Dynamic Causal Discovery

Submitted by

Ulrich Schaechtle

for the degree of Doctor of Philosophy

provided by

Royal Holloway, University of London

2015

Declaration

I, Ulrich Schaechtle, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed (Ulrich Schaechtle)
Date:

Der Mensch findet sich mitten unter Wirkungen und kann sich nicht enthalten, nach den Ursachen zu fragen; als ein bequemes Wesen greift er nach den nächsten als der besten und beruhigt sich dabei; besonders ist dies die Art des allgemeinen Menschenverstandes.

Man finds himself surrounded by effects and cannot but inquire into the causes. For the sake of convenience he seizes upon the nearest as the best, and contents himself with that. This is especially true of the common run of human understanding.

JOHANN WOLFGANG VON GOETHE, 1821

Abstract

Discovering rich causal models computationally can be key to creating human-like artificial intelligence. Recent research has detached the notion of time from approaches to causal inference and has thus obscured modelling of and inference over dynamic dependencies in causal systems. This thesis investigates how prior knowledge of temporal and dynamic dependencies, even if conceptually unrelated to causal inference, informs both modelling and inference over causal relations. Three novel methods are introduced incrementally and their contribution is positioned in the wider area of causal discovery.

The first method discovers causal relations within high-dimensional tensor data as they are typically recorded in non-experimental databases. The method allows simultaneous inclusion of numerous dimensions within the data analysis such as samples, time and domain variables construed as tensors. It relies on dynamically changing noise but it does not model it explicitly.

This explicit handling of changing noise levels, also known as the phenomenon of heteroskedasticity, is interpreted via a set of functional equations in the second method. This method not only exploits changing noise levels, but also simplifies assumptions made for causal discovery. However, as we expect heteroskedastic noise, it requires latent structural relations and variables for noise which produce heteroskedasticity. But learning such latent concepts begs for the discovery of more expressive models.

The third method addresses the discovery of more complex models by introducing time as an observed entity in the system and builds on probabilistic programming. `gpmem` is a probabilistic programming technique that uses Gaussian Processes and is proposed here to provide a statistical alternative to memoization.

We test all three methods on synthetic and real world data. Real world data-sets range over a variety of domains, for example healthcare, social sciences and biology. For these data-sets we achieve higher accuracy for causal discovery and more expressiveness than the current state-of-the-art. We use adequate and recent benchmarks for comparison.

Acknowledgements

First and foremost I would like to take the opportunity and thank my supervisor Kostas Stathis who has guided and supported my work on this thesis over the past four years. Throughout my PhD studies, Kostas encouraged me to take a new, unbiased perspective on artificial intelligence. His guidance led me onto a truly interesting journey and he helped me overcome all kinds obstacles while on it. I would further like to thank all the members of the DICE group that I had the pleasure to work with and to review and discuss papers with: Özgür Kafalı, Alfonso E. Romero, Paulo Ricca and Bedour Al-Rayes. Özgür and Alfonso kindly offered to read and give valuable feedback on early drafts of my papers providing valuable insights. This is also true for Alberto Paccanaro, who kindly read the first draft of the MCD and FlexCD papers. The DICE group shares its lab space with an interesting mix of people of whom Joe Reddington, Laurence O'Toole and Robert Michael Walsh came to my assistance whenever I needed advice on the proper use of English language.

I would also like to express my gratitude to Vikash K. Mansinghka. My research visit at the MIT Probabilistic Computing Project has fundamentally changed my views on machine learning and artificial intelligence - the discussions and input I got from Vikash left me inspired and full of new ideas. The members of the group were very supportive and I would like to especially thank Alexey Radul for helping me with the Venture Probabilistic Programming Platform and for patiently answering all my questions. I also thank Ben Zinberg, Vlad Firoiu, Taylor Campbell, Baxter Eaves, Feras Saad and Anthony Lu for the great discussions we had during my stay. Finally, this visit would not have been possible if it hadn't been for Josh Tenenbaum, who kindly answered my cold email and put me in touch with Vikash - Thank you.

The work in this thesis has been partially supported by the COMMODITY12 project. In this project I had the opportunity to visit HES-SO Valais in Sierre, Switzerland. I would like to thank Stefano Brumori and Michael I. Schumacher as well as Damien Zufferey for hosting me and providing me with a stimulating research environment.

Prior to joining RHUL, I studied for a master's degree at Imperial College. Here, I would like to acknowledge Aldo Faisal for not only giving a great course on Machine Learning but also for guiding my master's research towards the interface between time series analysis, generative modelling and scalable inference. Francesca Toni was an excellent personal tutor and helped to pave the way to get a funded PhD position. I would like to thank Matthias Brand for guiding me on my first academic research on a professional level, and supporting my first research in machine learning in a group of cognitive psychologists when I did my undergraduate degree in Duisburg. It was here

that I first learned about the process of scientific knowledge discovery, also thanks to Frank P. Schulte who gave an excellent lecture on the value of publishing in scientific journals. I would like to thank Lucas Carstens for all the advice in relation to my PhD studies I would also like to thank Jing Tang for reading my Thesis and for questioning EVERYTHING.

My family has been supportive throughout my journey. Most importantly I would like to thank my mother dearly, but all members of my family who have encouraged me deserve my dearest gratitude. Finally, I would like to thank Sophie. Without her, I would not have been able to pursue this dream.

List of Acronyms

AI Artificial Intelligence

ALS Alternating Least Square

ANM Additive Noise Model

CGM Continuous Glucose Monitoring

CPD Conditional Probability Distributions

DAG Directed Acyclic Graph

DP Dirichlet Process

DBN Dynamic Bayesian Network

FlexCD Flexible Causal Discovery

GDP Gross Domestic Product

GP Gaussian Process

HMC Hybrid (or Hamiltonian) Monte Carlo

ICA Independent Component Analysis

ILP Inductive Logic Programming

LiNGAM Linear Non-Gaussian Acyclic Model

MCMC Markov Chain Monte Carlo

MDP Markov Decision Processes

MLN Markov Logic Networks

MAP Maximum A Posteriori

MH Metropolis-Hastings

MBD Model-based Diagnosis

MCD Multidimensional Causal Discovery

PCA Principal Component Analysis

RJ Reversible Jump

SP Stochastic Process

VHGPR Variational Heteroskedastic Gaussian Process

wrt with respect to

Mathematical Notation

x, x'	Scalars
X	Random variable
$\mathbf{x}$	Column vector
$\mathbf{X}$	Matrix
$\mathcal{X}$	Tensor
$\mathcal{D}$	Data matrix
$\mathbb{E}$	Expectation
$\mathbb{H}$	Entropy
$\mathbb{KL}[P\|\|Q]$	Kullback-Leibler divergence of P and Q
$\mathcal{N}$	(Multivariate-) Gaussian
$\mathcal{GP}$	Gaussian Process
$X \to Y$	X has a causal effect on Y
$\mathcal{U}_{[0,1]}$	Uniform distribution between 0 and 1
μ	Mean
$\boldsymbol{\mu}$	Mean vector
σ	Standard deviation
$\boldsymbol{\Sigma}$	Covariance matrix
$m(x)$	Mean function
$k(x, x')$	Kernel or covariance function
$\mathbf{K} = \mathbf{K}(\mathbf{x}, \mathbf{x})$	Kernel or covariance function on a data vector
SE	Squared exponential covariance function on a data vector
LIN	Linear covariance function on a data vector
PER	Periodic covariance function on a data vector
RQ	Rational quadratic covariance function on a data vector
C	Constant covariance function on a data vector
WN	White noise covariance function on a data vector

$\not\perp\!\!\!\perp$	Statistical dependence
$\perp\!\!\!\perp$	Statistical independence
$\times_n$	n^{th} tensor product
$\mathbf{H}^{\dagger}$	Moore-Penrose pseudo-inverse of matrix $\mathbf{H}$
t	*true*
f	*false*
$\wedge$	Logical and
$\vee$	Logical or
$\mathbf{\Omega}$	Random Variable over an algebraic operator

Contents

1 Introduction

1.1 Observational Data is Everywhere

Daily, 2.5 exabytes (2.5 billion gigabytes) are collected and stored globally (McAfee et al., 2012). The data come from a diverse set of sources and both observation and collection can happen in various ways. This phenomenon was promoted by the increasing popularity of Internet, mobile technology and advancements in sensor technology (Atzori et al., 2010) as well as crowd-based activities (Doan et al., 2011). We are now able to collect data at any time in the most diverse areas of life. Examples include large software companies collecting data about our habits with regards to energy and heating management of our houses, the recording of consumer behaviour, bird enthusiasts collecting data about bird migration and the collection of electronic health records in large hospital owned databases.

Data itself is considered a commodity. Data-driven methodologies such as statistical or Bayesian analysis seem to be a promising way to gain novel knowledge computationally by processing this newly available commodity. Especially in the healthcare sector, there is hope that new and previously unavailable data will improve the situation of both the patients and the healthcare professionals. We can currently observe how data management changes in the healthcare process as health records become electronic and cloud-based. However, exploiting the newly available data proves to be a challenge. The strict requirements and the complex standards that the medical domain imposes on technical innovation are our main motivation for working with healthcare and medical data.

We have experienced the complexity of data analysis and data mining in the medical domain through our participation in the Commodity_{12} project (Kafalı et al., 2013). The Commodity_{12} consortium is an international board of experts developing a personal health system to assist in the provision of continuous and personalised health services to diabetic patients. The system consists of wearable and portable devices that acquire, monitor and communicate physiological data to a centralised database system. The

task of deriving important insights about an individual person's health status from large existing data sets leads directly to some of the questions that this thesis seeks to answer.

1.2 The Role of Causality

More generally, given a domain of applications, the question is how to produce novel, meaningful insights with the newly available data. This question has been attracting interest in the recent years, as the term Big Data (e.g. Lynch, 2008) has become omnipresent in the tech industry. Big Data enthusiasts proclaim that by gathering as much data as possible one can gain insights into problems otherwise impossible to tackle through the traditional method of scientific knowledge gain. Yet, one of the flagship technologies failed spectacularly. Google Flu Trends aimed to predict influenza epidemics with frequency of certain Google searches reporting very promising results (Ginsberg et al., 2009). However, the algorithm failed twice in the recent years. It significantly under-predicted influenza rates during the H1N1 pandemic in 2009 (Cook et al., 2011) and it significantly over-predicted influenza rates in 2012. Numerous explanations and mitigation strategies have been suggested (Huberty, 2014; Lazer et al., 2014). Interestingly, no one pointed towards the crucial discrepancy between a mere association-based analysis as it is performed with Google Flu Trends and a relationship of cause and effect which is one of the fundamental axioms in scientific thinking (e.g Pearl, 2000, epilogue). Were there other causes behind the observed search behaviour? We cannot know.

An awareness of cause and effect relationships is needed, today more than ever because we collect huge amounts of data that can directly affect our lives. Sensor technology is now affordable and therefore accessible to the wider population, people voluntarily track themselves and submit their vital information to centralised computing facilities (Swan, 2013). E-health technologies that are based on distributed electronic health records, summarising not only patient records but also medical sensor measurements, are already in use to tackle the most widespread and most cost-intensive chronic diseases in the world (Kafalı et al., 2013). The advent of such technologies, with the amounts of data they collect in combination with state-of-the-art data analysis (i.e. computational statistics, statistical machine learning, Big Data analysis, data mining), causes technology enthusiasts to promise great progress in healthcare. The enthusiasts promise the imminent emergence of an evidence-based as well as stratified medicine and healthcare system that will provide vastly improved medical treatment for everyone.

This promising view of future medicine is based on a number of misconceptions

about data and available technology. To motivate this work we are going to examine these misconceptions in more detail so that we can later highlight the most important areas where there is still work to be done, and where open problems pose interesting questions and give research directions. We put these misconceptions into the context of current scientific standards and methodologies as well as today's technical possibilities:

Misconception One: Big Data is Valid Medical Evidence. Evidence-based medicine advocates a review of the relevant literature to find systematic studies that report robust results from experiments. These experiments must be conducted with accepted scientific standards such as randomised trials to conclude recommendations that allow to take into account a level of uncertainty (such as confidence or weighting of evidence, e.g. Sackett et al., 1996). However, the vast amount of observational data collected through sensors and electronic health records certainly does not qualify as evidence in this context. The data does not come from randomised trials. Noise levels are typically higher than in a clean experimental setup for example when untrained individuals enter or modify data recordings. This can be the case with wearable sensor technologies when the patient changes position of the device or when untrained nurses enter complex measurements into health records.

Misconception Two: Big Data Analysis can replace Medical Decision Making. Evidence-based medicine requires a doctor to make a logically sound diagnosis or treatment decision based not only on this evidence, but also on her acquired knowledge. This is all the knowledge relevant for a certain case. Arguably the most successful Big Data analysis techniques of the recent works are ensemble methods such as Random Forest (Liaw and Wiener, 2002) and Deep Learning (Bengio, 2009).

These models function as black boxes: the insights that can be gained are limited. The data generating logic or functional mechanisms cannot be inspected and evaluated against readily available knowledge and logical reasoning. Furthermore, models cannot provide qualitative explanations for certain observations such as symptoms that are interpretable by humans and at the same time predict whether an observation will change, that is, a symptom ceases to occur given a change at another part of the model. Such a capability is crucial for data-driven intervention, for example treatment adaptation[1].

[1]The research field of neural symbolic computation (Garcez et al., 2008) aims to form hybrids between machine learning in the form of neural networks and symbolic, logical learning. One of its goals is to extract knowledge from neural networks. This is achieved through integration of what they call the statistical nature of learning and the logical nature of reasoning. This point of view

Avoiding such black boxes and providing logical and interpretable insights into the learned models is the main aim of Inductive Logic Programming (ILP) (e.g. Lavrac and Dzeroski, 1994). ILP provides rules as statements in logic programming that provide prediction of boolean or discrete data points. It has been successfully applied in the biomedical domain (Santos et al., 2012). Yet, to the best of our knowledge we do not know an ILP system that deals with continuous-valued regression problems in a way other than via constraints and artificial discretisation which comes with large numerical inaccuracies in the case of many observations.

Misconception Three: N = All. Collecting the data of the entire population is often unrealistic and more data does not necessarily mean better data. We have no guarantee that sampling bias disappears when setting the number of sampled data as large as possible. In fact, the contrary may be the case since it may be harder to separate and identify the data where the sample reflects the underlying population with random influences only and without systematic bias. In some situations we may have the background knowledge to differentiate biased samples from those that are relevant in our quest to collect as much data as possible.

Furthermore, many researchers argue that "Big Data" often refers to a vast amount of problems related to a large number of very *small data* sets (e.g. Ghahramani, 2015). This becomes a very important problem for stratified medicine: many if not most hospitalised patients suffer from a combination of illnesses as opposed to only one.

This leads to a problem of *small data*: few patients share the exact combinations of conditions and administered drugs, with the interaction between those often unknown. As a consequence, a scenario where N = All is highly unlikely in the medical domain.

Misconception Four: Evidence is all positive. For some experts, for example a doctor, having machine learning-based evidence available does not imply a "take it or leave it situation." Not taking into account evidence with machine learning can have legal consequences. So does a bad result when taking this kind of evidence into account. An informed decision has to be made whether or not we accept evidence generated by machine learning. We therefore argue that only sound logic and uncertainty computations in combination with transparent models can be deployed in many domains, above all the medical domain. This implies a quantification on confidence about all predictions or diagnoses must exist, be

contradicts our view of learning and reasoning in the presence of uncertainty. We argue that both data and knowledge driven inference is always logical.

it in the traditional frequentist or in the Bayesian sense (see e.g. Murphy, 2012, chapters 5 & 6).

1.3 Problem Characteristics

Causal inference is believed to play a significant role in both the way humans think (Körding et al., 2007) and in the process of scientific knowledge acquisition (e.g Pearl, 2000, epilogue). But what is causal inference and what are cause and effect? Diverse kinds of entities are described as causes or effects such as events, features, properties, changes of properties, objects, persons, relationships or instances of relationships (Spirtes et al., 2000, chapter 1). Spirtes and colleagues also state that the logical form can vary from particular (for an instance) to general (at all times) to universal (for all instances at all times).

The traditional method for causal knowledge acquisition from observations is the randomised experiment. Even here, a number of questions arise (e.g. Hyttinen et al., 2013):

- What is the optimal choice of manipulation?
- What is the optimal sequence of experiments to gain causal knowledge efficiently?
- How can one select an experiment that maximises insight given the researcher's background knowledge?

While these problems seem hard already we can characterise gaining scientific knowledge without randomised trials available as even harder. A randomised experiment provides the assumption that causal effects are either not conditioned on means of data acquisition at all or conditioned only in a well-known way. This is not true for observational data. For the rest of this thesis, whenever we say observational data (as opposed to experimental data), we mean data that is not acquired in a randomised trial. Other problems for causal analysis of observational data are unknown systematic confounders[2] and noise. While in an experimental setting one tries to render such influences purely random, that is, they are not systematically affecting a result, we cannot make this assumption for observational data since we do not have the necessary control over the context of the observation. We furthermore cannot make as strong assumptions about

[2]A confounder is defined as: "[...] a third variable that can make it appear (sometimes incorrectly) that an observed exposure is associated with an outcome. In other words, a confounder is an unobserved exposure associated with the exposure of interest and is a potential cause of the outcome of interest. Confounders lead to bias that distorts the magnitude of the relationship between two factors of interest."(Michigan Center for Public Health Preparedness, 2015)

temporal dependencies and feedback loops as we could in an experimentally controlled environment.

These problems render the finding of causal relations in observational data, and relating them to systems of observed and unobserved components over time, a hard task. Sometimes we expect causal influence to emerge between observed entities instantaneously, sometimes over time. In some cases, for example, in diagnosis, we try to find latent and previously unknown concepts as causal influences. We can characterise this problem as hard from a combinatorics perspective[3] too, as there are many observations in the real world that could be in a causal relationship (direct or indirect) to one another hypothetically. As explained above, modern scientific empiricism aims to tackle some of these challenges by designing randomised experiments, based on background knowledge of the problem at hand. We will now give two examples that share such problem characteristics but for which conducting such an experiment is not possible:

1. **Adverse Drug Effects**. We mentioned above that patients often suffer from multiple conditions and are administered numerous kinds of medications as treatment. How do these medications interact? Is it possible that they have a negative effect on the well-being of a patient? This is a prime example of a problem where, despite data being available, an experimental trial is impossible. We cannot conduct a trial for ethical reasons: it would be unethical and unlawful to enrol patients in a study to give them a cocktail of drugs with the aim to observe and test negative, harmful effects on their well-being. We will also struggle to select patients due to the combinatorics of the problem. Finding a sufficient number of patients that all suffer from the same combination of diseases and can be prescribed the same treatments, whilst ensuring that all influences resulting from sampling are random, is unlikely. At the same time, hospitals collect the data of their patients in electronic databases (e.g. Saeed et al., 2002). This means observational data is available as opposed to experimental data.

2. **Development Aid**. Where should a wealthy government or charity organisation invest its money to cause improvement in the third world? Does help developing local businesses or investing in the health infrastructure actually have an effect on the GDP of a poor country? A randomised experimental trial for this problem is clearly impossible. We can gather observations, but causal effects may be delayed in time. For example, an investment into the health infrastructure of a country may not immediately increase the GDP, but it could be that families can focus more on building their wealth instead of caring for the sick. In this case, an effect

[3]For non-sparse causal graphs, this has been characterised as NP-hard (e.g Claassen et al., 2013)

may show up in the data but it will not be an instantaneous one.

We see that there are pressing needs for methods that discover knowledge in domains where there is observational data available. We believe that this does not only motivate the development of automated causal discovery for time series, but it also shows that we are collecting data sets suitable for causal analysis that will eventually show the benefits of this methodology.

1.4 Research Aims

The main aim of this thesis is to introduce a collection of models and algorithms for causal discovery using data generated by complex dynamical systems and processes. Causal discovery seeks to develop algorithms that learn the structure of causal relations from observation. Current work in causal discovery (see Fig. 1.1) does not rely on a notion of time any more[4]. We can modularise the main aim into the following:

- Bridge the gap between probabilistic time series and dynamical systems research and causal discovery. In this context, we seek to introduce temporal dependence in models used for the state-of-the-art in modern causal discovery. Traditionally, the notion of time introduced confusion to the sciences researching causality. The work on graphical models and functional methods for causal discovery has provided criteria to help differentiate cause from effect in settings where an effect emerges instantaneously or where a timestamp is simply not available. The fact that these methods are conceptually independent from time makes it unclear how to deploy them in the context of dynamical systems. We aim to clarify this matter for situations where temporal dependencies are partially known and enter the model explicitly.

- Use prior knowledge about temporal and dynamical dependencies, even if conceptually unrelated to causal inference, to inform both modelling of and inference over causal relations. We aim to do this by providing representations of dynamically changing noise;

- Provide solutions that cater for several kinds of data, generated in various complex dynamic systems and processes. The data can hereby differ by various characteristics namely its dimensionality and size, which aspects of it are observed, how much we know about it as well as what kind of underlying mechanism generated it.

[4]older methodologies depended explicitly on time as a dimension (for examples, see Granger, 1969)

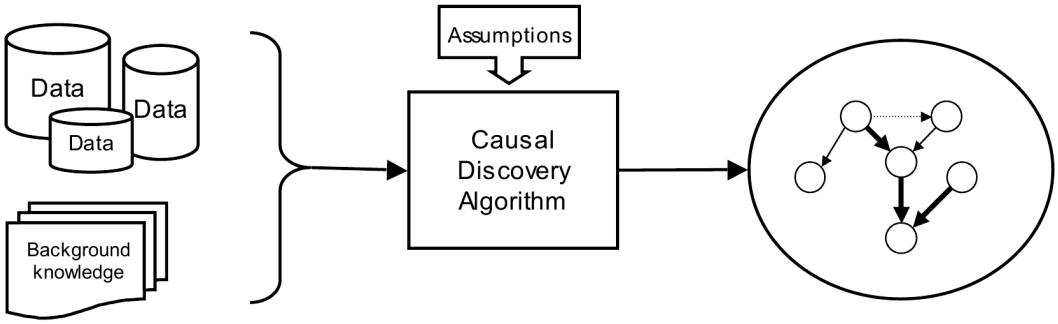


Figure 1.1: A diagram depicting causal discovery(Claassen, 2013).

- Develop computational models for hierarchical, flexible knowledge representation and symbolic reasoning to support causal discovery in the context of complex dynamical systems and processes.

To guarantee sound results, we provide a rigorous axiomatic and principled approach to derive new algorithms, and, where required, provide mathematical proofs.

1.5 Objectives

In order to achieve the aims of the work our objectives are as follows.

- We will study the integration of recent results from modern causal discovery into the state-of-the-art in probabilistic time-series analysis. The focus will be on multi-linear analysis with multidimensional arrays, nonparametric methods with dynamically changing measurement noise and automated symbolic analysis of simple but unstructured time series data.

- We will integrate prior knowledge about temporal and dynamical dependencies into criteria that we deploy to differentiate between cause and effect. We will use this prior knowledge for restrictions and assumptions to derive observable asymmetries between orthogonal causal directions.

- We will examine how data of various kinds, generated in dynamical systems, can be used for causal discovery. Specifically, we will investigate three kinds of observable data. Firstly, we will look at high-dimensional data, as it is recorded in large real-world databases. Secondly, our focus will be on data where noise is dynamically changing over the domain of the cause. Finally, we will study unstructured time series data where latent symbolic causes explain the dynamics of the data set.

- We will test how Venture (Mansinghka et al., 2014) as a generative, probabilistic programming language lives up to the requirements for inference and knowledge representation needed for causal discovery in dynamical systems.

- We will study symbolic reasoning to support causal discovery in the context of complex dynamical systems and processes.

The above objectives have in common that they create hard problems of probabilistic inference. We will investigate inference techniques with a focus on performance and complexity. As part of our studies we will research frequentist inference (Chapter 3). For more complicated models involving intractable computation, we will resort to variational inference (Chapter 4). Finally when our objective requires reasoning over symbolic representations in hierarchical models we will resort to probabilistic programming and Markov Chain Monte Carlo (MCMC) techniques (Chapter 5).

To measure the implications of the theoretical results and the performance of the introduced methods, we will always test on both synthetic and real-world data, whilst comparing our methods to relevant benchmarks and ensure a state-of-the-art performance.

1.6 Contribution

In the following chapters, we describe our efforts to introduce current perspectives on causal discovery using data generated in complex dynamical systems and processes. However, before we do this, we would like to highlight the academic contribution of this thesis which is fourfold:

1. We introduce Multidimensional Causal Discovery (MCD). We propose a notion of cause and effect over time. This notion is aimed at large data sets as they exist in real-world databases. Such a notion is novel to research as it allows to generalise cause and effect over time for cases where we model our data with tensor-theoretic approaches. We implement the notion into the MCD-algorithm for causal discovery. It provides expressive and accurate causal discovery for large, multidimensional data sets.

2. Our work relaxes common assumptions for causal discovery by generalising causal discovery for scenarios with dynamically changing noise levels where the dynamics depend on the input. Our generalisation is implemented into an efficient algorithm called FlexCD. The algorithm offers a performance gain for the problem of causal discovery in the case where only two variables are observed. The method further-

more provides competitive performance for prediction in terms of regression due to the use of a recent version of a Gaussian Process (GP).

3. We introduce the *Gaussian Process memoizer* (`gpmem`), a probabilistic programming technique that uses a GP to provide a statistical alternative to memoization, that is storing previously computed results and returning those instead of recomputing at a certain input. Memoizing a target procedure results in a "self-caching" wrapper that remembers previously computed values. GP memoization additionally produces a statistical emulator based on a GP whose predictions automatically improve whenever a new value of the target procedure becomes available. We implement this technique into the Venture probabilistic programming language and illustrate its performance for hierarchical models and Bayesian Optimization.

4. We show how to apply `gpmem` for functional causal reasoning in the context of probabilistic programming. In particular, we describe how a discovery method that allows for symbolic causal reasoning over time can be constructed. We measure performance in terms of prediction and inspect posterior distributions over symbolic representation.

We would like to stress that our contribution builds on and extends current research in different fields which gained much attention in the recent years (see section 2.5).

1.7 Previous Publications

This thesis resulted in peer-reviewed publications and submissions to journals. The papers are shorter versions of Chapters 3, 4 and 5. The submitted articles lack the broader context of dynamical causal discovery.

Multi-dimensional Causal Discovery.
Schaechtle, U., Stathis, K., and Bromuri, S. (2013).
International Joint Conference on Artificial Intelligence (IJCAI). Pages 1649 –1655.
FlexCD: Flexible Causal Discovery.
Schaechtle, U., and Stathis, K. (2015 - under review).
Submitted.
Probabilistic Programming with Gaussian Process Memoization.
Schaechtle, U., Zinberg, B., Radul, A., Stathis, K. and Mansinghka, V. K.
(2015 - under review).
Submitted.

1.8 Thesis Structure

Throughout this thesis, we assume that the reader has undergraduate background in a quantitative discipline, such as computer science, mathematics, physics or engineering. We further assume familiarity with fundamental linear algebra on a level one may find it in an introductory course in engineering. We exemplify causal discovery for dynamical systems in continuous-valued variable domains only, although most methods we use have counterparts in the discrete-valued domain.

We will start by giving the reader the relevant background needed to understand our contribution (Chapter 2). Basic probability theory and information theory will be introduced (section 2.1). We will re-visit Bayes' rules and talk about Bayesian machine learning. A brief survey of probabilistic knowledge representation will be provided (section 2.2). We will encounter computation that is not analytically tractable, which leads us to inference techniques that can mitigate this by giving sound approximation of such computation (section 2.3). We will finally provide the context of recent research for computational discovery of cause and effect (section 2.4). The final part of the background will put these methods into the context of the state-of-the-art and we will highlight recent research that is involved with the different fields described.

Equipped with the background knowledge, we will introduce temporal dependence in models used for causal discovery. First, we will show how we learn causal relations within high-dimensional tensor data (Chapter 3). We will examine and explain how our method allows the simultaneous inclusion of numerous dimensions within the data analysis such as samples, time and domain variables construed as tensors. The method will build on dynamic noise distributions but will not explicitly include those into the model. We abstract away from theses dynamics and important information could be lost. This is a weakness that we will try to mitigate in Chapter 4.

Here, we will explicitly define a set of functional equations providing an interpretation of noise in a causal dynamical system. The functional, relational model will include a latent structural noise function that dynamically changes the level of noise (section 4.1). We will introduce the necessary background on the functional equations we are using and on the specifics of functional causal discovery deployed (section 4.2). Equipped with this knowledge, a new criterion and an efficient algorithm for causal discovery will be described (section 4.3). We restrict the model to include a certain latent structural noise function that dynamically changes the level of noise. If we want to learn such a concept automatically we will need a more powerful method.

In Chapter 5 we will achieve this by introducing time in a causal system as an observed entity and we will build on probabilistic programming. We will explain pro-

gramming language constructs that are fairly new to machine learning research. We will introduce Venture GPs and the *Gaussian Process memoizer* showing how we can discover latent causal components in time series data (section 5.1.3). To show the flexibility of the technique, we will also explore Bayesian optimisation through GP memoization in section 5.5.

Finally, in Chapter 6 we will summarise the work presented and draw conclusions. We will discuss future work, first specific to the individual chapters, then in a more general context.

2 | Background

In this chapter we will work our way forward from simple probability calculus to sophisticated knowledge representation, using probabilistic frameworks to finally see how we can derive notions of cause and effect. We begin by presenting the basic building blocks required to derive the methods that will follow in later chapters.

2.1 Probability Theory

Machine intelligence research is traditionally focused on two aspects of the world (e.g. Russell, 2015):

1. there are objects in the world; and
2. there is uncertainty about what happens in the world.

Traditional Artificial Intelligence (AI) research focuses on the objects in the world (e.g. Russell and Norvig, 2010)[1]. Probability theory on the other hand is the framework to handle uncertainty logically. It is an extension of symbolic logic in the presence of uncertainty (Cox, 1946). Probabilistic modelling can assign uncertainty to noise, parameters and higher order structures.

Classical statistics models probability distributions with probability mass functions for discrete-valued variables. Two fundamental rules are defined for this context, the first rule is the sum rule:

$$P(X = x) = \sum_{y \in Y} P(X = x, Y = y) \text{ or in short: } P(x) = \sum_{y \in Y} P(x, y) \tag{2.1}$$

where X and Y are random variables, x is a value that X can take and the quantity $P(x, y)$ is the joint probability, that is the probability of x and y. In the context of random variables, we say that an event is a random variable taking a value, that is

[1] In this thesis, we cope with the existence of objects via probabilistic programming (see section 2.2.5) and its sampling semantics (Mansinghka et al., 2014).

$X = x$. The second fundamental rule is the product rule:

$$P(x, y) = P(x)\, P(y \mid x). \tag{2.2}$$

where we say that $P(y \mid x)$ is the probability of y given x. The probability $P(x)$ of x is also known as the marginal probability of x. From the two rules above follows Bayes' theorem:

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)} = \frac{P(x \mid y)P(y)}{\sum\limits_{y \in Y} P(x, y)} \tag{2.3}$$

For continuous variables we do not consider probability mass functions but instead probability density functions. The above rules still hold but we need to replace the sum in (2.1) with an integral:

$$P(x) = \int P(x, y)\mathrm{d}y \tag{2.4}$$

In the context of machine learning, this allows the following interpretation, replacing x by $\mathcal{D}$ to denote the data, and y by θ, which denotes the unknown parameters. m denotes the model of our choice. A typical task for machine learning is to estimate (learn) the unknown parameters given the model and what we have observed. Thus, we are interested in:

$$P(\theta \mid \mathcal{D}, m) = \frac{P(\mathcal{D} \mid \theta, m)\, P(\theta \mid m)}{P(\mathcal{D} \mid m)} \tag{2.5}$$

where the left hand side is the posterior of the parameters given the data, $P(\mathcal{D} \mid \theta, m)$ is the likelihood term and $P(\theta \mid m)$ is a prior on the parameters. Posterior refers to the posterior probability, that is after evidence has been taken into account. Prior refers to the prior probability, before evidence has been taken into account.

The Gaussian Distribution

The Gaussian distribution will be used throughout this thesis to model causes and causal mechanisms. It is defined as follows:

$$P(x) = \mathcal{N}(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{2.6}$$

where μ is the mean and σ^2 is the variance. The multivariate extension of the above distribution can be expressed as:

$$P(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}}} \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{2.7}$$

Accordingly, the parameters determining this distribution are the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. d is the dimensionality of $\mathbf{x}$. The Gaussian comes with interesting properties, for example, if $P(X, Y)$ is Gaussian, both the conditional and the marginal are Gaussian.

2.1.1 Information Theory

In section 2.4 and Chapter 4 we will see how information theory can inform us about the direction of causality. We will give a brief introduction to relevant quantities, starting with entropy. We treat the notion of entropy as a notion of surprise in the following chapters (Bishop, 2006; MacKay, 2003). The expected amount of information we gain by observing a random variable x is defined as

$$\mathbb{H}[x] = -\sum_{x \in X} P(x) \log P(x) \tag{2.8}$$

for discrete variables and

$$\mathbb{H}[x] = -\int P(x) \log P(x) \mathrm{d}x \tag{2.9}$$

for continuous ones. More formally, Shannon explains that the entropy is a lower bound on the number of bits needed to transmit the state of a discrete-valued random variable when we decide the base of the logarithm to be 2 (1948; as in Bishop, 2006). Let $P(x, y)$ be the joint distribution for two variables. The average additional information needed to specify y given x is the conditional entropy:

$$\mathbb{H}[y \mid x] = -\iint P(x, y) \log P(y \mid x) \, \mathrm{d}x \mathrm{d}y. \tag{2.10}$$

We can use (2.9) to define a measure of Gaussianity

$$\mathbb{J}[x] = \mathbb{H}[Y^{\mathcal{N}} = y] - \mathbb{H}[X = x], \tag{2.11}$$

where $Y^{\mathcal{N}}$ is a Gaussian random variable with the same covariance matrix as X (Hyvärinen and Oja, 2000). This quantity is known as negentropy.

Kullback-Leibler divergence

Kullback-Leibler divergence (or $\mathbb{KL}$ divergence) is also known as relative entropy. Let $P(x)$ and $Q(x)$ be two distributions, where $P(x)$ is not known and $Q(x)$ is an approximation of $P(x)$.

$$\mathbb{KL}[P||Q] = -\int P(x) \log \frac{Q(x)}{P(x)} \mathrm{d}x \tag{2.12}$$

The information theoretical interpretation of this quantity is the following: how many additional bits do we need to specify x when we use Q instead of the true distribution (Bishop, 2006)? This makes $\mathbb{KL}$ divergence a suitable measure of distance between two distributions, especially for machine learning where we often find ourselves assessing the goodness of approximations of distributions that are used to predict values of target variables.

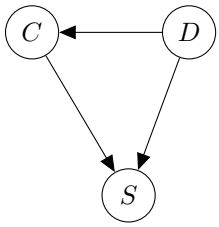


Figure 2.1: A simple Bayesian Network that describes our knowledge with regards to a disease D, a co-morbidity C for this disease and their interaction and effect on a symptom S. Vertices denote random variables. Edges denote dependencies.

2.2 Probabilistic Knowledge Representation

We can encode knowledge about the world and its uncertainties in different ways. One way is to draw a graph. A graphical model is a simple way describe uncertain entities and their relations. We can write down the dependency graph between entities of our world that we treat as random variables. In this framework, a node in a diagram depicts a random variable and an edge determines a probabilistic relation.

2.2.1 Bayesian Networks

We differentiate graphical models with directed relations known as Bayesian Networks (Pearl, 1986, 1988) and represented as a Directed Acyclic Graph (DAG) from graphical models with undirected relations represented as an undirected graph. We will focus on acyclic and directed graphs in this work[2].

The graphical representation of our world in form of a DAG determines how the joint probability distribution factors for our world. To illustrate this, let us investigate the relational model of a disease, a co-morbidity[3] and a symptom (Fig. 2.1). When viewed in the Bayesian context of (2.5) the resulting graph corresponds to the structure of the model (m) and the parametrisation (θ) which is given by its Conditional Probability Distributions (CPD), that is the probability of an event to be true when the values of other random variables are known, for example Tables 2.1, 2.2 and 2.3, where t stands for *true* and f for *false*.

The joint probability mass function factorises as:

$$P(S, C, D) = P(S \mid C, D) \times P(C \mid D) \times P(D) \tag{2.13}$$

We can read these networks as causal in the sense that they communicate a clear

[2]Extensions for cyclic graphs are known.

[3]When two disorders or illnesses occur in the same person, simultaneously or sequentially, they are described as comorbid. Comorbidity also implies interactions between the illnesses that affect the course and prognosis of both (National Institute of Drug Abuse, 2015).

Table 2.1: $P(S \mid C, D)$

		C, D			
		t,t	t,f	f,t	f,f
S	t	0.8	0.5	0.6	0.1
	f	0.2	0.5	0.4	0.9

Table 2.2: $P(C \mid D)$

		D	
		t	f
C	t	0.8	0.01
	f	0.2	0.99

Table 2.3: $P(D)$

D	t	0.1
	f	0.9

separation of cause and effect. Learning the structure of these networks given the vertices is hard, as the number of possible DAGs increases rapidly. The simplest way to learn such a network would be to search the space of possible networks while computing the likelihood of candidate structures (model m in (2.5)) and produce a score. The score is composed of the likelihood and a penalty term. This penalty decreases the score more the more complex the model gets. We will see in section 2.4 that this is not sufficient to find truly causal links between variables of interest.

We can extend the Bayesian Network formulation to represent temporal dependency. Such a Bayesian Network is known as a Dynamic Bayesian Network (DBN) (Murphy, 2002). A simple example for a number of measurements over time is a Markov chain (Fig. 2.2). Let each individual random variable in this chain be Gaussian white noise:

$$x \sim \mathcal{N}(0, \sigma_t). \tag{2.14}$$

This phenomenon is known as heteroskedasticity: a dynamically changing noise variance. Stable noise variance, where $\forall\, t:\ \sigma_t = \sigma$, is known as homoskedasticity. Let us now measure X_t along this chain, that is for different values of t. If we now drop the time dimension and model a probability density $P(X)$ as our distribution of the sampled measurements, this distribution will only be Gaussian in the homoskedastic case. Otherwise, in the heteroskedastic case, $P(X)$ will be non-Gaussian (e.g. Hyvärinen et al., 2010). This phenomenon shall become crucial for this thesis.

Probabilistic inference follows the same principles for DBNs as applied above. Many standard models in dynamical systems and time-series processing can be represented as DBN, such as Kalman Filters and Hidden Markov Models (e.g. Murphy, 2002).

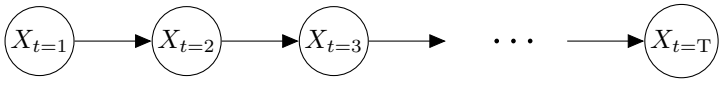


Figure 2.2: A Markov chain as a DBN. Random variable X is measured at different points in time t.

2.2.2 Bayesian Nonparametrics and Hierarchical Modelling

We have given a Bayesian interpretation of machine learning with (2.4). Here, we denoted the parameters of the model with θ. If our learning boils down to learning these parameters, we face a severe restriction: we can only learn models of the same parametric family. With more parameters θ, one can normally express more data generating functions. An intuitive example is the order n of a polynomial where $\theta_1, \cdots, \theta_n$ are the parameters, that is the coefficients of the polynomial. This insight is the foundation of Bayesian statistics. Many Bayesian nonparametric methods can be derived by letting $\theta \to \infty$. To render this computable and to ensure an adequately complex model we let the space of parameters grow with the amount of data that we observe. This comes with useful properties:

1. more, and more diverse data can be accounted for by more parameters;
2. the learning never saturates: the more data we get the better we expect our model to be; and
3. because we only apply the sum (2.1) and the product rule (2.2) instead of optimising a certain objective such as the loss function there can be no over-fitting.

Bayesian nonparametrics assign a prior belief on the distributions on a target. The most prominent examples of these are Dirichlet Process (DP) and the Gaussian Process (GP). A DP models a distribution on distributions. A GP models a distribution on functions.

2.2.3 Gaussian Processes

We will now show how this prior belief is expressed in a GP. Let the data be pairs of real-valued scalars $\{(x_i, y_i)\}_{i=1}^n$ (complete data will be denoted by column vectors $\mathbf{x}$, $\mathbf{y}$). We will drop the index for readability and we work exclusively on two-variable regression problems. In regression, one tries to estimate the functional relationship between two random variables X, Y of the kind $y = f(x)$.

The most important assumption we make when using a GP is that the space of possible functions f that map our regression input to regression output is normally distributed. Here, we make use of the aforementioned properties of the Gaussian: we can apply Bayes rule to transform the Gaussian prior on functions to a posterior that takes into account the observations.

Before we delve into the formulas, we would like to invite the reader to have a look at graphical representation of a GP. We can express a GP with a Bayesian Network, extending the Bayesian network formulation as outlined in previous sections (Fig. 2.3).

The resulting graphical representation is loosely based on the notation of plate models (Buntine, 1994). The plate notation was originally developed to describe repeated structure in probabilistic models. In this context it allows us to depict the nonparametric aspect of the model. In this version of a Bayesian Network, we represent functional relationships explicitly with a black square in the middle of the arrow[4].

Formally, a GP is an infinite-dimensional extension of the multivariate Gaussian distribution. The aim of applying a GP is to find a posterior distribution on the functions that regresses input $\mathbf{x}$ to output $\mathbf{y}$. Our prior for this is Gaussian $\mathbf{f}|\mathbf{x} \sim \mathcal{N}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big)$. While the Gaussian distribution is parametrised by mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, a GP is determined by its mean function $m(\mathbf{x})$ and its covariance function $k(\mathbf{x}, \mathbf{x}')$ (a.k.a. kernel) both of which depend on the input data for the regression. For any finite set of inputs $\mathbf{x}$, the marginal prior on $f(\mathbf{x})$ is the multivariate Gaussian

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x})),$$

where $m(\mathbf{x}) = \mathbb{E}_f\left[f(\mathbf{x})\right]$ and $k(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}_f\left(f(\mathbf{x}), f(\mathbf{x}')\right)$[5] In all examples below, our prior mean function m is identical to zero; this is the most common choice. The marginal likelihood can be expressed as:

$$p\left(f(\mathbf{x}) = \mathbf{y} \mid \mathbf{x}\right) = \int p\left(f(\mathbf{x}) = \mathbf{y} \mid f, \mathbf{x}\right) p(f|\mathbf{x})\, df \tag{2.15}$$

where $p(f|\mathbf{x}) = p(f) \sim \mathcal{GP}(m(x), k(x, x'))$ omitting (x) for readability. We can absorb $m(\mathbf{x})$ into the covariance function, so without loss of generality we can set $m(\mathbf{x})$ to 0. We can sample a vector of unseen data $\mathbf{y}^* = f(\mathbf{x}^*)$ from the predictive posterior with

$$\mathbf{y}^* \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2.16}$$

a multivariate normal with mean vector

$$\boldsymbol{\mu} = k(\mathbf{x}^*, \mathbf{x})\, k(\mathbf{x}, \mathbf{x})^{-1}\, \mathbf{y} \tag{2.17}$$

and covariance matrix

$$\boldsymbol{\Sigma} = k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, \mathbf{x}) k(\mathbf{x}, \mathbf{x})^{-1} k(\mathbf{x}, \mathbf{x}^*). \tag{2.18}$$

Often one assumes the values $\mathbf{y}$ are noisily measured, that is, one only sees the values of $\mathbf{y}_{\text{noisy}} = \mathbf{y} + \boldsymbol{\eta}$ where $\boldsymbol{\eta}$ is Gaussian white noise with variance σ^2_{noise}. In that case, the log-likelihood is

$$\log p\left(\mathbf{y}_{\text{noisy}} \mid \mathbf{x}\right) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log\left|\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I}\right| - \frac{n}{2}\log 2\pi \tag{2.19}$$

[4]This is a slight abuse of notation: the black square normally depicts a factor in a factor graph. We do not consider any factor graphs in this work, the black square will here always indicate a certain functional relationship.

[5] Note that $m(\mathbf{x}) = (m(x_i))_{i=1}^n$ and $k(\mathbf{x}, \mathbf{x}') = \big(k(x_i, x'_{i'})\big)_{1 \le i' \le n'}^{1 \le i \le n}$, where n' is the number of entries in $\mathbf{x}'$.

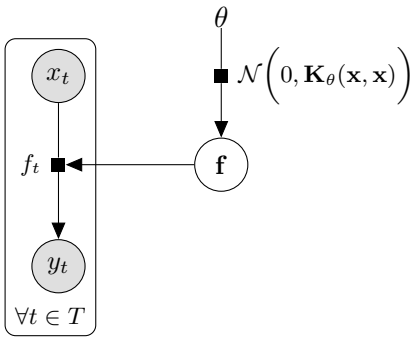


Figure 2.3: A GP depicted as a graphical model where x is the regression input or independent variable and y is the dependent variable. For every instance of the pairing x and y we describe a distribution of functions that map from x to y. Often we are interested in timestamps or time series. In a simple case, $x_t = t$ meaning our only input to the GP is a point in time. A vector of hyper-parameters $\boldsymbol{\theta}$ parametrises the covariance function which we have expressed as $\mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x})$. The hyper-parameters determine along with the observed data the covariance parameter of the multivariate Gaussian.

where n is the number of data points. For readability, we have written the entire covariance matrix for all combinations of $k(\mathbf{x}, \mathbf{x}')$ as $\mathbf{K}$. Without loss of generality, we can absorb the independent noise σ_{noise} into $\mathbf{K}$, so (2.19) simplifies to:

$$\log p\left(\mathbf{y}_{\text{noisy}} \mid \mathbf{x}\right) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}| - \frac{n}{2}\log 2\pi. \tag{2.20}$$

We will make use of both notions for independent noise in our work since they give rise to interesting algorithms.

GPs present a nonparametric way to express prior knowledge about the space of possible functions f. The prior knowledge enters the model in form of the covariance function. The covariance function governs high-level properties of the observed data such as linearity, periodicity and smoothness. The most widely used form of covariance function is the squared exponential which enforces the smoothness assumption:

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right), \tag{2.21}$$

where σ and ℓ are hyper-parameters: σ is a scaling factor and ℓ is the typical length-scale. Once one has decided on the form for the covariance, we can optimise for any unknown hyper-parameter:

$$\frac{\partial \log p\left(\mathbf{y}_{\text{noisy}} \mid \mathbf{x}\right)}{\partial \theta_i} = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \boldsymbol{\theta}_i}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{trace}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \boldsymbol{\theta}_i}\right) \tag{2.22}$$

The graphical representation (Fig. 2.3) reveals intricate functional relationships in the Bayesian network sense, where we related scalars and distributions functionally and

with statistical dependencies. It also shows a kind of hierarchy. This hierarchy exists because we optimise hyper-parameters, that is parameters influencing the distribution of the actual parameters that grow with the number of data points available. We can imagine a more complex hierarchical model, with a more intricate causal system determining the posterior distribution we are looking for. Such hierarchies can help in cases where Big Data is available in the form of many "small data", that is, where we have a lot of instances of related problems. Here, we may have to couple models logically by relating them and propagate uncertainties from instance to instance in a sound manner.

Zoubin Ghahramani states an important example for this (2015): in personalised medicine, we may have a large database with numerous patients but we may not have enough data for an individual patient with a certain combination of diseases as the space of possible combinations of diseases and prescribed drugs is extremely large. When building a model for this patient, we do not have much data available, but can couple this model with models of other patients who are similar in a hierarchy of models to exchange information. For example, we can imagine two diabetic patients, both having administered the same drugs and suffering from neuropathy. Both patients also suffer from cardiovascular disease. One patient suffers from tachycardia the other has had a stroke. Both complications belong to the class cardiovascular disease and are therefore hierarchically related.

2.2.4 Model Selection

Hierarchies are closely related to model comparison or selection[6]. Discussing model selection under a Bayesian viewpoint is particularly important for this thesis because causal discovery is a model selection task. For example, we compare two models $X \to Y$ and $X \leftarrow Y$ and we want to condition this selection based on what we know and what we have observed before. Selecting models and conditioning the selection process on data requires a quantitative version of Occam's razor to penalize overly complex models (MacKay, 2003). Bayesian methods automatically embody Occam's razor. As David Mackay puts it:

> "Bayes' theorem rewards models in proportion to how much they predicted the data that occurred. These predictions are quantified by a normalized probability distribution on $\mathcal{D}$. This probability of the data given model m_i, $P(\mathcal{D} \mid m_i)$, is called the evidence for m_i. A simple model m_1 makes only a limited range of predictions [...]; a more powerful model m_2, that has, for

[6]Imagine the following hierarchy: probability of model → probability of parameters → observations

example, more free parameters than m_1, is able to predict a greater variety of data sets."(2003)

2.2.5 Probabilistic Programming

In the previous sections we have seen how graphical models serve as a principled way to represent knowledge in an uncertain world. Standard Bayesian networks with all discrete variables pose an easy inference problem. In contrast, if we have a more complex model, for example the graphical representations involving plate notation and distributions on functions from the previous section (Fig. 2.3), or a model with multiple levels of hierarchy, the matter is different and inference is not simple, anymore. Crafting sound and efficient inference algorithms becomes very difficult as the necessary derivations often involve cumbersome algebra and non-trivial proofs of convergence. As a result, new models are often communicated using a mix of natural language, pseudocode and mathematical formulae (Roy, 2015). Inference is special-purpose and often not re-usable. Designing, implementing and testing inference algorithms is hard even for experts. Small changes in requirements or assumptions can result in a change of the model or affect the inference algorithm. This can cause a lot of work for an expert and can keep non-experts outside of the domain of machine learning, completely, since they may lack the necessary mathematical background.

Probabilistic programming seeks to overcome these shortcomings with two main features:

1. an expressive language able to accommodate both declarative and procedural semantics to represent probabilistic knowledge and dependencies in a sound way; and

2. a generalised inference engine that mitigates the drawbacks of one-off inference algorithm design.

Causal generative probabilistic programming languages (e.g. Goodman et al., 2008) define a generator for data from the probabilistic model in form of a simulator. This simulator calls a random number generator to produce stochastic samples from a distribution which is specified by the program. This is more general than the graphical model framework as it can accommodate procedural aspects such as recursion and control flow statements (Ghahramani, 2015).

We can observe data and subsequently condition on these observations when inferring latent variables. The process of conditioning involves trying to find inputs of a program conditioned on its observed output and the probabilistic program. A common approach is to implement the generalised inference engine using Markov Chain Monte

Carlo (MCMC) which is suitable thanks to its sampling-semantics and its compositional nature (see section 2.3.2).

For the work in this thesis, we choose Venture (Mansinghka et al., 2014). Venture is more expressive than most competing languages. We elaborate on this with a few examples, for an exhaustive comparison see the paper on Venture (Mansinghka et al., 2014). STAN (e.g. Kruschke, 2014) has only limited support for discrete random variables which is too limiting for causal discovery. Infer.NET (Minka et al., 2010) deploys an inference algorithm that prevents formulating a model with an unbounded number of random variables. Markov Logic Networks (MLN) (Richardson and Domingos, 2006) build on undirected graphical models. MLNs suffer from a semantic problem due to their capability to accommodate scenarios with different and unknown numbers of objects (see Russell, 2015) which renders them incompatible with Bayesian nonparametrics.

Probabilistic logic programming languages such as PRISM (Sato and Kameya, 1997), ICL (Poole, 1997) and ProbLog (De Raedt et al., 2007) are based on the declarative semantics of logic programming using Horn clauses (Horn, 1951). They provide limited to no support or guarantees of correctness (in the case of ProbLog) continuous-valued variables, which renders them problematic for machine learning for real-world problems and causal discovery. Problog2 claims to be more similar to functional probabilistic programming languages but also suffers from the inability to model unbounded sets of random variables and continuous-valued variables (Renkens et al., 2012).

The BLOG (Bayesian Logic) language (Milch et al., 2007) is more general in terms of its semantics compared to the probabilistic logic programming languages above. It provides first-order logic semantics in the presence of uncertainty. The authors of Venture provide substantial critique to BLOG:

> "BLOG is not higher-order - it does not support random variables that are themselves probabilistic procedures - nor does it provide a programmable inference mechanism. BLOG also does not support collapsed primitives that exhibit exchangeable coupling between applications, primitives that lack calculable probability densities, or primitives that can create and destroy latent variables while providing their own external inference mechanism." (Mansinghka et al., 2014)

Venture is the successor of Church (Goodman et al., 2008). It improves its predecessor in terms of expressiveness, as the language provides a built-in interface for stochastic procedures, where likelihood is unknown or hard to compute. It provides a simple interface for foreign stochastic procedures and is therefore easily extendible. The authors

of Venture report drastic performance gains over Church for inference which is key for causal discovery. A thorough comparison of probabilistic programming languages is out of scope of this thesis. For an exhaustive comparison of Venture with its competitors see the paper by Mansinghka and colleagues (2014).

Venture provides the causal generative semantics we are looking for to provide causal inference (see section 2.4) and supply valid evidence. In Listing 2.1, we depict a Venture program of the causal Bayesian network shown in the beginning of this section (Fig. 2.1). We will later learn that structural form of the expression coincides with the structural form needed for causal inference (section 2.4).

Venture also provides a general inference engine and is expressive, as it is Turing complete. It also enables efficient and customisable inference, which is crucial if we want to tackle the hard problem of causal inference.

Listing 2.1: Bayesian Network

```
1  /// SETTING UP THE MODEL
2  assume Disease = flip(0.1);
3  assume Co_morbidity = if(Disease) {flip(0.8)}
4                        else         {flip(0.01)};
5  assume Symptom = if(Disease){
6                      if(Co_morbidity){
7                         flip(0.8)
8                      } else {
9                         flip(0.6)
10                     }
11                 } else {
12                     if(Co_morbidity) {
13                        flip(0.5)
14                   } else {
15                        flip(0.1)
16                   }
17                };
18 /// OBSERVATION
19 observe Symptom = true;
20 ///INFERENCE
21 infer default_markov_chain(10);
```

2.3 Approximate Inference

Hierarchical and dynamical models often result in complex model structures that render exact inference impossible. This could be because the dimensionality of the hypothesis space we operate on is too high or, in the case of continuous variables, because a closed form solution may be intractable (Bishop, 2006). We will see later that this is an inherent problem of functional causal discovery where we are going to explicate assumptions and prior distributions on what it means when we say: "one variable causes another". In this work, we resort to two forms of approximative inference: Variational Inference and MCMC methods.

2.3.1 Variational Inference

Variational inference aims to approximate a target distribution by choosing a tractable approximative distribution and then minimising the distance between this approximation and the unknown target distribution. A suitable measure of distance between distributions is the $\mathbb{KL}$ divergence (2.12). Let $P(X)$ be some intractable distribution for which we are trying to find a good approximation. We assume a fully Bayesian setting with prior distributions on all latent variables and unknown parameters.

There are different formulations of variational inference. Below, we will give a brief introduction (2.23 -2.25) following Murphy (2012, chapter 21, e.q. 21.1 -21.7): Let $Q(X)$ be a suitable and tractable approximation to $P(X)$. The direction of the divergence (2.12) is inverted, since taking the expectation with respect to (wrt) $P(X)$ is known to be intractable, switching the sign, one gets:

$$\mathbb{KL}[Q||P] = \int Q(x) \log \frac{Q(x)}{P(x)} \mathrm{d}x \tag{2.23}$$

The expectation wrt Q is now tractable. The expression (2.23), however, is not since it requires evaluating the intractable normalisation constant. Murphy points out that in contrast, the unnormalised distribution

$$\tilde{P}(X) \triangleq P^*(X, \mathcal{D}) = P(X) \times P(\mathcal{D}) \tag{2.24}$$

is tractable to compute. This leads to the new objective function:

$$\begin{aligned} J(Q) &= \int Q(x) \log \frac{Q(x)}{\tilde{P}(x)} \mathrm{d}x \\ &= \int Q(x) \log \frac{Q(x)}{P(x) \times P(\mathcal{D})} \mathrm{d}x \\ &= \int Q(x) \log \frac{Q(x)}{P(x)} \mathrm{d}x - \log P(\mathcal{D}) \\ &= \mathbb{KL}[Q||P] - \log P(\mathcal{D}). \end{aligned} \tag{2.25}$$

$P(\mathcal{D})$ being a constant, minimising $J(Q)$ implies that our approximation Q becomes closer to P.

Later in this thesis, we will deploy the formulation of this approximation in terms of information theory for causal discovery by taking advantage of the notion of surprise that is associated with information entropy. Beside this very particular use case, the main advantage of variational inference lies in its properties: it is a bounded approximation that often provides analytically tractable solutions to its inference. Its disadvantage is that variational inference algorithms are often hard to design; and when the model changes slightly, they may have to be re-designed from scratch.

2.3.2 Markov Chain Monte Carlo

MCMC follows a completely different idea compared to variational inference. The intuition behind MCMC is drawing random samples and processing them to approximate probability distributions that are hard to represent otherwise. More formally, the Monte-Carlo principle states that we draw samples from a target distribution defined on a high-dimensional space which give an approximation of the posterior using (Andrieu et al., 2003):

$$P_N(X) \simeq \frac{1}{n} \sum_{i=1}^{n} \delta_{x^i}(x) \tag{2.26}$$

with $\delta_{x^i}(x)$ denoting the delta-Dirac mass located at x^i. MCMC is a strategy to generate this set. An MCMC algorithm requires a proposal mechanism, which is normally a representation of a proposal distribution that we draw samples from. With this proposal mechanism, we construct a Markov chain on a state space whose stationary distribution is the target distribution of interest. We can evaluate $P(X)$ up to a normalising constant, as opposed to sampling directly from it (Andrieu et al., 2003). We call the mechanism that produces individual entries for one or more random variables in the Markov chain a transition or MCMC-kernel.

In the following we will introduce the Metropolis-Hastings (MH) algorithm since it is the default inference algorithm in Venture. This algorithm is the oldest and most

popular MCMC algorithm:

Algorithm 1. Metropolis-Hastings

1: Initialise x^0
2: Define proposal distribution Q
3: Supply a way to evaluate P
4: **for** $i = 0$ to $n - 1$ **do**
5: $\quad u \sim \mathcal{U}_{[0,1]}$
6: $\quad x^* \sim Q(x^* \mid x^i)$
7: $\quad a = \min\{1, \frac{P(x^*) \times Q(x^i|x^*)}{P(x^i) \times Q(x^*|x^i)}\}$
8: $\quad$ **if** $u < a$ **then**
9: $\quad\quad x^{i+1} = x^*$
10: $\quad$ **else**
11: $\quad\quad x^{i+1} = x^i$
12: $\quad$ **end if**
13: **end for**
end

We sample from the proposal distribution (line 4) and check whether we are accepting a new sample (line 5 and 6) stochastically by calling a random number generator (line 3). The algorithm will spend more time in regions that are characteristic for the target distribution and less time in less characteristic ones. The samples are often autocorrelated. If this is the case, it makes sense to run several changes in parallel and mix them in the end to get the target. Furthermore, the algorithm often requires a so called "burn-in" time, that is all the steps to be taken before the actual stationary distribution is reached.

Many other MCMC algorithm can be expressed as special cases of MH, such as the Hybrid (or Hamiltonian) Monte Carlo (HMC) algorithm (Duane et al., 1987) or the Reversible Jump (RJ) MCMC algorithm (e.g., see Andrieu et al., 2001). HMC takes into account gradient information to improve convergence to the target distribution. RJ is an extension that deals with model selection problems and spaces where probabilities of samples cannot be compared directly, for example by "jumping" from a lower to a higher order polynomial as hypothesis.

MCMC algorithms are composite: we can chain inference on hierarchical structures by chaining different MCMC kernels together (Fig. 2.4). This makes it possible to learn both the structure and the parameter of a model within a single framework providing a sound approximation. The compositional aspect is also the reason why MCMC is so popular in probabilistic programming - it builds up program dependent inference by

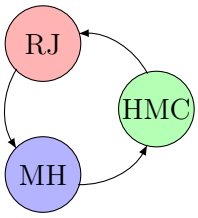


Figure 2.4: A chain cycling over three different MCMC-kernels. MCMC algorithms are composite: we can chain inference on hierarchical structures by chaining different MCMC kernels together. We see a chain of Metropolis-Hastings (MH), Hybrid (or Hamiltonian) Monte Carlo (HMC) and Reversible Jump (RJ) kernels.

chaining MCMC kernels accordingly.

The main advantage of MCMC is that one does not have to re-design the inference algorithm every time the model changes. MCMC methods, however, do have a disadvantage with regards to their convergence properties: convergence time is unpredictable for all but a few very well understood examples. Both auto-correlation of drawn samples and the ideal "burn-in" time are often unknown to the practitioner. These uncertainties can result in a lot of fine tuning needed for MCMC methods.

2.4 Functional Equations & Causality

We have mentioned in section 2.2.1 that a search-and-score approach to find causal structures is not sufficient to discover truly causal relations as opposed to merely statistical associations. We shall cite Karl Popper's definition of causal explanation (1959, chapter 3). He states that causally explaining an event means to deduce a statement which describes the event using deduction over background knowledge of the world[7]. This is fairly general, and while the reader may expect this chapter to open with a simple definition of the concept of causality itself we try to avoid this in order to not digress into the philosophical discourse of this matter. Spirtes and colleagues (2000, chapter 1) point out that mathematicians from Euclid to Newton to Kolmogorov provide axioms that use the notion of causality without defining it but investigate the consequences of the assumptions they make for their work. The research presented in this thesis will follow this approach as opposed to a philosophical one.

Why is statistically scoring structures not sufficient to discover relations of cause and effect? Going back to our previous example of diseases and symptoms, we find that we cannot express simple true statements in the language of pure statistics and probability, such as "disease causes symptoms". As Judea Pearl puts it:

[7]He further differentiates between singular conditions and universal laws - a distinction that we shall not examine in detail at this point.

> "Scientists seeking to express causal relationships must therefore supplement the language of probability with a vocabulary for causality, one in which the symbolic representation for the relation 'symptom causes disease' is distinct from the symbolic representation of 'symptoms are associated with disease.' Only after achieving such a distinction can we label the former sentence 'false', and the latter 'true'." (2001)

In his seminal work, Pearl introduces the notion of functional equations and graphical criteria to reason over cause and effect (Pearl, 2000). The set of functional or structural equations is defined as follows:

Definition Structural Equation Model (Pearl, 2000, 2012)
A structural equation model M is defined as follows:

- A set U of background or exogenous variables, representing factors outside the model, which nevertheless affect relationships within the model.
- A set $V = \{V_1, \cdots, V_n\}$ of observed endogenous variables, where each V_i is functionally dependent on a subset PA_i of $U \cup V \setminus \{V_i\}$.
- A set F of functions $\{f_1, \cdots, f_n\}$ such that each f_i determines the value of $V_i \in V$, $v_i = f_i(pa_i, u)$.
- A joint probability distribution $P(u)$ over U .

Pearl calls his notion structural equations, whereas we prefer the term functional equations since the latter maps directly to the way we represent the causal model in later chapters (3 to 5).

We will now relate this notation to our running example, the Bayesian Network of section 2.2.1 (Fig. 2.1):

$$\begin{aligned} d &:= \eta_d \\ c &:= f_D(d) \\ s &:= f_S(c, d) \end{aligned} \tag{2.27}$$

where := denotes assignment as opposed to algebraic equality[8]. Pearl's famous do-calculus (Pearl, 1995) reasons over post-intervention distributions. Such an intervention allows us to reason causally about the probability of an event $X = x$ if a certain condition were to be enforced uniformly over the population. This manifests in changing assignments in the functional equations (2.27) or in deletion of certain functional

[8]This has caused much confusion in the statistics community (see Pearl, 2000).

equations and deletion of certain edges in the causal DAG. For example, would we assert that a patient does not suffer from a co-morbidity, (2.27) would become:

$$\begin{aligned} d &:= \eta_d \\ c &:= \text{False} \\ s &:= f_S(c, d). \end{aligned} \tag{2.28}$$

The semantics of the structural equations and the $do(X = x)$ are surprisingly parallel to what we know from computer programming. It is known for example that probabilistic logic programs are a representation for causal functional equations as in the notion above (Poole, 2008). We argue that the same is true for the Venture programs introduced in section 2.2.5. One can easily emulate the $do(X = x)$ intervention with the `observe` X=x directive which constraints the value of the random variable X to x.

In the following, we will adopt the notion of perfect intervention introduced recently by Mooij and colleagues (2014). This notion allows us to define our working definition of causal discovery without relying too heavily on Pearl's algebraic framework (1995; 2012).

Definition (Mooij et al., 2014)
We say that X causes Y ($X \to Y$) if $P(Y = y \mid do(X = x)) \neq P(Y = y \mid do(X = x'))$ for some x, x', where $do(X = x)$ forces random variable X to have value x, but leaves the rest of the data generating mechanism untouched.

The questions that arises is the following: what if the actual intervention is not available and we have only data from the observational distribution $P(X, Y)$? Causal discovery seeks to find causal relations in such a situation. We differentiate between two approaches that aim to overcome this shortcoming, the constraint-based causal discovery initiated by Pearl himself (1991) and functional or assumption-based causal discovery, initiated in 2005 by work on linear models with non Gaussian additive noise (Shimizu et al., 2005, 2006).

2.4.1 Constraint-based Causal Discovery

These notions give rise to a simple constraint-based causal discovery algorithm (Pearl et al., 1991; Pearl, 2000; Spirtes et al., 2000). Constraint-based causal discovery usually relies on a two-step process to produce a causal DAG.

1. We start with a fully connected undirected graph where each node represents a variable that can potentially have an effect on another variable. We delete an edge for every observed statistical independence. This includes independence

conditioned on other nodes/random variables in the graph; with all unneeded edges removed.

2. We subsequently apply a set of rules to orient the remaining edges to determine the direction of cause and effect.

For the original versions of these rules, see the work cited above. These rules, however, have been changed for efficiency since then (Spirtes, 2001). Claassen and Heskes showed how these rules can be simplified to three rules formulated in symbolic logic (2011). Subsequently, they showed how their logic can be used to devise a more Bayesian treatment of constraint-based causal discovery by combining their approach with a search-and-score mechanism (Claassen and Heskes, 2012).

For some cases, purely constraint-based methods cannot output a unique solution. This holds true for the simplest non-trivial causal graph, viz., a graph with two nodes representing two variables. Here, we need to resort to functional or assumption-based causal discovery.

2.4.2 Assumptions and Functional Causal Discovery

What if we have only two variables? Can we formulate our problem in a way so that we can still perform causal discovery? A certain research direction claims that this is possible by making specific restrictions on our prior belief of the data-generating systems. Given such restrictions we resort to measures and methodology from sections 2.1 and 2.2 and see what the data implies for the underlying causal direction.

The intuition behind those restrictions is that, if $X \to Y$ then $P(Y \mid X)$ depends on X in a less complicated fashion than $P(X \mid Y)$ on Y. Note that the notion of complexity is not well defined in this context since all we have to work with are observations of the joint distribution. Therefore, one has to become creative to encode a notion of this complexity in the restrictive assumptions we made to provide criteria to differentiate between cause and effect. Two schools of thought have prevailed in this context:

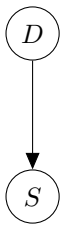


Figure 2.5: A simple Bayesian Network that describes our knowledge with regards to a disease D and its effect on a symptom S.

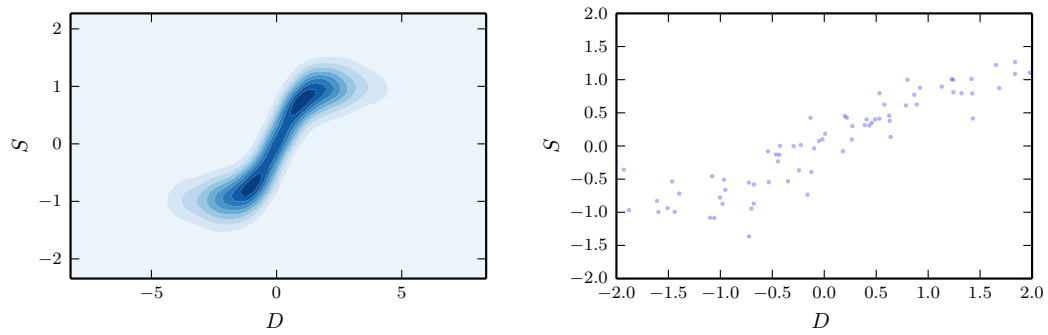


Figure 2.6: On the left, we see the joint probability distribution. On the right, we see a sample that needs to suffice to derive the direction of cause and effect. It is this sample, that we apply a criterion to that will serve as oracle for the causal direction.

Additive Noise Model (ANM) formulations that explicitly exploit a restriction of noise to be additive (Shimizu et al., 2005; Hoyer et al., 2008a; Zhang and Hyvärinen, 2009; Mooij et al., 2010; Peters and Bühlmann, 2014; Peters et al., 2014) and methods that make assumptions on measures from information theory (Daniušis et al., 2010; Janzing et al., 2012; Chen et al., 2014).

We shall illustrate both schools of thought using a two variable version of our previous example[9]. We imagine a continuous valued measurement $D = d$ which indicates a disease and a continuous-valued measurement $S = s$ that measures a symptom (Fig. 2.5).

We can formulate this as a system of functional equations:

$$\begin{aligned} d &:= \eta_d && \text{with:} \quad \eta_d \sim \mathcal{N}(0,1) \\ s &:= tanh(d) + \eta_s && \text{with:} \quad \eta_s \sim \mathcal{N}(0, 0.25) \end{aligned} \tag{2.29}$$

where η indicates uncertainty in form of noise, a.k.a error. For illustration purposes, we assume that we know the function that served as the data generating mechanism. We then take measurements, that is, we draw samples from the joint distribution (Fig. 2.6). Those samples will need to suffice to decide on the direction of cause and effect. The standard ANM approach builds on an independence test for the residuals (Hoyer et al., 2008a), where the residuals indicate the noise distribution. We assume that the error at all data points $\boldsymbol{\eta_s}$ is statistically independent from all measurements of the disease:

$$\boldsymbol{\eta_s} \perp\!\!\!\perp \mathbf{d}. \tag{2.30}$$

The vector notations for $\mathbf{d}$ and $\boldsymbol{\eta_s}$ denote all the measurements we have available for $D = d$ and all errors. We can illustrate this easily by modelling the conditional $P(Y \mid X)$

[9]This example is inspired by Figure 2 in Mooij et al. 2014.

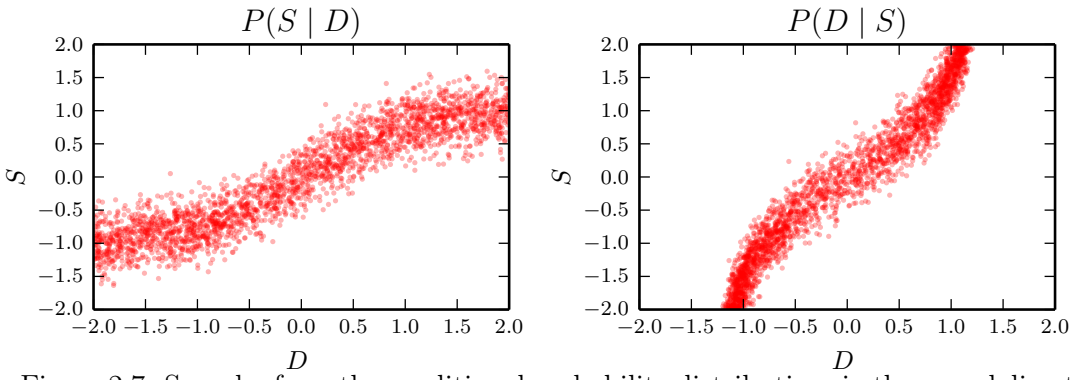


Figure 2.7: Samples from the conditional probability distributions in the causal direction (left) and vice versa (right), we see how the noise depends on the input in a more complicated fashion on the right hand side.

and draw samples from it (Fig. 2.7, left hand side). We see that the residuals to the mean stay stable over the input domain D. Let's say we model $Y \to X$ with functional equations analogous to (2.31), with some hypothetical regression function $f_?$:

$$\begin{aligned} s &:= \eta_s \\ d &:= f_?(s) + \eta_d \end{aligned} \tag{2.31}$$

Using such a system of functional equations to model the conditional $P(X \mid Y)$ yields:

$$\boldsymbol{\eta}_d \not\perp\!\!\!\perp \mathbf{s} \tag{2.32}$$

since the noise distribution now changes over the input domain (we can clearly see this in Fig, 2.7, right). The different manifestation of uncertainty in terms of the noise/error variables η_s, η_d provides the restriction that we want, that is, if $X \to Y$ then the dependence of $P(Y \mid X)$ on X is less complicated than the dependence of $P(X \mid Y)$ on Y. The complexity arises from the dependent and independent noise. We can estimate the functional equations with any regression method we would like, where η becomes the regression residuals. To compute the functional estimation, previous research mainly relied on GP. We can test (2.30) and (2.32) with standard statistical dependence tests.

A different way to derive asymmetry between cause and effect is by deploying measures from information theory (Daniušis et al., 2010; Janzing et al., 2012; Chen et al., 2014). The intuition behind this approach is that if $X \to Y$ then $P(X)$ encodes no information about $P(Y \mid X)$ and $P(Y \mid X)$ encodes no information about $P(X)$. We can measure this in terms of entropy and correlation with the slope a function. Instead of deriving the mathematical formulation, we shall illustrate this using our example (Fig. 2.8). We can explore the information relation between between $(S \mid D)$ & $P(D)$ and $(D \mid S)$ & $P(S)$ by exploring the data generating function $f = tanh$. Remember,

the ground truth is $D \to S$. So we expect no information between $(S \mid D)$ & $P(D)$ (i.e. in the causal direction) but we do for $(D \mid S)$ & $P(S)$. We can find and illustrate this by "feeding" a uniform distribution into the function instead of the true distribution of D. We find two ways in which there is information here. First, we realise that there is strong correlation between the $P(S)$ and the entropy of the conditional $\mathbb{H}(D \mid S)$. Intuitively, in areas with high $\mathbb{H}(D \mid S)$, many values of a small range of s map to a large range of d. Second, we find that the slope of f is correlated with $P(S)$. Both are not the case in the causal direction.

This insight leads to a variety of criteria of causal discovery (Janzing et al., 2012; Chen et al., 2014).

2.5 Recent Views

Next, we highlight the most recent views in the areas that are fundamental for this thesis.

The different assumptions one can make to differentiate between cause and effect are still not completely understood and are under ongoing investigation. A notable effort was conducted by Mooij and colleagues who try to establish a common benchmark that researchers can use to compare their methods (2014). Integration and generalisation of different assumptions is in progress, leading to more granular research and a better understanding of the conditions under which cause and effect are discoverable. Criteria from information theory have been simplified and extended from the bivariate to the multivariate case (Chen et al., 2014). For approaches based on the ANM-assumption, it was only recently discovered that in a linear setting with Gaussian noise the direction of cause and effect can be discovered when the noise variance is the same for all variables (Peters and Bühlmann, 2014).

Research on linear models with non-Gaussian noise is progressing, too (Hyvärinen and Smith, 2013; Shimizu, 2014). An example is a new way to treat latent confounders in this context (Tashiro et al., 2014). In chapter 3, we will show a multi-linear generalisation of this framework, which is based on tensor formulation. Representing data in terms of tensors for machine learning is a heavily researched field. For example recently, a multi-linear extension based on tensor formulation has been introduced for the Kalman Filter (Rogers et al., 2013). Kalman Filters can be viewed as a latent variable model for dynamical systems. They can also be expressed as a DBN. In general, it has been shown that for parameter estimation for a wide class of latent variable models in tensor formulation one can deploy a generalisation of the singular value decomposition (Anandkumar et al., 2014).

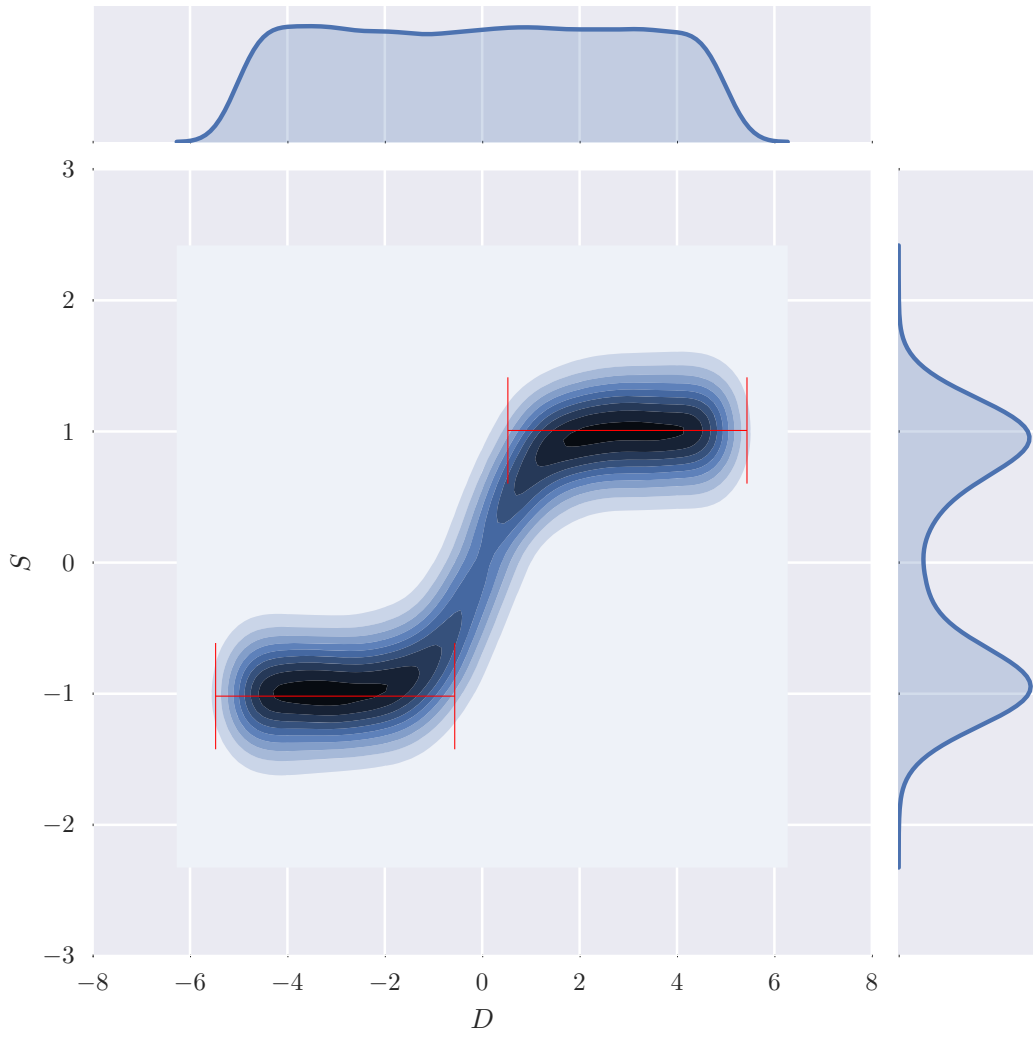


Figure 2.8: We see the density of the marginal $P(S)$ with a uniform input when $d \sim \mathcal{U}$. With red, we mark areas where $P(S)$ is high coinciding with low $|f'|$ and high $\mathbb{H}(D \mid S)$.

Consistency, as opposed to soundness, of ANMs is under discussion as well (Kpotufe et al., 2014). Multivariate extensions for ANMs have been added recently(e.g. Peters et al., 2014). Many implementations of ANMs build on GPs where normally the GP is used for standard regression. A novel interpretation however deploys a GP in the context of dimensionality reduction for causal discovery (Sgouritsa et al., 2015). The state of the art in GPs now involves symbolic computation whilst outperforming competing approaches in terms of prediction (Duvenaud et al., 2013; Lloyd et al., 2014). This has been demonstrated as part of the Automatic Statistician project[10]. A recent review on artificial intelligence and machine learning in the prestigious Nature journal named GP and probabilistic programming (as a form of symbolic computation) as amongst the most promising directions for future research (Ghahramani, 2015). Unifying symbolic computation in form of logic and programming languages with probabilistic techniques has been proclaimed to hold "enormous promise for artificial intelligence" (Russell, 2015).

2.6 Summary

We have provided the necessary background in "Bayesianism", information theory, probabilistic programming and causal discovery that is needed to understand the contribution of this thesis. We now know that:

- probability calculus is how we can logically deal with an uncertain world;
- Bayesian non-parametrics let the data speak for itself;
- probabilistic programming is a powerful way towards hierarchical causal models;
- we can perform inference, even if no closed form solution exists to our problem and the problem space is huge; and
- causal discovery is hard - but can be tackled by making prior assumptions and restrictions.

We have learned how we need Bayes rule and information theory to infer causes and effects. Bayesian methods are particularly important for Chapter 4 and 5. We deploy variational inference in Chapter 4 and MCMC in Chapter 5. Gaussianity is subject of discussion in Chapter 3 and in Chapter 4. We use probabilistic programming, approximate inference and Bayesian model selection in Chapter 5.

In the following we will show three ways of processing time in the context of the state-of-the-art in causal discovery. We will begin with a naive setting - assuming linear

[10] `http://www.automaticstatistician.com/`

relationships in tensor data without any missing data (Chapter 3). Insights gained in this Chapter about heteroskedasticity motivate Chapter 4, where we generalise causal discovery to non-linear context and allowing these (non-)-linearities to change over the space of input as it could be expected through time. The latent influences assumed in this Chapter lead to the following question: which latent causal influences can really be inferred for time series? In Chapter 5 we will investigate this and we will show how we can find latent causal relations in form of symbolic statements.

3 | Multidimensional Causal Discovery

We start with this chapter, because it implicitly exploits dynamic noise distributions, without explicating them in the model. This presents the simplest way to approach the problem of integrating dynamics into functional causal discovery which will be fine-grained in later chapters. In the following, we will motivate a generalisation of causal discovery over time in large multidimensional databases. We will explain linear non-Gaussian models and how they can be used for the learning of causal DAGs. We will introduce a tensor formulation of the data we are interested in. We will integrate known approaches of causal discovery with K-dimensional tensors to discover cause and effect in multidimensional settings. Finally, we will evaluate and analyse the work presented in this chapter. Code for the approach presented is available[1].

3.1 Generalisation of Causality over Time

Causal discovery seeks to develop algorithms that learn the structure of causal relations from observation. Such algorithms are gaining increasing importance since, as argued in (Tenenbaum et al., 2011), producing rich causal models computationally can be key to creating human-like artificial intelligence. Diverse problems in domains such as aeronautical engineering, social sciences and biomedical databases (Spirtes et al., 2010) have acted as motivating applications for causal discovery. These applications have created new insights by applying suitable algorithms to large amounts of non-experimental data, typically collected via the internet and/or recorded in databases.

We are motivated by a class of causal discovery problems, characterised by the need to analyse large and non-experimental data sets, where the observed data of interest are recorded as continuous-valued variables. For instance, consider the application of causal discovery algorithms to large biomedical databases recording data of diabetic

[1] `http://schaechtle.com/Software.html`

patients. In such databases we may wish to find causal relations between variables such as the administration of medications like insulin dosage and the effects it has on diabetes management, for example, the patients' glucose level (Kafalı et al., 2013). We are particularly concerned with data sets that are multidimensional, for example, consider insulin dosage and glucose level measurements for different patients over time. Insulin dosage and glucose level are continuous-valued variables. Patients, variables for a patient, and time are dimensions.

We have already seen in the previous chapter that a convenient way to represent cause-and-effect relations between variables is as directed edges between nodes in a graph. Such a graph, understood as a Bayesian Network (Pearl, 1988), allows us to factorise probability distributions of variables by defining one conditional distribution for each node given its causes. A more expressive representation of cause-and-effect relations is based on generative models that associate functions to variables, with the concomitant advantages of testability of results, checking equivalence classes between models and identifiability of causal effects (Pearl, 2000; Spirtes et al., 2000).

We are taking a generative modelling approach to discover cause-and-effect relations for multidimensional data. Our starting point is the generative model LiNGAM (Shimizu et al., 2006) as it relies on Independent Component Analysis (ICA) (Hyvärinen and Oja, 2000), which in turn is known to be easily extensible to data with multiple-dimensions. Multidimensional extensions for time series data using LiNGAM exist (Hyvärinen et al., 2008, 2010; Kawahara et al., 2011). However, for data sets such as the one for diabetic patients mentioned before, it is unclear how to answer even simple questions like *Does insulin dosage have an effect on glucose values?* There is at least one main problem that we see, namely, how to directly include more than one patient into the model learning process, as these approaches fit one model to a time series at a time. What we need here is an algorithm that abstracts away from selected dimensions of the data while preserving the causal information that we seek to discover.

In this chapter we present Multidimensional Causal Discovery (MCD) which we will introduce in the next sections. It discovers simple acyclic graphs of cause-and-effect relations between variables independently of the dimensions of the data. MCD integrates LiNGAM with tensor analytic techniques (Kolda and Bader, 2009) to support causal discovery in multidimensional data that was not possible before. MCD relies on a statistical decomposition that flattens higher dimensional data tensors into matrices and preserves the causal information. MCD is therefore suitable for structure learning of causal graphical models, where a causal relation can be generalised beyond dimension, for example, over all points in time. However, we can include not only time as a dimension, but also space or viewpoints, still resulting in plain knowledge representation

of cause-and-effect. This is crucial for an intuitive understanding of time series in the context of causal graphs. As part of our contribution we also prove that we can determine simple causal relations independently of the dimensionality of the data. We specify an algorithm for MCD and we evaluate the algorithm's performance.

We will structure this chapter as follows. The background on LiNGAM and the (multi) linear transformations relevant to understand the work will be reported in section 3.2. Then in section 3.4 we will introduce MCD for causal discovery in multi-linear data. The algorithm will be extensively evaluated on synthetic data as well as on real-world data in section 3.5. Apart from synthetic data, here, we will show the benefits of our method for time series data analysis within application domains such as medicine, meteorology and climate studies. Related work will be discussed in section 3.6, where we will put our approach into the scientific context of the multidimensional causal discovery. We will summarise the advantages of MCD in section 3.7.

3.2 Non-Gaussian Models

LiNGAM is a method for finding the instantaneous causal structure of non-experimental matrix data (Shimizu et al., 2006). By instantaneous we mean that causality is not time-dependent and is modelled in terms of functional equations (Pearl, 2000). The underlying functional equation for LiNGAM is formulated as in the work by Shimizu and colleagues (2005):

$$x_i = \sum_{k(j)<k(i)} b_{ij} x_j + e_i + c_i \tag{3.1}$$

with

$$\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{e}, \tag{3.2}$$

put together as

$$\mathbf{x} = \mathbf{A}\mathbf{e} \tag{3.3}$$

where

$$\mathbf{A} = (\mathbf{I} - \mathbf{B})^{-1}. \tag{3.4}$$

The observed variables are arranged in a causal order denoted by $k(j) < k(i)$. The value assigned to each variable x_i is a linear function of the values already assigned to the earlier variables x_j, plus a noise term e_i, and an optional constant term c_i that we will omit from now on. $\mathbf{x}$ is a column vector of length m. $\mathbf{e}$ is an error vector. Non-Gaussian error distributions are assumed. This has an empirical advantage since a change in variance of some Gaussian noise distribution (e.g.: over time) will induce non-Gaussian noise (Hyvärinen et al., 2010). The non-Gaussianity assumption yields also a practical advantage resulting in one unambiguous graph instead of a set of possible

graphs.

We know that a directed acyclic graph (DAG) can be expressed as a strictly lower triangular matrix (Bollen, 1989). Shimizu and colleagues define strictly lower triangular as lower triangular with all zeros at the diagonal (2006). Here, LiNGAM exploits the fact that we can permute an unknown matrix into a strictly lower triangular matrix, given enough entries in this matrix are zero. The matrix which is permuted in LiNGAM is the result of an ICA with additional processing.

Linear transformations in data, such as ICA, can reduce a problem into simpler, underlying components suitable for analysis. Regarding assumptions such as linearity and noise we can have the full spectrum of possible models. For our purposes, we are looking into methods that can be suitably integrated in causal discovery algorithms. The basis of each of these methods is a latent variable model. Here, we assume the n-dimensional data to be "generated" by $l \leq m$ latent (or hidden) variables. We can compute the data matrix by linearly mixing these components (Bishop, 2006; Hyvärinen and Oja, 2000):

$$x_j = a_{j1}y_1 + ... + a_{jm}y_m \tag{3.5}$$

for all j with $j \in [1, m]$. Similar for the complete sample:

$$\mathbf{x} = \mathbf{A}\mathbf{y}. \tag{3.6}$$

The above equation corresponds to (3.3), where $\mathbf{e} = \mathbf{y}$. We can extend this idea to the entire data set $\mathbf{X}$ as:

$$\mathbf{X} = \mathbf{A}\mathbf{Y}. \tag{3.7}$$

$\mathbf{X}$ is an $m \times n$ data matrix where m is the number of variables n is the number of cases or samples .

ICA builds on the sole assumption that the data is generated by a set of statistically independent components. Assuming such independence, we can transform the axis system determined by the m variables so that we can detect m independent components. We can use this in LiNGAM because the order of the independent components in ICA cannot be determined. As explained in (Hyvärinen and Oja, 2000), the reason for this indeterministic nature is that, since both $\mathbf{y}$ and $\mathbf{A}$ are unknown, one can freely change the order of the terms in the sum (3.5) and call any of the independent components the first one. Formally, a permutation matrix $\mathbf{P}$ and its inverse can be substituted in the model to give:

$$\mathbf{x} = \mathbf{B}\mathbf{P}^{-1}\mathbf{P}\mathbf{y}. \tag{3.8}$$

The elements of $\mathbf{P}\mathbf{y}$ are the original independent variables $\mathbf{y}$, but in another order. The matrix $\mathbf{B}\mathbf{P}^{-1}$ is just a new unknown mixing matrix, to be solved by the ICA algorithm.

We now describe the LiNGAM algorithm (LiNGAM discovery algorithm, Shimizu et al., 2005, page 3):

1. Given an $m \times n$ data matrix $\mathbf{X}$ ($m < n$), where each column contains one sample vector $\mathbf{x}$, first subtract the mean from each row of $\mathbf{x}$, then apply an ICA algorithm to obtain a decomposition $\mathbf{X} = \mathbf{AS}$ where $\mathbf{S}$ has the same size as $\mathbf{X}$ and contains in its rows the independent components. From now on we will work with (3.9):

$$\mathbf{W} = \mathbf{A}^{-1}. \tag{3.9}$$

2. Find the permutation of rows of $\mathbf{W}$ which yields a matrix $\tilde{\mathbf{W}}$ without any zeros on the main diagonal. In practice, small estimation errors will cause all elements of $\mathbf{W}$ to be non-zero, and hence the permutation is sought which minimises (3.10):

$$\sum_i \frac{1}{|\tilde{W}_{ii}|}. \tag{3.10}$$

3. Divide each row of $\tilde{\mathbf{W}}$ by its corresponding diagonal element, to yield a new matrix $\tilde{\mathbf{W}}'$ with all ones on the diagonal.

4. Compute an estimate $\hat{\mathbf{B}}$ of $\mathbf{B}$ using

$$\hat{\mathbf{B}} = \mathbf{I} - \tilde{\mathbf{W}}'. \tag{3.11}$$

5. To find a causal order, find the permutation matrix $\mathbf{P}$ (applied equally to both rows and columns) of $\hat{\mathbf{B}}$ yielding

$$\tilde{\mathbf{B}} = \mathbf{P}\hat{\mathbf{B}}\mathbf{P}^T, \tag{3.12}$$

   which is as close as possible to strictly lower triangular. This can be measured for instance using $\sum_{i \leq j} \tilde{B}_{ij}^2$.

Next, we will describe the background of how to represent multidimensional data.

3.3 Causal Discovery in Tensor Data

A tensor is a multi-way array or multidimensional array.

Definition (Cichocki et al., 2009) Let $I_1, I_2, \ldots, I_K \in K$ denote upper bounds. A tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2, \ldots, I_K}$ of order K is a K-dimensional array where elements $y_{i_1, i_2, \ldots, i_k}$ are indexed by $i_k \in \{1, 2, \ldots, I_k\}$ for k with $1 \leq k \leq K$.

Tensor analysis is applied in data sets with a high number of dimensions, other than the conventional matrix data (see Figure 3.1). An example where tensor analysis can

be applicable is time series in medical data. Here, we have a number of patients n, a number of treatment variables m such as medication and symptoms of a disease, and t discrete points in time at which the treatment data for different patients have been collected. This makes one $n \times m$ data matrix for each point in time t or a tensor of the dimension $n \times m \times t$.

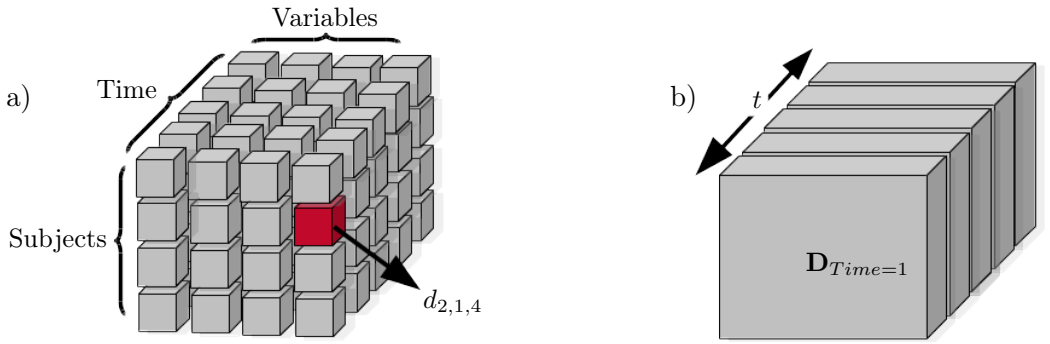


Figure 3.1: a) A three-dimensional tensor: with subjects (dimension 1), time (dimension 2), and variables (dimension 3) yielding a cube (instead of a matrix). b) We can frontally slice the data - each slice represents a snapshot of the variables for a fixed point in time.

Definition (Kolda and Bader, 2009) The order of a tensor is the number of its dimensions, also known as ways or modes.

We also need to define the n-dimensional tensor product.

Definition (Cichocki et al., 2009) The mode-n tensor matrix product $\mathcal{X} = \mathcal{G} \times_n A$ of a tensor $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N}$ and a matrix $\mathbf{A} \in \mathbb{R}^{I_n \times J_n}$ is a tensor
$\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_{n-1} \times I_n \times J_n \times \ldots \times J_N}$, with elements

$$x_{j_1, j_2, \ldots, j_{i-1}, i_n, j_n, \ldots, j_N} = \sum_{j_n=1}^{J_n} \mathcal{G}_{j_1, j_2, \ldots, j_N} a_{i_n j_n}. \tag{3.13}$$

Tensor decomposition can be described as a multi-linear extension of Principal Component Analysis (PCA), for a K-dimensional tensor:

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U_1} \times_2 \mathbf{U_1} \times_3 \ldots \times_k \mathbf{U_k}. \tag{3.14}$$

$\mathcal{G}$ is the core-tensor, that is the multidimensional extension of the latent variables $\mathbf{Y}$, with $\mathbf{U_1}, \cdots, \mathbf{U_k}$ being orthonormal.

Unlike for matrix decomposition, there is no trivial solution for computing the tensor decomposition. We use Alternating Least Square (ALS)-methods (e.g.: Tucker-ALS (Kolda and Bader, 2009)), since efficient implementations are available (Bader

et al., 2012). The decomposition can be optimised in terms of the components of a factorisation for every dimension iteratively (De Lathauwer et al., 2000).

In practical applications, it is useful to work with a projection of the actual decomposition. By projection, we mean a mapping of the information of an arbitrary tensor onto a second order tensor (a matrix). Such a mapping can be achieved using a linear operator represented as a matrix, which from now on we will refer to as projection matrix. ALS optimisation works on a projected decomposition as well. However, the resulting projection allows us to apply well-known methods for matrix data on very complex data sets.

K-dimensional Independent Component Analysis. We can determine an extension of ICA which is K-dimensional (Vasilescu and Terzopoulos, 2005). We can decompose a tensor $\mathcal{X}$ as the k-dimensional product of k matrices A_k and a core tensor $\mathcal{G}$:

$$\mathcal{X} = \mathcal{G} \times_{\mathbf{1}} \mathbf{A_1} \times_{\mathbf{2}} \mathbf{A_2} \times_{\mathbf{3}} ... \times_{\mathbf{k}} \mathbf{A_k}. \tag{3.15}$$

To extend ICA for higher-dimensional problem domains, we first need to relate ICA with PCA.

$$\begin{aligned} \mathbf{X} &= \mathbf{U\Sigma V^T} \\ &= (\mathbf{UH^{-1}})(\mathbf{H\Sigma V^T}) \\ &= \mathbf{AY}. \end{aligned} \tag{3.16}$$

$\mathbf{H}$ is a square matrix. Here we compute PCA with SVD (Singular Value Decomposition, a useful way to factorise a matrix; see e.g. Wall et al., 2003) as $\mathbf{U\Sigma V^T}$. For the K-dimensional ICA, we can make use of (3.16) and define the following sub-problem:

$$\begin{aligned} \mathbf{X_{(k)}} &= \mathbf{U_k \Sigma_k V_k^T} \\ &= (\mathbf{U_k H_k^{-1}})(\mathbf{H_k \Sigma_k V_k^T}) \\ &= \mathbf{A_k Y_k}. \end{aligned} \tag{3.17}$$

This sub-problem is due to (Vasilescu and Terzopoulos, 2005), who argue that the core tensor $\mathcal{G}$ enables us to compute the coefficient vectors via a tensor decomposition using a K-dimensional SVD algorithm.

3.4 From LiNGAM to MCD

The main idea of our work is to integrate LiNGAM with K-dimensional tensors in order to efficiently discover causal dependencies in multidimensional settings, such as time series data. The intuition behind MCD is that we want to flatten the data, that is decompose the data and project the decomposition on a matrix (see Fig. 3.2).

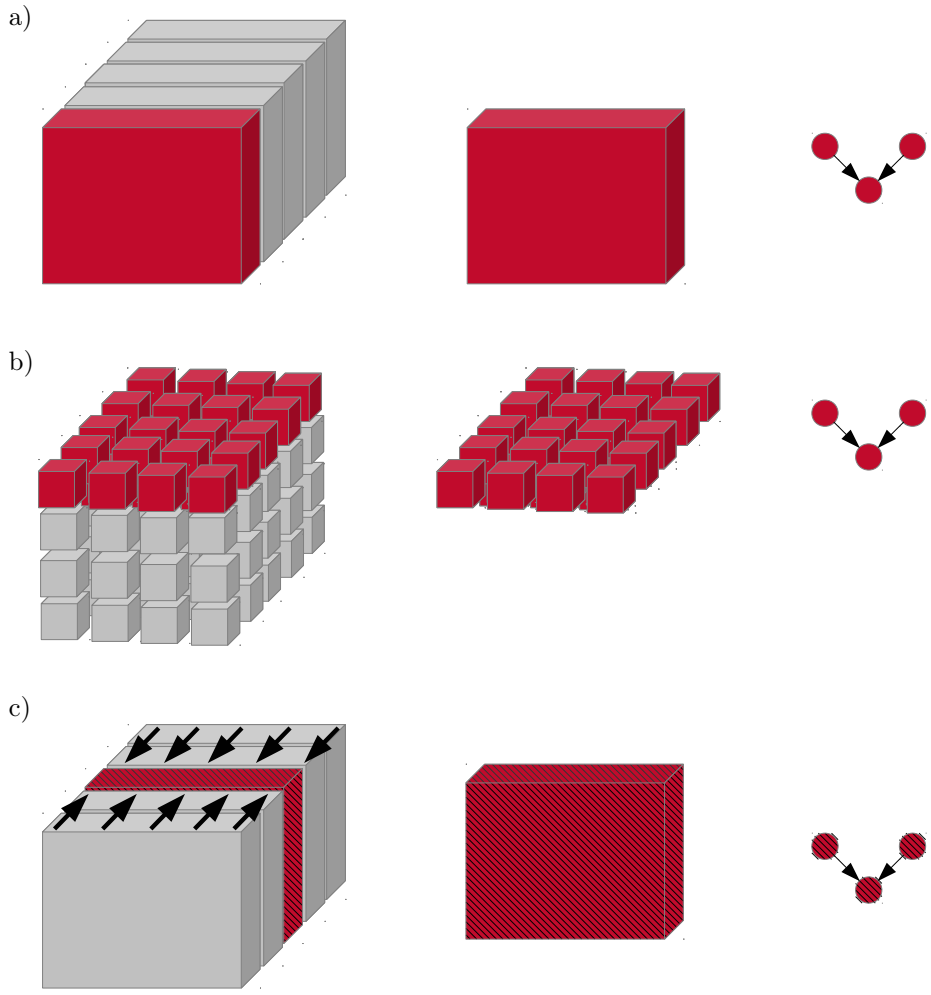


Figure 3.2: Causal analysis of time series data in tensor form. As before, the dimensions are subjects (dimension 1), time (dimension 2), and variables (dimension 3). (a) LiNGAM does not take into account temporal dynamics; it only inspects a snapshot, ignoring temporal correlations. (b) LiNGAM extensions investigate several points in time for one case. (c) MCD flattens the data whilst preserving the causal information and then applies linear causal discovery.

Definition Meta-dimensionality reduction is the process of reducing the order of a tensor via optimising the equation $\mathcal{X} = \mathcal{Y}_{Tucker} \times_1 \mathbf{U_1} \times_2 ... \times_k \mathbf{U_k}$ so that we can compute $\mathbf{X} = \mathcal{Y}_{Tucker} \times_1 \mathbf{U_1} \times_2 ... \times_{k-1} \mathbf{U_{k-1}}$.

After a meta-dimensionality reduction step, we can apply LiNGAM directly and, as a result, reduce the temporal complexity of causal inferences (compare Figures 3.3(a) and 3.3(b)). We interpret the output graph of the algorithm as an indicator of cause-and-effect that is significant according to the tensor analysis for a sufficient subset of all the tensor slices. However, before applying LiNGAM, we need to ensure that the information on causal dependencies is preserved when meta-dimensionality reduction has been applied.

Theorem 3.4.1. *Let $\mathcal{X} = \mathcal{G} \times_1 \mathbf{S_1} \times_2 \mathbf{S_2} ... \times_k \mathbf{S_k}$ be any decomposition of a data tensor $\mathcal{X}$ that can be computed using SVD. Let $\mathbf{X}$ be the projection (mapping) of $\mathcal{X}$ where we remove one or more tensor dimensions for meta-dimensionality reduction. The independent components of tensor data subspaces which are to be permuted are independent of previous projections in the meta-dimensionality reduction process.*

Proof. The decomposition is computed independently for all dimensions (see (3.17)) using SVD. Furthermore, we know that the tensor matrix product is associative:

$$\mathcal{D} = (\mathcal{E} \times_1 \mathbf{A} \times_2 \mathbf{B}) \times_3 \mathbf{C} = \mathcal{E} \times_1 (\mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}). \tag{3.18}$$

Therefore, as long as it is computed with SVD we can establish a relation between Tucker-Decomposition and Multi-Linear ICA. Vasilescu and Terzopoulos define this relation in the following way (2005)):

$$\begin{aligned} \mathcal{X} &= \mathcal{Y}_{Tucker} \times_1 \mathbf{U_1} \times_2 ... \times_k \mathbf{U_k} \\ &= \mathcal{Y}_{Tucker} \times_1 \mathbf{U_1}\mathbf{H_1^{-1}}\mathbf{H_1} \times_2 ... \times_k \mathbf{U_k}\mathbf{H_k^{-1}}\mathbf{H_k} \\ &= (\mathcal{Y}_{Tucker} \times_1 \mathbf{H_1} ... \times_k \mathbf{H_k}) \times_1 \mathbf{A_1} \times_2 ... \times_k \mathbf{A_k} \\ &= \mathcal{Y} \times_1 \mathbf{A_1} \times_2 ... \times_k \mathbf{A_k} \end{aligned} \tag{3.19}$$

where

$$\mathcal{Y} = \mathcal{Y}_{Tucker} \times_1 \mathbf{H_1} ... \times_k \mathbf{H_k}. \tag{3.20}$$

Now, when we look at an arbitrary projection of a data tensor $\mathcal{X}$ to a data matrix $\mathbf{X}$, we first need to generalise (3.19) for projections. This can be done by simply replacing the inverse with the Moore-Penrose pseudo-inverse (denoted with †), which is defined as:

$$\mathbf{H}^\dagger = (\mathbf{H^T H})^{-1}\mathbf{H^T} \tag{3.21}$$

where

$$\mathbf{H}\mathbf{H}^\dagger\mathbf{H} = \mathbf{H} \tag{3.22}$$

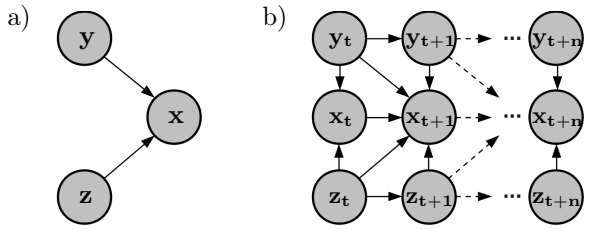


Figure 3.3: a) The causal graph determined by MCD b) A causal graph determined by extensions of LiNGAM for time series.

We can then rewrite (3.19) to the following:

$$\begin{aligned}\mathcal{X} &= \mathcal{Y}_{Tucker} \times_1 \mathbf{U_1} \times_2 ... \times_k \mathbf{U_k} \\ &= \mathcal{Y}_{Tucker} \times_1 \mathbf{U_1}\mathbf{H_1}^{\dagger}\mathbf{H_1} \times_2 ... \times_k \mathbf{U_k}\mathbf{H_k}^{\dagger}\mathbf{H_k} \\ &= (\mathcal{Y}_{Tucker} \times_1 \mathbf{H_1} ... \times_k \mathbf{H_k}) \times_1 \mathbf{A_1} \times_2 ... \times_k \mathbf{A_k} \\ &= \mathcal{Y} \times_1 \mathbf{A_1} \times_2 ... \times_k \mathbf{A_k}\end{aligned} \tag{3.23}$$

Finally, we look at the projection that we compute using Tucker-Decomposition. Here we project our tensor into a matrix by reducing the order of a K-dimension tensor. If one thinks of $\mathbf{H_k}$ in terms of a projection matrix, this can be expressed as:

$$\mathcal{X} = (\mathcal{Y}_{Tucker} \times_k \mathbf{H_k}) \times_1 \mathbf{U_1} \times_2 ... \times_{k-1} \mathbf{U_{k-1}} \times_k \mathbf{U_k}\mathbf{H_k}^{\dagger}. \tag{3.24}$$

We see that compared to (3.23) we find projections that allow the meta-dimensionality reduction in two places: at $(\mathcal{Y}_{Tucker} \times_k \mathbf{H_k})$ and at $\mathbf{U_k}\mathbf{H_k}^{\dagger}$. Accordingly, we define the projection of the decomposition to the matrix data $\mathbf{X}$ as:

$$\mathbf{X} = (\mathcal{Y}_{Tucker} \times_k \mathbf{H_k}) \times_1 \mathbf{U_1} \times_2 ... \times_{k-1} \mathbf{U_{k-1}} \tag{3.25}$$

Here, we see that $U_1 ... U_{k-1}$ are not affected by the projection. Therefore, by computing the $H_1 ... H_{K-1}$ we can still find the best result for ICA on our flattened tensor data, if we apply it on the projection. This means that if we want to apply multi-linear ICA with the aim on having statistically independent components in meta-dimensionality reduced space, we can resort to Tucker-Decomposition for reducing the order of the tensor - until we finally compute the interesting part that is used by ICA for LiNGAM. Thus, the theorem is proven.

On these grounds, we can see that LiNGAM works, without making additional assumptions on the temporal relations between variables, if it is combined with any decomposition that can be expressed in terms of SVD. The algorithm is summarised in

Algorithm 2, the variable-mode contains the causes and effects that we try to find.

Algorithm 2. Multidimensional Causal Discovery

1: **procedure** MCD($\mathcal{X}$, sample-mode, variable-mode)
2: $k_{sm} \leftarrow$ sample-mode
3: $k_{vm} \leftarrow$ variable-mode
4: $\mathcal{Y}_{projection}, \mathbf{U_{k_{sm}}}, \mathbf{U_{k_{vm}}} \leftarrow$ Tucker-ALS($\mathcal{X}$)
5: $\mathbf{X} \leftarrow \mathcal{Y}_{projection} \times_{k_{sm}} \mathbf{U_{k_{sm}}} \times_{k_{vm}} \mathbf{U_{k_{vm}}}$
6: $\mathbf{B} \leftarrow$ LiNGAM($\mathbf{X}$)
7: **end procedure**

end

It is worth noting how MCD exploits Gaussianity: unlike plain LiNGAM, we do not forbid Gaussian noise totally. In the tensor-analytically reduced dimensions, we can have Gaussian noise, which does not influence MCD, as long as we have non-Gaussian noise in the non-reduced dimensions to identify the direction of cause-and-effect. Having Gaussian noise in one dimension and non-Gaussian noise in the other is an empirical necessity if time is involved (Hyvärinen et al., 2010).

3.5 Evaluation

3.5.1 Experiments with Synthetic Data

We have simulated a 3-dimensional tensor with dimensions cases, variables and time. 5000 cases with 5 variables and 50 points in time have been produced. For each case, we have created the time series with a special case of a structural autoregressive model:

$$\mathbf{X_t} = \mathbf{c} + \sum_{\mathbf{p=0}}^{\mathbf{P}} \phi_{\mathbf{p}} \mathbf{X_{t-p}} + \epsilon_t \tag{3.26}$$

with

$$\phi_{\mathbf{p}} = \mathbf{B} \tag{3.27}$$

and

$$\epsilon_{\mathbf{t}} \sim \mathcal{N},\ \mathbf{c} \sim \mathcal{SN} \tag{3.28}$$

where $\mathcal{SN}$ is a sub or super-Gaussian distribution. In contrast to the classical autoregressive model we start indexing at $\mathbf{p} = 0$. This allows us to include instantaneous and time-lagged effects. Furthermore, we allowed each of the nodes (variables) to have either one or two incoming edges at random. In that manner we created three different kinds of data sets, one with time-lag $\mathbf{p} = 1$, one with time-lag $\mathbf{p} = 2$ and one with time-lag $\mathbf{p} = 3$ to test the MCD algorithm with. Each kind we created 500 times, so

that we could test the algorithm on a number of different data sets. We found the output of the algorithm to be correct in 73.00 % of all the data sets with time-lag $\mathbf{p} = 1$, 69.20 % with $\mathbf{p} = 2$ and 68.20 % with time-lag $\mathbf{p} = 3$. The decrease in accuracy can be explained by the increasing complexity of the time series function that comes with increasing $\mathbf{p}$. The algorithm's output was determined to be incorrect if there was any type of structural error in the graph, that is false positive or false negative findings. Due to this very conservative measure, we could achieve very high precision (ca. 99 %) and recall (ca. 96 %) when investigating the total number of correct classifications, that is whether there is a cause-effect-relation between one variable and another (i.e. $a \rightarrow b$ true or false). For pruning the edges in the LiNGAM part of the algorithm, we used a simple re-sampling method (described in Shimizu et al. 2006).

3.5.2 Application to Real-world Data

To show how MCD works on real-world problems, we applied it to three different real-world data sets. Where possible, we also applied an implementation of multi-trial version of Granger Causality (MTGC) (Seth, 2010) to compare MCD's results to something known to the community. Also, from all related methods, Granger Causality is the only method where there is an extension available for multiple realisations of the same process (Ding et al., 2006). However, the multiple realisations are interpreted in terms of repetitive trials with a single subject or case. This suggests dependence due to repetition instead of the desired independence of cases. For example, if we look at a number of subjects and their medical treatment over time, we expect the subjects to be independent from each other.

First of all, we applied MCD and MTGC to a data set on Diabetes (Frank and Asuncion, 2010). Here, the known ground truth was Insulin $\rightarrow$ Glucose. Glucose curves and insulin dose were analysed for 69 patients - the number of points in time differed from patient to patient, thus we had to cut them all to similar size. MCD successfully found the causal ground truth, MTGC did not and resulted in a cyclic graph.

Secondly, we investigated a data set with two variables, 72 points in time, 16 different places. The two variables were ozone and radiation with the assumed ground truth that radiation has an causal effect on ozone.[2] Again, MCD found the causal ground truth and MTGC did not and resulted in a cyclic graph.

Finally, we tested the algorithm on meteorological data[1]. 10,226 samples have been taken for how the weather conditions of one day cause the weather conditions of the second day. The variables that were measured were mean daily air temperature, mean

[2]The causal ground truth was given, data taken from `https://webdav.tuebingen.mpg.de/cause-effect/`

daily pressure at surface, mean daily sea level pressure and mean daily relative humidity. The ground truth was that the conditions on day t affect the conditions on day $t+1$ which was found by MCD. We did not apply MTGC here because of its conceptual dependency to the time-dimension.

3.6 Related Work

The most well-known example of causality for time series is Granger Causality (Granger, 1969). Granger affiliates his definition of causality with the time-dimension. Statistical tests regarding predictive power, when including a variable, detect an effect of this variable. Granger Causality cannot incorporate instantaneous effects, which is often cited as a drawback (Peters et al., 2012). MCD complements Granger Causality in this. Likewise, this is the case for transfer entropy (TE) (Schreiber, 2000): proven equivalent to Granger Causality for the case of Gaussian noise (Barnett et al., 2009), TE is bound to the notion of time. TE cannot detect instantaneous effects because potential asymmetries in the underlying information theory models are only due to different individual entropies and not due to information flow or causality.

Entner and Hoyer make use of similarities between causal relations over time to extend the Fast Causal Inference (FCI) algorithm (Spirtes, 2001) for time series (2010)). In contrast to MCD, FCI supports the modelling of latent confounding variables and it does not exploit the non-Gaussian noise assumptions.

The closest approaches to MCD are the approaches connecting LiNGAM to models of the Autoregressive-moving-average model (ARMA) class. For example, a link was established between LiNGAM and structural vector autoregression (Hyvärinen et al., 2008, 2010) in the context of non-Gaussian noise. The authors focus on an ICA interpretation of the autoregressive residuals. This was generalised for the entire ARMA class (Kawahara et al., 2011). These methods can be seen as a LiNGAM-based generalisation of Granger Causality, since they can take into account time-lagged and instantaneous effects. Similarly, the Time Series Models with Independent Noise, which can be used in a multi-variate, linear, non-linear setting, with or without instantaneous interactions (Peters et al., 2012). Instantaneous interactions can be observed either in situations with resolution of observations not high enough to perceive a time lag between cause and effect or in systems where an effect appears at the same point in time and without any time lag when the cause is changing (this is common for example in quantum physics).

The main difference between our approach and these LiNGAM extensions (and the other related work discussed earlier in this section) is the possibility to directly include

a number of dimensions in the analysis using the MCD algorithm. Previous research takes into account single time series, but does not allow abstracting away modes to produce simple and clear cause-and-effect relations. Here, it is unclear how to analyse multiple cases of multi-variate time series for causality. Only for Granger Causality there are methods available for a direct comparison of performance.

3.7 Summary

In this chapter we have proposed MCD, a method for learning causal relations within high-dimensional data, such as multi-variate time series, as they are typically recorded in non-experimental databases. The contribution presented in this chapter is the implementation of an algorithm that integrates linear non-Gaussian additive models (LiNGAM) with tensor analytic techniques and opens up new ways of understanding causal discovery in multidimensional data that was previously impossible. We have shown how the algorithm relies on a statistical decomposition that flattens higher dimensional data tensors into matrices. This decomposition preserves the causal information and is therefore suitable to be included in the structure learning process of causal graphical models, where a causal relation can be generalised beyond dimension, for example, over all points in time. Related methods either focus on a set of cases for instantaneous effects or look at a single case for effects at certain points in time. We have also evaluated the resulting algorithm and discussed its performance both with synthetic and real-world data.

One reason why the model works so well is related to heteroskedasticity, that is the dynamic variation of noise. It could be that the system of functional equations that model the data generating process consist of Gaussian white noise and for every point in time a different variance parameter applies to the Gaussian noise distribution. This has been shown to introduce non-Gaussianity (Hyvärinen et al., 2010) and therefore renders the causal relation discoverable. Is there a way to exploit this as a criterion for causal discovery?

4 | Flexible Causal Discovery

The last chapter showed how MCD exploits Gaussian noise by tensor-analytical dimensionality reduction. We argued that dynamical systems can introduce heteroskedasticity and heteroskedasticity results in non-Gaussian noise. The dimensionality reduction projected dimensions with random variables that follow a Gaussian noise distribution on lower dimensionality inducing non-Gaussian random variables. Essentially, we are simplifying the underlying Additive Noise Model (ANM). Thus, we remove heteroskedasticity and turn it into a different functional form to support causal discovery. One could argue that this is suboptimal. While reducing dimensionality we could loose information that is potentially instructive for causal discovery. For example, with MCD we can fabricate a case where the functional relation is periodically oscillating in a certain proportion to its periodically changing noise distribution. In this case, the collapsed data would be best described by a Gaussian (as opposed to a super-Gaussian), rendering the true causal model not identifiable for MCD.

Instead of removing this information, we could deal with dynamical systems through a generalisation of the notion of ANMs as presented in section 2.4 to account for heteroskedastic noise. Would such a generalisation improve performance? This thesis suggests that it does. We will investigate the previously introduced notion of functional ANM more closely. To do this, we need to take a step back from the previous approach and abstract away from temporal dependence over random variables to investigate static, instantaneous effects. The dynamic aspects of the system enter the functional equations through noise, only. We model this explicitly as an ANM with a set of functional equations. The functional equations are comprised of a mean function accounting for linearity, a homoskedastic GP component accounting for static levels of noise and a heteroskedastic GP component accounting for changing levels of noise. The interplay between these components allows us to discover cause and effect without making too strong assumptions about the nature of the true underlying causal function and noise, for example linearity or strictly additive noise. It also allows us to explicitly formalise noise terms that dynamically change over the input space, as it would happen

through heteroskedasticity that is caused by temporal dependence.

In the following, we will introduce relevant results for identifiability of causal direction and relate this directly to noise distributions. We will show how heteroskedasticity and non-Gaussianity can play a role in both linear and non-linear settings. The work generalises previous approaches to ANM in cases where only two continuous-valued variables have been observed. An efficient algorithm will be designed. We will prove correctness of its inference step for discovering causal directions. It will further be validated through extensive experimentation and comparison with the state-of-the-art for both artificial and real world data.

4.1 Different Kinds of Noise and Causality

Finding structures of cause and effect is fundamental for the process of scientific discovery. The causal discovery problem amounts to developing computational mechanisms capable of inferring such underlying causal structure from observational data. Given this underlying structure, such mechanisms determine whether $\mathbf{X}$ causes $\mathbf{Y}$ or vice versa. Typically, we assume a set of functional equations that model the underlying causal structure generating the data (Pearl, 2000)

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\eta}) \tag{4.1}$$

where f is a functional relation between the variables $\mathbf{x}$ and $\mathbf{y}$ and $\boldsymbol{\eta}$ is a noise term often referred to as error.

Given a functional representation, existing work discovers the direction of cause and effect in two ways:

1. by using graphical criteria and (conditional) independence testing (Pearl et al., 1991; Spirtes et al., 2000; Spirtes, 2001), commonly known as constrained-based causal discovery (section 2.4.1); and

2. by making assumptions on functions and drawing conclusions given observations and assumptions (functional causal discovery, see e.g. Shimizu et al. 2006; Hoyer et al. 2008a; Zhang and Hyvärinen 2009; Mooij et al. 2010; Peters et al. 2011; Schaechtle et al. 2013; Janzing et al. 2012; Chen et al. 2014, the background is described in section 2.4.2).

For some cases, purely constrained-based methods cannot output a unique solution. This holds for the simplest non-trivial causal graph, viz., a graph with two nodes representing two variables, which is the main focus of this chapter. To output a decision or a Bayesian posterior belief with regards to the direction of cause and effect, we need resort to functional causal discovery.

The idea behind functional causal discovery is that we can infer the direction of cause and effect if we make certain assumptions on the data generating process. One rather strong assumption is that noise is additive, resulting in a set of equations of the form:

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\eta}. \tag{4.2}$$

The intuition behind exploiting this functional assumption is the following (Hoyer et al., 2008a; Kpotufe et al., 2014):

a) $\mathbf{y}$ can be computed as an optimally fitted function (f_1) of $\mathbf{x}$ plus a noise term independent of $\mathbf{x}$, that is $\mathbf{x} \perp\!\!\!\perp \boldsymbol{\eta}$,

b) $\mathbf{x}$ cannot be computed as an optimally fitted function (f_2) of $\mathbf{y}$ plus independent noise.

This assumption is statistically testable: using the residuals $\boldsymbol{\eta}_1 = \mathbf{y} - f_1(\mathbf{x})$ and $\boldsymbol{\eta}_2 = \mathbf{x} - f_2(\mathbf{y})$. We decide $\mathbf{x} \to \mathbf{y}$ if $\boldsymbol{\eta}_1 \perp\!\!\!\perp \mathbf{x}$ but $\boldsymbol{\eta}_2 \not\perp\!\!\!\perp \mathbf{y}$. Reversely, we decide $\mathbf{y} \to \mathbf{x}$.

Even in the very restricted case of purely additive noise, we encounter problems: if we choose f to be a linear function and assume the noise to be Gaussian distributed, then b) is violated. As a result, in a linear setting we need the noise to be non-Gaussian to ensure the direction of cause and effect can be discovered (Shimizu et al., 2006; Lacerda et al., 2008; Hyvärinen et al., 2010; Kawahara et al., 2011; Schaechtle et al., 2013). Alternatively, if we choose f to be non-linear, most approaches do not rely directly on the noise distributions. In fact, different kinds of non-linearity alone can render the relation in (4.2) discoverable (Hoyer et al., 2008a; Zhang and Hyvärinen, 2009; Mooij et al., 2010; Peters et al., 2011).

What happens if the noise is not purely additive anymore? The most important example for this is heteroskedastic noise, where the variability of the noise distribution changes over the domain of the cause. Although ubiquitous in empirical data acquisition, heteroskedasticity has not gained much attention in existing work. One reason for this could be that the mathematics of the soundness on functional causal discovery is still in its infancy and heteroskedasticity complicates the matters further. Existing work abstracts away from heteroskedastic noise and input dependent changes in the causal mechanism, for example in biological systems that have been tuned to their input distributions through evolution (as pointed out in Mooij et al. 2010). The problem we see here is that we ignore a large fraction of existing real world problems (e.g. some biological systems) by excluding heteroskedastic scenarios from the set of problems where we desire to discover cause and effect.

The contribution of this chapter is that it introduces a criterion for causal inference that allows noise to be heteroskedastic, thus reflecting different variability of noise

in the data. Also, we present a new causal data generative process and we offer improvements on inferring causal direction. Our focus is on the two-variable problem where both variables are continuous-valued. We allow latent confounders only in the form of heteroskedastically dependent noise which is an instance of latent confounders manifested as dependent error variables (Shimizu and Bollen, 2014). The resulting algorithm, FlexCD (Flexible Causal Discovery), covers a wide variety of settings because the assumed noise can be comprised of both homoskedastic and heteroskedastic components. These settings include scenarios with different measurement noise and different underlying data generating functions. The interplay of both components is key for inferring cause and effect. We also present a Bayesian model, which produces posterior probability distributions given a change in causal variables.

4.2 Theory

4.2.1 Functional Causal Discovery and Noise

Existing frameworks for causal discovery exploit different assumptions to discover cause and effect. One such assumption is that the functional relation is linear and the noise is not Gaussian distributed. The Linear Non-Gaussian Acyclic Model (LiNGAM) (Shimizu et al., 2006) relies on the Independent Component Analysis (ICA). Within certain settings of ICA, one optimises negentropy (2.11) as a measure of non-Gaussianity. This can be approximated with a contrast function (e.g. eq. (26) in Hyvärinen and Oja 2000) where one can compute the contrast for each component simultaneously with the following matrix:

$$G(\mathcal{D}^\top \mathbf{W}) = \frac{1}{a_1} log\, cosh\, a_1\, \mathcal{D}^\top \mathbf{W}. \tag{4.3}$$

$\mathcal{D}$ is the two column data matrix comprised of the vectors $\mathbf{x}$ and $\mathbf{y}$ and $1 \leq a_1 \leq 2$ to approximate negentropy. $\mathbf{W}$ is a constant matrix that transforms the data into a space where the signals are statistically as independent from each other as possible. With a permutation-optimisation algorithm, this matrix is permuted to a lower triangular matrix that represents a directed acyclic graph. Entries in this matrix are linear regression coefficients.

For the case of Gaussian noise, a kind of non-linearity is required in the underlying data (Peters et al., 2011). We differentiate between non-linear and post-non-linear settings.

In a non-linear setting, GP have been proposed to account for the non-linearity that is needed to allow the discovery of cause and effect within the context of the ANM-algorithm (Hoyer et al., 2008a; Peters et al., 2014). ANM relies on conditions a) and b) from section 4.1. The method of probabilistic latent variable models for distinguishing

between cause and effect also uses GPs for the functional relations between two variables by exploiting the complexity of the (conditional) probability distributions (Mooij et al., 2010).

In a post-non-linear setting we apply a non-linear distortion to a non-linear function plus additive noise (Zhang and Hyvärinen, 2009). The function relies on (i) the assumption that the post-non-linearity is invertible and (ii) statistical testing of conditions a) and b) of section 4.1. Also, information theoretic decision criteria are available for causal discovery (Janzing et al., 2012; Chen et al., 2014). These are based on the idea of uncorrelatedness between the two actual processes that generate $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$.

4.2.2 Heteroskedastic Gaussian Processes

Some of the functional causal discovery methods explained above resort to GPs for regression (section 2.2.3). In a standard GP model one assumes that the output is generated by a simple ANM (4.2) with $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2)$. Instead of modelling f directly, f is assumed latent and assigned a GP prior given a mean and covariance function $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ is a function of the mean of all functions that map to y_i at x_i. Often, this mean function is assumed to be 0. For this chapter, we will *not* make this common assumption. It is simple to assume a global linear model to describe the mean of the GP, e.g. by letting $m(x)$ be a linear combination of functions. f is now defined as

$$f(\mathbf{x}) = m(\mathbf{x})^\top b + f_{\mu=0}(\mathbf{x}) \tag{4.4}$$

where the noise is modelled with a zero-mean with GP $f_{\mu=0}(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ and $b \sim \mathcal{N}(\mathbf{b}, \mathbf{B})$. The residuals of a global linear model are described with a zero-mean GP. Typically, for causal discovery we assume homoskedastic Gaussian noise (e.g Hoyer et al., 2008a). However, recently, heteroskedastic noise has been introduced in the form of a Variational Heteroskedastic Gaussian Process (VHGPR) (Lázaro-Gredilla and Titsias, 2011). In the context of (4.2), a heteroskedastic GP is a modification of GP as follows:

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}')\big),\ \boldsymbol{\eta}_i \sim \mathcal{N}\big(0, r(\mathbf{x})\big), \tag{4.5}$$

thus allowing for a change of the noise distribution in the observations $\mathbf{x}$ parametrised as:

$$r(\mathbf{x}) = e^{g(\mathbf{x})} \tag{4.6}$$

with $g(\mathbf{x}) \sim \mathcal{GP}(\mu_0, k_g(\mathbf{x}, \mathbf{x}'))$ to ensure that the noise variance is positive. Note that although marginal likelihood and predictive density are not analytically tractable, they can be approximated with a variational Bayesian method (section 2.3).

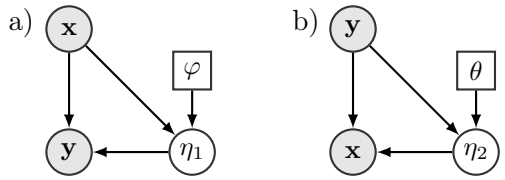


Figure 4.1: a) the heteroskedastic causal data generating process, when $\mathbf{x} \to \mathbf{y}$. b) the causal data generating process, when $\mathbf{y} \to \mathbf{x}$. Observable variables are depicted in grey, hidden ones in white. A squared node indicates an unknown vector of parameters.

4.3 FlexCD: A New Model and Algorithm

In this section, we present a novel, generalised model for causal discovery. The causal model builds on two observations about assumptions that are made in the ANM related literature.

Firstly, the algorithm for ANM (Additive Noise Model) (Hoyer et al., 2008a; Peters et al., 2014) assumes homoskedastic noise, abstracting away from scenarios where the residuals are allowed to have changing distributions. If the noise is not stable over the domain of a variable, ANM cannot cover it. This leads us to reformulate the functional equation for cause and effect which describes the hypothetical underlying data generating process (not the approximation described below for causal discovery):

$$\mathbf{y} = f_1(\mathbf{x}) + \boldsymbol{\eta}_1(\mathbf{x}, \varphi) \tag{4.7}$$

where an unknown vector of parameters φ governs the change in the noise over the input domain. The hypothesis induces a restriction on the noise. We assume a deterministic function $\boldsymbol{\eta}_1$ that maps $\mathbf{x}$ to the noise that is added. For reasons that will become clear below, we assume that the latent variable φ is Gaussian distributed and models a Gaussian prior on the space of possible noise distributions. This formulation resembles the one presented in (4.2), but it is not strictly additive anymore, since it gives rise to non-additive error terms, for example when $\boldsymbol{\eta}_1(\mathbf{x}, \varphi) = x * \varphi$, where φ can be drawn out of any Gaussian. We assume φ to be independent from $\mathbf{x}$ whereas the actual noise distribution is dependent of $\mathbf{x}$ (see Fig. 4.1).

This leads to the following constraints that we deploy for causal discovery:

$$\begin{aligned} &\mathbf{x} \perp\!\!\!\perp \boldsymbol{\varphi} \\ &\mathbf{x} \not\perp\!\!\!\perp \boldsymbol{\varphi} \mid \mathbf{e} \qquad \text{where } \mathbf{e} \sim \boldsymbol{\eta} \end{aligned} \tag{4.8}$$

Similarly, we model the anti-causal[1] direction with $\mathbf{x} = f_2(\mathbf{y}) + \boldsymbol{\eta}_2(\mathbf{y}, \theta)$.

Secondly, the work of Mooij and colleagues (2010) assumes that the distribution

[1] by anti-causal, we simply mean the non-causal direction, that is, if $X \to Y$ then we mean the direction from Y to X.

of the cause is "independent" from the causal mechanism, but as the authors note themselves, this assumption is often violated in biological systems (where the causal mechanisms may have been tuned to their input distributions through evolution).

We weaken the assumptions made in the ANM related literature by modelling the space of possible functions conditioned on the data with GPs. Deploying GPs for causal discovery is not new in this context but unlike previous research, we do not explicitly require a certain functional form such as linearity or non-linearity. More precisely, FlexCD provides a criterion that accounts for the possibility of linearity, non-linearity, homoskedastically distributed noise and noise distributions that change.

To derive why and how FlexCD works, we need to be aware of the fact that the error distribution and the parameters that govern the heteroskedasticity cannot be observed. Let φ be the prior on a set of deterministic noise functions. Assume that this prior is independent of $\mathbf{x}$ and governs the relation between input domain and noise with a finite set of parameters. Knowing the value of φ would result in knowing the function parameters that govern the noise. This would render the causal part of the model $\mathbf{x} \to \boldsymbol{\eta}_1(\mathbf{x}, \varphi)$ deterministic. For readability, we will write $\boldsymbol{\eta_1}$ for $\boldsymbol{\eta_1}(\mathbf{x}, \varphi)$. This determinism can be exploited by taking a parametric view on the non-parametric model. Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be finite sets of parameters corresponding to the noise-free latent function values at the training inputs for a GP (see Rasmussen and Williams, 2006, chapter 5). This leads to the following key assumption: assume that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, if optimally fit to the data, correspond to the function parameters with prior φ and determine the noise in the causal data generating process. In a non-parametric setting, the dependencies of these sets are $\mathbf{x} \to \boldsymbol{\alpha} \to \mathbf{y}$ in the causal direction and $\mathbf{y} \to \boldsymbol{\beta} \to \mathbf{x}$ in the anti-causal direction (see Fig. 4.2).

Janzing and colleagues (2012) show that, for a deterministic functional relationship, if $\mathbf{x}$ causes $\boldsymbol{\eta}$, then:

$$\mathbb{KL}(p(\mathbf{x})||\mathcal{E}_{\mathbf{x}}) \leq \mathbb{KL}(p(\boldsymbol{\eta})||\mathcal{E}_{\boldsymbol{\eta}}) \tag{4.9}$$

where $\mathbb{KL}(q||p)$ denotes the Kullback-Leibler divergence. This does not hold for any anti-causal deterministic functional relation. $\mathcal{E}_{\mathbf{x}}$ is chosen by Janzing and colleagues as a reference distribution where they define $\mathcal{E}$ to be an exponential manifold.

Definition *(Exponential Manifolds (Janzing et al., 2012)).* Let $\Omega \subseteq \mathbb{R}^d$ and assume a finite-dimensional vector space V of functions $f : \Omega \to \mathbb{R}$ is given. Then, V defines an exponential manifold $\mathcal{E}$ by the set of probability densities that can be written as

$$p(\mathbf{x}) \propto e^{V(\mathbf{x})}. \tag{4.10}$$

For any density p, $\mathbb{KL}(p||\mathcal{E})$ denotes the infimum of $\mathbb{KL}(p||q)$ with $q \in \mathcal{E}$. If there is a q with $\mathbb{KL}(p||\mathcal{E}) = \mathbb{KL}(p||q)$, it is called projection of p onto $\mathcal{E}$.

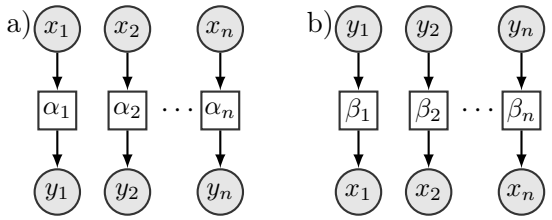


Figure 4.2: The non-parametric best guess for β, α for the parameters that govern how the noise of the causal data generating process between $\mathbf{x}$ and $\mathbf{y}$ varies over the input domain. a) how we treat the parameters in the direction $\mathbf{x} \to \mathbf{y}$. b) how we treat the parameters in the direction $\mathbf{y} \to \mathbf{x}$. β, α correspond to f, g in the VHGPR model.

Unlike Janzing and colleagues, we are interested in a latent approximation of a finite set of parameters that determine the noise. We therefore define a suitable set of reference distributions.

Definition *(Latent Exponential Manifolds).* Let Ω and V be defined as before. We put a prior belief on the space of deterministic functions that govern the heteroskedasticity in the data generating process. With the help of this prior we treat the space of possible parameters as hidden values themselves. We can define for latent distributions what (4.10) is for observed distributions:

$$p(\boldsymbol{\alpha}) \propto \mathbb{E}_{\mathbf{y}}[e^{V(\boldsymbol{\alpha}|\mathbf{y})}]. \tag{4.11}$$

Given this prior, we compute the expected values of the parameters that we treat as values of a hidden variable. This is illustrated in Fig. 4.2.

Given the two data generating models we see that we must have one of two deterministic relationships. Either $\mathbf{x} \to \boldsymbol{\eta_1}$ or $\mathbf{y} \to \boldsymbol{\eta_2}$ (see Fig. 4.1).

Theorem 4.3.1. *Whenever $\mathbf{x} \to \mathbf{y}$ is the ground truth, then it holds that:*

$$\mathbb{KL}(p(\boldsymbol{\beta})||\mathcal{E}_{\boldsymbol{\beta}}) \leq \mathbb{KL}(p(\boldsymbol{\alpha})||\mathcal{E}_{\boldsymbol{\alpha}}) \tag{4.12}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are defined as the optimal parameters for the functions for the underlying causal data generating process as defined in (4.7).

Proof: Due to (4.9) our definition of latent exponential manifolds, the following holds when $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are optimally fit and given the above described determinism prevails:

$$\mathbb{KL}(p(\mathbf{x})||\mathcal{E}_{\mathbf{x}}) \leq \mathbb{KL}(p(\boldsymbol{\alpha})||\mathcal{E}_{\boldsymbol{\alpha}}) \tag{4.13}$$

and

$$\mathbb{KL}(p(\boldsymbol{\alpha})||\mathcal{E}_{\boldsymbol{\alpha}}) \leq \mathbb{KL}(p(\mathbf{y})||\mathcal{E}_{\mathbf{y}}). \tag{4.14}$$

This holds since $\boldsymbol{\alpha} \neq \boldsymbol{\varphi}$. We can read from the graph that represents the nonparametric estimation that in this GP setting, $\boldsymbol{\alpha}$ blocks the path between $\mathbf{x}$ and $\mathbf{y}$. The anti-causal

hypothesis indicates:

$$\mathbb{KL}(p(\mathbf{y})||\mathcal{E}_{\mathbf{y}}) \leq \mathbb{KL}(p(\boldsymbol{\beta})||\mathcal{E}_{\boldsymbol{\beta}}) \tag{4.15}$$

and

$$\mathbb{KL}(p(\boldsymbol{\beta})||\mathcal{E}_{\boldsymbol{\beta}}) \leq \mathbb{KL}(p(\mathbf{x})||\mathcal{E}_{\mathbf{x}}). \tag{4.16}$$

However, from (4.7), the described independencies and the deterministic nature of the $\boldsymbol{\eta}_1$ function, (4.15) is violated. Consequently, due to transitivity, the theorem is proven. □

4.3.1 The FlexCD Algorithm

How can we deploy inequality (4.12) so that it is useful for causal discovery? We need to find a formulation that will not only guarantee (4.12) to hold, but that is also suitable to model (4.7). Therefore, we choose the causal process to be modelled via GPs with a simple modification of (4.4):

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\eta} = m(\mathbf{x})^\top b + f_{\mu=0}(\mathbf{x}) + \boldsymbol{\eta} \tag{4.17}$$

where $f_{\mu=0}(\mathbf{x}) \sim \mathcal{GP}(0, k_f(\mathbf{x}, \mathbf{x}'))$ and $\boldsymbol{\eta} \sim \mathcal{N}(0, r(\mathbf{x}))$ with $r(\mathbf{x})$ as in (4.6). The first summand in the right-hand side of the equality accounts for any linear relationships; we will only consider the space of simple linear functions for $m(\mathbf{x})$. The second summand covers non-linear relationships. The third summand models noise. If the variance parameter $r(\mathbf{x})$ is stable, $\boldsymbol{\eta}$ is Gaussian. However, if $r(\mathbf{x})$ is changing, $\boldsymbol{\eta}$ gives rise to heteroskedastic noise.

Previous research argued that heteroskedasticity is one possible scenario that directly implies non-Gaussianity in the noise (Hyvärinen et al., 2010). Therefore, heteroskedasticity renders the causal direction discoverable in the case of a linear functional relationship as underlying ground truth. This case is not restricted to independent noise (see, for example, the work on dependent noise through latent confounders (Hoyer et al., 2008b; Shimizu and Bollen, 2014)). The linear case can be covered by (4.17) through the linear mean function. In (4.17), the $\mathcal{GP}$ parameters $\mathbf{f}$ and $\mathbf{g}$ correspond to α in (4.7) (see Fig. 4.2). As Lázaro-Gredilla and Titsias (2011) showed, $\mathbf{f}$ and $\mathbf{g}$ can be learned with a variational method. Their variational approximation is an ideal choice for modelling $\mathbb{KL}(p(\boldsymbol{\alpha})||\mathcal{E}_{\boldsymbol{\alpha}}) = \mathbb{KL}(p(\mathbf{f}, \mathbf{g})||\mathcal{E}_{\mathbf{f},\mathbf{g}})$, as it approximates the intractable expectation (4.11) and provides parallel semantics to information geometric decision criteria based on $\mathbb{KL}$. The variational approach aims to find the best factorised variational densities to model $p(\mathbf{f}, \mathbf{g})$. The resulting $\mathbb{KL}$ is equivalent to our definition of latent exponential manifolds. (4.12) becomes:

$$\begin{aligned} &\mathbb{KL}(q(\mathbf{f}_\beta)q(\mathbf{g}_\beta)||p(\mathbf{f}_\beta, \mathbf{g}_\beta|\mathbf{x})) \\ &\leq \mathbb{KL}(q(\mathbf{f}_\alpha)q(\mathbf{g}_\alpha)||p(\mathbf{f}_\alpha, \mathbf{g}_\alpha|\mathbf{y})). \end{aligned} \tag{4.18}$$

We can give an intuitive interpretation of why this should indicate the causal direction. We calculate the $\mathbb{KL}$ divergence from the product of individual distributions of $\mathbf{f}$ and $\mathbf{g}$ to their joint posterior distribution. We therefore prefer the regression in which $\mathbf{f}$ and $\mathbf{g}$ are more independent as the plausible causal direction. As a result, we prefer the direction in which the noise is less heteroskedastic. This is in line with ANM related approaches: we prefer the direction in which the noise depends in a less complicated way on the data generating function.

We account for the linear parts of the causal mechanism as follows. Whereas a normal GP assigns a Gaussian prior to the linear coefficients b, we resort to ICA. We observe that (4.3) is a good estimator for the negative log-likelihood of the ICA model (Murphy, 2012):

$$G(\mathcal{D}^\top \mathbf{W}) \approx -log\, p(\mathbf{W}). \tag{4.19}$$

We can then minimise G to subsequently permute independent components to compute the linear coefficients of our simple mean functions (Shimizu et al., 2006). This results in an estimation for the coefficients of the linear functions relating $\mathbf{x}$ and $\mathbf{y}$ in both causal directions.

$$\begin{aligned} m(\mathbf{x})_1^\top b_1 &= b_1\mathbf{x} + c_1 \\ m(\mathbf{y})_2^\top b_2 &= b_2\mathbf{y} + c_2 \end{aligned} \tag{4.20}$$

For simplicity, we will drop the constants c_1 and c_2. Since (4.19) estimates the log-likelihood of our linear causal coefficient, we replace the Gaussian prior on the mean functions of the GP with this likelihood.

Including the linearity, as in the permutation of $\mathbf{W}$, the evidence described in (2.15) which we approximate with the second line of (4.18) becomes:

$$p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) = \iint p(\mathbf{y}|\mathbf{f},\mathbf{x})\, p(\mathbf{f}|\mathbf{W},\mathbf{x})\, p(\mathbf{W})\, \mathbf{dfdW}; \tag{4.21}$$

the evidence for the GP in the opposite direction becomes:

$$p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{y}) = \iint p(\mathbf{x}|\mathbf{f},\mathbf{y})\, p(\mathbf{f}|\mathbf{W},\mathbf{y})\, p(\mathbf{W})\, \mathbf{dfdW}. \tag{4.22}$$

The parameterisation of $\mathbf{f}$ in (4.21) differs from the parameterisation of $\mathbf{f}$ in (4.22) due to the choice of the best $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$. Both equations above are VHGPR models but instead of a zero-mean $\mathcal{GP}$, we deploy (4.20) as the mean function to model possible causal linearity in the data, therefore the dependence on $\mathbf{W}$. For readability we omit $\mathbf{g}$. By rearranging both sides of (4.18), the evidence (4.21,4.22) has a lower bound in each side for any choice of the variational probability densities $q(\mathbf{f})$ and $q(\mathbf{g})$ (Lázaro-Gredilla and Titsias, 2011).

We are now left with a number of optimisation problems. We need to find the optimal permutation of $\mathbf{W}$ which we need to combine with the VHGPR optimisation

for both hypotheses. In the following, we introduce an optimisation function that can be used for gradient based minimisation of all these optimisation problems in parallel.

$$\begin{aligned} F(\alpha, \beta, \mathbf{W}) = \quad & \mathbb{KL}(q(\mathbf{f}_\alpha)q(\mathbf{g}_\alpha)||p(\mathbf{f}_\alpha, \mathbf{g}_\alpha|\mathbf{y})) \\ & + \mathbb{KL}(q(\mathbf{f}_\beta)q(\mathbf{g}_\beta)\,||p(\mathbf{f}_\beta, \mathbf{g}_\beta\,|\mathbf{x})) \\ & + 2\,G(\mathcal{D}^\top \mathbf{W}). \end{aligned} \tag{4.23}$$

The model weights two components that allow causal discovery in the context of additive noise (Peters et al., 2011): (a) the non-linear functional relationships and (b) the evidence for linearity and for the distance from Gaussianity in the noise as it can result from heteroskedasticity.

We minimise (4.23) with respect to the hyper-parameters $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ parameterising (4.21) and (4.22), as well as the ICA-transformation matrix $\mathbf{W}$. We can do this in a gradient-descent based setting, since critical parameters do not depend on each other whilst we take the derivatives. Notes on covergence can be found in Appendix B.

$\boldsymbol{\alpha}$ and b_1 parameterise $\mathbf{f}$ and $\mathbf{g}$ for the VHGPR that is fitted in the direction $\mathbf{x} \to \mathbf{y}$, $\boldsymbol{\beta}$ and b_2 do the same for the opposite direction. Minimising the first summand of (4.23) is equivalent to maximising the lower bound of (4.21). It also guarantees an optimal and tractable approximation for $\mathbf{f}_\alpha, \mathbf{g}_\alpha$ in the second line of the decision criterion (4.18). Minimising the next summand is equivalent to maximising the lower bound of (4.22). It also provides an optimal and tractable approximation for $\mathbf{f}_\beta, \mathbf{g}_\beta$ in (4.18).

The last term approximates $p(\mathbf{W})$ in (4.21) and (4.22), weighting this information into the discovery process. This has been proven to be useful for causal discovery (Shimizu et al., 2006). As the first two lines of (4.23) indirectly depend on the last line, we perform the permutation steps during every iteration of the optimisation (algorithm 3, steps 6 to 9; analogue to Shimizu et al., 2006). We can further manipulate the first two summands to get a practical variational expression that allows to optimise efficiently. Details of the variational inference are omitted here, but are described in the work of Lázaro-Gredilla and Titsias (2011).

To determine the causal direction we check (4.18). The entire process of fitting the model and determining the causal direction is summarised in algorithm 3. Steps 5-8 are analogue to the LiNGAM discovery algorithm. Since we assume the two variable case, we can look up the linear coefficients of (4.20) in line 9 and 10.

The computation required for steps 5-8 and 13 is negligible for the two variable case. The complexity of an iteration amounts to twice the complexity of computing the VHGPR approximation resulting in a complexity of $\mathcal{O}(n^3)$, where n is the number of observations (see Lázaro-Gredilla and Titsias 2011).

Algorithm 3. Flexible Causal Discovery

1: **procedure** FLEXCD($\mathbf{X_{raw}}$)
2: $\mathbf{X} \leftarrow$ whitening($\mathbf{X_{raw}}$)
3: $\beta_\mathbf{0}, \alpha_\mathbf{0}, \mathbf{W}_0 \leftarrow$ initialise-$\beta, \alpha, \mathbf{W}_0$
4: **while** F not converged **do** ▷ Start, iterate over i
5: $\tilde{\mathbf{W}} \leftarrow \underset{\text{perm}(\mathbf{W}_i)}{\arg\min} \sum_i \frac{1}{|\mathbf{W}_i|}$
6: $\tilde{\mathbf{W}}^* \leftarrow$ divide rows of $\tilde{\mathbf{W}}$ by corresponding diagonal element
7: $\hat{\mathbf{B}} \leftarrow \mathbf{I} - \tilde{\mathbf{W}}^*$
8: $\mathbf{B} \leftarrow \underset{\text{perm}(\hat{\mathbf{B}})}{\arg\min} \sum_{i \leq j} \hat{\mathbf{B}}_{ij}^2$
9: $b_1^i \leftarrow \mathbf{B}_{2,1}$ ▷ Assign linearity parameter eq. (4.20)
10: $b_2^i \leftarrow \mathbf{B}_{1,2}$
11: compute $\nabla F(\boldsymbol{\alpha}_i)$ ▷ partial derivatives (4.23),line 1
12: compute $\nabla F(\boldsymbol{\beta}_i)$ ▷ partial derivatives (4.23), line 2
13: compute $\nabla F(\mathbf{W}_i)$ ▷ (4.23), line 3
14: $\boldsymbol{\alpha}_{i+1}, \boldsymbol{\beta}_{i+1}, \mathbf{W}_{i+1} \leftarrow$gradient-descent- step($\nabla F(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \mathbf{W}_i)$)
15: check joint-convergence
16: **end while**
17: **if** eq. (4.18) holds **then**
18: assign $\mathbf{x} \rightarrow \mathbf{y}$
19: **else**
20: assign $\mathbf{y} \rightarrow \mathbf{x}$
21: **end if**
22: **end procedure**

end

4.3.2 Practical Considerations

For computing G we mean-centre the data to make the estimation of $\mathbf{W}$ simpler and better conditioned. We also whiten the data with singular value decomposition (see Hyvärinen and Oja, 2000, section 5.2). Since the components are used as the coefficients of a global linear model underlying the GP, we assume that the global model of the data generating process follows an ICA data model, and that the expectations are evaluated correctly (in line with previous research on fixed-point ICA (Hyvärinen, 1999)). To compute G with a gradient-based approach we have to compute G in two steps, one

for each row. We impose a Lagrangian constraint that each row is of unit length and subsequently de-correlate the rows with each other. With a normalisation step during each iteration we project each row back onto the constraint surface.

We use the same kernels k_f and k_g within VHGPR for both $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x})$ and $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{y})$. For k_f we choose a sum of a squared exponential covariance function with isotropic distance measure, plus a covariance function for a constant function. For k_g we choose a sum of a squared exponential covariance function with isotropic distance measure plus independent noise (we will not delve into the details of covariance functions here, for a good explanation, see Rasmussen and Williams, 2006). Although we choose the same covariance function for both $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x})$ and $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{y})$, we learn two sets of parameters each, that is we parametrise k_f and k_g twice. To initialise the parameters, we run a homoskedastic GP. For simplicity, we use a gradient descent algorithm to optimise $F(\alpha, \beta, \mathbf{W})$. The convergence properties can be derived from convexity of the GP and computation of ICA plus practical considerations presented above. A typical run takes ca. 5 seconds to optimise on an i7CPU.

4.4 Evaluation of FlexCD

4.4.1 Artificial Data

For the evaluation with synthetic data, we considered two different functional relations:

$$\begin{aligned} f_1(x_i) &= \sin(a_1 x_i) + \sqrt{x_i} + e \\ f_2(x_i) &= v_i \sin\left((a_1 x_i) + \sqrt{x_i}\right)) + (1 - v_i) a_2 x_i + c_i + e \end{aligned} \tag{4.24}$$

where a_1 and a_2 are parameters, c_i is a constant and v_i is an entry in the vector $\mathbf{v}$. $\mathbf{v}$ is a boolean vector that enables us to combine the two functions, creating "regions" of ones and zeros $\mathbf{v} = \{1, 1, 1, 1, 0, 0, 0, 0, 1, ...\}$. c_i is an entry in the vector $\mathbf{c}$ that adds constants in order to provide a seemingly smooth functional relationship (illustrated in Fig. 4.3). This models a change in the causal mechanism. For generating test data sets, we made the following choices: a_1 and a_2 were varied for each data set. a_1 is drawn from a uniform distribution between 0.5 and 1.5 and a_2 from a Gaussian one. $\mathbf{x}$ is also uniformly distributed between 0 and 30. To the functional relationships we added either homoskedastic or heteroskedastic noise. For the homoskedastic case the noise is Gaussian. For the heteroskedastic case, we simulated noise with a Gaussian distribution where the variance parameter is changing over the domain of the cause. This results in a total of four experimental conditions: two functions $\times$ two different kinds of noise. For each condition we created 50 data sets with 500 data points each resulting in a total of 200 synthetic data sets.

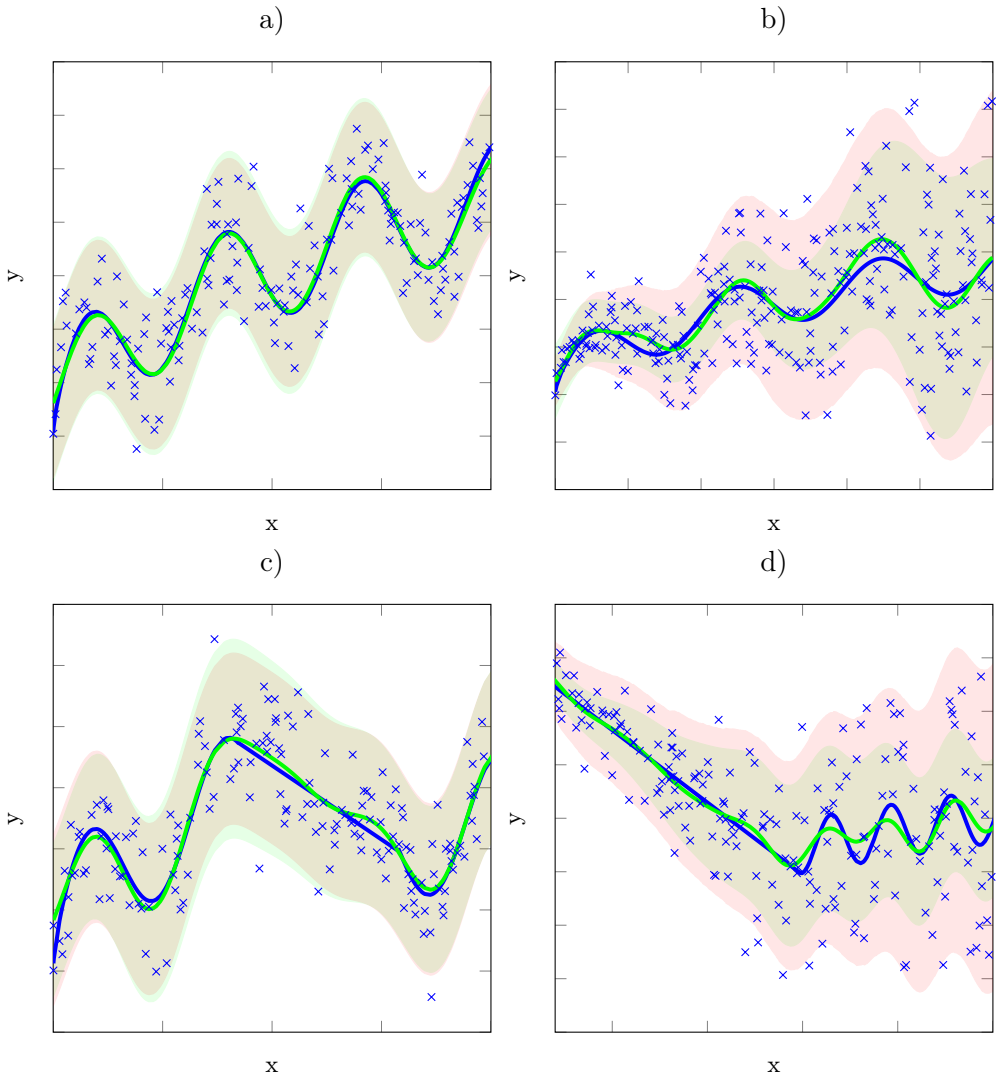


Figure 4.3: Data generated with function f_1 (a) and b)) and f_2 (c) and d)) from (4.24). Homoskedastic noise is added in a) and c). Heteroskedastic noise is added in b) and d). Data and actual functions are depicted in blue, FlexCD's best guess in bright green. The actual noise is light red, FlexCD's estimation of the noise is light green, overlap of both is green-grey. FlexCD inferred the correct direction of cause and effect in all examples that we chose to illustrate. FlexCD estimates the data generating function and the noise well, depicted through the agreement of actual and estimated function and noise.

Table 4.1: Accuracy (in percent) of finding the true causal direction in 200 different data sets. f_1 and f_2 are described in (4.24). Either homoskedastic and heteroskedastic noise is added.

	FlexCD	IGCI	LiNGAM	ANM	GPI
f_1, hom.	**100**	<48*	<48*	100	<48*
f_1, het.	**94**	<48*	48	<48*	100
f_2, hom.	**94**	76	78	100	96
f_2, het.	**84**	64	58	82	100
Overall	**93**	<48*	48	70	84

We compare our results with a benchmark consisting of Information Geometric Causal Inference (IGCI) (Janzing et al., 2012), the linear non-Gaussian acyclic model (LiNGAM) (Shimizu et al., 2006), the additive noise model (ANM) (Hoyer et al., 2008a) and the probabilistic latent variable model for distinguishing between cause and effect (GPI) (Mooij et al., 2010). The background of the benchmark methods has been discussed in the section about functional causal discovery. FlexCD obtains the best results in the entire benchmark (see tab. 4.1). We have marked results that are significantly lower than chance with an asterisk. Such results arise because of the way the data is generated (4.24). This happens for example with ANM in the case of a non-linear function with heteroskedastic noise. The ANM depends on the assumption that where $\mathbf{x} \to \mathbf{y}$, $\mathbf{y}$ can be computed as a function of $\mathbf{x}$ plus a noise term independent of $\mathbf{x}$, whereas $\mathbf{x}$ cannot be computed as a function of $\mathbf{y}$ plus independent noise. We generated the data in a way that the noise increases with $\mathbf{x}$. That means we create a dependency that ANM takes as evidence for an inverse causal direction, introducing a strong bias to the result. Something similar happens to LiNGAM in the case of a non-linear function with homoskedastic noise. We use a sine-like data generating mechanism here. If we flip cause and effect, we could fit a line between the two, providing a reasonable guess (averaged over all data points). Considering everything which differs from $\sin x = 0$ as noise, the sine oscillations will provide "noise" that is distant from Gaussian. LiNGAM interprets this as a linear relationship with non-Gaussian noise, again introducing a strong bias towards the inverse direction of cause and effect.

Tab. 4.2 summarises the average runtime for one of the above data sets. Note that for these experiments, the parameters for our code as well as for the benchmark are

Table 4.2: Average runtime in seconds.

FlexCD	IGCI	LiNGAM	ANM	GPI
250.88	0.001	0.11	7.62	231.10

tuned so that the runtime is longer than typical for using either algorithm out of the box. We see that the two algorithms that perform best, FlexCD and GPI, are significantly slower in this experimental configuration than the others. We also see that FlexCD is only minimally slower than GPI but exhibits a significant performance gain. Note that all algorithms come with a number of free parameters, thus, a complete comparison of performance, runtime and parameter configuration is not trivial. In general, we have tuned the parameters for all algorithms in a way that the performance in terms of causal discovery is optimal.

Matlab code and re-producible experiments can be found in `www.schaechtle.com/Software.html`.

4.4.2 Real World Data

We tested our algorithm on 81 real world data sets[2]. These data were selected out of 88 cause-effect pairs, where we excluded data sets with binary variables and data sets that were comprised of more than two variables. To increase computational speed we subsampled 500 datapoints. Additionally, we ran the experiments again subsampling 100 datapoints to show that FlexCD is more robust than previous alogrithms with less observations available. The ground truth was provided by the database owners. FlexCD outperformed all other algorithms (see tab. 4.3). We believe this result is partly due to heteroskedasticity being ubiquitous in empirical data.

We will elaborate on this last point about heteroskedasticity by discussing two cause-effect pairs where FlexCD found the correct causal direction. In Fig. 4.4 a) the causal ground truth is that the altitude of where a temperature measurement is taken has a causal influence on the temperature measured. This is an inherently heteroskedastic problem. We know that maritime regions with near-zero altitude have milder seasonal differences, then for the next 1000 meters, noise will increase with distance from the sea. Noise will decrease in high mountain areas where it is very cold. Therefore, we would expect noise to be lower the higher the altitude gets.

In Fig. 4.4 b) the causal ground truth is that the age of an abalone snail (as indicated by the number of rings in the shell) has causal effect on its length (as in the longest shell measurement). Noise is increasing with the age of the snail. This makes sense, since the additive noise term summarises all latent confounders affecting the measurements. Without any further knowledge, we can assume that the older the snail, the higher the chance that something affects the growth of the snail over the course of time and confounds our measurement.

[2] `https://webdav.tuebingen.mpg.de/cause-effect/`

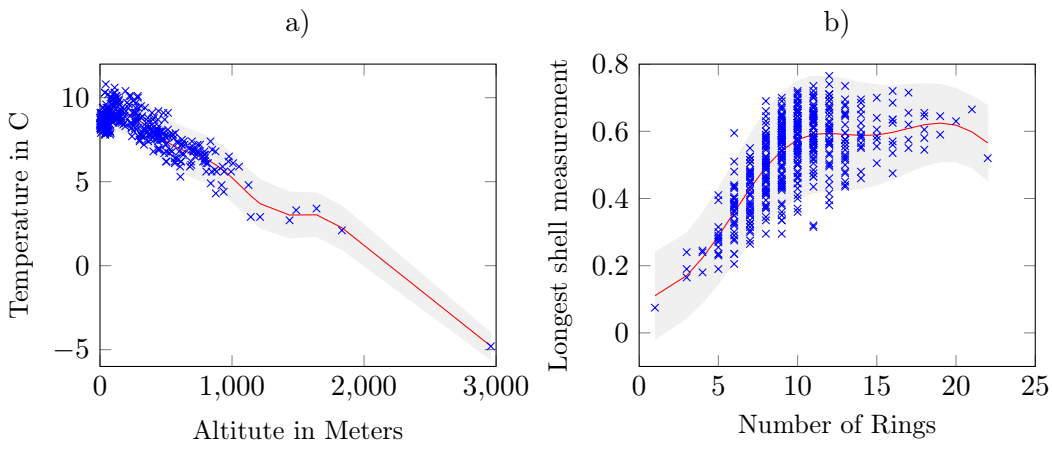


Figure 4.4: Heteroskedasticity within the cause-effect pair database. a) The causal ground truth is that the altitude of a temperature measurement (x-axis) has a causal influence on the temperature measured (y-axis). b) depicts the causal effect the age of an abalone snail (as indicated by the number of rings in the shell, x-axis) has on its longest shell measurement (y-axis). In both graphs, we plot FlexCD's prediction in form of a red line and it's guess for the estimated noise in grey.

For both examples, FlexCD models the heteroskedastic nature of the noise adequately.

Table 4.3: Accuracy (in percent) and standard deviation of finding the true causal direction in 81 real world data sets where n is the number of samples used per data set and w indicates whether or not (✓,✗) the weighting suggested by the database owners was taken into account.

n	w.	FlexCD	IGCI	LiNGAM	ANM	GPI
100	✗	**73 ± 5**	60 ± 3	66 ± 1	53 ± 3	60 ± 4
100	✓	**71 ± 5**	66 ± 4	63 ± 1	57 ± 4	66 ± 4
500	✗	**75 ± 2**	66 ± 3	65 ± 1	61 ± 1	68 ± 2
500	✓	**74 ± 2**	72 ± 3	61 ± 1	67 ± 2	74 ± 2

4.5 Summary

FlexCD is a novel causal discovery method that allows noise to be heteroskedastic, reflecting different variability of noise in the data. The model reflects changes in the causal mechanism given the changes in the cause. The implementation of the model provides a competitive predictive capability, due to a state-of-the-art regression model in use. FlexCD offers significant improvements on inferring causal direction in comparison with other approaches for functional causal discovery. The method does especially well

in scenarios where the prior assumptions of previous approaches such as LiNGAM and ANM cause those methods to fail. Its implementation is efficient due to the variational algorithm deployed.

As opposed to the approaches of the previous chapter, we made the dynamics explicit. We have encoded assumptions about dynamically changing noise into the functional equations. To do this, we have assumed a latent heteroskedastic noise component that is conditioned on the data.

How could we weaken the assumptions of this approach? Is there a way to let the data speak for themselves with regards to the existence of latent structures?

5 | Abstract Dynamical Causality & Probabilistic Programming

In the previous chapter, we have made the assumption of a heteroskedastic noise component and exploited it for causal discovery. We did not discover this component in a data-driven way.

The data-driven discovery of latent components is one of the main tasks of machine learning (see e.g. Bishop, 2006, for a latent variable perspective on various kinds of machine learning). Ideally, we would like to include the discovery of latent components, such as heteroskedastic noise, into the causal discovery process instead of restricting ourselves to relations between observable entities. In fact, this is the main focus for many applications of causal reasoning. For example, we define medical diagnosis as "The act or process of identifying or determining the nature and cause of a disease or injury through evaluation of patient history, examination, and review of laboratory data." (The Free Dictionary: Diagnosis, 2015). The "nature or cause" is often not observable. All we can observe are the symptoms over the course of time.

In diagnosis, we find symbolic representations for such latent causes that form our hypothesis space such as as "Disease X" or "Condition Y". Symbolic description of continuous-valued data domains is hard as it entails intelligent abstraction from a set of coordinates in $\mathbb{R}^n$ to infer the structure that causes observations to take a certain shape. The abstraction itself relies on sound empirical inference which, as we showed in previous chapters (2 and 4), is best done with GPs. Symbolic reasoning on the other hand is native to probabilistic programming. A generative causal probabilistic programming framework that deals with GP is therefore desirable.

In this chapter, we introduce a probabilistic programming technique that provides the semantics necessary for such reasoning. We will present GP memoization (`gpmem`) as a linguistic representation of a Stochastic Process (SP) for GPs. We will implement this into a framework of probabilistic programming. We then aim to demonstrate the expressiveness of the technique and show how it can solve hard problems. Examples

for such hard problems are regression in the presence of outliers, learning of symbolic expressions for continuous-valued time series, data-driven diagnosis with clinical decision making and Bayesian Optimization. `gpmem` introduces a statistical alternative to standard memoization which refers to the storing of previously computed values for functions. Our contribution is threefold:

- we introduce an efficient implementation of `gpmem` in form of a self-caching wrapper that remembers previously computed values;
- we illustrate the statistical emulator that `gpmem` produces and how it improves with every data-point that becomes available; and
- we show how one can solve hard problems of state-of-the-art machine learning related to GP using `gpmem` in a Bayesian fashion with only a few lines of Venture code.

`gpmem` is an extension of previous approaches to stochastic memoization. Previous approaches such as the one introduced for Dirichlet Processes in the Church language (Goodman et al., 2008) deal with the discrete-valued domain. We introduce memoization for the continuous-valued domain. This is important for problems of regression and optimization where target functions should not be artificially discretized.

We present results that were previously not available. `gpmem` allows us to implement causally structured hyper-priors, which has never been done in probabilistic programming. Furthermore, we use it to discover latent causal structure for kernel composition. This was not available before in a Bayesian fashion. Finally, we use `gpmem` for Bayesian optimization with Thompson sampling which is also novel.

The chapter is structured as follows, we will first provide the theory on memoization. We will explain programming in Venture and GPs in Venture (section 5.1). We introduce `gpmem` and its use in probabilistic programming and Bayesian modelling (section 5.2). Finally, we will show how we can apply `gpmem` on problems of causally structured hierarchical priors for hyper-parameter inference (section 5.3), structure discovery for Gaussian Processes (section 5.4) and Bayesian Optimization (section 5.5) including experiments with real world and synthetic data. Code to reproduce all the experiments is available[1].

[1] currently, one can download the code under `https://github.com/Schaechtle/matrixGP`. However, this will soon be migrated to `http://probcomp.csail.mit.edu/venture/`.

5.1 Gaussian Processes in Probabilistic Programming

Probabilistic programming could be revolutionary for machine intelligence due to universal inference engines and the rapid prototyping for novel models (Ghahramani, 2015). Venture, for example, supports inference in terms of empirical gradient-ascent, MCMC and variational approximation. This accelerates the design and testing of new models as well as the incorporation of complex prior knowledge which currently is a difficult and time consuming task. Probabilistic programming languages aim to provide a formal language to specify probabilistic models in the style of computer programming and can represent any computable probability distribution as a program. In this work, we will introduce new features of Venture, a recently developed probabilistic programming language. We consider Venture the most compelling of the probabilistic programming languages because it is the first probabilistic programming language suitable for general purpose use (Mansinghka et al., 2014). Venture comes with scalable performance on hard problems and with a general purpose inference engine. The inference engine deploys Markov Chain Monte Carlo (MCMC) methods (for an introduction, see Andrieu et al., 2003). MCMC lends itself to models with complex structures such as probabilistic programs or hierarchical Bayesian non-parametric models since they can provide a vehicle to express otherwise intractable integrals necessary for a fully Bayesian representation. MCMC is scalable, often distributable and also compositional. That is, one can arbitrarily chain MCMC kernels to infer over several hierarchically connected or nested models as they will emerge in probabilistic programming.

One powerful model rarely treated in probabilistic programming languages are GPs. In addition to our use of GPs for flexible causal discovery (chapter 4), GPs are gaining increasing attention for representing unknown functions by posterior probability distributions in various fields such as machine learning (Rasmussen and Williams, 2006), signal processing (Clifton et al., 2013), computer vision (Kemmler et al., 2013) and biomedical data analysis (Shepherd and Owenius, 2012). Making GPs available in probabilistic programming is crucial to allow a language to solve a wide range of problems. Hard problems include but are not limited to hierarchical prior construction (Neal, 1997), Bayesian Optimization (Snoek et al., 2012) and systems for inductive learning of symbolic expressions such as the one introduced in the Automated Statistician project (Duvenaud et al., 2013; Lloyd et al., 2014). Learning such symbolic expressions is a hard problem that requires careful design of approximation techniques since standard inference method do not apply.

5.1.1 Memoization

Memoization is the practice of storing previously computed values of a function so that future calls with the same inputs can be evaluated by lookup rather than recomputation. Research on the Church language (Goodman et al., 2008) pointed out that although memoization does not change the semantics of a determinstic program, it does change that of a stochastic program. The authors provide an intuitive example: let f be a function that flips a coin and return "head" or "tail". The probability that two calls of f are equivalent is 0.5. However, if the function call is memoized, it is 1.

In fact, there is an infinite range of possible caching policies (specifications of when to use a stored value and when to recompute), each potentially having a different semantics. Any particular caching policy can be understood by random world semantics (Poole, 1993; Sato, 1995) over the stochastic program: each possible world corresponds to a mapping from function input sequence to function output sequence (McAllester et al., 2008). In Venture, these possible worlds are first-class objects, known as *traces* (Mansinghka et al., 2014).

5.1.2 The Venture Language

Venture is a probabilistic programming language that allows both the design of custom inference algorithms and arbitraly nested and hierarchical models. It comes with two different front end syntaxes, one Scheme-like and one JavaScript-like.

Venture provides a powerful and concise way to represent probability distributions, including distributions with a dynamically determined and unbounded set of random variables. In this chapter we will use only the four basic Venture directives: `assume`, `observe`, `sample` and `infer` (Mansinghka et al., 2014).

- `assume` induces a hypothesis space for (probabilistic) models including random variables by binding the result of a supplied expression to a supplied symbol.

- `sample` samples the value of the supplied expression within the current model program. Such an expression is evaluated within a trace of the model program. Thus, the value of an expression within a model program is a random variable, whose randomness comes from the distribution on possible execution traces of the program.

- `observe` constrains the supplied expression to have the supplied value. In other words, all samples taken after an `observe` are conditioned on the observed data.

- `infer` uses the supplied inference program changes the modeled probability distribution. This will result approximate sampling from the true posterior, conditioned

on the model and constraints introduced by `assume` and `observe`. The posterior on any random variable can then be approximately sampled by calling `sample` to extract values from the modeled distribution.

`infer` is commonly done using the Metropolis–Hastings algorithm (MH) (Metropolis et al., 1953). Many of the most popular MCMC algorithms can be interpreted as special cases of MH (Andrieu et al., 2003). We can outline the MH algorithm as follows. The following two-step process is repeated as long as desired (say, for T iterations[2]): First we sample x^* from a proposal distribution q:

$$x^* \sim q(x^* \mid x^t); \tag{5.1}$$

then we accept this proposal ($x^{t+1} \leftarrow x^*$) with probability

$$\alpha = \min\left\{1, \frac{p(x^*)q(x^t \mid x^*)}{p(x^t)q(x^* \mid x^t)}\right\}; \tag{5.2}$$

if the proposal is not accepted then we take $x^{t+1} \leftarrow x^t$.

Venture includes a built-in generic MH inference program which performs the above steps on any specified set of random variables in the model program. In that inference program, partial execution traces play the role of x above.

5.1.3 Venture GPs

We will first introduce common covariance functions (a.k.a. kernels) as they will become important for our models below:

$$\mathrm{SE} = \sigma^2 \exp(-\frac{(x-x')^2}{2\ell^2}) \tag{5.3}$$

$$\mathrm{LIN} = \sigma^2(xx') \tag{5.4}$$

$$\mathrm{C} = \sigma^2 \tag{5.5}$$

$$\mathrm{WN} = \sigma^2\delta_{x,x'} \tag{5.6}$$

$$\mathrm{RQ} = \sigma^2\left(1 + \frac{(x-x')^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{5.7}$$

$$\mathrm{PER} = \sigma^2 \exp\left(\frac{2\sin^2(\pi(x-x')/p}{\ell^2}\right). \tag{5.8}$$

SE stands for squared-exponential, LIN for linear, C for constant, WN for white noise, RQ for rational quadratic and PER for periodic. Custom covariance functions can be implemented in Python outside of Venture. All of the above will be shipped with

[2]choosing the right T is difficult. Convergence results are only known for simplest cases. Often, the choice of T is a matter of trial and error

Venture in future releases[3].

We demonstrate a simple smoothing GP in Venture. The Venture procedure `make-gp` takes as input a mean function and a covariance function, and outputs a procedure for sampling from a Gaussian process. In effect, each call to this procedure samples from (2.16) conditioned on the return values of all previous samples. `make-gp` allows us to perform GP inference in Venture with only a few lines of code. We can concisely express a wide variety of GPs: simple smoothing with fixed hyper-parameters, or a prior on hyper-parameters, or a custom covariance function. Inference on hyper-parameters can be performed using Venture's built-in MH operator or a custom inference strategy.

Venture code to create and sample from a GP with a smoothing kernel and hyper-parameters is shown in Listing 5.1.

Listing 5.1: Venture GP

```
// HYPER-PARAMETERS
1 assume sf = tag(quote(hyper), 0, gamma(1,1)))
2 assume l = tag(quote(hyper), 1, gamma(1,1)))

// COVARIANCE FUNCTION
3 assume se = make_squaredexp(sf, l)

// MAKE GAUSSIAN PROCESS
4 assume gp = make_gp( 0, se)

// INCORPORATE OBSERVATIONS
5 observe gp(array x[1],...x[n])= array(y[1],...,y[n])

// INFER HYPER-PARAMETERS
6 infer mh(quote(hyper), one, 1)))
```

The first two lines declare the hyper-parameters. We tag both of them to belong to the "scope" `quote(hyper)`. Scopes denote a collection of related random choices. Scopes may be further subdivided into blocks, on which block proposals can be made. In the program above we assign two blocks, 0 and 1. These tags are supplied to the inference program (in this case, MH) to specify on which random variables inference should be done. In this chapter, we use MH inference throughout and do not use block proposals; MH inference is done on one variable at a time.

[3]The covariance functions will be available as type Venture-Function.

The `assume` directives describe the GP model: `sf` and `l` (corresponding to σ and ℓ in the squared exponential covariance function (5.3)) are drawn from independent $\Gamma(1, 3)$ distributions (line 1 and 2). The squared exponential covariance function can be defined outside the Venture code in a conventional programming language (e.g. Python) and imported as a foreign SP. In that way, the user can define custom covariance functions using his or her language and libraries of choice, without having to port existing code into Venture's modelling language. In the above, the factory function `make-squaredexp`, which produces a squared exponential function with the supplied hyper-parameters, is imported from Python (line 3, we have omitted the Python code).

Finally, we declare `gp` to be a GP with mean zero and covariance function `se` (line 4). We observe data (line 5) and take an MH step (line 6).

A Bayesian Interpretation

We illustrate and compare Bayesian and frequentist view points on GP with a simple example (Fig. 5.1). We show how in a simple model, two outliers can bias a maximum a posteriori inference. The data in Fig. 5.1 are generated with:

$$y = 2x + 15 \tag{5.9}$$

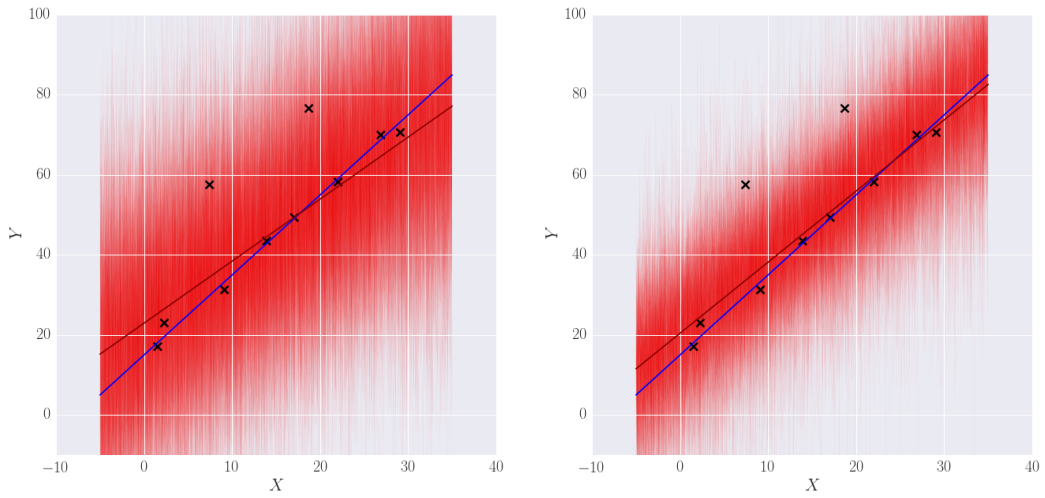
and outliers are generated with a parallel line:

$$\hat{y} = 2x + 40. \tag{5.10}$$

We add some small amount of white noise. To illustrate the flexibility of the universal inference engine we demonstrate the difference between Maximum A Posteriori (MAP) and MH. We generate eight data points with (5.9) and two with (5.10). Since we suspect the underlying data generating mechanism to be linear, we fit a linear kernel (5.4) with a constant covariance (5.5) as intercept and some white noise (5.6):

$$\mathbf{K} = \mathrm{LIN} + \mathrm{C} + \mathrm{WN}. \tag{5.11}$$

where we upper case matrix notation denotes the covariance matrix of the complete training data. Omitting the scaling parameter for the linear kernel, there are two hyper-parameters to learn, that is the noise variance and the hyper-parameter for the constant function. MAP inference fits the single one best line and accounts for the outliers with a large noise scaling parameter. MH preserves the uncertainty. It assigns a small amount of probability mass to a different scaling parameter and a larger constant. The resulting prediction (indicated by the predictive mean in figure 5.1) is closer to the true underlying function. We conclude that in example a Bayesian treatment is better than a frequentist treatment.



(a) MAP (b) MH

Figure 5.1: (a) depicts MAP inference on the data, (b) depicts MH for hyper-parameter inference. The blue line is the actual data generating function. Red are samples drawn from the posterior. The dark red line is the posterior predictive mean. We see that the MH shifts the posterior closer to the ground truth than MAP.

5.2 Gaussian Process Memoization: `gpmem`

We implement `gpmem` by memoizing a target procedure in a wrapper that remembers previously computed values. The technique gives rise to a number of use-cases of GPs in a fully Bayesian setting which are otherwise hard to implement.

Algorithm 4. The restricted function

```
1: function f_RESTR(x)
2:     if x ∈ D then
3:         return D[x]
4:     else
5:         raise Exception('Illegal input')
6:     end if
7: end function
end
```

Data modelling as a special case of `gpmem`

From the standpoint of computation, a data set of the form $\{(x_i, y_i)\}$ can be thought of as a function $y = f_{\text{restr}}(x)$, where f_{restr} is restricted to only allow evaluation at a

specific set of inputs x (Alg. 4).

Modelling the data set with a GP then amounts to trying to learn a smooth function f_{emu} ("emu" stands for "emulator") which extends f to its full domain. Indeed, if f_{restr} is a foreign procedure made available as a black-box to Venture, whose secret underlying pseudo code is in Alg. 4, then the `observe` code can be rewritten using `gpmem` as follows (Listing 5.2, where the data set `D` has keys `x[1]`,...,`x[n]`):

Listing 5.2: Observation with `gpmem`

```
1 assume_list (f_compute f_emu) =  gpmem( f_restr)
2 for i=1 to n:
3   predict f_compute( x[i])
4   infer mh(quote(hyper-parameters), one, 100)
5 sample (f_emu( array( 1, 2, 3))
```

Here, `f_compute` "probes" all data points calling f_{restr}. This rewriting has at least two benefits: (i) readability (in some cases), and (ii) amenability to active learning. As to (i), the statistical code of creating a Gaussian process is replaced with a memoization-like idiom, which will be more familiar to programmers. As to (ii), when using `gpmem`, it is quite easy to decide incrementally which data point to sample next: for example, the loop from `x[1]` to `x[n]` could be replaced by a loop in which the next index `i` is chosen by a supplied decision rule. In this way, we could use `gpmem` to perform online learning using only a subset of the available data.

5.3 Causal Structure for Hyper-Priors

The probability of the hyper-parameters of a GP as defined above and given covariance function structure $\mathbf{K}$ is:

$$P(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{K}) = \frac{P(\mathbf{D} \mid \boldsymbol{\theta}, \mathbf{K}) P(\boldsymbol{\theta} \mid \mathbf{K})}{P(\mathbf{D} \mid \mathbf{K})}. \tag{5.12}$$

Let the $\mathbf{K}$ be the sum of a smoothing and a white noise (WN) kernel. For this case, Neal (1997) suggested the problem of outliers in data as a use-case for a hierarchical Bayesian treatment of Gaussian processes[4]. The work suggests a hierarchical system of hyper-parameterization. Here, we draw hyper-parameters from Γ distributions:

$$\ell^{(t)} \sim \Gamma(\alpha_1, \beta_1),\ \sigma^{(t)} \sim \Gamma(\alpha_2, \beta_2) \tag{5.13}$$

[4]In (Neal, 1997) the sum of an SE plus a constant kernel is used. We keep the WN kernel for illustrative purposes.

and in turn sample the α and β from Γ distributions as well:

$$\alpha_1^{(t)} \sim \Gamma(\alpha_\alpha^1, \beta_\alpha^1),\ \alpha_2^{(t)} \sim \Gamma(\alpha_\alpha^2, \beta_\alpha^2), \cdots \tag{5.14}$$

Assuming the covariance structure is additive comprised of a smoothing and a white noise kernel, one can represent this kind of model using `gpmem` with only a few lines of code (Listing 5.3).

Listing 5.3: Causal Hyper-Priors

```
/// SETTING UP THE MODEL
assume alpha_sf = tag(quote(hyperhyper), 0, gamma(7, 1))
assume beta_sf = tag(quote(hyperhyper), 1, gamma(7, 1))
assume alpha_l = tag(quote(hyperhyper), 2, gamma(7, 1))
assume beta_l = tag(quote(hyperhyper), 3, gamma(7, 1))

// Parameters of the covariance function
assume sf = tag(quote(hyper), 0, gamma(alpha_sf, beta_sf)))
assume l = tag(quote(hyper), 1, gamma(alpha_l, beta_l)))
assume sigma = tag(quote(hyper), 2, uniform_continuous(0, 2))

// The covariance function
assume se = make_squaredexp(sf, l)
assume wn = make_whitenoise(sigma)
assume composite_covariance = add_funcs(se, wn)

/// PERFORMING INFERENCE
// Create a prober and emulator using gpmem
assume f_restr = get_neal_blackbox()
assume (f_compute, f_emu) = gpmem(f_restr, composite_covariance)

// Probe all data points
for n ... N
    predict f_compute(get_neal_data_xs(n))

// Infer hypers and hyperhypers
infer repeat(100, do(
    mh(quote(hyperhyper), one, 2),
    mh(quote(hyper), one, 1)))
```

The hyper-prior structure is causal. The values of α and β have an effect on the distribution of the hyper-parameters for the GP. We can express the causal relationship

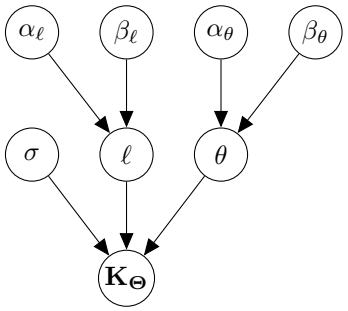


Figure 5.2: A causal graphical model depicting the hyper-parameter structure described with equations (5.13) and (5.14) and in lines 1:4 of Listing 5.3. $\mathbf{K}_{\mathbf{\Theta}}$ is the composite covariance matrix where $\mathbf{\Theta}$ is a certain parametrisation of the covariance function. ℓ and θ are the hyper-parameters of the SE kernel. σ parametrises the WN kernel.

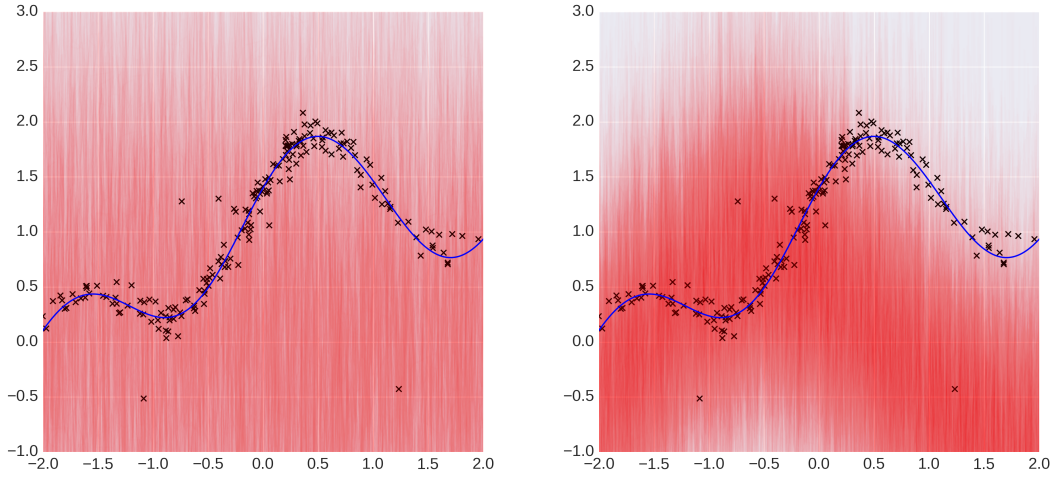
as a causal DAG (Fig. 5.2).

Neal provides a custom inference algorithm setting and evaluates it using the following synthetic data problem (Neal, 1997). Let f be the underlying function that generates the data:

$$f(x) = 0.3 + 0.4x + 0.5\sin(2.7x) + \frac{1.1}{(1+x^2)} + \eta \quad with \;\; \eta \sim \mathcal{N}(0, \sigma) \tag{5.15}$$

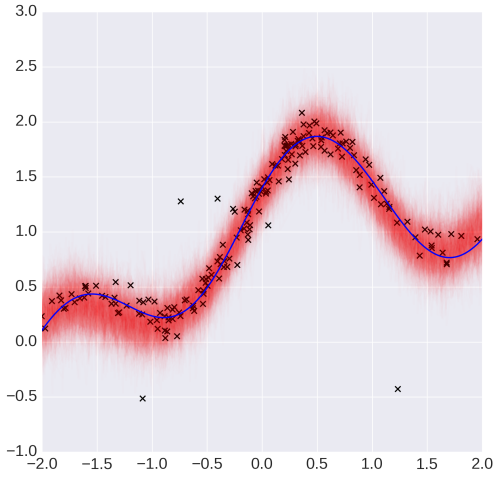
We synthetically generate outliers by setting $\sigma = 0.1$ in 95% of the cases and to $\sigma = 1$ in the remaining cases. `gpmem` can capture the true underlying function within only 100 MH steps on the hyper-parameters to get a good approximation for their posterior (see Fig. 5.3). Note that Neal devises an additional noise model and performs a large number of Hybrid-Monte Carlo and Gibbs steps. We illustrate the hyper-parameter by showing the shift of the distribution on the noise parameter σ (Fig. 5.4). We see that `gpmem` learns the posterior distribution well, the posterior even exhibits a bimodal histogram when sampling σ 100 times reflecting the two modes of data generation, that is normal noise and outliers[5].

[5]For this pedagogical example we have increased the probability for outliers in the data generation slightly from 0.05 to 0.2, to make the illustration more clear



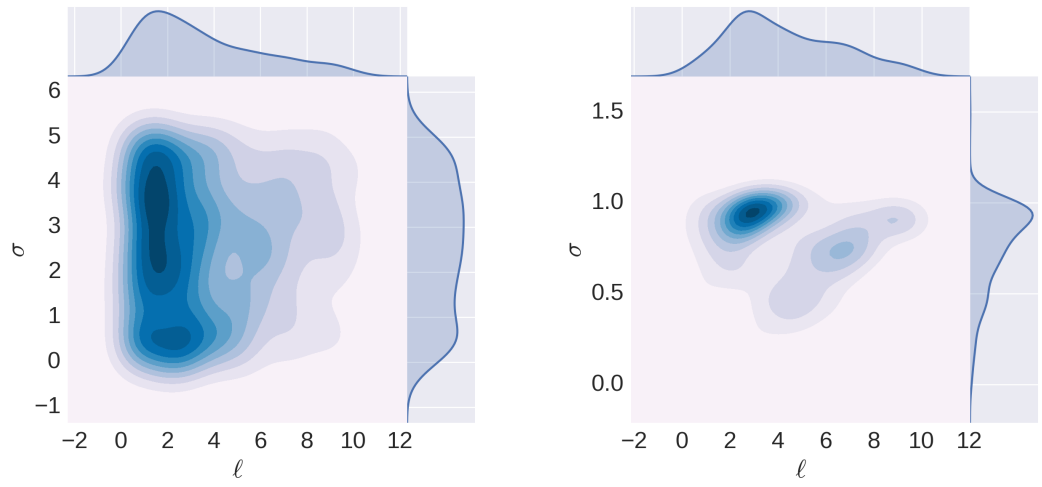
(a) Prior Inference

(b) Observed



(c) Inferred

Figure 5.3: (a)-(c) shows `gpmem` on Neal's example. We see that prior renders functions all over the place (a). After `gpmem` observes some data-points, an arbitrary smooth trend with a high level of noise is sampled. After running inference on the hierarchical system of hyper-parameters we see that the posterior reflects the actual curve well. Outliers are treated as such and do not confound the GP.



(a) Before

(b) After

Figure 5.4: Hyper-parameter inference on the parameter of the noise kernel. We show a 100 samples drawn from the distribution on σ. One can clearly recognise the shift from the uniform prior $\mathcal{U}(0, 5)$ to a double peak distribution around the two modes - normal and outlier.

Broader applicability of `gpmem`

Broader applicability of `gpmem` arises in two ways - either in cases where one wants to design custom inference algorithms (e.g. MAP vs. MH) or with regards to the source of the data.

More generally, `gpmem` is relevant not just when a data set is available, but also whenever we have at hand a function f_{restr} which is expensive or impractical to evaluate many times. `gpmem` allows us to model f_{restr} with a GP-based emulator f_{emu}, and also to use f_{emu} during the learning process to choose, in an online manner, an effective set of probe points $\{x_i\}$ on which to use our few evaluations of f_{restr}. This idea is illustrated in detail in Section 5.5. Before doing this, we will illustrate another benefit of having a probabilistic programming apparatus for GP modelling: the linguistically unified treatment of inference over structure and inference over parameters. This unification makes interleaved joint inference over structure and parameters very natural, and allows us to give a short, elegant description of what it means to "learn the covariance function," both in prose (using symbolic logic) and in code. Furthermore, the example in Section 5.4 below recovers the performance of current state-of-the-art GP-based models.

5.4 Discovery of Latent Causal Structure

The space of possible kernel compositions is infinite. Combining inference over this space with the problem of finding a good parameterization that could potentially explain the observed data best poses a hard problem. The natural language interpretation of the meaning of a kernel and its composition renders this a problem of symbolic computation. Duvenaud and colleagues note that a sum of kernels can be interpreted as logical OR operations and kernel multiplication as logical AND (2013). This is due to the kernel rendering two points similar if k_1 OR k_2 outputs a high value in the case of a sum. Respectively, multiplication of two kernels results in high values only if k_1 AND k_2 have high values (see Fig. 5.5 exemplifies how to interpret global vs. local aspects and its symbolic analog respectively). In the following, we will refer to covariance functions that are not composite as base covariance functions.

Knowledge about the composite nature of covariance functions is not new. However, until recently, the choice and the composition of covariance functions were done ad-hoc. The Automatic Statistician Project[6] came up with an approximate search over the possible space of kernel structures (Duvenaud et al., 2013; Lloyd et al., 2014). However, a fully Bayesian treatment was not available before. To the best of our knowledge, we are the first to present a fully Bayesian solution to this. We deploy a probabilistic context free grammar for our prior on structures (see Fig. 5.6)

Our probabilistic programming based MCMC framework approximates the following intractable integrals of the expectation for the prediction:

$$\mathbb{E}[y^* \mid x^*, \mathbf{D}, \mathbf{K}] = \iint f(x^*, \boldsymbol{\theta}, \mathbf{K})\, P(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{K})\, P(\mathbf{K}|\boldsymbol{\Omega}, s, n)\, \mathbf{d}\boldsymbol{\theta}\mathbf{d}\mathbf{K}. \tag{5.16}$$

We will explain the terms above one by one below. The approximation is done by sampling from the posterior probability distribution of the hyper-parameters and the possible kernel:

$$y^* \approx \frac{1}{T} \sum_{t=1}^{T} f(x^*|\boldsymbol{\theta}^{(t)}, \mathbf{K}^{(t)}). \tag{5.17}$$

In order to provide the sampling of the kernel, we introduce a stochastic process that simulates the grammar for algebraic expressions of covariance function algebra:

$$\mathbf{K}^{(t)} \sim P(\mathbf{K} \mid \boldsymbol{\Omega}, s, n) \tag{5.18}$$

Here, we start with a set of possible kernels and draw a random subset. For this subset of size n, we sample a set of possible operators that operate on the base kernels.

The marginal probability of a kernel structure which, allows us to sample, is characterized by the probability of a uniformly chosen subset of the set of n possible covariance

[6] `http://www.automaticstatistician.com/`

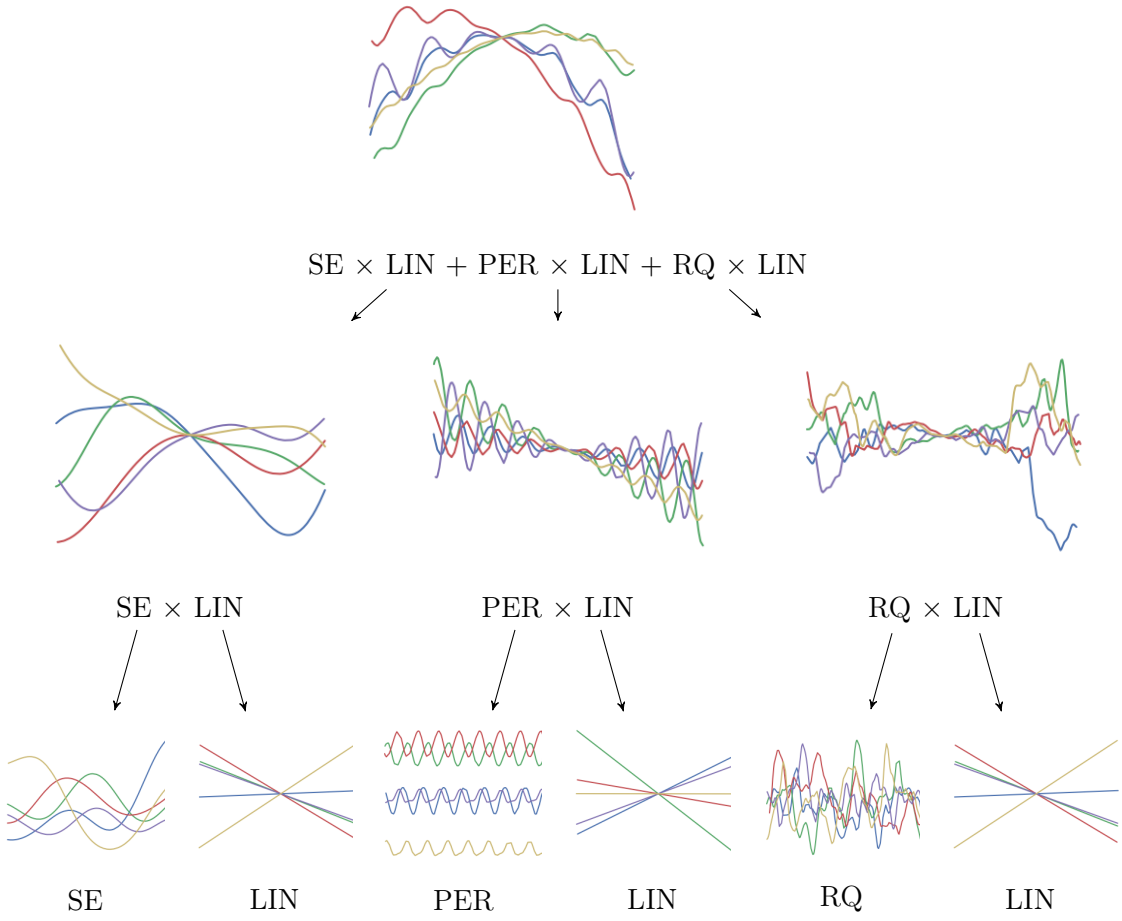


Figure 5.5: Composition of covariance function parsed using the example SE × LIN + PER × LIN + RQ × LIN. We deploy kernel summation (+) and kernel multiplication (×).

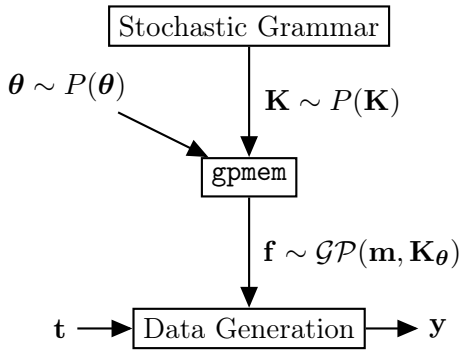


Figure 5.6: Graphical description of Bayesian GP structure learning.

functions times the probability of sampling a global or a local structure, which is given by a binomial distribution:

$$P(\mathbf{K} \mid \mathbf{\Omega}, s, n) = P(\mathbf{\Omega} \mid s, n) \times P(s \mid n) \times P(n), \tag{5.19}$$

with

$$P(\mathbf{\Omega} \mid s, n) = \binom{n}{r} p_{+\times}^{r} (1 - p_{+\times})^{n-r} \tag{5.20}$$

and

$$P(s \mid n) = \frac{n!}{\mid s \mid!} \tag{5.21}$$

where $P(n)$ is a prior on the number of base kernels used which can sample from a discrete uniform distribution. This will strongly prefer simple covariance structures with few base kernels since individual base kernels are more likely to be sampled in this case due to (5.21). Alternatively, we can approximate a uniform prior over structures by weighting $P(n)$ towards higher numbers. It is possible to also assign a prior for the probability to sample global or local structures. However, we have assigned complete uncertainty to this with probability $P = 0.5$.

Many equivalent covariance structures can be sampled due to covariance function algebra and equivalent representations with different parametrisation (Lloyd et al., 2014). Certain covariance functions can differ in terms of the hyper-parametrisation but can be absorbed into a single covariance function with a different parametrisation. To inspect the posterior of these equivalent structures we convert each kernel expression into a sum of products and subsequently simplify. Rules for this simplification can be found in appendix A.

The Automatic Statistician Project is implemented using deterministic optimization and search. We reproduce results from this Project in a Bayesian fashion using `gpmem` (Listing 5.4). We first define a prior on the hypothesis space. Note that, as in the implementation of the Automatic Statistician, we upper-bound the complexity of the space of covariance functions we want to explore. We also put vague priors on hyper-parameters.

Listing 5.4: Learning Latent Causal Structure

```
// GRAMMAR FOR KERNEL STRUCTURE
assume kernels = list(se, wn, lin, per, rq) // defined as above
// prior on the number of kernels
assume p_number_k = uniform_structure(n)
assume sub_s = tag(quote(grammar), 0,
                           subset(kernels, p_number_k))
assume grammar = proc(l) {
  // kernel composition
  if (size(l) <= 1)
       { first(l) }
  else { if (flip())
                  { add_funcs(first(l), grammar(rest(l))) }
              else { mult_funcs(first(l), grammar(rest(l))) }
       }
  }
assume K = tag(quote(grammar), 1, grammar(sub_s))
assume (f_compute f_emu) = gpmem(f_restr, K)

// Probe all data points
for n ... N
    predict f_compute(get_data_xs(n))

// PERFORMING INFERENCE
infer  repeat(2000, do(
                        mh(quote(grammar), one, 1),
                        for kernel ∈ K
                            mh(quote(hyper_kernel), one, 1)))
```

We defined the space of covariance structures in a way that allows us to reproduce results for covariance function structure learning as in the Automatic Statistician. This leads to coherent results, for example, for the CO_2 data set (see Rasmussen and Williams, 2006 for a full description). We find a posterior where the most probable structure is almost identical to the highest scoring result reported in previous experiments using a search-and-score method (Duvenaud et al., 2013, for our result, see below: Appendix C) . For the airline data set describing monthly totals of international airline passengers (Box et al., 1997, according to Duvenaud et al., 2013), we see that the two most probable alternatives for a structural description both recover the data dynamics (Fig. 5.7). However, the components factor in a different way due to different

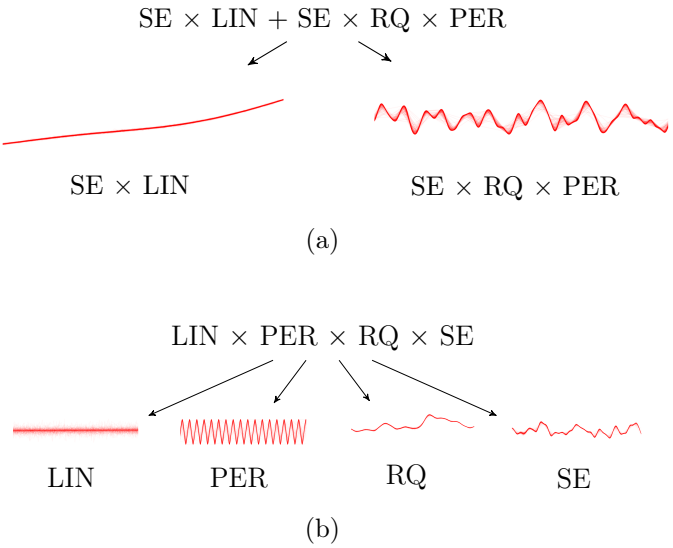


Figure 5.7: We see two possible hypotheses for the composite structure of $\mathbf{K}$. (a) Most frequent sample drawn from the posterior on structure. We have found two global components. First, a smooth trend (LIN × SE) with a non-linear increasing slope. Second, a periodic component with increasing variation and noise. (b) Second most frequent sample drawn from the posterior on structure. We found one global component. It is comprised of local changes that are periodic and with changing variation.

parametrisation of the individual base kernels.

Confident about our results, we can now query the data for certain structures being present. We illustrate this using the CO_2 data with an expressive hypothesis space. We assume a relatively simple one consisting of only four kernels, a linear, a smoothing, a periodic and a white noise kernel. In this experiment, we resort to the white noise kernel instead RQ (similar to Lloyd et al., 2014). We can now run the algorithm, compute a posterior of structures. We can also query this posterior distribution for the marginal of certain simple structures to occur. We demonstrate this in Fig. 5.8 We present an extensive case study deploying the method for discovery of latent causal structure in Appendix D.

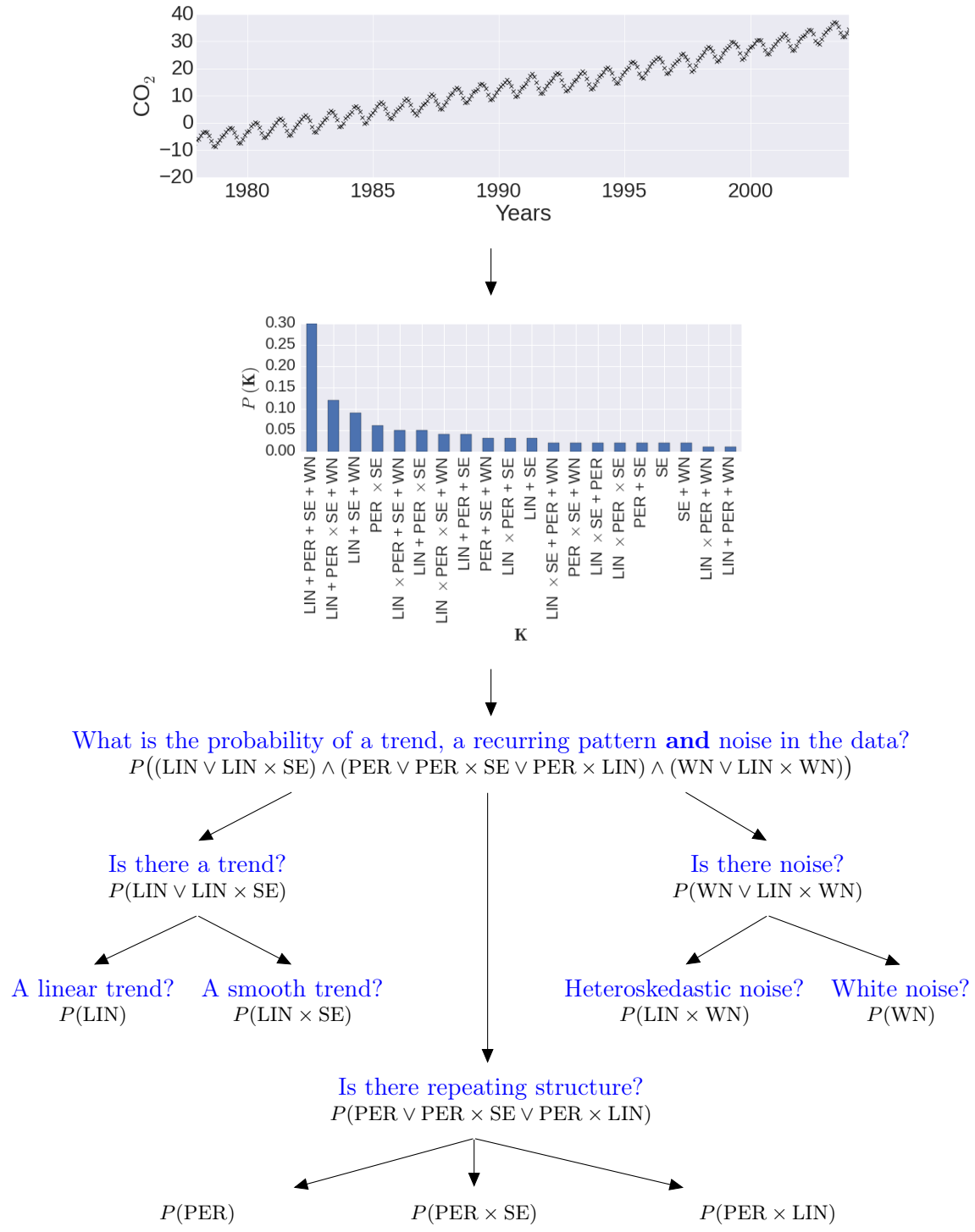


Figure 5.8: We start with an unstructured time series. The data is used to compute a posterior on symbolic structure. Like in symbolic reasoning we can query the data, checking if some logical statements are probable to be true. Here, we are interested in three qualitative aspects of the data: (i) Is it true that there is a trend? (ii) Is it true that there are recurring patterns and (iii) is it true that there is noise in the data?

5.5 Bayesian Optimization

Bayesian optimization casts the problem of finding the global maximum of an unknown function as a hierarchical decision problem (Ghahramani, 2015). Evaluating the actual function may be very expensive, for example in computation time or or memory consumption. For one example, when searching for the best configuration for the learning algorithm of a large convolutional neural network, a large amount of computational work is required to evaluate a candidate configuration, and the space of possible configurations is high-dimensional. Another typical example, alluded to in Section 5.3, is data acquisition: for machine learning problems in which a large body of data is available, it is often desirable to choose the right queries to produce a data set on which learning will be most effective. In continuous settings, many Bayesian optimization methods employ GPs (e.g. Snoek et al., 2012). To demonstrate Bayesian optimization with `gpmem` we have implemented a version of Thompson sampling using GPs in Venture[7].

5.5.1 Thompson Sampling Framework

Thompson sampling (Thompson, 1933) is a widely-used Bayesian framework for solving exploration-exploitation problems. Our implementation has two notable features: (i) the ability to search over a broader space of contexts than the parametric families that are typically used, and (ii) the parsimony of the resulting probabilistic program.

We describe the setup of Thompson sampling for Markov Decision Processes (MDP). An agent is to take a sequence of actions $a_1, a_2, \ldots$ from a (possibly infinite) set of possible actions $\mathcal{A}$. After each action, a reward $r \in \mathbb{R}$ is received, according to an unknown conditional distribution $P_{\text{true}}(r|a)$. The agent's goal is to maximize the total reward received for all actions. In Thompson sampling, the Bayesian agent accomplishes this by placing a prior distribution $P(\theta)$ on the possible "contexts" $\theta \in \Theta$. Here a context is a believed model of the conditional distributions $\{P(r|a)\}_{a\in\mathcal{A}}$, or at least, a believed statistic of these conditional distributions which is sufficient for deciding an action a. One example of such a sufficient statistic is the conditional mean $V(a|\theta) = \mathbb{E}[r|a,\theta]$, which can be thought of as a value function. Thompson sampling thus has the following steps, repeated as long as desired:

1. Sample a context $\theta \sim P(\theta)$.
2. Choose an action $a \in \mathcal{A}$ which (approximately) maximizes $V(a|\theta) = \mathbb{E}[r|a,\theta]$.

[7]The work described in section 5.5.1 and 5.5.2 was produced in collaboration with Ben Zinberg, who included some of it in his MEng thesis (2015).

3. Let r_{true} be the reward received for action a. Update the believed distribution on θ, i.e., $P(\theta) \leftarrow P_{\text{new}}(\theta)$ where $P_{\text{new}}(\theta) = P(\theta \mid a \mapsto r_{\text{true}})$.

Note that when $\mathbb{E}[r|a,\theta]$ (under the sampled value of θ for some points a) is far from the true value $\mathbb{E}_{P_{\text{true}}}[r|a]$, the chosen action a may be far from optimal, but the information gained by probing action a will improve the belief θ. This amounts to "exploration." When $\mathbb{E}[r|a,\theta]$ is close to the true value except at points a for which $\mathbb{E}[r|a,\theta]$ is low, exploration will be less likely to occur, but the chosen actions a will tend to receive high rewards. This amounts to "exploitation." Roughly speaking, exploration will happen until the context θ is reasonably sure that the unexplored actions are probably not optimal, at which time the sampler will exploit by choosing actions in regions it knows to have high value.

Typically, when Thompson sampling is implemented, the search over contexts $\theta \in \Theta$ is limited by the choice of representation. In traditional programming environments, θ often consists of a few numerical parameters for a family of distributions of a fixed functional form. With work, a mixture of a few functional forms is possible; but without a probabilistic programming machinery, implementing a rich context space Θ would be an unworkably large technical burden. In a probabilstic programing language, however, the representation of heterogeneously structured or infinite-dimensional context spaces is quite natural. Any computable model of the conditional distributions $\{P(r|a)\}_{a\in\mathcal{A}}$ can be represented as a stochastic procedure $(\lambda(a)\ldots)$. Thus, for computational Thompson sampling, the most general context space $\widehat{\Theta}$ is the space of program texts of a probabilistic programming language like Venture. Any other context space Θ has a natural embedding as a subset of $\widehat{\Theta}$.

5.5.2 Thompson Sampling in Venture

Because Venture supports sampling and inference on (stochastic) procedure-valued random variables (and the generative models which produce those procedures), Venture can capture arbitrary context spaces as described above. To demonstrate, we have implemented Thompson sampling in Venture in which the contexts θ are GPs over the action space $\mathcal{A} = \mathbb{R}$. That is, $\theta = (\mu, K)$, where the mean μ is a computable function $\mathcal{A} \to \mathbb{R}$ and the covariance K is a computable (symmetric, positive-semidefinite) function $\mathcal{A} \times \mathcal{A} \to \mathbb{R}$. This represents a GP $\{R_a\}_{a\in\mathcal{A}}$, where R_a represents the reward for action a. Computationally, we represent a context not as a pair of infinite lookup tables for μ and K, but as a finite data structure $\theta = (K_{\text{prior}}, \sigma, \ell, \mathbf{a}_{\text{past}}, \mathbf{r}_{\text{past}})$, where

- $K_{\text{prior}} = K_{\text{prior},\sigma,\ell}$ is a procedure, with parameters σ, ℓ, to be used as the prior covariance function: $K_{\text{prior}}(a, a') = \sigma^2 \exp\left(-\frac{(a-a')^2}{2\ell^2}\right)$

- σ and ℓ are (hyper)parameters for K_{prior}
- $\mathbf{a}_{\text{past}} = (a_i)_{i=1}^n$ are the previously probed actions
- $\mathbf{r}_{\text{past}} = (r_i)_{i=1}^n$ are the corresponding rewards

To simplify the treatment, we take prior mean $\mu_{\text{prior}} \equiv 0$. The mean and covariance for θ are obtained by the usual conditioning formula:

$$\begin{aligned}
\mu(\mathbf{a}) &= \mu\left(\mathbf{a} \mid \mathbf{a}_{\text{past}}, \mathbf{r}_{\text{past}}\right) \\
&= K_{\text{prior}}(\mathbf{a}, \mathbf{a}_{\text{past}})\, K_{\text{prior}}(\mathbf{a}_{\text{past}}, \mathbf{a}_{\text{past}})^{-1}\, \mathbf{r}_{\text{past}} \\
K(\mathbf{a}, \mathbf{a}) &= K\left(\mathbf{a}, \mathbf{a} \mid \mathbf{a}_{\text{past}}, \mathbf{r}_{\text{past}}\right) \\
&= K_{\text{prior}}(\mathbf{a}, \mathbf{a}) - K_{\text{prior}}(\mathbf{a}, \mathbf{a}_{\text{past}})\, K_{\text{prior}}(\mathbf{a}_{\text{past}}, \mathbf{a}_{\text{past}})^{-1}\, K_{\text{prior}}(\mathbf{a}_{\text{past}}, \mathbf{a}).
\end{aligned}$$

Note that even in this simple example, the context space Θ is not a finite-dimensional parametric family, since the vectors $\mathbf{a}_{\text{past}}$ and $\mathbf{r}_{\text{past}}$ grow as more samples are taken. Θ is, however, quite easily representable as a computational procedure together with parameters and past samples, as we do in the representation $\theta = (K_{\text{prior}}, \sigma, \ell, \mathbf{a}_{\text{past}}, \mathbf{r}_{\text{past}})$.

5.5.3 Implementation with `gpmem`

As a demonstration, we use Thompson sampling to optimize an unknown function $V(x)$ (the value function) using `gpmem`.

We assume V is made available to Venture as a black-box. The code for optimizing V is given in Listing 5.5. For step 3 of Thompson sampling, the Bayesian update, we not only condition on the new data (the chosen action a and the received reward r), but also perform inference on the hyper-parameters σ, ℓ using a Metropolis–Hastings sampler. These two inference steps take 1 line of code: 0 lines to condition on the new data (as this is done automatically by `gpmem`), and 1 line to call Venture's built-in `MH` operator. The results are shown in Figures 5.9 and 5.10. We can see from the figure that, roughly speaking, each successive probe point a is chosen either because the current model V_{emu} thinks it will have a high reward, or because the value of $V_{\text{emu}}(a)$ has high uncertainty. In the latter case, probing at a decreases this uncertainty and, due to the smoothing kernel, also decreases the uncertainty at points near a. We thus see that our Thompson sampler simultaneously learns the value function and optimizes it.

In Listing 5.5, we see code for a complete Bayesian optimization system with Thompson sampling using `gpmem`. In the loop (line 21), `f_compute` is called to probe the value of `f` at a new argument. The new argument, `(mc_argmax f_emu_pointwise mc_sampler)`, is a Monte Carlo estimate of the maximum pointwise sample of `f_emu`

(itself a stochastic quantity), with the Monte Carlo samples being drawn in this case uniformly between -20 and 20.

Listing 5.5: Bayesian optimization using `gpmem`

```
1  assume sf = tag(quote(hyper), 0, uniform_continuous(0, 10))
2  assume l = tag(quote(hyper), 1, uniform_continuous(0, 10))
3  assume se = make_squaredexp(sf, l)
4  assume blackbox_f = get_bayesopt_blackbox()
5  assume (f_compute, f_emu) = gpmem(blackbox_f, se)

6  define uniform_candidate = proc(prev_xs) {
7    uniform_continuous(-20, 20)
8  }

9  define mc_argmax = proc(func, prev_xs) {
     // Monte Carlo estimator for the argmax of func.
10   run(do(
11     candidate_xs <- mapv(proc(i) {uniform_candidate(prev_xs)},
12                          linspace(0, 19, 20)),
13     candidate_ys <- mapv(func, candidate_xs),
14     lookup(candidate_xs, argmax_of_array(candidate_ys))))
15 }

16 define emulator_point_sample = proc(x) {
17   run(sample(lookup(
18     f_emu(array(x)),
19     0)))
20 }

21 infer repeat(15, do(pass,
        // Phase 2: Call f_compute on the next probe point
22      predict f_compute(
23                  mc_argmax(emulator_point_sample, '_)),
        // Phase 1: Hyper-parameter inference
24      mh(quote(hyper), one, 50)))
```

After each new call to `f_compute`, the Metropolis–Hastings algorithm is used to perform inference on the hyper-parameters of the covariance function in the GP model in light of the new conditioning data (line 24). Once enough calls to `f_compute` have been made (in our case we stopped at 15 calls), we can inspect the full list of probed

(a, r) pairs with `extract_stats`. The answer to our maximization problem is simply the pair having the highest r; but our algorithm also learns more potentially useful information. This entails the right action to take to balance steps to gain information and steps to climb the curve to reach a nearby optimum.

5.6 Summary

We have introduced novel stochastic processes for a probabilistic programming language and unified them into a new technique. We have shown that this technique is useful for probabilistic programming with continuous-valued random variables. It can be applied to causal discovery as illustrated with causal hyper-prior systems and causal discovery for latent symbolic concepts for time series data. We have demonstrated how flexible non-parametric models can be treated in Venture in only a few lines of code. We evaluated our contribution on a range of hard problems for state-of-the-art Bayesian non-parametrics. The programs implemented with `gpmem` showed competitive performance in all of them.

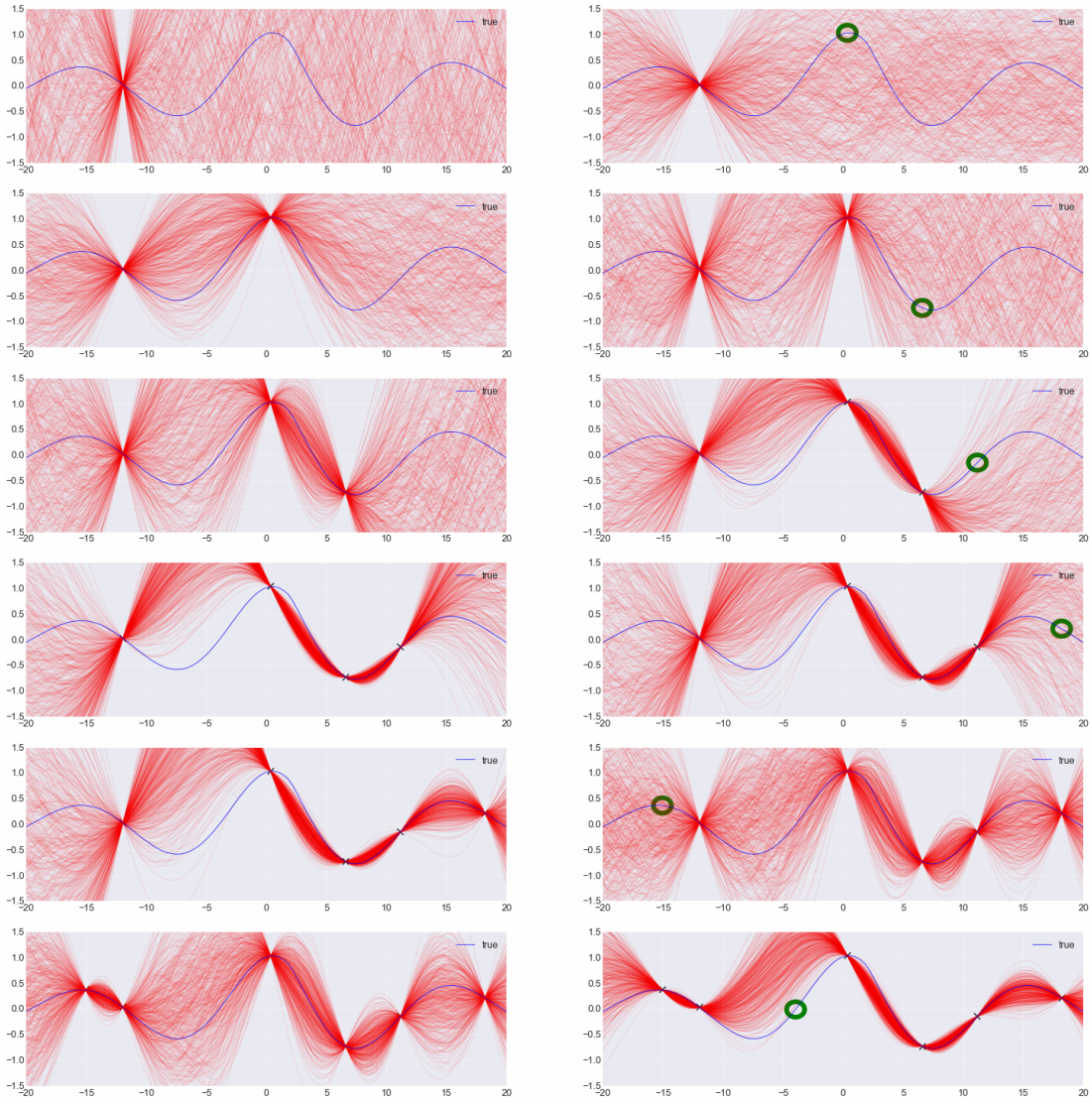


Figure 5.9: Dynamics of Thompson sampling in Venture. The blue curve is the true function V, and the red region is a blending of 100 samples of the curve generated (jointly) by a GP-based emulator V_{emu}. The left and right columns show the state of V_{emu} before and after hyper-parameter inference is run on the new data, respectively. In the right column, the next chosen probe point is circled in green. Each successive probe point a is the (stochastic) maximum of V_{emu}, sampled point-wise and conditioned on the values of the previously probed points. Note that probes tend to happen at points either where the value of V_{emu} is high, or where V_{emu} has high uncertainty. The sequence continues in Fig. 5.10.

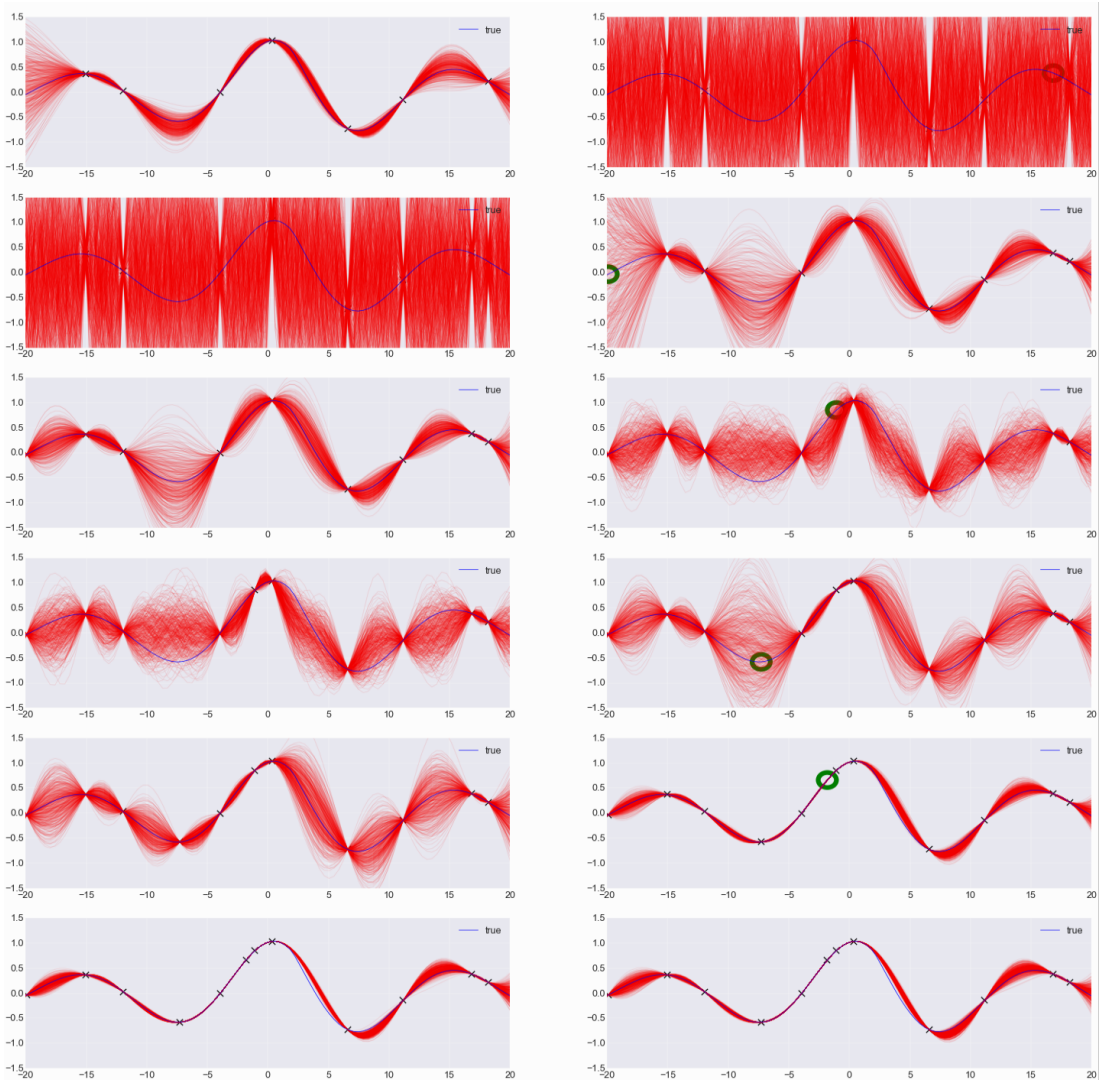


Figure 5.10: Dynamics of Thompson sampling in Venture (continued). We can see, for example, that after the seventh probe point in the sequence (top), the MH sampler chose a "crazy" set of hyper-parameters, which was corrected at the next inference step.

6 | Conclusion

In this thesis, we have studied models and algorithms for causal discovery and tested them using data generated in complex dynamical systems and processes. The work has strived to bridge the existing gap between probabilistic time series, dynamical systems research and causal discovery. A large effort of the work has focused on computational models and algorithms that exploit prior knowledge about temporal and dynamical dependencies in data. Even if such relations can be considered conceptually unrelated to causal inference, the thesis studies how they can be used to inform both modelling of and inference over causal relations. In addition, the solutions provided by the thesis cater for several kinds of data, generated in various types of complex dynamic systems and processes. Such data can differ by various characteristics like dimensionality and size, whether they are observed or not, how much we know about it, and what kind of underlying mechanism generated it.

First, we have introduced Multidimensional Causal Discovery (MCD), suggesting a notion of cause and effect over time. This notion was aimed at large data sets as they exist in real-world databases, which we processed in form of tensors. We have implemented the notion of cause and effect into the MCD-algorithm for causal discovery. The method is more expressive than related approaches, as it generalises causal effects over points in time. In contrast to conventional approaches, the algorithm does not output large graphs containing a node for every random variable for every point in time. It outputs causal relations over variables that statistically hold true over the entire time that is reflected in the data. We have also shown the algorithms accuracy and its performance compared to relevant benchmarks.

We have then introduced new, weaker assumptions for causal discovery by generalising causal discovery for scenarios with dynamically changing noise levels. The FlexCD model and algorithm was explained and its application to a widely accepted benchmark data set was illustrated. We demonstrated performance gains for the two-variable problem of causal discovery. The method relies on GPs with dynamically changing levels of noise. Inspecting this noise, it prefers the direction in which the noise is less het-

eroskedastic. This is in line with ANM related approaches. The direction in which the noise depends in a less complicated way on the data generating function is preferred. To measure heteroskedasticity we deployed a novel GP that accounts for different levels of noise.

Finally, we have demonstrated `gpmem`, a new probabilistic programming technique that propels causal discovery. We have provided an efficient implementation of `gpmem` in form of a self-caching wrapper that remembers previously computed values. `gpmem` produces a statistical emulator, which improves with every data-point that becomes available. The efficacy of the technique has been demonstrated with a number of problems of the state-of-the-art in machine learning. It provides a fully Bayesian treatment of model selection tasks, needed for causal discovery, that was not available before. To demonstrate this, we applied `gpmem` to help discover symbolic statements that cause certain dynamics to emerge over time. These concepts are latent, and are discovered in the data with empirical inference. We have also shown its applicability to the learning of hierarchical, causal hyper-parameter structures where hyper-parameters are dependent similar to a causal Bayesian network. Finally, we have shown an application of the technique to Bayesian Optimization.

In summary, we have presented a number of methods, criteria and algorithms for causal discovery that uses data generated in various kinds of dynamical systems. All these methods have been tested with both synthetic and real-world data yielding competitive performance.

6.1 Future Work

In the following, we will first elaborate on chapters 3 to 5 one by one, identifying smaller future efforts that directly result from the research described in these chapters. We will then discuss more general future work that continues the journey started four years ago and take further the experience and insights gained (section 6.1.4).

6.1.1 Multidimensional Causal Discovery

The practical value of MCD analysis needs to be determined by applying it to more real-world data sets and comparing it to other causal inference methods for non-experimental data. The real-world data analysed here are rather simple as they contain relations between two variables only. It has been quite difficult to find multidimensional time series where the underlying causality is clear. Here it would be useful to see how we can also include discrete-valued variables into the MCD analysis. In most cases of non-experimental data sets, we can find discrete-valued variables as well as continuous-

valued ones.

Also, in the current method, the tensor analytic process of flattening the data relies on the variance of the linear interaction between the decomposed subspaces. A more direct integration of this aspect into the LiNGAM discovery process is desirable. We aim to address this issue in future research too.

6.1.2 Flexible Causal Discovery

Future work involves a Markov Chain Monte Carlo implementation of the FlexCD algorithm, preferably in the context of probabilistic programming with methods described in chapter 5. It should be straightforward to implement the heteroskedastic GP with MCMC samplers. Will performance be worse here than with the $\mathbb{KL}$-base approximation as expected? The FlexCD algorithm should also be extended for problems with more than two variables. Graph-based algorithms that form a hybrid between ANM and constraint-based causal discovery are known (Tillman et al., 2009) and should also be extensible for FlexCD.

6.1.3 Abstract Dynamical Causality & Probabilistic Programming

The closing chapter of our contribution provided novel means of causal discovery that were previously unavailable. Future work that directly builds on this includes finding novel problems that one could tackle with memoization in continuous-valued uncertain domains. The use of `gpmem` its implication for computation and the trade off between time and space complexity can be tested in various applications. An example for this would be the computation of Fibonacci numbers. This has never been approached via a regression framework. Another important question is which online-learning methods can be recovered with `gpmem`.

A necessary and also interesting experiment would be an extensive empirical comparison of performance with regards to different bottlenecks. In the relevant literature, slice sampling and HMC have been deployed for MCMC-based GP (Neal, 1997). We resort to probabilistic programming-based MH, but both slice sampling and HMC are available in Venture. What are the performance gains and how can they be measured?

Another pressing insight from this research is that we need to experiment with approximative inference for GP-inference (not the Venture inference routines) in order to cope with large data sets and potential problems that their size brings.

6.1.4 Future Directions

Identifying a suitable benchmark for causal discovery is the most important bottleneck. We think that the choice of data sets for the benchmark which has served in many papers on causal discovery (recently described in the article by Mooij et al., 2014) may not be the ideal to evaluate causal discovery in general. For example, we see that the cardinality of a data dimension indicates it being the cause: often the continuous-valued cause is artificially discretised in this data set whereas the effect is not. Also, some of the data sets are prime examples for latent confounders. An example for this is the case where the latitude of a country is treated as the cause for its GDP. Furthermore, this benchmark consists almost exclusively of bivariate problems. A general causal discovery framework should be able to cope with bivariate problems, multivariate problems, time series data, symbolic causal discovery, latent concepts and latent confounders. Each problem requires a set of real world data sets a new causal discovery algorithm can work on. The ill-defined concepts regarding causality make it difficult to find data sets where the researcher can be certain about the ground truth, that is the true causal directionality. One must also decide on the variety of the functional form of synthetic data to test the assumptions. While this effort seems trivial, it is key to allow progress in causal discovery.

Another highly interesting direction of research is a deeper study of Bayesianism in causal discovery. How can we mix prior assumptions about observed asymmetries for cause and effect in a truly Bayesian way?

Many claims made by the causal discovery community about identifiability are proven using probability density functions as a gold standard to model probability distributions. Do the results on identifiability still hold when distributions are modelled with a sampler as in probabilistic programming? Or could it be that asymmetries arise between causal and anti-causal models in their likelihood? One argument of functional causal discovery is that $P(Y \mid X)$ depends in a less complex manner on $P(X)$ than $P(X \mid Y)$ depends on $P(Y)$. We mentioned in section 2.4.2 that complexity is ill-defined here. But this only holds true for the traditional ways of modelling probability distributions rather than for probabilistic programming. In cases where asymmetries between cause and effect do not arise simply through means of a different way of modelling probability distributions, we would like to define suitable measures of complexity over execution traces of probabilistic programs. These could be more accurate than the measures defined in previous work (Mooij et al., 2010; Daniušis et al., 2010; Janzing et al., 2012; Chen et al., 2014).

For many practitioners computational complexity is the bottleneck of interest. We mentioned that the causal discovery for graphical models can be NP-hard. We have

also begun to show symbolic reasoning for causal discovery. Previous research showed significant performance gains in machine learning through lifted computation (e.g. Choi et al., 2010, 2011; Choi and Amir, 2012): parameter sharing and parameter typing over semantically related concepts is expected to gain significant benefits. Identifying and modifying the right framework for causal discovery will propel causal inference in real world applications. Concepts of generalisation and abstraction to propel computation come in many forms. We have chosen probabilistic programming, specifically Venture for a first step in that direction showing a symbolic language of kernel algebra that can be learned. We should elevate the symbolic structure to a context where it can learn statements of a programming language such as Venture.

Finally, we think that research is now ready to transfer results from causal discovery and probabilistic programming to the medical domain. Highly structured time series data is readily available (Saeed et al., 2002). A study of adverse drug effects (as described in the introduction) is feasible once the methods described in this thesis are close to optimal when it comes to both time and space complexity. Given such optimality, the work described can be directly applied to data from existing electronic medical records. Aims for such a project would be to unify and test causal inference domains where variables can be both discrete-valued and continuous-valued. An approach should be Bayesian, not only to cope with missing values whilst providing estimates of uncertainty and confidence but also because we have shown the broad applicability of Bayesian methods like GPs in previous chapters. A software suite that is functional in the medical domain should exploit logical structures present in the data. For the case of the MIMIC2 database (Saeed et al., 2002), ontological coding is available - allowing logical inference over instances of observations. Finally, such a project should be conducted in cooperation with medical experts, able to differentiate between real and unreal effects, and to formulate interesting causal queries for a probabilistic model.

Appendix

A - Covariance Simplification

```
1  SE × SE                    → SE
2  {SE,PER,C,WN} × WN         → WN
3  LIN + LIN                  → LIN
4  {SE,PER,C,WN,LIN} × C      → {SE,PER,C,WN,LIN}
```

Rule 1 is derived as follows:

$$\begin{aligned}
\sigma_c^2 \exp(-\frac{(x-x')^2}{2\ell_c^2}) &= \sigma_a^2 \exp(-\frac{(x-x')^2}{2\ell_a^2}) \times \sigma_b^2 \exp(-\frac{(x-x')^2}{2\ell_b^2}) \\
&= \sigma_c^2 \exp(-\frac{(x-x')^2}{2\ell_a^2}) \times \exp(-\frac{(x-x')^2}{2\ell_b^2}) \\
&= \sigma_c^2 \exp\left(-\frac{(x-x')^2}{2\ell_a^2} - \frac{(x-x')^2}{2\ell_b^2}\right) \\
&= \sigma_c^2 \exp\left(-\frac{(x-x')^2}{2\ell_c^2}\right)
\end{aligned} \tag{6.1}$$

For stationary kernels that only depend on the lag vector between x and x' it holds that multiplying such a kernel with a WN kernel we get another WN kernel (Rule 2). Take for example the SE kernel:

$$\sigma_a^2 \exp\left(-\frac{(x-x')^2}{2\ell_c^2}\right) \times \sigma_b \delta_{x,x'} = \sigma_a \sigma_b \delta_{x,x'} \tag{6.2}$$

Rule 3 is derived as follows:

$$\theta_c(x \times x') = \theta_a(x \times x') + \theta_b(x \times x') \tag{6.3}$$

Multiplying any kernel with a constant obviously changes only the scale parameter of a kernel (Rule 4).

B - Convergence FlexCD Objective Function

We solve different optimisation problems with a single objective function. We need to find the optimal permutation of $\mathbf{W}$ which is needed for combination with the VHGPR optimisation for both hypotheses.

The key for understanding why this optimisation is correct is that convergence follows from the fact that gradient steps on each of the summands converge and that the summands are independent from one another. In detail, we say that we would like to maximize independence of:

1. $\mathbf{f}_{\boldsymbol{\alpha}}$ and $\mathbf{g}_{\boldsymbol{\alpha}}$ in terms of $\mathbb{K}$, with respect to $\boldsymbol{\alpha}$;
2. $\mathbf{f}_{\boldsymbol{\beta}}$ and $\mathbf{g}_{\boldsymbol{\beta}}$ in terms of $\mathbb{K}$, with respect to $\boldsymbol{\beta}$;
3. linear components in terms of a contrast function with respect to $\mathbf{w}_1$ and $\mathbf{w}_2$, where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2]$.

The trick here is that we can treat $\mathbf{w}_1$ and $\mathbf{w}_2$ as constants when optimizing 1. and 2. For 3., we make the necessary assumption that to create linear components that are statistically as independent as possible, this does not depend on the other function of interest. This function describes the dependence relationship between $\mathbf{f}_{\boldsymbol{\alpha}}$ and $\mathbf{g}_{\boldsymbol{\alpha}}$ or $\mathbf{f}_{\boldsymbol{\beta}}$ and $\mathbf{g}_{\boldsymbol{\beta}}$ respectively. 3. *only* depends on the contrast function that maximizes statistical independence for linear components. The objective therefore dictates, that 3. is independent of 1. and 2. Thus, convergence for the $\mathbf{w}_1$ and $\mathbf{w}_2$ follows from Hyvärinen, 1999, Appendix A.

$\mathbf{f}_{\boldsymbol{\alpha}}$, $\mathbf{g}_{\boldsymbol{\alpha}}$ does not share free parameters with $\mathbf{f}_{\boldsymbol{\beta}}$, $\mathbf{g}_{\boldsymbol{\beta}}$, thus, convergence for 1. and 2. follows from Lázaro-Gredilla and Titsias, 2011.

C - Discovery of Latent Causal Structure with CO_2 Data

Fig. C-1 illustrates coherent results for the CO_2 data set (see Rasmussen and Williams, 2006 for a description). We have cut the tail of the distribution for space reasons since the number of possible structures is large. We have computed leave-one crossvalidation for quality insurance purposes resulting in 545 independent Markov chains. Each chain consisted of 2000 nested steps each. We see the final sample of the each of the 545 chains. Note that Duvenaud et al. (2013) report LIN $\times$ SE + PER $\times$ SE + RQ $\times$ SE. We see that the most probable structure is almost identical with what was reported by previous work. Given the total of 545 samples, the frequency of the highest scoring structure corresponds to an MH posterior probability of approximately 0.15 that this structure is true.

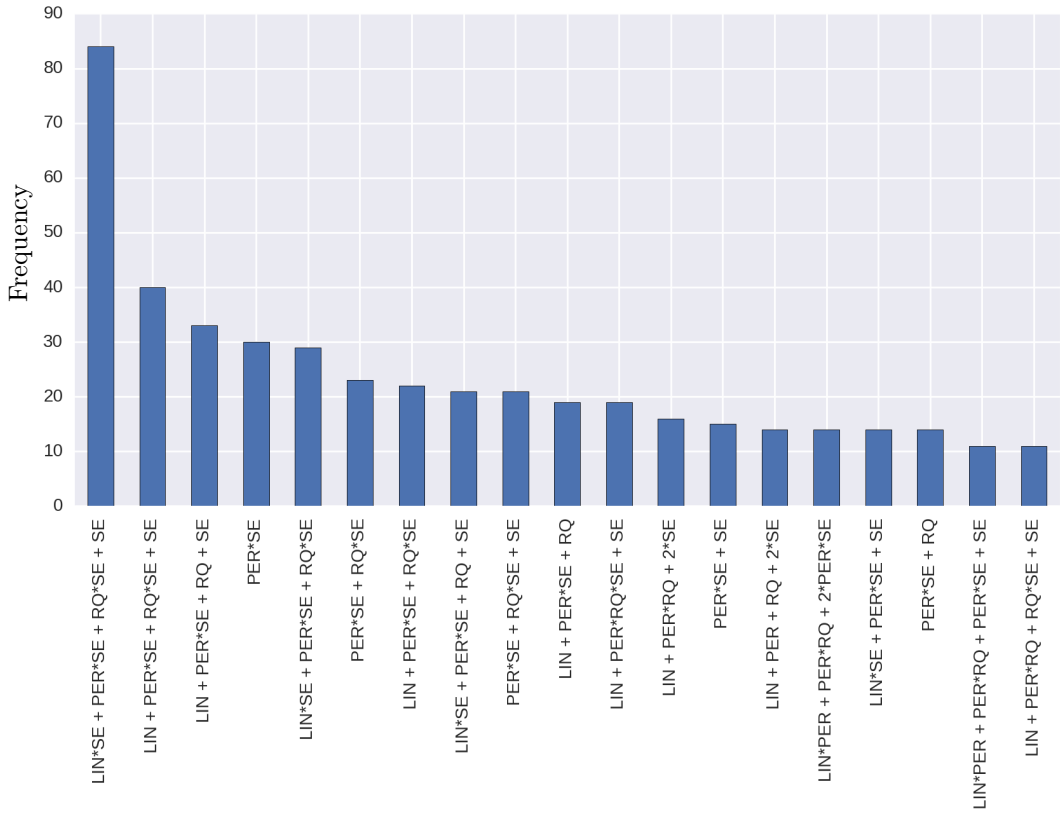


Figure C-1: Posterior on structure of the CO_2 data. Frequency refers to the frequency of a sample when we sample with MH at the end of a chain.

D - Case Study: Discovery of Latent Causal Structure with Medical Data in COMMODITY$_{12}$

The COMMODITY$_{12}$ project has designed, built, and performed trials with an intelligent Personal Health System (PHS) for the analysis of multi-parametric medical data (Kafalı et al., 2013). The specific focus of the work has been to develop a multi-layered multi-parametric infrastructure for continuous analysis of the advancement of diabetes, type 1 and 2, and the interaction of these two types with cardiovascular co-morbidities. This system is capable of taking the benefit of multi-parametric data, such as physiological measurements, genetic data, laboratory reports, and other measurements related to a person's activity, lifestyle and surrounding environment. The system also provides both to the health-care worker, as well as thr patient, with clinically relevant indicators for the prevention and management of diabetes type 1 and 2, and co-morbidities.

The management of diabetes is becoming an increasingly important problem worldwide with recent efforts aiming at controlling short and long term complications. The normalisation of blood glucose is one of the parameters that must be monitored by a personal health system according to a formal model of the disease (i.e., medical guidelines). This is especially important for *Type 1* diabetic patients, who need to have their glucose levels monitored continuously. In this context, two of the most important short-term complications of diabetes are hypoglycemia and hyperglycemia (very low and high blood glucose; respectively), both of which can be life-threatening (Knobbe and Buckingham, 2005; Palerm and Bequette, 2007; Turksoy et al., 2013).

We will now apply the latent causal structure learning that we have developed and use it for the analysis of the Continuous Glucose Monitoring (CGM) data collected in the COMMODITY$_{12}$ project. For this research project glucose measurements of subjects suffering from type 1 diabetes were collected which form the time series of interest. In a clinical trial we have measured glucose continuously for 24 hours with five subjects. Measurements were recorded every 5 minutes (Fig. D-1). The two specific medical tasks that we will seek to demonstrate are: prediction in terms of extrapolating glucose values and diagnosis in terms of finding symbolic (qualitative) structure underlying the observations represented by the data.

For prediction, we extrapolate 6 steps ahead in time which corresponds to 30 minutes. There are two important points to be noted here:

- If we predict a Hypoglycemia episode 30 minutes ahead, then there is enough time for the COMMODITY$_{12}$ system to send an alert to the patient to increase

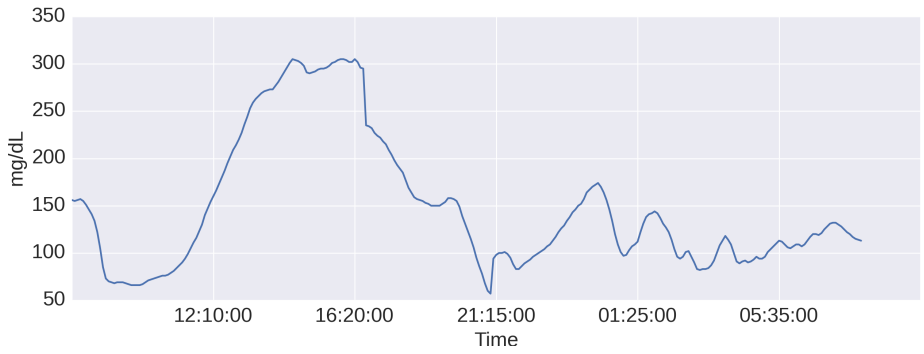


Figure D-1: CGM data for one patient. We see an episode of Hypoglycemia at 21:14

glucose intake. This could potentially avoid dangerous situations in practice with little effort.

- We sample from the posterior distribution which means that every prediction comes with a degree of confidence defined by our posterior belief state and the evidence available. Areas with more variability in the drawn posterior samples indicate more uncertainty about what is the actual, true underlying function that generates the data.

Data-driven Diagnosis with Prediction for Clinical Decision Making

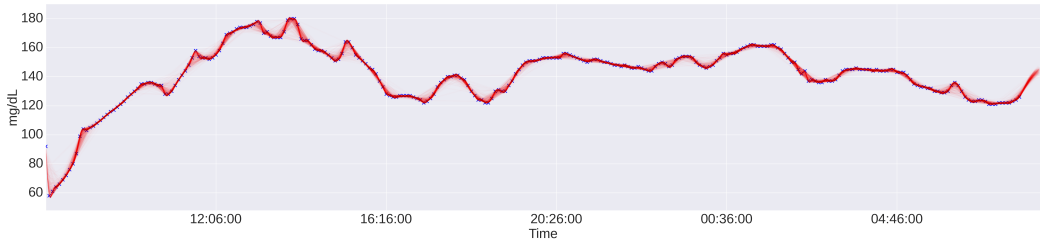
Finding symbolic, qualitative structures that are causing certain observations to emerge was one of our main objectives for data-driven diagnosis. The idea of symbolic reasoning is a re-occurring pattern of artificial intelligence research. We have seen above that our realisation of this is not only a gimmick that arises when one tries to emulate human thinking but a way to gain competitive predictive performance. We will now investigate what are the symbolic structures that we have "diagnosed". We will see that those structures are not only coherent among patients but also make intuitive sense when there explanatory power is analysed.

We find that all posterior distributions peak at the same kernel structure, which implies a Gaussian process with the following kernel structure:

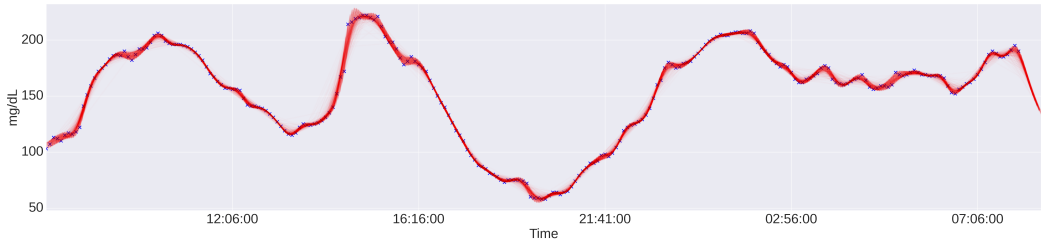
$$\mathbf{K} = LIN + PER + RQ + SE. \tag{6.4}$$

The most probable structure in the posterior suggests that there are four global components that comprise the time-series:

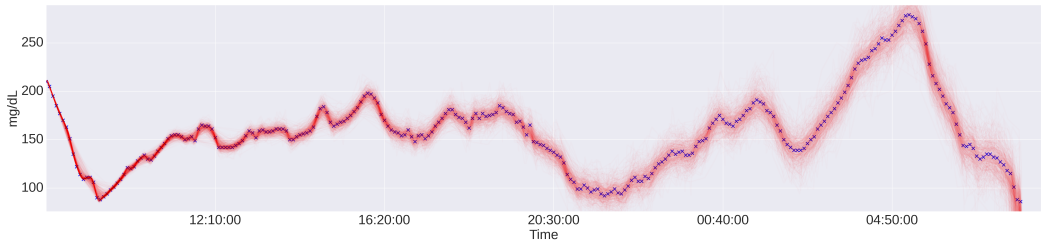
1. a linear trend (LIN);
2. a periodic component (PER);



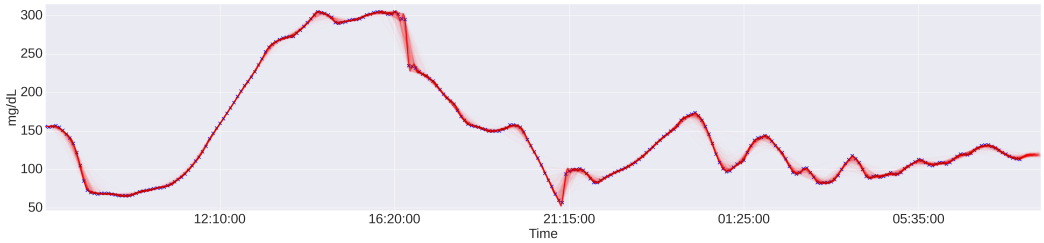
(a)



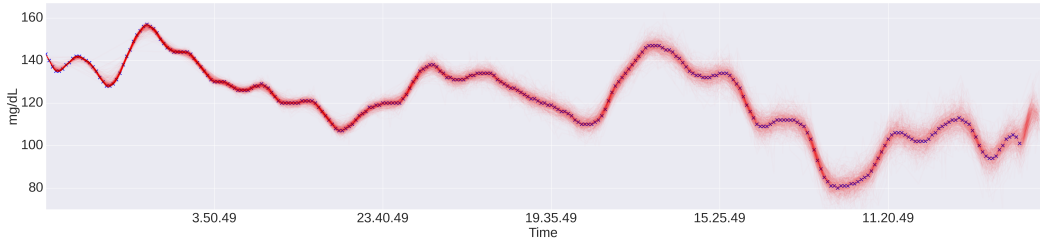
(b)



(c)



(d)



(e)

Figure D-2: (a) - (e) Posterior Samples drawn from a GP learned with latent causal structure learning. Blue dots correspond to the data, while red lines denote samples from the posterior.

3. a smooth trend (SE); and

4. noise (RQ).

All of the above components contribute to what we see as time series (compare the posterior distribution of symbolic components with the observed time series of Fig. D-3 - D-7 (b) with D-3 - D-7 (a); note that the y-axis for all structures (a) depicts unnormalized samples. For a posterior probability, divide the Monte Carlo samples by 100).

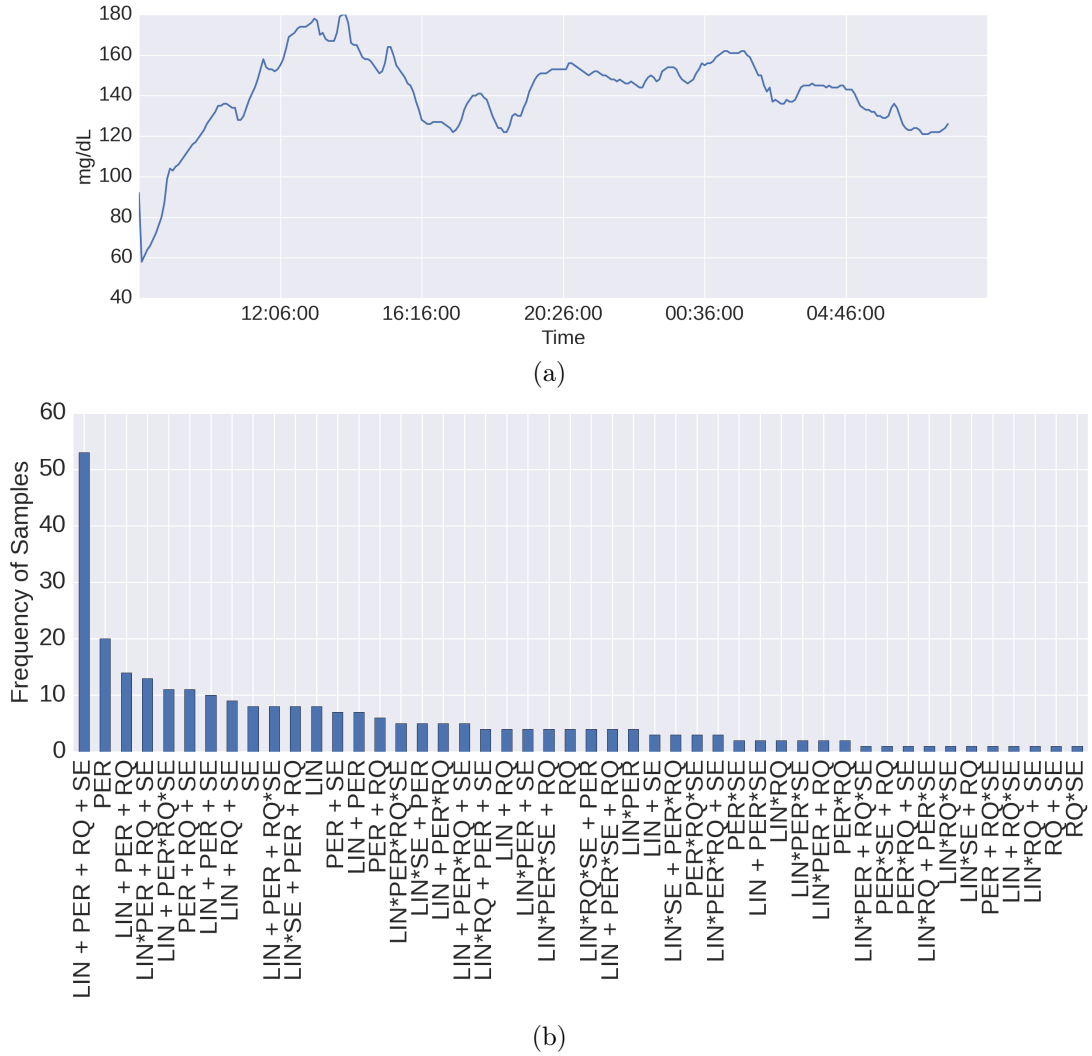
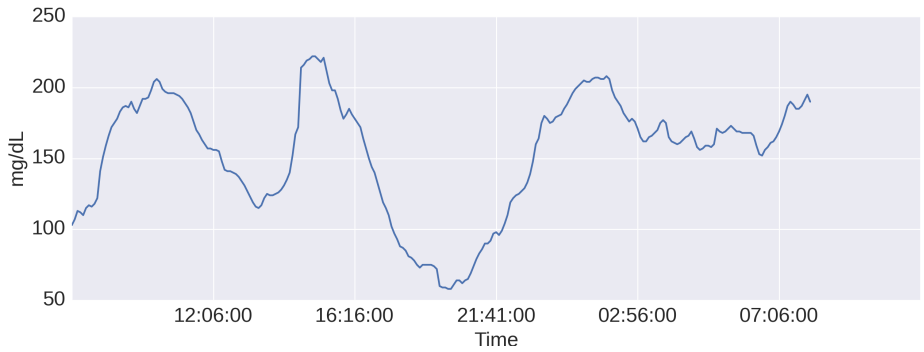
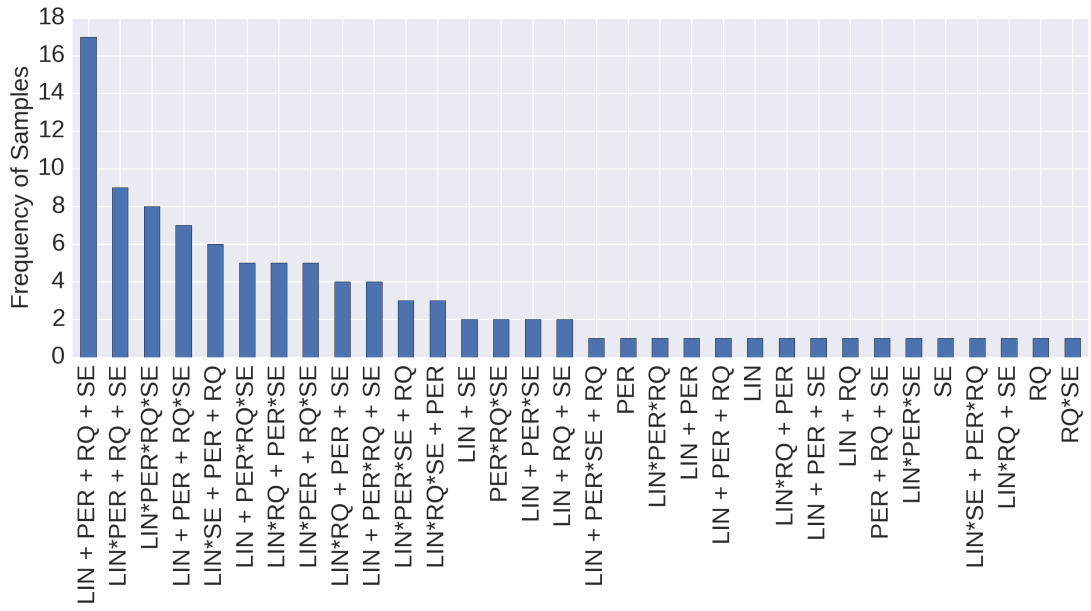


Figure D-3: (a) Glucose data of patient 1. (b) 100 samples drawn from the posterior distribution on kernel structure.

We will now illustrate how we can use our approach for data driven diagnosis in



(a)



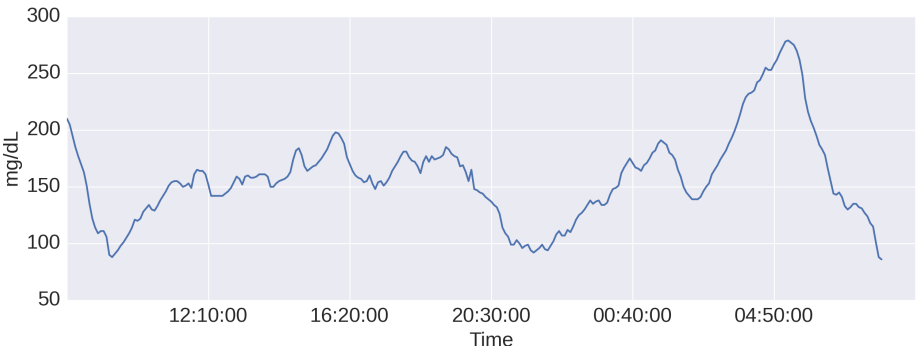
(b)

Figure D-4: (a) Glucose data of patient 2. (b) 100 samples drawn from the posterior distribution on kernel structure.

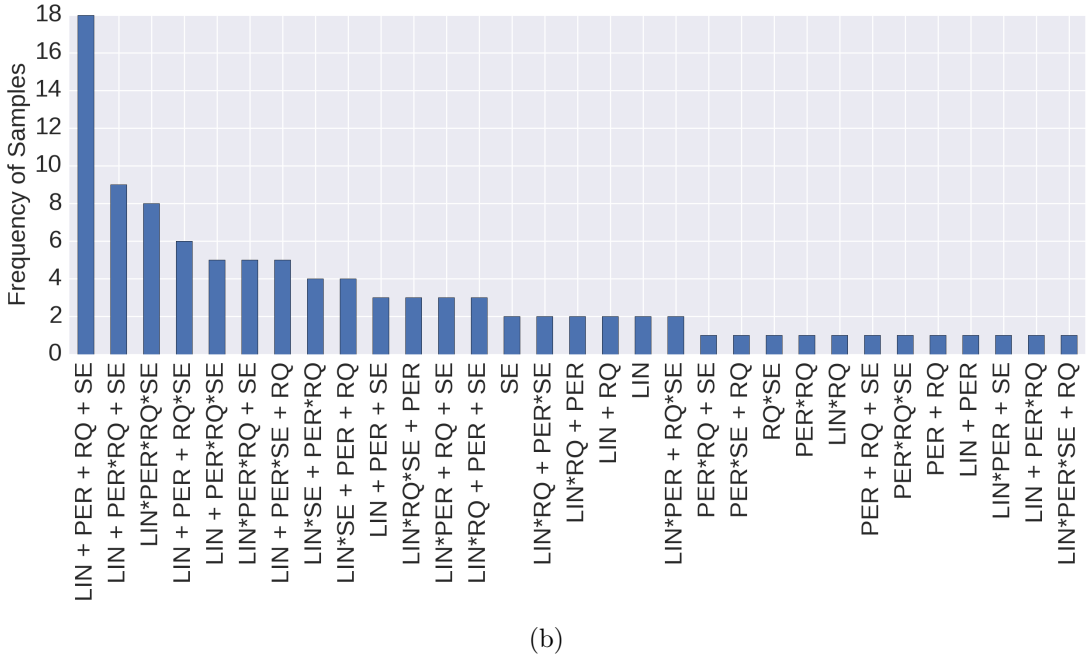
combination with prediction for clinical decision making using the data of a patient, patient 3, as an example. We see that for patient 3, like all patients, the agent posterior state of belief indicates a globally recurring pattern, that is $P(PER)$ is high. With high level reasoning, a clinical decison maker can build on this insight to make more accurate predictions. We illustrate this schematically in Fig. D-8.

Prediction of Future Values

In the following we will show how we use our algorithm to predict future values.



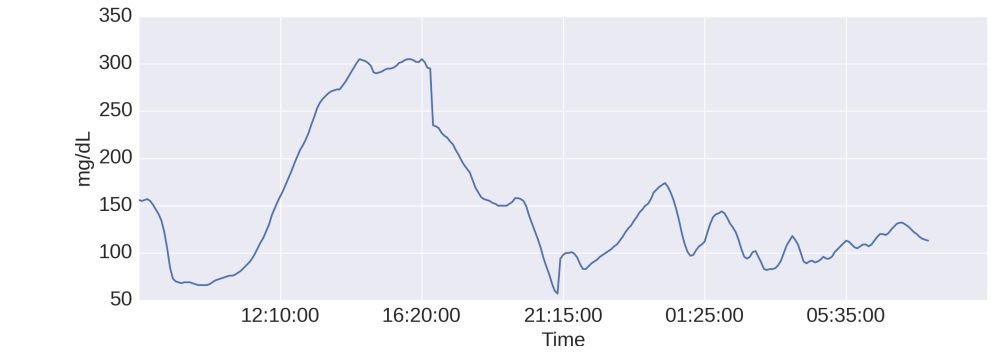
(a)



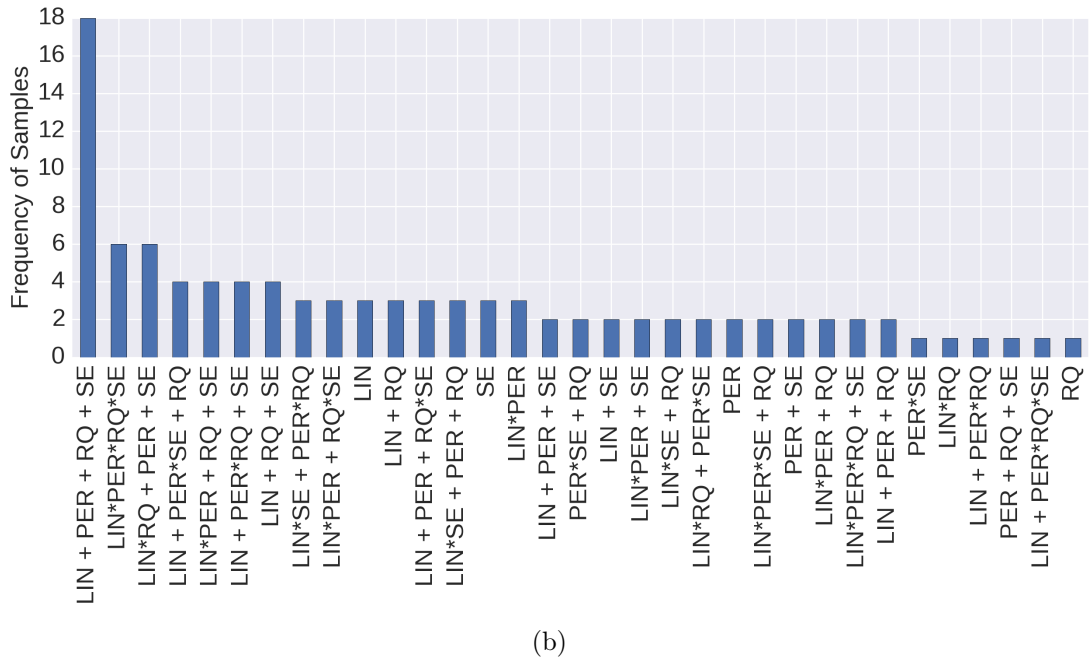
(b)

Figure D-5: (a) Glucose data of patient 3. (b) 100 samples drawn from the posterior distribution on kernel structure.

We would like to emphasise that our approach does not need the amount of insights of an expert, as it has been required by previous work, for example see Sparacino et al. (2007). We elaborate why the prediction of future values is useful and adequate. Fig. D-9 illustrates the predictive performance of our software on patient 3. We have selected patient 3 because this patient's curve includes a steep decrease in blood glucose towards the end of the measurements. Such a decrease is what we want to predict, taking previously computed values as training data to avoid Hypoglycemia. Moreover, our algorithm comes with an online component as well. Our agent observes all the new data that comes in. These observations are registered in its memory without training



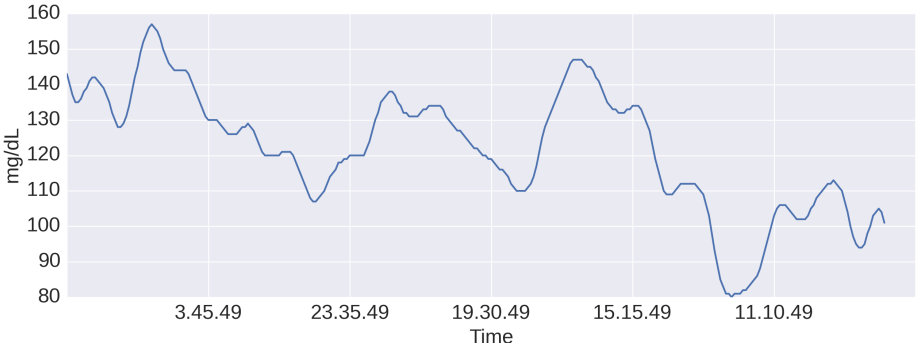
(a)



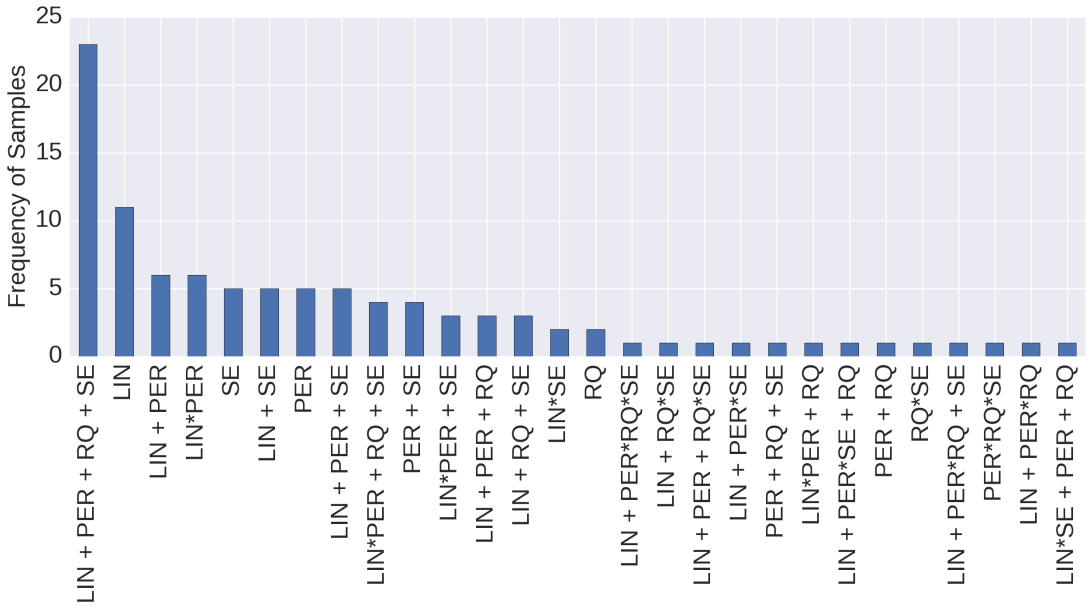
(b)

Figure D-6: (a) Glucose data of patient 4. (b) 100 samples drawn from the posterior distribution on kernel structure.

any kind of learning algorithm. We can predict 5, 10, 15, 25 and 30 minutes ahead (the full details are provided in Fig. D-9). Since we apply a fully Bayesian method, we are also supplied with a confidence interval for every prediction.

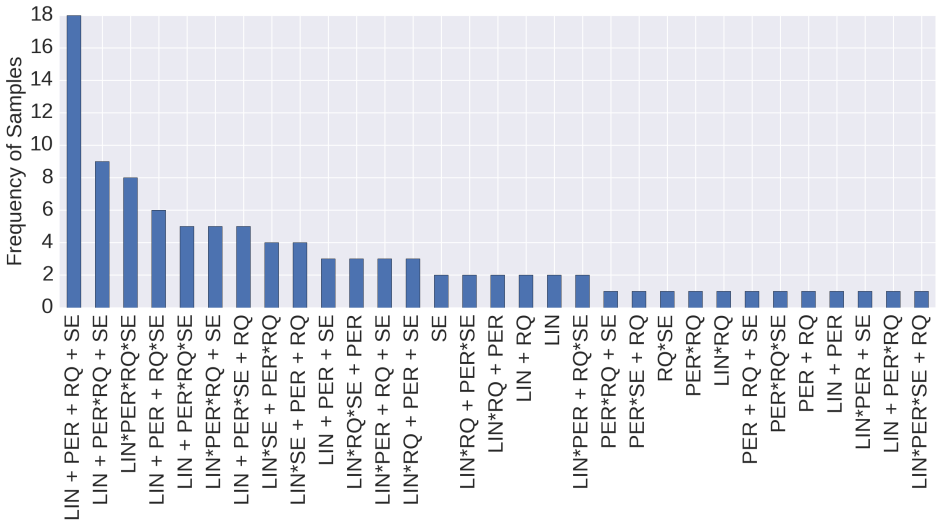


(a)

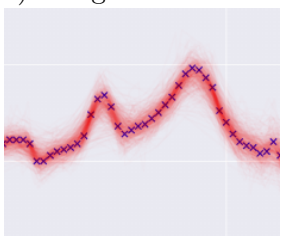


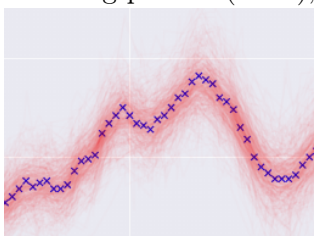
(b)

Figure D-7: (a) Glucose data of patient 5. (b) 100 samples drawn from the posterior distribution on kernel structure.



$P(PER)$ is high. We are confident that there are recurring patters (PER), e.g:





and

We can now make prediction, given previous observations

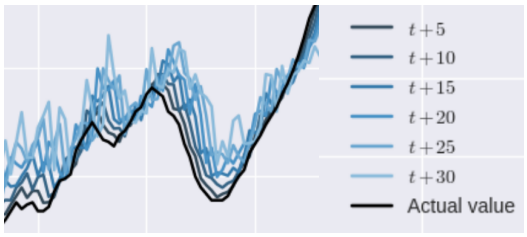
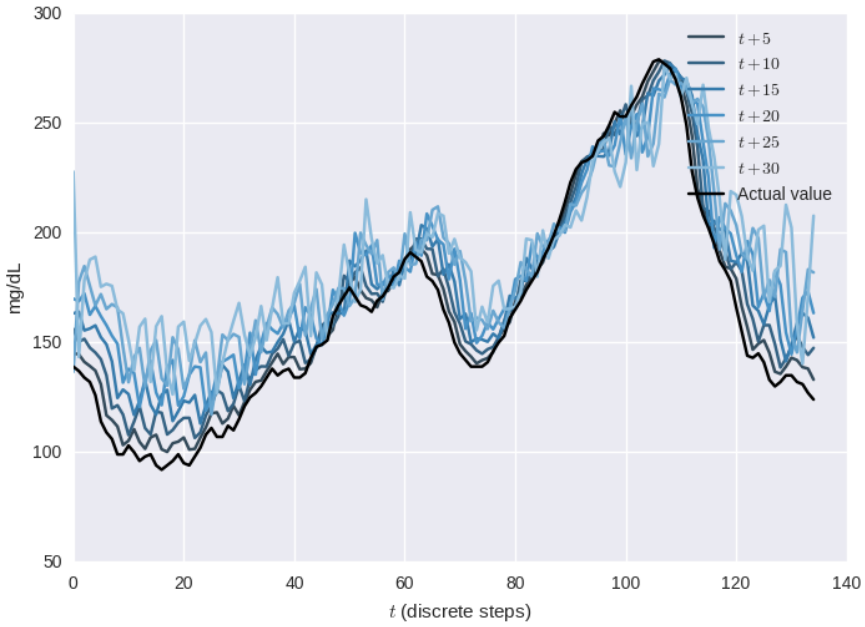
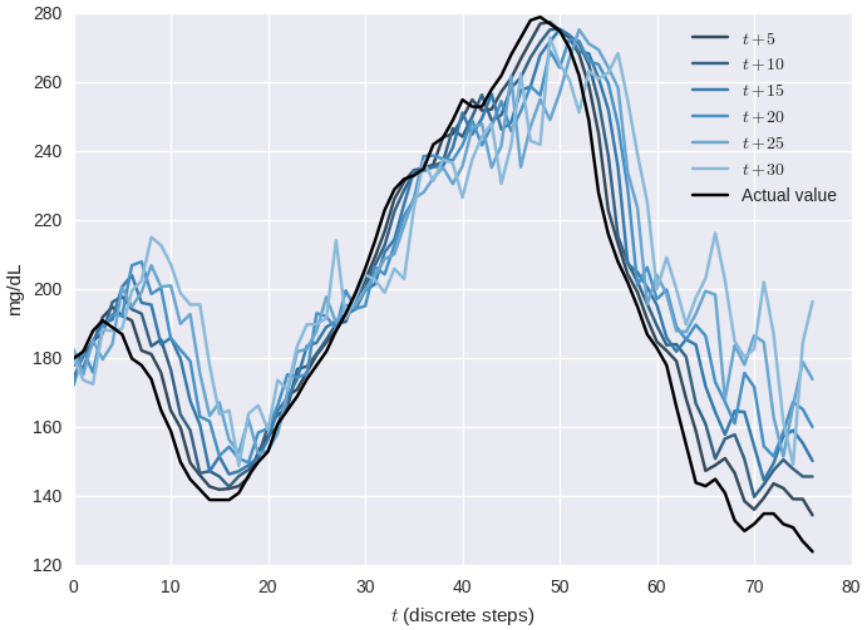


Figure D-8: We see how a posterior on high level structure for prediction can inform the decision making process. On top, we see a posterior distribution on structure, modelled with samples. We see that it is very likely that a pattern is recurring. With this information, we can predict ahead of time.



(a)



(b)

Figure D-9: Glucose prediction ahead of time, up to 30 minutes. We took (a) 50 % and (b) 70 % of the the data for patient 3 as training data to learn both the structure and the hyper-parameters.

List of Figures

List of Tables

Bibliography

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2014). Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43.

Andrieu, C., Djurić, P. M., and Doucet, A. (2001). Model selection by mcmc computation. *Signal Processing*, 81(1):19–37.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Bader, B. W., Kolda, T. G., et al. (2012). Matlab tensor toolbox version 2.5. Available online.

Barnett, L., Barrett, A. B., and Seth, A. K. (2009). Granger causality and transfer entropy are equivalent for gaussian variables. *Physical Review Letters*, 103(23):238701.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer New York.

Bollen, K. A. (1989). Structural equations with latent variables.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1997). *Time series analysis: forecasting and control*.

Buntine, W. L. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research (JAIR)*, 2:159–225.

Chen, Z., Zhang, K., Chan, L., and Schölkopf, B. (2014). Causal discovery via reproducing kernel hilbert space embeddings. *Neural Computation*, pages 1–34.

Choi, J. and Amir, E. (2012). Lifted relational variational inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Corvallis: AUAI Press.

Choi, J., Amir, E., and Hill, D. J. (2010). Lifted inference for relational continuous models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 51, page 61801. Corvallis: AUAI Press.

Choi, J., Guzman-Rivera, A., and Amir, E. (2011). Lifted relational kalman filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2092–2099. AAAI Press.

Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley.

Claassen, T. (2013). *Causal discovery and logic*. PhD thesis, Radboud University Nijmegen.

Claassen, T. and Heskes, T. (2011). A logical characterization of constraint-based causal discovery. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Corvallis: AUAI Press.

Claassen, T. and Heskes, T. (2012). A bayesian approach to constraint based causal inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Corvallis: AUAI Press.

Claassen, T., Mooij, J. M., and Heskes, T. (2013). Learning sparse causal models is not np-hard. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Corvallis: AUAI Press.

Clifton, L., Clifton, D. A., Pimentel, M. A. F., Watkinson, P. J., and Tarassenko, L. (2013). Gaussian processes for personalized e-health monitoring with wearable sensors. *Biomedical Engineering, IEEE Transactions on*, 60(1):193–197.

Cook, S., Conrad, C., Fowlkes, A. L., and Mohebbi, M. H. (2011). Assessing google flu trends performance in the united states during the 2009 influenza virus a (h1n1) pandemic. *PloS one*, 6(8):e23610.

Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13.

Daniušis, P., Janzing, D., Mooij, J., Zscheischler, J., Steudel, B., Zhang, K., and Schölkopf, B. (2010). Inferring deterministic causal relations. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342.

De Raedt, L., Kimmig, A., and Toivonen, H. (2007). Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2468–2473.

Ding, M., Chen, Y., and Bressler, S. L. (2006). 17 granger causality: Basic theory and application to neuroscience. *Handbook of time series analysis*, page 437.

Doan, A., Ramakrishnan, R., and Halevy, A. Y. (2011). Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, 195(2):216–222.

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1166–1174.

Entner, D. and Hoyer, P. O. (2010). On causal discovery from time series data using FCI. *Probabilistic Graphical Models*.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository. Available online.

Garcez, A. S., Lamb, L. C., and Gabbay, D. M. (2008). *Neural-symbolic cognitive reasoning*. Springer Science & Business Media.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

Ginsberg, J., Mohebbi, M. H., Patel, R. S., Brammer, L., Smolinski, M. S., and Brilliant, L. (2009). Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014.

Goodman, N. D .and Mansinghka, V. K., Roy, D., Bonawitz, K., and Tenenbaum, J. (2008). Church: A language for generative models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 220–229.

Granger, C. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438.

Horn, A. (1951). On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(01):14–21.

Hoyer, P. O., Janzing, D., Mooij, J., Peters, J., and Schölkopf, B. (2008a). Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 689–696.

Hoyer, P. O., Shimizu, S., Kerminen, A. J., and Palviainen, M. (2008b). Estimation of causal effects using linear non-gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 49(2):362–378.

Huberty, M. (2014). I expected a model t, but instead i got a loom: Awaiting the second big data revolution. In *BRIE-ETLA Conference*.

Hyttinen, A., Eberhardt, F., and Hoyer, P. O. (2013). Experiment selection for causal discovery. *The Journal of Machine Learning Research*, 14(1):3041–3071.

Hyvärinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *Neural Networks*, 10(3):626–634.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430.

Hyvärinen, A., Shimizu, S., and Hoyer, P. O. (2008). Causal modelling combining instantaneous and lagged effects: an identifiable model based on non-gaussianity. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 424–431.

Hyvärinen, A. and Smith, S. M. (2013). Pairwise likelihood ratios for estimation of non-gaussian structural equation models. *The Journal of Machine Learning Research*, 14(1):111–152.

Hyvärinen, A., Zhang, K., Shimizu, S., and Hoyer, P. O. (2010). Estimation of a structural vector autoregression model using non-gaussianity. *The Journal of Machine Learning Research*, 11:1709–1731.

Janzing, D., Mooij, J., Zhang, K., Lemeire, J., Zscheischler, J., Daniušis, P., Steudel, B., and Schölkopf, B. (2012). Information-geometric approach to inferring causal directions. *Artificial Intelligence*, 182:1–31.

Kafalı, Ö., Bromuri, S., Sindlar, M., van der Weide, T., Aguilar Pelaez, E., Schaechtle, U., Alves, B., Zufferey, D., Rodriguez-Villegas, E., Schumacher, M. I., and Stathis, K. (2013). Commodity$_{12}$: A smart e-health environment for diabetes management. *Journal of Ambient Intelligence and Smart Environments, IOS Press*.

Kawahara, Y., Shimizu, S., and Washio, T. (2011). Analyzing relationships among arma processes based on non-gaussianity of external influences. *Neurocomputing*, 74(12):2212–2221.

Kemmler, M., Rodner, E., Wacker, E., and Denzler, J. (2013). One-class classification with gaussian processes. *Pattern Recognition*, 46(12):3507–3518.

Knobbe, E. J. and Buckingham, B. (2005). The extended kalman filter for continuous glucose monitoring. *Diabetes technology & therapeutics*, 7(1):15–27.

Kolda, T. and Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Körding, K. P., Beierholm, U., Ma, W. J., Quartz, S., Tenenbaum, J. B., and Shams, L. (2007). Causal inference in multisensory perception. *PLoS ONE*, 2(9):e943.

Kpotufe, S., Sgouritsa, E., Janzing, D., and Schölkopf, B. (2014). Consistency of causal inference under the additive noise model. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kruschke, J. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.

Lacerda, G., Spirtes, P., Ramsey, J., and Hoyer, P. O. (2008). Discovering cyclic causal models by independent components analysis. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 366–374. Corvallis: AUAI Press.

Lavrac, N. and Dzeroski, S. (1994). Inductive logic programming. In *WLP*, pages 146–160. Springer.

Lázaro-Gredilla, M. and Titsias, M. (2011). Variational heteroscedastic gaussian process regression. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 841–848.

Lazer, D., Kennedy, R., King, G., and Vespignani, A. (2014). The parable of google flu: Traps in big data analysis. *Science*, 343(6176):1203–1205.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2014). Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Lynch, C. (2008). Big data: How do your data grow? *Nature*, 455(7209):28–29.

MacKay, D. J. C. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Mansinghka, V. K., Selsam, D., and Perov, Y. (2014). Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., and Barton, D. (2012). Big data. *The management revolution. Harvard Business Review*, 90(10):61–67.

McAllester, D., Milch, B., and Goodman, N. D. (2008). Random-world semantics and syntactic independence for expressive languages. Technical report.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Michigan Center for Public Health Preparedness (2015). Definition: Confounding in epiccentral: One-stop resource for epidemiology definitions, examples and activities.

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D., and Kolobov, A. (2007). BLOG: Probabilistic models with unknown objects. *Statistical relational learning*, page 373.

Minka, T., Winn, J., Guiver, J., and Knowles, D. (2010). Infer.net 2.4. microsoft research cambridge.

Mooij, J. M., Peters, J., Janzing, D., Zscheischler, J., and Schölkopf, B. (2014). Distinguishing cause from effect using observational data: methods and benchmarks. *arXiv preprint arXiv:1412.3773*.

Mooij, J. M., Stegle, O., Janzing, D., Zhang, K., and Schölkopf, B. (2010). Probabilistic latent variable models for distinguishing between cause and effect. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1687–1695.

Murphy, K. P. (2002). *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. The MIT Press, Cambridge, MA.

National Institute of Drug Abuse (2015). What is comorbidity?

Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression and classification. *arXiv preprint physics/9701026*.

Palerm, C. C. and Bequette, B. W. (2007). Hypoglycemia detection and prediction using continuous glucose monitoring—a study on hypoglycemic clamp data. *J Diabetes Sci Technol*, 1(5):624–629.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, 82(4):669–688.

Pearl, J. (2000). *Causality: models, reasoning, and inference*, volume 47. Cambridge University Press.

Pearl, J. (2001). Bayesianism and causality, or, why i am only a half-bayesian. In *Foundations of Bayesianism*, pages 19–36. Springer.

Pearl, J. (2012). The do-calculus revisited. *arXiv preprint arXiv:1210.4852*.

Pearl, J., Verma, T., et al. (1991). A theory of inferred causation. In *Second International Conference on teh Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann San Mateo, CA.

Peters, J. and Bühlmann, P. (2014). Identifiability of gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228.

Peters, J., Janzing, D., and Schoelkopf, B. (2012). Causal inference on time series using structural equation models. *arXiv preprint arXiv:1207.5136*.

Peters, J., Mooij, J., Janzing, D., and Schölkop, B. (2011). Identifiability of causal graphs using functional models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Corvallis: AUAI Press.

Peters, J., Mooij, J., Janzing, D., and Schölkopf, B. (2014). Causal discovery with continuous additive noise models. *The Journal of Machine Learning Research*, 15(1):2009–2053.

Poole, D. (1993). Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81–129.

Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56.

Poole, D. (2008). The independent choice logic and beyond. In *Probabilistic inductive logic programming*, pages 222–243. Springer-Verlag.

Popper, K. (1959). *The logic of scientific discovery*. Hutchinson.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Renkens, J., Shterionov, D., Van den Broeck, G., Vlasselaer, J., Fierens, D., Meert, W., Janssens, G., and De Raedt, L. (2012). Problog2: From probabilistic programming to statistical relational learning. In *Proceedings of the NIPS Probabilistic Programming Workshop*.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1):107–136.

Rogers, M., Li, L., and Russell, S. J. (2013). Multilinear dynamical systems for tensor time series. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2634–2642.

Roy, D. (2015). Probabilistic-programming.org.

Russell, S. (2015). Unifying logic and probability. *Communications of the ACM*, 58(7):88–97.

Russell, S. and Norvig, P. (2010). *Artificial intelligence: a modern approach*. Prentice hall.

Sackett, D. L., Rosenberg, W. M., Gray, J. A., Haynes, R. B., and Richardson, W. S. (1996). Evidence based medicine: what it is and what it isn't. *BMJ: British Medical Journal*, 312(7023):71.

Saeed, M., Lieu, C., Raber, G., and Mark, R. G. (2002). MIMIC ii: a massive temporal icu patient database to support research in intelligent patient monitoring. In *Computers in Cardiology, 2002*, pages 641–644. IEEE.

Santos, J. C. A., Nassif, H., Page, D., Muggleton, S. H., and Sternberg, M. J. E. (2012). Automated identification of protein-ligand interaction features using inductive logic programming: A hexose binding case study. *BMC bioinformatics*, 13(1):162.

Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In *In Proceedings of the International Conference on Logic Programming*. Citeseer.

Sato, T. and Kameya, Y. (1997). Prism: a language for symbolic-statistical modeling. In *Proceedings of the Internation Joint Conference on Artificial Intelligence (IJCAI)*, volume 97, pages 1330–1339.

Schaechtle, U., Stathis, K., and Bromuri, S. (2013). Multi-dimensional causal discovery. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1649–1655. AAAI Press.

Schreiber, T. (2000). Measuring information transfer. *Physical Review Letters*, 85(2):461–464.

Seth, A. K. (2010). A matlab toolbox for granger causal connectivity analysis. *Journal of Neuroscience Methods*, 186(2):262.

Sgouritsa, E., Janzing, D., Hennig, P., and Schölkopf, B. (2015). Inference of cause and effect with unsupervised inverse regression. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 847–855.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*.

Shepherd, T. and Owenius, R. (2012). Gaussian process models of dynamic pet for functional volume definition in radiation oncology. *Medical Imaging, IEEE Transactions on*, 31(8):1542–1556.

Shimizu, S. (2014). Lingam: Non-gaussian methods for estimating causal structures. *Behaviormetrika*, 41(1):65–98.

Shimizu, S. and Bollen, K. (2014). Bayesian estimation of causal direction in acyclic structural equation models with individual-specific confounder variables and non-gaussian distributions. *The Journal of Machine Learning Research*, 15(1):2629–2652.

Shimizu, S., Hoyer, P. O., Hyvärinen, A., and Kerminen, A. (2006). A linear non-gaussian acyclic model for causal discovery. *The Journal of Machine Learning Research*, 7:2003–2030.

Shimizu, S., Hyvärinen, A., Kano, Y., and Hoyer, P. O. (2005). Discovery of non-gaussian linear causal models using ICA. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 526–533. Corvallis: AUAI Press.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2951–2959.

Sparacino, G., Zanderigo, F., Corazza, S., Maran, A., Facchinetti, A., and Cobelli, C. (2007). Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series. *Biomedical Engineering, IEEE Transactions on*, 54(5):931–937.

Spirtes, P. (2001). An anytime algorithm for causal inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 213–231.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*, volume 81.

Spirtes, P., Glymour, C. N., Scheines, R., and Tillman, R. (2010). Automated search for causal relations: Theory and practice. Technical report, Department of Philosophy, Carnegie Mellon University.

Swan, M. (2013). The quantified self: Fundamental disruption in big data science and biological discovery. *Big Data*, 1(2):85–99.

Tashiro, T., Shimizu, S., Hyvärinen, A., and Washio, T. (2014). Parcelingam: A causal ordering method robust against latent confounders. *Neural computation*, 26(1):57–83.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285.

The Free Dictionary: Diagnosis (2015). The free dictionary.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294.

Tillman, R., Gretton, A., and Spirtes, P. (2009). Nonlinear directed acyclic structure learning with weakly additive noise models. 22:1847–1855.

Turksoy, K., Bayrak, E. S., Quinn, L., Littlejohn, E., Rollins, D. K., and Cinar, A. (2013). Hypoglycemia early alarm systems based on multivariable models. *Industrial & Engineering Chemistry Research*.

Vasilescu, M. A. O. and Terzopoulos, D. (2005). Multilinear independent components analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 547–553. IEEE.

Wall, M. E., Rechtsteiner, A., and Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer.

Zhang, K. and Hyvärinen, A. (2009). On the identifiability of the post-nonlinear causal model. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 647–655. Corvallis: AUAI Press.

Zinberg, B. (2015). Bayesian optimization as a probabilistic meta-program. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.