argmin}} \left(g(\mathbf{z}) + \frac{\rho}{2} \|P\boldsymbol{\beta}^{\tau+1} + Q\mathbf{z} - \mathbf{c} + \mathbf{u}^{\tau}\|_2^2 \right) \\
\mathbf{u}^{\tau+1} &:= \mathbf{u}^{\tau} + P\boldsymbol{\beta}^{\tau+1} + Q\mathbf{z}^{\tau+1} - \mathbf{c}.
\end{aligned}$$

If $g(z)$ is equal to $\lambda\|z\|_1$, then the update on z is simply

$$\mathbf{z}^{\tau+1} = S_{\lambda/\rho}(P\boldsymbol{\beta}^{\tau+1} + \mathbf{u}^{\tau}),$$

where $S_{\lambda/\rho}$ is the soft thresholding operator defined as

$$S_{\lambda/\rho}(x) = \max(0, x - \lambda/\rho) - \max(0, -x - \lambda/\rho).$$

Based on the above, we can design a simple algorithm to solve a nonsmooth optimisation problem in a decentralised fashion. Moreover, this algorithm converges provided the following stopping criterion is satisfied:

$$\|\boldsymbol{\beta}^{\tau} - \mathbf{z}^{\tau}\|_2 \leq \epsilon_{primal}, \;\; \|\rho(\mathbf{z}^{\tau} - \mathbf{z}^{\tau-1})\|_2 \leq \epsilon_{dual},$$

where, the tolerances $\epsilon_{primal} > 0$ and $\epsilon_{dual} > 0$ can be set via an "absolute plus relative" criterion, e.g.

$$\begin{aligned}
\epsilon_{primal} &= \sqrt{n}\epsilon_{abs} + \epsilon_{rel} \max(\|\boldsymbol{\beta}^{\tau}\|_2, \|\mathbf{z}^{\tau}\|_2), \\
\epsilon_{dual} &= \sqrt{n}\epsilon_{abs} + \epsilon_{rel}\rho\|\mathbf{u}^{\tau}\|,
\end{aligned}$$

where ϵ_{abs} and ϵ_{rel} are absolute and relative tolerances. More details can be found in [27].

Distributed Computation When Number of Basis Function is Large ($N \gg M$)

When the number of candidate dictionary functions is very large, at least larger than the number of observations M, the associated computational complexity can become prohibitive. To alleviate this dictionary function scale-up problem, the creation of a distributed version of the algorithm becomes necessary. To achieve scale-up and allow parallelisation, we need to partition across the set of candidate functions in

order to define small optimisation sub-programmes. Each such sub-programme needs to deal with its split of candidate functions independently and then update the variables shared between the sub-programmes. This corresponds to a standard sharing problem (cf. [27]).

In our case, we partition the parameter vector $\boldsymbol{\beta}$ as $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n)$, with $\boldsymbol{\beta}_i \in \mathbb{R}^{N_i}$, where $\sum_{i=1}^n N_i = N$. We then accordingly partition the dictionary matrix $\mathbf{X}$ as $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_n]$, with $\mathbf{X}_i \in \mathbb{R}^{M \times N_i}$. Thus $\mathbf{X}\boldsymbol{\beta} = \sum_{i=1}^n \mathbf{X}_i\boldsymbol{\beta}_i$, i.e. $\mathbf{X}_i\boldsymbol{\beta}_i$ can be thought of as a 'partial' prediction of $\mathbf{y}$ using only the candidate functions referenced in $\boldsymbol{\beta}_i$. Let $\boldsymbol{\Theta}$ be a diagonal matrix with values $\sqrt{\alpha_j^k}$ on the diagonal and admitting the same partitioning as the variables $\boldsymbol{\beta}_i$, that is every matrix $\boldsymbol{\Theta}_j$ is a block of $\boldsymbol{\Theta}$ with indices in rows and columns spanning between $\sum_{i=1}^j N_i + 1$ and $\sum_{i=1}^{j+1} N_i$.

Now the update on $\boldsymbol{\beta}$, i.e. the optimisation programme defined in (6.102), becomes

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\sum_{i=1}^N \mathbf{X}_i\boldsymbol{\beta}_i - \mathbf{y}\|_2^2 + \lambda \sum_{i=1}^N \|\boldsymbol{\Theta}_i\boldsymbol{\beta}_i\|_1.$$

In order to apply ADMM, we introduce new variables $\mathbf{z}_i$ equal to $\mathbf{X}_i\boldsymbol{\beta}_i$. Doing so, we can rewrite the programme as follows:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\sum_{i=1}^N \mathbf{z}_i - \mathbf{y}\|_2^2 + \lambda \sum_{i=1}^N \|\boldsymbol{\Theta}_i\boldsymbol{\beta}_i\|_1, \\ \text{subject to} \quad & \mathbf{X}_i\boldsymbol{\beta}_i - \mathbf{z}_i = 0, \ i = 1, \ldots, N, \end{aligned}$$

with the new variables $\mathbf{z}_i \in \mathbb{R}^M$. The ADMM algorithm can be applied directly to this problem as follows:

$$\begin{aligned} \boldsymbol{\beta}_i^{\tau+1} &:= \operatorname*{argmin}_{\boldsymbol{\beta}_i} \frac{\rho}{2}\|\mathbf{X}_i\boldsymbol{\beta}_i - \mathbf{z}^k + \mathbf{u}^k\|_2^2 + \lambda\|\boldsymbol{\Theta}_i\boldsymbol{\beta}_i\|_1 \\ \mathbf{z}^{\tau+1} &:= \operatorname*{argmin}_{\mathbf{z}} \frac{1}{2}\|\sum_{i=1}^N \mathbf{z}_i - \mathbf{y}\|_2^2 \\ &\quad + \sum_{i=1}^N \frac{\rho}{2}\|\mathbf{X}_i\boldsymbol{\beta}_i^{\tau+1} - \mathbf{z}_i + \mathbf{u}_i^\tau\|_2^2 \\ \mathbf{u}_i^{\tau+1} &:= \mathbf{u}_i^\tau + \mathbf{X}\boldsymbol{\beta}^{\tau+1} - \mathbf{z}^{\tau+1}. \end{aligned} \tag{6.107}$$

where $\mathbf{z}$ stacks the variables $\mathbf{z}_i$ into a vector. The update on $\mathbf{z}$ is performed according to the sharing problem solution. That is, we first fix the average $\overline{\mathbf{z}} = 1/N \sum_{i=1}^N \mathbf{z}_i$ and

then solve the following programme with respect to $\mathbf{z}_i$:

$$\begin{aligned} \min_{\mathbf{z}_i} \ & \|N\overline{\mathbf{z}} - \mathbf{y}\|_2^2 + \sum_{i=1}^{N} \frac{\rho}{2} \|\mathbf{X}_i \boldsymbol{\beta}_i^{\tau+1} - \mathbf{z}_i + \mathbf{u}_i^{\tau}\|_2^2 \\ \text{subject to: } & \overline{\mathbf{z}} = 1/N \sum_{i=1}^{N} \mathbf{z}_i. \end{aligned}$$

This gives a closed form solution

$$\mathbf{z}_i^{\tau+1} = \overline{\mathbf{z}}^{\tau+1} + \mathbf{X}_i \boldsymbol{\beta}_i^{\tau+1} + \mathbf{u}^{\tau} - \overline{\mathbf{X}\boldsymbol{\beta}}^{\tau+1} - \overline{\mathbf{u}}^{\tau},$$

where $\overline{\mathbf{X}\boldsymbol{\beta}}^{\tau+1} = (1/N) \sum_{i=1}^{N} \mathbf{X}_i \boldsymbol{\beta}_i^{\tau+1}$ and $\overline{\mathbf{u}}^{\tau+1} = (1/N) \sum_{i=1}^{N} \mathbf{u}_i^{\tau+1}$. Hence the update on the whole vector $\mathbf{z}$ can be performed using a much smaller problem

$$\overline{\mathbf{z}}^{\tau+1} = \operatorname{argmin} \frac{1}{2} \|N\overline{\mathbf{z}} - \mathbf{y}\|_2^2 + \sum_{i=1}^{N} \frac{\rho}{2} \|\overline{\mathbf{z}} - \overline{\mathbf{X}\boldsymbol{\beta}^{\tau+1}} - \overline{\mathbf{u}}^{\tau}\|_2^2.$$

Additionally, substituting the expression for $\mathbf{z}_i^{\tau+1}$ into the update for $\mathbf{u}_i$, we find that

$$\mathbf{u}_i^{\tau+1} = \overline{\mathbf{X}\boldsymbol{\beta}}^{k+1} + \overline{\mathbf{u}}^{\tau} - \overline{\mathbf{z}}^{\tau+1},$$

which shows that, as in the sharing problem, all the dual variables are equal. Using a single dual variable $\mathbf{u}^k \in \mathbb{R}^m$, eliminating $\mathbf{z}_i$, and defining

$$\mathbf{b}_i^{\tau} = \mathbf{X}_i \boldsymbol{\beta}_i^{\tau} + \overline{\mathbf{z}}^{\tau} - \overline{\mathbf{X}\boldsymbol{\beta}}^{\tau} - \mathbf{u}^{\tau},$$

we arrive at:

$$\boldsymbol{\beta}_i^{\tau+1} = \operatorname*{argmin}_{\boldsymbol{\beta}_i} \left(\frac{\rho}{2} \|\mathbf{X}_i \boldsymbol{\beta}_i - \mathbf{b}_i^{\tau}\|_2^2 + \lambda \|\boldsymbol{\Theta}_i \boldsymbol{\beta}_i\|_1 \right), \tag{6.108}$$

$$\overline{\mathbf{z}}^{\tau+1} = \frac{1}{N+\rho} \left(\mathbf{y} + \rho \overline{\mathbf{X}\boldsymbol{\beta}}^{\tau+1} + \rho \mathbf{u}^k \right), \tag{6.109}$$

$$\mathbf{u}^{\tau+1} = \mathbf{u}^{\tau} + \overline{\mathbf{X}\boldsymbol{\beta}}^{\tau+1} - \overline{\mathbf{z}}^{\tau+1}. \tag{6.110}$$

The update of $\boldsymbol{\beta}_i$ is a non-smooth optimisation problem (6.108) that is solved again using ADMM as described above. Note, however, that it is not necessary to distribute this part of the computation, since the computation of the outer iteration loop is already distributed. Hence, ADMM here serves only as a solver for the non-smooth optimisation problem (6.108).

Distributed Computation When Number of Observations is Large ($M \gg N$)

When the number of observations, M, is very large or when data are naturally collected or stored in a distributed fashion, it is convenient or, even, indispensable to be able to recourse to a distributed version of the identification algorithm [27]. To this end, we partition the dictionary matrix $\mathbf{X}$ and data vector $\mathbf{y}$ as $\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_n \end{bmatrix}$, and $\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix}$, with $\mathbf{X}_i \in \mathbb{R}^{M_i \times N}$ where $\sum_{i=1}^{n} M_i = M$. In order to apply ADMM, we introduce new variables $\mathbf{z}_i$ equal to $\mathbf{X}_i \boldsymbol{\beta}_i$. Now the update on $\boldsymbol{\beta}$, i.e. the optimisation programme defined in (6.102), takes the following consensus form:

$$\begin{aligned} \min \quad & \frac{1}{2} \| \sum_{i=1}^{n} \mathbf{X}_i \boldsymbol{\beta}_i - \mathbf{y}_i \|_2^2 + \lambda \|\mathbf{z}\|_1, \\ \text{subject to} \quad & \boldsymbol{\Theta} \boldsymbol{\beta}_i - \mathbf{z} = 0, \ i = 1, \ldots, n. \end{aligned} \tag{6.111}$$

Based on this we arrive at the following distributed algorithm [27]:

$$\boldsymbol{\beta}_i^{k+1} = \underset{\boldsymbol{\beta}_i}{\operatorname{argmin}} \left(\frac{1}{2} \|\mathbf{X}_i \boldsymbol{\beta}_i - \mathbf{b}_i\|_2^2 + \frac{\rho}{2} \|\boldsymbol{\Theta} \boldsymbol{\beta}_i - \mathbf{z}^k + \mathbf{u}_i^k\|_2^2 \right), \tag{6.112}$$

$$\mathbf{z}^{k+1} = S_{\lambda/\rho} \left(\overline{\boldsymbol{\beta}}^{k+1} + \overline{\mathbf{u}}^k \right), \tag{6.113}$$

$$\mathbf{u}^{k+1} = \mathbf{u}_i^k + \boldsymbol{\beta}_i^{k+1} - \mathbf{z}^{k+1}. \tag{6.114}$$

The update for $\boldsymbol{\beta}_i$ in (6.112) takes the form of a ridge regression problem, with the following analytical solution:

$$\boldsymbol{\beta}_i^{k+1} = \left(\mathbf{X}_i^\top \mathbf{X}_i + \rho \boldsymbol{\Theta}^\top \boldsymbol{\Theta} \right)^{-1} \left(\mathbf{X}_i^\top \mathbf{y}_i + \rho (\mathbf{z}^k - \mathbf{u}_i^k) \right).$$

6.7 Algorithms for Chapter 3

Let's revisit the nonconvex optimisation problem in Section 3.3.2

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{n=1}^{N} \| \|\boldsymbol{\beta}_n\|_{\ell_2} \|_{\ell_0},$$

where λ is the regularisation parameter.

6.7.1 Sparse Prior for Chapter 3

The same as the sparse prior in (6.3), i.e.,

$$p(\boldsymbol{\beta}) = \prod_{i=1}^{N} p(\boldsymbol{\beta}_i) \tag{6.115}$$

where $p(\boldsymbol{\beta}_i)$ is structured as follows instead

$$p(\boldsymbol{\beta}_i) \propto \exp\left[-\frac{1}{2}\sum_{j=1}^{C} g(\beta_i^{[j]})\right] = \prod_{j=1}^{C} \exp\left[-\frac{1}{2} g(\beta_i^{[j]})\right] = \prod_{j=1}^{C} p(\beta_i^{[j]}), \tag{6.116}$$

with $g(\beta_i^{[j]})$ being a given function of $\beta_i^{[j]}$. Generally, $\boldsymbol{\beta}$ in (6.115) is sparse, and therefore certain sparsity properties should be enforced on $\boldsymbol{\beta}$. To this effect, the function $g(\cdot)$ is usually chosen to be a concave, non-decreasing function of $|\beta_i^{[j]}|$ [214]. Examples of such functions $g(\cdot)$ include Generalised Gaussian priors and Student's *t* priors (see [134, 214] for details).

Computing the posterior mean $\mathbb{E}(\boldsymbol{\beta}|\mathbf{y})$ is typically intractable because the posterior $p(\boldsymbol{\beta}|\mathbf{y})$ is highly coupled and non-Gaussian. To alleviate this problem, ideally one would like to approximate $p(\boldsymbol{\beta}|\mathbf{y})$ as a Gaussian distribution for which efficient algorithms to compute the posterior exist [22]. For this, the introduction of lower bounding *super-Gaussian* priors $p(\beta_i^{[j]})$, i.e.,

$$p(\beta_i^{[j]}) = \max_{\gamma_i > 0} \mathcal{N}(\beta_i^{[j]}|0, \gamma_i)\varphi(\gamma_i),$$

can be used to obtain an analytical approximation of $p(\boldsymbol{\beta}|\mathbf{y})$ [134].

Note that problem (9.8) has a block-wise structure, i.e. the solution $\boldsymbol{\beta}$ is expected to be block-wise sparse. Therefore, sparsity promoting priors should be specified for $p(\boldsymbol{\beta}_i)$, $\forall i$. To do this, for each block $\boldsymbol{\beta}_i$, we define a hyper-parameter γ_i such that

$$\begin{aligned} p(\boldsymbol{\beta}_i) &= \max_{\gamma_i > 0} \mathcal{N}(\boldsymbol{\beta}_i|\mathbf{0}, \gamma_i \mathbf{I}_C)\varphi(\gamma_i) \\ &= \max_{\gamma_i > 0} \prod_{j=1}^{C} \mathcal{N}(\beta_i^{[j]}|0, \gamma_i)\varphi(\gamma_i), \end{aligned} \tag{6.117}$$

where $\varphi(\gamma_i)$ is a nonnegative function, which is treated as a hyperprior with γ_i being its associated hyperparameter. Throughout, we call $\varphi(\gamma_i)$ the "*potential function*". This Gaussian relaxation is possible if and only if $\log p(\sqrt{\beta_i})$ is concave on $(0, \infty)$.

Then we have

$$p(\boldsymbol{\beta}) = \prod_{i=1}^{N} p(\boldsymbol{\beta}_i) = \max_{\boldsymbol{\gamma}>\mathbf{0}} \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \boldsymbol{\Gamma})\varphi(\boldsymbol{\gamma}), \tag{6.118}$$

where

$$\begin{aligned} \boldsymbol{\gamma}_i &= [\gamma_i, \ldots, \gamma_i] \in \mathbb{R}^C, \ \boldsymbol{\Gamma}_i = \operatorname{diag}[\boldsymbol{\gamma}_i], \\ \boldsymbol{\gamma} &= [\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_N] \in \mathbb{R}^{NC}, \ \boldsymbol{\Gamma} = \operatorname{diag}[\boldsymbol{\gamma}]. \end{aligned} \tag{6.119}$$

6.7.2 Optimisation Algorithm

Optimisation Problem Definition

The optimisation problem is defined exactly the same as (6.13) in Proposition 4, simply by replacing $\mathbf{B}$ with identity matrix.

Convex-Concave Procedure

Based the same optimisation principle in Section 6.4 and algorithm derivation in Section 6.5, we then obtain the following analytic form for the negative gradient of $v(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}$ (using the chain rule):

$$\begin{aligned} \boldsymbol{\alpha}^k \triangleq & -\nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}, \boldsymbol{\Theta}^*)^\top|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k} \\ =& \nabla_{\boldsymbol{\gamma}} \left[\log|\boldsymbol{\Gamma}^{-1} + \mathbf{X}^\top \boldsymbol{\Theta}^* \mathbf{X}| + \log|\boldsymbol{\Gamma}|\right] \\ =& \operatorname{diag}\{\left[-(\boldsymbol{\Gamma}^k)^{-1} + \mathbf{X}^\top \boldsymbol{\Theta}^* \mathbf{X}\right]^{-1}\} \cdot \operatorname{diag}\{-(\boldsymbol{\Gamma}^k)^{-2}\} + \operatorname{diag}^{-1}\{\boldsymbol{\Gamma}^k\} \\ =& \underbrace{\left[\ \boldsymbol{\alpha}_1^k \ \middle|\ \cdots \ \middle|\ \boldsymbol{\alpha}_N^k \ \right]}_{N \text{ Blocks}} \\ =& \left[\ \underbrace{\alpha_1^k, \ldots, \alpha_1^k}_{C \text{ Elements}} \ \middle|\ \cdots \ \middle|\ \underbrace{\alpha_N^k, \ldots, \alpha_N^k}_{C \text{ Elements}} \ \right]. \end{aligned} \tag{6.120}$$

Therefore, the iterative procedures (6.31) and (6.32) for $\boldsymbol{\beta}^{k+1}$ and $\boldsymbol{\gamma}^{k+1}$, respectively, can be formulated as

$$\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] = \underset{\boldsymbol{\gamma}\succ\mathbf{0}, \boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \boldsymbol{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \sum_{i=1}^{N} \left(\frac{\boldsymbol{\beta}_i^\top \boldsymbol{\beta}_i}{\gamma_i} + C\gamma_i \alpha_i^k\right). \tag{6.121}$$

The optimal $\boldsymbol{\gamma}$ components are obtained as:

$$\gamma_i = \frac{\|\boldsymbol{\beta}_i\|_2}{\sqrt{C\alpha_i^k}}. \tag{6.122}$$

If $\boldsymbol{\gamma}$ is fixed, we have

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^{\top} \boldsymbol{\Theta}^{*} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + 2 \sum_{i=1}^{N} \|w_i^k \cdot \boldsymbol{\beta}_i\|_2, \tag{6.123}$$

where

$$w_i^k = C\alpha_i^k. \tag{6.124}$$

We can then inject this into (6.122), which yields

$$\gamma_i^{k+1} = \frac{\|\boldsymbol{\beta}_i^{k+1}\|_2}{\sqrt{C\alpha_i^k}}. \tag{6.125}$$

The pseudo code is summarised in Algorithm 5.

Algorithm 5 Reweighted Group ℓ_1 type algorithm for Gaussian likelihood

1: Collect C heterogeneous groups of time series data from the system of interest (assuming the system can be described by (3.3));
2: Select the candidate basis functions that will be used to construct the dictionary matrix described;
3: Initialise the unknown w as a unit vector;
4: Fix/given the known inverse covariance matrix $\boldsymbol{\Pi}^{-1} = \boldsymbol{\Theta} = \boldsymbol{\Theta}^{*}$.
5: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^{+}$ as trail and error which may be empirically helpful;
6: **for** $k = 0, \ldots, k_{\max}$ **do**
7: Solve the following weighted minimisation problem over $\boldsymbol{\beta}$, subject to the convex constraints (2.41):

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^{\top} \boldsymbol{\Theta}^{*} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \lambda \sum_{i=1}^{N} \|w_i^k \cdot \boldsymbol{\beta}_i\|_2; \tag{6.126}$$

8: $\gamma_i^{k+1} = \frac{\|\boldsymbol{\beta}_i^{k+1}\|_2}{\sqrt{w_i^k}}, \boldsymbol{\gamma}_i^{k+1} = \left[\gamma_i^{k+1}, \ldots, \gamma_i^{k+1}\right] \in \mathbb{R}^{C}$;
9: $\boldsymbol{\gamma}^{k+1} = \left[\boldsymbol{\gamma}_1^{k+1}, \ldots, \boldsymbol{\gamma}_N^{k+1}\right] \in \mathbb{R}^{NC}$, $\boldsymbol{\Gamma}^{k+1} = \operatorname{diag}\left[\boldsymbol{\gamma}^{k+1}\right]$;
10: $\boldsymbol{\alpha}^{k+1} = \operatorname{diag}\{\left[-(\boldsymbol{\Gamma}^{k+1})^{-1} + \mathbf{X}^{\top}\boldsymbol{\Theta}^{*}\mathbf{X}\right]^{-1}\} \cdot \operatorname{diag}\{-(\boldsymbol{\Gamma}^{k+1})^{-2}\} + \operatorname{diag}^{-1}\{\boldsymbol{\Gamma}^{k+1}\}$;
11: $\boldsymbol{\alpha}^{k+1}$ is structured as $\Big[\underbrace{\alpha_1^{k+1}, \ldots, \alpha_1^{k+1}}_{C \text{ Elements}} \Big| \cdots \Big| \underbrace{\alpha_N^{k+1}, \ldots, \alpha_N^{k+1}}_{C \text{ Elements}} \Big]$;
12:
13: $w_i^{k+1} = C\alpha_i^{k+1}$;
14: **if** a stopping criterion is satisfied **then**
15: Break;
16: **end if**
17: **end for**

Connection to Semidefinite Programming and the Sparse Multiple Kernel Method

The iteration in (6.121) can be rewritten in the following compact form

$$\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] = \underset{\boldsymbol{\gamma}\succeq\mathbf{0},\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top} \boldsymbol{\Theta}^{*} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right) + \boldsymbol{\beta}^{\top}\boldsymbol{\Gamma}^{-1}\boldsymbol{\beta} - \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^{k}, \boldsymbol{\Theta}^{k})^{\top}\boldsymbol{\gamma}. \tag{6.127}$$

This is equivalent to the following Semidefinite Programming (SDP) by using the standard procedure in [28]

$$\begin{aligned} \min_{\mathbf{z},\boldsymbol{\beta},\boldsymbol{\gamma}} \quad & \mathbf{z} - \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^{k}, \boldsymbol{\Theta}^{k})^{\top}\boldsymbol{\gamma} \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{z} & \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top} & \boldsymbol{\beta}^{\top} \\ \mathbf{y}-\mathbf{X}\boldsymbol{\beta} & (\boldsymbol{\Theta}^{*})^{-1} & \mathbf{0} \\ \boldsymbol{\beta} & \mathbf{0} & \boldsymbol{\Gamma} \end{bmatrix} \succeq \mathbf{0} \\ & \boldsymbol{\gamma} \succeq \mathbf{0} \end{aligned}$$

The cost of solving this SDP is at least N^3 as well as M. Therefore, solving this SDP is too costly for all but problems with a small number of variables. This means that the number of samples, the dimension of the system, etc., can not be too large simultaneously. In this SDP formulation, $\boldsymbol{\Gamma}$ is closely related to the sparse multiple kernel presented in [35]. Certain choice of kernels may introduce some good properties or help reduce algorithmic complexity. In our case, we choose $\boldsymbol{\Gamma}$ to have a diagonal or a DC kernel structure.

ADMM Implementation

Essentially, Algorithm (5) consists of a reweighted Group Lasso algorithm (6.126). Algorithm (5) can be implemented using the Alternating Direction Method of Multipliers (ADMM) [27]. This ADMM parallelisation allows to distribute the algorithmic complexity to different threads and to build a platform for scalable distributed optimisation. This is key to be able to deal with problems of large dimensions.

More specifically, step on *Maximum a Posterior* estimation can be solved using ADMM instead:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right)^{\top} \boldsymbol{\Theta}^{*} \left(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right) + 2\sum_{i=1}^{N} \|\mathbf{z}_i\|_2, \\ \text{subject to} \quad & w_i^{k}\boldsymbol{\beta}_i - \mathbf{z}_i = 0, \quad i = 1, \ldots, N. \end{aligned} \tag{6.128}$$

The optimisation programmes with respect to different variables can be solved separately as follows:

$$\begin{aligned}
\boldsymbol{\beta}^{\tau+1} &= \left(\mathbf{X}^\top \boldsymbol{\Theta}^k \mathbf{X} + \rho \mathbf{I}\right)^{-1} \left(\mathbf{X}^\top \boldsymbol{\Theta}^k \mathbf{y} + \rho \left(\mathbf{z}^k - \mathbf{u}^k\right)\right), \\
\mathbf{z}_i^{\tau+1} &= \mathcal{S}_{\lambda/\rho}\left(\boldsymbol{\beta}_i^{\tau+1} + \mathbf{u}^\tau\right), \quad i = 1, \ldots, N, \\
\mathbf{u}^{\tau+1} &= \mathbf{u}^\tau + \boldsymbol{\beta}^{\tau+1} - \mathbf{z}^{\tau+1}.
\end{aligned}$$

$\mathcal{S}$ is the vector soft thresholding operator $\mathcal{S}_\kappa : \mathbb{R}^C \to \mathbb{R}^C$ is

$$\mathcal{S}_\kappa(a) = (1 - \kappa/\|a\|)_+ a, \tag{6.129}$$

where $\mathcal{S}_\kappa(0) = 0$. This formula reduces to the scalar soft thresholding operator when a is a scalar. More details can be found in [27].

6.8 Algorithms for Chapter 4

Revisit the optimisation problem (4.16) in Chapter 4

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{n=1}^{N} \|\mathbf{D}_n \boldsymbol{\beta}_n\|_{\ell_0} + \lambda_2 \sum_{n=1}^{N} \|\|\boldsymbol{\beta}_n\|_{\ell_2}\|_{\ell_0}, \tag{6.130}$$

and the ℓ_1 convex relaxation in (4.22)

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{n=1}^{N} \|\mathbf{D}_n \boldsymbol{\beta}_n\|_{\ell_1} + \lambda_2 \sum_{n=1}^{N} \|\boldsymbol{\beta}_n\|_{\ell_2}. \tag{6.131}$$

6.8.1 Sparse Prior for Chapter 4

As defined in (4.8) (drop the subscript i for simplicity), where in the bracket $m, m = 1, \ldots, M$ is the time index

$$\begin{aligned}
\boldsymbol{\beta} &\triangleq \left[\ \beta_1(1), \ \ldots, \ \beta_1(M) \ \middle|\ \ldots \ \middle|\ \beta_N(1), \ \ldots, \ \beta_N(M)\ \right]^\top \\
&= \left[\ \boldsymbol{\beta}_1^\top \ \middle|\ \ldots \ \middle|\ \boldsymbol{\beta}_N^\top\ \right]^\top \in \mathbb{R}^{MN}.
\end{aligned} \tag{6.132}$$

We first specially designed a new matrix a new $\mathbf{B}$ matrix:

$$\begin{aligned}
\mathbf{B} \triangleq \left[\begin{array}{c} \mathbf{I} \\ \hline \mathbf{D} \end{array}\right] &= \left[\begin{array}{ccc|ccc} \mathbf{I}_1 & & & & & \\ & \ddots & & & & \\ & & \mathbf{I}_N & & & \\ \hline \mathbf{D}_1 & & & & & \\ & \ddots & & & & \\ & & \mathbf{D}_N & & & \end{array}\right] \in \mathbb{R}^{\aleph \times MN}, \\
&\mathbf{I}_n \in \mathbb{R}^{M \times M}, \mathbf{D}_n \in \mathbb{R}^{\bar{\aleph} \times M}, \\
&\aleph = \sum_{n=1}^{N} M + \sum_{n=1}^{N} \bar{\aleph}.
\end{aligned} \tag{6.133}$$

Using the specially designed matrix $\mathbf{B}$, we can penalise a) the number of switches that occur using the upper part of $\mathbf{B}$ and b) the number of non-zero element in every identified model using the lower part of $\mathbf{B}$.

Next, we define the sparse prior. Fist, the hyperparameter vector is structured block-wise as follows:

$$\begin{aligned}
\tilde{\boldsymbol{\gamma}} &= \Big[\underbrace{\tilde{\gamma}_1, \ldots, \tilde{\gamma}_1}_{M \textbf{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\gamma}_N, \ldots, \tilde{\gamma}_N}_{M \textbf{ elements}} \Big], \\
\bar{\boldsymbol{\gamma}} &= \Big[\underbrace{\bar{\gamma}_{11}, \ldots, \bar{\gamma}_{1\bar{\aleph}}}_{\bar{\aleph} \textbf{ elements}} \Big| \ldots \Big| \underbrace{\bar{\gamma}_{N1}, \ldots, \bar{\gamma}_{N\bar{\aleph}}}_{\bar{\aleph} \textbf{ elements}} \Big], \\
\tilde{\boldsymbol{\Gamma}} &= \operatorname{diag}\left[\tilde{\boldsymbol{\gamma}}\right], \bar{\boldsymbol{\Gamma}} = \operatorname{diag}\left[\bar{\boldsymbol{\gamma}}\right], \\
\boldsymbol{\Gamma} &= \left[\begin{array}{cc} \tilde{\boldsymbol{\Gamma}} & \\ & \bar{\boldsymbol{\Gamma}} \end{array}\right] = \operatorname{diag}\left[\tilde{\boldsymbol{\gamma}}, \bar{\boldsymbol{\gamma}}\right],
\end{aligned} \tag{6.134}$$

then get the sparse prior as

$$\begin{aligned} p(\mathbf{B}\boldsymbol{\beta}) &= \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|0,\boldsymbol{\Gamma})\varphi(\boldsymbol{\Gamma}) \\ &= \prod_{n=1}^{N} p(\boldsymbol{\beta}_n) \cdot \prod_{n=1}^{N} p(\mathbf{D}_n\boldsymbol{\beta}_n) \\ &= \mathcal{N}(\boldsymbol{\beta}|0,\tilde{\boldsymbol{\Gamma}})\varphi(\tilde{\boldsymbol{\Gamma}}) \cdot \mathcal{N}(\mathbf{D}\boldsymbol{\beta}|0,\bar{\boldsymbol{\Gamma}})\varphi(\bar{\boldsymbol{\Gamma}}) \\ &= \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{\beta}_n|0,\tilde{\gamma}_n\mathbf{I}_{\bar{\aleph}})\varphi(\tilde{\gamma}_n) \cdot \prod_{n=1}^{N} \mathcal{N}(\mathbf{D}_n\boldsymbol{\beta}_n|0,\bar{\gamma}_n\mathbf{I}_{\bar{\aleph}})\varphi(\bar{\gamma}_n) \\ &= \prod_{n=1}^{N}\prod_{i=1}^{M} \mathcal{N}(\{\mathbf{I}_n\}_{i,:}\,\boldsymbol{\beta}_n|0,\tilde{\gamma}_n)\varphi(\tilde{\gamma}_n) \cdot \prod_{n=1}^{N}\prod_{i=1}^{\bar{\aleph}} \mathcal{N}(\{\mathbf{D}_n\}_{i,:}\,\boldsymbol{\beta}_n|0,\gamma_n)\varphi(\bar{\gamma}_n). \end{aligned} \tag{6.135}$$

6.8.2 Optimisation Algorithm

Optimisation Problem Definition

The optimisation problem is defined exactly the same as (6.13) in Proposition 4.

Convex-Concave Procedure

We first look at the hyperparameter $\tilde{\boldsymbol{\Gamma}}$. Since

$$\sum_{i=1}^{M}\left(\frac{\boldsymbol{\beta}_n^\top\boldsymbol{\beta}_n}{\tilde{\gamma}_n} + \tilde{\alpha}_n^k\tilde{\gamma}_n\right) \geq 2\left\|\sqrt{M\tilde{\alpha}_n^k}\cdot\boldsymbol{\beta}_n\right\|_{\ell_2}, \tag{6.136}$$

the optimal $\boldsymbol{\gamma}$ can be obtained as:

$$\tilde{\gamma}_n^{k+1} = \frac{\|\boldsymbol{\beta}_n\|_{\ell_2}}{\sqrt{M\tilde{\alpha}_n^k}}, \forall i. \tag{6.137}$$

Recall the expression for $\boldsymbol{\alpha}^k$ in (6.36)

$$\begin{aligned} \boldsymbol{\alpha}^k \triangleq& \nabla_\gamma v(\boldsymbol{\gamma},\boldsymbol{\Pi})^\top|_{\gamma=\gamma^k} \\ =& \nabla_\gamma\left(-\log|\boldsymbol{\Theta}^*| + \log|\boldsymbol{\Gamma}| + \log|\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X}|\right)^\top|_{\gamma=\gamma^k} \\ =& -\operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\} \circ \operatorname{diag}\left\{\mathbf{B}(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X})^{-1}\mathbf{B}^\top\right\} \circ \operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\} \\ & + \operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\}, \\ =& \begin{bmatrix} \alpha_1^k & \cdots & \alpha_N^k \end{bmatrix} \end{aligned}$$

where $\boldsymbol{\Pi}^{-1} = \boldsymbol{\Theta} = \boldsymbol{\Theta}^*$. Slightly different from the above expression, $\tilde{\boldsymbol{\alpha}}^k$ can be derived in a decomposed way

$$\begin{aligned}
\tilde{\boldsymbol{\alpha}}^k =& \nabla_{\tilde{\boldsymbol{\gamma}}} v(\tilde{\boldsymbol{\gamma}}, \bar{\boldsymbol{\gamma}}^k, \boldsymbol{\Theta}^*)^\top|_{\tilde{\boldsymbol{\gamma}}=\tilde{\boldsymbol{\gamma}}^k} \\
=& \nabla_{\tilde{\boldsymbol{\gamma}}} \left[\log|(\tilde{\boldsymbol{\Gamma}})^{-1} + \mathbf{D}^\top(\bar{\boldsymbol{\Gamma}}^k)^{-1}\mathbf{D} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X}| + \log|\tilde{\boldsymbol{\Gamma}}| + \log|\bar{\boldsymbol{\Gamma}}^k|\right]|_{\tilde{\boldsymbol{\gamma}}=\tilde{\boldsymbol{\gamma}}^k} \\
=& \operatorname{diag}\left\{\left((\tilde{\boldsymbol{\Gamma}}^k)^{-1} + \mathbf{D}^\top(\bar{\boldsymbol{\Gamma}}^k)^{-1}\mathbf{D} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X}\right)^{-1}\right\} \cdot \operatorname{diag}\left\{-(\tilde{\boldsymbol{\Gamma}}^k)^{-2}\right\} \\
&+ \operatorname{diag}^{-1}\left\{\tilde{\boldsymbol{\Gamma}}^k\right\}, \\
=& \Big[\underbrace{\tilde{\alpha}^k_{11}, \ldots, \tilde{\alpha}^k_{1M}}_{M \text{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\alpha}^k_{N1}, \ldots, \tilde{\alpha}^k_{NM}}_{M \text{ elements}} \Big].
\end{aligned} \tag{6.138}$$

It is found that $\tilde{\alpha}^k_n$ is a function of $\tilde{\gamma}^k_n$. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}$ first to calculate $\tilde{\boldsymbol{\gamma}}^{k+1}$.

We then look at the hyperparameter $\bar{\boldsymbol{\Gamma}}$. Since

$$\frac{\boldsymbol{\beta}_n^\top \{\mathbf{D}_n\}_{i,:}^\top \{\mathbf{D}_n\}_{i,:} \boldsymbol{\beta}_n}{\bar{\gamma}_{ni}} + \bar{\alpha}^k_{ni}\bar{\gamma}_{ni} \geq 2\left|\sqrt{\bar{\alpha}^k_{ni}} \cdot \{\mathbf{D}_n\}_{i,:} \boldsymbol{\beta}_n\right| \tag{6.139}$$

the optimal $\bar{\boldsymbol{\gamma}}$ can be obtained as

$$\bar{\gamma}^{k+1}_{ni} = \frac{|\{\mathbf{D}_n\}_{i,:} \boldsymbol{\beta}_n|}{\sqrt{\bar{\alpha}^k_{ni}}}, \forall n = 1, \ldots, N, i = 1, \ldots, \bar{\aleph}. \tag{6.140}$$

And $\bar{\boldsymbol{\alpha}}^k$ is defined as

$$\bar{\boldsymbol{\alpha}}^k = \Big[\underbrace{\bar{\alpha}^k_{11}, \ldots, \bar{\alpha}^k_{1\bar{\aleph}}}_{\bar{\aleph} \text{ elements}} \Big| \ldots \Big| \underbrace{\bar{\alpha}^k_{N1}, \ldots, \bar{\alpha}^k_{N\bar{\aleph}}}_{\bar{\aleph} \text{ elements}} \Big]. \tag{6.141}$$

It should be noted that, we can obtain the following analytic form for the negative gradient of $v(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}$ using basic principles in convex analysis as (using chain rule):

$$\begin{aligned}
\bar{\boldsymbol{\alpha}}^k =& \nabla_{\bar{\boldsymbol{\gamma}}} v(\tilde{\boldsymbol{\gamma}}^k, \bar{\boldsymbol{\gamma}}, \boldsymbol{\Theta}^*)^\top|_{\bar{\boldsymbol{\gamma}}=\bar{\boldsymbol{\gamma}}^k} \\
=& \nabla_{\bar{\boldsymbol{\gamma}}} \left[\log|(\tilde{\boldsymbol{\Gamma}}^k)^{-1} + \mathbf{D}^\top\bar{\boldsymbol{\Gamma}}^{-1}\mathbf{D} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X}| + \log|\tilde{\boldsymbol{\Gamma}}^k| + \log|\bar{\boldsymbol{\Gamma}}|\right]|_{\bar{\boldsymbol{\gamma}}=\bar{\boldsymbol{\gamma}}^k} \\
=& \operatorname{diag}\left\{\mathbf{D}\left((\tilde{\boldsymbol{\Gamma}}^k)^{-1} + \mathbf{D}^\top(\bar{\boldsymbol{\Gamma}}^k)^{-1}\mathbf{D} + \mathbf{X}^\top\boldsymbol{\Theta}^*\mathbf{X}\right)^{-1}\mathbf{D}^\top\right\} \cdot \operatorname{diag}\left\{-(\bar{\boldsymbol{\Gamma}}^k)^{-2}\right\} \\
&+ \operatorname{diag}^{-1}\left\{\bar{\boldsymbol{\Gamma}}^k\right\} \\
=& \Big[\underbrace{\bar{\alpha}^k_{11}, \ldots, \bar{\alpha}^k_{1\bar{\aleph}}}_{\bar{\aleph} \text{ elements}} \Big| \ldots \Big| \underbrace{\bar{\alpha}^k_{N1}, \ldots, \bar{\alpha}^k_{N\bar{\aleph}}}_{\bar{\aleph} \text{ elements}} \Big].
\end{aligned} \tag{6.142}$$

It is found that $\bar{\alpha}^k_{ni}$ is a function of $\bar{\gamma}^k_{ni}$. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}_n$ first to calculate $\bar{\boldsymbol{\gamma}}^{k+1}_{ni}$.

Two new variables are defined, i.e.,

$$\tilde{w}_n^k \triangleq \sqrt{\sum_{i=1}^{M} \tilde{\alpha}_{ni}^k}, \tag{6.143}$$

and

$$\bar{w}_{ni}^k \triangleq \sqrt{\bar{\alpha}_{ni}^k}. \tag{6.144}$$

The pseudo code is summarised in Algorithm 6.

Algorithm 6 Reweighted *Fused Group* ℓ_1 type algorithm for Gaussian likelihood

1: Select the candidate basis functions that will be used to construct the dictionary matrix described;
2: Initialise $\tilde{w}_n^k = 1, \bar{w}_{ni}^k = 1,\ \forall i$;
3: Fix/given the known inverse covariance matrix $\mathbf{\Pi}^{-1} = \mathbf{\Theta} = \mathbf{\Theta}^*$;
4: Fix $\tilde{\lambda} = 1$ and $\bar{\lambda} = 1$ or select $\tilde{\lambda} \in \mathbb{R}^+$ and $\bar{\lambda} \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
5: **for** $k = 1, \ldots, k_{\max}$ **do**
6: Solve the following weighted *Fused Group Lasso* type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^\top \mathbf{\Theta}^* \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \tilde{\lambda} \sum_{n=1}^{N} \left\| \tilde{w}_n^k \cdot \boldsymbol{\beta}_n \right\|_{\ell_2} + \bar{\lambda} \sum_{n=1}^{N} \sum_{i=1}^{\bar{\aleph}} \left\| \bar{w}_{ni}^k \cdot \{\mathbf{D}_n\}_{i,:} \bar{\boldsymbol{\beta}}_n \right\|_{\ell_1}; \tag{6.145}$$

7: $\tilde{\gamma}_n^{k+1} = \frac{\left\|\boldsymbol{\beta}_n^{k+1}\right\|_{\ell_2}}{\tilde{w}_n^k}$, and $\tilde{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\tilde{\gamma}_1^{k+1}, \ldots, \tilde{\gamma}_1^{k+1}}_{M \text{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\gamma}_N^{k+1}, \ldots, \tilde{\gamma}_N^{k+1}}_{M \text{ elements}} \Big]$;
8: $\bar{\gamma}_{ni}^{k+1} = \frac{|\{\mathbf{D}_n\}_{i,:}\boldsymbol{\beta}_n^{k+1}|}{\bar{w}_{ni}^k}, \forall n, i$, and $\bar{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\bar{\gamma}_{11}^{k+1}, \ldots, \bar{\gamma}_{1\bar{\aleph}}^{k+1}}_{\bar{\aleph} \text{ elements}} \Big| \ldots \Big| \underbrace{\bar{\gamma}_{N1}^{k+1}, \ldots, \bar{\gamma}_{N\bar{\aleph}}^{k+1}}_{\bar{\aleph} \text{ elements}} \Big]$;
9: $\mathbf{C}^{k+1} = \left((\tilde{\mathbf{\Gamma}}^{k+1})^{-1} + \mathbf{D}^\top (\bar{\mathbf{\Gamma}}^{k+1})^{-1} \mathbf{D} + \mathbf{X}^\top \mathbf{\Theta}^* \mathbf{X} \right)^{-1}$;
10: $\tilde{\alpha}_n^{k+1} = -\frac{\mathbf{C}^{k+1}}{(\tilde{\gamma}_n^{k+1})^2} + \frac{1}{\tilde{\gamma}_n^{k+1}}$;
11: $\tilde{w}_n^{k+1} = \sqrt{M \cdot \tilde{\alpha}_n^{k+1}}$;
12: $\bar{\alpha}_{ni}^{k+1} = -\frac{\{\mathbf{D}_n\}_{i,:}\mathbf{C}^{k+1}\{\mathbf{D}_n\}_{i,:}^\top}{(\bar{\gamma}_n^{k+1})^2} + \frac{1}{\bar{\gamma}_n^{k+1}}$;
13: $\bar{w}_{ni}^{k+1} = \sqrt{\bar{\alpha}_{ni}^{k+1}}$;
14: **if** a stopping criterion is satisfied **then**
15: Break;
16: **end if**
17: **end for**

Chapter 7

Algorithms for Likelihood in Exponential Family

In this Chapter, we go beyond the nonlinear system where the basis functions are assumed to be "linear in parameters" like Assumption 2. We will consider the more general model class like Assumption 3. Furthermore, we will cover the data likelihood beyond Gaussian distribution. Therefore, we move from linear regression with structural sparsity to the more general nonlinear regression with structural sparsity problems. Similar to Chapter 6, this chapter is organised as follows. In Section 7.1, we introduce the data likelihood considered in this Chapter, belonging to exponential family. In Section 7.2, the sparse prior and some variations will be introduced as a controller for the structural sparsity. Next in Section 7.3, the optimisation problem from a Bayesian perspective will be defined. Then in Section 7.5, the general optimisation algorithms are proposed. In the last Section, optimisation algorithms for two special prior will be proposed respectively.

7.1 Likelihood in Exponential Family

Assume the the distribution of data likelihood belongs to exponential family [153, 46, 105], i.e.,

$$\begin{aligned} p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) &= h(\boldsymbol{\beta})a(\boldsymbol{\theta})\exp\left(\sum_{s=1}^{S}\eta_s(\boldsymbol{\theta})\cdot T_s(\boldsymbol{\beta})\right) \\ &= a(\boldsymbol{\theta})\exp\left(\sum_{s=1}^{S}\eta_s(\boldsymbol{\theta})\cdot T_s(\boldsymbol{\beta}) + B(\boldsymbol{\beta})\right) \\ &\triangleq a(\boldsymbol{\theta})\exp\left(-E(\boldsymbol{\beta},\boldsymbol{\theta})\right) \end{aligned} \tag{7.1}$$

where $E(*)$ is called the *energy function*.

The exponential families include many of the most commonly used probability distributions. Among many others, the family includes the following: normal, exponential, gamma, chi-squared, beta, Dirichlet, Bernoulli, categorical, Poisson, Wishart, Inverse Wishart. A number of common distributions are exponential families, but only when certain parameters are fixed and known. For example: binomial (with fixed number of trials), multinomial (with fixed number of trials), negative binomial (with fixed number of failures). Most of the distributions can be covered by exponential family.

7.2 Sparse Prior

7.2.1 Generalised Sparse Prior

We still employ the sparse prior in super-Gaussian distribution in Section 6.2 of Chapter 6, i.e.,

$$\begin{aligned} p(\mathbf{B}\boldsymbol{\beta}) &= \prod_{i=1}^{\aleph} \mathcal{N}(\mathbf{B}_{i,:}\boldsymbol{\beta}|0,\gamma_i)\varphi(\gamma_i) \\ &= \max_{\boldsymbol{\gamma}\succ\mathbf{0}} \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|0,\boldsymbol{\Gamma})\varphi(\boldsymbol{\gamma}), \end{aligned}$$

where

$$\boldsymbol{\gamma} = [\gamma_1,\ldots,\gamma_{\aleph}] \in \mathbb{R}^{\aleph},\ \boldsymbol{\Gamma} = \operatorname{diag}[\boldsymbol{\gamma}].$$

In this Section, we vary the structure of $\mathbf{B}$ with easy identified name which will be used throughout the thesis. The "generalised" sparse prior is named after the "generalised lasso" in [190].

$\mathbf{B}_{\text{Identity}}$ First, we introduce the simplest case, identity matrix,

$$\mathbf{B} = \mathbf{B}_{\text{Identity}} \triangleq \mathbf{I} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \in \mathbb{R}^{\aleph\times\aleph}. \tag{7.2}$$

Like (4.12) in Section 4.4.2, we introduce $\mathbf{B}_{\text{Switch-I}}$

$$\begin{aligned} \mathbf{B} = \mathbf{B}_{\text{Switch-I}} &\triangleq \begin{bmatrix} \mathbf{D}_1 & & \\ & \ddots & \\ & & \mathbf{D}_O \end{bmatrix} \in \mathbb{R}^{\aleph\times N}, \\ \mathbf{D}_o &\triangleq \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{\aleph_o\times(\aleph_o+1)}, \\ \aleph = \sum_{o=1}^{O} \aleph_o,\ & N = \sum_{o=1}^{O}(\aleph_o+1), o = 1,\ldots,O. \end{aligned} \tag{7.3}$$

Rather than diagonalise the $\mathbf{D}$ matrix, we introduce $\mathbf{B}_{\text{Switch-II}}$

$$\begin{aligned}
\mathbf{B} = \mathbf{B}_{\text{Switch-II}} &\triangleq \begin{bmatrix} \mathbf{D}_1 & \dots & \mathbf{D}_O \end{bmatrix} \in \mathbb{R}^{\aleph \times N}, \\
\mathbf{D}_o &\triangleq \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{\aleph_o \times (\aleph_o + 1)}, \\
\aleph = \aleph_o, N &= \sum_{o=1}^{O} (\aleph_o + 1), o = 1, \dots, O.
\end{aligned} \tag{7.4}$$

Like the one introduced in (4.23) for trend filtering, we define $\mathbf{B}_{\text{Trend-I}}$

$$\begin{aligned}
\mathbf{B} = \mathbf{B}_{\text{Trend-I}} &\triangleq \begin{bmatrix} \mathbf{D}_1 & & \\ & \ddots & \\ & & \mathbf{D}_O \end{bmatrix} \in \mathbb{R}^{\aleph \times N}, \\
\mathbf{D}_o &\triangleq \begin{bmatrix} 1 & -2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{\aleph_o \times (\aleph_o + 2)}, \\
\aleph = \sum_{o=1}^{O} \aleph_o, N &= \sum_{o=1}^{O} (\aleph_o + 2), o = 1, \dots, O.
\end{aligned} \tag{7.5}$$

Rather than diagonalise the $\mathbf{D}$ matrix, we introduce $\mathbf{B}_{\text{Trend-II}}$

$$\begin{aligned}
\mathbf{B} = \mathbf{B}_{\text{Trend-II}} &\triangleq \begin{bmatrix} \mathbf{D}_1 & \dots & \mathbf{D}_C \end{bmatrix} \in \mathbb{R}^{\aleph \times N}, \\
\mathbf{D}_o &\triangleq \begin{bmatrix} 1 & -2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{\aleph_o \times (\aleph_o + 2)}, \\
\aleph = \aleph_o, N &= \sum_{o=1}^{O} (\aleph_o + 2), o = 1, \dots, O.
\end{aligned} \tag{7.6}$$

7.2.2 Group Sparse Prior

In this Section, suppose $\boldsymbol{\beta}$ is structured block-wise as

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \hline \vdots \\ \hline \boldsymbol{\beta}_O \end{bmatrix} \in \mathbb{R}^N \tag{7.7}$$

where $\boldsymbol{\beta}_o \in \mathbb{R}^{N_o}$ and $N = \sum_{o=1}^{O} N_o$. Furthermore, we introduce the $\mathbf{B}$ matrix as $\mathbf{B}_{\text{Group}}$

$$\begin{aligned} \mathbf{B} = \mathbf{B}_{\text{Group}} &\triangleq \begin{bmatrix} \mathbf{B}_1 & & \\ & \ddots & \\ & & \mathbf{B}_O \end{bmatrix} \in \mathbb{R}^{\aleph \times N}, \\ \mathbf{B}_o \in \mathbb{R}^{\aleph_o \times N_o}, \aleph &= \sum_{o=1}^{O} \aleph_o, N = \sum_{o=1}^{O} N_o. \end{aligned} \tag{7.8}$$

Then we have

$$\mathbf{B}\boldsymbol{\beta} = \mathbf{B}\begin{bmatrix} \boldsymbol{\beta}_1 \\ \hline \vdots \\ \hline \boldsymbol{\beta}_O \end{bmatrix} = \begin{bmatrix} \mathbf{B}_1\boldsymbol{\beta}_1 \\ \hline \vdots \\ \hline \mathbf{B}_O\boldsymbol{\beta}_O \end{bmatrix} = \begin{bmatrix} \{\mathbf{B}_1\}_{1,:}\boldsymbol{\beta}_1 \\ \vdots \\ \hline \{\mathbf{B}_1\}_{\aleph_1,:}\boldsymbol{\beta}_1 \\ \hline \vdots \\ \hline \{\mathbf{B}_O\}_{1,:}\boldsymbol{\beta}_O \\ \vdots \\ \{\mathbf{B}_O\}_{\aleph_O,:}\boldsymbol{\beta}_O \end{bmatrix}. \tag{7.9}$$

Next, we define the sparse prior. Fist, the hyperparameter vector is structured block-wise as follows:

$$\begin{aligned} \boldsymbol{\gamma} &= \Big[\underbrace{\gamma_1, \ldots, \gamma_1}_{\aleph_1 \text{ elements}} \Big| \cdots \Big| \underbrace{\gamma_O, \ldots, \gamma_O}_{\aleph_O \text{ elements}} \Big], \\ \boldsymbol{\Gamma} &= \operatorname{diag}[\boldsymbol{\gamma}]. \end{aligned} \tag{7.10}$$

and define the sparse prior as

$$\begin{aligned} p(\mathbf{B}\boldsymbol{\beta}) &= \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|0, \boldsymbol{\Gamma})\varphi(\boldsymbol{\Gamma}) \\ &= \prod_{o=1}^{O} p(\mathbf{B}_o\boldsymbol{\beta}_o) \\ &= \prod_{o=1}^{O} \mathcal{N}(\mathbf{B}_o\boldsymbol{\beta}_o|0, \gamma_o\mathbf{I}_{\aleph})\varphi(\gamma_o) \\ &= \prod_{o=1}^{O}\prod_{i=1}^{\aleph_o} \mathcal{N}(\{\mathbf{B}_o\}_{i,:}\boldsymbol{\beta}_o|0, \gamma_o)\varphi(\gamma_o). \end{aligned} \tag{7.11}$$

The algorithm is introduced later in Section 7.5.1.

7.2.3 Fused Sparse Prior

In this Section, suppose $\boldsymbol{\beta}$ is structured block-wise as follows:

$$\boldsymbol{\beta} = \left[\begin{array}{c} \tilde{\boldsymbol{\beta}}_1 \\ \hline \vdots \\ \hline \tilde{\boldsymbol{\beta}}_{\tilde{O}} \end{array}\right] = \left[\begin{array}{c} \bar{\boldsymbol{\beta}}_1 \\ \hline \vdots \\ \hline \bar{\boldsymbol{\beta}}_{\bar{O}} \end{array}\right]$$

where $\tilde{\boldsymbol{\beta}}_{\tilde{O}} \in \mathbb{R}^{\tilde{N}_{\tilde{o}}}$ and $\bar{\boldsymbol{\beta}}_{\bar{O}} \in \mathbb{R}^{\bar{N}_{\bar{o}}}$ and $N = \sum_{\tilde{o}=1}^{\tilde{O}} \tilde{N}_{\tilde{o}} = \sum_{\bar{o}=1}^{\bar{O}} \bar{N}_{\bar{o}}$. It should be noted that $\tilde{\boldsymbol{\beta}}_{\tilde{o}}$ may have different dimension with $\bar{\boldsymbol{\beta}}_{\bar{O}}$, i.e. $\tilde{N}_{\tilde{o}} \neq \bar{N}_{\bar{o}}$. Furthermore, we introduce the $\mathbf{B}$ matrix as $\mathbf{B}_{\text{Group}}$, where the corresponding $\mathbf{B}$ matrix as $\mathbf{B}_{\text{Fused}}$

$$\begin{aligned} \mathbf{B} = \mathbf{B}_{\text{Fused}} \triangleq \left[\begin{array}{c} \tilde{\mathbf{B}} \\ \hline \bar{\mathbf{B}} \end{array}\right] = \left[\begin{array}{ccc} \tilde{\mathbf{B}}_1 & & \\ & \ddots & \\ & & \tilde{\mathbf{B}}_{\tilde{O}} \\ \hline \bar{\mathbf{B}}_1 & & \\ & \ddots & \\ & & \bar{\mathbf{B}}_{\bar{O}} \end{array}\right] \in \mathbb{R}^{\aleph \times N}, \\ \tilde{\mathbf{B}}_{\tilde{o}} \in \mathbb{R}^{\tilde{\aleph}_{\tilde{o}} \times \tilde{N}_{\tilde{o}}}, \bar{\mathbf{B}}_{\bar{o}} \in \mathbb{R}^{\bar{\aleph}_{\bar{o}} \times \bar{N}_{\bar{o}}}, \\ \aleph = \sum_{\tilde{o}=1}^{\tilde{O}} \tilde{\aleph}_{\tilde{o}} + \sum_{\bar{o}=1}^{\bar{O}} \bar{\aleph}_{\bar{o}}, N = \sum_{\tilde{o}=1}^{\tilde{O}} \tilde{N}_{\tilde{o}} = \sum_{\bar{o}=1}^{\bar{O}} \bar{N}_{\bar{o}}. \end{aligned} \tag{7.12}$$

Next, we define the sparse prior. First, the hyperparameter vector is structured block-wise as follows:

$$\begin{aligned} \tilde{\boldsymbol{\gamma}} &= \Big[\underbrace{\tilde{\gamma}_1, \ldots, \tilde{\gamma}_1}_{\tilde{\aleph}_1 \text{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\gamma}_{\tilde{O}}, \ldots, \tilde{\gamma}_{\tilde{O}}}_{\tilde{\aleph}_{\tilde{O}} \text{ elements}} \Big], \\ \bar{\boldsymbol{\gamma}} &= \Big[\underbrace{\bar{\gamma}_{11}, \ldots, \bar{\gamma}_{1\bar{\aleph}_1}}_{\bar{\aleph}_1 \text{ elements}} \Big| \ldots \Big| \underbrace{\bar{\gamma}_{\bar{O}1}, \ldots, \bar{\gamma}_{\bar{O}\bar{\aleph}_{\bar{O}}}}_{\bar{\aleph}_{\bar{O}} \text{ elements}} \Big], \\ \tilde{\boldsymbol{\Gamma}} &= \operatorname{diag}\left[\tilde{\boldsymbol{\gamma}}\right], \bar{\boldsymbol{\Gamma}} = \operatorname{diag}\left[\bar{\boldsymbol{\gamma}}\right], \\ \boldsymbol{\Gamma} &= \left[\begin{array}{cc} \tilde{\boldsymbol{\Gamma}} & \\ & \bar{\boldsymbol{\Gamma}} \end{array}\right] = \operatorname{diag}\left[\tilde{\boldsymbol{\gamma}}, \bar{\boldsymbol{\gamma}}\right]. \end{aligned} \tag{7.13}$$

Then define the sparse prior as

$$\begin{aligned} p(\mathbf{B}\boldsymbol{\beta}) &= \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|0,\boldsymbol{\Gamma})\varphi(\boldsymbol{\Gamma}) \\ &= \mathcal{N}(\tilde{\mathbf{B}}\boldsymbol{\beta}|0,\tilde{\boldsymbol{\Gamma}})\varphi(\tilde{\boldsymbol{\Gamma}})\cdot\mathcal{N}(\bar{\mathbf{B}}\boldsymbol{\beta}|0,\bar{\boldsymbol{\Gamma}})\varphi(\bar{\boldsymbol{\Gamma}}) \\ &= \prod_{\tilde{o}=1}^{\tilde{O}} p(\tilde{\mathbf{B}}_{\tilde{o}}\tilde{\boldsymbol{\beta}}_{\tilde{o}})\cdot\prod_{\bar{o}=1}^{\bar{O}} p(\bar{\mathbf{B}}_{\bar{o}}\bar{\boldsymbol{\beta}}_{\bar{o}}) \\ &= \prod_{\tilde{o}=1}^{\tilde{O}} \mathcal{N}(\tilde{\mathbf{B}}_{\tilde{o}}\tilde{\boldsymbol{\beta}}_{\tilde{o}}|0,\tilde{\gamma}_{\tilde{o}}\mathbf{I}_{\tilde{\aleph}})\varphi(\tilde{\gamma}_{\tilde{o}})\cdot\prod_{\bar{o}=1}^{\bar{O}} \mathcal{N}(\bar{\mathbf{B}}_{\bar{o}}\bar{\boldsymbol{\beta}}_{\bar{o}}|0,\bar{\gamma}_{\bar{o}}\mathbf{I}_{\bar{\aleph}})\varphi(\bar{\gamma}_{\bar{o}}) \\ &= \prod_{\tilde{o}=1}^{\tilde{O}}\prod_{i=1}^{\tilde{\aleph}_{\tilde{o}}} \mathcal{N}(\{\tilde{\mathbf{B}}_{\tilde{o}}\}_{i,:}\tilde{\boldsymbol{\beta}}_{\tilde{o}}|0,\tilde{\gamma}_{\tilde{o}})\varphi(\tilde{\gamma}_{\tilde{o}})\cdot\prod_{\bar{o}=1}^{\bar{O}}\prod_{i=1}^{\bar{\aleph}_{\bar{o}}} \mathcal{N}(\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}|0,\bar{\gamma}_{\bar{o}})\varphi(\bar{\gamma}_{\bar{o}}). \end{aligned} \tag{7.14}$$

The algorithm is summarised in Section 7.5.2.

7.3 Optimisation Problem Definition

Once we introduce the likelihood in (7.1) and the variational prior in (6.9), we can get the following approximated optimisation problem jointly on $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\Theta}$.

Proposition 10 *Given the likelihood with exponential family distribution* $p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta})\exp\left(-E(\boldsymbol{\beta},\boldsymbol{\theta})\right)$ *as in* (7.1) *and sparse prior with super Gaussian distribution* $p(\mathbf{B}\boldsymbol{\beta}) = \max_{\boldsymbol{\gamma}\succ\mathbf{0}}\mathcal{N}(\mathbf{B}\boldsymbol{\beta}|0,\boldsymbol{\Gamma})\varphi(\boldsymbol{\gamma})$ *as in* (6.9)*, the unknown parameter* $\boldsymbol{\beta}$*, hyperparameter* $\boldsymbol{\gamma}$*, and parameter of the family* $\boldsymbol{\theta}$ *can be approximately obtained by solving the following optimisation problem*

$$\min_{\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) \tag{7.15}$$

with

$$\begin{aligned} \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) &= \boldsymbol{\beta}^\top\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta} + 2\boldsymbol{\beta}^\top\left[\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta}^*\right] + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} \\ &+ \log|\boldsymbol{\Gamma}| + \log|\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) + \mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}| - 2\log a(\boldsymbol{\theta})\cdot b(\boldsymbol{\beta}^*,\boldsymbol{\theta}) - 2\sum_{i=1}^{\aleph}\log\varphi(\gamma_i) \end{aligned} \tag{7.16}$$

where $\boldsymbol{\beta}^*$ *is arbitrary, and*

$$\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) \triangleq \nabla E(\boldsymbol{\beta},\boldsymbol{\theta})|_{\boldsymbol{\beta}^*}, \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) \triangleq \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta})|_{\boldsymbol{\beta}^*}.$$

and

$$b(\boldsymbol{\beta}^*,\boldsymbol{\theta}) \triangleq \exp\left\{-\left(\frac{1}{2}\boldsymbol{\beta}^{*\top}\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta}^* - \boldsymbol{\beta}^{*\top}\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) + E(\boldsymbol{\beta}^*,\boldsymbol{\theta})\right)\right\}.$$

Proof *Given the likelihood with exponential family distribution*

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta})\exp\left(-E(\boldsymbol{\beta},\boldsymbol{\theta})\right)$$

as in (7.1) *and sparse prior with super Gaussian distribution*

$$p(\mathbf{B}\boldsymbol{\beta}) = \max_{\boldsymbol{\gamma}\succ\mathbf{0}} \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|0,\boldsymbol{\Gamma})\varphi(\boldsymbol{\gamma})$$

as in (6.9)*, we go straightly into the marginal likelihood*

$$\begin{aligned}
&\int p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta})\mathcal{N}(\mathbf{B}\boldsymbol{\beta}|\mathbf{0},\boldsymbol{\Gamma})\prod_{i=1}^{\aleph}\varphi(\gamma_i)d\boldsymbol{\beta} \\
=&a(\boldsymbol{\theta})\int \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}\mathcal{N}(\mathbf{B}\boldsymbol{\beta}|\mathbf{0},\boldsymbol{\Gamma})\prod_{i=1}^{\aleph}\varphi(\gamma_i)d\boldsymbol{\beta} \\
=&a(\boldsymbol{\theta})\int \exp\left(\sum_{s=1}^{S}\eta_s(\boldsymbol{\theta})\cdot T_s(\boldsymbol{\beta})+B(\boldsymbol{\beta})\right)\mathcal{N}(\mathbf{B}\boldsymbol{\beta}|\mathbf{0},\boldsymbol{\Gamma})\prod_{i=1}^{\aleph}\varphi(\gamma_i)d\boldsymbol{\beta}.
\end{aligned} \tag{7.17}$$

Typically, this integral is intractable or has no analytical solution.

We first fix $\boldsymbol{\theta}$, the parameter of the family. For example, the mean and covariance can be fixed if the family is Gaussian. Performing a Taylor series expansion around some point $\boldsymbol{\beta}^*$, $E(\boldsymbol{\beta},\boldsymbol{\theta})$ can be approximated as

$$E(\boldsymbol{\beta},\boldsymbol{\theta}) \approx E(\boldsymbol{\beta}^*,\boldsymbol{\theta}) + (\boldsymbol{\beta}-\boldsymbol{\beta}^*)^\top\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) + \frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}^*)^\top\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})(\boldsymbol{\beta}-\boldsymbol{\beta}^*) \tag{7.18}$$

where $\mathbf{g}(\cdot)$ is the gradient and $\mathbf{H}(\cdot)$ is the Hessian of the energy function E

$$\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) \triangleq \nabla E(\boldsymbol{\beta},\boldsymbol{\theta})|_{\boldsymbol{\beta}^*} = -\left(\sum_{s=1}^{S}\eta_s(\boldsymbol{\theta})\cdot\nabla T_s(\boldsymbol{\beta})|_{\boldsymbol{\beta}^*} + \nabla B(\boldsymbol{\beta})|_{\boldsymbol{\beta}^*}\right), \tag{7.19a}$$

$$\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) \triangleq \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta})|_{\boldsymbol{\beta}^*} = -\left(\sum_{s=1}^{S}\eta_s(\boldsymbol{\theta})\cdot\nabla\nabla T_s(\boldsymbol{\beta})|_{\boldsymbol{\beta}^*} + \nabla\nabla B(\boldsymbol{\beta})|_{\boldsymbol{\beta}^*}\right). \tag{7.19b}$$

To derive the cost function in (7.16)*, we introduce the posterior mean and covariance*

$$\mathbf{m}_\beta = \boldsymbol{\Sigma}_\beta\cdot\left[\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) + \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta}^*\right], \tag{7.20a}$$

$$\boldsymbol{\Sigma}_\beta = \left[\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) + \mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\right]^{-1}. \tag{7.20b}$$

Then define the following quantities

$$b(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \triangleq \exp\left\{-\left(\frac{1}{2}\boldsymbol{\beta}^{*\top}\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta}^* - \boldsymbol{\beta}^{*\top}\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + E(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right)\right\}, \tag{7.21a}$$

$$c(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \triangleq \exp\left\{\frac{1}{2}\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right\}, \tag{7.21b}$$

$$d(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \triangleq \sqrt{|\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})|}, \tag{7.21c}$$

$$\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \triangleq \mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta}^*. \tag{7.21d}$$

Now the approximated likelihood $p(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\theta})$ is a exponential of quadratic, then Gaussian,

$$\begin{aligned}
&p(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\theta}) \\
=&a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta}, \boldsymbol{\theta})\} \\
\approx&a(\boldsymbol{\theta}) \cdot \exp\left\{-\left(\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})(\boldsymbol{\beta} - \boldsymbol{\beta}^*) + (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\top\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + E(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right)\right\} \\
=&a(\boldsymbol{\theta}) \cdot \exp\left\{-\left(\frac{1}{2}\boldsymbol{\beta}^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\beta}^\top\left[\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta}^*\right]\right)\right\} \\
&\cdot \exp\left\{-\left(\frac{1}{2}\boldsymbol{\beta}^{*\top}\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta}^* - \boldsymbol{\beta}^{*\top}\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + E(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right)\right\} \\
=&a(\boldsymbol{\theta}) \cdot b(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \cdot \exp\left\{-\left(\frac{1}{2}\boldsymbol{\beta}^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\beta}^\top\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right)\right\} \\
&\cdot \exp\left\{\frac{1}{2}\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) - \frac{1}{2}\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right\} \\
=&a(\boldsymbol{\theta}) \cdot b(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \cdot c(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \\
&\cdot \exp\left\{-\left(\frac{1}{2}\boldsymbol{\beta}^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\beta}^\top\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + \frac{1}{2}\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})^\top\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right)\right\} \\
=&(2\pi)^{M/2}a(\boldsymbol{\theta})b(\boldsymbol{\beta}^*, \boldsymbol{\theta})c(\boldsymbol{\beta}^*, \boldsymbol{\theta})d(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \cdot \mathcal{N}(\boldsymbol{\beta}|\hat{\boldsymbol{\beta}}^*, \mathbf{H}^{-1}(\boldsymbol{\beta}^*, \boldsymbol{\theta})) \\
\triangleq&A(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \cdot \mathcal{N}(\boldsymbol{\beta}|\hat{\boldsymbol{\beta}}^*, \mathbf{H}^{-1}(\boldsymbol{\beta}^*, \boldsymbol{\theta})),
\end{aligned} \tag{7.22}$$

where

$$\begin{aligned}
A(\boldsymbol{\beta}^*, \boldsymbol{\theta}) &= (2\pi)^{M/2}a(\boldsymbol{\theta})b(\boldsymbol{\beta}^*, \boldsymbol{\theta})c(\boldsymbol{\beta}^*, \boldsymbol{\theta})d(\boldsymbol{\beta}^*, \boldsymbol{\theta}), \\
\hat{\boldsymbol{\beta}}^* &= -\mathbf{H}^{-1}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) = \boldsymbol{\beta}^* - \mathbf{H}^{-1}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}).
\end{aligned}$$

We can write the approximate marginal likelihood as

$$\begin{aligned}
&A(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \int \mathcal{N}(\boldsymbol{\beta}|\hat{\boldsymbol{\beta}}^*, \mathbf{H}^{-1}(\boldsymbol{\beta}^*, \boldsymbol{\theta})) \cdot \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|\mathbf{0}, \boldsymbol{\Gamma}) \prod_{i=1}^{\aleph} \varphi(\gamma_i) d\boldsymbol{\beta} \\
=&a(\boldsymbol{\theta}) \cdot b(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \cdot \int \exp\left\{-\left(\frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\beta}^\top \hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\right)\right\} \mathcal{N}(\mathbf{B}\boldsymbol{\beta}|\mathbf{0}, \boldsymbol{\Gamma}) \prod_{i=1}^{\aleph} \varphi(\gamma_i) d\boldsymbol{\beta} \\
=&\frac{a(\boldsymbol{\theta}) \cdot b(\boldsymbol{\beta}^*, \boldsymbol{\theta})}{(2\pi)^{\aleph/2} |\boldsymbol{\Gamma}|^{1/2}} \int \exp\{-\hat{E}(\boldsymbol{\beta}, \boldsymbol{\theta})\} d\boldsymbol{\beta} \prod_{i=1}^{\aleph} \varphi(\gamma_i),
\end{aligned} \tag{7.23}$$

where

$$\hat{E}(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\beta}^\top \hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B}\boldsymbol{\beta}. \tag{7.24}$$

Equivalently, we get

$$\hat{E}(\boldsymbol{\beta}) = \frac{1}{2}(\boldsymbol{\beta} - \mathbf{m}_\beta)^\top \boldsymbol{\Sigma}_\beta^{-1} (\boldsymbol{\beta} - \mathbf{m}_\beta) + \hat{E}(\mathbf{y}), \tag{7.25}$$

where $\mathbf{m}_\beta$ *and* $\boldsymbol{\Sigma}_\beta$ *are given in* (7.20)*. From* (7.20a) *and* (7.20b)*, the data-dependent term can be re-expressed as*

$$\begin{aligned}
\hat{E}(\mathbf{y}) =& \frac{1}{2}\mathbf{m}_\beta^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\mathbf{m}_\beta + \mathbf{m}_\beta^\top \mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + \frac{1}{2}\mathbf{m}_\beta^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B}\mathbf{m}_\beta \\
=& \min_{\boldsymbol{\beta}} \left[\frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\beta}^\top \hat{\mathbf{g}}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B}\boldsymbol{\beta}\right] \\
=& \min_{\boldsymbol{\beta}} \left[\frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta}^*\right) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B}\boldsymbol{\beta}\right].
\end{aligned} \tag{7.26}$$

Using (7.25)*, we can evaluate the integral in* (7.23) *to obtain*

$$\int \exp\left\{-\hat{E}(\boldsymbol{\beta})\right\} d\boldsymbol{\beta} = \exp\left\{-\hat{E}(\mathbf{y})\right\} (2\pi)^{\aleph} |\boldsymbol{\Sigma}_\beta|^{1/2}. \tag{7.27}$$

Applying a $-2\log(\cdot)$ *transformation to* (7.23)*, we have*

$$\begin{aligned}
&-2\log\left[\frac{a(\boldsymbol{\theta})\cdot b(\boldsymbol{\beta}^*,\boldsymbol{\theta})}{(2\pi)^{\aleph/2}\,|\boldsymbol{\Gamma}|^{1/2}}\int\exp\{-\hat{E}(\boldsymbol{\beta})\}d\boldsymbol{\beta}\prod_{i=1}^{\aleph}\varphi(\gamma_i)\right]\\
\propto&-2\log a(\boldsymbol{\theta})\cdot b(\boldsymbol{\beta}^*,\boldsymbol{\theta})+\hat{E}(\mathbf{y})\\
&+\log|\boldsymbol{\Gamma}|+\log|\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})+\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}|-2\sum_{i=1}^{\aleph}\log\varphi(\gamma_i)\\
\propto&\boldsymbol{\beta}^\top\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta}+2\boldsymbol{\beta}^\top\hat{\mathbf{g}}(\boldsymbol{\beta}^*,\boldsymbol{\theta})+\boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}\\
&+\log|\boldsymbol{\Gamma}|+\log|\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})+\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}|\\
&-2\log a(\boldsymbol{\theta})\cdot b(\boldsymbol{\beta}^*,\boldsymbol{\theta})-2\sum_{i=1}^{\aleph}\log\varphi(\gamma_i).
\end{aligned}\tag{7.28}$$

Therefore we get the following cost function to be minimised in (7.16) *over* $\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}$

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta})=&\boldsymbol{\beta}^\top\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta}+2\boldsymbol{\beta}^\top\left[\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta})-\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta}^*\right]+\boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}\\
&+\log|\boldsymbol{\Gamma}|+\log|\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})+\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}|-2\log a(\boldsymbol{\theta})\cdot b(\boldsymbol{\beta}^*,\boldsymbol{\theta})-2\sum_{i=1}^{\aleph}\log\varphi(\gamma_i).
\end{aligned}$$

It can be easily found that the first line of $\mathcal{L}$ *is quadratic programming with* ℓ_2 *regularizer. The second line is all about the hyperparameter* $\boldsymbol{\gamma}$ *and the parameter of the exponential family* $\boldsymbol{\theta}$*.*

Once the estimate on $\boldsymbol{\beta}$ *and* $\boldsymbol{\gamma}$ *are obtained, the cost function is alternatively optimised over* $\boldsymbol{\theta}$*. The new estimated* $\boldsymbol{\beta}$ *can substitute* $\boldsymbol{\beta}^*$ *and repeat the estimation iteratively.*

■

Remark 18 *Throughout the thesis, we assume* $\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})$ *is invertible. Actually from the derivation, we cannot guarantee that Hessian matrix* $\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})$ *is positive semidefinite therefore* $\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})$ *maybe not invertible since its matrix property is determined by* $\boldsymbol{\beta}^*$*. Without exhaustive validation, we assume*

$$\mathbf{H}^{-1}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\triangleq\hat{\mathbf{H}}(\boldsymbol{\beta}^*,\boldsymbol{\theta}).\tag{7.29}$$

Furthermore, since $\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})$ *is invertible, we let* $\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})$ *decomposed as follows*

$$\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})=\mathbf{X}^\top(\boldsymbol{\beta}^*,\boldsymbol{\theta})\mathbf{X}(\boldsymbol{\beta}^*,\boldsymbol{\theta}).\tag{7.30}$$

If $\mathbf{X}$ *is identity matrix, then* $\boldsymbol{\Pi}(\boldsymbol{\beta}^*,\boldsymbol{\theta})=\hat{\mathbf{H}}(\boldsymbol{\beta}^*,\boldsymbol{\theta})$*.*

We note that in (7.19), $\boldsymbol{\beta}^*$ may not be the mode (i.e., the lowest energy state), which means the gradient term $\mathbf{g}$ may not be zero. Therefore the selection of $\boldsymbol{\beta}_1^*$ remains to be problematic. We give the following Corollary to address this issue, which is more general.

Corollary 2 *Suppose*

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta}, \boldsymbol{\theta}) + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta}, \tag{7.31}$$

we define a new cost function

$$\begin{aligned}\hat{\mathcal{L}}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}) &\triangleq E(\boldsymbol{\beta}, \boldsymbol{\theta}) + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta} \\ &+ \log |\boldsymbol{\Gamma}| + \log |\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B}| - 2 \log a(\boldsymbol{\theta}) \cdot b(\boldsymbol{\beta}^*, \boldsymbol{\theta}) - 2 \sum_{i=1}^{\aleph} \log \varphi(\gamma_i).\end{aligned} \tag{7.32}$$

Instead of minimising $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta})$, we can solve the following optimisation problem to get $\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}$

$$\min_{\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}} \hat{\mathcal{L}}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}).$$

Proof *Since the likelihood is*

$$p(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta}, \boldsymbol{\theta})\},$$

then

$$\min_{\boldsymbol{\beta}} E(\boldsymbol{\beta}, \boldsymbol{\theta}) + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta}$$

is exactly the regularised maximum likelihood estimation *with ℓ_2 type regulariser.*

We look at the first part of $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta})$ in (7.16)*, and define them as*

$$\mathcal{L}_0(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}) \triangleq \boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \boldsymbol{\beta} + 2 \boldsymbol{\beta}^\top \left[\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) \boldsymbol{\beta}^*\right] + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta},$$

then

$$\begin{aligned}&\min_{\boldsymbol{\beta}} \mathcal{L}_0(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}) \\ =&\min_{\boldsymbol{\beta}} \frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) (\boldsymbol{\beta} - \boldsymbol{\beta}^*) + (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\top \mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + E(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta} \\ \approx&\min_{\boldsymbol{\beta}} E(\boldsymbol{\beta}, \boldsymbol{\theta}) + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} \boldsymbol{\beta}\end{aligned} \tag{7.33}$$

where, given (7.19),

$$\begin{aligned}\mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) &= \nabla E(\boldsymbol{\beta},\boldsymbol{\theta})\\ \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) &= \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta}).\end{aligned} \tag{7.34}$$

Such quadratic approximation to $E(\boldsymbol{\beta},\boldsymbol{\theta}) + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}$ *is actually the same as the approximation procedure in Trust-Region Methods where a region is defined around the current iterate within which they trust the model to be an adequate representation of the objective function [219, pp.65].*

Similar to [219, eq.(4.3)], to obtain each step, we seek a solution of the subproblem at iteration k

$$\begin{aligned}&\min_{\boldsymbol{\beta}} E^k(\boldsymbol{\beta},\boldsymbol{\theta}) + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}\\ =&\min_{\boldsymbol{\beta}} \frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}^k)^\top\mathbf{H}(\boldsymbol{\beta}^k,\boldsymbol{\theta})(\boldsymbol{\beta}-\boldsymbol{\beta}^k) + (\boldsymbol{\beta}-\boldsymbol{\beta}^k)^\top\mathbf{g}(\boldsymbol{\beta}^k,\boldsymbol{\theta}) + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}\end{aligned} \tag{7.35}$$

Suppose

$$\boldsymbol{\beta}^* = \operatorname*{argmin}_{\boldsymbol{\beta}} E(\boldsymbol{\beta},\boldsymbol{\theta}) + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta},$$

then inject $\boldsymbol{\beta}^*$ *into* $\min_{\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta})$*, we can optimise* (7.32) *instead of* (7.16)*, i.e.,* $\min_{\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}} \hat{\mathcal{L}}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta})$*.*

■

7.4 Optimisation Algorithm

In this Section, we propose iterative optimisation algorithms to estimate β, $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ alternatively.

7.4.1 Optimisation for unknown parameter β and hyperparameter γ

Convex-concave procedure

We first target for the estimation of unknown parameter β and hyperparameter γ. We first initialise $\boldsymbol{\theta}$ to some reasonable value or have already had a good estimate of $\boldsymbol{\theta}$, denoted as $\boldsymbol{\theta}^*$. Then $\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)$ is a known symmetric matrix, assuming to be positive semidefinite. In the sequel, we show that the stated program can be formulated as a **convex-concave procedure** (CCCP) for β and γ.

Proposition 11 *The following programme*

$$\min_{\boldsymbol{\beta},\boldsymbol{\gamma}} \mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma})$$

with the cost function defined as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\beta},\boldsymbol{\gamma}) \triangleq \boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)\boldsymbol{\beta} + 2\boldsymbol{\beta}^\top\left[\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right] + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} \\ + \log|\boldsymbol{\Gamma}| + \log|\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)|\end{aligned} \tag{7.36}$$

can be formulated as a convex-concave procedure (CCCP), where $\boldsymbol{\beta}^*$ and $\boldsymbol{\theta}^*$ can be arbitrary real vector.

Proof Fact on convexity: *the function*

$$\begin{aligned}u\left(\boldsymbol{\beta},\boldsymbol{\Gamma}\right) =& \boldsymbol{\beta}^\top\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)\boldsymbol{\beta} + 2\boldsymbol{\beta}^\top\left[\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right] + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta} \\ \propto& (\boldsymbol{\beta}-\boldsymbol{\beta}^*)^\top\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)(\boldsymbol{\beta}-\boldsymbol{\beta}^*) + 2\boldsymbol{\beta}^\top\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*) + \boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}\end{aligned} \tag{7.37}$$

is convex jointly in $\boldsymbol{\beta}$, $\boldsymbol{\Gamma}$ due to the fact that $f(\mathbf{x},Y) = \mathbf{x}^\top\mathbf{Y}^{-1}\mathbf{x}$ is jointly convex in $\mathbf{x}$, $\mathbf{Y}$ (see, [28, p.76]). Hence u as a sum of convex functions is convex.

Fact on concavity: *the function*

$$v(\boldsymbol{\Gamma}) = \log|\boldsymbol{\Gamma}| + \log|\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)| \tag{7.38}$$

is jointly concave in $\boldsymbol{\Gamma}$, $\boldsymbol{\Pi}$. We exploit the properties of the determinant of a matrix

$$|A_{22}||A_{11} - A_{12}A_{22}^{-1}A_{21}| = \left|\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22}\end{pmatrix}\right| = |A_{11}||A_{22} - A_{21}A_{11}^{-1}A_{12}|.$$

Then we have

$$\begin{aligned}v(\boldsymbol{\Gamma}) &= \log|\boldsymbol{\Gamma}| + \log|\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)| \\ &= \log\left(|\boldsymbol{\Gamma}||\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)|\right) \\ &= \log\left|\begin{pmatrix}\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*) & \mathbf{B}^\top \\ \mathbf{B} & -\boldsymbol{\Gamma}\end{pmatrix}\right| \\ &= \log\left|\boldsymbol{\Gamma} + \mathbf{B}\mathbf{H}^{-1}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)\mathbf{B}^\top\right| + \log|\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)|\end{aligned} \tag{7.39}$$

which is a log*-determinant of an affine function of semidefinite matrices $\boldsymbol{\Pi}$, $\boldsymbol{\Gamma}$ and hence concave.*

Therefore, we can derive the iterative algorithm solving the CCCP. We have the following iterative convex optimisation program by calculating the gradient of concave part.

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}, \boldsymbol{\gamma}^k, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)), \tag{7.40}$$

$$\boldsymbol{\gamma}^{k+1} = \underset{\boldsymbol{\gamma}\succeq\mathbf{0}}{\operatorname{argmin}}\, u(\boldsymbol{\beta}^k, \boldsymbol{\gamma}, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)) + \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*))^\top \boldsymbol{\gamma}. \tag{7.41}$$

■

Remark 19 *Remark on the* log *of potential function in Proposition 11. Refer to the same contents in Remark 13.*

Derivation of iterative reweighted ℓ_1 algorithm

Using basic principles in convex analysis, we then obtain the following analytic form for the negative gradient of $v(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}$ is (using chain rule):

$$\begin{aligned}
\boldsymbol{\alpha}^k \triangleq & \nabla_{\boldsymbol{\gamma}} v\left(\boldsymbol{\gamma}, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\right)^\top |_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k} \\
= & \nabla_{\boldsymbol{\gamma}} \left(\log|\mathbf{B}^\top \boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)| + \log|\boldsymbol{\Gamma}|\right)^\top |_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k} \\
= & -\operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\} \circ \operatorname{diag}\left\{\mathbf{B}\left(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\right)^{-1}\mathbf{B}^\top\right\} \circ \operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\} \\
& + \operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\} \\
= & \begin{bmatrix} \alpha_1^k & \cdots & \alpha_N^k \end{bmatrix}
\end{aligned} \tag{7.42}$$

where $\circ$ denote the Hadamard product operation (entrywise multiplication) and diag denote the operation to get diagonal elements of matrix. Then equivalently, we have

$$\alpha_i^k = -\frac{\mathbf{B}_{i,:}(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*))^{-1}\mathbf{B}_{i,:}^\top}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}. \tag{7.43}$$

Therefore, the iterative procedures (7.40) and (7.41) for $\boldsymbol{\beta}^{k+1}$ and $\boldsymbol{\gamma}^{k+1}$ can be formulated as

$$\begin{aligned}
& \left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] \\
= & \underset{\boldsymbol{\gamma}\succeq\mathbf{0}, \boldsymbol{\beta}}{\operatorname{argmin}} (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)(\boldsymbol{\beta} - \boldsymbol{\beta}^*) + 2\boldsymbol{\beta}^\top \mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) + \sum_{i=1}^{\aleph} \left(\frac{\boldsymbol{\beta}^\top \mathbf{B}_{i,:}^\top \mathbf{B}_{i,:}\boldsymbol{\beta}}{\gamma_i} + \alpha_i^k \gamma_i\right) \\
= & \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta} + 2\boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right) + \sum_{i=1}^{\aleph} \left(\frac{\boldsymbol{\beta}^\top \mathbf{B}_{i,:}^\top \mathbf{B}_{i,:}\boldsymbol{\beta}}{\gamma_i} + \alpha_i^k \gamma_i\right).
\end{aligned} \tag{7.44}$$

Or in the compact form

$$\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta} + 2\boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right) + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1} \mathbf{B} + \sum_{i=1}^{\aleph} \alpha_i^k \gamma_i. \tag{7.45}$$

Since

$$\frac{\boldsymbol{\beta}^\top \mathbf{B}_{i,:}^\top \mathbf{B}_{i,:} \boldsymbol{\beta}}{\gamma_i} + \alpha_i^k \gamma_i \geq 2 \left| \sqrt{\alpha_i^k} \cdot \mathbf{B}_{i,:} \boldsymbol{\beta} \right|,$$

the optimal $\boldsymbol{\gamma}$ can be obtained as:

$$\gamma_i = \frac{|\mathbf{B}_{i,:}\boldsymbol{\beta}|}{\sqrt{\alpha_i^k}}, \forall i. \tag{7.46}$$

From (7.42), it is found that α_i^k is a function of γ_i^k. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}$ first to calculate $\boldsymbol{\gamma}^{k+1}$. If we define

$$w_i^k \triangleq \sqrt{\alpha_i^k} = \sqrt{-\frac{\mathbf{B}_{i,:}(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^k, \boldsymbol{\theta}^*))^{-1}\mathbf{B}_{i,:}^\top}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}}. \tag{7.47}$$

$\boldsymbol{\beta}^{k+1}$ can be obtained as follows

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta} + \boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right) + \sum_{i=1}^{\aleph} \|w_i^k \cdot \mathbf{B}_{i,:}\boldsymbol{\beta}\|_{\ell_1}. \tag{7.48}$$

We can then inject this into (7.46), which yields

$$\gamma_i^{k+1} = \frac{|\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}|}{w_i^k}, \forall i. \tag{7.49}$$

As we found in the expression for α_i in (7.43), α_i^k is function of γ^k, therefore γ^{k+1} is function of γ^k and $\boldsymbol{\beta}^{k+1}$. We notice that the update for $\boldsymbol{\beta}^{k+1}$ is to use the *lasso* or ℓ_1*-regularised* regression type optimisation. The pseudo code is summarised in Algorithm 7.

Algorithm 7 Reweighted ℓ_1 type algorithm for likelihood in exponential family of distributions

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta})|; \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown $\mathbf{w}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
5: **for** $k = 1, \ldots, k_{\max}$ **do**
6:

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \lambda \sum_{i=1}^{\aleph} \|w_i^k \cdot \mathbf{B}_{i,:}\boldsymbol{\beta}\|_{\ell_1}; \tag{7.50}$$

7: $\gamma_i^{k+1} = \left|\frac{\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}}{w_i^k}\right|$;
8: $\mathbf{C}^{k+1} = \left(\mathbf{B}^\top(\boldsymbol{\Gamma}^{k+1})^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^{k+1},\boldsymbol{\theta}^*)\right)^{-1}$;
9: ${\alpha_i}^{k+1} = -\frac{\mathbf{B}_{i,:}\mathbf{C}^{k+1}\mathbf{B}_{i,:}^\top}{(\gamma_i^{k+1})^2} + \frac{1}{\gamma_i^{k+1}}$;
10: ${w_i}^{k+1} = \sqrt{{\alpha_i}^{k+1}}$;
11: **if** a stopping criterion is satisfied **then**
12: Break.
13: **end if**
14: **end for**

Derivation of iterative reweighted ℓ_2 algorithm

In (7.44), instead of formulating a convex program for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ jointly, they are optimised respectively.

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} (\boldsymbol{\beta}-\boldsymbol{\beta}^*)^\top \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)(\boldsymbol{\beta}-\boldsymbol{\beta}^*) + 2\boldsymbol{\beta}^\top \mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*) + \sum_{i=1}^{\aleph} \left\| \frac{\mathbf{B}_{i,:}\boldsymbol{\beta}}{\sqrt{\gamma_i^k}} \right\|_{\ell_2}^2, \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)\boldsymbol{\beta} + \boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1}\mathbf{B}\boldsymbol{\beta} \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1}\mathbf{B}\boldsymbol{\beta}, \qquad (7.51) \\
\gamma_i^{k+1} &= \underset{\gamma_i \geq 0}{\operatorname{argmin}} \frac{(\boldsymbol{\beta}^{k+1})^\top \mathbf{B}_{i,:}^\top \mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}}{\gamma_i} + \alpha_i^k \gamma_i, \forall i. \qquad (7.52)
\end{aligned}$$

Once $\boldsymbol{\beta}^{k+1}$ is obtained, $\boldsymbol{\gamma}^{k+1}$ has a closed form solution to (7.52), i.e.,

$$\gamma_i^{k+1} = \sqrt{\frac{(\boldsymbol{\beta}^{k+1})^\top \mathbf{B}_{i,:}^\top \mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}}{\alpha_i^k}},$$

where α_i^k is the same as (7.43)

$$\alpha_i^k = -\frac{\mathbf{B}_{i,:}(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^{k+1},\boldsymbol{\theta}^*))^{-1}\mathbf{B}_{i,:}^\top}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}.$$

But unlike (7.47), w_i^k is defined as

$$w_i^k \triangleq \frac{1}{\sqrt{\gamma_i^k}}.$$

The pseudo code is summarised in Algorithm 8.

Algorithm 8 Reweighted ℓ_2 type algorithm for likelihood in exponential family of distributions

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta}); \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $w_i^1 = 1,\ \forall i$;
5: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7:

$$\begin{aligned} \boldsymbol{\beta}^{k+1} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \frac{\lambda}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1}\mathbf{B}\boldsymbol{\beta} \\ &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \frac{\lambda}{2}\sum_{i=1}^{\aleph} \|w_i^k \cdot \mathbf{B}_{i,:}\boldsymbol{\beta}\|_{\ell_2}^2; \end{aligned} \tag{7.53}$$

8: $\mathbf{C}^k = \left(\mathbf{B}^\top(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^{k+1},\boldsymbol{\theta}^*)\right)^{-1}$;
9: $\alpha_i^{k+1} = -\frac{\mathbf{B}_{i,:}\mathbf{C}^k\mathbf{B}_{i,:}^\top}{(\gamma_i^k)^2} + \frac{1}{\gamma_i^k}$;
10: $\gamma_i^{k+1} = \frac{\left|\mathbf{B}_{i,:}\boldsymbol{\beta}^{k+1}\right|}{\sqrt{\alpha_i^{k+1}}}$;
11: $w_i^{k+1} = \frac{1}{\sqrt{\gamma_i^{k+1}}}$;
12: **if** a stopping criterion is satisfied **then**
13: Break.
14: **end if**
15: **end for**

7.4.2 Optimisation for the parameter of the exponential family θ

Once $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are given or estimated to be $\boldsymbol{\beta}^*$ and $\boldsymbol{\gamma}^*$, $\boldsymbol{\theta}$ will be estimated, the cost function is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*,\boldsymbol{\theta}) = (\boldsymbol{\beta}^*)^\top \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})(\boldsymbol{\beta}^*) + 2(\boldsymbol{\beta}^*)^\top [\mathbf{g}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta})\boldsymbol{\beta}^*] + (\boldsymbol{\beta}^*)^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{*-1}\mathbf{B}\boldsymbol{\beta}^* \\ + \log|\boldsymbol{\Gamma}^*| + \log|\mathbf{H}(\boldsymbol{\beta}^*,\boldsymbol{\theta}) + \mathbf{B}^\top(\boldsymbol{\Gamma}^*)^{-1}\mathbf{B}| - 2\log a(\boldsymbol{\theta}) \cdot b(\boldsymbol{\beta}^*,\boldsymbol{\theta}) - 2\sum_{i=1}^{\aleph} \log\varphi(\gamma_i). \end{aligned}$$

By dropping the constant, the estimate $\boldsymbol{\theta}^*$ can be obtained by minimising the new cost function over $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*,\boldsymbol{\theta}) \tag{7.54}$$

where

$$\mathcal{L}(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{\theta}) = \boldsymbol{\beta}^{*\top} [2\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta}^*] + \log |\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}) + \mathbf{B}^\top (\boldsymbol{\Gamma}^*)^{-1}\mathbf{B}| - 2\log a(\boldsymbol{\theta}) \cdot b(\boldsymbol{\beta}^*, \boldsymbol{\theta}).$$

If the likelihood is Gaussian distributed, an easy recognised algorithm can be found in Section 6.5.3.

Inspired by the implementation of Generalised Method of Moment (GMM) [76, 77], a Nobel Prize in Economics winning work by Lars Hansen, we propose two ways to optimise for $\boldsymbol{\theta}$.

The first one is inspired by two-step feasible GMM, where after the find estimation of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, i.e., at the iteration $k = k_{\text{end}}$ of the CCCP (7.40) and (7.41)

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{\beta}} u(\boldsymbol{\beta}, \boldsymbol{\gamma}^k, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)), \\
\boldsymbol{\gamma}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{\gamma} \succeq \mathbf{0}} u(\boldsymbol{\beta}^k, \boldsymbol{\gamma}, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)) + \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*))^\top \boldsymbol{\gamma},
\end{aligned}$$

we have

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\beta}^{k_{\text{end}}}, \boldsymbol{\gamma}^{k_{\text{end}}}, \boldsymbol{\theta}). \tag{7.55}$$

The second one is inspired by iterated GMM, where at each iteration k of the CCCP (7.40) and (7.41), we perform

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{\beta}} u(\boldsymbol{\beta}, \boldsymbol{\gamma}^k, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)), \\
\boldsymbol{\gamma}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{\gamma} \succeq \mathbf{0}} u(\boldsymbol{\beta}^k, \boldsymbol{\gamma}, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)) + \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*))^\top \boldsymbol{\gamma}, \\
\boldsymbol{\theta}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}, \boldsymbol{\theta}).
\end{aligned} \tag{7.56}$$

7.4.3 Implementations

Iterative Solvers

One of the main step in Algorithm 7 and 8 are the penalised MAP optimisation, i.e., *reweighted ℓ_1 regression* (7.50) and *reweighted ℓ_2 regression* (7.53). This is more general yet able to build upon the *Alternating Direction Method of Multipliers* (ADMM) method [27]. When E is smooth, general iterative methods can be used to carry out the β-minimization step. Of particular interest are methods that only require the ability to compute $\nabla_\beta E$ for a given β. Such methods can scale to relatively

large problems. Examples include the standard gradient method, the (nonlinear) conjugate gradient method, and the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [29, 116]; see [219] for further details.

In this Thesis, we will derive corresponding ADMM algorithm for our method which we use mostly in our experiments. Empirical study shows that ADMM scheme is superior in the penalised MAP optimisation problem, both in convergence speed and solution consistency.

Short-Cut to Computation of the Updated Weight w

The other main step in Algorithm 7 and 8 is the computation to invert the covariance matrix $\mathbf{C} = \left(\mathbf{B}^\top \boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}, \boldsymbol{\theta}^*)\right)^{-1}$ which typically requires $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ memory. We next show that the computation of inverting a matrix can be a by-product of the iterative solvers, namely, Quasi-Newton method such as L-BFGS. Unfortunately, this only applies to *reweighted ℓ_2 regression* (7.53), but not *reweighted ℓ_1 regression* (7.50) yet.

In (7.53) of Algorithm 8, the following minimisation problem is typically solved by iterative solvers.

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta}, \boldsymbol{\theta}^*) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1}\mathbf{B}\boldsymbol{\beta}.$$

We denote

$$f^k(\boldsymbol{\beta}) \triangleq E(\boldsymbol{\beta}, \boldsymbol{\theta}^*) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1}\mathbf{B}\boldsymbol{\beta}. \tag{7.57}$$

And we assume $\min f^k(\boldsymbol{\beta})$ is solved by Newton method where the gradient and Hessian information of $f^k(\boldsymbol{\beta})$ is needed. Typically, the symbolic expression of gradient $\frac{\partial f^k(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ and Hessian $\frac{\partial^2 f^k(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^\top}$ will be cached at the beginning.

It should be noted that $\mathbf{H}(\boldsymbol{\beta})$ may not be unnecessary if Quasi-Newton method method is employed since the Hessian information can be evaluated/approximated by the gradient. We change the iteration index k (see the 4^{th} line of Algorithm 8) as epoch, while the iteration index is denoted as τ for solving (7.53) by using iterative

solvers like L-BFGS. Recall the expression of (7.51) in the reverse order, we can write

$$\begin{aligned}
&\boldsymbol{\beta}^{\text{epoch}+1} \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, f(\boldsymbol{\beta}) \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta}, \boldsymbol{\theta}^*) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^{\text{epoch}})^{-1}\mathbf{B}\boldsymbol{\beta} \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta} + \boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^{\text{epoch}})^{-1}\mathbf{B}\boldsymbol{\beta} \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \frac{1}{2}\boldsymbol{\beta}^\top \left(\mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) + \mathbf{B}^\top (\boldsymbol{\Gamma}^{\text{epoch}})^{-1}\mathbf{B}\right)\boldsymbol{\beta} + \boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right).
\end{aligned} \tag{7.58}$$

Next, we consider the maximum iteration of τ as $\{\tau\}_{\max}$ which can be fixed a priori or reach to some number when a stopping criterion is satisfied. Let

$$\boldsymbol{\beta}^{\text{epoch}+1} = \boldsymbol{\beta}^{\text{epoch}+1,1} = \boldsymbol{\beta}^{\text{epoch},\{\tau\}_{\max}}. \tag{7.59}$$

Then we have

$$\begin{aligned}
&\boldsymbol{\beta}^{\text{epoch},\tau+1} \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, f^{\text{epoch}}(\boldsymbol{\beta}) \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta}, \boldsymbol{\theta}^*) + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^{\text{epoch}})^{-1}\mathbf{B}\boldsymbol{\beta} \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^{\text{epoch},\tau}, \boldsymbol{\theta}^*)\boldsymbol{\beta} + \boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^{\text{epoch},\tau}, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^{\text{epoch},\tau}, \boldsymbol{\theta}^*)\boldsymbol{\beta}^{\text{epoch},\tau}\right) \\
&\qquad + \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{B}^\top (\boldsymbol{\Gamma}^{\text{epoch}})^{-1}\mathbf{B}\boldsymbol{\beta} \\
&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \frac{1}{2}\boldsymbol{\beta}^\top \left(\mathbf{H}(\boldsymbol{\beta}^{\text{epoch},\tau}, \boldsymbol{\theta}^*) + \mathbf{B}^\top (\boldsymbol{\Gamma}^{\text{epoch}})^{-1}\mathbf{B}\right)\boldsymbol{\beta} \\
&\qquad + \boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^{\text{epoch},\tau}, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^{\text{epoch},\tau}, \boldsymbol{\theta}^*)\boldsymbol{\beta}^{\text{epoch},\tau}\right).
\end{aligned} \tag{7.60}$$

As in Newton's method, one uses a second order approximation to find the minimum of a function $f(\boldsymbol{\beta})$. The Taylor series of $f(\boldsymbol{\beta})$ around an iterate is:

$$f(\boldsymbol{\beta}^\tau + \Delta\boldsymbol{\beta}) \approx f(\boldsymbol{\beta}^\tau) + \nabla f(\boldsymbol{\beta}^\tau)^T \Delta\boldsymbol{\beta} + \frac{1}{2}\Delta\boldsymbol{\beta}^T \tilde{\mathbf{H}}\, \Delta\boldsymbol{\beta},$$

where (∇f) is the gradient and $\tilde{\mathbf{H}}$ an approximation to the Hessian matrix. It also should be noted that $\tilde{\mathbf{H}}$ is for f rather than $\mathbf{H}$ for E . The gradient of this

approximation (with respect to $\Delta\boldsymbol{\beta}$) is

$$\nabla f(\boldsymbol{\beta}^\tau + \Delta\boldsymbol{\beta}) \approx \nabla f(\boldsymbol{\beta}^\tau) + \tilde{\mathbf{H}}\,\Delta\boldsymbol{\beta}$$

and setting this gradient to zero (which is the objective of optimisation) provides the Newton step:

$$\Delta\boldsymbol{\beta} = -\tilde{\mathbf{H}}^{-1}\nabla f(\boldsymbol{\beta}^\tau),$$

The Hessian approximation $\tilde{\mathbf{H}}$ is chosen to satisfy

$$\nabla f(\boldsymbol{\beta}^\tau + \Delta\boldsymbol{\beta}) = \nabla f(\boldsymbol{\beta}^\tau) + \tilde{\mathbf{H}}\,\Delta\boldsymbol{\beta},$$

which is called the "secant equation" (the Taylor series of the gradient itself). In more than one dimension $\tilde{\mathbf{H}}$ is underdetermined. In one dimension, solving for $\tilde{\mathbf{H}}$ and applying the Newton's step with the updated value is equivalent to the secant method. The various quasi-Newton methods differ in their choice of the solution to the secant equation (in one dimension, all the variants are equivalent). Most methods (but with exceptions, such as Broyden's method) seek a symmetric solution ($\tilde{\mathbf{H}}^\top = \tilde{\mathbf{H}}$); furthermore, the variants listed below can be motivated by finding an update $\tilde{\mathbf{H}}^{\tau+1}$ that is as close as possible to $\tilde{\mathbf{H}}^\tau$ in some norm; that is, $\tilde{\mathbf{H}}^{\tau+1} = \operatorname{argmin}_H \|\tilde{\mathbf{H}} - \tilde{\mathbf{H}}^\tau\|_V$ where V is some positive definite matrix that defines the norm. An approximate initial value of $\tilde{\mathbf{H}}^0 = \mathbf{I} * \beta$ is often sufficient to achieve rapid convergence. Note that $\tilde{\mathbf{H}}^0$ should be positive definite. The unknown $\boldsymbol{\beta}^\tau$ is updated applying the Newton's step calculated using the current approximate Hessian matrix $\tilde{\mathbf{H}}^\tau$

- $\Delta\boldsymbol{\beta}^\tau = -\alpha^\tau(\tilde{\mathbf{H}}^\tau)^{-1}\nabla f(\boldsymbol{\beta}^\tau)$, with α chosen to satisfy the Wolfe conditions;

- $\boldsymbol{\beta}^{\tau+1} = \boldsymbol{\beta}^\tau + \Delta\boldsymbol{\beta}^\tau$;

- The gradient computed at the new point $\nabla f(\boldsymbol{\beta}^{\tau+1})$, and

$$y^\tau = \nabla f(\boldsymbol{\beta}^{\tau+1}) - \nabla f(\boldsymbol{\beta}^\tau),$$

is used to update the approximate Hessian $\tilde{\mathbf{H}}^{\tau+1}$, or directly its inverse $(\tilde{\mathbf{H}}^{\tau+1})^{-1}$ using the Sherman-Morrison formula. For example, using L-BFGS,

$$\tilde{\mathbf{H}}^{\tau+1} = \tilde{\mathbf{H}}^{\tau} + \frac{y^{\tau}(y^{\tau})^{\top}}{(y^{\tau})^{\top}\Delta\boldsymbol{\beta}^{\tau}} - \frac{\tilde{\mathbf{H}}^{\tau}\Delta\boldsymbol{\beta}^{\tau}(\tilde{\mathbf{H}}^{\tau}\Delta\boldsymbol{\beta}^{\tau})^{\top}}{\Delta(\boldsymbol{\beta}^{\tau})^{\top}\tilde{\mathbf{H}}^{\tau}\,\Delta\boldsymbol{\beta}^{\tau}}, \tag{7.61a}$$

$$(\tilde{\mathbf{H}}^{\tau+1})^{-1} = \left(\mathbf{I} - \frac{\Delta\boldsymbol{\beta}^{\tau}(y^{\tau})^{\top}}{(y^{\tau})^{\top}\Delta\boldsymbol{\beta}^{\tau}}\right)\tilde{\mathbf{H}}^{\tau}\left(\mathbf{I} - \frac{y^{\tau}\Delta(\boldsymbol{\beta}^{\tau})^{\top}}{(y^{\tau})^{\top}\Delta\boldsymbol{\beta}^{\tau}}\right) + \frac{\Delta\boldsymbol{\beta}^{\tau}\Delta(\boldsymbol{\beta}^{\tau})^{\top}}{(y^{\tau})^{\top}\,\Delta\boldsymbol{\beta}^{\tau}}; \tag{7.61b}$$

- A key property of the BFGS and DFP updates is that if $\tilde{\mathbf{H}}^{\tau}$ is positive definite and α^{τ} is chosen to satisfy the Wolfe conditions then $\tilde{\mathbf{H}}^{\tau+1}$ is also positive definite.

In Algorithm 9, we modify Algorithm 8 with Quasi-Newton method solving the minimisation problem (7.53). It should be noted that, if L-BFGS algorithm is employed, the Hessian matrix needs not to be cached a priori but approximated by gradient, i.e., approximating $\tilde{\mathbf{H}}^{\tau+1}$ or directly $(\tilde{\mathbf{H}}^{\tau+1})^{-1}$ by $\nabla f(\boldsymbol{\beta}^{\text{epoch},\tau+1})$ and $\nabla f(\boldsymbol{\beta}^{\text{epoch},\tau})$.

Semidefinite Programming for Arbitrary $\boldsymbol{\Gamma}$

To this end, the covariance matrix for the super Gaussian distribution $\boldsymbol{\Gamma}$, i.e., the hyperparameter, is diagonal. The underlying implication of "diagonal" is the independence of $\boldsymbol{\gamma}_i$ therefore $\mathbf{B}_{i,:}\boldsymbol{\beta}$ are independent with each other, $\forall i$. In other words, all the parameters of a model are irrelevant. Apparently, it should be careful to make such assumption. For example, in financial models, some parameters are co-moving together; in gene networks, some parameters are changing in response to certain common external perturbation. In this Section, we will show that the neat and simple form with ℓ_1- or ℓ_2- regularisation disappears when the covariance matrix $\boldsymbol{\Gamma}$ is *not* diagonal, yet semidefinite positive, as an illustration, $\boldsymbol{\Gamma}$ may look like $\begin{bmatrix} \gamma_1 & \gamma_1\gamma_2 \\ \gamma_1\gamma_2 & \gamma_2 \end{bmatrix}$.

We first vectorise all the parameters γ_i into a single vector $\boldsymbol{\gamma}$. From (7.30) in Remark 18, we know

$$\mathbf{H}(\boldsymbol{\beta}^{*}, \boldsymbol{\theta}) = \mathbf{X}^{\top}(\boldsymbol{\beta}^{*}, \boldsymbol{\theta})\mathbf{X}(\boldsymbol{\beta}^{*}, \boldsymbol{\theta}).$$

Algorithm 9 Algorithm 8 with Quasi-Newton Update

Cache symbolically the likelihood, loss function and its gradient over $\boldsymbol{\beta}$

$$\begin{aligned}\text{Likelihood: } p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) &= a(\boldsymbol{\theta})\cdot\exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}\\ \text{Loss function: } f(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) &= E(\boldsymbol{\beta},\boldsymbol{\theta})+\lambda\boldsymbol{\beta}^\top\mathbf{B}^\top\boldsymbol{\Gamma}^{-1}\mathbf{B}\boldsymbol{\beta}\\ \text{Gradient function: } \tilde{\mathbf{g}}(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta})\end{aligned}$$

where we fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;

Initialise the unknown hyperparameter $\boldsymbol{\gamma}$, i.e., $\boldsymbol{\gamma}^1$, as an arbitrary positive vector such as unit vector;

Initialise the unknown parameter $\boldsymbol{\beta}$, i.e., $\boldsymbol{\beta}^{1,1}$, as an arbitrary real vector such as zero vector;

Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;

Initialise the Hessian matrix $\tilde{\mathbf{H}}^{1,1}$ as an arbitrary positive definite matrix such as identity or be calculated explicitly with $\nabla\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\theta})$ given $\boldsymbol{\beta}^{1,1}$ and $\boldsymbol{\theta}^*$;

for $\text{epoch} = 1,\ldots,\text{epoch}_{\max}$ **do**

  for $\tau = 1,\ldots,\tau_{\max}$ **do**

    Choose $\alpha^{\text{epoch},\tau}$ for the next step to via line search under Wolfe conditions;

    $\Delta\boldsymbol{\beta}^{\text{epoch},\tau} = -\alpha^{\text{epoch},\tau}(\tilde{\mathbf{H}}^{\text{epoch},\tau})^{-1}\tilde{\mathbf{g}}(\boldsymbol{\beta}^{\text{epoch},\tau},\boldsymbol{\gamma}^{\text{epoch}},\boldsymbol{\theta}^*)$;

    $\boldsymbol{\beta}^{\text{epoch},\tau+1} = \boldsymbol{\beta}^{\text{epoch},\tau} + \Delta\boldsymbol{\beta}^{\text{epoch},\tau}$;

    Compute the gradient at the new point $\tilde{\mathbf{g}}(\boldsymbol{\beta}^{\text{epoch},\tau+1},\boldsymbol{\gamma}^{\text{epoch}},\boldsymbol{\theta}^*)$;

    $y^{\text{epoch},\tau} = \tilde{\mathbf{g}}(\boldsymbol{\beta}^{\text{epoch},\tau+1},\boldsymbol{\gamma}^{\text{epoch}},\boldsymbol{\theta}^*) - \tilde{\mathbf{g}}(\boldsymbol{\beta}^{\text{epoch},\tau},\boldsymbol{\gamma}^{\text{epoch}},\boldsymbol{\theta}^*)$;

    Approximate $\tilde{\mathbf{H}}^{\text{epoch},\tau+1}$ using $y^{\text{epoch},\tau}, \Delta\boldsymbol{\beta}^{\text{epoch},\tau}, \tilde{\mathbf{H}}^{\text{epoch},\tau}$ like (7.61a);

    Approximate $(\tilde{\mathbf{H}}^{\text{epoch},\tau+1})^{-1}$ using $y^{\text{epoch},\tau}, \Delta\boldsymbol{\beta}^{\text{epoch},\tau}, \tilde{\mathbf{H}}^{\text{epoch},\tau}$ like (7.61b);

  end for

  $\mathbf{C}^{\text{epoch}} = \left(\tilde{\mathbf{H}}(\boldsymbol{\beta}^{\text{epoch},\tau_{\max}},\boldsymbol{\theta}^*)\right)^{-1}$;

  $\alpha_i^{\text{epoch}+1} = -\frac{\mathbf{B}_{i,:}\mathbf{C}^{\text{epoch}}\mathbf{B}_{i,:}^\top}{(\gamma_i^{\text{epoch}})^2} + \frac{1}{\gamma_i^{\text{epoch}}}$;

  $\boldsymbol{\beta}^{\text{epoch}+1} = \boldsymbol{\beta}^{\text{epoch},\tau_{\max}}$;

  $\gamma_i^{\text{epoch}+1} = \frac{\left|\mathbf{B}_{i,:}\boldsymbol{\beta}^{\text{epoch}+1}\right|}{\sqrt{\alpha_i^{\text{epoch}+1}}}$, $\boldsymbol{\Gamma}^{k_{\text{epoch}+1}} = \text{diag}(\boldsymbol{\gamma}^{\text{epoch}+1})$;

  if a stopping criterion is satisfied **then**

    Break;

  end if

end for

We can rewrite the iteration (7.45) in the following compact form

$$\begin{aligned}\left[\boldsymbol{\beta}^{k+1}, \boldsymbol{\gamma}^{k+1}\right] &= \underset{\boldsymbol{\beta},\boldsymbol{\gamma}}{\operatorname{argmin}} \boldsymbol{\beta}^\top \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta} + 2\boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right) \\ &\qquad + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1}\mathbf{B} - \nabla_\gamma v(\boldsymbol{\gamma}^k, \boldsymbol{\theta}^*)^\top \boldsymbol{\gamma} \\ &= \underset{\boldsymbol{\beta},\boldsymbol{\gamma}}{\operatorname{argmin}} \boldsymbol{\beta}^\top \mathbf{X}^\top(\boldsymbol{\beta}^*, \boldsymbol{\theta})\mathbf{X}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} + 2\boldsymbol{\beta}^\top \left(\mathbf{g}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*) - \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\boldsymbol{\beta}^*\right) \\ &\qquad + \boldsymbol{\beta}^\top \mathbf{B}^\top \boldsymbol{\Gamma}^{-1}\mathbf{B} - \nabla_\gamma v(\boldsymbol{\gamma}^k, \boldsymbol{\theta}^*)^\top \boldsymbol{\gamma}.\end{aligned}$$

This is equivalent to the following SDP by using the standard procedure in [28]

$$\begin{aligned} \min_{\mathbf{z},\boldsymbol{\beta},\boldsymbol{\gamma}} \quad & \mathbf{z} - \nabla_\gamma v(\boldsymbol{\gamma}^k, \boldsymbol{\theta}^*)^\top \boldsymbol{\gamma} \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{z} & (\mathbf{X}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta})^\top & (\mathbf{B}\boldsymbol{\beta})^\top \\ \mathbf{X}(\boldsymbol{\beta}^*, \boldsymbol{\theta})\boldsymbol{\beta} & \mathbf{I} & \mathbf{0} \\ \mathbf{B}\boldsymbol{\beta} & \mathbf{0} & \boldsymbol{\Gamma} \end{bmatrix} \succeq \mathbf{0} \\ & \boldsymbol{\gamma} \succeq \mathbf{0}. \end{aligned}$$

The cost of solving this SDP is at least $(\aleph + M)^3$, where $\aleph$ is the number rows and M is typically the number of examples. Therefore, solving this SDP is too costly for all but problems with a small number of variables. This means that the number of samples, the dimension of the system, and potential number of the unknown parameters etc., can not be too large simultaneously. In this SDP formulation, $\boldsymbol{\Gamma}$ is closely related to the sparse multiple kernel presented in [35]. Certain choice of kernels may introduce some good properties or help reduce algorithmic complexity.

7.5 Optimisation Algorithm with Structural Sparsity

7.5.1 Algorithm for Group Spare Prior in Section 7.2.2

Consider the group spare prior defined in Section 7.2.2. For $o = 1, \ldots, O$, we have

$$\sum_{i=1}^{\aleph_o} \left(\frac{\boldsymbol{\beta}_o^\top \{\mathbf{B}_o\}_{i,:}^\top \{\mathbf{B}_o\}_{i,:} \boldsymbol{\beta}_o}{\gamma_o} + \alpha_{oi}^k \gamma_o \right) \geq 2 \left\| \sqrt{\sum_{i=1}^{\aleph_o} \alpha_{oi}^k} \cdot \mathbf{B}_o \boldsymbol{\beta}_o \right\|_{\ell_2}, \tag{7.62}$$

the optimal $\boldsymbol{\gamma}$ can be obtained as:

$$\gamma_o^{k+1} = \frac{\|\mathbf{B}_o \boldsymbol{\beta}_o\|_{\ell_2}}{\sqrt{\sum_{i=1}^{\aleph_o} \alpha_{oi}^k}}, \forall i. \tag{7.63}$$

It should be noted that $\boldsymbol{\alpha}^k$ is still the same as in (7.42), but structured as

$$\boldsymbol{\alpha}^k = \Big[\underbrace{\alpha_{11}^k, \ldots, \alpha_{1\aleph_1}^k}_{\aleph_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\alpha_{O1}^k, \ldots, \alpha_{O\aleph_O}^k}_{\aleph_O \textbf{ elements}} \Big]. \tag{7.64}$$

It is found that α_o^k is a function of γ_o^k. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}$ first to calculate $\boldsymbol{\gamma}^{k+1}$. If we define

$$w_o^k \triangleq \sqrt{\sum_{i=1}^{\aleph_o} \alpha_{oi}^k}, . \tag{7.65}$$

The pseudo code is summarised in Algorithm 10.

Algorithm 10 Reweighted *Generalised Group* ℓ_1 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta}, \boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta}, \boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta}, \boldsymbol{\theta})|; \mathbf{H}(\boldsymbol{\beta}, \boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta}, \boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $w_i^1 = 1, \ \forall i$;
5: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Group Lasso* type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \, E(\boldsymbol{\beta}, \boldsymbol{\theta}^*) + \lambda \sum_{o=1}^{O} \left\| w_o^k \cdot \mathbf{B}_o \boldsymbol{\beta}_o \right\|_{\ell_2}; \tag{7.66}$$

8: $\gamma_o^{k+1} = \frac{\left\|\mathbf{B}_o \beta_o^{k+1}\right\|_{\ell_2}}{w_o^k}$, and $\boldsymbol{\gamma}^{k+1} = \Big[\underbrace{\gamma_1^{k+1}, \ldots, \gamma_1^{k+1}}_{\aleph_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\gamma_O^{k+1}, \ldots, \gamma_O^{k+1}}_{\aleph_O \textbf{ elements}} \Big]$;
9:
10: $\mathbf{C}^{k+1} = \left(\mathbf{B}^\top (\boldsymbol{\Gamma}^{k+1})^{-1} \mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*) \right)^{-1}$;
11: $\alpha_{oi}^{k+1} = -\frac{\{\mathbf{B}_o\}_{i,:} \mathbf{C}^{k+1} \{\mathbf{B}_o\}_{i,:}^\top}{(\gamma_o^{k+1})^2} + \frac{1}{\gamma_o^{k+1}}, \forall i$;
12: $w_o^{k+1} = \sqrt{\sum_{i=1}^{\aleph_o} \alpha_{oi}^{k+1}}, \forall o$;
13: **if** a stopping criterion is satisfied **then**
14: Break;
15: **end if**
16: **end for**

Similarly, we can derive the reweighted *generalised group* ℓ_2 type algorithm. The pseudo code is summarised in Algorithm 11.

Algorithm 11 Reweighted *Generalised Group* ℓ_2 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta})|; \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $w_i^1 = 1, \ \forall i$;
5: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Group* ℓ_2 type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \lambda \sum_{o=1}^{O} \left\| w_o^k \cdot \mathbf{B}_o \boldsymbol{\beta}_o \right\|_{\ell_2}^2; \tag{7.67}$$

8: $\mathbf{C}^k = \left(\mathbf{B}^\top (\boldsymbol{\Gamma}^k)^{-1} \mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*)\right)^{-1}$;
9: $\alpha_{oi}^{k+1} = -\frac{\{\mathbf{B}_o\}_{i,:} \mathbf{C}^k \{\mathbf{B}_o\}_{i,:}^\top}{(\gamma_o^k)^2} + \frac{1}{\gamma_o^k}$, ;
10: $\gamma_o^{k+1} = \frac{\|\mathbf{B}_o \boldsymbol{\beta}_o\|_{\ell_2}}{\sqrt{\sum_{i=1}^{\aleph_o} \alpha_{oi}^{k+1}}}$, and $\boldsymbol{\gamma}^{k+1} = \Big[\underbrace{\gamma_1^{k+1}, \ldots, \gamma_1^{k+1}}_{\aleph_1 \text{ elements}} \Big| \cdots \Big| \underbrace{\gamma_O^{k+1}, \ldots, \gamma_O^{k+1}}_{\aleph_O \text{ elements}} \Big]$;
11: $w_o^{k+1} = \frac{1}{\sqrt{\gamma_o^{k+1}}}$;
12: **if** a stopping criterion is satisfied **then**
13: Break;
14: **end if**
15: **end for**

For a special case, where all the $\mathbf{B}$ matrix are identity matrix. This is similar to the Group Lasso

$$\begin{aligned}\mathbf{B} &= \begin{bmatrix} \mathbf{I}_1 & & \\ & \ddots & \\ & & \mathbf{I}_O \end{bmatrix} \in \mathbb{R}^{\aleph \times \aleph}. \\ \mathbf{I}_o &\in \mathbb{R}^{\aleph_o \times \aleph_o}, \aleph = \sum_{o=1}^{O} \aleph_o, o = 1, \ldots, O.\end{aligned} \tag{7.68}$$

Then for $o = 1, \ldots, O$, we have

$$\sum_{i=1}^{\aleph_o} \left(\frac{\boldsymbol{\beta}_o^\top \boldsymbol{\beta}_o}{\gamma_o} + \alpha_o^k \gamma_o \right) \geq 2 \left\| \sqrt{\aleph_o \alpha_o^k} \cdot \boldsymbol{\beta}_o \right\|_{\ell_2}. \tag{7.69}$$

the optimal $\boldsymbol{\gamma}$ can be obtained as:

$$\gamma_o^{k+1} = \frac{\|\mathbf{B}_o \boldsymbol{\beta}_o\|_{\ell_2}}{\sqrt{\aleph_o \alpha_o^k}}, \forall o. \tag{7.70}$$

It should be noted that $\boldsymbol{\alpha}^k$ is still the same as in (7.42), but structured as

$$\boldsymbol{\alpha}^k = \Big[\underbrace{\alpha_1^k, \ldots, \alpha_1^k}_{\aleph_1 \text{ elements}} \Big| \ \ldots \ \Big| \underbrace{\alpha_O^k, \ldots, \alpha_O^k}_{\aleph_O \text{ elements}} \Big]. \tag{7.71}$$

It is found that α_o^k is a function of γ_o^k. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}$ first to calculate $\boldsymbol{\gamma}^{k+1}$. If we define

$$w_o^k \triangleq \sqrt{\aleph_o \alpha_o^k}, \tag{7.72}$$

The pseudo code is summarised in Algorithm 12.

Similarly, we can derive the reweighted *Group* ℓ_2 type algorithm. The pseudo code is summarised in Algorithm 13.

7.5.2 Algorithm for Fused Sparse Prior in Section 7.2.3

We first look at the hyperparameter $\tilde{\boldsymbol{\Gamma}}$. Since For $\tilde{o} = 1, \ldots, \tilde{O}$, since

$$\sum_{i=1}^{\tilde{\aleph}_{\tilde{o}}} \left(\frac{\tilde{\boldsymbol{\beta}}_{\tilde{o}}^\top \{\tilde{\mathbf{B}}_{\tilde{o}}\}_{i,:}^\top \{\tilde{\mathbf{B}}_{\tilde{o}}\}_{i,:} \boldsymbol{\beta}_{\tilde{o}}}{\tilde{\gamma}_{\tilde{o}}} + \tilde{\alpha}_{\tilde{o}}^k \tilde{\gamma}_{\tilde{o}} \right) \geq 2 \left\| \sqrt{\tilde{\aleph}_{\tilde{o}} \tilde{\alpha}_{\tilde{o}}^k} \cdot \tilde{\mathbf{B}}_{\tilde{o}} \boldsymbol{\beta}_{\tilde{o}} \right\|_{\ell_2}, \tag{7.75}$$

Algorithm 12 Reweighted *Group* ℓ_1 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta}); \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $w_i^1 = 1,\ \forall i$;
5: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Group Lasso* type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \lambda \sum_{o=1}^{O} \left\| w_o^k \cdot \boldsymbol{\beta}_o \right\|_{\ell_2}; \tag{7.73}$$

8: $\gamma_o^{k+1} = \frac{\left\|\beta_o^{k+1}\right\|_{\ell_2}}{w_o^k}$, and $\boldsymbol{\gamma}^{k+1} = \Big[\underbrace{\gamma_1^{k+1}, \ldots, \gamma_1^{k+1}}_{\aleph_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\gamma_O^{k+1}, \ldots, \gamma_O^{k+1}}_{\aleph_O \textbf{ elements}} \Big]$;
9: $\mathbf{C}^{k+1} = \left((\boldsymbol{\Gamma}^{k+1})^{-1} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*) \right)^{-1}$;
10: $\alpha_o^{k+1} = -\frac{\mathbf{C}^{k+1}}{(\gamma_o^{k+1})^2} + \frac{1}{\gamma_o^{k+1}}$;
11: $w_o^{k+1} = \sqrt{\aleph_o \alpha_o^{k+1}}$;
12: **if** a stopping criterion is satisfied **then**
13: Break;
14: **end if**
15: **end for**

Algorithm 13 Reweighted *Group* ℓ_2 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta}); \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $w_i^1 = 1,\ \forall i$;
5: Fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Group* ℓ_2 type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \lambda \sum_{o=1}^{O} \left\| w_o^k \cdot \boldsymbol{\beta}_o \right\|_{\ell_2}^2; \tag{7.74}$$

8: $\mathbf{C}^k = \left((\boldsymbol{\Gamma}^k)^{-1} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*) \right)^{-1}$;
9: $\alpha_o^{k+1} = -\frac{\mathbf{C}^k}{(\gamma_o^k)^2} + \frac{1}{\gamma_o^k}$;
10: $\gamma_o^{k+1} = \frac{\left\| \boldsymbol{\beta}_o^{k+1} \right\|_{\ell_2}}{\sqrt{\aleph_o \alpha_o^{k+1}}}$, and $\boldsymbol{\gamma}^{k+1} = \Big[\underbrace{\gamma_1^{k+1}, \ldots, \gamma_1^{k+1}}_{\aleph_1 \text{ elements}} \Big| \ldots \Big| \underbrace{\gamma_O^{k+1}, \ldots, \gamma_O^{k+1}}_{\aleph_O \text{ elements}} \Big]$;
11: $w_o^{k+1} = \frac{1}{\sqrt{\gamma_o^{k+1}}}$;
12: **if** a stopping criterion is satisfied **then**
13: Break;
14: **end if**
15: **end for**

the optimal $\boldsymbol{\gamma}$ can be obtained as:

$$\tilde{\gamma}_{\tilde{o}}^{k+1} = \frac{\left\|\tilde{\mathbf{B}}_{\tilde{o}}\tilde{\boldsymbol{\beta}}_{\tilde{o}}\right\|_{\ell_2}}{\sqrt{\tilde{\aleph}_{\tilde{o}}\tilde{\alpha}_{\tilde{o}}^k}}, \forall i. \tag{7.76}$$

It should be noted that $\tilde{\boldsymbol{\alpha}}^k$ is still the same as in (7.42), but structured as

$$\tilde{\boldsymbol{\alpha}}^k = \Big[\underbrace{\tilde{\alpha}_{11}^k, \ldots, \tilde{\alpha}_{1\tilde{\aleph}_1}^k}_{\tilde{\aleph}_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\alpha}_{\tilde{O}1}^k, \ldots, \tilde{\alpha}_{\tilde{O}\tilde{\aleph}_{\tilde{O}}}^k}_{\tilde{\aleph}_{\tilde{O}} \textbf{ elements}} \Big]. \tag{7.77}$$

Recall the expression for $\boldsymbol{\alpha}^k$ in (7.42)

$$\begin{aligned}
&\nabla_{\gamma} v\left(\boldsymbol{\gamma}, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\right)^{\top}|_{\gamma=\gamma^k} \\
=&\nabla_{\gamma}\left(\log|\mathbf{B}^{\top}\boldsymbol{\Gamma}^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)| + \log|\boldsymbol{\Gamma}|\right)^{\top}|_{\gamma=\gamma^k} \\
=&-\operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\} \circ \operatorname{diag}\left\{\mathbf{B}\left(\mathbf{B}^{\top}(\boldsymbol{\Gamma}^k)^{-1}\mathbf{B} + \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*)\right)^{-1}\mathbf{B}^{\top}\right\} \circ \operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\} \\
&+\operatorname{diag}\left\{(\boldsymbol{\Gamma}^k)^{-1}\right\}.
\end{aligned}$$

Slightly different from the above expression, $\tilde{\boldsymbol{\alpha}}^k$ can be derived in a decomposed way

$$\begin{aligned}
\tilde{\boldsymbol{\alpha}}^k =&\nabla_{\tilde{\gamma}} v(\tilde{\boldsymbol{\gamma}}, \bar{\boldsymbol{\gamma}}^k, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*))^{\top}|_{\tilde{\gamma}=\tilde{\gamma}^k} \\
=&\nabla_{\tilde{\gamma}}\left[\log|\tilde{\mathbf{B}}^{\top}(\tilde{\boldsymbol{\Gamma}})^{-1}\tilde{\mathbf{B}} + \bar{\mathbf{B}}^{\top}(\bar{\boldsymbol{\Gamma}}^k)^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*)| + \log|\tilde{\boldsymbol{\Gamma}}| + \log|\bar{\boldsymbol{\Gamma}}^k|\right]|_{\tilde{\gamma}=\tilde{\gamma}^k} \\
=&\operatorname{diag}\{\tilde{\mathbf{B}}\left[(\tilde{\mathbf{B}}^{\top}\tilde{\boldsymbol{\Gamma}}^k)^{-1}\tilde{\mathbf{B}} + \bar{\mathbf{B}}^{\top}(\bar{\boldsymbol{\Gamma}}^k)^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*)\right]^{-1}\tilde{\mathbf{B}}^{\top}\} \cdot \operatorname{diag}\{-(\tilde{\boldsymbol{\Gamma}}^k)^{-2}\} \\
&+\operatorname{diag}^{-1}\{\tilde{\boldsymbol{\Gamma}}^k\} \\
=&\Big[\underbrace{\tilde{\alpha}_{11}^k, \ldots, \tilde{\alpha}_{1\tilde{\aleph}_1}^k}_{\tilde{\aleph}_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\alpha}_{\tilde{O}1}^k, \ldots, \tilde{\alpha}_{\tilde{O}\tilde{\aleph}_{\tilde{O}}}^k}_{\tilde{\aleph}_{\tilde{O}} \textbf{ elements}} \Big].
\end{aligned} \tag{7.78}$$

It is found that $\tilde{\alpha}_{\tilde{o}}^k$ is a function of $\tilde{\gamma}_{\tilde{o}}^k$. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}$ first to calculate $\tilde{\boldsymbol{\gamma}}^{k+1}$.

We then look at the hyperparameter $\bar{\boldsymbol{\Gamma}}$. For $\bar{o} = 1, \ldots, \bar{O}$, since

$$\frac{\bar{\boldsymbol{\beta}}_{\bar{o}}^{\top}\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}^{\top}\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}}{\bar{\gamma}_{\bar{o}i}} + \bar{\alpha}_{\bar{o}i}^k\bar{\gamma}_{\bar{o}i} \geq 2\left|\sqrt{\bar{\alpha}_{\bar{o}i}^k} \cdot \{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}\right| \tag{7.79}$$

the optimal $\bar{\boldsymbol{\gamma}}$ can be obtained as:

$$\bar{\gamma}_{\bar{o}i}^{k+1} = \frac{|\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}|}{\sqrt{\bar{\alpha}_{\bar{o}i}^k}}, \forall \bar{o} = 1, \ldots, \bar{O}, i = 1, \ldots, \bar{\aleph}_{\bar{o}}. \tag{7.80}$$

And $\bar{\boldsymbol{\alpha}}^k$ is defined as

$$\bar{\boldsymbol{\alpha}}^k = \Big[\underbrace{\bar{\alpha}^k_{11}, \ldots, \bar{\alpha}^k_{1\bar{\aleph}_1}}_{\bar{\aleph}_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\bar{\alpha}^k_{\bar{O}1}, \ldots, \bar{\alpha}^k_{\bar{O}\bar{\aleph}_{\bar{o}}}}_{\bar{\aleph}_{\bar{O}} \textbf{ elements}} \Big]. \tag{7.81}$$

It should be noted that, we can obtain the following analytic form for the negative gradient of $v(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}$ using basic principles in convex analysis as (using chain rule):

$$\begin{aligned}
\bar{\boldsymbol{\alpha}}^k =& \nabla_{\bar{\gamma}} v(\tilde{\boldsymbol{\gamma}}^k, \bar{\boldsymbol{\gamma}}, \mathbf{H}(\boldsymbol{\beta}^*, \boldsymbol{\theta}^*))^\top|_{\bar{\gamma}=\bar{\gamma}^k} \\
=& \nabla_{\bar{\gamma}} \left[\log|\tilde{\mathbf{B}}^\top(\tilde{\boldsymbol{\Gamma}}^k)^{-1}\tilde{\mathbf{B}} + \bar{\mathbf{B}}^\top\bar{\boldsymbol{\Gamma}}^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*)| + \log|\tilde{\mathbf{B}}^\top\tilde{\boldsymbol{\Gamma}}^k\tilde{\mathbf{B}}| + \log|\bar{\boldsymbol{\Gamma}}| \right] |_{\bar{\gamma}=\bar{\gamma}^k} \\
=& \operatorname{diag}\{\bar{\mathbf{B}} \left[\tilde{\mathbf{B}}^\top(\tilde{\boldsymbol{\Gamma}}^k)^{-1}\tilde{\mathbf{B}} + \bar{\mathbf{B}}^\top(\bar{\boldsymbol{\Gamma}}^k)^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1}, \boldsymbol{\theta}^*) \right]^{-1} \bar{\mathbf{B}}^\top\} \cdot \operatorname{diag}\{-(\bar{\boldsymbol{\Gamma}}^k)^{-2}\} \\
& + \operatorname{diag}^{-1}\{\bar{\boldsymbol{\Gamma}}^k\} \\
=& \Big[\underbrace{\bar{\alpha}^k_{11}, \ldots, \bar{\alpha}^k_{1\bar{\aleph}_1}}_{\bar{\aleph}_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\bar{\alpha}^k_{\bar{O}1}, \ldots, \bar{\alpha}^k_{\bar{O}\bar{\aleph}_{\bar{o}}}}_{\bar{\aleph}_{\bar{O}} \textbf{ elements}} \Big].
\end{aligned} \tag{7.82}$$

It is found that $\bar{\alpha}^k_{\bar{o}i}$ is a function of $\bar{\gamma}^k_{\bar{o}i}$. Therefore we need to estimate $\boldsymbol{\beta}^{k+1}_{\bar{o}}$ first to calculate $\bar{\boldsymbol{\gamma}}^{k+1}_{\bar{o}i}$.

Two new variables related to weights are defined:

$$\tilde{w}^k_{\tilde{o}} \triangleq \sqrt{\sum_{i=1}^{\tilde{\aleph}_{\tilde{o}}} \tilde{\alpha}^k_{\tilde{o}i}}, \tag{7.83}$$

and

$$\bar{w}^k_{\bar{o}i} \triangleq \sqrt{\bar{\alpha}^k_{\bar{o}i}}, \tag{7.84}$$

The pseudo code is summarised in Algorithm 14.

Similarly, we can derive the reweighted *generalised fused group* ℓ_2 type algorithm. The pseudo code is summarised in Algorithm 15.

Similarly, let $\mathbf{B}$ be an identity matrix

$$\begin{bmatrix} \tilde{\mathbf{B}}_1 & & \\ & \ddots & \\ & & \tilde{\mathbf{B}}_{\tilde{O}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_1 & & \\ & \ddots & \\ & & \mathbf{I}_{\tilde{O}} \end{bmatrix}.$$

The pseudo code is summarised in Algorithm 16.

Algorithm 14 Reweighted *Generalised Fused Group* ℓ_1 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta}); \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $\tilde{w}_{\tilde{o}}^1 = 1, \bar{w}_{\bar{o}i}^1 = 1, \ \forall i$;
5: Fix $\tilde{\lambda} = 1$ and $\bar{\lambda} = 1$ or select $\tilde{\lambda} \in \mathbb{R}^+$ and $\bar{\lambda} \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Fused Group Lasso* type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \tilde{\lambda}\sum_{\tilde{o}=1}^{\tilde{O}} \left\| \tilde{w}_{\tilde{o}}^k \cdot \tilde{\mathbf{B}}_{\tilde{o}}\tilde{\boldsymbol{\beta}}_{\tilde{o}} \right\|_{\ell_2} + \bar{\lambda}\sum_{\bar{o}=1}^{\bar{O}}\sum_{i=1}^{\bar{\aleph}_{\bar{o}}} \left\| \bar{w}_{\bar{o}i}^k \cdot \{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}} \right\|_{\ell_1}; \tag{7.85}$$

8: $\tilde{\gamma}_{\tilde{o}}^{k+1} = \frac{\left\|\tilde{\mathbf{B}}_{\tilde{o}}\tilde{\boldsymbol{\beta}}_{\tilde{o}}^{k+1}\right\|_{\ell_2}}{\tilde{w}_{\tilde{o}}^k}$, and $\tilde{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\tilde{\gamma}_1^{k+1}, \ldots, \tilde{\gamma}_1^{k+1}}_{\tilde{\aleph}_1 \text{ elements}} \Big| \cdots \Big| \underbrace{\tilde{\gamma}_{\tilde{O}}^{k+1}, \ldots, \tilde{\gamma}_{\tilde{O}}^{k+1}}_{\tilde{\aleph}_{\tilde{O}} \text{ elements}} \Big]$;

9: $\bar{\gamma}_{\bar{o}i}^{k+1} = \frac{|\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}^{k+1}|}{\bar{w}_{\bar{o}i}^k}$, and $\bar{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\bar{\gamma}_{11}^{k+1}, \ldots, \bar{\gamma}_{1\bar{\aleph}_1}^{k+1}}_{\bar{\aleph}_1 \text{ elements}} \Big| \cdots \Big| \underbrace{\bar{\gamma}_{\bar{O}1}^{k+1}, \ldots, \bar{\gamma}_{\bar{O}\bar{\aleph}_{\bar{O}}}^{k+1}}_{\bar{\aleph}_{\bar{O}} \text{ elements}} \Big]$;

10: $\mathbf{C}^{k+1} = \left(\tilde{\mathbf{B}}^\top(\tilde{\boldsymbol{\Gamma}}^{k+1})^{-1}\tilde{\mathbf{B}} + \bar{\mathbf{B}}^\top(\bar{\boldsymbol{\Gamma}}^{k+1})^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1},\boldsymbol{\theta}^*)\right)^{-1}$;

11: $\tilde{\alpha}_{\tilde{o}i}^{k+1} = -\frac{\{\tilde{\mathbf{B}}_{\tilde{o}}\}_{i,:}\mathbf{C}^{k+1}\{\tilde{\mathbf{B}}_{\tilde{o}}\}_{i,:}^\top}{(\tilde{\gamma}_{\tilde{o}}^{k+1})^2} + \frac{1}{\tilde{\gamma}_{\tilde{o}}^{k+1}}$;

12: $\tilde{w}_{\tilde{o}}^{k+1} = \sqrt{\sum_{i=1}^{\tilde{\aleph}_{\tilde{o}}} \tilde{\alpha}_{\tilde{o}i}^{k+1}}, \quad \forall \tilde{o} = 1, \ldots, \tilde{O}$;

13: $\bar{\alpha}_{\bar{o}i}^{k+1} = -\frac{\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\mathbf{C}^{k+1}\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}^\top}{(\bar{\gamma}_{\bar{o}}^{k+1})^2} + \frac{1}{\bar{\gamma}_{\bar{o}}^{k+1}}$;

14: $\bar{w}_{\bar{o}i}^{k+1} = \sqrt{\bar{\alpha}_{\bar{o}i}^{k+1}}, \quad \forall \bar{o} = 1, \ldots, \bar{O}, i = 1, \ldots, \bar{\aleph}_{\bar{o}}$;

15: **if** a stopping criterion is satisfied **then**
16: Break;
17: **end if**
18: **end for**

Algorithm 15 Reweighted *Generalised Fused Group* ℓ_2 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta}); \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $\tilde{w}_{\tilde{o}}^1 = 1, \bar{w}_{\bar{o}i}^1 = 1, \ \forall i$;
5: Fix $\tilde{\lambda} = 1$ and $\bar{\lambda} = 1$ or select $\tilde{\lambda} \in \mathbb{R}^+$ and $\bar{\lambda} \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Fused Group* ℓ_2 type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \tilde{\lambda}\sum_{\tilde{o}=1}^{\tilde{O}} \left\| \tilde{w}_{\tilde{o}}^k \cdot \tilde{\mathbf{B}}_{\tilde{o}}\tilde{\boldsymbol{\beta}}_{\tilde{o}} \right\|_{\ell_2}^2 + \bar{\lambda}\sum_{\bar{o}=1}^{\bar{O}}\sum_{i=1}^{\bar{\aleph}_{\bar{o}}} \left\| \bar{w}_{\bar{o}i}^k \cdot \{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}} \right\|_{\ell_2}^2; \tag{7.86}$$

8: $\mathbf{C}^k = \left(\tilde{\mathbf{B}}^\top(\tilde{\boldsymbol{\Gamma}}^k)^{-1}\tilde{\mathbf{B}} + \bar{\mathbf{B}}^\top(\bar{\boldsymbol{\Gamma}}^k)^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1},\boldsymbol{\theta}^*)\right)^{-1}$;
9: $\tilde{\alpha}_{\tilde{o}i}^{k+1} = -\frac{\{\tilde{\mathbf{B}}_{\tilde{o}}\}_{i,:}\mathbf{C}^k\{\tilde{\mathbf{B}}_{\tilde{o}}\}_{i,:}^\top}{(\tilde{\gamma}_{\tilde{o}}^k)^2} + \frac{1}{\tilde{\gamma}_{\tilde{o}}^k}$;
10: $\tilde{\gamma}_{\tilde{o}}^{k+1} = \frac{\left\|\tilde{\mathbf{B}}_{\tilde{o}}\tilde{\boldsymbol{\beta}}_{\tilde{o}}^{k+1}\right\|_{\ell_2}}{\sqrt{\sum_{i=1}^{\tilde{\aleph}_{\tilde{o}}}\tilde{\alpha}_{\tilde{o}i}^{k+1}}}$, and $\tilde{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\tilde{\gamma}_1^{k+1},\ldots,\tilde{\gamma}_1^{k+1}}_{\tilde{\aleph}_1 \textbf{ elements}} \Big| \cdots \Big| \underbrace{\tilde{\gamma}_{\tilde{O}}^{k+1},\ldots,\tilde{\gamma}_{\tilde{O}}^{k+1}}_{\tilde{\aleph}_{\tilde{O}} \textbf{ elements}} \Big]$;
11: $\tilde{w}_{\tilde{o}}^{k+1} = \frac{1}{\sqrt{\tilde{\gamma}_{\tilde{o}}^{k+1}}}, \quad \forall \tilde{o} = 1,\ldots,\tilde{O}$;
12: $\bar{\alpha}_{\bar{o}i}^{k+1} = -\frac{\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\mathbf{C}^k\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}^\top}{(\bar{\gamma}_{\bar{o}}^k)^2} + \frac{1}{\bar{\gamma}_{\bar{o}}^k}$;
13: $\bar{\gamma}_{\bar{o}i}^{k+1} = \frac{|\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}^{k+1}|}{\sqrt{\bar{\alpha}_{\bar{o}i}^{k+1}}}$, and $\bar{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\bar{\gamma}_{11}^{k+1},\ldots,\bar{\gamma}_{1\bar{\aleph}_1}^{k+1}}_{\bar{\aleph}_1 \textbf{ elements}} \Big| \cdots \Big| \underbrace{\bar{\gamma}_{\bar{O}1}^{k+1},\ldots,\bar{\gamma}_{\bar{O}\bar{\aleph}_{\bar{O}}}^{k+1}}_{\bar{\aleph}_{\bar{O}} \textbf{ elements}} \Big]$;
14: $\bar{w}_{\bar{o}i}^{k+1} = \frac{1}{\sqrt{\bar{\gamma}_{\bar{o}i}^{k+1}}}, \quad \forall \bar{o} = 1,\ldots,\bar{O}, i = 1,\ldots,\bar{\aleph}_{\bar{o}}$;
15: **if** a stopping criterion is satisfied **then**
16: Break;
17: **end if**
18: **end for**

Algorithm 16 Reweighted *Fused Group* ℓ_1 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta}); \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $\tilde{w}_{\tilde{o}}^1 = 1, \bar{w}_{\bar{o}i}^1 = 1, \ \forall i$;
5: Fix $\tilde{\lambda} = 1$ and $\bar{\lambda} = 1$ or select $\tilde{\lambda} \in \mathbb{R}^+$ and $\bar{\lambda} \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Fused Group Lasso* type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \tilde{\lambda}\sum_{\tilde{o}=1}^{\tilde{O}} \left\|\tilde{w}_{\tilde{o}}^k \cdot \tilde{\boldsymbol{\beta}}_{\tilde{o}}\right\|_{\ell_2} + \bar{\lambda}\sum_{\bar{o}=1}^{\bar{O}}\sum_{i=1}^{\bar{\aleph}_{\bar{o}}} \left\|\bar{w}_{\bar{o}i}^k \cdot \{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}\right\|_{\ell_1}; \tag{7.87}$$

8: $\tilde{\gamma}_{\tilde{o}}^{k+1} = \frac{\left\|\tilde{\beta}_{\tilde{o}}^{k+1}\right\|_{\ell_2}}{\tilde{w}_{\tilde{o}}^k}$, and $\tilde{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\tilde{\gamma}_1^{k+1}, \ldots, \tilde{\gamma}_1^{k+1}}_{\tilde{\aleph}_1 \text{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\gamma}_{\tilde{O}}^{k+1}, \ldots, \tilde{\gamma}_{\tilde{O}}^{k+1}}_{\tilde{\aleph}_{\tilde{O}} \text{ elements}} \Big]$;
9: $\bar{\gamma}_{\bar{o}i}^{k+1} = \frac{|\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}^{k+1}|}{\bar{w}_{\bar{o}i}^k}$, and $\bar{\boldsymbol{\gamma}}^{k+1} = \Big[\underbrace{\bar{\gamma}_{11}^{k+1}, \ldots, \bar{\gamma}_{1\bar{\aleph}_1}^{k+1}}_{\bar{\aleph}_1 \text{ elements}} \Big| \ldots \Big| \underbrace{\bar{\gamma}_{\bar{O}1}^{k+1}, \ldots, \bar{\gamma}_{\bar{O}\bar{\aleph}_{\bar{O}}}^{k+1}}_{\bar{\aleph}_{\bar{O}} \text{ elements}} \Big]$;
10: $\mathbf{C}^{k+1} = \left((\tilde{\boldsymbol{\Gamma}}^{k+1})^{-1} + \bar{\mathbf{B}}^\top(\bar{\boldsymbol{\Gamma}}^{k+1})^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1},\boldsymbol{\theta}^*)\right)^{-1}$;
11: $\tilde{\alpha}_{\tilde{o}}^{k+1} = -\frac{\mathbf{C}^{k+1}}{(\tilde{\gamma}_{\tilde{o}}^{k+1})^2} + \frac{1}{\tilde{\gamma}_{\tilde{o}}^{k+1}}$;
12: $\tilde{w}_{\tilde{o}}^{k+1} = \sqrt{\tilde{\aleph}_{\tilde{o}} \cdot \tilde{\alpha}_{\tilde{o}}^{k+1}}$;
13: $\bar{\alpha}_{\bar{o}i}^{k+1} = -\frac{\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\mathbf{C}^{k+1}\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}^\top}{(\tilde{\gamma}_{\tilde{o}}^{k+1})^2} + \frac{1}{\tilde{\gamma}_{\tilde{o}}^{k+1}}$;
14: $\bar{w}_{\bar{o}i}^{k+1} = \sqrt{\bar{\alpha}_{\bar{o}i}^{k+1}}$;
15: **if** a stopping criterion is satisfied **then**
16: Break;
17: **end if**
18: **end for**

Algorithm 17 Reweighted *Fused Group* ℓ_2 type algorithm for likelihood in exponential family

1: Symbolic cache likelihood, gradient and Hessian expressions

$$p(\mathbf{y}|\boldsymbol{\beta},\boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\boldsymbol{\beta},\boldsymbol{\theta})\}; \mathbf{g}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla E(\boldsymbol{\beta},\boldsymbol{\theta}); \mathbf{H}(\boldsymbol{\beta},\boldsymbol{\theta}) = \nabla\nabla E(\boldsymbol{\beta},\boldsymbol{\theta});$$

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}^1$ as a unit vector;
3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;
4: Initialise $\tilde{w}_{\tilde{o}}^1 = 1, \bar{w}_{\bar{o}i}^1 = 1, \ \forall i$;
5: Fix $\tilde{\lambda} = 1$ and $\bar{\lambda} = 1$ or select $\tilde{\lambda} \in \mathbb{R}^+$ and $\bar{\lambda} \in \mathbb{R}^+$ as trail and error which may be empirically helpful;
6: **for** $k = 1, \ldots, k_{\max}$ **do**
7: Solve the following *Fused Group* ℓ_2 type algorithm

$$\boldsymbol{\beta}^{k+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} E(\boldsymbol{\beta},\boldsymbol{\theta}^*) + \tilde{\lambda}\sum_{\tilde{o}=1}^{\tilde{O}} \left\|\tilde{w}_{\tilde{o}}^k \cdot \tilde{\boldsymbol{\beta}}_{\tilde{o}}\right\|_{\ell_2}^2 + \bar{\lambda}\sum_{\bar{o}=1}^{\bar{O}}\sum_{i=1}^{\bar{\aleph}_{\bar{o}}} \left\|\bar{w}_{\bar{o}i}^k \cdot \{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}\right\|_{\ell_2}^2; \tag{7.88}$$

8: $\mathbf{C}^k = \left((\tilde{\boldsymbol{\Gamma}}^k)^{-1} + \bar{\mathbf{B}}^\top(\bar{\boldsymbol{\Gamma}}^k)^{-1}\bar{\mathbf{B}} + \mathbf{H}(\boldsymbol{\beta}^{k+1},\boldsymbol{\theta}^*)\right)^{-1}$;
9: $\tilde{\alpha}_{\tilde{o}}^{k+1} = -\frac{\mathbf{C}^k}{(\tilde{\gamma}_{\tilde{o}}^k)^2} + \frac{1}{\tilde{\gamma}_{\tilde{o}}^k}$;
10: $\tilde{\gamma}_{\tilde{o}}^{k+1} = \frac{\|\tilde{\boldsymbol{\beta}}_{\tilde{o}}^{k+1}\|_{\ell_2}}{\sqrt{\tilde{\aleph}_{\tilde{o}} \cdot \tilde{\alpha}_{\tilde{o}}^{k+1}}}$, and $\tilde{\boldsymbol{\gamma}}^{k+1} = \Big[\ \underbrace{\tilde{\gamma}_1^{k+1}, \ldots, \tilde{\gamma}_1^{k+1}}_{\tilde{\aleph}_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\tilde{\gamma}_{\tilde{O}}^{k+1}, \ldots, \tilde{\gamma}_{\tilde{O}}^{k+1}}_{\tilde{\aleph}_{\tilde{O}} \textbf{ elements}} \ \Big]$;
11: $\tilde{w}_{\tilde{o}}^{k+1} = \frac{1}{\sqrt{\tilde{\gamma}_{\tilde{o}}^{k+1}}}$;
12: $\bar{\alpha}_{\bar{o}i}^{k+1} = -\frac{\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\mathbf{C}^k\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}^\top}{(\bar{\gamma}_{\bar{o}}^k)^2} + \frac{1}{\bar{\gamma}_{\bar{o}}^k}$;
13: $\bar{\gamma}_{\bar{o}i}^{k+1} = \frac{|\{\bar{\mathbf{B}}_{\bar{o}}\}_{i,:}\bar{\boldsymbol{\beta}}_{\bar{o}}^{k+1}|}{\sqrt{\bar{\alpha}_{\bar{o}i}^{k+1}}}$, and $\bar{\boldsymbol{\gamma}}^{k+1} = \Big[\ \underbrace{\bar{\gamma}_{11}^{k+1}, \ldots, \bar{\gamma}_{1\bar{\aleph}_1}^{k+1}}_{\bar{\aleph}_1 \textbf{ elements}} \Big| \ldots \Big| \underbrace{\bar{\gamma}_{\bar{O}1}^{k+1}, \ldots, \bar{\gamma}_{\bar{O}\bar{\aleph}_{\bar{O}}}^{k+1}}_{\bar{\aleph}_{\bar{O}} \textbf{ elements}} \ \Big]$;
14: $\bar{w}_{\bar{o}i}^{k+1} = \frac{1}{\sqrt{\bar{\gamma}_{\bar{o}i}^{k+1}}}$;
15: **if** a stopping criterion is satisfied **then**
16: Break;
17: **end if**
18: **end for**

Similarly, we can derive the reweighted *fused group* ℓ_2 type algorithm. The pseudo code is summarised in Algorithm 17.

Remark 20 (Remark on Algorithms in Section 7.5) *If the Algorithms in Section needs to be efficiently implemented like the one in Algorithm 9. The Maximum a Posterior procedure, a.k.a.,the* argmin *step, in the* ℓ_2 *type Algorithms, can be similarly substituted by the "inner for loop" procedures in Algorithm 9. More specifically, these Algorithms include Algorithm 11, 13, 15, 17.*

Remark 21 (Remark on the extension to train deep neural networks) *A possible and intuitive algorithm extension to train deep neural networks with structural sparsity can be found later in Section 14.1.2 and 14.1.4 of Chapter 14.*

Chapter 8

Algorithms for Online Model Selection

Our method allows inference of model structures that can be decomposed as linear combinations of nonlinear functions chosen from a dictionary set. We note, however, that the identification of parameters nonlinearly embedded in these functions is a non-trivial task. As we saw in the previous section, a naive approach consists in augmenting the set of dictionary functions with various candidate nonlinearities for which nonlinearly embedded parameters are given specific values. We would then rely on the approximation of the true nonlinearities as a linear combination of these dictionary functions, i.e. on estimating the true nonlinearity as an "interpolation" from discretely valued candidate nonlinearities.

On the other hand, filtering methods have been widely used to estimate parameters for a given (nonlinear) parametric structure [209]. The main issue with filtering methods is that they require *a priori* knowledge of such model structures and cannot easily be used to infer model structures in other ways than by trying individual structures and comparing them using model selection criteria. Typically, this process has a very high computational cost.

In the following section, we show how our model structure inference (described in Algorithm 5) can be used to identify nonlinear terms that can benefit from further refinement using filtering approaches. For example, if the right hand side of one of the equations in the identified model was to contain a linear combination of the form $\frac{0.2684}{1+x} + \frac{0.7292}{1+x^3}$, a new parametric structure could be created where this term is replaced by $\frac{V_{\max}}{K+x^h}$. This new parametric model structure can then serve as the starting point for filtering methods, which are then used to estimate the values of the parameters in the new parametric structure. Furthermore, the parameters identified using our model structure inference method can be used as initial guesses or priors for the filtering methods.

8.1 Extended Kalman Filter

As mentioned above, our model structure inference method can be used to identify nonlinear terms that can benefit from further refinement in their structure. Let the new parametric structure obtained through such refinement be given by:

$$\begin{cases} \mathbf{x}_{t+1} = \mathbf{g}(\mathbf{x}_t, \mathbf{u}_t, \gamma) + \boldsymbol{\xi}_t, \\ \mathbf{z}_t = \mathbf{x}_t + \boldsymbol{\eta}_t, \\ \mathbf{x}_1 = g_0, \end{cases} \tag{8.1}$$

where g_0 is the initial guess of the state vector $\mathbf{x}$.

Extended or unscented Kalman filtering are celebrated methods used to identify both the state variables in $\mathbf{x}$ and the parameters in γ of a given parametric model structure such as the one provided in (8.1). Simultaneous identification of state variables and parameters can be done using a "state extension" approach where constant parameters such as those contained in the model parameter vector γ are considered as additional state variables with a rate of change equal to zero. In this way, constant parameters are treated as constant functions of time as opposed to constant numbers [200].

The parameters identified using our model structure inference method (see Algorithm 5) can be used as initial guesses or priors for the parameters γ.

Filtering can be made more tractable by considering that the unknown parameters γ evolve according to a Brownian motion. For this, we introduce a new variable $\boldsymbol{\phi}_k$ and consider the following linear process model:

$$\begin{bmatrix} \boldsymbol{\phi}_{t+1} \\ \gamma_{t+1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ \Delta\tau & I \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_t \\ \gamma_t \end{bmatrix} + \boldsymbol{\varrho}_t \ , \tag{8.2}$$

where $\boldsymbol{\varrho}_t$ has covariance:

$$\mathbf{Q}_{\boldsymbol{\varrho}} := \sigma^2 \begin{bmatrix} \Delta\tau & \Delta\tau^2/2 \\ \Delta\tau^2/2 & \Delta\tau^3/3 \end{bmatrix} \ ,$$

where σ^2 must be chosen *a priori*. We further define the augmented state parametric structure as:

$$\begin{aligned} \bar{\mathbf{x}}_t &\triangleq \begin{bmatrix} \mathbf{x}_t \\ \boldsymbol{\phi}_t \\ \gamma_t \end{bmatrix}, \qquad \bar{\mathbf{g}}(\bar{\mathbf{x}}_t, \mathbf{u}_t) \triangleq \begin{bmatrix} \mathbf{g}(\mathbf{x}_t, \mathbf{u}_t, \gamma_t) \\ \boldsymbol{\phi}_t \\ \gamma_t + \Delta t \boldsymbol{\phi}_t \end{bmatrix}, \\ \bar{\boldsymbol{\xi}}_t &\triangleq \begin{bmatrix} \boldsymbol{\xi}_t \\ \boldsymbol{\varrho}_t \end{bmatrix}, \qquad\qquad \bar{g}_0 \triangleq \begin{bmatrix} \mathbf{g}_0 \\ \boldsymbol{\phi}_0 \\ \gamma_0 \end{bmatrix}, \end{aligned} \tag{8.3}$$

where $\bar{g}_0$ is the initial state estimate.

We can now write the full augmented dynamic model as

$$\begin{cases} \bar{\mathbf{x}}_{t+1} = \bar{\mathbf{g}}(\bar{\mathbf{x}}_t, \mathbf{u}_t) + \bar{\boldsymbol{\xi}}_t, \\ \mathbf{z}_t = [\mathbf{I}_{n_{\mathbf{x}}}, \mathbf{0}]\bar{\mathbf{x}}_t + \boldsymbol{\eta}_t, \\ \bar{\mathbf{x}}_1 = \bar{g}_0. \end{cases} \tag{8.4}$$

The new process noise $\bar{\boldsymbol{\xi}}_t$ has positive definite covariance matrix

$$\bar{\mathbf{Q}}_t = \begin{bmatrix} \mathbf{Q}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{\varrho} \end{bmatrix} .$$

Using such a state extension approach, the problem of parameter estimation is converted into a problem of state estimation, for which the goal is to estimate the extended state $\bar{\mathbf{x}}$ from measurements of the output $\mathbf{z}$. More precisely, we are trying to determine the initial conditions $\bar{g}_0$, which, when used to initialise the system (8.1), generates the observed output $\mathbf{z}$.

8.2 Algorithm combining model structure identification and model refinement

In this section we present Algorithm 18, which constitutes the main algorithm combining 1) our model structure identification method (Algorithm 5) with 2) model refinement of model structures using filtering. Model structure identification is done off-line and thus requires *batched data* (historical sensor measurements) which were collected *a priori*. Once a 'rough' model structure is obtained, model refinement can be performed on-line by feeding *streaming data* (sensor measurements that arrive in real-time).

In Algorithm 18, we define a trial as the application of the model structure identification procedure described in Algorithm 5 using a given set of dictionary functions and a given regularisation parameter λ.

Algorithm 18 Online Model Selection Algorithm

IDENTIFICATION:
Require: Batched Data
procedure IDENTIFICATION(S trials)
    for $s = 1, \ldots, S$ **do**
        Choose a regularisation parameter λ_s;
        Choose a set of dictionary functions;
        Using the set of dictionary functions, construct $\mathbf{X}_s$ from batched data;
        $\mathcal{M}_s = \texttt{IDENTIFICATION}(\lambda_s, \mathbf{X}_s)$; % Apply Algorithm 5to get a model $\mathcal{M}_s$;
    end for
    Pick the top $\hat{S}$ ranked $\mathcal{M}_s$ models based on a certain model selection criterion.
end procedure
Update:
procedure UPDATE($\hat{S}$ trials)
    Update candidate functions as stated in the introduction to Section 8.1;
end procedure
FILTERING:
Require: Streaming Data
procedure FILTERING($\hat{S}$ trials)
    while New data $\mathbf{z}_t$ is available **do**
        for $s = 1, \ldots, \hat{S}$ **do**
            $\mathcal{M}_s^{\text{new}} = \texttt{FILTERING}(\mathbf{z}_t)$; % Apply Filtering techniques to refine model $\mathcal{M}_s$
        end for
    end while
    if Not convergent **then goto** *IDENTIFICATION*
    end if
end procedure

Chapter 9

Algorithms for Fault Diagnosis

9.1 Fault Diagnosis Problem Formulation

We slightly change the notation of the linear regression model (4.28) in Chapter 4.5.2. we can write (4.28) into a vector form:

$$e_i(t+1) = f_i(\mathbf{x}(t))\boldsymbol{\beta}_i^{\text{true}} + \eta_i(t), \tag{9.1}$$

with

$$f_i(\mathbf{x}(t)) = [f_{i1}(\mathbf{x}(t)), \ldots, f_{iN}(\mathbf{x}(t))] \in \mathbb{R}^N, \tag{9.2}$$

$$\boldsymbol{\beta}_i^{\text{true}} = [\beta_{i1}, \ldots, \beta_{iN}]^\top \in \mathbb{R}^N. \tag{9.3}$$

As stated in Definition 2, if there are no faults occurring in the system, the dynamics of the system will evolve according to (9.1). The **e**xpected output for the next sampling time is defined to be

$$e_i^{[\mathbf{e}]}(t+1) = f_i(\mathbf{x}(t))\boldsymbol{\beta}_i^{\text{true}}. \tag{9.4}$$

From (9.1) and (9.4), it is easy to show that $e_i(t+1) - e_i^{[\mathbf{e}]}(t+1)$ is a stochastic variable with zero mean and variance σ^2. If there are faults occurring in the system, the corresponding weights will change from $\boldsymbol{\beta}_i^{\text{true}}$ to $\boldsymbol{\beta}_i^{\text{fault}}$. Similar to the definition of $\boldsymbol{\beta}_i^{\text{true}}$, $\boldsymbol{\beta}_i^{\text{fault}} = [\beta_{i1}^{[\mathbf{f}]}, \ldots, \beta_{iN}^{[\mathbf{f}]}]^\top$. We thus have:

$$e_i^{[\mathbf{f}]}(t+1) = f_i(\mathbf{x}(t))\boldsymbol{\beta}_i^{\text{fault}} + \eta_i(t), \tag{9.5}$$

where $e_i^{[\mathbf{f}]}$ is the output when there are **f**aults.

From (9.4) and (9.5), it is easy to find that $e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)$ is a stochastic variable with

$$\begin{cases} \text{mean:} & f_i(\mathbf{x}(t))(\boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}}) \\ \text{variance:} & \sigma^2 \end{cases}$$

Denoting

$$y_i = e_i^{[\mathbf{f}]} - e_i^{[\mathbf{e}]}, \boldsymbol{\beta}_i = \boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}},$$

we have:

$$y_i(t+1) = f_i(\mathbf{x}(t))\boldsymbol{\beta}_i + \eta_i(t). \tag{9.6}$$

Remark 22 *We formulate the faults identification problem as a linear regression problem. The dependent variable* $e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)$ *is the difference between the expected output and the faulty output; the unknown variable we want to estimate is the difference between the faulty transmission weights and the true transmission weights.*

9.2 Fault Detection and Isolation Algorithm

There are three problems of interest based on the formulation in (9.6): a) detection of a fault; b) isolation of a fault, i.e. determination of the type, location and time of occurrence of a fault; and c) identification of the size and time-varying behaviour of a fault. In the noiseless case, when there are no faults, $\forall i$, y_i and $\boldsymbol{\beta}_i$ are both equal to zero. On the other hand, when there are faults, certain y_i are nonzero. So the faults can be *detected* by identifying the entries y_i that are nonzero. However, in the noisy case, even when there are no faults, y_i is nonzero most of the time since it is a stochastic variable with zero mean. This can be interpreted in a probabilistic way by Chebyshev's Inequality:

$$p(|e_i(t+1) - e_i^{[\mathbf{e}]}(t+1)| \geq l\sigma = \sigma^*) \leq \frac{1}{l^2}$$

where $l \in \mathbb{R}^+$. According to this inequality, when there are no faults, the deviation between true and expected outputs, i.e. $|e_i(t+1) - e_i^{[\mathbf{e}]}(t+1)|$ cannot be much greater than zero with high probability. On the other hand, when there is a fault, the deviation between faulty and expected outputs, i.e. $|e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)|$ should be much greater than zero with high probability.

From an isolation point of view and Chebyshev's inequality, when $|e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)|$ is much greater than σ, the fault can be *isolated* with high probability (e.g. if the threshold is set to $l\sigma = 10\sigma$, then the probability is 99%).

9.3 Fault Identification Algorithm

If at time t_1 faults have been detected and isolated, the remaining task is to perform fault *identification*, i.e. to identify the location of the faults or equivalently to find the nonzero entries in β_i. Assuming that M successive data points, including the initial

data point at t_1, are sampled and defining

$$\begin{aligned} \mathbf{y}_i &\triangleq [y_i(t_1), \ldots, y_i(M)]^\top \in \mathbb{R}^M, \\ \mathbf{X}_i &\triangleq \begin{bmatrix} f_i(\mathbf{x}(t_1)) \\ \vdots \\ f_i(\mathbf{x}(M)) \end{bmatrix} \in \mathbb{R}^{M \times N}, \\ \boldsymbol{\eta}_i &\triangleq [\eta_i(t_1), \ldots, \eta_i(M)]^\top \in \mathbb{R}^M, \end{aligned} \tag{9.7}$$

we can write N independent equations of the form:

$$\mathbf{y}_i = \mathbf{X}_i \boldsymbol{\beta}_i + \boldsymbol{\eta}_i, \quad i = 1, \ldots, N. \tag{9.8}$$

Based on the formulation in (9.8), our goal is to find $\boldsymbol{\beta}_i$ given the output data stored in $\mathbf{y}_i$. $\mathbf{y}$ is the difference between the faulty measurements and the expected measurements, or namely, the **error measurements**; and $\boldsymbol{\beta}$ is the difference between the faulty parameters and the true parameters, or namely, the **faults**. We address this linear regression problem under the following assumption.

Assumption 10 *A maximum of S system parameters are faulty, i.e. $\boldsymbol{\beta}_i$ has at most S non-zero entries. In other words, $\boldsymbol{\beta}_i$ is S-sparse or mathematically, $\|\boldsymbol{\beta}_i\| \leq S$. The constant S is assumed unknown to the system administrator.*

On the other-hand, the size of $\mathbf{y}$ equals to the number of samples needed to identify the location of the faults after the they occur. From a practical viewpoint, the number of samples should be as small as possible. However, standard least square approaches to (9.7) cannot meet this goal as they require at least $2N$ samples. Moreover, the solution to the standard least square problem is generically dense (hence, violating Assumption 10) and cannot be used to identify which transmission lines are likely to be faulty by identification of the nonzero entries of the estimated $\boldsymbol{\beta}^{\text{fault}} - \boldsymbol{\beta}^{\text{true}}$.

Based on Algorithm 3, we can summarise the fault diagnosis algorithm in Algorithm 19.

Algorithm 19 Fault Diagnosis Algorithm

Set a threshold σ^* as indicated in Section 9.2, e.g. $\sigma^* = 10 \times \sigma$;
for $k = 0, \ldots, T$ **do**
    $\%$ *T is an integer indicating the number of diagnosis rounds*;
    for $i = 1, \ldots, N$ **do**
        Collect the output data $e_i(t+1)$ in (9.1);
        Calculate the expected output $e_i^{[\mathbf{e}]}(t+1)$ in (9.4);
        if $|e_i(t+1) - e_i^{[\mathbf{e}+]}(t+1)| > \sigma^*$ **then**
            Fault is detected for β_i; $\%$ {*fault detection procedure*}
            Compute $y_i(t+1)$ in (9.6);
            if $|y_i(t+1)| > \sigma^*$ **then**
                Isolate fault i; $\%$ {*fault isolation procedure*}
            end if
        end if
        Set $M \leftarrow k$;
        Apply Algorithms propose in Chapter 6.6 to identify the faults $\hat{\boldsymbol{\beta}}_i$; $\%$ {*fault identification procedure*}
    end for
    if $\forall i$, $\|\hat{\boldsymbol{\beta}}_i\|_0$ converge to some constant **then**
        Break;
    end if
end for
An estimate for the faults $\hat{\boldsymbol{\beta}}$ in (9.8), $i = 1, \ldots, N$.

Part III

Applications

Chapter 10

Biochemical Reaction Network Identification

10.1 Identification from Single Time Series Data

In this example, we consider a classical dynamical system in systems/synthetic biology, the repressilator, which we use to illustrate the reconstruction problem at hand. The repressilator is a synthetic three-gene regulatory network where the dynamics of mRNAs and proteins follow an oscillatory behaviour [55]. A discrete-time mathematical description of the repressilator, which includes both transcription and translation dynamics, is given by the following set of discrete-time equations:

$$\begin{aligned}
\frac{x_1(t+1)-x_1(t)}{\Delta t} &= -\gamma_1 x_1(t) + \frac{\alpha_1}{(1+x_6^{n_1}(t))} + \xi_1(t),\\
\frac{x_2(t+1)-x_2(t)}{\Delta t} &= -\gamma_2 x_2(t) + \frac{\alpha_2}{(1+x_4^{n_2}(t))} + \xi_2(t),\\
\frac{x_3(t+1)-x_3(t)}{\Delta t} &= -\gamma_3 x_3(t) + \frac{\alpha_3}{(1+x_5^{n_3}(t))} + \xi_3(t),\\
\frac{x_4(t+1)-x_4(t)}{\Delta t} &= -\gamma_4 x_4(t) + \beta_1 x_1(k) + \xi_4(t),\\
\frac{x_5(t+1)-x_5(t)}{\Delta t} &= -\gamma_5 x_5(k) + \beta_2 x_2 + \xi_5(t),\\
\frac{x_6(t+1)-x_6(t)}{\Delta t} &= -\gamma_6 x_6(t) + \beta_3 x_3(t) + \xi_6(t).
\end{aligned}$$

Here, x_1, x_2, x_3 (resp. x_4, x_5, x_6) denote the concentrations of the mRNA transcripts (resp. proteins) of genes 1, 2, and 3, respectively. ξ_i, $\forall i$ are i.i.d. Gaussian noise. $\alpha_1, \alpha_2, \alpha_3$ denote the maximum promoter strength for their corresponding gene, $\gamma_1, \gamma_2, \gamma_3$ denote the mRNA degradation rates, $\gamma_4, \gamma_5, \gamma_6$ denote the protein degradation rates, $\beta_1, \beta_2, \beta_3$ denote the protein production rates, and n_1, n_2, n_3 the Hill coefficients. The set of equations in (10.1) corresponds to a topology where gene 1 is repressed by gene 2, gene 2 is repressed by gene 3, and gene 3 is repressed by gene 1. Take gene 1 for example. The hill coefficient n_1 will typically have a value within a range from 1 to 4 due to biochemical constraints. The core question here is: how can we determine the topology and kinetic parameters of the set of equations in (10.1) from time series data of $x_1, \ldots, x_6$?

Note that we do not assume *a priori* knowledge of the form of the nonlinear functions appearing on the right-hand side of the equations in (10.1), e.g., whether the degradation obeys fi rst-order or enzymatic catalysed dynamics or whether the proteins are repressors or activators. It should also be noted that many linear and nonlinear functions can be used to describe the dynamics of GRNs in terms of biochemical kinetic laws, e.g., first-order functions $f([S]) = [S]$, mass action functions

$f([S_1], [S_2]) = [S_1] \cdot [S_2]$, Michaelis-Menten functions $f([S]) = V_{\max}[S]/(K_M + [S])$, or Hill functions $f([S]) = V_{\max}[S]^n/(K_M^n + [S]^n)$. These kinetic laws typical of biochemistry and GRN models will aid in the definition of the dictionary function matrix. Next we show how the network construction problem of the repressilator model in (10.1) can be formulated in a linear regression form.

We construct a candidate dictionary matrix $\mathbf{X}$, by selecting as candidate basis functions, nonlinear functions typically used to represent terms appearing in biochemical kinetic laws of GRN models. As a proof of concept, we only consider Hill functions as potential nonlinear candidate functions. The set of Hill functions with Hill coefficient h, both in activating and repressing from, for each of the 6 state variables are:

$$\text{hill}_h(t) \triangleq \left[\frac{1}{1+x_1^h(t)}, \ldots, \frac{1}{1+x_6^h(t)}, \frac{x_1^h(t)}{1+x_1^h(t)}, \ldots, \frac{x_6^h(t)}{1+x_6^h(t)}\right]_{1\times 12}, \tag{10.1}$$

where h represents the Hill coefficient. In what follows we consider that the Hill coefficient can take any of the following integer values: 1, 2, 3 or 4. Since there are 6 state variables, we can construct the dictionary matrix $\mathbf{X}$ with 6 (dictionary functions for linear terms) $+(4 * 12)$ (dictionary functions for Hill functions) $= 54$ columns.

$$\mathbf{X} = \begin{bmatrix} x_1(0) & \ldots & x_6(0) & \text{hill}_1(0) & \ldots & \text{hill}_4(0) \\ \vdots & & \vdots & \vdots & & \vdots \\ x_1(M-1) & \ldots & x_6(M-1) & \text{hill}_1(M-1) & \ldots & \text{hill}_4(M-1) \end{bmatrix} \in \mathbb{R}^{M\times(6+48)}. \tag{10.2}$$

Then the output can be defined as

$$\mathbf{y}_i \triangleq \left[\frac{x_i(1)-x_i(0)}{\Delta t}, \ldots, \frac{x_i(M)-x_i(M-1)}{\Delta t}\right]^\top \in \mathbb{R}^{M\times 1}, i = 1, \ldots, 6.$$

Considering the dictionary matrix $\mathbf{X}$ given in (10.2), the corresponding target $\boldsymbol{\beta}_i$ for the "correct" model in (10.1) should be:

$$\boldsymbol{\beta}_{\text{true}} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\beta}_3, \boldsymbol{\beta}_4, \boldsymbol{\beta}_5, \boldsymbol{\beta}_6] =$$

$$\begin{bmatrix} -\gamma_1(=-0.3) & 0 & 0 & \beta_1(=1.4) & 0 & 0 \\ 0 & -\gamma_2(=-0.4) & 0 & 0 & \beta_2(=1.5) & 0 \\ 0 & 0 & -\gamma_3(=-0.5) & 0 & 0 & \beta_3(=1.6) \\ 0 & 0 & 0 & -\gamma_4(=-0.2) & 0 & 0 \\ 0 & 0 & 0 & 0 & -\gamma_5(=-0.4) & 0 \\ 0 & 0 & 0 & 0 & 0 & -\gamma_6(=-0.6) \\ \mathbf{0}_{47\times 1} & \mathbf{0}_{45\times 1} & \mathbf{0}_{46\times 1} & & & \\ \alpha_1(=4) & \alpha_2(=3) & \alpha_3(=5) & \mathbf{0}_{48\times 1} & \mathbf{0}_{48\times 1} & \mathbf{0}_{48\times 1} \\ \mathbf{0}_{0\times 1} & \mathbf{0}_{2\times 1} & \mathbf{0}_{1\times 1} & & & \end{bmatrix}. \tag{10.3}$$

with values in brackets indicating the correct parameter values.

To generate the time-series data, we took 'measurements' every 1 unit between $t = 0$ and $t = 50$ (arbitrary units) from random initial conditions which are drawn from a standard uniform distribution on the open interval $(0, 1)$. Thus a total of 51 measurements for each state are collected (including the initial value). It should be noted that the number of rows is less than the number of columns in the dictionary matrix.

We here investigate the performance of various algorithms including ours (Algorithm 1 in the main text) for different signal-to-noise ratios of the data. We define the signal-to-noise ratio (SNR) as

$$\text{SNR(dB)} \triangleq 20 \log_{10} \frac{\|\mathbf{X}\boldsymbol{\beta}_{\text{true}}\|_2}{\|\boldsymbol{\Xi}\|_2}.$$

We considered SNRs ranging from 0 dB to 25 dB for each generated weight. To compare the reconstruction accuracy of the various algorithms considered, we use the root of normalised mean square error (RNMSE) as a performance index, i.e.

$$\frac{\|\boldsymbol{\beta}_{\text{estimate}} - \boldsymbol{\beta}_{\text{true}}\|_2}{\|\boldsymbol{\beta}_{\text{true}}\|_2} \tag{10.4}$$

For each SNR, we performed 200 independent experiments and calculated the average RNMSE for each SNR over these 200 experiments. In each "experiment", we simulated the repressilator model with random initial conditions drawn from a standard uniform distribution on the open interval $(0, 1)$. The parameters were

drawn from a standard uniform distribution with the true values β_{true} in (10.3) taken as the mean and variations around the mean values no more than 10% of the true values. In `MATLAB`, one can use $\beta_{\texttt{true}}$`.*(0.9 + 0.2*rand(54,6))` to generate the corresponding parameter matrix for each experiment.

Based on these settings, we compared Algorithm 4 with nine other state-of-the-art sparse linear regression algorithms available at [54]. [54] provides access to a free `MATLAB` software package managed by David Donoho and his team and contains various tools for finding sparse solutions of linear systems, least-squares with sparsity, various pursuit algorithms, and more. We compare our identification algorithm with nine other state-of-the-art sparse linear regression algorithms available at [54], namely, BP (Basis Pursuit), IRWLS (Iteratively ReWeighted Least Squares), ISTBlock (Iterative Soft Thresholding, block variant with least squares projection), LARS (Least-Angle RegreSsion), MP (Matching Pursuit), OMP (Orthogonal Matching Pursuit), PFP (Polytope Faces Pursuit), Stepwise (Forward Stepwise), and StOMP (Stagewise Orthogonal Matching Pursuit).

In Fig. 10.1, we plot, for various SNRs, the average RNMSE obtained using our centralised algorithm (both implementation using the generic parser `CVX` and centralised ADMM implementations) and other algorithms in [54]. In Fig. 10.2, we plot, for the various SNRs considered, the average computational running time required by our algorithm and the other algorithms from [54]. During this comparison, the inputs for the algorithms listed in these algorithms are always the same, i.e. the dictionary matrix $\mathbf{X}$ and the data contained in $\mathbf{y}$. The initialisation and pre-specified parameters for these algorithms were set to their default values provided in [54]. Interested readers can download the package from [54] and reproduce the results presented here under the default settings of the solvers therein.

It should be noted that the dictionary matrices in all the experiments are rank deficient, i.e. neither column rank nor row rank are full. As a consequence, both the MP and OMP algorithm fail to converge or yield results with extremely large RNMSE. As these two algorithms cannot satisfactorily be used, they have been removed from the comparison results presented in Fig. 10.1 and Fig. 10.2. It can be seen from Fig. 10.1 that our algorithm outperforms all the other algorithms in [54] in terms of RNMSE.

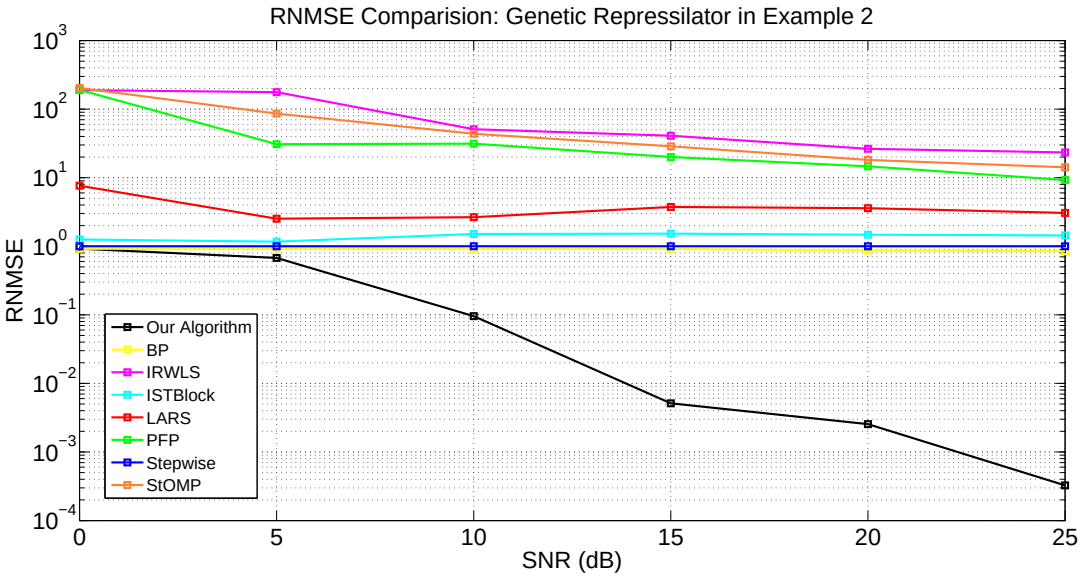


Fig. 10.1 Root of Normalised Mean Square Error $\|\beta_{\text{estimate}} - \beta_{\text{true}}\|_2 / \|\beta_{\text{true}}\|_2$ averaged over 200 independent experiments for the signal-to-noise ratios 0 dB, 5 dB, 10 dB, 15 dB, 20 dB, and 25 dB.

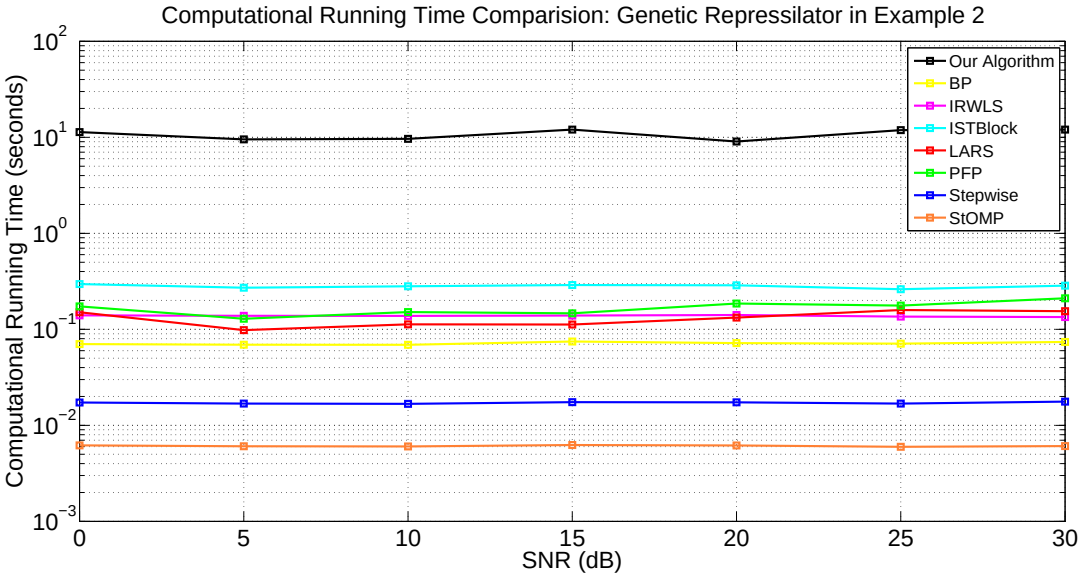


Fig. 10.2 Computational running time averaged over 200 independent experiments for the signal-to-noise ratios 0 dB, 5 dB, 10 dB, 15 dB, 20 dB, and 25 dB.

10.2 Identificaton from Multiple Heterogeneous Time Series Datasets

In this Section, we use numerical simulations to show the effectiveness of the proposed algorithm. To compare the identification accuracy of the various algorithms considered, we use the root of normalised mean square error (RNMSE) as a performance index, i.e.

$$\textbf{RNMSE} = \frac{\|\boldsymbol{\beta}_{\text{estimate}} - \boldsymbol{\beta}_{\text{true}}\|_2}{\|\boldsymbol{\beta}_{\text{true}}\|_2}.$$

Several factors affect the RNMSE, e.g. number of experiments C, measurement noise intensity, dynamic noise intensity, length of single time series data M, number of candidate basis functions N. For brevity of exposition, we only show results pertaining to change of RNMSE over number of experiment C and length of single time series for one experiment, all in the noiseless case. More results related to other factors that may affect RNMSE will be shown in a future journal publication presenting these results in more details.

As an illustrative example, we consider a model of an eight species generalised repressilator [181], which is a system where each of the species represses another species in a ring topology. The corresponding dynamic equations are as follows:

$$\begin{aligned} \dot{x}_{1t} &= \frac{p_{11}}{p_{12}^{p_{13}} + x_{8t}^{p_{13}}} + p_{14} - p_{15}x_{1t}, \\ \dot{x}_{it} &= \frac{p_{i1}}{p_{i2}^{p_{i3}} + x_{i-1,t}^{p_{i3}}} + p_{i4} - p_{i5}x_{it}, \ \forall i = 2, \dots 8, \end{aligned} \tag{10.5}$$

where p_{ij}, $i = 1, \dots, 8,\ j = 1, \dots, 5$. We assume the mean value for these parameters across different species and experiments are $\bar{p}_{i1} = 40$, $\bar{p}_{i2} = 1$, $\bar{p}_{i3} = 3$, $\bar{p}_{i4} = 0.5$, $\bar{p}_{i5} = 1$, $\forall i$. We simulate the ODEs in Eq. (10.5) to generate the time series data. In each "experiment" or simulation of Eq. (10.5), the initial conditions are randomly drawn from a standard uniform distribution on the open interval $(0, 1)$. The parameters in each experiment vary no more than 20% of the mean values. In `MATLAB`, one can use $\bar{p}_{ij}$`*(0.8 + 0.4*rand(1))` to generate the corresponding parameters for each experiment.

The numerical simulation procedure can be summarised as follows:

1. The deterministic system of ODEs (10.5) is solved numerically with an adaptive fourth-order Runge-Kutta method;

$$\mathbf{X} = \begin{bmatrix} x_1(1) & \dots & x_8(1) & \text{hill}(x_1(1),1,0,3) & \dots & \text{hill}(x_8(1),1,3,3) & 1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ x_1(M) & \dots & x_8(M) & \text{hill}(x_1(M),1,0,3) & \dots & \text{hill}(x_8(M),1,3,3) & 1 \end{bmatrix} \in \mathbb{R}^{M\times 25}. \tag{10.7}$$

2. As explained in (5.14), Gaussian measurement noise with variance σ^2 is added to the corresponding time-series data obtained in the previous step[1];

3. The data is re-sampled with uniform intervals[2];

4. The local polynomial regression framework in [48] is applied to estimate the first derivative;

5. A dictionary matrix is constructed;

6. Algorithm in Chapter 3 is used to identify the model.

The candidate dictionary matrix $\mathbf{X}$ in step 5) above is constructed by selecting as candidate nonlinear basis functions typically used to represent terms appearing in ODE models of Gene Regulatory Networks. As a proof of concept, we only consider Hill functions as potential nonlinear candidate functions. The set of Hill functions with Hill coefficient h, both in activating and repressing form, for the i^{th} state variables at time instant t_k are:

$$\text{hill}(x_i(t), K, h_{\text{num}}, h_{\text{den}}) \triangleq \frac{x_i^{h_{\text{num}}}(t)}{K^{h_{\text{den}}} + x_i^{h_{\text{den}}}(t)} \tag{10.6}$$

where h_{num} and h_{den} represent the Hill coefficients. When $h_{\text{num}} = 0$, the Hill function has a repression form, whereas an activation form is obtained for $h_{\text{num}} = h_{\text{den}} \neq 0$.

In our identification experiment, we assume h_{num}, h_{den} and K to be known. We are interested in identifying the regulation type (linear or Hill type, repression or activation) and the corresponding parameters p_{i1}, the degradation rate constant p_{i4} and the basal expression rate p_{i5}, $\forall i$. Since there are 8 state variables, we can construct the dictionary matrix $\mathbf{X}$ with 8 (basis functions for linear terms) $+(2*8)$ (basis functions for Hill functions, both repression and activation form) $+1$ (constant unit vector) $= 25$ columns. The corresponding matrix $\mathbf{X}$ is given in Eq. (10.10)

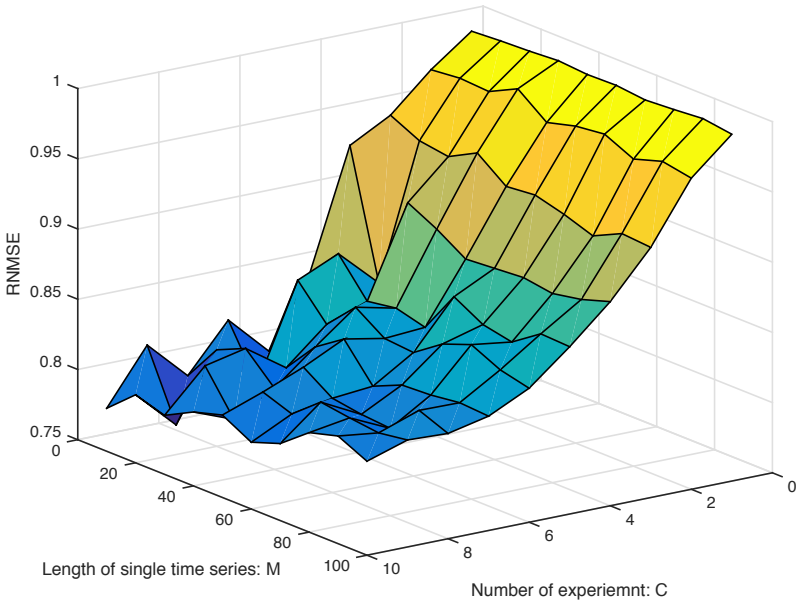
[1] In the example presented here, we consider the noiseless case corresponding to $\sigma = 0$.
[2] In this example, interval length is set to 1.

For a fixed number of experiments C and length of single time series M, we compute the RNMSE over 50 simulations by varying initial conditions and parameters p_{ij}. The RNMSE over C and M are shown in Fig. 10.3(a) and Fig. 10.3(b), using both group Lasso and Algorithm 5 with the maximal iteration number $k_{\max} = 5$ (see line 4 in Algorithm 5). Inspection of the results presented in Fig. 10.3(a) and Fig. 10.3(b) clearly show that Algorithm 5 outperforms significantly group Lasso in terms of RNMSE.

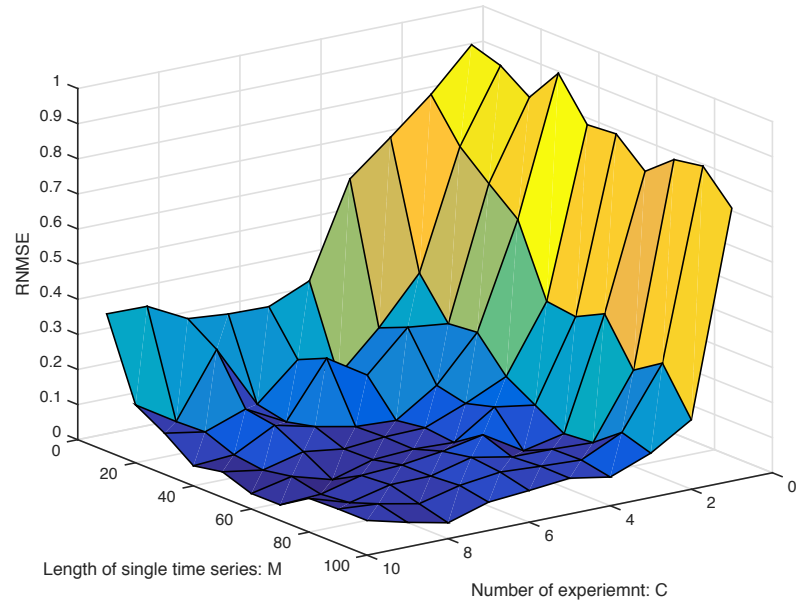
10.3 Online Model Selection

10.3.1 Background

Transcription and translation are two intrinsically slow processes (time scale of minutes in bacteria). While, on one hand, this implies there is no stringent need to observe cells with high sampling frequencies, it also means that identification/control experiments of biomolecular circuits usually last hours to days, i.e. much longer than similar experiments carried out on electrical or mechanical systems. Over such long time frames it is necessary to **(a)** effectively trap cells and **(b)** observe their internal dynamics. We also need to extract single cell trajectories while we **(c)** stimulate them with time-varying profiles of the molecules that serve as inducers for the network of interest. Most importantly it is necessary to achieve these objectives with minimally invasive techniques, i.e. using methods that ideally, will not affect the processes we want to quantify (a point not to be overlooked as factors like heat, e.g. generated by the light used to obtain microscopy images, or mechanical stresses, e.g. used to physically hold cells in place while imaging them, will trigger stress responses in cells). For these reasons we need to continuously **(i)** supply cells with nutrients and **(ii)** remove toxic metabolites while **(iii)** retaining the ability to condition their microenvironment to expose them to the appropriate externally applied stimuli. All these requirements, combined, significantly limit the technologies that can be used to identify and control biomolecular circuits ***in vivo***: for example commonly used methods, such as flask-based sampling or bioreactors, are unable to provide us with single cell trajectories. Microfluidics, enabling us to fabricate transparent microchannels where cells can be trapped and observed while being exposed to a continuous flow of nutrients and chemicals controlled by a computer, does allow to meet the requirements mentioned above.



(a) Group Lasso (first iteration of Algorithm 5). The minimal RNMSE is around 0.75



(b) Algorithm 5 with maximal iteration number $k_{\max} = 5$. The minimal RNMSE is almost 0.

Fig. 10.3 Algorithm comparison in terms of RNMSE $\|\boldsymbol{\beta}_{\text{estimate}} - \boldsymbol{\beta}_{\text{true}}\|_2 / \|\boldsymbol{\beta}_{\text{true}}\|_2$ averaged over 50 independent experiments.

In the view of the above, we will consider the setup documented in [124] as the reference platform for the ***in vivo*** implementation of our model selection approach. In this configuration, described in Fig 10.4, a microfluidic device containing the cells carrying the network of interest, is mounted on the stage of a fully automated microscope that takes phase contrast and fluorescence images of the cells at regular time intervals. Such images are used by the computer to locate cells (phase contrast) and estimate the amount of protein (fluorescence imaging) in real-time via a custom image processing algorithm developed in MATLAB. The computer, then, uses a set of fluidic pressure actuators to vary the level of inducer the cells are exposed to. Interestingly, this configuration allows us to continuously **(a)** update our model online and, potentially, **(b)** automtically carry out multiple model-selection iterations within the same experiment, a unique feature of this approach [163].

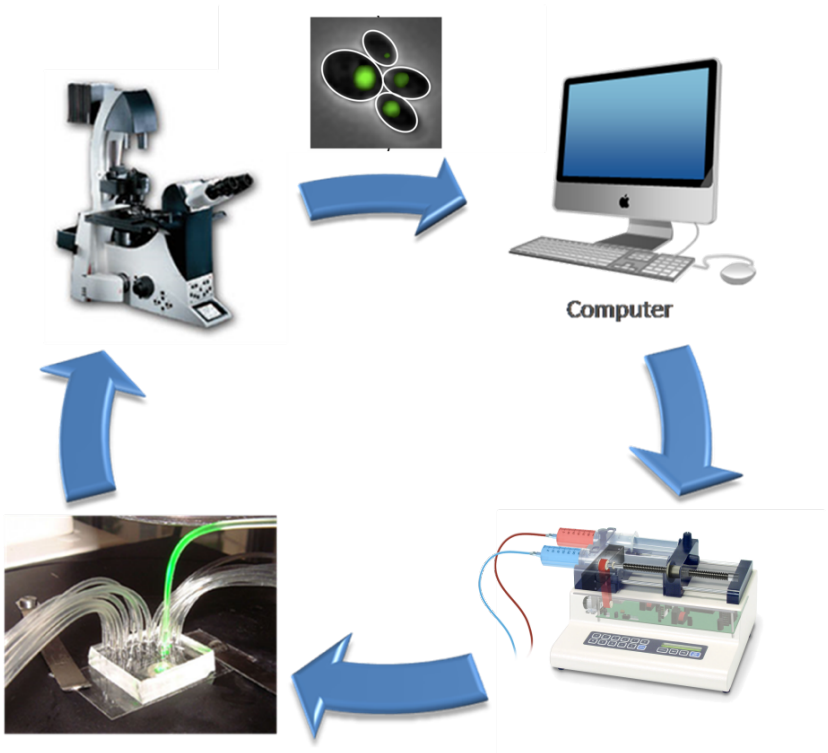


Fig. 10.4 **Technological platform for** *in-vivo* **model selection of synthetic circuits.** In this closed loop configuration the computer (upper right corner) takes images of the cells in the microfluidic device (lower left corner) via a microscope (upper left corner), quantifies the output of the network of interest in real time and applies the next sample of input(s) via the fluidic pressure actuation system (lower right corner).

In order to extend the experimental throughput and increase our model discrimination capabilities we will use the MDAW microfluidic device described in [59]: in this device 8 independent model selection experiments can be carried out at the same time. We will seed the same strain in each of the 8 chambers and image the 8 chambers at regular intervals. In so doing we will obtain 8 independent datasets (each formed by an exponentially growing number of single cell trajectories) that we will use to design and implement our model selection experiment.

Mathematical model

Throughout this Chapter, we will assume that the process of interest can be modelled by a discrete-time system of the form:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{g}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\beta}) + \boldsymbol{\xi}_t, \\ \mathbf{z}_t &= \mathbf{x}_t + \boldsymbol{\eta}_t, \end{aligned} \tag{10.8}$$

where the $\mathbf{x}_t = [x_{1,t}, \ldots, x_{n_{\mathbf{x}},t}] \in \mathbb{R}^{n_{\mathbf{x}}}$ is the state vector at discrete time point t; $\boldsymbol{\beta}$ represents the vector of parameters to be identified; the function $\mathbf{g} : \mathbb{R}^{n_{\mathbf{x}}} \times \mathbb{R}^{n_{\mathbf{u}}} \times \mathbb{R}^{n_\beta} \to \mathbb{R}^{n_{\mathbf{x}}}$ is nonlinear and depends (explicitly) on the input vector $\mathbf{u}_t \in \mathbb{R}^{n_{\mathbf{u}}}$. The process noise $\boldsymbol{\xi}_t \in \mathbb{R}^{n_{\mathbf{x}}}$ and measurement noise $\boldsymbol{\eta}_t \in \mathbb{R}^{n_{\mathbf{x}}}$ are assumed to be mutually independent Gaussian random variables with known positive covariance matrices $\mathbf{Q}_t$ and $\mathbf{R}_t$, respectively.

The state vector $\mathbf{x}_t$ usually contains concentrations of certain chemical species of interest, such as mRNAs or proteins. The output signal $\mathbf{z}_t$ represents the quantities we can measure experimentally.

10.3.2 Questions of interest

1. Estimation of the model structure, i.e. the functional structure of $\mathbf{g}_n(\cdot)$ in (10.8).

2. Estimation the parameter vector $\boldsymbol{\beta}$ therein.

3. Identification of a single model from multiple datasets emanating from perturbation experiments performed on systems of the form given in (10.8) that differ in terms of their parameters but not their parametric structure.

$$\mathbf{X} = \begin{bmatrix} x_{11} & \dots & x_{81} & \text{hill}(x_{11}, 0.5, 0, 2) & \dots & \text{hill}(x_{81}, 1.5, 0, 3) & \text{hill}(x_{81}, 1.5, 3, 3) & 1 \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1M} & \dots & x_{8M} & \text{hill}(x_{1M}, 0.5, 0, 2) & \dots & \text{hill}(x_{8M}, 1.5, 0, 3) & \text{hill}(x_{8M}, 1.5, 3, 3) & 1 \end{bmatrix} \in \mathbb{R}^{M \times 73}. \tag{10.10}$$

For example, in the ideal noiseless case, a simple self-induction gene network can be described as [169]:

$$\begin{aligned} \frac{dx_t}{dt} &= -kx_t + \frac{V_{\max} x_t^h}{K_M + x_t^h}, \\ z_t &= x_t. \end{aligned} \tag{10.9}$$

where $\frac{dx_t}{dt}$ is a numerical estimation of the time derivative of x_t (see appendix of [140] for details), k is the decay rate of gene product x, $V_{\max}$ is the maximum gene expression rate, K_M is the threshold value in terms of the concentration of gene product x that results in a production rate of $0.5V_{\max}$, and h is the Hill coefficient (a.k.a. the cooperativity coefficient) associated with the self-induction of gene x.

The identification problem can be formulated as: given some time series data corresponding to discrete time point measurements of the gene product, i.e., $z_1, \cdots, z_{M+1}$, can the model given in (10.9) be identified or approximated?

10.3.3 Simulations

We use the example in the previous Section 10.2 with the same parameters and initial condition settings for both model and Algorithm 5.

We are interested in identifying the regulation type (linear or Hill type, repression or activation) and the corresponding parameters p_{i1}, the basal expression rate p_{i4} and the degradation rate constant p_{i5}, as well as $K_i, \forall i$. Since there are 8 state variables, we can construct the dictionary matrix $\mathbf{X}$ with 8 (basis functions for linear terms) $+(8 * 8)$ (8 Hill functions with $K_i \in \{0.5, 1.5\}$ and $h_{num}, h_{den} \in \{2, 3\}$, both repression and activation form) $+1$ (constant unit vector) $= 73$ columns. The corresponding matrix $\mathbf{X}$ is given in Eq. (10.10). Note that none of the Hill functions in the set of dictionary functions has a value of K_i equal to 1.

To quantify the identification accuracy of the algorithm, we use the root of normalised mean square error (RNMSE) as a performance index, i.e.

$$\textbf{RNMSE} = \frac{\|\bar{\boldsymbol{\beta}}_{\text{estimate}} - \bar{\boldsymbol{\beta}}_{\text{true}}\|_2}{\|\bar{\boldsymbol{\beta}}_{\text{true}}\|_2}$$

where $\bar{\boldsymbol{\beta}}_{\text{true}}$ (resp. $\bar{\boldsymbol{\beta}}_{\text{estimate}}$) represents the average of the C parameters values (resp. the average of the C identified parameters values). Similarly to what we showed in Fig. 1 of [140], we observe that a larger number of experiments C or a larger length of single time series data M leads to a smaller RNMSE value.[3] In our simulation, we take $C = 10$ and $M = 100$. The corresponding RNMSE for the application of Algorithm 5 to the identification of model (10.5) is RNMSE = 0.047 when 50 independent experiments are considered.

Since the identification procedure for each state variable is independent, we only focus on the identification of the dynamics $\dot{x}_1$. Similar results are obtained for the identification of the other equations Both the linear term x_1 and the constant term can be identified with an average parameter estimation value of $\bar{p}_{14} = 0.501 \approx 0.5$ and $\bar{p}_{15} = 1.07 \approx 1$.

In our result, both dictionary functions $\frac{1}{0.5+x_8^3(t)}$ and $\frac{1}{1.5+x_8^3(t)}$ are selected by Algorithm 5 to be part of the dynamics of $dx_1(t)/dt$, and the average of the corresponding parameters over $C = 10$ experiments are 8 and 35.7, respectively. This means that $\frac{40}{1+x_8^3(t)}$ can be approximated by $\frac{8}{0.5+x_8^3(t)} + \frac{35.7}{1.5+x_8^3(t)}$. The corresponding fitting result can be found in Figure 10.5.

Next we turn to model refinement ("Filter" in Algorithm 18) using an Unscented Kalman filter [200]. For this, we consider the following equation for dx_1/dt:

$$\frac{dx_1(t)}{dt} = \frac{\gamma_1}{\gamma_2 + x_8^3(t)} + \gamma_3 - \gamma_4 x_1(t). \tag{10.11}$$

where $\frac{dx_1(t)}{dt}$ is a numerical estimation of the time derivative of $x_1(t)$. where the new parametric structure $\frac{\gamma_1}{\gamma_2+x_8^3(t)}$ has been used to replace the term $\frac{8}{0.5+x_8^3(t)} + \frac{35.7}{1.5+x_8^3(t)}$ that was identified by Algorithm 5. Fig. 10.6 shows the evolution of the estimated values of the parameters in equation (10.11) as a function of the number of streaming data iterations of the Unscented Kalman Filter.

[3] The RNMSE values for varying values of C and M are not shown here due to space limitation.

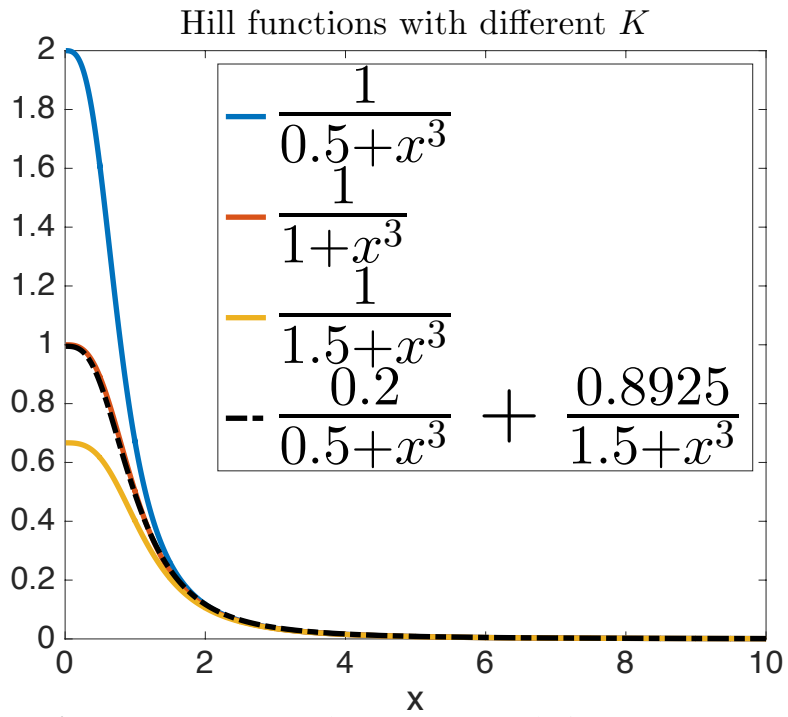


Fig. 10.5 After refinement iterations (see Figure 10.6) the parameters γ_1 and γ_2 of the new structure $\frac{\gamma_1}{\gamma_2+x_8^3}$ selected to replace $\frac{8}{0.5+x_8^3} + \frac{35.7}{1.5+x_8^3}$ (identified from Algorithm 5) are estimated to be $\gamma_1 = 39.99$ and $\gamma_2 = 0.9998$, respectively.

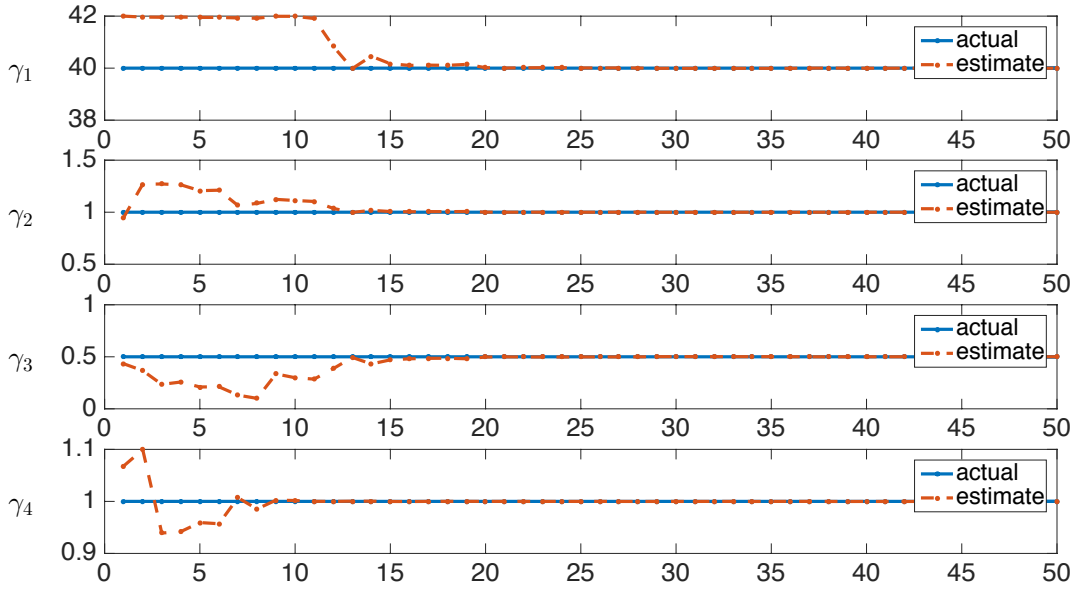


Fig. 10.6 Evolution of the estimated values of the parameters in equation (10.11) as a function of the number of streaming data iterations of the Unscented Kalman Filter.

10.4 Identificaton Switched Biochemical Reation Networks

Consider a biochemical system with n species $X_1, \ldots, X_n$. We denote the concentration of species X_j as x_j. Let $\mathcal{U}$ be the set of uni-species reactions and $\mathcal{B}$ be the set of bi-species reactions. A uni-species reaction $i \in \mathcal{U}$ is defined by the index $r_i \in \{1, \ldots, n\}$ of its single reactant species, the associated real-valued rate constant $k_i > 0$, and the integer product coefficients for each species $c_{i,j} \geq 0$: $m_i X_{r_i} \xrightarrow{k_i} c_{i,1}X_1 + \ldots + c_{i,n}X_n$. A bi-species reaction $i \in \mathcal{B}$ is defined by the indices $r_{i,1}, r_{i,2} \in \{1, \ldots, n\}$ of its two reactant species, the real-valued rate constant $k_i > 0$, and the integer product coefficients for each species $c_{i,j} \geq 0$: $m_i X_{r_{i,1}} + n_i X_{r_{i,2}} \xrightarrow{k_i} c_{i,1}X_1 + \ldots + c_{i,n}X_n$. Using the law of mass action, the dynamics of the concentrations $x_j \geq 0$ of species X_j are given according to the ordinary differential equations

$$\begin{aligned}\dot{x}_j = &- \sum_{i \in \mathcal{U} | r_i = j} k_i x_j^{m_i} - \sum_{i \in \mathcal{B} | r_{i,1} = j} k_i x_j^{m_i} x_{r_{i,2}}^{n_i} - \sum_{i \in \mathcal{B} | r_{i,2} = j} k_i x_{r_{i,1}}^{m_i} x_j^{n_i} \\ &+ \sum_{i \in \mathcal{U}} c_{i,j} k_i x_{r_i} + \sum_{i \in \mathcal{B}} c_{i,j} k_i x_{r_{i,1}} x_{r_{i,2}}.\end{aligned} \tag{10.12}$$

We can expand (10.12) for more than two species, though this can be rarely found in reality due to highly improbable simultaneous three-species molecular collision mechanisms.

Eq. (10.12) can be modelled using the general form: $\dot{\mathbf{x}} = \mathbf{S}\mathbf{v}(\mathbf{x})$, where $\mathbf{x}$ is the vector of species whose elements are x_j, $\mathbf{S}$ is the stoichiometry matrix and $\mathbf{v}(\mathbf{x})$ is a vector of propensity functions. The matrix $\mathbf{S}$ and the propensity vector $\mathbf{v}(\mathbf{x})$ can be built based on the biochemical reactions and their rates. Hence, without loss of generality we can assume that $\mathbf{S}$ is a matrix whose elements are real constants and $\mathbf{v}(\mathbf{x})$ is a vector whose elements are nonlinear functions of $\mathbf{x}$ as in (10.12). Biochemical processes can go through different phases in time; for example, a cell cycle in bacteria or diurnal alternations in plants. These switches, which are typically triggered by time dependent processes or by some external force, can be fitted into our model as follows: $\dot{\mathbf{x}} = \mathbf{S}^{\alpha(t)}\mathbf{v}(\mathbf{x})$, where $\alpha(t)$ is a sequence of integers in a bounded set and $\mathbf{S}^{\alpha(t)}$ takes values from an unknown set $\{\mathbf{S}_1, \ldots, \mathbf{S}_{N_{modes}}\}$ depending on time.

In what follows, we consider the system dynamics expressed in discrete-time and subjected to additive i.i.d. Gaussian noise $\xi(k)$ with known statistics.

$$\mathbf{x}(k+1) = \mathbf{S}^{\alpha(k)}\mathbf{v}(\mathbf{x}(k)) + \xi(k). \tag{10.13}$$

We consider time-series data obtained from a chaotic Lorenz Oscillator implemented *in vitro* using DNA computations [176]. From the associated biochemical reactions, a polynomial ODE can be derived using the law of mass action. We artificially generate data using this oscillator model but change certain parameters at certain time. This can be realised *in vitro* by changing experiment conditions or enzyme concentrations. The Lorenz oscillator can be described by the discretised differential equations

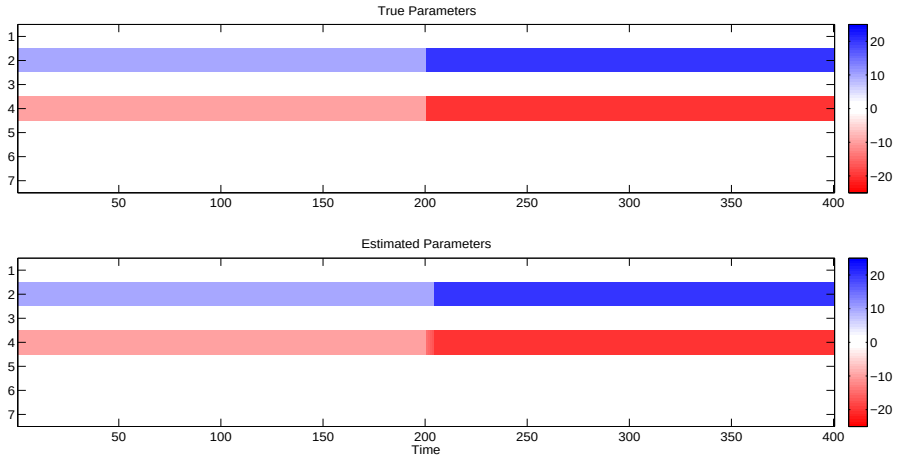
$$\begin{aligned}&\left[\frac{y_1(k+1)-y_1(k)}{\delta t}, \frac{y_2(k+1)-y_2(k)}{\delta t}, \frac{y_3(k+1)-y_3(k)}{\delta t}\right]\\ =&\left[p_1(k)(y_2(k)-y_1(k)), y_1(k)(p_2(k)-y_2(k)), y_1(k)y_2(k)-k_2(k)y_3(k)\right].\end{aligned}$$

where we the fix sampling time to $\delta t = 0.02$ (arbitrary units).

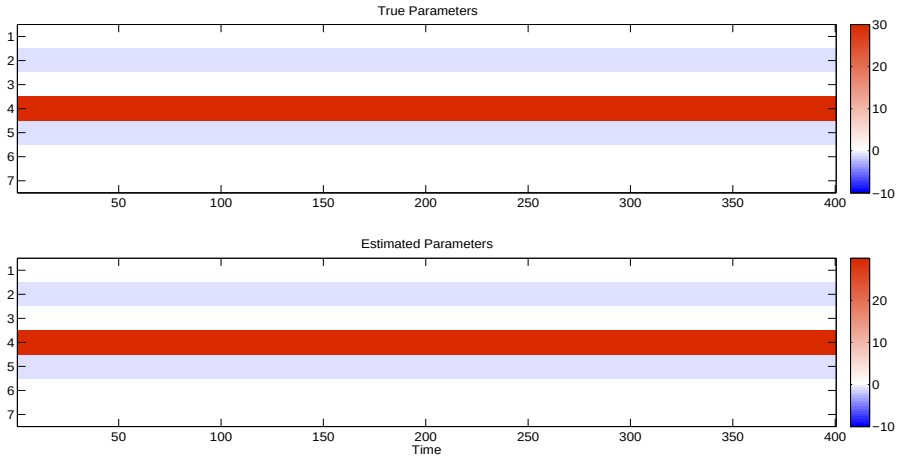
Initially ("Mode 1"), the parameters are $p_1 = 10, p_2 = 30, p_3 = 8/3$. From $k = 201$ to $k = 400$ ("Mode 2"), the parameters are changed to $p_1 = 10, p_2 = 30, p_3 = 4$. For the kinetics of y_1 and y_3, the nonlinear dynamics change after switching from Mode 1 to Mode 2. For y_2, the parameters do not switch. We construct the basis functions as

$$\left(y_1^0(k)y_2^0(k)y_3^0(k), \quad y_1^0(k)y_2^0(k)y_3^1(k), \quad \ldots, \quad y_1^{n_1}y_2^{n_2}(k)y_3^{n_3}(k)\right).$$

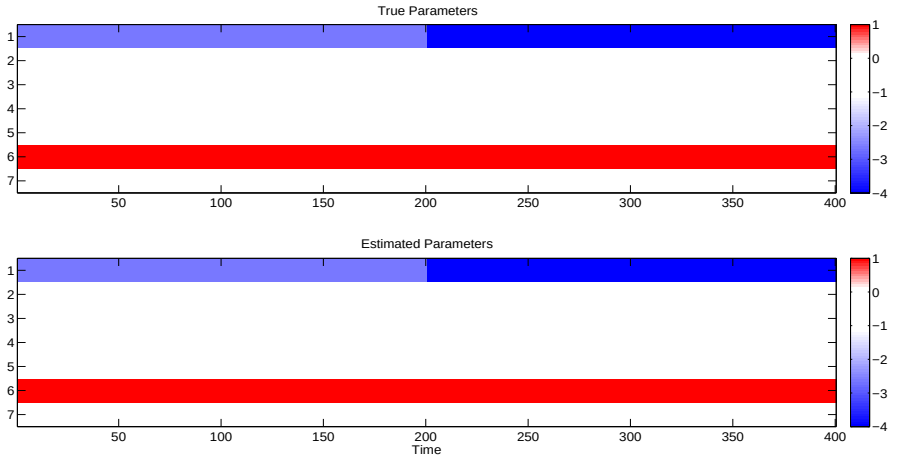
We index the parameter vector $\boldsymbol{\beta}(k)$ as $[w^{000}(k), w^{001}(k), \ldots, w^{n_1n_2n_3}(k)]$, choose $\lambda = 1$ and $\rho = 100$ and set the initial condition to $[y_1(1), y_2(1), y_3(1)] = [0.2444, -2.217, 2.314]$. Finally, we set $n_1 = 1$, $n_2 = 1$ and $n_3 = 1$. The true and estimated parameters' evolution over time are shown in Figure10.7.



(a) True and estimated parameters for y_1.



(b) True and estimated parameters for y_2.



(c) True and estimated parameters for y_3.

Fig. 10.7 True (upper panel) and estimated (lower panel) parameters' evolution over time. The horizontal axis represents time, whereas the vertical axis represents the estimated coefficients. From top to bottom, the index goes from 001 to 111.

Chapter 11

Complex Network Reconstruction

11.1 Centralised Identification

A classical example in physics, engineering and biology is the Kuramoto oscillator network [182]. We consider a network where the Kuramoto oscillators are nonidentical (each has its own natural oscillation frequency ω_i) and the coupling strengths between nodes are not the same. The corresponding discrete-time dynamics can be described by

$$\frac{\phi_i(t+1)-\phi_i(t)}{\Delta t}=\omega_i+\sum_{j=1,j\neq i}^{n} w_{ij}g_{ij}(\phi_j(k)-\phi_i(k))+\xi_i(k),\quad i=1,\ldots,n, \tag{11.1}$$

where $\phi_i \in [0, 2\pi)$ is the phase of oscillator i, ω_i is its natural frequency, and the coupling function g_{ij} is a continuous and smooth function, usually taken as $\sin,\ \forall i,j$. w_{ij} represent the coupling strength between oscillators i and j thus $[w_{ij}]_{n\times n}$ defines the topology of the network. Here, assuming we don't know the exact form of g_{ij}, we reconstruct from time-series data of the individual phases ϕ_i a dynamical network consisting of n Kuramoto oscillators, i.e., we identify the coupling functions $g_{ij}(\cdot)$ as well as the model parameters, i.e., ω_i and w_{ij}, $i,j=1,\ldots,n$.

To define the dictionary matrix $\mathbf{X}$, we assume that all the dictionary functions are functions of a pair of state variables only and consider 5 candidate coupling functions g_{ij}: $\sin(x_j-x_i)$, $\cos(x_j-x_i)$, x_j-x_i, $\sin^2(x_j-x_i)$, and $\cos^2(x_j-x_i)$. Based on this, we define the dictionary matrix as

$$\begin{aligned}\mathbf{X}_{ij}(x_j(k),x_i(k)) \triangleq &[\sin(x_j(k)-x_i(k)),\cos(x_j(k)-x_i(k)),x_j(k)-x_i(k),\\ &\sin^2(x_j(k)-x_i(k)),\cos^2(x_j(k)-x_i(k))]\in\mathbb{R}^5.\end{aligned}$$

To also take into account the natural frequencies, we add to the last column of $\mathbf{X}_i$ a unit vector. This leads to the following dictionary matrix $\mathbf{X}_i$:

$$\mathbf{X}_i \triangleq \begin{bmatrix} \mathbf{X}_{i1}(x_1(1),x_i(1)) & \ldots & \mathbf{X}_{in}(x_n(1),x_i(1)) & 1\\ \vdots & \vdots & \vdots & \vdots\\ \mathbf{X}_{i1}(x_1(M),x_i(M)) & \ldots & \mathbf{X}_{in}(x_n(M),x_i(M)) & 1\end{bmatrix}\in\mathbb{R}^{M\times(5n+1)}.$$

Then the output can be defined as

$$\mathbf{y}_i \triangleq \left[\frac{\phi_i(2)-\phi_i(1)}{\Delta t},\ldots,\frac{\phi_i(M+1)-\phi_i(M)}{\Delta t}\right]^\top\in\mathbb{R}^{M\times 1},\quad i=1,\ldots,n.$$

To generate the time-series data, we simulated a Kuramoto network with $n = 100$ oscillators, for which 10% of the non-diagonal entries of the weight matrix $[w_{ij}]_{n\times n}$ are nonzero (assuming g_{ii} and w_{ii} are zeros), and the non-zero w_{ij} values are drawn from a standard uniform distribution on the interval $[-10, 10]$. The natural frequencies ω_i are drawn from a normal distribution with mean 0 and variance 10. In order to create simulated data, we simulated the discrete-time model (11.1) and took 'measurements data points' every $\Delta t = 0.1$ between $t = 0$ and $t = 45$ (in arbitrary units) from random initial conditions drawn from a standard uniform distribution on the open interval $(0, 2\pi)$. Thus a total of 451 measurements for each oscillator phase $\phi_i \in \mathbb{R}^{451\times 501}$ are collected (including the initial value). Once again, it should be noted that the the number of rows of the dictionary matrix is less than that of columns.

We here investigate the performance of various algorithms including ours (Algorithm 4 in the main text) for different signal-to-noise ratios of the data. We define the signal-to-noise ratio (SNR) as $\mathrm{SNR(dB)} \triangleq 20\log_{10}(\|\mathbf{X}\boldsymbol{\beta}_{\mathrm{true}}\|_2/\|\boldsymbol{\Xi}\|_2)$. We considered SNRs ranging from 0 dB to 25 dB for each generated weight. To compare the reconstruction accuracy of the various algorithms considered, we use the root of normalised mean square error (RNMSE) as a performance index, i.e.

$$\frac{\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2}{\|\boldsymbol{\beta}\|_2}$$

, where $\hat{\boldsymbol{\beta}}$ is the estimate of the true weight $\boldsymbol{\beta}$. For each SNR, we performed 200 independent experiments and calculated the average RNMSE for each SNR over these 200 experiments. In each "experiment", we simulated a Kuramoto network with $n = 100$ oscillators, for which 10% of the non-diagonal entries of the weight matrix $[w_{ij}]_{n\times n}$ were nonzero (assuming g_{ii} and w_{ii} are always zero). The non-zero w_{ij} values were drawn from a standard uniform distribution on the interval $[-10, 10]$. The natural frequencies ω_i were drawn from a normal distribution with mean 0 and variance 10.

Based on these settings, we compared Algorithm 4 with nine other state-of-the-art sparse linear regression algorithms available at [54]. [54] provides access to a free `MATLAB` software package managed by David Donoho and his team and contains various tools for finding sparse solutions of linear systems, least-squares with sparsity, various pursuit algorithms, and more. We compare our identification algorithm with nine other state-of-the-art sparse linear regression algorithms available at [54], namely, BP (Basis Pursuit), IRWLS (Iteratively ReWeighted Least

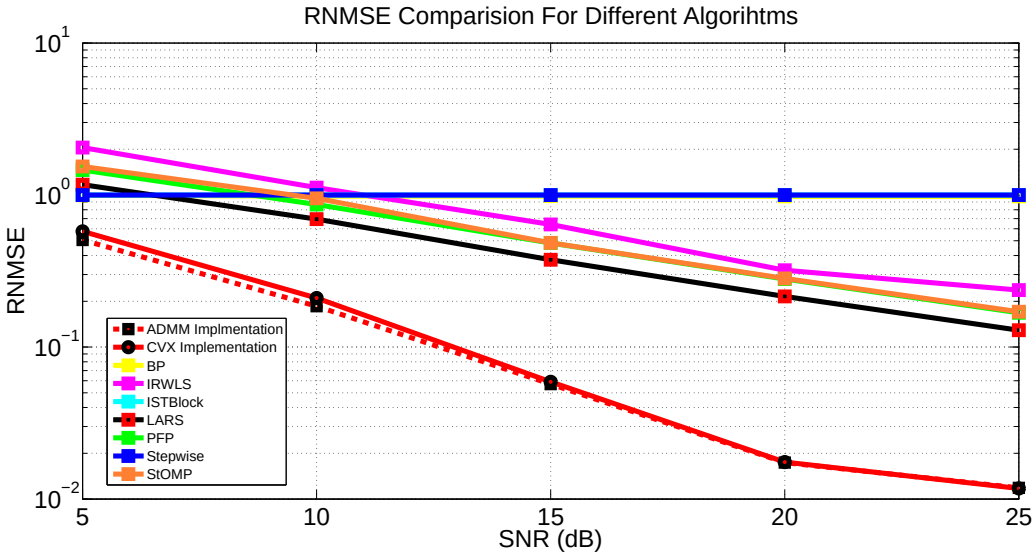
Squares), ISTBlock (Iterative Soft Thresholding, block variant with least squares projection), LARS (Least-Angle RegreSsion), MP (Matching Pursuit), OMP (Orthogonal Matching Pursuit), PFP (Polytope Faces Pursuit), Stepwise (Forward Stepwise), and StOMP (Stagewise Orthogonal Matching Pursuit).

In Fig. 11.1(a), we plot, for various SNRs, the average RNMSE obtained using our centralised algorithm (both implementation using the generic parser `CVX` and centralised ADMM implementations) and other algorithms in [54]. In Fig. 11.1(b), we plot, for the various SNRs considered, the average computational running time required by our algorithm and the other algorithms from [54]. During this comparison, the inputs for the algorithms listed in these algorithms are always the same, i.e. the dictionary matrix $\mathbf{X}$ and the data contained in $\mathbf{y}$. The initialisation and prespecified parameters for these algorithms were set to their default values provided in [54]. Interested readers can download the package from [54] and reproduce the results presented here under the default settings of the solvers therein.

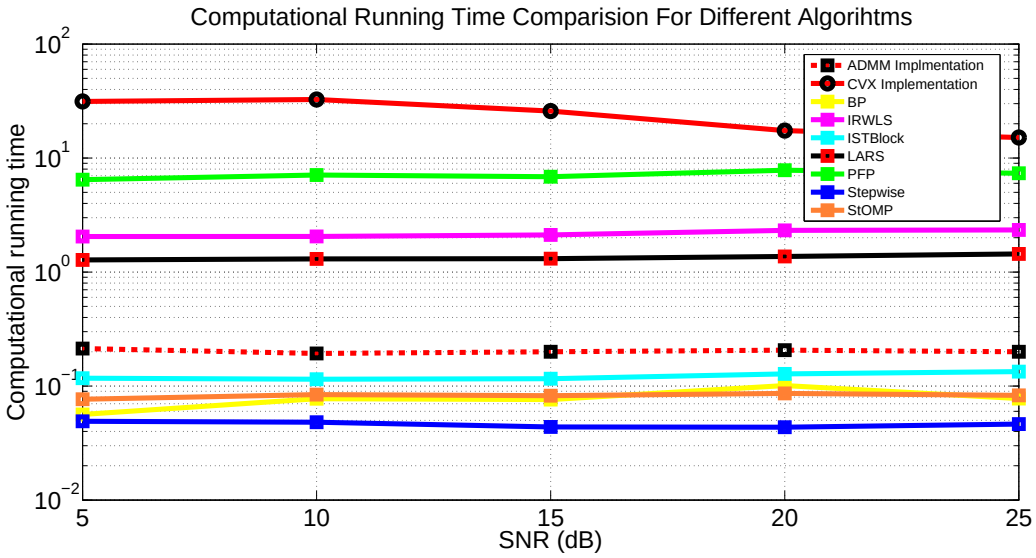
It should be noted that the dictionary matrices in all the experiments are rank deficient, i.e. neither column rank nor row rank are full. As a consequence, both the MP and OMP algorithm fail to converge or yield results with extremely large RNMSE. As these two algorithms cannot satisfactorily be used, they have been removed from the comparison results presented in Figures 11.1(a) to Figures 11.1(b). It can be seen from Figure 11.1 that our algorithm outperforms all the other algorithms in [54] in terms of RNMSE. The `CVX` implementation requires more computational running time compared to all other algorithms. However, the ADMM implementation is competitive in both accuracy and speed.

Discussion

It can be seen from Fig. 11.1 that our algorithm outperforms all the other algorithms in [54] in terms of RNMSE. However, the implementation using `CVX` requires more computational running time compared to all other algorithms. There are potentially two reasons for this. The first one is that our algorithm is implemented using the `CVX` package as a parser [69]. Parsers similar to `CVX` also include `YALMIP` [119]. `CVX` and `YALMIP` call generic SDP solvers, e.g. `SDPT3` [193] or `SeDuMi` [183], to solve the convex optimisation problem at hand (we use `SeDuMi`). While these solvers are reliable for wide classes of optimisation problems, they are not specifically optimised in terms of algorithmic complexity to exploit the specific structure of particular problems, such as ours. The second reason comes from the 5^{th} step of



(a) Parameter RNMSE $\|\beta_{\text{estimate}} - \beta_{\text{true}}\|_2 / \|\beta_{\text{true}}\|_2$ over various SNRs.



(b) Computational running time (in seconds) over various SNRs.

Fig. 11.1 Parameter RNMSE $\|\beta_{\text{estimate}} - \beta_{\text{true}}\|_2 / \|\beta_{\text{true}}\|_2$ and computational running time averaged over 200 independent experiments for the signal-to-noise ratios 0 dB, 5 dB, 10 dB, 15 dB, 20 dB, and 25 dB.

Algorithm 4 where the $M \times N$ matrix $\sigma^2\mathbf{I} + \mathbf{X}\mathbf{U}^{(k)}\mathbf{W}^{(k)}\mathbf{X}^\top$ has to be inverted to update the weights for the next iteration. Though a pruning rule has been discussed, such inversion at each iteration is inevitable compared to the algorithms considered in [54].

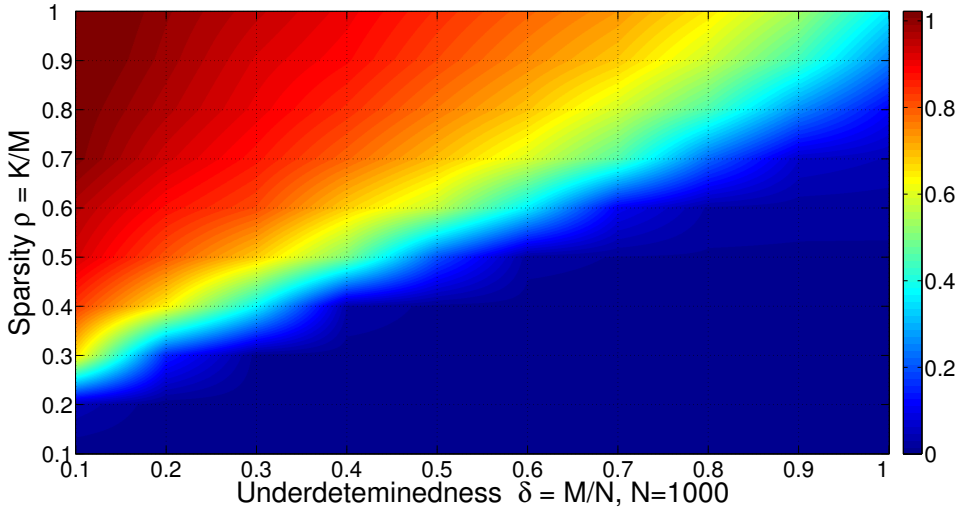
11.2 Distributed Identification

For all the following examples dealt with using ADMM implementations, we set the penalty parameter in the augmented Lagrangian $\rho = 1$ and the scalar regularisation parameter $\lambda = 0.001\|\mathbf{X}^T\mathbf{y}\|_\infty$. We also considered termination tolerances $\epsilon^{abs} = 10^{-4}$ and $\epsilon^{abs} = 10^{-2}$ and set the ADMM iteration number to be 200 and the weight updating iteration number to be 5 throughout all algorithms. We set the threshold for pruning $\boldsymbol{\gamma}$ to be 10^{-3}.

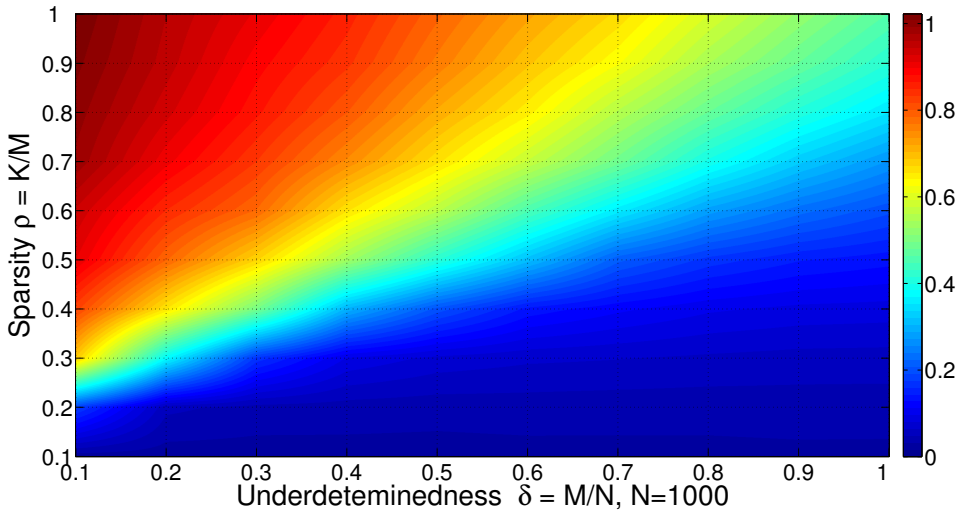
Now we introduce another performance index **Phase Diagram.** A useful performance index is the *Phase Diagram* [Donoho and Stodden], which is used to illustrate how sparsity levels (defined as $\rho = K/M$, where K counts the number of the nonzero elements in $\boldsymbol{\beta}$) and underdeteminedness ($\delta = M/N$) affect the performance of an algorithm. It should be noted that the phase diagrams presented here are empirical. More on phase diagrams, including the results, will be presented later.

Next we investigate the relation between the underdeteminedness $\delta = M/N$ and the sparsity level $\rho = K/M$, i.e. the phase diagram. We use the same specifications of the dictionary matrix as above. To recap, sine functions are used as candidate functions to construct the dictionary vector as $\mathbf{X}_{ij}(k) \triangleq [\sin(x_j(k) - x_i(k))] \in \mathbb{R}$. We assume the natural frequencies θ_i to be zero for all i. Therefore, the column size of the dictionary matrix $\mathbf{X}_i$ equals to the dimension of the network, i.e. $N = n$. However, The network topology is generated differently from the centralised identification. Here, we change the nonzero entries proportion from 10% to K/N.

In Figure 11.2, show the parameter RNMSE in the noiseless case for both Algorithm 4 and direct Lasso reconstruction. From Figure 11.2, we can clearly find a "cliff" or "breakdown" from dark blue to light blue area. Such cliff occurs for values of the RNMSE around 0.5. By Lasso reconstruction we mean one single iteration of Algorithm 4. We varied the underdeteminedness $\delta = M/N$ from 0.05 to 1 with $N = 1000$. The dark blue area, below the cliff, indicates the region within which the algorithm recovered the underlying model with near zero error. Above the cliff in the coloured area, the algorithm was unable to recover the correct model. As one



(a) Parameter RNMSE for Algorithm 4.



(b) Parameter RNMSE for Lasso reconstruction.

Fig. 11.2 Phase diagrams for Algorithm 4 (left panel) and Lasso reconstruction (right panel) in the noiseless case. The number of measurements required to correctly reconstruct the network with Algorithm 4 is significantly lower in comparison with Lasso.

proceeds further above the cliff, the ability of the algorithm to recover the model progressively drops. Each colour indicates a different RNMSE over 50 realisations. Considering both subfigures in Figure 11.2, we can see that each of the phase diagrams shows a transition from accurate model recovery to very inaccurate as the underlying complexity of the model increases. The "breakdown" in the error level shows where the algorithm stops recovering the correct underlying model. Note how increased model noise drives the breakdown point lower, at sparser models. The noise level also limits the accuracy, even when the correct model type is recovered, as the coefficients are estimated with more noise when the noise in the underlying data increases [Donoho and Stodden]. Carrying out additional tests for various network sizes N (results not shown), surprisingly, we observed that the corresponding phase diagrams remain very similar and exhibit the same trends. This implies that, if the sparsity K can be estimated based on expert knowledge, a rough estimate of the number of observations needed to guarantee an *a priori* defined reconstruction performance can be chosen. In practice, K tends to be very small while N is usually large due to the need to *a priori* consider a large number of candidate nonlinear dictionary functions. From the phase diagrams, it also can be implied that the more candidate functions we introduce, the more observations we need when sparsity is fixed.

Based on the above, it is informative to revisit the comparison where $N = 501$. The underdeterminess $M/N = 450/501 \approx 0.9$ and the sparsity level $K/M \approx 10/450 \approx 0.02$. Regarding various SNRs, reconstruction for data with high SNRs yields better performance. This is congruent with the results presented in the phase diagrams of Figure 11.2.

Finally, it is interesting to compare the results obtained through direct Lasso reconstruction (the first iteration of Algorithm 4) and Algorithm 4. Figure 11.2 validates that Algorithm 4 delivers solutions with higher sparsity level in comparison with Lasso reconstruction. Moreover, the number of measurements required to correctly identify the structure in the noiseless case is much lower for Algorithm 4 in comparison with Lasso. This is a major benefit when identifying systems with low sampling rate and low number of measurements. In the presence of noise it is hard to quantify the performance criteria for correct reconstruction, since a larger number of models can fit the data. However, reconstructed models obtained using Algorithm 4 are typically sparser in comparison with models obtained using Lasso.

Next, we compare the computational time for different network sizes by distributing the computations over various numbers of CPU cores. In this comparison,

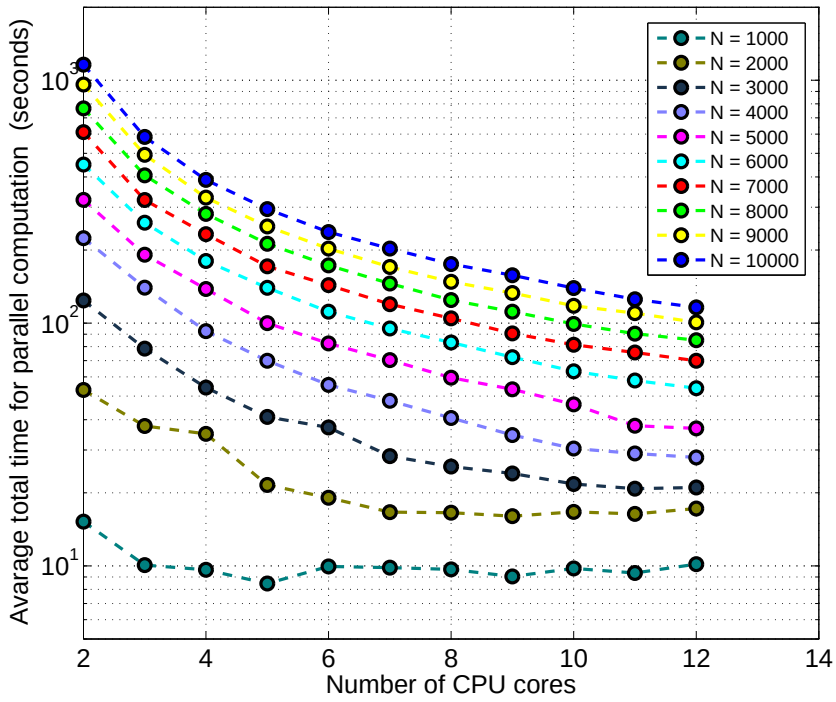


Fig. 11.3 Average computational running time for the β_i^{t+1} update in Algorithm 4 for a fixed sparsity $K = 10$ when different numbers of CPU cores and different network dimensions ranging from $N = 1000$ to $N = 10000$, are considered. Units are in seconds.

we consider only sine functions as candidate functions to construct the dictionary element, i.e. $\mathbf{X}_{ij}(k) \triangleq [\sin(x_j(k) - x_i(k))] \in \mathbb{R}$. We assume the natural frequencies θ_i to be zero for all i. Therefore, the column size of the dictionary matrix $\mathbf{X}_i$ equals the dimension of the network, i.e. $N = n$. Here, N is ranging from 1000 to 10000 and the sparsity K is fixed to 10. For each N, time-series are generated in the same manner as above, while M is fixed at 500. In the implementation of the distributed algorithm, we use the `MATLAB` command `parfor` and `matlabpool('size')` to parallelise the β_k-update in Algorithm 4. The parameter `size` is varied from 2 to 12. Experiments over the five SNRs (5 dB, 10 dB, 15 dB, 20 dB, 25 dB) are performed for each N. From the results we found that the computation time over each SNR varied slightly (at least within the same magnitude), which is consistent with the simulation results presented in Figure 11.1. In Figure 11.3, we show the average computational running time for different number of CPU cores used and for different network dimensions ranging from 1000 to 10000. Each point in the figure is obtained by averaging over 5×50 experiments for each network size.

Discussion

One of the major drawbacks of centralised algorithm is their high computational complexity in comparison with other approaches. To alleviate this drawback, we have derived a decentralised version of the algorithm, which can be distributed between several computational units. We illustrate the effectiveness of our decentralisation algorithm by distributing the computation between several computer cores and consequently showed an almost a log-linear gain in computational time for large-scale systems.

The presented algorithm has a better performance in terms of Normalised Mean Square Error in comparison with state-of-the-art-methods including Lasso and reweighed ℓ_1 methods with a logarithmic penalty function [32]. Moreover, we empirically show that the number of observations required to obtain a correct reconstruction in the noiseless case using the presented algorithm is much lower than that of the standard Lasso algorithm. In the presence of noise, such an empirical comparison is hard to perform due to the identifiability problem. That is, we can obtain several models fitting the data with the same sparsity level but completely different sparsity patterns. Hence finding a model matching the "true" structure can simply be impossible.

Phase diagrams can provide additional insight into regression problems and the algorithms used to solve them. In [6], the authors use dictionary matrices with

Gaussian i.i.d. entries and provide the first rigorous analysis that explains why phase transitions are ubiquitous in random ℓ_1 regression problems. In our setting, entries of dictionary matrices are not Gaussian i.i.d, hence the theoretical results from [6] do not apply. However, we observed that the phase diagrams for various network size N remain very similar and exhibit the same trends and "breakdowns". Taking into account the results presented in [6], the similarities that we observed among the phase diagrams might be more than just empirical observations. This will constitute the basis for one of our future work directions.

Chapter 12

Fault Diagnosis of Power System

12.1 Introduction

Power networks are large-scale spatially distributed systems. Being critical infrastructures, they possess strict safety and reliability constraints. The design of monitoring schemes to diagnose anomalies caused by unpredicted or sudden faults on power networks is thus of great importance [170]. To be consistent with the international definition of the fault diagnosis problem, the recommendations of the IFAC Technical Committee *SAFEPROCESS* is accordingly employed in what follows. Namely, this work proposes a method to: 1) decide whether there is an occurrence of a fault and the time of this occurrence (*i.e. detection*), 2) establish the location of the detected fault (*i.e. isolation*), and 3) determine the size and time-varying behaviour of the detected fault (*i.e. identification*).

Since power networks are typically large-scale and have nonlinear dynamics, fault diagnosis over transmission lines can be a very challenging problem. This chapter draws inspiration from the fields of signal processing and machine learning to combine compressive sensing and variational Bayesian inference techniques so as to offer an efficient method for fault diagnosis.

Most of the literature available on fault diagnosis focuses on systems approximated by linear dynamics [49] Beyond linear systems descriptions, the dynamics of buses in power networks can be described by the so-called swing equations where the active power flows are nonlinear functions of the phase angles. Works that have considered fault detection and isolation in power networks include [171, 230, 125]. [171] focuses on distributed fault detection and isolation using linearised swing dynamics and the faults are considered to be additive. The method developed in [230] is used to to detect sensor faults assuming that such faults appear as biased faults added to the measurement equation. In [125], a fault detection and isolation residual generator is presented for nonlinear systems with additive faults. The nonlinearities in [125] are not imposed *a priori* on the model structure but treated as disturbances with some known patterns.

To summarise, the works [49, 171] use linear systems to characterise the dynamics of power networks and the faults are assumed to be additive. Though the system dynamics are nonlinear in [230, 125], the faults are still assumed to be additive. The methods developed on the basis of these conservative assumptions yield several problems. Firstly, the linear approximation to nonlinear swing equations can only be used when the phase angles are close to each other. However, when the system is strained and faults appear, phase angles can often be far apart. Therefore, a linear

approximation is inappropriate in strained power network situations. Secondly, it is well-known that a large portion of power system faults occurring in transmission lines do not involve additive faults, e.g. a short-circuit fault occurring on the transmission lines between generators would correspond to some changes in the parameters of the nonlinear terms appearing in the swing equation [109]. Furthermore, the inevitable and frequent introduction of new components in a power network contributes to the vulnerability of transmission lines, which, if not appropriately controlled, can lead to cascading failures [84]. Such cascading failures cannot be captured by additive faults. Finally, the methods mentioned above only address fault detection and isolation rather than identification, which is crucial to take appropriate actions when faults occur on transmission lines.

Contributions. The power networks considered in this chapter are described by the *nonlinear swing equations* with additive process noise. The faults are assumed to occur on the transmission lines of the power network. The problem of fault diagnosis, i.e. detection, isolation and identification, of such nonlinear power networks is formulated as a compressive sensing or sparse signal recovery problem. To solve this problem we consider a sparse Bayesian formulation of the fault identification problem, which is then casted as a nonconvex optimisation problem. Finally, the problem is relaxed into a convex problem and solved efficiently using an iterative reweighted ℓ_1-minimisation algorithm. The resulting efficiency of the proposed method enables real-time detection of faults in large-scale networks.

The outline of this application is as follows. Section 12.2 introduces the nonlinear model of power networks considered in this chapter. Section 12.3 formulates the fault diagnosis problem of power networks as a compressive sensing or sparse signal recovery problem. Section 12.4 applies the method to a power network with 20 buses and 80 transmission lines and, finally, Section 12.5 concludes and discusses several future problems.

12.2 Power System Model

Power systems are examples of complex systems in which generators and loads are dynamically interconnected. Hence, they can be seen as networked systems, where each bus is a node in the network. We assume that all the buses in the network are connected to synchronous machines (motors or generators). The nonlinear model for the active power flow in a transmission line connected between bus i and bus j is given as follows. For $i = 1, \ldots, n$, the behaviour of bus/node i can be represented

by the swing equation [171, 230, 109]

$$m_i\ddot{\delta}_i(t) + d_i\dot{\delta}_i(t) - P_{mi}(t) = -\sum_{j\in N_i} P_{ij}(t), \tag{12.1}$$

where δ_i is the phase angle of bus i, m_i and d_i are the inertia and damping coefficients of the motors and generators, respectively, P_{mi} is the mechanical input power, P_{ij} is the active power flow from bus i to j, and N_i is the neighbourhood set of bus i where bus j and i share a transmission line or communication link.

Considering that there are no power losses nor ground admittances, and letting $V_i = |V_i|e^{\tilde{j}\delta_i}$ be the complex voltage of bus i where $\tilde{j}$ represents the imaginary unit, the active power flow between bus i and bus j, P_{ij}, is given by:

$$P_{ij}(t) = w_{ij}^{(1)}\cos(\delta_i(t) - \delta_j(t)) + w_{ij}^{(2)}\sin(\delta_i(t) - \delta_j(t)), \tag{12.2}$$

where $w_{ij}^{(1)} = |V_i||V_j|G_{ij}$ and G_{ij} is the branch conductance between bus i and bus j; and $w_{ij}^{(2)} = |V_i||V_j|B_{ij}$ and B_{ij} is the branch susceptance between bus i and bus j.

If we let $\xi_i(t) = \delta_i(t)$ and $\zeta_i(t) = \dot{\delta}_i(t)$, each bus can be assumed to have double integrator dynamics. The dynamics of bus i can thus be written:

$$\dot{\xi}_i(t) = \zeta_i(t), \tag{12.3}$$

$$\dot{\zeta}_i(t) = u_i(t) + v_i(t), \tag{12.4}$$

where ξ_i, ζ_i are scalar states, $v_i(t)$ is a known scalar external input, and u_i is the power flow

$$v_i(t) = \frac{P_{mi}(t)}{m_i} \tag{12.5}$$

$$u_i(t) = -\frac{d_i}{m_i}\zeta_i(t) - \frac{1}{m_i}\sum_{j\in N_i}[w_{ij}^{(1)}\cos(\xi_i(t) - \xi_j(t)) + w_{ij}^{(2)}\sin(\xi_i(t) - \xi_j(t))]. \tag{12.6}$$

The variables ξ_i and ζ_i can be interpreted as phase and frequency in the context of power networks.

In [171], the $\cos(\cdot)$ terms are neglected (no branch conductance between buses) and it is assumed that phase angles are close to each other. The dynamics in (12.1) are then linearised to yield

$$m_i\ddot{\delta}_i(t) + d_i\dot{\delta}_i(t) - P_{mi}(t) = -\sum_{j\in N_i} w_{ij}^{(2)}(\delta_i(t) - \delta_j(t)). \tag{12.7}$$

Each bus i is assumed to have double integrator dynamics as described in (12.3) and (12.4). $u_i(t)$ in (12.6) becomes a linear equation

$$u_i(t) = -\frac{d_i}{m_i}\xi_i(t) - \frac{1}{m_i}\sum_{j\in N_i} w_{ij}^{(2)}(\xi_i(t) - \xi_j(t)). \tag{12.8}$$

For the linearised system (12.8), a bus k is faulty if for some functions $f_{\xi k}(t)$ and $f_{\zeta k}(t)$ not identical to zero either $\dot{\xi}_i(t) = \zeta_i(t) + f_{\xi k}(t)$, or $\dot{\zeta}_i(t) = u_i(t) + v_i(t) + f_{\zeta k}(t)$. The functions $f_{\xi k}(t)$ and $f_{\zeta k}(t)$ are referred to as fault signals. Model-based or observer-based fault diagnosis methods are available for power networks (see [171] and reference therein). However, specific aspects need careful consideration when dealing with fault diagnosis in power networks. Firstly, the simplified linear model can only be used when the phase angles are close to each other. However, when the system is strained and faults appear, phase angles can often be far apart.

In transmission systems the $\sin(\cdot)$ term in (12.2) is the dominating one. To perform a linearisation, one often assumes "small angle differences" between nodes and hence "small" power flows. This typically works well under normal operation. However, if the power system is under a lot of strain, i.e. if power flows are closer to the theoretical maximum, the angle difference becomes close to 90 degrees and the nonlinearity of the $\sin(\cdot)$ term becomes quite noticeable. In particular, if, in a transient state, the angle difference exceeds 90 degrees, generators typically loose synchrony and trip. This is not captured by linear models. In such circumstances, the linear model cannot be used to approximate the nonlinear model in (12.1) anymore. Secondly, power networks are highly distributed and interconnected, and more than one transmission line can be faulty at a given time. Thirdly, to be more realistic, some process noise ε_i should be incorporated into the second-order system (12.1) for each bus i:

$$m_i\ddot{\delta}_i(t) + d_i\dot{\delta}_i(t) - P_{mi}(t) = -\sum_{j=1}^{n} P_{ij}(t) + \varepsilon_i(t), \tag{12.9}$$

Based on the *swing* equation above, the state space model (12.3) and (12.4) can then be rewritten under the form:

$$\dot{\xi}_i(t) = \zeta_i(t), \tag{12.10}$$

$$\dot{\zeta}_i(t) = u_i(t) + v_i(t) + \varepsilon_i(t), \tag{12.11}$$

where the noise $\varepsilon_i(t)$ is assumed to be i.i.d. Gaussian with

$$\mathbb{E}(\varepsilon_i(p)) = 0$$
$$\mathbb{E}(\varepsilon_i(p)\varepsilon_i(q)) = \epsilon_i^2\delta(p-q).$$

Remark 23 *Here we only consider a dynamical system model with process noise ε_i since, in power networks, the measurement noise is small and would typically not have a catastrophic effect on the performance of detection algorithms [187].*

12.3 Fault Diagnosis Problem of Nonlinear Power Systems

Given the model and explanation above, we primarily focus on the following setting in this chapter.

Definition 5 *If a power network can be described by (12.10) and (12.11), the transmission line between bus i and bus j is faulty when $w_{ij}^{(1)}$ changes to a new scalar $w_{ij}^{[\mathbf{f}](1)}$ and/or $w_{ij}^{(2)}$ changes to a new scalar $w_{ij}^{[\mathbf{f}](2)}$, where $w_{ij}^{(1)}$ and $w_{ij}^{(2)}$ are the weights for the $\cos$ and $\sin$ terms defined in (12.6).*

Based on the considerations above and Definition 5, the problem that we are interested in solving is the following:

Problem 11 *Having access to the measurements and the distribution of the noise, how can we detect the occurrence and magnitude of a fault, namely, how can we estimate the magnitude of the errors $w_{ij}^{(1)} - w_{ij}^{[\mathbf{f}](1)}$ and $w_{ij}^{(2)} - w_{ij}^{[\mathbf{f}](2)}$, $\forall i, j$, using the smallest possible number of samples.*

In what follows we make the following assumption.

Assumption 11 *The power networks described by (12.10) and (12.11) are fully measurable, i.e. the phase angles of all the buses can be measured.*

12.3.1 Model Transformation

Applying the forward Euler discretisation scheme to (12.10) and (12.11) and assuming the discretisation step Δt is constant for all k, we obtain the following

discrete-time system approximation to the continuous-time system (12.10) and (12.11):

$$\frac{\xi_i(t+1)-\xi_i(t)}{\Delta t}=\zeta_i(t), \tag{12.12}$$

$$\frac{\zeta_i(t+1)-\zeta_i(t)}{\Delta t}=u_i(t)+v_i(t)+\eta_i(t), \tag{12.13}$$

where the noise $\eta_i(t_k)$ is assumed to be i.i.d. Gaussian distributed: $\eta_i(t_k)\sim\mathcal{N}(0,\sigma_i^2)$, with $\mathbb{E}(\eta_i(t_p))=0$, $\mathbb{E}(\eta_i(t_p)\eta_i(t_q))=\sigma_i^2\delta(t_p-t_q)$.

Defining the new variable

$$e_i(t+1)\triangleq-\frac{(\zeta_i(t+1)-\zeta_i(t))}{\Delta t}-\frac{d_i\zeta_i(t)}{m_i}+\frac{P_{mi}(t)}{m_i}, \tag{12.14}$$

we have

$$e_i(t+1)=\frac{1}{m_i}\sum_{j\in N_i}[w_{ij}^{(1)}\cos(\xi_i(t)-\xi_j(t))+w_{ij}^{(2)}\sin(\xi_i(t)-\xi_j(t))]+\eta_i(t), \tag{12.15}$$

where e_i, the power flow measurement, is treated as the output of the system. Since the state variables $\zeta(t+1)$ and $\zeta(t)$, the parameters Δt, d_i and m_i, and the input P_{mi} are known, the quantity $e_i(t+1)$ can be computed in real time. It should be noted that "real time" is to be understood as "within the sampling time Δt of the sensors in power generators".

By defining $\mathbf{x}(t)=[\xi_1(t),\ldots,\xi_N(t)]$ we can write (12.14) into a vector form:

$$e_i(t+1)=f_i(\mathbf{x}(t))\boldsymbol{\beta}_i^{\text{true}}+\eta_i(t), \tag{12.16}$$

with

$$\begin{aligned}
f_i(\mathbf{x}(t))&=[f_i^{(1)}(\mathbf{x}(t)),f_i^{(2)}(\mathbf{x}(t))]\in\mathbb{R}^{2n},\\
f_i^{(1)}(\mathbf{x}(t))&=[\cos(\xi_i(t)-\xi_1(t)),\ldots,\cos(\xi_i(t)-\xi_N(t))]\in\mathbb{R}^n,\\
f_i^{(2)}(\mathbf{x}(t))&=[\sin(\xi_i(t)-\xi_1(t)),\ldots,\sin(\xi_i(t)-\xi_N(t))]\in\mathbb{R}^n,\\
\boldsymbol{\beta}_i^{\text{true}}&=[\boldsymbol{\beta}_i^{(1)},\boldsymbol{\beta}_i^{(2)}]^{\mathrm{T}}\in\mathbb{R}^{2n},\\
\boldsymbol{\beta}_i^{(1)}&=[w_{i1}^{(1)},\ldots,w_{iN}^{(1)}]\in\mathbb{R}^n,\\
\boldsymbol{\beta}_i^{(2)}&=[w_{i1}^{(2)},\ldots,w_{iN}^{(2)}]\in\mathbb{R}^n,
\end{aligned}$$

where $f_i(\mathbf{x}(t))$ represents the transmission functions and $\boldsymbol{\beta}_i$ represents the corresponding transmission weights associated to the topology of the network.

Remark 24 *In real power systems, a sampling frequency for phasor measurement unit (PMU) as high as* 2500 *samples per second can be achieved. In this case, the sampling time* Δt *is* $4 * 10^{-5}$ *second and the Euler discretisation* $\frac{\xi_i(t+1)-\xi_i(t)}{\Delta t}$ *will typically provide a good approximation of* $\dot{\xi}_i(t)$*.*

12.3.2 Fault Diagnosis Algorithm

As stated in Definition 5, if there are no faults occurring in the transmission lines between bus i and other buses, the dynamics of the power networks will evolve according to (12.16). The **e**xpected output for the next sampling time is defined to be

$$e_i^{[\mathbf{e}]}(t+1) = f_i(\mathbf{x}(t))\boldsymbol{\beta}_i^{\text{true}}. \tag{12.17}$$

From (12.16) and (12.17), it is easy to show that $e_i(t+1) - e_i^{[\mathbf{e}]}(t+1)$ is a stochastic variable with zero mean and variance σ^2. If there are faults occurring in the transmission lines between bus i and other buses, the corresponding transmission weights will change from $\boldsymbol{\beta}_i^{\text{true}}$ to $\boldsymbol{\beta}_i^{\text{fault}}$. Similar to the definition of $\boldsymbol{\beta}_i^{\text{true}}$, $\boldsymbol{\beta}_i^{\text{fault}} = [\boldsymbol{\beta}_i^{[\mathbf{f}](1)}, \boldsymbol{\beta}_i^{[\mathbf{f}](2)}]^{\text{T}}$ where $\boldsymbol{\beta}_i^{[\mathbf{f}](1)} = [w_{i1}^{[\mathbf{f}](1)}, \ldots, w_{iN}^{[\mathbf{f}](1)}]$ and $\boldsymbol{\beta}_i^{[\mathbf{f}](2)} = [w_{i1}^{[\mathbf{f}](2)}, \ldots, w_{iN}^{[\mathbf{f}](2)}]$. We thus have:

$$e_i^{[\mathbf{f}]}(t+1) = f_i(\mathbf{x}(t))\boldsymbol{\beta}_i^{\text{fault}} + \eta_i(t), \tag{12.18}$$

where $e_i^{[\mathbf{f}]}$ is the output when there are **f**aults.

From (12.17) and (12.18), it is easy to find that $e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)$ is a stochastic variable with mean $f_i(\mathbf{x}(t))(\boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}})$ and variance σ^2. Denoting

$$y_i = e_i^{[\mathbf{f}]} - e_i^{[\mathbf{e}]}, \boldsymbol{\beta}_i = \boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}},$$

we have:

$$y_i(t+1) = f_i(\mathbf{x}(t))\boldsymbol{\beta}_i + \eta_i(t). \tag{12.19}$$

Remark 25 *We formulate the faults identification problem as a linear regression problem. The dependent variable* $e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)$ *is the difference between the expected output and the faulty output; the unknown variable we want to estimate is the difference between the faulty transmission weights and the true transmission weights.*

There are three problems of interest based on the formulation in (12.19): a) detection of a fault; b) isolation of a fault, i.e. determination of the type, location and time of

occurrence of a fault; and c) identification of the size and time-varying behaviour of a fault. In the noiseless case, when there are no faults, $\forall i$, y_i and $\boldsymbol{\beta}_i$ are both equal to zero. On the other hand, when there are faults, certain y_i are nonzero. So the faults can be *detected* by identifying the entries y_i that are nonzero. However, in the noisy case, even when there are no faults, y_i is nonzero most of the time since it is a stochastic variable with zero mean. This can be interpreted in a probabilistic way by Chebyshev's Inequality:

$$p(|e_i(t+1) - e_i^{[\mathbf{e}]}(t+1)| \geq l\sigma = \sigma^*) \leq \frac{1}{l^2}$$

where $l \in \mathbb{R}^+$. According to this inequality, when there are no faults, the deviation between true and expected outputs, i.e. $|e_i(t+1) - e_i^{[\mathbf{e}]}(t+1)|$ cannot be much greater than zero with high probability. On the other hand, when there is a fault, the deviation between faulty and expected outputs, i.e. $|e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)|$ should be much greater than zero with high probability.

From an isolation point of view and Chebyshev's inequality, when $|e_i^{[\mathbf{f}]}(t+1) - e_i^{[\mathbf{e}]}(t+1)|$ is much greater than σ, the fault can be *isolated* with high probability (e.g. if the threshold is set to $l\sigma = 10\sigma$, then the probability is 99%).

If at time t_0 faults have been detected and isolated, the remaining task is to perform fault *identification*, i.e. to identify the location of the faults or equivalently to find the nonzero entries in w_i. Assuming that $M+1$ successive data points, including the initial data point at t_0, are sampled and defining $N = 2n$ and

$$\begin{aligned}
\mathbf{y}_i &\triangleq [y_i(t_1), \ldots, y_i(t_M)]^{\mathrm{T}} \in \mathbb{R}^M, \\
\mathbf{X}_i &\triangleq \begin{bmatrix} f_i^{(1)}(\mathbf{x}(t_0)) & f_i^{(2)}(\mathbf{x}(t_0)) \\ \vdots & \vdots \\ f_i^{(1)}(\mathbf{x}(t_{M-1})) & f_i^{(2)}(\mathbf{x}(t_{M-1})) \end{bmatrix} \\
&= \begin{bmatrix} f_i(\mathbf{x}(t_0)) \\ \vdots \\ f_i(\mathbf{x}(t_{M-1})) \end{bmatrix} \in \mathbb{R}^{M \times N}, \\
\boldsymbol{\eta}_i &\triangleq [\eta_i(t_0), \ldots, \eta_i(t_{M-1})]^{\mathrm{T}} \in \mathbb{R}^M,
\end{aligned} \tag{12.20}$$

we can write N independent equations of the form:

$$\mathbf{y}_i = \mathbf{X}_i \boldsymbol{\beta}_i + \boldsymbol{\eta}_i, \quad (i = 1, \ldots, n). \tag{12.21}$$

Based on the formulation in (12.21), our goal is to find β_i given the output data stored in $\mathbf{y}_i$.

To solve for β_i in (12.21) amounts to solving a linear regression problem. This can can be done using standard least square approaches. It should be noted that the linear regression problem for bus i in (12.21) is independent from the the linear regression problems for the other buses. In what follows, we will focus on finding the solution to one of these linear regression problem and omit the subscripts i in (12.21) for simplicity of notation. We thus write

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\eta}, \tag{12.22}$$

where $\mathbf{y}$ is the difference between the faulty measurements and the expected measurements, or namely, the **error measurements**; and $\boldsymbol{\beta}$ is the difference between the faulty parameters and the true parameters, or namely, the **faults**. We address this linear regression problem under the following assumption.

Assumption 12 *A maximum of S transmission lines are faulty, i.e. $\boldsymbol{\beta}$ has at most S non-zero entries. In other words, $\boldsymbol{\beta}$ is S-sparse or mathematically, $\|\mathbf{w}\|_0 \leq S$. The constant S is assumed unknown to the system administrator.*

Remark 26 *Assumption 12 is realistic for small values of S since in the context of a power system, it is typically not the case that all the transmission lines are faulty simultaneously. Furthermore, since buses in power networks are typically sparsely connected the number of faults is typically much smaller than the size of the network n, i.e. $S \ll n$. Therefore $S \ll N = 2n$.*

On the other-hand, the size of $\mathbf{y}$ equals to the number of samples needed to identify the location of the faults after the they occur. From a practical viewpoint, the number of samples should be as small as possible. However, standard least square approaches to (12.22) cannot meet this goal as they require at least $2N$ samples. Moreover, the solution to the standard least square problem is generically dense (hence, violating Assumption 12) and cannot be used to identify which transmission lines are likely to be faulty by identification of the nonzero entries of the estimated $\boldsymbol{\beta}^{\text{fault}} - \boldsymbol{\beta}^{\text{true}}$.

Based on Algorithm 19, we can summarise the fault diagnosis algorithm for nonlinear power systems in Algorithm 20.

Algorithm 20 Diagnosis for faults

Set a threshold σ^* as indicated in Section 12.3.2, e.g. $\sigma^* = 10 \times \sigma$;
for $k = 0, \ldots, T$ **do**
    $\%$ *T is an integer indicating the number of diagnosis rounds*;
    Collect $\xi_i(t)$ and $\zeta_i(t)$ in (12.12) and (12.13)
    for $i = 1, \ldots, N$ **do**
        Calculate the output data $e_i(t+1)$ in (12.14);
        Calculate the expected output $e_i^{[\mathbf{e}]}(t+1)$ in (12.17);
        if $|e_i(t+1) - e_i^{[\mathbf{e}+]}(t+1)| > \sigma^*$ **then**
            Fault is detected for bus i; $\%$ {*fault detection procedure*}
            Compute $y_i(t+1)$ in (12.19);
            if $|y_i(t+1)| > \sigma^*$ **then**
                Isolate bus i; $\%$ {*fault isolation procedure*}
            end if
        end if
        Set $M \leftarrow k$;
        Apply Algorithms propose in Chapter 6.6 to identify the faults $\hat{\boldsymbol{\beta}}_i$; $\%$ {*fault identification procedure*}
    end for
    if $\forall i$, $\|\hat{\boldsymbol{\beta}}_i\|_0$ converge to some constant **then**
        Break;
    end if
end for
An estimate for the faults $\hat{\boldsymbol{\beta}}$ in (12.21), $i = 1, \ldots, n$.

12.4 Numerical Study

The effectiveness of our theoretic developments is here illustrated for a randomly generated power network with 20 buses. If all the buses are fully connected, the possible number of transmission lines is 380. We assume that the number of transmission lines is 79 (i.e. we assume that the sparsity of the network is around 20%). Its dynamics can be described by the nonlinear swing equations described in (12.10) and (12.11). $w_{ij}^{(1)}$ and $w_{ij}^{(2)}$ are positive real numbers as shown in Fig. 12.3(a). Let the noise variance $\sigma^2 = 1$. All the parameter values are selected to be similar to those in [109, 146].

Since the sampling frequency is around 50 Hz for the PMU [109, 146], we assume the sampling interval to be $20\ ms$. We thus assume that the discretisation step in Section 12.3 is performed using a sampling interval $\Delta t = 20\ ms$.

Consider the power networks model in (12.10) and (12.11). At time instant $t = 3s$, there are faults occurring in five transmission lines simultaneously. Specifically, a randomly chosen set of faults can be described as follows:

$$\forall (i,j) \in \{(5,18),(7,2),(11,15),(16,18),(19,9)\}$$

, $w_{ij}^{(1)}$ and $w_{ij}^{(2)}$ in (12.6) respectively (which correspond to $\cos$ and $\sin$ terms) are set to zeros. 5 buses are involved in these transmission lines, i.e. buses 5, 7, 11, 16 and 19. Following the procedure in Algorithm 20, we want to detect and isolate these 5 buses. After detection and isolation, the identification procedure will be performed. We consider $\sigma^* = 10\sigma = 10$ to initialise Algorithm 20.

First, we detect and isolate the buses with $|y_i(t+1)| > \sigma^*$. In Fig. 12.1, it can be seen that at time instant $t = 3.02s$ (only one sampling time after the faults occur), $|y_5|$, $|y_7|$, $|y_{11}|$, $|y_{16}|$ and $|y_{19}|$ are much greater than σ^* (we set $\sigma^* = 10$ here). Therefore, we can draw the conclusion that buses 5, 7, 11, 16 and 19 are faulty and should be isolated. Next, we identify the faults that occur in the transmission lines connecting the previously isolated buses, i.e. buses 5, 7, 11, 16 and 19. In Fig. 12.2, the time trajectory of the sparsity of the estimated fault $\|\hat{\boldsymbol{\beta}}_i\|_0$, i.e. $\|\boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}}\|_0$ (see Remark 25), for $i = 5, 7, 11, 16, 19$ are depicted starting at the time point $t = 3.02s$ when the faults are detected. We set the pruning threshold to 10^{-3} during the identification procedure of the faults. We define a positive integer n^* to indicate the number of identification rounds which are required to terminate the identification procedure, e.g. $n^* = 10$. As shown in Fig. 12.2, at time instant $t = 3.52s$, the sparsity of the

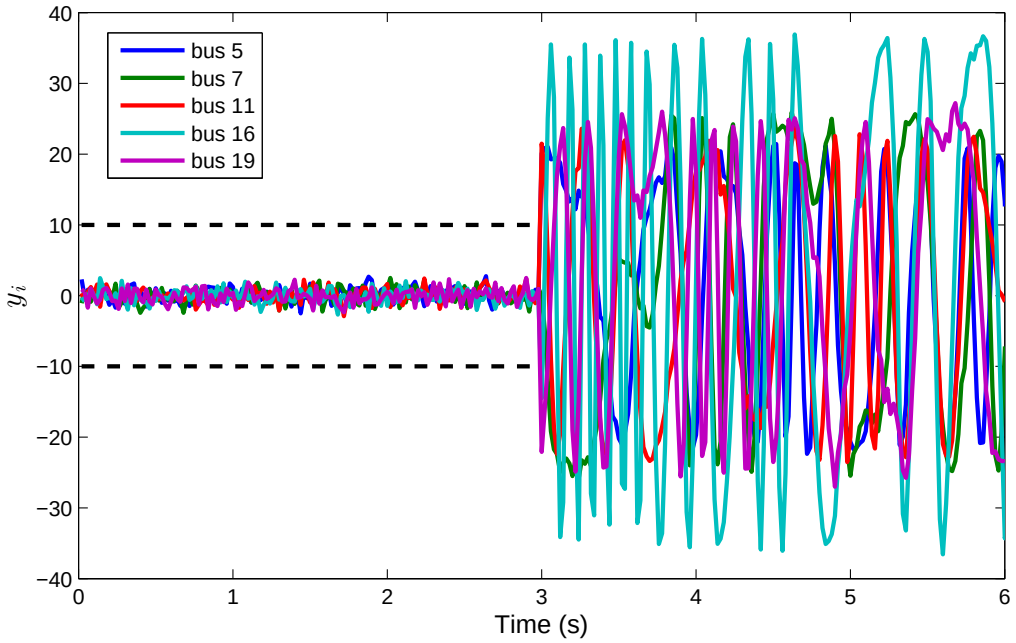


Fig. 12.1 Time-series of y_i for all buses. The black dashed lines indicate the threshold σ^* in Algorithm 20. The coloured solid lines are the phase angle measurements for bus i, $i = 5, 7, 11, 16, 19$. At time instant $t = 3.02s$, $|y_5|$, $|y_7|$, $|y_{11}|$, $|y_{16}|$ and $|y_{19}|$ are much greater than σ^* ($\sigma^* = 10$ here).

estimated fault, i.e. $\|\mathbf{w}_i^{\text{fault}} - \mathbf{w}_i^{\text{true}}\|_0$ for bus $i = 5, 7, 11, 16, 19$ all become equal to 2 and remain unchanged afterwards. At time instant $t = 3.72s$, only $n^* = 10$ sampling rounds after $t = 3.52s$, we terminate the identification procedure as the sparsity for all the estimated faults is considered to be stable.

In Fig. 12.3(a) and Fig. 12.3(b), we illustrate the true weight matrix and the estimated absolute error matrix $|\boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}}|$. As we can see, all the 5 faults that are occurring in the transmission lines have been identified with high accuracy.

12.5 Conclusion and Discussion

In this Chapter, we addressed the problem of automatic fault diagnosis in large-scale power networks where the buses are described by second-order nonlinear swing equations with process noise. In particular, this work focused on a class of transmission lines faults. We combined tools from compressive sensing and variational Bayesian inference to develop a method to detect, isolate and identify

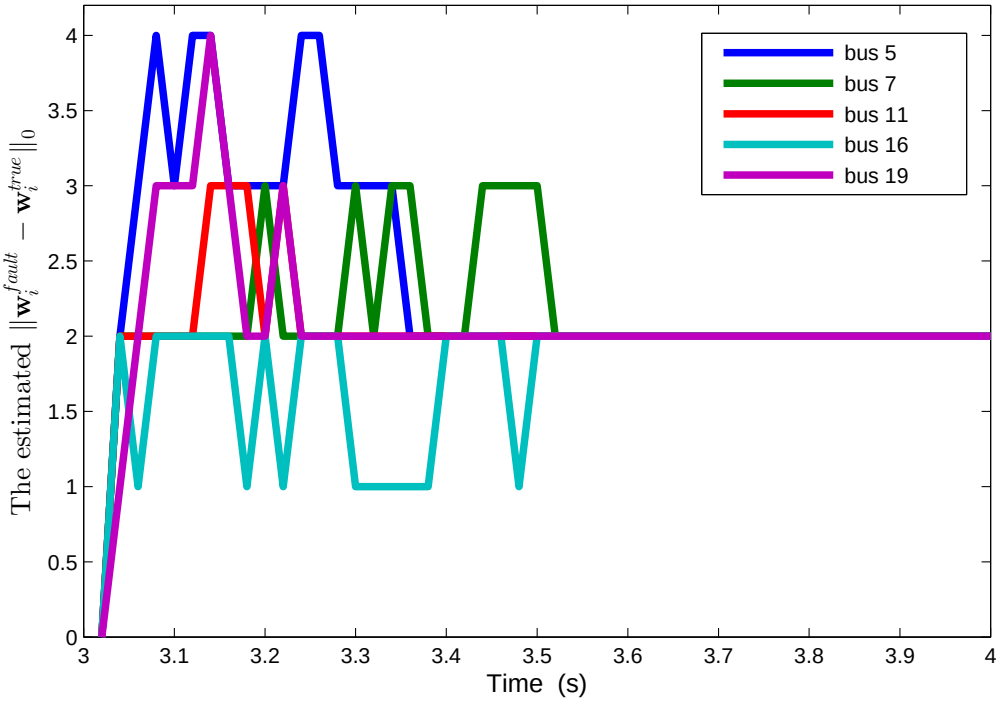
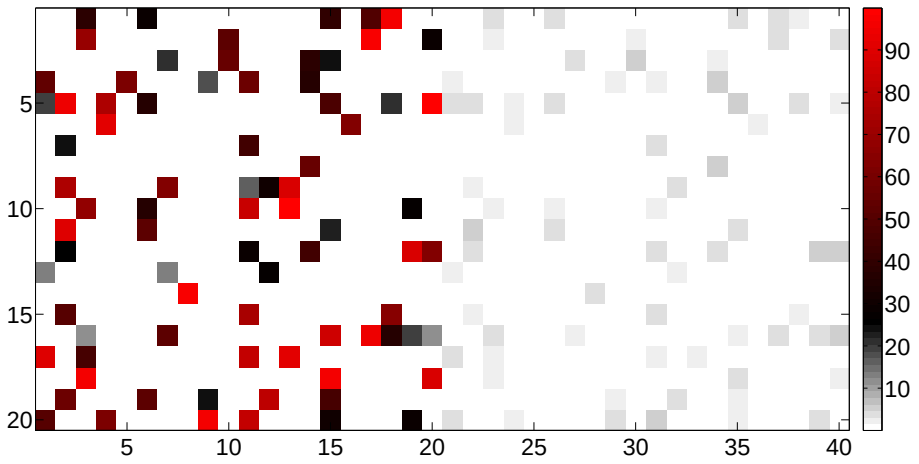


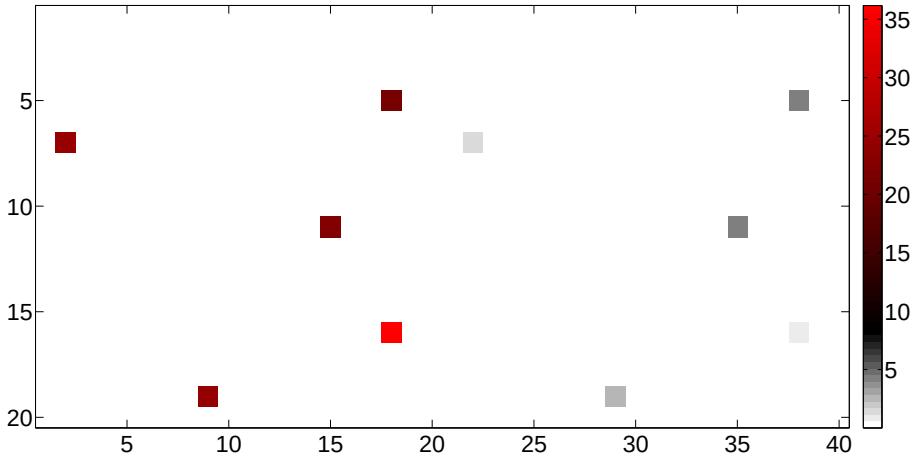
Fig. 12.2 Time-series of the sparsity of the estimated fault, i.e. $\|\mathbf{w}_i^{\text{fault}} - \mathbf{w}_i^{\text{true}}\|_0$ for bus $i = 5, 7, 11, 16, 19$.

the faults. An illustrative example showed the application of the proposed method to fault diagnosis in nonlinear power networks.

Beyond the results in this chapter, some issues still remain for further investigation. This chapter assumed that the system is fully measurable. Current work aims to extend the proposed framework to fault diagnosis with partially measured power systems.



(a) True weight matrix



(b) Absolute error weight matrix: $|\boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}}|$ (see Remark 25)

Fig. 12.3 Identification of transmission lines faults: (a) describes the true weight matrix with around 20% nonzero entries. The left half of the matrix corresponds to the weights for $\cos(\cdot)$ terms while the right half is for $\sin(\cdot)$ terms. (b) represents the absolute error weight matrix, which is defined as $|\boldsymbol{\beta}_i^{\text{fault}} - \boldsymbol{\beta}_i^{\text{true}}|$. The non-zero terms in the heat map correspond directly to the faulty transmission lines: (5,18),(7,2),(11,15),(16,18),(19,9).

Part IV

Conclusion and Future Direction

Chapter 13

Conclusion

In summary, this thesis makes some attempts to address the following challenges arising in big time series data analytics

General Problem Formulation for Nonlinear System Identification: Modern time series data are often ultra-high dimensional (e.g. biology data, power systems data). Furthermore, the type of data used for modelling are collected in different fashions, e.g., from single or heterogeneous sources; collected "statically" or "streamingly". The underlying dynamical systems are often large-scale (e.g. gene networks, power networks). However, the representation of nonlinear dynamical systems is extremely simple or *sparse*. This thesis adapts and extends the regularised sparse learning formulation in different aspects to address various nonlinear identification problems.

Both a selection of time-invariant and time-varying nonlinear dynamical systems are covered. For time-invariant system, the classic nonlinear system identification problem from single dataset is addressed in the beginning. Then we move to a more practical and significant yet complicated scenario where heterogeneous datasets are used simultaneously. Such datasets typically contain (a) data from several replicates of an experiment performed on a biological system of interest and/or (b) data measured from a biochemical system subjected to different experimental conditions, for example, changes/perturbations in biological inductions, temperature, gene knock-out, gene over-expression, etc. For time-varying systems, the regime-switch system identification problem is considered, i.e., the problem of identifying both the switching points and the nonlinear model structure within each regime. Then the abrupt change point detection problem is considered. Using these, the classic trending filtering and fault diagnosis problems are revisited. All the identification problems are formulated as various ℓ_0 type optimisation problems. In the end, we discuss some technical issues on data processing arising from practical applications.

Efficient Nonlinear System Identification Algorithms: These algorithms are not distinct and can be formulated in a unified way using Bayesian Learning with structural sparse prior. Furthermore, we suggest a series of iterative reweighted convex relaxation schemes for connecting these algorithms to popular algorithms including Lasso, Group-Lasso, Generalised-Lasso, Fused-Lasso and Graphical-Lasso. In this part, we go beyond from simple nonlinear model class to more general class; from data likelihood in Gaussian distribution to the more general exponential family. The estimation of the stochastic term also discussed including ARMA and ARCH. Many optimisation framework, such as (stochastic) gradient descent, Newton method,

Quasi-Newton method, alternating direction method of multiplier can be seamlessly integrated into our formulation as either centralised or distributed optimisation strategy to address high dimensionality and large scale problems. These algorithms largely enrich not only the family of time series modelling algorithms but also sparse signal recovery/modelling/estimation algorithms in various communities.

Future Directions Two future research directions based on the output of this thesis are pointed out, both related to "neurons". The first is focusing on theory and algorithm about modelling/identification/learning on deep neural networks. The second is focusing applications in neuroscience: understanding the neural basis of decision making using mathematical modelling from big data. Some promising results have showed the feasibility and potential impact of these directions.

Chapter 14

Future Direction

In this Chapter, several future works will be discussed. The begin with, I will discuss the possibility to the linear system identification problem. Fortunately, two papers are published on the internet in which I was co-authored and initialised the idea, see Yue et al. [225], Jin et al. [96]. However, I will not include them in my thesis as a contribution of mine.

Recently, deep learning using deep neural network representation has been successfully applied in many artificial intelligence applications. Interestingly, learning/modelling/identifying a (deep) neural network is essentially a *Nonlinear System Identification* problem as introduced by Lennart Ljung in [117, pp. 154] twenty years ago. Maybe due to some historical reason, neural networks in Lennart Ljung's book [117] has not been discussed a lot, only 2 pages contents are about neural network in his 609 pages book, say, multi-layer networks and recurrent neural network. Now in this thesis, it is probably a good time to discuss more on learning (identification) of deep neural networks from the perspective of nonlinear system identification. Some promising results have showed the feasibility and potential impact of the proposed directions.

It seems that the mechanism of human brain governing intelligence inspired greatly the research in deep learning. Vice versa, to understand how brain works is prominent and been hot for decades. In particular, the decision making process in human is of great interest for myself. Given the incredible amount of neural data, such as EEG, ECG, fMRI, etc., all recorded in terms of time series, there is great potential to model the brain network to understand the mechanism.

14.1 Future Direction I: Bayesian Deep Learning

14.1.1 Background on Deep Learning and Deep Neural Networks

It is commonly accepted for a deep learning system, the underlying neural network (NN) has to be big and complex. We argue that this perception may not be true. Many of the neurons and their associated connections, both incoming and outgoing ones, can be *dropped permanently* which results in a NN with much smaller size. This is very similar to sparse distributed representations in brain. The human neocortex has roughly 100 billion neurons, but at any given time only a small percent are active in performing a particular cognitive function [130]. For the non-sequence or non time dependent data, the active neurons may be fixed and not change over time [43]. Dropping neurons is also the key idea in Dropout [86, 180], a successful

regularisation technique to prevent overfitting in NNs. In their work, the neurons are *dropped temporarily in training*. In the end for prediction, the model is still of full size and fully connected.

Hereafter, we aim at training a simple network when it can achieve comparable performance to the fully connected NN, but with number of neurons and connections as few as possible. Dropping connections may be not difficult by introducing weight decay regularisers. However, dropping neurons is challenging. On one hand, the weight decay regularisation can't penalise *all* the connections associated with one neuron simultaneously. On the other hand, it is attempted to suppress the neurons to fire such as the use of rectifier as activation function [66], regularisation techniques like K-L sparsity in the sparse autoencoder variants [103, 20], or constraints like max-norm [178, 67]. However, a neuron not firing in training still can't be dropped for testing and prediction since her connections' weights are not zeros. As an alternative, network pruning by dropping connections below a threshold has been widely studied to compress a pre-trained fully connected NN models reduce the network complexity and over-fitting, see early work [111, 78] and more recently [75, 74]. Unfortunately, such pruning strategy may not effectively drop neurons. For example, a NN may consist of large number of neurons but few connections. Though, the model size/storage space may not be challenging but bring another challenge for chip design for storage and computation, e.g. (mobile) GPU, FPGA, etc.

We use the following notation throughout this Chapter. Bold lower case letters ($\mathbf{x}$) denote vectors, bold upper case letters ($\mathbf{X}$) denote matrices, and standard weight letters (x) denote scalar quantities. We use subscripts to denote variables as well $\mathbf{W}^\ell$ (such as $\mathbf{W}^1 : n^0 \times n^1, \mathbf{W}^2 : n^1 \times n^2$). n^0 is the number of features of the input. We use subscripts to denote either entire rows ($\mathbf{W}^\ell_{p,:}$ for the p-th row of $\mathbf{W}^\ell$) or entire columns ($\mathbf{W}^\ell_{:,q}$ for the q-th column of $\mathbf{W}^\ell$). We use the standard capital letter with subscript to denote the element index of a specific variable: $W^1_{p,q}$ denotes the element at row p column q of the variable $\mathbf{W}^1$. We also use $\mathbf{O}^\ell$ to be the indicator for the neurons in layer ℓ. For example, $\mathbf{O}^0$ consist of n^0 neurons in the input layer, indexed as $O^0_1, \ldots, O^0_{n^0}$.

We start with the case of a three layer NN with *a single hidden layer*. The generalisation to multiple layers is straightforward. Denote by $\mathbf{W}^1, \mathbf{W}^2$ the weight matrices connecting the first layer to the hidden layer and connecting the hidden layer to the output layer respectively. These linearly transform the layers' inputs before applying some element-wise non-linearity $\sigma(\cdot)$. Denote by $\mathbf{b}$ the biases by which we

shift the input of the non-linearity. We assume the model to output n^2 dimensional vectors while its input is n^0 dimensional vectors, with K hidden units. Thus $\mathbf{W}^1$ is a $n^0 \times n^1$ matrix, $\mathbf{W}^2$ is a $n^1 \times n^2$ matrix, and $\mathbf{b}$ is a n^1 dimensional vector. A standard NN model would output the following given some input $\mathbf{x}$

$$\widehat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}^1 + \mathbf{b})\mathbf{W}^2 \tag{14.1}$$

To use the NN model for regression we might use the Euclidean loss (also known as "square loss"),

$$E_{\text{regression}} = \frac{1}{2N}\sum_{n=1}^{N} ||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 \tag{14.2}$$

where $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ are N observed outputs, and $\{\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_N\}$ being the outputs of the model with corresponding observed inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.

To use the model for classification, predicting the probability of $\mathbf{x}$ being classified with label $1, ..., D$, we pass the output of the model $\widehat{\mathbf{y}}$ through an element-wise softmax function to obtain normalised scores: $\widehat{p}_{nd} = \exp(\widehat{y}_{nd}) / \left(\sum_{d'} \exp(\widehat{y}_{nd'})\right)$. Taking the log of this function results in a *softmax* loss,

$$E_{\text{classification}} = -\frac{1}{N}\sum_{n=1}^{N} \log(\widehat{p}_{n,c_n}) \tag{14.3}$$

where $c_n \in [1, 2, ..., D]$ is the observed class for input n.

During optimisation regularisation terms are often added. Some of the well known regularisation include ℓ_1 regularisation and ℓ_2 regularisation, defined as

$$\begin{aligned} \texttt{l1\_regulariser} &\triangleq \lambda_{\ell_1} \sum_{\ell=1}^{L} (\|\mathbf{W}^\ell\|_1 + \|\mathbf{b}^\ell\|_1) \\ \texttt{l2\_regulariser} &\triangleq \lambda_{\ell_2} \sum_{\ell=1}^{L} (\|\mathbf{W}^\ell\|_2^2 + \|\mathbf{b}^\ell\|_2^2) \end{aligned} \tag{14.4}$$

where λ_{ℓ_1} and λ_{ℓ_2} are often called weight decay or regularisation parameter which needs fined tuned.

Then it results in a minimisation objective (often referred to as cost),

$$\mathcal{L} \triangleq E + \texttt{l1\_regulariser} \quad \text{or} \quad \mathcal{L} \triangleq E + \texttt{l2\_regulariser}, \tag{14.5}$$

or a mixture of `l1_regulariser` and `l2_regulariser`, which is known as elastic net.

The goal of introducing `l1_regulariser` and `l2_regulariser` is to penalise the connections' weights between neurons to prevent overfitting. However, the application of such regularisers alone in deep neural network are not as successful as in linear regression and logistic regression. On the other hand, in the hardware computation especially using GPU, dropping connections may not save computation time and memory unless some special coding and processing is used [74]. The introduction of dropout achieve great success to avoid over-fitting in practice [86, 180] with these two regularisers. These regularisation techniques are suitable for preventing overfitting but may not be helpful in simplifying the NN structure. We believe that the key to automatically simplify a NN structure in training is to define proper regulariser by exploring the sparsity structure of the NN in a deep learning system.

14.1.2 Structural Sparsity in Deep Neural Network

Multi Layer Perceptron

Hereafter, we are seeking a strategy to drop neurons. Using the standard setup for NN, we have the weight matrix from layer $\ell-1$ to layer ℓ,

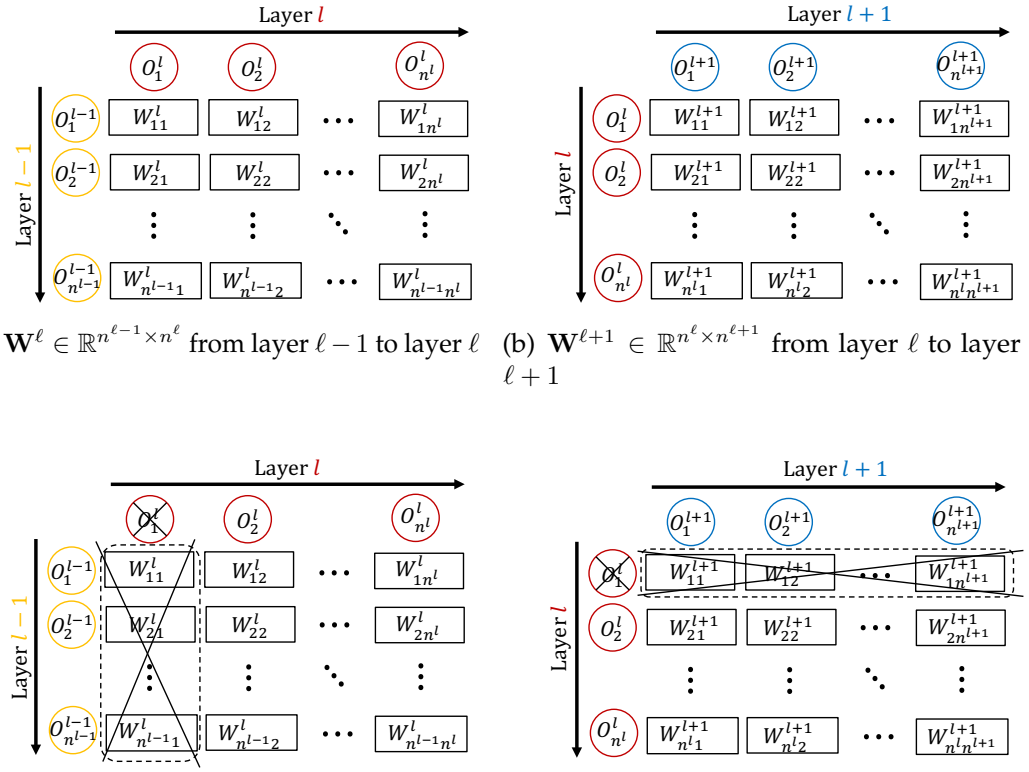
$$\mathbf{W}^{\ell} = \left[(\mathbf{W}^{\ell}_{1,:})^{\top}, \ldots, (\mathbf{W}^{\ell}_{n^{\ell-1},:})^{\top}\right]^{\top} = \left[\mathbf{W}^{\ell}_{:,1}, \ldots, \mathbf{W}^{\ell}_{:,n^{\ell}}\right] \tag{14.6}$$

where $\mathbf{W}^{\ell}_{i,:}$ denote the i-th row of $\mathbf{W}^{\ell}$, $i = 1, \ldots, n^{\ell-1}$; it encodes the incoming connections' weights from layer $\ell-1$ to the i-th neuron in layer ℓ, i.e., O^{ℓ}_i. Similarly, $\mathbf{W}^{\ell}_{:,j}$ denote the j-th column of $\mathbf{W}^{\ell}$, $j = 1, \ldots, n^{\ell}$; it encodes the outgoing connections' weights of the j-th neuron in layer ℓ, i.e., O^{ℓ}_i to all the neurons in the next layer, i.e., layer $\ell+1$. In particular, O^0_i denotes the i-th feature/neuron in input layer.

We first introduce two new regularisers, the first one is called `li_regulariser`(λ_{ℓ_i})

$$\texttt{li\_regulariser} \triangleq \lambda_{\ell_i} \sum_{\ell=1}^{L} \sum_{j=1}^{n^{\ell}} \|\mathbf{W}^{\ell}_{:,j}\|_2 = \lambda_{\ell_i} \sum_{\ell=1}^{L} \sum_{j=1}^{n^{\ell}} \sqrt{\sum_{i=1}^{n^{\ell-1}} \left(W^{\ell}_{ij}\right)^2} \tag{14.7}$$

This is used to regularise the incoming connections' weights of all the neurons across different layers over the whole network. The conceptual idea of removing all the incoming weights to neuron O^{ℓ}_1 from the neurons in layer $\ell-1$ therefore removal of herself is illustrated by comparing in Fig. 14.1(a) and14.1(c).



(a) $\mathbf{W}^{\ell} \in \mathbb{R}^{n^{\ell-1} \times n^{\ell}}$ from layer $\ell - 1$ to layer ℓ (b) $\mathbf{W}^{\ell+1} \in \mathbb{R}^{n^{\ell} \times n^{\ell+1}}$ from layer ℓ to layer $\ell + 1$

(c) $\mathbf{W}^{\ell} \in \mathbb{R}^{n^{\ell-1} \times n^{\ell}}$ from layer $\ell - 1$ to layer ℓ (d) $\mathbf{W}^{\ell+1} \in \mathbb{R}^{n^{\ell} \times n^{\ell+1}}$ from layer ℓ to layer $\ell + 1$

Fig. 14.1 A graphical illustration on the strategy of removing neurons. O_k^{ℓ} denotes the k-th neuron in layer ℓ , W_{ij}^{ℓ} denotes the weight of connection from neuron i in layer ℓ to neuron j in layer $\ell + 1$. The bottom figures showed a neuron can be removed either when all incoming connections' weights to her or her outgoing connections' weights are zeros *simultaneously*.

The second one is called `lo_regulariser`(λ_{ℓ_o})

$$\texttt{lo\_regulariser} \triangleq \lambda_{\ell_o} \sum_{\ell=1}^{L} \sum_{i=1}^{n^{\ell-1}} \|\mathbf{W}_{i,:}^{\ell}\|_2 = \lambda_{\ell_o} \sum_{\ell=1}^{L} \sum_{i=1}^{n^{\ell-1}} \sqrt{\sum_{j=1}^{n^{\ell}} \left(W_{ij}^{\ell}\right)^2} \tag{14.8}$$

This is used to regularise the outgoing connections' weights of all the neurons across different layers over the whole network. The conceptual idea of removing all the outgoing weights from neuron O_1^{ℓ} to the neurons in layer $\ell + 1$ therefore removal of herself is illustrated by comparing in Fig. 14.1(b) and 14.1(d). The key idea of

introducing the two regularisers is to embed a dropping mechanism in a deep NN training process. Such a dropping mechanism is guided by the two regularisers.

Convolutional Neural Network

In convolutional neural network, each filter consist of a couple of neurons, the idea is to "drop" the filter thereafter a batch of neurons simultaneously. To integrate over the filters, we reformulate the convolution as a linear operation – an inner-product to be exact. Let $\mathcal{F}^{k^\ell} \in \mathbb{R}^{h \times w \times K^{\ell-1}}$ for $k^\ell = 1, ..., K^\ell$ be the CNN's filters with height h, width w, and $K^{\ell-1}$ channels in the ℓ'th layer. The input to the layer is represented as a 3 dimensional tensor $\mathbf{x} \in \mathbb{R}^{H^{\ell-1} \times W^{\ell-1} \times K^{\ell-1}}$ with height $H^{\ell-1}$, width $W^{\ell-1}$, and $K^{\ell-1}$ channels. Convolving the filters with the input with a given stride s is equivalent to extracting patches from the input and performing a matrix product: we extract $h \times w \times K^{\ell-1}$ dimensional patches from the input with stride s and vectorise these. Collecting the vectors in the rows of a matrix we obtain a new representation for our input $\overline{\mathbf{x}} \in \mathbb{R}^{n \times hwK^{\ell-1}}$ with n patches. The vectorised filters form the columns of the weight matrix $\mathbf{W}^\ell \in \mathbb{R}^{hwK^{\ell-1} \times K_\ell}$. The convolution operation is then equivalent to the matrix product $\overline{\mathbf{x}}\mathbf{W}_\ell \in \mathbb{R}^{n \times K_\ell}$. The columns of the output can be re-arranged to a 3 dimensional tensor $\mathbf{y} \in \mathbb{R}^{H^\ell \times W^\ell \times K^\ell}$ (since $n = H^\ell \times W^\ell$). Pooling can then be seen as a non-linear operation on the matrix $\mathbf{y}$.

Then we introduce a new regulariser over each filter

$$\sum_{k=1}^{K^\ell} \|\mathbf{W}^\ell_{:,k}\|_2 \tag{14.9}$$

The idea can be illustrated in Figure.14.2. Then we define the following regulariser to drop filters in CNN

$$\texttt{lcnn\_regulariser} \triangleq \lambda_{\ell_{cnn}} \sum_{\ell=1}^{L} \sum_{k=1}^{K^\ell} \|\mathbf{W}^\ell_{:,k}\|_2 \tag{14.10}$$

Or equivalently, we regularise the tensor directly. Suppose

$$\mathcal{F}^{k^\ell} \triangleq \left[f^{k^\ell}_{x,y,z}\right] \in \mathbb{R}^{h \times w \times K^{\ell-1}},$$

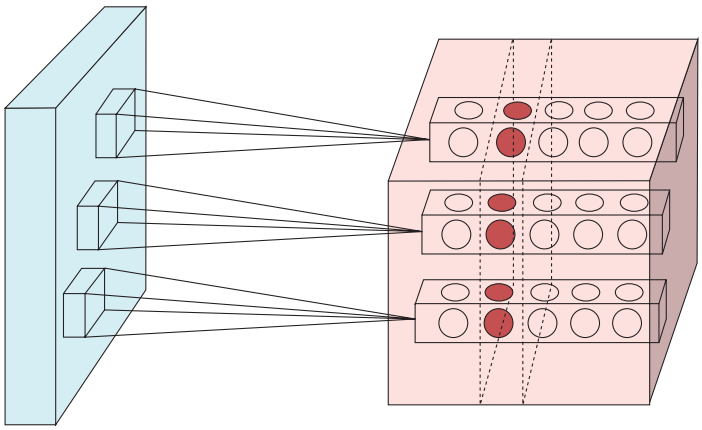


Fig. 14.2 A graphical illustration on the strategy of removing the filters in convolutional neural networks

we have

$$\texttt{lcnn\_regulariser} \triangleq \lambda_{\ell_{cnn}} \sum_{\ell=1}^{L} \sum_{k^{\ell}=1}^{K^{\ell}} \sqrt{\sum_{x=1}^{h} \sum_{y=1}^{w} \sum_{z=1}^{K^{\ell-1}} \left(f_{x,y,z}^{k^{\ell}}\right)^2}. \tag{14.11}$$

Recurrent Neural Network

The RNN dynamics can be described using deterministic transitions from previous to current hidden states. The deterministic state transition is a function

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \rightarrow h_t^{l}.$$

For classical RNNs, this function is given by

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\} .$$

Given an input sequence $\boldsymbol{x} = (x_1, \ldots, x_T)$, a standard recurrent neural network (RNN) computes the hidden vector sequence $\boldsymbol{h} = (h_1, \ldots, h_T)$ and output vector sequence $\boldsymbol{y} = (y_1, \ldots, y_T)$ by iterating the following equations from $t = 1$ to T:

$$h_t = \mathcal{H}\left(\mathbf{W}_{xh} x_t + \mathbf{W}_{hh} h_{t-1} + b_h\right) \tag{14.12}$$

$$y_t = \mathbf{W}_{hy} h_t + b_y \tag{14.13}$$

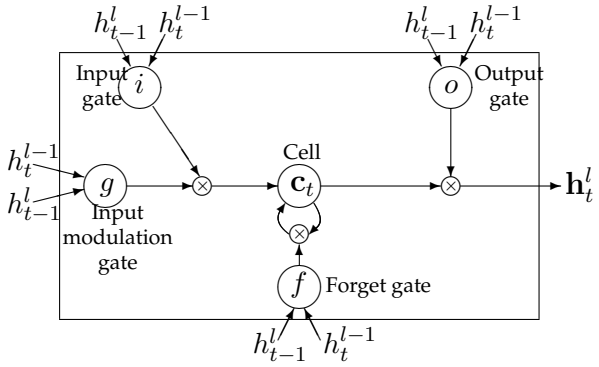


Fig. 14.3 A graphical representation of LSTM memory cells (there are minor differences in comparison to Graves [70]).

where the W terms denote weight matrices (e.g. $\mathbf{W}_{xh}$ is the input-hidden weight matrix), the b terms denote bias vectors (e.g. b_h is hidden bias vector) and $\mathcal{H}$ is the hidden layer function.

The LSTM has complicated dynamics that allow it to easily "memorize" information for an extended number of timesteps. The "long term" memory is stored in a vector of *memory cells* $c_t^l \in \mathbb{R}^n$. Although many LSTM architectures that differ in their connectivity structure and activation functions, all LSTM architectures have explicit memory cells for storing information for long periods of time. The LSTM can decide to overwrite the memory cell, retrieve it, or keep it for the next time step. The LSTM architecture used in our experiments is given by the following equations Graves [70]:

$$\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l \tag{14.14}$$

$$i_t^\ell = \text{sigm}\left(\mathbf{W}_{xi}^\ell h_t^{\ell-1} + \mathbf{W}_{hi} h_{t-1}^\ell + \mathbf{W}_{ci}^\ell c_{t-1}^\ell + b_i^\ell\right) \tag{14.15}$$

$$f_t^\ell = \text{sigm}\left(\mathbf{W}_{xf}^\ell h_t^{\ell-1} + \mathbf{W}_{hf}^\ell h_{t-1}^\ell + \mathbf{W}_{cf}^\ell c_{t-1}^\ell + b_f^\ell\right) \tag{14.16}$$

$$o_t^\ell = \text{sigm}\left(\mathbf{W}_{xo}^\ell h_t^{\ell-1} + \mathbf{W}_{ho}^\ell h_{t-1}^\ell + \mathbf{W}_{co}^\ell c_t^{\ell-1} + b_o^\ell\right) \tag{14.17}$$

$$g_t^\ell = \tanh\left(\mathbf{W}_{xc}^\ell h_t^{\ell-1} + \mathbf{W}_{hc}^\ell h_{t-1}^\ell + b_c^\ell\right) \tag{14.18}$$

$$c_t^\ell = f_t^\ell \odot c_{t-1}^\ell + i_t^\ell \odot g_t^\ell \tag{14.19}$$

$$h_t^\ell = o_t^\ell \odot \tanh(c_t^\ell) \tag{14.20}$$

In these equations, sigm and $\tanh$ are applied element-wise. Figure 14.3 illustrates the LSTM equations.

The idea of compressing recurrent neural networks is inspired from *Model Reduction* technique in Control Theory [238]. By a reduction of the model's associated state space dimension or degrees of freedom, an approximation to the original model is computed. This reduced-order model can then be evaluated with lower accuracy but in significantly less time. A schematic illustration is showed in Fig.14.4

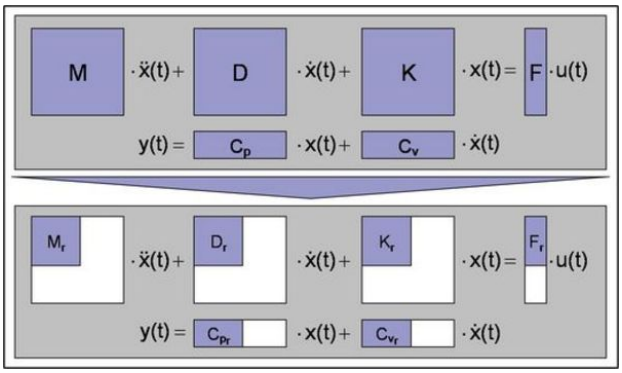


Fig. 14.4 A graphical illustration on the model reduction technique in control theory

Drop States First, we try to regularise the number of hidden states. We first define the following matrices

$$\begin{aligned}
\mathbf{W}_i^\ell &= \begin{bmatrix} \mathbf{W}_{xi}^\ell & \mathbf{W}_{hi}^\ell & \mathbf{W}_{ci}^\ell & b_i^\ell \end{bmatrix} \\
\mathbf{W}_f^\ell &= \begin{bmatrix} \mathbf{W}_{xf}^\ell & \mathbf{W}_{hf}^\ell & \mathbf{W}_{cf}^\ell & b_f^\ell \end{bmatrix} \\
\mathbf{W}_o^\ell &= \begin{bmatrix} \mathbf{W}_{xo}^\ell & \mathbf{W}_{ho}^\ell & \mathbf{W}_{co}^\ell & b_o^\ell \end{bmatrix} \\
\mathbf{W}_c^\ell &= \begin{bmatrix} \mathbf{W}_{xc}^\ell & \mathbf{W}_{hc}^\ell & b_c^\ell \end{bmatrix}
\end{aligned} \tag{14.21}$$

and introduce a new term

$$\sum_{i=1}^{n} \left\| \begin{bmatrix} \mathbf{W}_i^\ell[i,:] & \mathbf{W}_f^\ell[i,:] & \mathbf{W}_o^\ell[i,:] & \mathbf{W}_c^\ell[i,:] \end{bmatrix} \right\|_2 \tag{14.22}$$

This is used to regularise the neurons in i_t^ℓ, f_t^ℓ, o_t^ℓ, g_t^ℓ respectively.

Then we concatenate the four matrices into a compact one

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_i^\ell & \mathbf{W}_f^\ell & \mathbf{W}_o^\ell & \mathbf{W}_c^\ell \end{bmatrix}. \tag{14.23}$$

The new regulariser can be defined as follows

$$\texttt{lrnn\_state\_regulariser} \triangleq \lambda_{\ell_{rnn}} \sum_{\ell=1}^{L} \sum_{i=1}^{n} \|\mathbf{W}_{\ell}\left[i,:\right]\|_2. \tag{14.24}$$

Drop Feedback Next, we try to regularise the *incoming* feedback from the states

$$\mathbf{W}_x^{\ell} = \begin{bmatrix} \mathbf{W}_{xi}^{\ell} \\ \mathbf{W}_{xf}^{\ell} \\ \mathbf{W}_{xo}^{\ell} \\ \mathbf{W}_{xc}^{\ell} \end{bmatrix}, \quad \mathbf{W}_h^{\ell} = \begin{bmatrix} \mathbf{W}_{hi}^{\ell} \\ \mathbf{W}_{hf}^{\ell} \\ \mathbf{W}_{ho}^{\ell} \\ \mathbf{W}_{hc}^{\ell} \end{bmatrix}, \quad \mathbf{W}_c^{\ell} = \begin{bmatrix} \mathbf{W}_{ci}^{\ell}, \\ \mathbf{W}_{cf}^{\ell}, \\ \mathbf{W}_{co}^{\ell}. \end{bmatrix}. \tag{14.25}$$

We then can introduce the new term for the neurons at layer ℓ

$$\sum_{j=1}^{n} \left(\|\mathbf{W}_x^{\ell}\left[:,j\right]\|_2 + \|\mathbf{W}_h^{\ell}\left[:,j\right]\|_2 + \|\mathbf{W}_c^{\ell}\left[:,j\right]\|_2 \right). \tag{14.26}$$

The three terms are used to regularise the neurons in $h_t^{\ell-1}$, h_{t-1}^{ℓ} and c_{t-1}^{ℓ} respectively.

The new regulariser can be defined as follows

$$\texttt{lrnn\_feedback\_regulariser} \triangleq \lambda_{\ell_{rnn}} \sum_{\ell=1}^{L} \sum_{j=1}^{n} \left(\|\mathbf{W}_x^{\ell}\left[:,j\right]\|_2 + \|\mathbf{W}_h^{\ell}\left[:,j\right]\|_2 + \|\mathbf{W}_c^{\ell}\left[:,j\right]\|_2 \right). \tag{14.27}$$

14.1.3 Identifiability of Deep Neural Networks

The other challenging yet very important issue will be on the proof of identifiability which is related to Section 5.1 in Chapter 5. Unfortunately, I have no theoretical clue yet. From a practical point of view, the identified model never seems to be unique. Counter examples to uniqueness can be found ubiquitously. However, from a quick implementation using the proposal previously in Section 14.1.2, we proposed the hypothesis that the sparsest deep neural network with fewest active neurons and connection may be unique.

We started with a simple sparse linear regression problem which is a classic problem in compressive sensing or sparse signal recovery. The inputs and outputs were synthetically generated as follows. First, a random feature matrix $\Phi \in \mathbb{R}^{m \times n}$, often overcomplete, was created whose columns are each drawn uniformly from the surface of the unit sphere in $\mathbb{R}^n$. Next, sparse coefficient vectors $x_0 \in \mathbb{R}^n$ are

randomly generated with d nonzero entries. Nonzero magnitudes $\bar{x}_0$ are drawn i.i.d. from an experiment-dependent distribution. Signals are then computed as $y = \Phi x_0 \in \mathbb{R}^m$, and then contaminated by adding noise $\xi \in \mathbb{R}^m$ with certain distribution. i.e., $y = \Phi x_0 + \xi$. In compressive sensing or sparse signal recovery setting, several algorithms will be presented with y and Φ and attempts to estimate x_0. Such training can be formulated by a neural network where an extreme case will be there is only one hidden layer and there is only one neuron on thin layer. Minimisation of a cost function with mean square error as loss and ℓ_1 as regulariser over the weight will typically yield the exact solution if Φ satisfy conditions like restricted isometry property.

Rather than using a single hidden layer and single neuron for training, we specified a multi-layer structure and there are more than one neurons in each layer. To be simple, the activation function is assumed to be linear. Therefore, the training of x_0 is not the main concern under the deep neural network framework but the prediction error for the test set is more interesting. In our experiment, the number of example in training set and test set are the same. We used the standard normalised mean square error (NMSE) metric , i.e. NMSE $= \frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{N} y_t^2}$, to evaluate the prediction accuracies of the models.

It seems that deep neural architecture with multiple layers and many neurons is overly used for this simple example. It should be naturally expected that the prediction error is as small as possible especially after adding regularisation technique such as Dropout. However, the results seems to be counter-intuitive while our method yield impressive performance.

First of all, we set the number of features n to be 20 and there are 2 nonzero elements in x_0. Only one hidden layer is specified, with 5 neurons in this layer. Therefore, $\mathbf{W}^1 \in \mathbb{R}^{20\times 5}$ and the output layer $\mathbf{W}^2 \in \mathbb{R}^5$ .After each layer, we applied Dropout with a keeping probability of 50%. The number of example was set to be 1000 (half for training and half for testing) which is much greater than the number of unknown weight ($20 \times 5 + 5 = 105$). The setup of experiment was as follows: optimizer: `AdamOptimizer`; number of epochs: 100; learning rate: 0.001; batch size: 1; dropout keep probability : 50%.

In all cases, we ran 1000 independent trials to generate different feature matrix and output. As an illustration, we show the training result in one trial where the prediction NMSE using Dropout is the lowest among all the trials. In this trail, the spare vector $x_0 = [0, 0, 3.87308349, 0, 0, 0, 0, 0, 0, -8.23781791, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, where the 3^{rd} and 10^{th} entries are nonzeros. The estimated weights using Dropout

are shown below, both $\mathbf{W}^1$ in (14.28) and $\mathbf{W}^2$ in (14.29) are not sparse and implying a fully connected architecture. The test NMSE is around 0.54.

$$\mathbf{W}^1_{Dropout} = \begin{bmatrix} 0 & 0 & 0 & -0.01453323 & 0.06075698 \\ -0.05635324 & 0 & -0.02587643 & 0.02911515 & -0.01041718 \\ 0.02563882 & 0.05031364 & 0.03376988 & 0.01993434 & -0.03179494 \\ 0.06203449 & 0.02862295 & -0.06700613 & 0.02385385 & -0.02911432 \\ -0.48254526 & -0.35333461 & 0.31129083 & 0.3545627 & -0.31979144 \\ 0.03181251 & -0.07274029 & 0.05249952 & 0.04767575 & -0.02953613 \\ 0.04364435 & 0 & -0.03578129 & -0.03502097 & 0.09711245 \\ -0.04102893 & -0.06275055 & 0 & -0.06409876 & -0.05218389 \\ 0.0200471 & 0.06717232 & 0 & 0.02837713 & 0.03758603 \\ -0.03055474 & 0.0289463 & 0.06301561 & 0.03308195 & 0.01662179 \\ -0.02746344 & 0.07223324 & 0.04476647 & 0.01322776 & 0.04655014 \\ -0.01112585 & 0 & -0.037157 & -0.03381626 & 0.02151454 \\ 0.04563131 & -0.03387317 & -0.04606552 & 0 & 0.01086553 \\ 0.03301461 & -0.02328412 & 0.0114607 & -0.01552058 & 0 \\ -2.000736 & -1.94960344 & 2.02926755 & 1.93757319 & -1.96851373 \\ -0.05102381 & 0.02301042 & -0.07785907 & 0.01081117 & 0.0626013 \\ 0.02743321 & 0.03834696 & 0.06928469 & 0 & 0 \\ -0.04644512 & 0. & -0.01497171 & 0.02810199 & 0 \\ 0.0628076 & 0.04429785 & 0.01758143 & 0.01070064 & -0.02718436 \\ 0 & 0 & -0.0419367 & 0.06928124 & -0.05641071 \end{bmatrix} \tag{14.28}$$

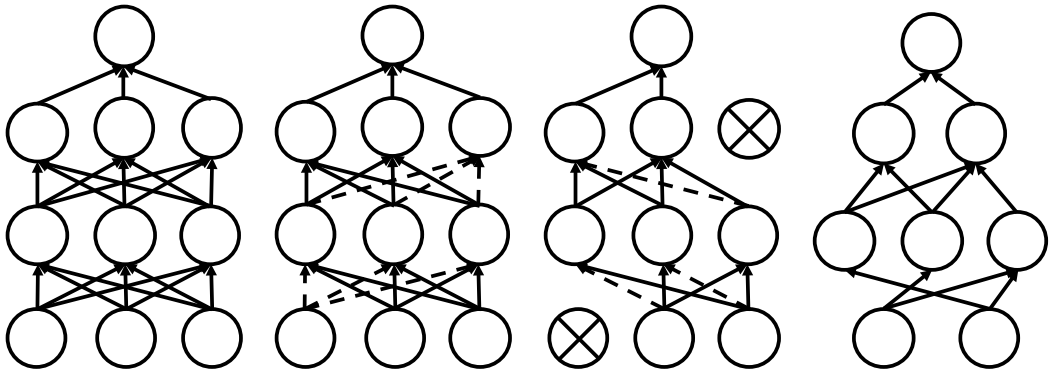
$$\mathbf{W}^2_{Dropout} = \begin{bmatrix} -0.10229997 \\ -0.11288397 \\ 0.11892998 \\ 0.12453081 \\ -0.11404949 \end{bmatrix} \tag{14.29}$$

Using the same data, the training result using DropNeuron can be found below, both $\mathbf{W}^1$ in (14.30) and $\mathbf{W}^2$ in (14.31) are very sparse. In $\mathbf{W}^1$, only two non zeros weights are found, they are $W^1_{3,2} = -0.6687693$ and $W^1_{10,2} = 1.42591035$; and in $\mathbf{W}^2$, there is only one nonzero entry $\mathbf{W}^2_2 = -5.74600601$. The test NMSE is surprisingly low at around 0.00036.

$$\mathbf{W}^1_{DropNeuron} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -0.6687693 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1.42591035 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{14.30}$$

$$\mathbf{W}^2_{DropNeuron} = \begin{bmatrix} 0 \\ -5.74600601 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{14.31}$$

It is a fact that the only two nonzero entries of $\mathbf{W}^1$ both appear in the second column of $\mathbf{W}^1$. This means that only the second neuron in the hidden layer is necessary to be kept while dropping all the other neurons. Similarly, the second neuron in the output layer is necessary to exist. Meanwhile, we notice that $W^1_{3,2} \times W^2_2 = (-0.6687693) \times (-5.74600601) = 3.8427524171$ and $W^1_{10,2} \times W^2_2 = 1.42591035 \times (-5.74600601) = -8.19328944082$, which are very close to the nonzero entry in $x_0 = [0, 0, 3.87308349, 0, 0, 0, 0, 0, 0, -8.23781791, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$. If we investigate the structure of (14.30) and (14.31) again, and consider the effect of linear activation function, the estimated network architecture by dropping unnecessary neurons almost reveal the true additive structure of the third and tenth feature. A conceptual



(a) Full size model (b) Drop Neurons (c) Drop Connections (d) Small size NN

Fig. 14.5 A graphical illustration of DropNeuron strategy in regression problem

Table 14.1 Summary of statistics for Sparse Regression

Regularisation	$\mathbf{W}^{\text{FC1}}\%$	$\mathbf{W}^{\text{FC2}}\%$	$\mathbf{W}^{\text{total}}\%$	NMSE	NMSE (no prune)
ℓ_1+P	58%	100%	60%	0.54	0.54
ℓ_1+DN+P	16.00%	44.47%	54.11%	0.00036	0.00036
Regularisation	$\mathbf{O}^{\text{input}}\%$	$\mathbf{O}^{\text{FC1}}\%$	$\mathbf{O}^{\text{output}}\%$	$\mathbf{O}^{\text{total}}\%$	Compression Rate
ℓ_1+P	$\frac{20}{20} = 100\%$	$\frac{5}{5} = 100\%$	$\frac{1}{1} = 100\%$	$\frac{26}{26} = 100\%$	1.67
ℓ_1+DN+P	$\frac{2}{20} = 10\%$	$\frac{1}{5} = 20\%$	$\frac{1}{1} = 100\%$	$\frac{4}{26} = 15.38\%$	35

illustration for the strategy of dropping neurons for the regression problem can be found in Fig. 14.5.

14.1.4 Training Bayesian Deep Neural Network with Structural Sparsity

Motivated from Algorithm 9, an Algorithm on training Bayesian deep neural network with structural sparsity can be **intuitively** summarised in the following. Due to space limitation, we split the Algorithm into two parts: Algorithm 21 and Algorithm 22. The former one is the initialisation for the latter.

Remark 27 *It should be mentioned again this algorithm is purely from intuition without guarantee either theoretically or experimentally. Nevertheless, it is by no means compatible with the classic backpropagation.*

Algorithm 21 Initialisation for Algorithm 22

1: Cache symbolically the likelihood, loss function and its gradient over $\mathbf{W}_l$ at each layer l (on a tensor graph)

$$\text{Likelihood: } p(\mathbf{y}|\mathbf{W}, \boldsymbol{\theta}) = a(\boldsymbol{\theta}) \cdot \exp\{-E(\mathbf{W}, \boldsymbol{\theta})\}$$

$$\text{Loss function: } f(\mathbf{W}, \boldsymbol{\gamma}, \boldsymbol{\theta}) = E(\mathbf{W}, \boldsymbol{\theta}) + \lambda \sum_{l=1}^{L} \mathbf{W}_l^\top \mathbf{B}_l^\top \boldsymbol{\Gamma}_l^{-1} \mathbf{B}_l \mathbf{W}_l$$

$$\text{Gradient function: } \tilde{\mathbf{g}}_l(\mathbf{W}, \boldsymbol{\gamma}, \boldsymbol{\theta}) = \nabla_{\mathbf{W}_l} f(\mathbf{W}, \boldsymbol{\gamma}, \boldsymbol{\theta}) \text{ using Backgropagation}$$

where we fix $\lambda = 1$ or select $\lambda \in \mathbb{R}^+$ as trail and error which may be empirically helpful;

2: Initialise the unknown hyperparameter $\boldsymbol{\gamma}_l$, i.e., $\boldsymbol{\gamma}_l^1$, as an arbitrary positive vector such as unit vector;

3: Fix/given the known parameter of the exponential family $\boldsymbol{\theta} = \boldsymbol{\theta}^*$;

4: Initialise the Hessian matrix $\tilde{\mathbf{H}}_l^{1,1}$ as an arbitrary positive definite matrix such as Identity $\mathbf{I}$ or be calculated explicitly with $\nabla\nabla_{\mathbf{W}} f(\mathbf{W}, \boldsymbol{\gamma}, \boldsymbol{\theta})$ given $\mathbf{W}^{1,1}$ and $\boldsymbol{\theta}^*$;

5: Set $\text{epoch}_{\text{swtich}}$ as a natural number to switch from Gradient Descent method to Quasi-Newton method;

- *For simplicity, $\mathbf{B}$ is assumed to be identity matrix. The key difference with classic backpropagation lies in line 13 to 27. If this part is removed, the algorithm downgraded to the typical backpropagation using stochastic gradient descent or/and quasi-Newton method.*

- *Similar to prior specification and algorithmic manipulation in Chapter 6 and 7, the $\mathbf{B}$ matrix can be defined structurally as well. It is particularly promising for modelling sequence data such as time series data using recurrent neural network.*

- *The Bayesian theoretic grounded explanation could quantify the parameter and prediction uncertainty which will be potentially important for decision making.*

14.1.5 Implementation on Mobile Device Chips

Convolutional neural networks (CNNs) have shown reliable results on real-world applications such as image classification [108, 184, 80] and object detection [65, 161] given adequate computing and memory resources. Concurrent to these progresses, the deployment of CNNs on mobile devices is gaining more and more attention [102, 220, 57]. One of the most critical issues in mobile applications of CNNs is that mobile devices have strict constraints in terms of computing power, battery,

Algorithm 22 Training Bayesian Deep Neural Network with Structural Sparsity

Execute Algorithm 21 for initialisation;
for epoch $= 1, \ldots,$ epoch$_{\max}$ **do**
  Gradient Descent method update for backpropagation, e.g., SGD, ADAM, etc;
  for $\tau = 1, \ldots, \tau_{\max}$ **do**
    for $l = 1, \ldots, L$ **do**
      Choose fixed step size $\hat{\alpha}_l^{\text{epoch},\tau}$ or via line search under Wolfe condition;
      $\Delta \mathbf{W}_l^{\text{epoch},\tau} = -\hat{\alpha}_l^{\text{epoch},\tau} \tilde{\mathbf{g}}_l(\mathbf{W}_l^{\text{epoch},\tau}, \boldsymbol{\gamma}_l^{\text{epoch}}, \boldsymbol{\theta}^*)$;
      $\mathbf{W}_l^{\text{epoch},\tau+1} = \mathbf{W}_l^{\text{epoch},\tilde{\tau}} + \Delta \mathbf{W}_l^{\text{epoch},\tilde{\tau}}$;
    end for
  end for
  if epoch > epoch$_{\text{swtich}}$ **then**
    Switch to Quasi-Newton method update for backpropagation, e.g. L-BFGS;
    for $\tilde{\tau} = 1, \ldots, \tilde{\tau}_{\max}$ **do**
      for $l = 1, \ldots, L$ **do**
        Fix step size $\tilde{\alpha}_l^{\text{epoch},\tau}$ or line search under Wolfe condition;
        $\Delta \mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}} = -\tilde{\alpha}_l^{\text{epoch},\tilde{\tau}} (\tilde{\mathbf{H}}_l^{\text{epoch},\tilde{\tau}})^{-1} \tilde{\mathbf{g}}_l(\mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}}, \boldsymbol{\gamma}_l^{\text{epoch}}, \boldsymbol{\theta}^*)$;
        $\mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}+1} = \mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}} + \Delta \mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}}$;
        Compute the new gradient $\tilde{\mathbf{g}}_l(\mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}+1}, \boldsymbol{\gamma}_l^{\text{epoch}}, \boldsymbol{\theta}^*)$;
        $y_l^{\text{epoch},\tilde{\tau}} = \tilde{\mathbf{g}}_l(\mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}+1}, \boldsymbol{\gamma}_l^{\text{epoch}}, \boldsymbol{\theta}^*) - \tilde{\mathbf{g}}_l(\mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}}, \boldsymbol{\gamma}_l^{\text{epoch}}, \boldsymbol{\theta}^*)$;
        Approximate $\tilde{\mathbf{H}}_l^{\text{epoch},\tilde{\tau}+1}$ using $y_l^{\text{epoch},\tilde{\tau}}, \Delta \mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}}, \tilde{\mathbf{H}}_l^{\text{epoch},\tilde{\tau}}$;
        Approximate $(\tilde{\mathbf{H}}_l^{\text{epoch},\tilde{\tau}+1})^{-1}$ using $y_l^{\text{epoch},\tilde{\tau}}, \Delta \mathbf{W}_l^{\text{epoch},\tilde{\tau}+\tau_{\max}}, \tilde{\mathbf{H}}_l^{\text{epoch},\tilde{\tau}}$;
      end for
    end for
    $\mathbf{C}_l^{\text{epoch}} = \left(\tilde{\mathbf{H}}_l(\mathbf{W}^{\text{epoch},\tau_{\max}+\tilde{\tau}_{\max}}, \boldsymbol{\theta}^*)\right)^{-1}$;
    $\alpha_{l_i}^{\text{epoch}+1} = -\frac{\mathbf{B}_{l_i,:}\mathbf{C}^{\text{epoch}}\mathbf{B}_{l_i,:}^\top}{(\gamma_{l_i}^{\text{epoch}})^2} + \frac{1}{\gamma_{l_i}^{\text{epoch}}}$;
    $\mathbf{W}_l^{\text{epoch}+1} = \mathbf{W}_l^{\text{epoch},\tilde{\tau}_{\max}}$;
    $\gamma_{l_i}^{\text{epoch}+1} = \frac{\left|\mathbf{B}_{l_i,:}\mathbf{W}^{\text{epoch}+1}\right|}{\sqrt{\alpha_{l_i}^{\text{epoch}+1}}}, \quad \boldsymbol{\Gamma}_l^{k_{\text{epoch}+1}} = \text{diag}(\boldsymbol{\gamma}_l^{\text{epoch}+1})$
  end if
  if a stopping criterion is satisfied **then**
    Break;
  end if
end for

and memory capacity. Thus, it is imperative to obtain CNNs tailored to the limited resources of mobile devices.

On mobile applications, it is typically assumed that training is performed on the server and test or inference is executed on the mobile devices [41, 57]. To improve test-time performance on mobile devices, a line of recent research efforts have focused on binarising/quantising CNNs. Courbariaux et al. [41] introduce neural networks with binary weights and activations at run-time, which they call binarised neural networks (BNNs). Rastegari et al. [159] propose two efficient approximations to standard convolutional neural networks: Binary-Weight-Networks and XNOR-Networks, where the former one binarizes weights and the latter one binarizes both weights and activations. Zhou et al. [239] propose DoReFa-Net, which trains convolutional neural networks that not only have low bitwidth weights and activations, but also using low bitwidth parameter gradients. During the forward pass, these network architectures drastically reduce memory size and accesses, and replace most arithmetic operations with bitwise operations, which is expected to substantially improve power-efficiency. Semiconductor manufacturers like IBM [57] and Intel [204, 237] have been involved in the research and development of related chips. However, these works usually cause severe prediction accuracy degradation when quantising weights and activations to less than 4-bit numbers, especially on complex tasks such as classification on CIFAR-100 or ImageNet.

Related Work

Quantized Neural Networks: High precision parameters are not very important in achieving high performance in deep networks. Recent research efforts [39, 91, 237] have considerably reduced computaion complexity by using low bitwidth weights and low bitwidth activations. Zhou et al. [239] further generalized these schemes and proposed to train CNNs with low bitwidth gradients. Their method, called DoReFa-Net, perform well when bitwidth is larger than 4. However, with a smaller bitwidth, the performance of their highly quantized networks (e.g.,binarised) deteriorates rapidly due to the destructive property of binary quantization. In fact, Hubara et al. [91] experiments with different combinations of bit-widths for weights and activations, and shows 4-bit quantized CNN can achieve comparable accuracy as their 32-bit counterpart. However, large performance degradation occurs when quantising weights and activations to 2-bit numbers.

Binarised/Ternarized Neural Networks: The binary representation for deep models is not a new topic. At the very beginning of neural network, inspired biolog-

ically, the unit step function has been used as the activation function [194, 17]. It has been known that binary activation can use spiking response to provide event-based computation and communication (consuming energy only when necessary) and therefore is energy-efficient [57]. Recently, Courbariaux et al. introduce Binarised-Neural-Networks (BNNs) [41, 40], neural networks with binary weights and activations at run-time. Different from Courbariaux et al.'s work, Rastegari et al. [159] introduce simple, efficient, and accurate approximations to CNNs by binarising the weights and even the intermediate representations in CNNs. Their binarization method aims at finding the best approximations of the convolutions using binary operations. Esser et al. [57] ternarize CNNs and implement them on IBM TrueNorth chip. All these works drastically reduce memory consumption, and replace most arithmetic operations with bitwise operations, which potentially lead to a substantial increase in power-efficiency.

Proposed Strategy

A potential strategy is proposed to introduce sparsity into the network architecture. The weight-values and activations of CNNs are ternary $(\{-1, 0, +1\})$ and binary $(\{0, +1\})$ respectively, which means convolutions can be implemented by only addition, subtraction (without multiplication) and dropping the connections whose weight values are zeros. We further introduce a kernel regularizer to intensify the neuron sparsity during the training process. We demonstrate that moderate sparsity leads to only mild accuracy degration while requiring a significantly less memory and fewer connections.

14.2 Future Direction II: Decision Making in Neuroscience

14.2.1 Cognitive Design Principles for Real-Time Decision Making using Neural Big Data

Decision making is pervasive in nature. It occurs whenever an animal makes a choice from several alternatives on the basis of a subjective value that it places on them. For a long time, this study has been axiomatic, addressing the question of "How should one choose when faced with uncertain outcomes?"

For a large part of the 20th century, research on human choice was dominated by economic theories, particularly *rational choice* and *revealed preferences* theories. This approach starts from a limited set of properties that are imposed on choices (rationality axioms). It then determines to what extent choices can be represented by the maximisation of some latent mathematical function, typically referred to as the *utility* or *value function* [205]. This led to the development of utility theory, more specifically, expected utility theory and mean-variance analysis (portfolio analysis). Apart from theoretical development, behavioural finance also starts from choice, but it rejects the idea that choice reflects the maximisation of a rational (expected) utility function. Instead it places emphasis on one important feature common to the axiomatic approach: the agent chooses "as if" maximising utility. But how can we determine where choice really comes from? Or more specifically, was there any biological foundation to the economic theories of choice? Which neural circuity was involved? What algorithms were employed? Recent advances in neuroscience technology are enabling a deeper understanding of the cognitive processes involved in decision making. Nowadays, neuroscience measurements allow us to capture data underpinning the entire process of decision making, from initial perception of a "stimulus" (which conveys new information and/or new investment options), to valuation and motivation, and the very act of choosing. Some fascinating results provide supporting evidence, including the discovery of, and subsequently, ability to manipulate, the very value of (utility) signals that constitute the core of the axiomatic theory [155, 133, 154, 60]. So far, neurobiology only played a supporting role in the quest for a better understanding of human behaviour in investment decision making, helping to differentiate between existing valuation models, or elucidating the biophysical mechanics and implementation algorithms behind human economic decision making. However, in recent years evidence has emerged that there is

significant neurobiological variation that does not map into parametric variation of even the best economic models [149, 64, 168, 164]. Such neurobiological variation could be used to identify potential behavioural variation that would be otherwise missed if one were to follow economic theory alone. Investigating the neural basis of the decision making under uncertainty will not only provide a mechanistic account of human decision-making but also provide some of the foundations for theories of human behaviour. We will combine whole-brain functional neuroimaging, with functional magnetic resonance imaging (fMRI), electroencephalography (EEG), eye-tracking, and mathematical models to study the neural basis of the human decision making process under various conditions.

14.2.2 Background

(Sub)cortical network provides evidence for understanding neural basis in decision making

Different valuation systems map to different networks In the computations involved in making a choice, behaviour can be driven by different valuation systems. These systems can operate in the domain of rewards (that is, appetitive outcomes) and punishments (that is, aversive outcomes). Although the exact nature and number of valuation systems is still being debated, conceptually the Pavlovian, habitual and goal-directed systems provide a useful operational division of the valuation problem according to the style of the computations that are performed by each. More and more evidence is showing that the neural basis of these three distinct valuation systems is established on the complex connectivity among (sub)cortical areas rather than specific area alone. **In Pavlovian systems**, a network that includes the basolateral amygdala, the ventral striatum and the orbitofrontal cortex underlie the learning processes through which neutral stimuli become predictive of the value of outcomes [33, 88]. Specifically, the central nucleus of the amygdala, through its connections to the brainstem nuclei and the core of the nucleus accumbens, seems to be involved in nonspecific preparatory responses, whereas the basolateral complex of the amygdala seems to be involved in more specific responses through its connections to the hypothalamus and the periaqueductal grey. In contrast to Pavlovian systems, which value only a small set of responses, **habit systems** can learn to assign values to stimulus-response associations (which indicate the action that should be taken in a particular state of the world) on the basis of previous experience, through a process of trial-and-error. Studies using several species and methods suggest that

the dorsolateral striatum might play a crucial part in the control of habits [14, 221]. Furthermore, it has been suggested that stimulus-response representations might be encoded in cortico-thalamic loops [221]. Finally, in contrast to the habit system, **goal-directed systems** assigns values to actions by computing action-outcome associations and then evaluating the rewards that are associated with the different outcomes. Several lines of evidence from these various methods also point to an involvement of the basolateral amygdala and the mediodorsal thalamus (which, in combination with the dorsolateral prefrontal cortex, form a network that Balleine has called the "associative cortico-basal-ganglia loop" [14]).

Modelling and analysis from a control-theoretic perspective

There are two main features of these brain networks related to valuation, especially in the sense of control theory: **(a)** the networks are dynamical systems, that is, the (hidden) neuronal dynamics are evolving over time; **(b)** the networks are causal, that is, dynamics in one neuronal population influence the dynamics in another and these interactions can be modulated by experimental manipulations or endogenous brain activity. In the view of these characteristics, the best option is to **use a control-theoretic approach to model and analyse the neuronal dynamics**.

Mathematical modelling. *System Identification* is the term used in the Control Community for the area of constructing mathematical models of dynamical systems from measured input/output signals [117]. Theories and methodologies for such model construction have been developed in many different research communities (to some extent independently). For example, the term *Machine Learning* has become very common in recent years. In the cognitive neuroscience community, *Dynamic Causal Modelling* (DCM), which is used to model brain responses and provides estimates of neurobiologically interpretable quantities such as the effective strength of synaptic connections among neuronal populations and their context-dependent modulation, has gradually become part of mainstream neuroimaging analysis techniques [99]. In the seminal paper by Prof. K. Friston *et al.* [99], DCM was firstly introduced as "a fairly standard nonlinear system identification procedure using Bayesian estimation of the parameters of deterministic input-state-output dynamic systems". Its applications cover a wide range of domains in cognitive neuroscience, including language, motor processes, vision and visual attention, memory, perceptual decision making, and learning.

Analysis. The design of feedback control strategies is the main subject of control theory and engineering which has been instrumental for the efficient and safe op-

eration of a plethora of technological devices. Control theory and engineering has also started to provide important tools useful in various biological applications. One important example is the discovery of negative feedback and oscillator motifs, which encode control strategies for stabilisation and synchronisation of systems respectively, from bacterial gene networks to *C. elegans* neuronal networks [4]. DCM has been using (linear or nonlinear) differential equations for describing (hidden) neuronal dynamics. In this work, we propose to extend DCM into a framework allowing researchers to elucidate control strategies or design principles in neuroscience.

Discovery of the neural basis using whole-brain functional neuroimaging dynamics

What is the neural basis of decision-making and how are decisions implemented in the human brain? This question is hard to address in humans, because opportunities to record the activity of single human neural cells are extremely limited. Typically, instead of single neuron time series data whole-brain functional neuroimaging, together with functional magnetic resonance imaging (fMRI) and electroencephalography (EEG) are used. These tools are used for two reasons. Firstly, they allow us to test predictions about the brain regions involved in a given task and their interactions. Secondly, these methods allow us to validate our computational models, since we can test whether quantities that are computed by specific models (such as choice value or prediction error signals) can be observed in human participants performing the task in the scanner.

It also should be noted that DCM was introduced in 2003 for fMRI data [99] and made available as open-source software within the Statistical Parametric Mapping (SPM) software. The mathematical basis and implementation of DCM for fMRI have since been refined and extended repeatedly. DCM have also been implemented for EEG and MEG as well [47, 100].

Links between subject measures and cognitive behaviour

More and more evidence has shown that there are correlations between subject measures and cognitive behaviour, beyond the decision making process. **Subject measures** include one or more of the followings: demographics (age, sex, income, education level, drug use, etc.), psychometrics (IQ, language performance, etc.) and other behavioural measures such as 'rule-breaking behaviour'. In a recent published study in *Nature Neuroscience* [174], relationships between individual subjects'

functional connectomes and 280 behavioural and demographic measures were investigated relating imaging to non-imaging data from 461 subjects in the Human Connectome Project. The study suggested that there is a link between a specific pattern of brain connectivity and the demographic and psychometric measures.

Another interesting and important measure is the **eye movement** (blink frequency, blink duration, fixation frequency, fixation duration, pupil diameter, and horizontal vergence). Advertisers use data about where human subjects look and when to better capture attention visually. Designers employ it to improve products. Game and phone developers utilise it to offer the latest in hands-free interaction. Years of research have found that our tiny, rapid eye movements called saccades serve as a window into the brain for psychologists as well as for advertisers. Saccades can be used to elucidate our inner mental and neurological disorders, such as autism, attention-deficit hyperactivity disorder, Parkinson's disease, etc. [199]. In [106, 107], the research suggests that eye-fixations actually guide the comparison process. Using eye-tracking, they have shown that a simple extension of drift-diffusion models popular in psychology, which allows for the integration process to be biased towards the item being fixated, is able to explain the psychometric and eye-tracking data with remarkable quantitative and qualitative procession.

14.2.3 Hypothesis and Objectives

Hypotheses

Subject variations in human decision making are determined by brain networks connectivity variations. Such biological variations can be reflected by the subject measures variations. The Human Connectome Project shows that the brain network connectivity can be explained by functional neuroimaging data, such as fMRI, EEG, etc. In this project, we will test the possibility of inferring brain networks connectivity variations from the large amount of functional neuroimaging and eye movement data.

General objectives

The main objectives of this project are:

- **O1.** Development of generic theoretical approaches and algorithms for inferring brain effective connectivity using functional neuroimaging data.

- **O2.** Application of these methods to discover the neural basis underlying decision making process, e.g., computation of values in simple choice.

- **O3.** Design and implementation of easy-to-use software packages that implement the developed methods and minimise the level of expertise needed for their use.

14.2.4 Problems and Plan

Overview

Our broad strategy is to model the dynamics of data obtained from fMRI, EEG, eye-tracking combined with psychophysical test to study the neurobiological basis of decision making. This will be realised through a series of packages (**P**). In **P 1**, we will collect functional neuroimaging data, i.e., fMRI and EEG, as well as eye movement data while the subject is performing psychophysical tasks. From **P 2** to **P 4**, we will develop the theoretical tools and algorithms for data processing, dynamic modelling and model analysis. In **P 5**, we will engineer the developed algorithms based on an *Apache Spark*™ framework. In **P 6**, we will use *deep learning* to infer relationships between subject measures and dynamical neural networks.

P 1: Experiment design, data collection and case study

There are several research groups studying how the human brain makes decision using a combination of behavioural studies, eye-tracking, EEG, fMRI and mathematical model. Most of their studies are focusing on certain brain areas instead of on dynamical neural networks. Furthermore, these neuroimaging tools are mainly used to identify where in the brain such functions are located, rather than to characterise how a particular cognitive function is implemented in the brain. Nevertheless, these approaches and the associated discoveries provide valuable and insightful knowledge. In particular, the behaviour and psychophysics experiment design is a key first step. Decision making behaviour experiment is primarily carried out in the visual domain. Participants are typically asked to see faint, grainy images and asked to either detect whether a 'signal' (typically an oriented pattern) is present or absent with manual button presses, sometimes recording eye movements as well. The Rangel Group at Caltech and the Bossaerts Group at Melbourn also share the stimulus set to facilitate future experiment replication.

A potential starting point is to repeat these psychophysics experiments, however, with measuring the whole-brain functional neuroimaging dynamics simultaneously. However considering the financial and time cost, such exhaustive replication is implausible. A recent publication by Larsen and O'Doherty [110] demonstrated the use of a combined EEG and fMRI approach during a simple binary choice task to study the temporal aspects of valuation and choice in humans. The results speak to the time course of engagement of different brain areas, with an initial recruitment of posterior cortical areas, and a successive shift during choice processes to more anterior areas. The fact that additional signals emerged later in time in the dorsomedial prefrontal cortex suggests that this area might support post-decision action-selection rather than decision *per se*. These discoveries illuminated on the temporal dynamics of decision-making in the brain, suggested a distributed architecture for valuation in a highly coordinated way. In another simple binary choice task, eye-tracking experiments were carried out to characterise the properties of the value comparison process so as to select the best options in making a choice. Their research results suggested that eye-fixations actually guide the comparison process when the subjects looked back and forth between options in order to make a choice, long after the identity of the options was known [106]. Similar results were discovered for three choices task [107]. It is known that eye fixation is part of visual attention and generated by the participation of many brain areas including most of the early visual processing area [94]. Visual information enters the primary visual cortex via the lateral geniculate to the superior colliculus. From there, visual information progresses along two parallel hierarchical streams, 'dorsal stream' and 'ventral stream'. Not surprisingly, these areas are greatly overlapping with the ones in decision making. A integrative, systems-level investigation into the complex wiring of potential effective brain areas may help answer the question: where and how values are compared in order to make a choice.

P 1.1 In this work package, we plan to repeat the behaviour and eye tracking experiments in [106, 107]. EEG and fMRI data will be simultaneously recorded and pre-processed using standard software, e.g. [110]. The functional neuroimaging data acquisition procedure is standard but highly technical. These experiments will be implemented through a collaboration with related research groups where the 512 Hz EEG data were acquired using a XYZ1™ 128+2 channel cap system with eight flat-type active electrodes (six facials, two mcastoids). At the start of each recording session, each connection was stable with offsets within a $\pm$25 mV range. Data were recorded unreferenced and unfiltered with ACTIVIEW software. The fMRI data

were acquired using a XYZ2™ a 3T scanner. Scan parameters were optimized to obtain robust signals in vmPFC, but also to allow whole brain coverage: 45 slices recorded at a 30° angle, repetition time (TR) = 2 s, echo time (TE) = 28 ms, voxel size $3\times 3\times 3.35$ mm, 440 volumes for each of the three experimental sessions. In addition, a 1 mm isotropic T1-weighted structural scan was acquired for each participant to enable localization of the activations.

P 1.2 In this work package, single-channel EEG will be recorded along with multi-channel EEG as described in **P 1.1**. It is known that single-channel EEG is far more easy to popularise in home based application compared with multi-channel EEG. However, one of the major disadvantages of single-channel EEG lies in the limited amount of information it can acquire compared with multi-channel EEG. These problems limit the application of single-channel EEG. In order to overcome the low dimensionality difficulty of single-channel EEG data, EEG data will be recorded simultaneously using multi-channel EEG and single-channel EEG from the same subject, and inferring label/features from multi-channel EEG data. Such labels/features will be introduced in **P 6**. This will allow us to apply the knowledge obtained from multi-channel EEG by observing single-channel EEG only.

P 2: EEG subcortical source localisation

While there is a growing body of fMRI evidence implicating a corpus of brain regions in value-based decision-making in humans, the limited temporal resolution of fMRI cannot address the relative temporal precedence of different brain regions in decision-making. To address this question, EEG data is our main concern for modelling purpose. fMRI data were also acquired from the same participants for source localisation as in [110].

EEG measures the potential differences on the scalp yielded by volume currents within the brain. It means the current sources responsible for the electromagnetic activity are inside the brain and, therefore, **hidden**. In other words, the location and magnitude of the current sources needs to be estimated from measurement. In this project, such inverse problem, termed as brain source reconstruction, will be addressed. More specifically, we will use a large number of fixed dipoles that fill the search space (the grey matter surface for example) and estimate their amplitude [45]. The solution is based on the linear mapping between the dipole moments for a fixed set of dipoles distributed inside the brain and a set of signals recorded by electrodes/gradiometers placed outside the head. This relation is given by

$Y = LJ + \epsilon$ where the sources J are mapped to channels through the subject-specific leadfield matrix L, with data Y and noise ϵ [45].

There are three facts associated with this work package regarding the data obtained in **P 1**. First, prior knowledge needs to be incorporated, e.g., different smoothing functions, medical knowledge, fMRI priors [63]. Second, as previously pointed out in the Background section, many functional brain areas related to decision making are located in the subcortical region. Therefore, a more complex head model is needed to cover this region, e.g., head models based finite element model instead of the canonical single shell head model, which is extensively used for simulations and provided with the SPM toolbox. Complex head models introduce more unknown sources locations, i.e., equivalent current dipole. It means the column size is much greater than the row size of the leadfield matrix L, which makes the inverse problem ill-posed. Third, when data from different subjects are grouped together for mix-effect regression, the size of the data will grow [83]. The consequence of the second and third fact is the introduction of a large leadfield matrix in both dimensions.

Method In the view of the first fact, the best option is to **use Variational Bayesian scheme with the Free Energy as a cost function**. It allows one to implement most popular EEG inversion schemes (Minimum Norm, LORETA, etc.) within the same generic Bayesian framework. It also provides a cost-function in terms of the variational Free energy - an approximation to the marginal likelihood or evidence of the solution. The key ingredient is the specification of the prior covariance of source activity. This prior covariance accommodates the basic distinctions between commonly employed regularisation schemes in the source reconstruction literature, and is generalised by the use of multiple and sparse spatial priors. However, the second and third challenge may become a bottleneck for the application of the Bayesian inversion scheme. The update rules for the hyperparameters depend on computing the posterior weight covariance matrix, which requires an inverse operation (in fact, Cholesky decomposition) of order $\mathcal{O}(M^3)$ in complexity and $\mathcal{O}(M^2)$ in memory storage, with M the number of free parameters. Meanwhile for large data sets, with computation scaling approximately in $\mathcal{O}(N^3)$, with N the number of observations, the algorithm becomes prohibitively expensive to run. Since the same variational Bayesian approach will also be applied in the nonlinear system identification algorithm (e.g., DCM), the details of a potential solution will be introduced in the next work package.

P 3: Development of automatic nonlinear system identification algorithms for big data and big networks

As described in the **Background** section, DCM seems to offer the best theoretical approach to address the dynamical modelling or system identification problem. However, it may not be the appropriate tool for the modern functional neuroimaging data and brain network from several aspects. There are **three key features/challenges** of such time series data, i.e., high dimensionality ('big data'), large scale ('big network') and nonlinearity. The dimensionality, or the complexity, grows with the sample size, and "ultra-high" refers to the case where the dimensionality increases at a more-than-polynomial rate. Scale, or size, refers to the dimension of the system, i.e., the number of state variables. Though large network discovery using DCM has been considered, Since DCMs are Bayesian in all aspects, i.e., each parameter is constrained by a prior distribution and Bayesian inversion not only provides posterior densities for each model parameter but also yields an approximation to the log model evidence, the 'cubic complexity' facts exist in DCM as well. Meanwhile, as pointed out in [99], "a further extension would be to go beyond bilinear approximations to allow for interactions among the states". However, such extension is not trivial since additional model structures, particularly nonlinear structures, would need to be introduced. Furthermore, the number of Bayesian model selection will grow exponentially with the number of candidate model structures. Traditionally, proposal for a candidate hypothetical model structure needs careful tuning and deep domain knowledge about the brain system under study, in order to reduce the model search space. However, unlike linear structure, the possibility of nonlinear structure can be huge. These challenges associated with DCM yield a need for the development of **automatic nonlinear system identification algorithms for big data and big network**.

Method In [135], W. Pan derived a repository of nonlinear system identification algorithms using a Empirical Bayesian framework. By analysing the log-evidence cost function, the cost function can be reformulated as a nonconvex cost function in both parameter and hyper-parameter space. Thus, a convex-concave procedure can be carried out iteratively. It leads to a series of iterative reweighted convex relaxation schemes for connecting these algorithms to popular algorithms including Lasso, Group-Lasso, Generalised-Lasso, Fused-Lasso and Graphical-Lasso. The Alternating Direction Method of Multipliers [27] framework can be seamlessly integrated as a distributed optimisation strategy to address high dimensionality and large scale

problems. As a by-product, the source localisation problem in **P 2** can be addressed using the same set of algorithms.

P 4: Analysis of dynamical system using system control theory

In this work package, we plan to do a holistic analysis on the dynamical model obtained from **P 3** using mathematical tools from control theory. Once the dynamical model is obtained, many of the natural control-theoretic questions that one would normally pose for such a system are precisely those that leading neuroscientists are asking, if sometimes in a different language: What is special about the information-processing capabilities, or input/output behaviours, of such networks, and how does one characterise these behaviours? How does one estimate time-varying internal states, such as the concentrations of proteins and other chemical substances, from input/output experiments (observer problem)? What subsystems appear repeatedly? Where lie the main sensitivities affecting robustness of the system? What is the reason that there are cascades and feedback loops? More generally, what can one say, if anything, about stability, oscillations, and other dynamical properties of such complex systems? These questions, from a control-theoretic perspective, are well-addressed in the literature, for example, top control journals such as Automatica or IEEE Transactions on Automatic Control. Extensive literature review and analysis will be performed. It should be mentioned that these questions are typically asked by control engineers when designing machines such as cars, missiles, etc. These performance metrics provide important labels/features to quantitatively characterise the system. For example, robustness, the ability to maintain performance in the face of perturbations and uncertainty should be universal from brain to engineering systems.

P 5: Open source software development

In this work package, we plan to create and distribute easy-to-use software aimed at a broad scientific audience. The algorithm we developed in **P 2** and **P 3** are designed to deal with big data and based on distributed optimisation algorithm. The distributed computation platform based on **Apache Spark™** framework, within our own Data Science Institute at Imperial College London, offers the possibility for implementing these algorithms. We will code a set of *Python* scripts and algorithms that can be used by non-mathematically trained neuroscientists to help them define and solve their mathematical modelling problems based on measured functional

neuroimaging data. We will then develop an intuitive and easy-to-use graphical user interface around these algorithms in order to automatically pre-process data, identify optimal solutions and present the results to the end-user. To ensure efficient code development, we will take advantage of some open source library such as the Machine Learning library and the Convex Optimization library.

P 6: Link between subject measures and dynamical networks to preference

In this work package, we will investigate the underlying relationships between different subject measures and different dynamical network patterns.

P 6.1 A natural choice of method for investigating underlying relationships between two sets of variables is canonical correlation analysis (CCA) [90], a procedure that seeks maximal correlations between combinations of variables in both sets. At the first stage of this project, we will use CCA to estimate pairs of canonical variates along which sets of subject measures and patterns of brain connectivity co-vary in a similar way across subjects.

P 6.2 In this work package, we want to model the performance metrics in **P 4** using a machine learning approach. Three sets of features will be constructed. We will use subject measures as the first set of features, i.e., demographics (age, sex, income, education level, drug use, etc.), psychometrics (IQ, language performance, etc.) and other behavioural measures such as 'rule-breaking behaviour'. Then we will use the eye movement measures as the second set of features, i.e., blink frequency, blink duration, fixation frequency, fixation duration, pupil diameter, and horizontal vergence. Finally, we will perform a time-frequency analysis to the single-channel EEG we collected in **P 1.2** using complex Morlet wavelets to extract features that capture the mixture of frequencies and their interrelations at different points in time as features. These EEG related features constitute to be the third set. To achieve a good prediction performance, we will employ **Deep Learning** [19].

References

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

[2] Abur, A. and Exposito, A. G. (2004). *Power system state estimation: theory and implementation*. CRC Press.

[3] Akaike, H. (1974). A new look at the statistical model identification. *Automatic Cimperialthesis.bibontrol, IEEE Transactions on*, 19(6):716–723.

[4] Alon, U. (2007a). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall/CRC.

[5] Alon, U. (2007b). *An introduction to systems biology: design principles of biological circuits*, volume 10. CRC press.

[6] Amelunxen, D., Lotz, M., McCoy, M. B., and Tropp, J. A. (2014). Living on the edge: Phase transitions in convex programs with random data. *Information and Inference*, page iau005.

[7] Arkin, A., Ross, J., and McAdams, H. H. (1998). Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected escherichia coli cells. *Genetics*, 149(4):1633–1648.

[8] Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, pages 337–404.

[9] Babacan, S., Molina, R., and Katsaggelos, A. (2010). Bayesian compressive sensing using laplace priors. *Image Processing, IEEE Transactions on*, 19(1):53–63.

[10] Babacan, S. D. (2009). *Bayesian techniques for image recovery*. PhD thesis, Northwestern University.

[11] Bach, F. R. and Jordan, M. I. (2004). Learning graphical models for stationary time series. *Signal Processing, IEEE Transactions on*, 52(8):2189–2199.

[12] Bai, E. (1998). An optimal two-stage identification algorithm for hammerstein–wiener nonlinear systems. *Automatica*, 34(3):333–338.

[13] Baillie, R. T. and Chung, S.-K. (2002). Modeling and forecasting from trend-stationary long memory models with applications to climatology. *International Journal of Forecasting*, 18(2):215–226.

[14] Balleine, B. W. (2005). Neural bases of food-seeking: affect, arousal and reward in corticostriatolimbic circuits. *Physiology & behavior*, 86(5):717–730.

[15] Barahona, M. and Poon, C. (1996). Detection of nonlinear dynamics in short, noisy time series. *Nature*, 381(6579):215–217.

[16] Barber, D. and Cemgil, A. (2010). Graphical models for time-series. *Signal Processing Magazine, IEEE*, 27(6):18–28.

[17] Barlett, P. L. and Downs, T. (1992). Using random weights to train multilayer networks of hard-limiting units. *IEEE Transactions on Neural Networks*, 3(2):202–210.

[18] Baxter, M. and King, R. G. (1999). Measuring business cycles: approximate band-pass filters for economic time series. *Review of economics and statistics*, 81(4):575–593.

[19] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127.

[20] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828.

[21] Billings, S. A. (2013). *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons.

[22] Bishop, C. (2006). *Pattern Recognition and Machine Learning*, volume 4. Springer New York.

[23] Bloomfield, P. (1992). Trends in global temperature. *Climatic change*, 21(1):1–16.

[24] Bloomfield, P. and Nychka, D. (1992). Climate spectra and detecting climate change. *Climatic Change*, 21(3):275–287.

[25] Box, G. E., Jenkins, G. M., and Reinsel, G. C. (2011). *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons.

[26] Boyd, S., El Ghaoul, L., Feron, E., and Balakrishnan, V. (1987). *Linear matrix inequalities in system and control theory*, volume 15. Society for Industrial Mathematics.

[27] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

[28] Boyd, S. and Vandenberghe, L. (2004). *Convex optimisation*. Cambridge university press.

[29] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

[30] Candès, E., Romberg, J., and Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223.

[31] Candès, E. and Tao, T. (2005). Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215.

[32] Candès, E., Wakin, M., and Boyd, S. (2008). Enhancing sparsity by reweighted ℓ_1 minimisation. *Journal of Fourier Analysis and Applications*, 14(5):877–905.

[33] Cardinal, R. N., Parkinson, J. A., Hall, J., and Everitt, B. J. (2002). Emotion and motivation: the role of the amygdala, ventral striatum, and prefrontal cortex. *Neuroscience & Biobehavioral Reviews*, 26(3):321–352.

[34] Cerone, V., Piga, D., and Regruto, D. (2011). Enforcing stability constraints in set-membership identification of linear dynamic systems. *Automatica*, 47(11):2488–2494.

[35] Chen, T., Andersen, M., Ljung, L., Chiuso, A., and Pillonetto, G. (2014). System identification via sparse multiple kernel-based regularization using sequential convex optimization techniques. *Automatic Control, IEEE Transactions on*, 59(11):2933–2945.

[36] Chen, T., Ohlsson, H., and Ljung, L. (2012). On the estimation of transfer functions, regularizations and gaussian processes—revisited. *Automatica*, 48(8):1525–1535.

[37] Christiano, L. J. and Fitzgerald, T. J. (2003). The band pass filter*. *international economic review*, 44(2):435–465.

[38] Cochrane, J. H. (2009). *Asset Pricing*. Princeton university press.

[39] Courbariaux, M., Bengio, Y., and David, J.-P. (2014). Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*.

[40] Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131.

[41] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

[42] Craigmile, P. F., Guttorp, P., and Percival, D. B. (2004). Trend assessment in a long memory dependence model using the discrete wavelet transform. *Environmetrics*, 15(4):313–335.

[43] Cui, Y., Surpur, C., Ahmad, S., and Hawkins, J. (2016). A comparative study of htm and other neural network models for online sequence learning with streaming data. In *Proceedings of the International Joint Conference on Neural Networks*.

[44] Dai, W. and Milenkovic, O. (2009). Subspace pursuit for compressive sensing signal reconstruction. *Information Theory, IEEE Transactions on*, 55(5):2230–2249.

[45] Dale, A. M. and Sereno, M. I. (1993). Improved localizadon of cortical activity by combining eeg and meg with mri cortical surface reconstruction: a linear approach. *Journal of cognitive neuroscience*, 5(2):162–176.

[46] Darmois, G. (1935). Sur les lois de probabilitéa estimation exhaustive. *CR Acad. Sci. Paris*, 260(1265):85.

[47] David, O., Kiebel, S. J., Harrison, L. M., Mattout, J., Kilner, J. M., and Friston, K. J. (2006). Dynamic causal modeling of evoked responses in eeg and meg. *NeuroImage*, 30(4):1255–1272.

[48] De Brabanter, K., De Brabanter, J., De Moor, B., and Gijbels, I. (2013). Derivative estimation with local polynomial fitting. *The Journal of Machine Learning Research*, 14(1):281–301.

[49] Ding, S. X. (2008). *Model-based fault diagnosis techniques: design schemes, algorithms, and tools*. Springer.

[50] Donoho, D. (2006). Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306.

[51] Donoho, D. and Elad, M. (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202.

[52] Donoho, D. and Huo, X. (2001). Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862.

[Donoho and Stodden] Donoho, D. and Stodden, V. Breakdown point of model selection when the number of variables exceeds the number of observations. *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1916–1921.

[54] Donoho, D. L., Stodden, V. C., and Tsaig, Y. (2007). About SparseLab. *Online accessible: http://sparselab.stanford.edu*.

[55] Elowitz, M. and Leibler, S. (2000). A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–8.

[56] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the Econometric Society*, pages 987–1007.

[57] Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D. R., et al. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, page 201604850.

[58] Evgeniou, T., Pontil, M., and Poggio, T. (2000). Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1–50.

[59] Ferry, M., Razinkov, I., and Hasty, J. (2011). Microfluidics for synthetic biology: from design to execution. *Methods in enzymology*, 497:295.

[60] Figner, B., Knoch, D., Johnson, E. J., Krosch, A. R., Lisanby, S. H., Fehr, E., and Weber, E. U. (2010). Lateral prefrontal cortex and self-control in intertemporal choice. *Nature neuroscience*, 13(5):538–539.

[61] Figueiredo, M. A. and Bioucas-Dias, J. M. (2010). Restoration of poissonian images using alternating direction optimization. *Image Processing, IEEE Transactions on*, 19(12):3133–3145.

[62] Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

[63] Friston, K., Harrison, L., Daunizeau, J., Kiebel, S., Phillips, C., Trujillo-Barreto, N., Henson, R., Flandin, G., and Mattout, J. (2008). Multiple sparse priors for the m/eeg inverse problem. *NeuroImage*, 39(3):1104–1120.

[64] Frydman, C., Camerer, C., Bossaerts, P., and Rangel, A. (2011). Maoa-l carriers are better at making optimal financial decisions under risk. *Proceedings of the Royal Society of London B: Biological Sciences*, 278(1714):2053–2059.

[65] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[66] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323.

[67] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

[68] Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438.

[69] Grant, M., Boyd, S., and Ye, Y. (2008). CVX: MATLAB software for disciplined convex programming. *Online accessiable: http://cvxr.com*.

[70] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[71] Greenland, S. and Longnecker, M. P. (1992). Methods for trend estimation from summarized dose-response data, with applications to meta-analysis. *American journal of epidemiology*, 135(11):1301–1309.

[72] Haber, R. and Unbehauen, H. (1990). Structure identification of nonlinear dynamic systems: survey on input/output approaches. *Automatica*, 26(4):651–677.

[73] Hamilton, J. D. (1994). *Time Series Analysis*, volume 2. Princeton University Press.

[74] Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*.

[75] Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143.

[76] Hansen, L. P. (1982). Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the Econometric Society*, pages 1029–1054.

[77] Hansen, L. P. and Singleton, K. J. (1982). Generalized instrumental variables estimation of nonlinear rational expectations models. *Econometrica: Journal of the Econometric Society*, pages 1269–1286.

[78] Hassibi, B. and Stork, D. G. (1993). *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann.

[79] Hastie, T. and Tibshirani, R. (2009). *The Elements of Statistical Learning*, volume 2. Springer.

[80] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[81] Hegland, M. (2007). Approximate maximum a posteriori with gaussian process priors. *Constructive Approximation*, 26(2):205–224.

[82] Heinrich, G., Ludwig, M., Qian, J., Kubala, B., and Marquardt, F. (2011). Collective dynamics in optomechanical arrays. *Physical review letters*, 107(4):043603.

[83] Henson, R. N., Wakeman, D. G., Litvak, V., and Friston, K. J. (2011). A parametric empirical bayesian framework for the eeg/meg inverse problem: generative models for multi-subject and multi-modal integration. *Frontiers in human neuroscience*, 5.

[84] Hines, P., Balasubramaniam, K., and Sanchez, E. C. (2009). Cascading failures in power grids. *Potentials, IEEE*, 28(5):24–30.

[85] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[86] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[87] Hodrick, R. J. and Prescott, E. C. (1997). Postwar us business cycles: an empirical investigation. *Journal of Money, credit, and Banking*, pages 1–16.

[88] Holland, P. C. and Gallagher, M. (2004). Amygdala–frontal interactions and reward expectancy. *Current opinion in neurobiology*, 14(2):148–155.

[89] Horn, R. and Johnson, C. (1990). *Matrix analysis*. Cambridge university press.

[90] Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, pages 321–377.

[91] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*.

[92] Hunter, D. R. and Lange, K. (2004). A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37.

[93] Ingolia, N. T. and Weissman, J. S. (2008). Systems biology: reverse engineering the cell. *Nature*, 454(7208):1059–1062.

[94] Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194–203.

[95] Ji, S., Xue, Y., and Carin, L. (2008). Bayesian compressive sensing. *Signal Processing, IEEE Transactions on*, 56(6):2346–2356.

[96] Jin, J., Pan, W., Pham, D. L., Yuan, Y., Tomlin, C. J., Webb, A., and Goncalves, J. (2016). On identification of dynamical structure functions: A sparse bayesian learning approach. *arXiv preprint arXiv:1605.09543*.

[97] Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

[98] Kaltenbach, H.-M., Dimopoulos, S., and Stelling, J. (2009). Systems analysis of cellular networks under uncertainty. *FEBS letters*, 583(24):3923–3930.

[99] karl j friston, lee m harrison, w. d. p. (2003). Dynamic causal modelling. 19(4):1273–1302.

[100] Kiebel, S. J., Garrido, M. I., Moran, R., Chen, C.-C., and Friston, K. J. (2009). Dynamic causal modeling for eeg and meg. *Human brain mapping*, 30(6):1866–1876.

[101] Kim, S.-J., Koh, K., Boyd, S., and Gorinevsky, D. (2009). ℓ_1 trend filtering. *Siam Review*, 51(2):339–360.

[102] Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. (2015). Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

[103] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[104] Kollar, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. The MIT Press.

[105] Koopman, B. O. (1936). On distributions admitting a sufficient statistic. *Transactions of the American Mathematical society*, 39(3):399–409.

[106] Krajbich, I., Armel, C., and Rangel, A. (2010). Visual fixations and the computation and comparison of value in simple choice. *Nature neuroscience*, 13(10):1292–1298.

[107] Krajbich, I. and Rangel, A. (2011). Multialternative drift-diffusion model predicts the relationship between visual fixations and choice in value-based decisions. *Proceedings of the National Academy of Sciences*, 108(33):13852–13857.

[108] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[109] Kundur, P., Balu, N. J., and Lauby, M. G. (1994). *Power system stability and control*, volume 4. McGraw-hill New York.

[110] Larsen, T. and O'Doherty, J. P. (2014). Uncovering the spatio-temporal dynamics of value-based decision-making in the human brain: a combined fmri–eeg study. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 369(1655):20130473.

[111] LeCun, Y., Denker, J. S., Solla, S. A., Howard, R. E., and Jackel, L. D. (1989). Optimal brain damage. In *NIPs*, volume 89.

[112] Leontaritis, I. and Billings, S. (1985). Input-output parametric models for non-linear systems part i: deterministic non-linear systems. *International journal of control*, 41(2):303–328.

[113] Leser, C. (1961). A simple method of trend construction. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 91–107.

[114] Levitt, S. D. (2004). Understanding why crime fell in the 1990s: Four factors that explain the decline and six that do not. *Journal of Economic perspectives*, pages 163–190.

[115] Link, W. A. and Sauer, J. R. (1994). Estimating equations estimates of trends. *Bird Populations*, 2:23–32.

[116] Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

[117] Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall.

[118] Ljung, L., Hjalmarsson, H., and Ohlsson, H. (2011). Four encounters with system identification. *European Journal of Control*, 17(5):449.

[119] Lofberg, J. (2004). Yalmip: A toolbox for modeling and optimization in MATLAB. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289. IEEE.

[120] Lucas, R. E. (1980). Two illustrations of the quantity theory of money. *The American Economic Review*, pages 1005–1014.

[121] Lygeros, J., Johansson, K. H., Simic, S. N., Zhang, J., and Sastry, S. S. (2003). Dynamical properties of hybrid automata. *Automatic Control, IEEE Transactions on*, 48(1):2–17.

[122] Mankiw, N. (2014). *Principles of macroeconomics*. Cengage Learning.

[123] Materassi, D. and Salapaka, M. V. (2012). On the problem of reconstructing an unknown topology via locality properties of the wiener filter. *Automatic Control, IEEE Transactions on*, 57(7):1765–1777.

[124] Menolascina, F., Fiore, G., Orabona, E., De Stefano, L., Ferry, M., Hasty, J., di Bernardo, M., and di Bernardo, D. (2014). In-vivo real-time control of protein expression from endogenous and synthetic gene networks. *PLoS Comput Biol*, 10(5):e1003625.

[125] Mohajerin Esfahani, P., Vrakopoulou, M., Andersson, G., and Lygeros, J. (2012). A tractable nonlinear fault detection and isolation technique with application to the cyber-physical security of power systems. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 3433–3438.

[126] Mosheiov, G. and Raveh, A. (1997). On trend estimation of time-series: a simple linear programming approach. *Journal of the operational research society*, 48(1):90–96.

[127] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

[128] Narendra, K. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *Neural Networks, IEEE Transactions on*, 1(1):4–27.

[129] Needell, D. and Vershynin, R. (2009). Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of computational mathematics*, 9(3):317–334.

[130] Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325.

[131] Osborn, D. R. (1995). Moving average detrending and the analysis of business cycles†. *Oxford Bulletin of Economics and Statistics*, 57(4):547–558.

[132] Ozay, N., Sznaier, M., Lagoa, C. M., and Camps, O. I. (2012). A sparsification approach to set membership identification of switched affine systems. *Automatic Control, IEEE Transactions on*, 57(3):634–648.

[133] Padoa-Schioppa, C. and Assad, J. A. (2006). Neurons in the orbitofrontal cortex encode economic value. *Nature*, 441(7090):223–226.

[134] Palmer, J., Wipf, D., Kreutz-Delgado, K., and Rao, B. (2005). Variational EM algorithms for non-Gaussian latent variable models. *Advances in neural information processing systems*, 18:1059.

[135] Pan, W. (2015). *Sparse Inference of Nonlinear Dynamical Systems from Time Series Data*. PhD thesis, Imperial College London.

[136] Pan, W., Sootla, A., and Stan, G.-B. (2014a). Distributed Reconstruction of Nonlinear Networks: An ADMM Approach. *The International Federation of Automatic Control Cape Town, South Africa*. arXiv:1403.7429.

[137] Pan, W., Yuan, Y., Dai, W., Ellis, T., Gonçalves, J., Barahona, M., and Stan, G.-B. (2015 (in preparation)). Learning the Nonlinear Structure of Large-Scale Dynamical Systems from Noisy Observations. *Physical Review Letters*.

[138] Pan, W., Yuan, Y., Gonçalves, J., and Stan, G.-B. (2012). Reconstruction of Arbitrary Biochemical Reaction Networks : A Compressive Sensing Approach. In *IEEE 51st Annual Conference on Decision and Control (CDC)*. IEEE. arXiv:1205.1720.

[139] Pan, W., Yuan, Y., Gonçalves, J., and Stan, G.-B. (2016). A sparse bayesian approach to the identification of nonlinear state-space systems. *IEEE Transactions on Automatic Control*, 61(1):182–187.

[140] Pan, W., Yuan, Y., Ljung, L., Gonçalves, J., and Stan, G.-B. (2015 (submitted)). Nonlinear Biochemical Reaction Networks Identification From Heterogeneous Datasetss. In *IEEE 54st Annual Conference on Decision and Control (CDC)*. IEEE.

[141] Pan, W., Yuan, Y., Sandberg, H., Gonçalves, J., and Stan, G.-B. (2013). Real-time Fault diagnosis for large-scale nonlinear power networks. In *IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 2340–2345. IEEE.

[142] Pan, W., Yuan, Y., Sandberg, H., Gonçalves, J., and Stan, G.-B. (2015). Online fault diagnosis for nonlinear power systems. *Automatica*, 55:27–36.

[143] Pan, W., Yuan, Y., Sootla, A., and Stan, G.-B. (2014b). Inference of Switched Biochemical Reaction Networks Using Sparse Bayesian Learning. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, page 51.

[144] Paoletti, S., Juloski, A. L., Ferrari-Trecate, G., and Vidal, R. (2007). Identification of hybrid systems a tutorial. *European journal of control*, 13(2):242–260.

[145] Papachristodoulou, A. and Recht, B. (2007). Determining interconnections in chemical reaction networks. In *American Control Conference, 2007. ACC'07*, pages 4872–4877. IEEE.

[146] Pavella, M., Ernst, D., and Ruiz-Vega, D. (2000). *Transient stability of power systems: a unified approach to assessment and control*. Springer.

[147] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

[148] Pelckmans, K., Suykens, J. A., Van Gestel, T., De Brabanter, J., Lukas, L., Hamers, B., De Moor, B., and Vandewalle, J. (2002). Ls-svmlab: a matlab/c toolbox for least squares support vector machines. *Tutorial. KULeuven-ESAT. Leuven, Belgium*.

[149] Pessiglione, M., Seymour, B., Flandin, G., Dolan, R. J., and Frith, C. D. (2006). Dopamine-dependent prediction errors underpin reward-seeking behaviour in humans. *Nature*, 442(7106):1042–1045.

[150] Pillonetto, G., Chiuso, A., and De Nicolao, G. (2011). Prediction error identification of linear systems: a nonparametric gaussian regression approach. *Automatica*, 47(2):291–305.

[151] Pillonetto, G. and De Nicolao, G. (2010). A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93.

[152] Pillonetto, G. and De Nicolao, G. (2011). Kernel selection in linear system identification part i: A gaussian process perspective. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 4318–4325. IEEE.

[153] Pitman, E. J. G. (1936). Sufficient statistics and intrinsic accuracy. In *Mathematical Proceedings of the cambridge Philosophical society*, volume 32, pages 567–579. Cambridge Univ Press.

[154] Plassmann, H., O'Doherty, J., and Rangel, A. (2007). Orbitofrontal cortex encodes willingness to pay in everyday economic transactions. *The Journal of neuroscience*, 27(37):9984–9988.

[155] Platt, M. L. and Glimcher, P. W. (1999). Neural correlates of decision variables in parietal cortex. *Nature*, 400(6741):233–238.

[156] Poggio, T. and Shelton, C. (2002). On the mathematical foundations of learning. *American Mathematical Society*, 39(1):1–49.

[157] Pollock, D. (2000). Trend estimation and de-trending via rational square-wave filters. *Journal of Econometrics*, 99(2):317–334.

[158] Rasmussen, C. E. (2006). Gaussian processes for machine learning.

[159] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer.

[160] Reinsch, C. H. (1967). Smoothing by spline functions. *Numerische mathematik*, 10(3):177–183.

[161] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

[162] Rockafellar, R. (1996). *Convex analysis*, volume 28. Princeton university press.

[163] Ruess, J., Parise, F., Milias-Argeitis, A., Khammash, M., and Lygeros, J. (2015). Iterative experiment design guides the characterization of a light-inducible gene expression circuit. *Proceedings of the National Academy of Sciences*, 112(26):8148–8153.

[164] Rutledge, R. B., Skandali, N., Dayan, P., and Dolan, R. J. (2015). Dopaminergic modulation of decision making and subjective well-being. *The Journal of Neuroscience*, 35(27):9811–9822.

[165] Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer.

[166] Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press.

[167] Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

[168] Set, E., Saez, I., Zhu, L., Houser, D. E., Myung, N., Zhong, S., Ebstein, R. P., Chew, S. H., and Hsu, M. (2014). Dissociable contribution of prefrontal and striatal dopaminergic genes to learning in economic games. *Proceedings of the National Academy of Sciences*, 111(26):9615–9620.

[169] Setty, Y., Mayo, A., Surette, M., and Alon, U. (2003). Detailed map of a cis-regulatory input function. *Proceedings of the National Academy of Sciences*, 100(13):7702.

[170] Shahidehpour, M., Tinney, F., and Fu, Y. (2005). Impact of security on power systems operation. *Proceedings of the IEEE*, 93(11):2013–2025.

[171] Shames, I., Teixeira, A. M., Sandberg, H., and Johansson, K. H. (2011). Distributed fault detection for interconnected second-order systems. *Automatica*.

[172] Singleton, K. J. (1988). Econometric issues in the analysis of equilibrium business cycle models. *Journal of Monetary Economics*, 21(2):361–386.

[173] Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P., Hjalmarsson, H., and Juditsky, A. (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724.

[174] Smith, S. M., Nichols, T. E., Vidaurre, D., Winkler, A. M., Behrens, T. E., Glasser, M. F., Ugurbil, K., Barch, D. M., Van Essen, D. C., and Miller, K. L. (2015). A positive-negative mode of population covariation links brain connectivity, demographics and behavior. *Nature neuroscience*, 18(11):1565–1567.

[175] Söderström, T. and Stoica, P. (1988). *System identification*. Prentice-Hall, Inc.

[176] Soloveichik, D., Seelig, G., and Winfree, E. (2010). Dna as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398.

[177] Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, prediction, and search*, volume 81. The MIT Press.

[178] Srebro, N. and Shraibman, A. (2005). Rank, trace-norm and max-norm. In *Learning Theory*, pages 545–560. Springer.

[179] Sriperumbudur, B. K. and Lanckriet, G. R. (2009). On the convergence of the concave-convex procedure. In *NIPS*, volume 9, pages 1759–1767.

[180] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[181] Strelkowa, N. and Barahona, M. (2010). Switchable genetic oscillator operating in quasi-stable mode. *Journal of The Royal Society Interface*, page rsif20090487.

[182] Strogatz, S. (2000). From kuramoto to crawford: exploring the onset of synchronisation in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1):1–20.

[183] Sturm, J. F. (1999). Using sedumi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653.

[184] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[185] Taflove, A. and Hagness, S. C. (2005). *Computational electrodynamics*. Artech house.

[186] Talluri, K. T. and Van Ryzin, G. J. (2006). *The theory and practice of revenue management*, volume 68. Springer Science & Business Media.

[187] Tate, J. E. and Overbye, T. J. (2008). Line outage detection using phasor angle measurements. *Power Systems, IEEE Transactions on*, 23(4):1644–1652.

[188] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

[189] Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.

[190] Tibshirani, R. J., Taylor, J. E., Candes, E. J., and Hastie, T. (2011). *The solution path of the generalized lasso*. Stanford University.

[191] Tipping, M. (2001). Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244.

[192] Tipping, M. and Faul, A. (2003). Fast marginal likelihood maximisation for sparse bayesian models. In *Proceedings of the ninth international workshop on artificial intelligence and statistics*, volume 1.

[193] Toh, K.-C., Todd, M. J., and Tütüncü, R. H. (1999). Sdpt3–a MATLAB software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581.

[194] Toms, D. (1990). Training binary node feedforward neural networks by back propagation of error. *Electronics letters*, 26(21):1745–1746.

[195] Tropp, J. et al. (2004). Greed is good: Algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on*, 50(10):2231–2242.

[196] Tropp, J. et al. (2006). Just relax: Convex programming methods for identifying sparse signals in noise. *Information Theory, IEEE Transactions on*, 52(3):1030–1051.

[197] Tropp, J. and Gilbert, A. (2007). Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666.

[198] Tsay, R. S. (2005). *Analysis of financial time series*, volume 543. John Wiley & Sons.

[199] Tseng, P.-H., Cameron, I. G., Pari, G., Reynolds, J. N., Munoz, D. P., and Itti, L. (2013). High-throughput classification of clinical populations from natural viewing eye movements. *Journal of neurology*, 260(1):275–284.

[200] Van Der Merwe, R. and Wan, E. A. (2001). The square-root unscented kalman filter for state and parameter-estimation. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 6, pages 3461–3464. IEEE.

[201] Vanlier, J., Tiemann, C., Hilbers, P., and van Riel, N. (2013). Parameter uncertainty in biochemical models described by ordinary differential equations. *Mathematical biosciences*, 246(2):305–314.

[202] Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

[203] Varian, H. R. and Repcheck, J. (2010). *Intermediate microeconomics: a modern approach*, volume 7. WW Norton New York.

[204] Venkatesh, G., Nurvitadhi, E., and Marr, D. (2016). Accelerating deep convolutional networks using low-precision and sparsity. *arXiv preprint arXiv:1610.00324*.

[205] Von Neumann, J. and Morgenstern, O. (1947). *Theory of games and economic behavior*. Princeton university press.

[206] Wahba, G. (1990). *Spline models for observational data*, volume 59. Siam.

[207] Wahlberg, B., Boyd, S., Annergren, M., and Wang, Y. (2012). An ADMM algorithm for a class of total variation regularized estimation problems. *16th IFAC Symposium on System Identification*.

[208] Wainwright, M. and Jordan, M. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

[209] Walter, E., Pronzato, L., and Norton, J. (1997). *Identification of parametric models from experimental data*, volume 1. Springer Berlin.

[210] Wiener, N. (1966). Nonlinear problems in random theory. *Nonlinear Problems in Random Theory, by Norbert Wiener, pp. 142. ISBN 0-262-73012-X. Cambridge, Massachusetts, USA: The MIT Press, August 1966.(Paper)*, 1.

[211] Winful, H. G. and Rahman, L. (1990). Synchronized chaos and spatiotemporal chaos in arrays of coupled lasers. *Physical Review Letters*, 65(13):1575.

[212] Wipf, D. and Nagarajan, S. (2010). Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):317–329.

[213] Wipf, D. and Rao, B. (2004). Sparse Bayesian learning for basis selection. *Signal Processing, IEEE Transactions on*, 52(8):2153–2164.

[214] Wipf, D., Rao, B., and Nagarajan, S. (2011). Latent variable bayesian models for promoting sparsity. *Information Theory, IEEE Transactions on*, 57(9):6236–6255.

[215] Wipf, D. P. (2006). *Bayesian methods for finding sparse representations*. PhD thesis, University of California, San Diego.

[216] Wipf, D. P. (2011). Sparse estimation with structured dictionaries. In *Advances in Neural Information Processing Systems*, pages 2016–2024.

[217] Wipf, D. P., Owen, J. P., Attias, H. T., Sekihara, K., and Nagarajan, S. S. (2010). Robust bayesian estimation of the location, orientation, and time course of multiple correlated neural sources using meg. *NeuroImage*, 49(1):641–655.

[218] Wipf, D. P. and Rao, B. D. (2007). An empirical bayesian strategy for solving the simultaneous sparse approximation problem. *Signal Processing, IEEE Transactions on*, 55(7):3704–3716.

[219] Wright, S. and Nocedal, J. (1999). Numerical optimization. *Springer Science*, 35:67–68.

[220] Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828.

[221] Yin, H. H. and Knowlton, B. J. (2006). The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, 7(6):464–476.

[222] Yin, S. and Kaynak, O. (2015). Big data for modern industry: challenges and trend. *Proceedings of the IEEE*.

[223] Yuan, Y. (2012). *Decentralised network prediction and reconstruction algorithms*. PhD thesis, University of Cambridge.

[224] Yuan, Y., Stan, G., Warnick, S., and Goncalves, J. (2011). Robust dynamical network structure reconstruction. *Special Issue on System Biology, Automatica*, 47:1230–1235.

[225] Yue, Z., Thunberg, J., Pan, W., Ljung, L., and Goncalves, J. (2016). Linear dynamic network reconstruction from heterogeneous datasets. *arXiv preprint arXiv:1612.01963*.

[226] Yuille, A. L. and Rangarajan, A. (2003). The concave-convex procedure. *Neural computation*, 15(4):915–936.

[227] Zadeh, L. et al. (1956). On the identification problem. *Circuit Theory, IRE Transactions on*, 3(4):277–281.

[228] Zangwill, W. I. (1969). *Nonlinear programming: a unified approach*. Prentice-Hall Englewood Cliffs, NJ.

[229] Zavlanos, M., Julius, A., Boyd, S., and Pappas, G. (2011). Inferring stable genetic networks from steady-state data. *Automatica*, 47(6):1113–1122.

[230] Zhang, Q., Zhang, X., Polycarpou, M. M., and Parisini, T. (2014). Distributed sensor fault detection and isolation for multimachine power systems. *International Journal of Robust and Nonlinear Control*, 24(8-9):1403–1430.

[231] Zhang, Z. (2012). *Sparse signal recovery exploiting spatiotemporal correlation*. PhD thesis.

[232] Zhang, Z., Jung, T.-P., Makeig, S., and Rao, B. (2013a). Compressed sensing for energy-efficient wireless telemonitoring of noninvasive fetal ecg via block sparse bayesian learning. *Biomedical Engineering, IEEE Transactions on*, 60(2):300–309.

[233] Zhang, Z., Jung, T.-P., Makeig, S., and Rao, B. (2013b). Compressed sensing of eeg for wireless telemonitoring with low energy consumption and inexpensive hardware. *Biomedical Engineering, IEEE Transactions on*, 60(1):221–224.

[234] Zhang, Z. and Rao, B. D. (2011). Sparse signal recovery with temporally correlated source vectors using sparse bayesian learning. *Selected Topics in Signal Processing, IEEE Journal of*, 5(5):912–926.

[235] Zhang, Z. and Rao, B. D. (2012). Extension of sbl algorithms for the recovery of block sparse signals with intra-block correlation. *arXiv preprint arXiv:1201.0862*.

[236] Zhao, S. and Wei, G.-W. (2003). Jump process for the trend estimation of time series. *Computational Statistics & Data Analysis*, 42(1):219–241.

[237] Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. (2017). Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*.

[238] Zhou, K., Doyle, J. C., Glover, K., et al. (1996). *Robust and optimal control*, volume 40. Prentice hall New Jersey.

[239] Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. (2016). Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.