Structural learning for continuous data using graphical models

Benjamin Szili

A thesis submitted to the

School of Mathematics and Statistics

University of Glasgow

for the degree of

Doctor of Philosophy

September 2022

Abstract

The field of statistics, and science as a whole has continuously been improving the study of increasingly complex structures, not only focusing on their individual components but also how they interaction with each other and their dependencies. This thesis is centered around learning the structure of data using graphical models.

Graphical models allow the visual representation of dependence relations between a set of random variables through graphs, specified by some type of model formula. Whether the goal is probabilistic inference, mainly dealing with belief propagation, or causal inference, focusing on interventions, it is very important to be able to learn the underlying structure given a set of variables.

In graph theory, this can be achieved by applying structural learning algorithms, either based on constraints or some scoring function. While there are a number of such algorithms that have been used extensively and are effective, they are somewhat limited by the type of relationships they can learn.

Current methods excel at learning a network structure when the variables of interest are discrete, or if continuous, they are Gaussian. The main objective of this thesis is therefore to create a structural learning algorithm that is able to establish dependence relations that are not between discrete or Gaussian variables.

Initially relevant literature and key methodology on graphical models, kernel methods and information theory were reviewed. The aim was to use a measurement that can reliably detect pairwise and conditional dependencies between random variables that are not necessarily Gaussian. The main contribution of the thesis is then the incorporation of kernel methods and Mutual Information to create new structural learning algorithm using graphs to visualize the learned structure.

The resulting algorithm with three variants is then applied in a number of settings. First, a simulation setup using the post non-linear noise model on synthetic data was examined to compare performance of the new algorithm to a current approach. As the structure was known from the setup, the focus in this context was to see whether the algorithm successfully

learns the structure.

The remaining two settings then present cases where the new algorithm can be applied to provide insight and improve inference by accounting for the dependence structure. One of these settings is focusing on distinguishing handwritten digits, initially using Gaussian Process latent variable models (GP-LVMs). The second setting then applies the algorithm in to the field of phonetics, the task focusing on identifying speakers based on sound data.

Declaration

I, Benjamin Szili, hereby declare that this thesis titled 'Structural learning for continuous data using graphical models' and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly attributed.

Acknowledgements

I would like to take this opportunity and thank my supervisors, Mu Niu and Tereza Neocleous. Their guidance and support has been invaluable and I am grateful to have worked with them.

I would also like to thank Ludger Evers, Manuele Leonelli and Theodore Papamarkou, who have also been part of the supervising team at one point or another. I am thankful for each of their input.

I am grateful for the support provided by from the School of Mathematics and Statistics, especially during the pandemic.

Finally, I would like to thank my family and friends for encouraging me through every step of the way, even at the most difficult times.

Contents

List of Figures

Chapter 1

1 Introduction

1.1 Graphical models and structural learning

Graphical models allow the visual representation of dependence relations between a set of random variables through graphs. They provide an output (in the form of graphs) that are visually intuitive and are often much easier to interpret than other typical output. The potential to represent even more complex underlying structure within a dataset makes them and ideal visualization tool even during exploratory analysis. The nodes of a graph are associated with the variables – which can be discrete or continuous –, while the edges encode the relationships between them. A key part to show these relationships is the Markov property of graphical models.

The graphs themselves are specified by some type of model formula. While this can be as simple as a list of nodes an edges, it can also be the factorization of the joint probability density of the random variables. In some cases, these formulae and the associated graphs can then be compared as a method of model selection.

Probabilistic graphical models (PGMs) (Koller and Friedman, 2009) are important in most learning problems, becoming a common method for modeling uncertainty in several areas. such as computer vision, speech processing, bioinformatics, signal processing, communications and the area of artificial intelligence in general (Pernkopf et al, 2014). They provide a straightforward visual representation of joint probability distributions, exploiting conditional dependence relations between random variables of interest, tackling both uncertainty and complexity (Jordan, 1999).

A subset of graphical models, Bayesian Networks can additionally factorize the global probability distribution of a set of variables as a product of conditional probability distributions. This makes Bayesian Networks a helpful tool for inference, as the conditional dependencies are visualized through the graphs as well as described by the product.

Whether the inference is probabilistic, mainly dealing with belief propagation, or causal in-

ference, focusing on interventions, it is very important to be able to learn the underlying structure given a set of variables. In graph theory, this can be achieved by applying structural learning algorithms. Typically, these algorithm either use a set of constraints - such as tests of pairwise and conditional independence - or by evaluating the structure based on a scoring function, which is then optimized. These algorithms then have many different variations, mainly based on what constraints or scoring functions they use. Some of these variants are very effective at learning the underlying structure of a set of random variables, excelling at learning a network structure when the variables of interest are discrete, or Gaussian, but they are somewhat limited by the type different structures. It then became of interest whether it is possible to identify a set of constraints, or a scoring function that performs well where previous learning algorithms may not.

The main objective of this thesis is therefore to create a structural learning algorithm that is able to establish dependence relations that are not between discrete or Gaussian variables, and to dependencies that are not necessarily between the means. In order to find an alternative to current methodology – while avoiding potential loss of information from discretization–, it is important to note the steps that need to be taken in order to construct a structural learning algorithm, identifying the necessary building blocks to do so.

First, it is required to find a measure that can find such relationships between variables in a cost-effective and accurate manner. Second, once such a metric is found, researched and tested, the next step is to extend it to learn conditional dependencies such that it is still effective. Then, it should be possible to include in a structural learning algorithm, either as a score in a score based, or as part of a test in a constraint based algorithm.

1.2 Outline

Section 2 lays out the topics relevant to the thesis. This covers an introduction to graph theory, describing key definitions, including the Markov property and how it can be used to represent independence. Followed by these are basics on Bayesian Networks, how they can be useful for inference, as well as the idea of equivalence classes, which often leads to difficulties during causal inference. Some details on probabilistic and causal inference are given, which underlies the need for structural learning. Two common structural learning algorithms are described, demonstrating the main approaches to learn overall structure – either through constraints, such as independence tests or through scoring functions using

optimization techniques – as well as the key steps to do so. These elements will be a part of either a **pairwise phase** or a **conditional phase**.

As the following step, some background on kernels is covered. Some key properties of kernels are introduced, along with a few examples for their application on learning problems. Then, after describing the kernel trick and mentioning a few common kernels, dot products are then extended to a more general level. Both the case of defining a kernel or the mapping first were examined, along with some implications. The concept of Reproducing Kernel Hilbert Spaces is introduced, which has a key role in finding kernel methods that can measure independence as a formal test

Afterwards, another approach is considered to examine the network structure, with the introduction of Mutual Information. Therefore, some of the basics of Information Theory are reviewed. This includes a brief description of information as a measurable quantity, leading to the laws of information. Using the average of information, entropy is introduced, which is a key element to define Mutual Information and its potential in measuring independence. After the initial description for discrete random variables, the extension to continuous variables is also presented. An appealing feature is that while potentially difficult to calculate, the framework can be used for both the pairwise and the conditional case.

Section 3 then continues on to review key methodology based on these previous topics. First, to establish pairwise independence, a plausible approach is the use of kernel methods. Therefore, an overview on Kernel Covariance is covered in terms of measuring independence, potentially even for non-Gaussian continuous settings.

Then, some later variants of the Kernel Covariance are examined and their capacity as a tool to measure independence is established. One such version using the Hilbert-Schimdt Independence Criterion, is introduced, capable of testing independence between non-Gaussian random variables. This results in a powerful potential tool to use, specifically during the pairwise phase.

While using kernel methods may allow one to learn the pairwise dependence structure, and there have been great strides taken to use these methods to establish conditional independence, using such methods appear to be overly complex and computationally expensive. It

is therefore of interest to examine whether there are other, more straightforward ways to achieve this.

Returning to information theory, it was established that for continuous random variables, it is often not plausible to calculate Mutual Information due to the complex form of the differential entropies necessary. Nevertheless, as it can normally be extended to examine conditional and not strictly linear dependencies between continuous random variables, it is of interest to see if there is a way to use it. Therefore, a way to estimate Mutual Information for continuous variables is described, when the exact measurement is not readily obtainable. The approximation of entropy using the Kozachenko-Leonenko estimator is first described and then is extended to obtain estimates for Mutual Information, as well as Conditional Mutual Information through the nearest neighbour method.

Section 4 gathers all previous building blocks in order to create a structural learning algorithm with a pairwise and conditional phase. It describes the choice of using either the estimated Mutual Information, or independence tests based on the Hilbert-Schmidt Independence Criterion for the pairwise phase. For Mutual Information, this includes choosing the parameters of the approximation and the approach to interpret the obtained estimates. This led on to two different variants within the pairwise phase, one based on ranking, the other on a threshold. The potential effect of different values of K on the approximation is observed, informing the choice for a threshold. Overall, this provided three options during the pairwise phase. Then, for all the three options, during the conditional phase, the approximations of conditional mutual information are used to account for the conditional dependencies. Finally, a visual representation of these dependencies are returned as part of the output of the algorithm. A short illustration was also added for the two main directions – using ranking, or some way to account for all relations during the pairwise phase.

Section 5 then describes the main testing ground for the versions of the algorithm using Mutual Information estimates in the pairwise phase using simulated data. The simulation setting enabled the enforcing of the underlying structure through different link functions between random variables. Once a sample was generated, the variants of the algorithm were applied using different combinations for the link functions, observing how well the method recognizes the relationships within these enforced structures. As a point of comparison, one of the most commonly using current structural learning algorithms, the PC-algorithm was also applied for these settings in order to compare performance, illustrated through several

examples. As a further step, some of the examples were then repeated using different seeds, and the frequencies of specific assigned edges were recorded and compared. Overall, this led to mostly abandoning the ranking method through the pairwise phase, with potential exception to the case of having high-dimensional data and the interest in identifying only the most relevant dependencies. Part of this research was presented and published at the 4th International Conference on Statistics (Szili et al, 2022),a 5 page paper consisting of the fundamental theory and an early variant of the algorithm with a few toy examples.

Section 6 then moved on to apply the algorithm to the Chars74K dataset (De Campos et al., 2009). Using 16425 Kannada characters generated by 25 volunteers, it the dataset consists of 55 scaled images of handwritten digits, each containing 256 pixels. The main goal was to apply the algorithm in a manner that will assist in distinguishing handwritten digits. After describing the data, a brief overview is given on Gaussian Processes (GPs) as well as GP-Latent Variable Models (GP-LVM). After a GP-LVM is applied to the data with 5 and 10 input variables respectively, the new method was applied to the resulting latent variables. Then, using insight gained from the network structure, a few of the random variables were selected to assess how well they separate the handwritten digits. In addition, model performance was also assessed using silhouette plots. The model diagnostics, such as average silhouette widths suggested that the GP-LVMs did not perform well in general. However, visualizing output using the dimensions that were selected by examining the learned network structure and the underlying links, the digits were still well separated.

Section 7 describes the application of the algorithm on a dataset containing Vowel sounds. After a brief introduction to phonetics, a common approach to comparing evidence in forensic science is described, as well as some background on factorizing decomposable models. Then, the data used is described, followed by examining the network structure for a sample of 25 speakers. Next the joint density was factorized using a triangulated graph based on the learned structure. Then, the strength of evidence using the learned structure is compared to alternative approaches through an example, when all variables were assumed to be independent, and when the keywords were considered independent of each other.

Finally, Section 8 then concludes the thesis, summarizing the work done.

Chapter 2

2 Background

2.1 Introduction to relevant graph theory

2.1.1 Basic background on graph theory

Graphical models are a type of models which use graphs to visualize the structure of random variables, reflecting the dependence relations between them. They also a form of multivariate analysis, with origins linked to fields such as path analysis (Wright, 1921) and statistical physics (Gibbs,1902), ultimately combining probability and graph theory. The graphs themselves are a structure $G = (V, E)$ containing a set of nodes V (or vertices) and a a set of edges E (or arcs) connecting them. Two nodes are called *adjacent* if an edge connects them, and if a graph contains an edge between each pair of nodes, it is a *complete* graph. Even when not complete, one can distinguish *connected* graphs, when there is a path between every pair of vertices.

For graph G, one can define any subset of it as a *subgraph*, which contain all the nodes in the subset as well as every edge that connect them. The definition of a complete graph can then also be extended for any subgraph if all nodes are connected to each other by an edge. In addition, if a subset is maximally complete, it defines a *clique*, an important element in graphical modelling. Furthermore, a node or a subgraph s can *separate* two nodes or subgraphs if all paths connecting the two travel through s. If a path ends at the node where it started, but otherwise only travels through a node once, it is called a chordless cycle. A graph with no such cycles with length greater than 3 are then *triangulated*

In graphical modelling, the nodes are assigned to random variables, which can be both discrete and continuous. Additionally, the edges indicate dependence relations, and therefore can visualize independence, where the joint density of two random variables X and Y can be factorized into the product of their respective marginal densities, that is

$$f_{X,Y}(x, y) = f_X(x) f_Y(y). \tag{2.1.1}$$

Similarly, the graphs also represent conditional independence, a key element in graphical

modeling. Given three random variables X, Y and Z, X and Y are conditionally independent *given* Z, if for each z, X and Y are independent when $Z = z$. Looking at a graph, one can begin interpreting relationships with the pairwise Markov property: if two nodes are not adjacent to each other, the associated random variables are conditionally independent given the remaining variables (Edwards, 2000). Globally speaking, any two subgraphs that are separated by a third signify conditional independence between the subsets, given the third set - called the global Markov property.

In order to create the associated graphs, model formulae are often used. Some Markov fields can be represented by their clique factorization to represent their joint distribution. For Bayesian networks, it is the factorization of the joint distribution into a product of conditional distributions (see Equation 2.1.2). In some discrete cases, such as a loglinear model, one can use the sum of the terms associated with interaction terms within the model. When the resulting graph is triangulated, the model is then decomposable (Darroch, Lauritzen and Speed, 1980). This is a desirable property, as it allows more efficient estimation of associated parameters. In addition, it simplifies the application of conditional tests to delete edges, and provides easier interpretation.

Often, graphical modelling is part of a model selection process, where adding or removing edges represent the different potential models. This can be done through stepwise selection, manipulating edges based on some criterion; through some search techniques seeking simple models that represent the data (Edwards and Havránek, 1987); or through an overall information criteria. Stepwise selection, typically backward elimination is common for constraint-based learning algorithms, while search techniques are often part of score-based learning algorithms. The model selection process then plays a key part in learning the overall structure.

2.1.2 Overview on Bayesian Networks and basics of inference

Graphical models can also be specified as a triplet $(\mathbf{X}, G, \mathcal{P})$, representing the probabilistic structure $\mathcal{P}$ between a set of random variables $\mathbf{X} = \{X_1, X_2, ..., X_p\}$ through a graph structure $G = (V, E)$, where each node $v_i \in V$ is associated with a random variable X_i and $e_{ij} \in E$ is the edge between node v_i, v_j (Nagarajan et al.,2013). A Bayesian network (BN), a subset of graphical models, uses annotated Directed Acyclic Graphs (DAGs) to do so, where edges are directed and none of these edges connect a node to itself - that is, there are no cycles.

The edges for these graphs are directed, shown by arrows, and therefore the paths go in one direction only. If a node has an edge pointed toward another, they are called the parent and child respectively. This can be generalized to a directed path between two nodes, originating from the ancestor node leading to the descendant node. This can then be extended to sets of nodes as well.

Next, it is important to observe how the Markov properties apply to DAGs. When a directed edge is assigned between two nodes, it is making use of the pairwise Markov property, representing pairwise dependence between adjacent nodes. Then, representing conditional independence relations should also be possible through the global Markov property, which is called *d-separation* for DAGs. For undirected graphs, when all paths between two nodes contain a set, it separates them and therefore the two are conditionally independent given the set. The case for DAGs is similar, but the path configuration is an important additional element. If a node has converging directed edges, it is called a *collider*. Then, a directed path between two nodes can be either active or blocked, active paths indicating dependence between the nodes. A path is blocked if it has a collider that it is not conditioned on, or a noncollider that it is conditioned on. Then, a node d-separates two other nodes if it blocks all paths between them (Verma and Pearl, 1990).

Bayesian networks have become a common representation tool for encoding and visualizing uncertainties in systems (Heckerman et al, 1995). A crucial result for these network is Bayes's theorem ($P(A, B) = P(A|B)P(B) = P(B|A)P(A)$). This enables thinking in the Bayesian framework, updating the probability, or belief regarding an event after observing some other event and is the foundation that leads to the form of Equation 2.1.2, as using Bayes' theorem gives a way to factorize joint distributions, which are then assigned a graph that represents the structure suggested by the factorization.

Graphs provide a visually straightforward way to encode structural information between variables into G, as the joint probability distribution $\mathcal{P}$ (or joint density function in the continuous case) of $\mathbf{X}$, include the probabilistic dependencies between the random variables. A key property of Bayesian networks, enabled by the idea of D-separation within DAGs and the Markov property (Pearl, 1988), is that the global probability distribution can be factorized as a product of conditional probability distributions (Koller and Friedman, 2009),

in the form

$$\mathcal{P} = P(\mathbf{X}) = \prod_{v \in V} P(X_v | X_{pa(v)}), \tag{2.1.2}$$

where $X_{pa(v)}$ are the parents of node v in the graph - indicated by directed edges from the parents to the node. This is also called recursive factorization. These conditional distributions are then often used to demonstrate conditional independence of some variables in the network. However, there are often more than one way to visualize information on conditional independence. The collection of these DAGs are called an equivalence class. The probabilistic structure is the same for graphs in an equivalent class - and therefore the factorization is the same as well -, the graph structures representing them can be different. As an example, Figure 2.1 shows a simple case of where three DAGs all show that Y and Z are conditionally independent given X, and Figure 2.2 a fourth DAG which is not equivalent. The difference between the structure of graphs in an equivalent class can then potentially affect inference and structural learning (Acid and de Campos, 2003), especially when trying to establish causality.

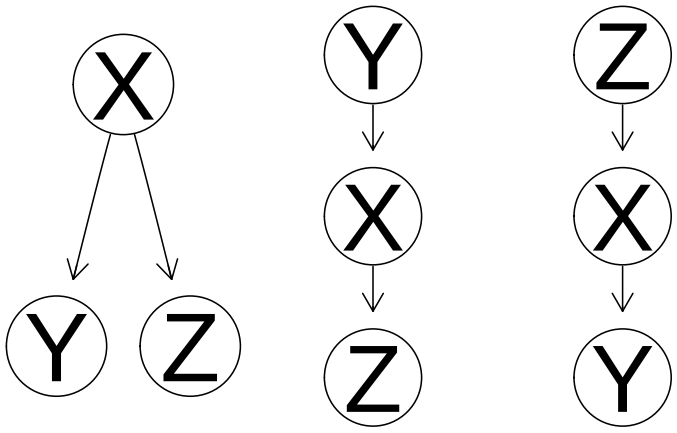


Figure 2.1: Example of three DAGs of an equivalent class. These DAGs, while having different visual interpretation, have the same factorization

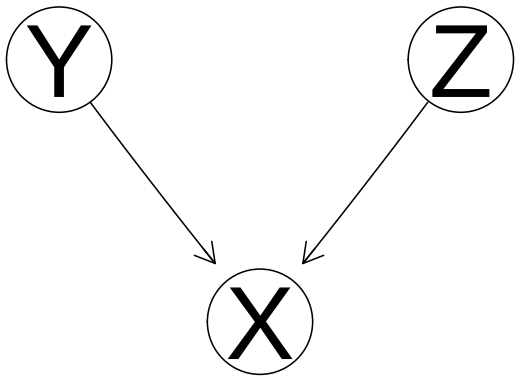


Figure 2.2: Example of DAG that is not equivalent to those in Figure 2.1,as this DAG cannot be factorized in the same form.

When the causal relationships implied by the direction of the edges in the graph are less certain, it is possible to instead use its *moral graph*, the equivalent undirected form of a DAG. Moral graphs can be obtained by removing all edge directions, and adding edges between the parents within the graph. For example, the moral graph of Figure 2.2 would have no directed edges, and an additional edge between Y and.

As a further note, it is also possible to use *chain graphs*, which have directed and undirected edges (Lauritzen and Wermuth, 1989). While DAGs encode independence relations between random variables, chain graphs typically assume symmetric associations (Lauritzen and Wermuth, 1989) and not causal associations (Andersson, Madigan and Perlman,). However, in these cases, the Markov properties of both directed and undirected graphs need to be considered and therefore are often difficult to interpret.

Inference using Bayesian networks can be divided into two main areas: probabilistic inference (or evidential reasoning), and causal inference, used to answer different types of questions. Probabilistic inference uses new evidence to compute posterior probabilities or densities once the network structure is available (Nagarajan et al.,2013). This can also be seen as computing some conditional probability distribution over some hidden (H) nodes, given evidence from other nodes (E)

$$P(H|E) = \frac{P(H,E)}{P(E)}.$$

The evidence could be new observations from one of the variables (hard evidence), or new

distribution for one of the variables (soft evidence). The process of using the evidence to update the rest of the BN is then called belief propagation, mostly used for discrete variables, although there has been work generalizing the process (Yedidia et al, 2005). Inference algorithms then make use the conditional relations inferred from missing edges (Jensen, 1996). Quite often, this is accompanied by observing the moral graph, and then creating a junction tree.

One of the most common methods using probabilistic inference involves the use of conditional probability queries, obtaining a marginal posterior probability distribution of a variable of interest. This is making use of two components of Bayesian Networks: the DAG encapsulating the structure of the probability distribution by visualizing conditional independence relations, and the set of conditional probability distributions (CPDs) to describe a variable given its parents (Pearl, 1988). The graphs are often shown together with conditional probability tables, which further describe the steps for the belief propagation process.

The first step usually involves learning the global structure, which are then represented by the edges of the graph (Lam and Bacchus, 1994). Then, examining the local structure, identifying and estimating associated parameters allows representation of the CPDs. In simpler cases, this can be done by calculating probability tables, showing the probabilities of values a node can take given the state of its parents, although the number of parameters can increase rapidly as soon as the effect of other ascendants are included as well. It is important to note that arc directions within the DAG of the Bayesian network do not necessarily imply causality, and therefore handling equivalence classes is less challenging, as the aim is more specific to the variables being updated given new evidence.

On the other hand, causal Bayesian networks assume that arc directions also imply causality. This allows to represent the way a system behave under *intervention*. This can be done by introducing

$$\Pr(Y = j|\mathrm{do}(X = i)), \tag{2.1.3}$$

the probability that $Y = j$ when X has been fixed to i by an external intervention – noted by the do(.) statement –, typically assuming that the intervention happens before conditioning. This means that given X, Y and Z, we have

$$\Pr(X = x|Y = y, \text{do}(Z = z) = \frac{\Pr(X = x, Y = y|\text{do}(Z = z))}{\Pr(Y = y|\text{do}(Z = z)}.$$

In addition, as the edge direction imply causality, it also affects how a network behaves under intervention. If there is a directed edge from X to Y, then fixing Y does not affect X

$$\Pr(X = i|\text{do}(Y = j)) = \Pr(X = i) \neq \Pr(X = i|Y = j)),$$

but an intervention on X does affect Y

$$\Pr(Y = j|\text{do}(X = i)) = \Pr(Y = j|X = i)) \neq \Pr(Y = j).$$

A common objective is then often to estimate the distribution of Y, a variable of interest, after the intervention $\text{do}(X = x)$. Using an estimate for the joint distribution $\hat{p}_{ij} = n_{ij}/N$ This would take form (Edwards, 1999)

$$\hat{\Pr}(Y = j|\text{do}(X = i) = \hat{\Pr}(Y = j|X = i) = \hat{p}_{ij}/\hat{p}_{i+}. \tag{2.1.4}$$

These estimations can become difficult, however, in case there is some unobserved influence of another variable, *confounding* the effect of the intervention, not necessarily as a cause of the variables of interest.

Implying causal relations also leads to an issue when it comes to handling equivalence classes, as different DAG visualizations might lead to completely different interpretations. However, including causality provides a more clear and meaningful view, and there are arguments supporting that even within an equivalence class, there might be an underlying BN that accounts for the correct causal structure (Pearl, 2009) – especially as regardless the graph representation, the probabilistic structure is the same for all the DAGs in the class. This approach additionally requires that each variable is conditionally independent of its non-effects, that the underlying dependence structure does exist, and that there are no latent variables that affect the variables in the model but are not observed.

Inference using these models are then focused on interventions, exploring the causal effect of manipulating a variable in the network on the posterior probabilities of interest. In some cases, this allows the calculation of the effect of an ideal intervention in a network by ma-

nipulating a variable, then modifying the graph structure to represent the intervention.

While probabilistic inference is not affected as much by different visualization of the factorization for a DAG as causal inference, both approaches require an appropriate network structure to infer posterior probabilities efficiently. Therefore, it is important to have structural learning tools that are able to successfully detect dependencies from observed data of the random variables of interest – in the case of probabilistic Bayesian networks, at least up to equivalence class. The next section briefly provides some background on structural learning algorithms, and introduces two of the common algorithms currently used.

2.1.3 Review on network structural learning for Bayesian Networks

Structural learning is part of the process of fitting a Bayesian Network (Nagarajan et al, 2013) – representing probabilistic dependencies between a set of random variables as a DAG. Essentially, it is a model selection process to find a graph structure that represents the set of dependence relations for the set of variables on a pairwise and conditional level. Used for gaining information on dependencies in the domain, or as a starting phase for future inference – such as classification methods–, the two main types of structural learning algorithms reflect the ways the network structure is obtained. These are known as constraint-based and score-based learning algorithms (and their combination known as hybrid algorithms).

Constraint-based learning algorithms – based on the Inductive Causation algorithm by Verma and Pearl (1991) – create structure based on some constraints that examine dependence relations. Typically, this starts with performing pairwise independence tests between each pair of random variable, represented by edges between them. These are followed conditional independence tests, where some edges are removed based on the results. Then, the remaining arcs are often assigned directionality through some orientation rule. The most well known practical algorithm is the PC algorithm (Spirtes et al, 1991), a later variant of the Inductive Causation algorithm, which has since been often used as a basis to come up with newer constraint-based algorithms. One of the main concerns of using constraint-based algorithms is high computation cost induced by the number of conditional independence tests. Another common issue is that the output may only be up to equivalence class - when the graph obtained from the conditional dependencies can be visualized in more than one way. This could lead to different interpretation of the relationship between the random variables, even though their factorization is the same. While one of the advantages of using graphical mod-

els in general is having a clear, intuitive representation, the different forms presented by graphs within an equivalence class would be associated with different meanings. This can be especially crucial for causal networks, where the direction of an arc within the Bayesian network implies a causal relationship, and can be potentially avoided using moral graphs

Instead of independence tests, score-based learning algorithms use optimization techniques. These assign some score to each potential network structure, which can be achieved through a range of scoring functions. A commonly used example is the Hill-Climbing algorithm (described in Section 2.1.3.), which is one of the greedy search algorithms, adding, deleting and reverting network arcs until the assigned score can no longer be improved. When using score-based algorithms, one must take note that the choice of starting point and the order of fitted arcs may result in a local maximum score, as well as the impact of the scoring function of choice.

2.1.4 Constraint-based learning algorithm

Originating from the Inductive Causation algorithm (Verman and Pearl, 1991), the most common constraint-based learning algorithm is some variant of the PC algorithm (Spirtes et al, 1993).The main phases are described below.

Starting out with a complete graph, the algorithm first establishes independence relations using an independence test - which can vary based on the type of random variables. This is done between each pair of random variables and ensures that these are reflected by the presence – or absence – of edges. Then, based on the cardinality of the remaining edges, conditional independence within cliques is examined, and additional edges are removed accordingly. For continuous random variables, the tests are typically based on Pearson's linear correlation, although there are different options for categorical variables or ordered factors. Finally, an orientation rule is applied when possible (Meek, 1995), resulting in an output graph representing the data up to equivalence class. The algorithm is outlined in more detail in Algorithm 1.

As an example, a possible way to obtain the graph shown in Figure 2.2, the algorithm would first assign edges between X, Y and Z. Then, if Y and Z were found to be independent, the edge between Y and Z would be removed, then using the orientation rules, would assign the direction of the edges to point to X from Y and Z.

Algorithm 1 PC Algorithm

Require: A list of conditional independence relations on a set of vertices V.

1. Form the complete graph C on the vertex set V. For each pair of variables a, b determine if a, b are independent. If so, delete $a - b$ from C. Call the graph that results from this procedure applied to each pair of variables C_0.

2. For each pair of variables a, b adjacent in C_n, consider S_{ab}, the union of the set of vertices adjacent to a on undirected paths between a and b with the set of vertices adjacent to b on undirected paths between a and b.

i. If S_{ab} does not have cardinality greater than n, go to the next pair of vertices adjacent in C_n.

ii. If S_{ab} does have cardinality greater than n, determine if a, b are independent conditional on any subsets of S_{ab} of cardinality $n + 1$. If so, delete $a - b$. Call the graph that results from this procedure applied to each pair of variables C_{n+1}. Continue until a value $f + 1$ of n is reached such that β is not satisfied for any pair.

3. Take each triple of vertices a, b, c such that a, b are adjacent in C_f, and so are b, c but a, c are not adjacent in C_f.

a. Consider P_{ac}, the union of the set of vertices adjacent to a on undirected paths between a and c and and the set of vertices adjacent to c on undirected paths between c and a.

b. Orient $a - b - c$ as a $\rightarrow$ b $\leftarrow$ c if and only if a and c are dependent on every subset of P_{ac} containing b.

c. Output equivalence class graphs consistent with these orientations.

The independence tests used for constraint-based algorithms typically fall under some type of hypothesis testing. An important example is the asymptotic χ^2 test, geared toward larger sample sizes. When an initial graph structure is triangulated and therefore represents decomposable models, these tests can be used for edge deletion. Models using these tests are mainly used in the hierarchical interaction framework.

Independence relations also need to be established in the conditional case, not just pairwise. When the null distribution is known to establish such dependencies, permutation tests are

often used, which enable calculations of conditional distributions of some test statistic. As this allows relaxation of distributional assumptions, as well as not necessarily requiring a large sample size, constraint-based learning algorithms would then be ideal to use, if a test statistic is found that could be used as part of a formal test of independence, with some critical value that can be compared to some distribution.

Overall, the PC algorithm as a constraint-based algorithm can learn the structure of discrete, or linearly correlated data well and is therefore often used. The main interest is to develop methodology that can also learn the structure of data containing continuous variables that may not be linearly correlated, but there are still important connections that could be learned.

2.1.5 Score-based learning algorithm

Unlike constraint-based learning algorithm, score-based learning algorithms use optimization techniques instead of independence tests. As with most optimization problems, the main difference in applying different versions of a score-based algorithm is the way a scoring function assigns scores to each potential network structure. When learning possible graphs, most of the scoring functions are decomposable, expressing the score for the whole network as a variable-wise product, allowing optimization through local updates (Pensar et al, 2017). One of the most common score-based methods is some variant of the Hill-Climbing algorithm.

The mainly used scoring functions apply the Minimal Description Length (MDL) score (Lam and Bacchus) and the BDe score (Heckerman et al, 1995). The MDL score is based on similarly named MDL principle (Rissanen, 1989), aiming to minimize the combined length needed to describe the structure and the associated data, ensuring balance between network complexity and how well the structure represents the observed frequencies. Based on universal coding, this is done through the use of some encoding scheme (Cover and Thomas, 1991). On the other hand, the BDe score, rooted in Bayesian statistics, is assigned by calculating the posterior probability of network structures given the observed data. After choosing a prior probability of the network structure, the search then aims to find the structure where BDe is maximized.

Once the scoring function is chosen, the optimization can become increasingly complex given

the number of potential structures, as this could require calculations for all possible configurations for edges between nodes, as well as edge directions when an edge is present. A common way to get around this is using heuristic search, a simple, but common example being a greedy search. After initializing with a network, it is compared repeatedly to a structures with one single change - such as edge deletion, addition or reversal. The largest improvement in score is then chosen as the next baseline for the following change in the network. This is repeated until a local maximum is reached.

The Hill-Climbing algorithm typically starts with an empty graph. Then, making one change to the network structure at a time, the scoring function is evaluated until the score can no longer be improved. As a greedy search, the algorithm does not return to a previous structure with lower score. When using such score-based algorithms, one must take note that the choice of starting point and the order of fitted arcs may result in a local maximum score - especially when applying greedy searches -, as well as the impact of the scoring function of choice. The algorithm is outlined in Algorithm 2.

Algorithm 2 Hill-Climbing Algorithm

1. Choose a network structure G over V, usually (but not necessarily) empty.
2. Compute the score of G, denoted as $Score_G = Score(G)$.
3. Set $maxscore = Score_G$.
4. Repeat the following steps as long as $maxscore$ increases:
   a. for every possible arc addition, deletion or reversal not resulting in a cyclic network:
      i. compute the score of the modified network G^*, $Score_{G^*} = Score(G*)$:
      ii. if $Score_{G*} > Score_G$, set $G = G^*$ and $Score_G = Score^*_G$.
   b. update $maxscore$ with the new value of $Score_G$.
5. Return the directed acyclic graph G.

It then appears that score-based learning algorithms offer more flexibility through the choice of scoring functions, potentially helpful for cases with continuous data where learning the structure through linear relationships may not be of interest. In addition, greedy searches are fairly efficient as the scores used usually decompose. However, as the order of the different

edges added and removed may vary, the learned structure with the highest assigned scored could change with repeated application of such algorithms.

In literature, the majority of Bayesian networks belong to one of two categories, depending on the random variables used to construct them (Nagarajan et al., 2013). The first and most common class is Discrete Bayesian networks, where variables are discrete, resulting in multinomial local and global distributions. The global distribution can then be represented as conditional probability tables. The second class is Gaussian Bayesian networks (Geiger and Heckerman, 1994), where the global distribution is multivariate normal and the local distributions are univariate normal.

Each variable in this case is essentially linearly regressed on their parents, assuming linear dependence in the network structure. The conditional distribution of a node is then given by a normal distribution, where the mean is linearly related to its parents, and its variance is independent of it the parent vertices. For random variables $(X_1, ..., X_n)$, network structure S and associated parameters $\boldsymbol{\theta_s}$, the local likelihood is then the linear regression

$$p(x_i|\mathbf{pa}_i, \boldsymbol{\theta_i}, S) = N(m_i + \Sigma_{x_j \in \mathbf{pa}_i} b_{ji} x_j, \sigma_i), \tag{2.1.5}$$

where $N(\mu, v)$ is a normal distribution with mean μ and variance $\sigma > 0$, m_i is the conditional mean of X_i, b_{ji} a regression coefficient measuring strength of dependency between x_j and x_i (where x_j is on of x_i's parents), and σ_i is the residual variance of X_i given the the parents of the node. While it is also possible to have a mixture of discrete and continuous variables in a model, but typically, discrete nodes can only have discrete parents (Jordan, 1999). While the resulting graphs often look very similar (although naming conventions may be different), discrete networks often are accompanied by associated probability tables.

Although these are the two most common classes of Bayesian networks, it is of interest whether a network structure can be learned when the random variables of interest are continuous, but not linearly dependant on their parents – for example, the dependence between a variable and its parents is in its variance–, or potentially not normally distributed. Trying to learn such a network structure with non-Gaussian continuous setting may result in some relationships not being represented, even if there is some dependence between the variables, as the current learning algorithms are not necessarily configured to detect them. This is due to the restriction of learning algorithms that continuous variables are typically assumed to

be marginally and jointly Gaussian.

When encountering continuous variables that cannot be included in a Gaussian BN, the general approach currently is discretization. The random variables are approximated through binning so that they may be included in a Discrete BN, as this class of networks has a number of structural learning algorithms available. Alternatively, one can introduce binary variables to associate with continuous variables, dichotomizing them at the means of these variables (Cox and Wermuth, 1994). However, the main issue with discretization of continuous variables is the loss of information. This loss then could lead to different interpretations regarding conditional independence (Spirtes, 1993), as well as the relationships between variables, depending on how the continuous scale is broken down, especially if these relationship are more complex.

Therefore, it is of interest to develop some methodology that does not require discretization of such continuous variables to learn the network structure, but could still recognize dependencies that are not necessarily linear. In order to do so, it is necessary to identify either some scoring function or constraint that can be applied to non-Gaussian continuous variables that can measure independence of in a wider extent, both pairwise and conditional. This will eventually lead three algorithm variants, which were then applied to enhance a dimension reduction tool for image recognition, as well as a very intuitive visual representation that led to an alternative factorization used to identify different speakers in a dataset. The next section describes key methodology to use such measures.

2.2 Kernel methods

2.2.1 Introduction

Learning problems often arise in the form of pattern recognition, or classification (Schölkopf and Smola, 2001). In the simplest, binary case, it deals with assigning a new observation to one of two classes. Initially, observations are part of some non-empty set $\mathcal{X}$, and are typically not restricted in what they represent. However, when it comes to learning problems, it is also important to have some structure associated with these observations in order to extend knowledge to data that has not been seen yet. In classification, this means being able to assign new input to one of the classes. To do so, it is necessary to have a way to tell when data points are similar to each other.

During supervised learning, when class labels are normally available to some extent, the similarity of the classification can be assessed easily simply by observing which class some observations belong to. The main aim, however, is to compare similarity based on the data that is used to create the classification rule in the first place, which introduces the idea of a similarity measure.

Consider a function that, given x and x', a real number characterizing their similarity is obtained. This function takes the form

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

$$(x, x') \to k(x, x'),$$

define a **kernel** (Schölkopf and Smola, 2005). Typically, it is assumed that these functions are symmetric, which means $k(x, x') = k(x', x)$. As it is not straightforward to describe these similarity measures in general, a simple example for a kernel is the dot product, or inner product $\langle \mathbf{x}, \mathbf{x}' \rangle$, calculating the cosine angle between vectors $\mathbf{x}$ and $\mathbf{x}'$ of length 1.

$$\langle \mathbf{x}, \mathbf{x}' \rangle = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}'_i. \tag{2.2.1}$$

Using this simple kernel, it then becomes possible to calculate the norm of a vector:

$$||\mathbf{x}|| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}. \tag{2.2.2}$$

When applied to two different vectors, this kernel calculates the distance between two vectors, which is a simple way to demonstrate how similar they are by calculating how close to each other they are. However, an important requirement is that the vectors, or variables of interest exists in a space where the dot product exists. Then, $\mathcal{X}$ should be considered a subset of the vector space $\mathbb{R}^{\mathbb{N}}$.

For a kernel to be utilized as a similarity measure, a feature space $\mathcal{H}$ is required, introduced through mapping

$$\Phi : \mathcal{X} \to \mathcal{H}$$

$$x \mapsto \mathbf{x} := \Phi(x).$$

An example for this mapping is through monomials, where the product of variables contains the relevant information about the data. They often form basis functions and are used as part of polynomial classifiers (Schürmann, 1996), where the obtained products are then used for learning. In a simple case, given a two dimensional space for X_1 and X_2, $\mathcal{X} = \mathbb{R}^2$, a nonlinear mapping would then appear as

$$\Phi : \mathbb{R}^2 \to \mathcal{H} = \mathbb{R}^3,$$

$$(X_1, X_2) \mapsto (X_1^2, X_2^2, X_1 X_2).$$

While this is fairly straightforward in this case, the number of monomials rapidly increase along with the input dimensions. However, there often exists a way to compute a dot product without having to specify the exact mapping. Given the type of mapping used, the kernels as similarity measures can be extended to more general cases, potentially nonlinear. This mapping then allows linear algebra to be used as a tool for learning methods, enabling the kernel to take form

$$k(x, x') := \langle \mathbf{x}, \mathbf{x}' \rangle = \langle \Phi(x), \Phi(x') \rangle.$$

Using this form to represent kernels in order to calculate a dot product then does not require computation of mapping Φ.

For supervised learning problems, kernels can then be used to classify new observations. In the simple, binary case, the dot product is often enough to define a decision boundary between the two classes by calculating the average of each class, then checking which one is closer to the new input and assign it to that class. While there are more complex classi-

fication methods available, this leads to a line of methods where the classes of interest are separable by some hyperplane in a dot product space $\mathcal{H}$ (Vapnik and Lerner, 1963). This includes finding an optimal hyperplane which separates classes to highest extent. Usually, in order to do so, some objective function is required subject to some constraints based on the parameters, which leads to constrained optimization problems. These problems are solved by introducing Lagrange multipliers α_i and a Lagrangian L is then minimized with respect to variables of interest as well as maximized with respect to the multipliers, essentially finding a saddle point.

In addition, representing the dot product in some feature space $\mathcal{H}$ as a mapped function evaluation that allows to express key formulae in terms of the input in $\mathcal{X}$. This is referred to as the *kernel trick*, and is often used to express more complex classifiers in a simpler, original space (Boser et al, 1992). The kernel trick can also be extended to problems other than supervised learning, such as regression, similarly turning it into constrained optimization by introducing Lagrange multipliers. Furthermore, applying the kernel method to represent dot products in feature spaces can be used for feature extraction, such as Principal Component Analysis (PCA), where information is originally represented as a liner combination of random variables. In addition to previous constraints, applying kernel methods to PCA becomes a nonlinear feature extraction function, requiring the solution of an eigenvalue problem.

2.2.2 Defining a kernel and defining a mapping

Through some of these applications, it appears that an important use of kernel methods is to represent the data in a different feature space. Given the question of interest, the choice of the kernel also becomes important. In addition, the objective can also become finding a feature space where a kernel computes the dot product in that space. By choosing a kernel first, the assumptions made can be relaxed, the domains of the data no longer requiring a structure – apart from being non-empty –, generalizing kernel methods to a wider area. One of the more popular choice is the Gaussian kernel

$$k(x, x') = exp(-\frac{||x - x'||^2}{2\sigma^2}), \quad \sigma > 0, \tag{2.2.3}$$

as well as the polynomial kernel

$$k(x, x') = \langle x, x' \rangle^d, \tag{2.2.4}$$

with the special case when we have a degree-1 polynomial kernel$k(x, x') = (1 + x^T x)^d$, essentially the linear kernel.

Kernels can also be represented as a matrix – called the Gram matrix of k–, which given function $k : \mathcal{X}^2 \to \mathbb{R}$, or $\mathbb{C}$, is an $n \times n$ matrix where

$$K_{ij} := k(x_i, x_j).$$

The kernels used for similarity measures are typically positive definite, which means that for all $a_i, x_i (i = 1, ..., n)$, the function k of choice satisfies

$$\sum_{i=1,j=1}^{N} a_i a_j k(x_i, x_j) \geq 0,$$

or

$$\sum_{i=1,j=1}^{N} a_i a_j K_{ij} \geq 0,$$

which implies positivity on the diagonal, $k(x, x) \geq 0$ ($K_{ii} \geq 0$) for all $x \in \mathbb{X}$, and symmetry $k(x_i, x_j) = k(x_j, x_i)$ ($K_{ij} = K_{ji}$).

These kernels can then be interpreted as a more general version of a dot product. Although kernels are not automatically linear in arguments, the Cauchy-Shwarz inequality does apply for positive definite kernels. That is, given $x_1, x_2 \in \mathcal{X}$

$$|k(x_1, x_2)|^2 \leq k(x_1, x_1)k(x_2, x_2).$$

Then, given the the kernel chosen is real-valued and positive definite, the mapping from $\mathcal{X}$ to the space of functions in $\mathbb{R}$, that is $\mathbb{R}^{\mathcal{X}} := \{f : \mathcal{X} \to \mathbb{R}\}$ through

$$\mathbf{\Phi} : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$$

$$x \mapsto k(., x),$$

where $\mathbf{\Phi}(x)$ assigns $k(x', x)$ to x', or $\mathbf{\Phi}(x)(.) = k(., x)$. Through this mapping, observations are then represented as functions on $\mathcal{X}$ by how similar they are to the rest of the input. It is also important to ensure that there exists a dot product that satisfies the kernel trick.

In the case when the mapping is defined first, if it maps $\mathcal{X}$ into a dot product space, one can then find a positive definite kernel by writing

$$\sum_{i,j}^{N} c_i c_j k(x_i x_j) = \langle \sum_{i}^{N} c_i \mathbf{\Phi}(x_i), \sum_{j}^{N} c_j \mathbf{\Phi}(x_j) \rangle = | \sum_{i}^{N} c_i \mathbf{\Phi}(x_i)|^2 \geq 0. \tag{2.2.5}$$

This then allows construction of kernels from a feature map. The dot product spaces necessary are also called pre-Hilbert spaces, and can be turned into a Hilbert space by completing it in the norm representing the dot product $||f|| := \langle f, f \rangle$. The mapping is then often called a Reproducing Kernel Hilbert Space. A Reproducing Kernel Hilbert Space (RKHS) is a space of functions where points are evaluated by a continuous linear functional. If functions f and g in the RKHS are close in norm $||f - g||$, they are close pointwise $|f(x) - f(g)|$ as well. From the point evaluation functional, one can then construct the reproducing kernel (Hoffman et al, 2008). They require the kernel k to have the reproducing property

$$\langle k(x, .), k(x', .\rangle = k(x, x'), \tag{2.2.6}$$

and to span a Hilbert space $\mathcal{H}$. By combining these two requirements, an RKHS then also uniquely determines a kernel(Schölkopf and Smola, 2001).

Keeping these points in mind, there are several factors that are influenced by the choice of kernel. First, it is a measure of similarity for the data. The linear dot product space associated with it is then a linear representation of the data. The kernel also defines the function space that can be used for learning, as its expansion can be used to determine the form of possible solution for the learning problem. In addition, penalties are also affected by the choice, as for every RKHS with a reproducing kernel, there is an associated regularization operator. A kernel can also encode prior knowledge in a Bayesian setting by defining a covariance function for data that are correlated, as well as prior probabilities of different functions.

Typically, it is common to assume that regardless the choice of kernel, it is positive definite. Helpful strategies to obtain a kernel include using the linear combination of kernels, or the pointwise product of two kernels (Parthasaranthy and Schmidt, 1972). In the special case of multiplying a a kernel with a rank-one kernel,

$$k_f(x, x') = f(x)k(x, x')f(x'),$$

where f is a positive functions are called conformal transformations (Amari and Wu, 1999). It is also possible to account for correlations being more relevant over shorter distances by including by using locality-improved kernels (Schölkopf and Smola, 2001). Furthermore, one can choose to represent underlying probability distributions by choosing a Fisher kernel (or natural kernel), these class of function originate from generative models. There are several other special cases, often tailored to assist with the specific requirements arising from the question of interest. While it is often unclear at first what kernel may be useful for a problem at hand, it is through incorporating prior knowledge along with insight from the dataset where the choice can become well informed.

Having introduced kernels, one can then note that in some cases, functions can be expressed in terms of kernels (Hoffman et al, 2008). Specifically, a number of optimization problems can be approached by writing the objective function as kernel expansions over sample points by using the **representer theorem** (Kimeldorf and Wahba, 1971). Given RKHS $\mathcal{H}$ and associated kernel k, a strictly monotonic increasing function $\Omega : [0, \infty) \to \mathbb{R}$ and an arbitrary loss function $c : (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$, the theorem states that each minimizer $f \in \mathcal{H}$ of the regularized risk functional

$$c((x_1, y_1, f(x_1)), ..., (x_n, y_n, f(x_n))) + \Omega(||f||^2_{\mathcal{H}})$$

can also take the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x). \tag{2.2.7}$$

,

if f is in RKHS $\mathcal{H}$. Otherwise, it is necessary to project f into such RKHS through an orthogonal projection, which then allows f to be approximated by a finite linear combination of kernels.

One of the main importances of the representer theorem is that while solving an optimization problem in an infinite-dimensional space $\mathcal{H}$ with linear combinations of kernels centered on arbitrary points of $\mathcal{X}$, the solution lies in the span of n particular kernels centered on the training points. In these cases, it is common to set some of α_i in the loss function to 0. While the kernel representation can simplify optimization problems, the expansion can become too complex in practice, having too many terms, which is usually tackled by approximating the representation in the RKHS norm. This result then enables the extension of a number of statistical models, such as exponential RKHS models and Markov networks.

Overall, as kernels measure similarity in their respective feature spaces, it is of interest to see how they can be used for structural learning. As the main point, it is plausible that some kernel method can be used to indicate the lack of similarity, extending it as far as to measure independence. Then, given a formal setting, such a kernel could then be used to add or remove edges in a learning algorithm. Initially, the aim is to establish pairwise independence where the random variables are continuous, but not Gaussian, followed by exploring the possibility of using kernel methods to establish conditional independence in the same setting.

This section gave some outlining information about kernels, and how they might be useful to measure independence. Section 3.1 will then further elaborate how kernel methods may be used to obtain pairwise independence criteria, and will be used to identify a formal test for pairwise independence where the random variables need not be Gaussian. In addition, this introduction to kernels will be extended further in Section 6, where Gaussian process latent variable models are introduced.

2.3 Mutual Information - review on information theory

The field of information theory is primarily focused on the transmission, processing, extraction, and utilization of information. It can be interpreted as studying the uncertainty regarding how information, travels through channels as messages. These channels are presumed to add additional noise to the message, and so one of the main goals is to obtain as much information as possible while minimizing the noise. While previously not a very clear concept, information in this context is a measurable quantity that is well-defined (Shannon, 1948).

Shannon proposed that in information theory, information can be treated as a a physical quantity. Quite often, this can be considered a message that is encoded, and is then communicated through some channel which adds in noise. The initially encoded message is then decoded by a different component of the system and recovers the message. The amount of information that can travel through a channel is definite and is governed by the so-called laws of information (Shannon and Weaver, 1949). The summary of the basic laws are:

- There is an upper limit, called channel capacity, to information that can travel through a channel.
- The channel capacity is reduced as the amount of noise increases in a channel.
- The channel capacity can be closely approached by encoding of the data.

Information is typically measured in bits, one bit of information enabling the choice between two equally likely outcomes, represented by a binary digit. As the number of outcomes increase, so does the amount of information required to choose a specific outcome. As the choices are represented by binary digits, the number of bits n required to choose one out of m equiprobable outcomes is $\log_2 m = n$. An important note is that while binary digits are values of binary variables, bits refer to amount of information, and are therefore different quantities.

When the outcomes of different events are no longer equiprobable, some events are more likely to occur than others. Given probability $p(A)$ of an event, the Shannon information then measures how unexpected an event is in bits

$$\text{Shannon information} = \log \frac{1}{p(A)} \text{bits},$$

often written as $-\log p(A)$bits. Then, given a random variable x, it is also possible to calculate the average Shannon information spanning over all possible outcomes of x. Defined by its probability distribution, this average is called the **entropy** of the random variable.

Entropy is a key measurement in the field of information theory. In this context, the entropy $H(X)$ of random variable $\mathbf{X}$ is interpreted as the amount of information, or uncertainty associated with $\mathbf{X}$. While entropy itself measures uncertainty, as uncertainty is reduced, information is gained, and are therefore directly linked. If information is received on a random variable, it is also equivalent to have uncertainty (entropy) taken away. The entropy of a discrete variable is calculated as

$$H(X) = E[-\log(P(X))] = -\sum_{i=1}^{n} P(x_i) \log P(x_i), \tag{2.3.1}$$

where $P(X)$ is the probability mass function of $\mathbf{X}$. Given this, it is also possible to define the conditional entropy $H(X|Y)$ as the amount of information required to describe random variable $\mathbf{X}$, given that another random variable $\mathbf{Y}$ is already known. It and can be calculated as

$$H(X|Y) = -\sum p(x,y) \log \frac{p(x,y)}{p(y)}, \tag{2.3.2}$$

or alternatively, due to the chain rule of entropy we also have

$$H(X|Y) = H(X,Y) - H(X),$$

Where $H(X,Y)$ is the joint entropy of $\mathbf{X}$ and $\mathbf{Y}$. Introducing joint entropy allows the use of some useful properties:

- $H(X,Y)$ is non-negative ($H(X,Y) \geq 0$)
- $H(X,Y)$ is always greater than or equal to individual marginal entropies ($H(X,Y) \geq max[H(X), H(Y)]$
- $H(X,Y)$ is always less then or equal to the sum of marginal entropies ($H(X,Y) \leq H(X) + H(Y)$), being equal if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent.

In the case when the amount of entropy is maximized across a distribution of values, it is

called the maximum entropy distribution. It is mostly used to ensure that a communication channel transmits as much information as possible, and its form is based on the values of the random variable (Reza, 1961).

Another related measure is the channel capacity of a system, measuring the maximum amount of information that can travel through a system about the input, given the output. This can then be compared to the channel of interest, where the information transmission typically depends on the entropy of the input, the output and the noise in the channel. High entropy for the output suggest good possibility for information transmission, its level affected by the entropy of the input and noise. In addition, low noise allows the output to get close to the channel capacity, but is reduced as noise increases. Finally, in the special case of noiseless channels, in the case of noiseless channels, Shannon's source coding theorem states that it is possible to encode data before transmission in a way so that it conveys the maximal amount of information.

Using these different entropies it is possible to define Mutual Information (MI), a fairly common information-theoretic measurement between random variables, as well as a potential way to measure independence (Cover and Thomas, 1991). Unlike linear correlation, Mutual Information can also represent dependencies which do not appear in the covariance. Closely linked to entropy, it can be viewed as examining two variables, and measuring the amount of information obtained about one variable by observing the other. Given $\mathbf{X}$ and $\mathbf{Y}$, it can be written as:

$$I(X;Y) = H(X) - H(X|Y), \tag{2.3.3}$$

where $H(X)$ is the marginal entropy of $\mathbf{X}$ and $H(X|Y)$ is the conditional entropy of $\mathbf{X}$ given $\mathbf{Y}$.

Therefore, mutual information can be interpreted as the reduction in uncertainty about $\mathbf{X}$ after observing $\mathbf{Y}$. Often used in topics such as feature selection and Independent Component Analysis (Hyvärinen et al, 2001), Mutual Information $I(X,Y)$ is

- symmetric $I(X,Y) = I(Y,X)$
- non-negative $I(X,Y) \geq 0$

- $I(X;Y) = 0$ if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent

The last point follows from the fact that $I(X;Y) = 0$ is achieved when $H(X) = H(X|Y)$, meaning that observing $\mathbf{Y}$ does not change the amount of information needed to describe $\mathbf{X}$).

In order to use Mutual Information to measure dependence between random variables, a normalized form for mutual information has been proposed (Strehl and Ghosh, 2002) in the form

$$NI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}, \tag{2.3.4}$$

which then ranges from 0 – signifying independence – to 1, which would suggest $X = Y$. An important note is that while the result is a real number both marginal entropies are positive or negative, but if only one of them is negative, it will become a complex number.

Once Mutual Information is obtained, it is then possible to calculate the Conditional Mutual Information (CMI), the expected mutual information between random variables $\mathbf{X}$ and $\mathbf{Y}$, given that $\mathbf{Z}$ has been observed. It takes the form

$$I(X;Y|Z) = H(X,Z) + H(Y,Z) - H(Z) - H(X,Y,Z). \tag{2.3.5}$$

Alternatively, written in relation to MI,one can write

$$I(X;Y|Z) = I(X,Y,Z) - I(X,Z)$$

A smaller value indicates that observing $\mathbf{Z}$ decreases the dependence between $\mathbf{X}$ and $\mathbf{Y}$, where 0 implies conditional independence. This is an appealing property in the context of structural learning of networks, where conditional dependence is a key feature. On the other hand, the maximal amount of conditional dependence is bounded by conditional entropies. Given X, Y and Z, we have

$$I(X;Y|Z) \leq \min\{H(X|Z), H(Y|Z)\}, \tag{2.3.6}$$

generalizing the result for finding the maximum of unconditional Mutual Information (Zvárová,

1974) through the use of entropies.

Furthermore, the proposed normalized version of mutual information has also been extended to the conditional case (Richiardi, 2007) in the form

$$NI(X;Y|Z) = \frac{I(X;Y|Z)}{\sqrt{H(X|Z)H(Y|Z)}},$$

where $NI(X;Y|Z) = 0$ if X and Y are independent given Z, and 1 if $X = Y$ given Z.

The use of CMI then appears to be a potentially useful measure of dependence. While applicable for discrete variables, it is key to ensure that these measurements can be obtained for continuous variables as well.

A key distinction to note is that for discrete random variables, entropy can be viewed to as an absolute measure of uncertainty that is well defined. For continuous variables, the measurement is more relative (Gómez-Villegas et al, 2014), as the randomness, or the potential outcomes of a continuous variable is infinite (Ihara 1993). This would mean that as each instance of a continuous variable is specified with infinite precision, the amount of information communicated is infinite as well (Stone, 2015), implying all continuous random variables have the same entropy. In this case, to obtain different values for different variables, the differential entropy is required, ignoring infinite terms, which given density $f(x)$ takes the form

$$H(X) = -\int_S f(x)ln[f(x)]dx, \tag{2.3.7}$$

where S is is the support set of the random variable. Similarly, the joint differential entropy for $X_1, ..., X_n$ can be written as

$$H(X_1, ..., X_n) = -\int f(x_1, ..., x_n)ln[f(x_1, ..., x_n)]dx_1, ..., dx_n.$$

Then, the conditional entropy can also be obtained using these forms, and thus extending the definition to Mutual Information in both pairwise and conditional cases. This can be obtained in closed form for some specific distributions such as jointly Gaussian random

variables

$$I(X,Y) = -\frac{1}{2} ln(1-\rho^2), \tag{2.3.8}$$

where ρ is the correlation coefficient. It is worth noting that in this case, the entropies required to obtain $I(X,Y)$ all depend only on the covariance matrix of the random variables. Unfortunately, when aiming to extend further, the calculation needed to estimate mutual information can become too complex to for non-Gaussian distributions, or not plausible to obtain in closed form. Therefore, it is of interest to find a reliable and efficient way to estimate Mutual Information through approximation.

Chapter 3

3 Key methodology

Throughout the previous section, the foundation of the relevant topics were described. This section is focused on identifying the key developments for these topic that were then used in order to create a structural learning algorithm.

3.1 Independence using kernel methods

Through searching for a suitable measure of independence, several kernel methods refer to the work of Renyi (1959). Given X and Y being random variables on a probability space, neither of them being constant with probability 1, a proposed set of desired properties for a measure of dependence $\delta(\cdot, \cdot)$ are:

A) $\delta(X, Y)$ is defined for any pair of random variables X and Y neither of them being constant with probability 1.

B) $\delta(X, Y) = \delta(Y, X)$, that is δ is symmetric.

C) $0 \leq \delta(X, Y) \leq 1$

D) $\delta(X, Y) = 0$ if and only if X and Y are independent.

E) $\delta(X, Y) = 1$ if there is a strict dependence between X and Y , i. e. either $X = g(Y)$ or $Y = f(X)$ where g and f are Borel -measurable functions.

F) If the Borel-measurable functions f and g map the real axis in a one-fo-one way onto itself, $\delta(f(X), g(Y)) = \delta(X, Y)$

G) If the joint distribution of X and Y is normal, then $\delta(X, Y) = |R(X, Y)|$ where $R(X, Y)$ is the correlation coefficient of X and Y.

Along with some other suggestions, one of such metrics proposed is the maximal correlation

$$\delta(X, Y) = \sup_{f,g} R(f(X), g(Y)). \tag{3.1.1}$$

The remainder of this subsection is largely focused on reviewing a line of kernel methods similar in form, adhering to these properties, mainly originating from a Technical Report by

Gretton et al. (2003), slightly rewritten to obtain notations more familiar in statistics. The original paper provides some required definitions, moves on the idea of kernel covariance, which is helpful to understand future variants of a dependence measurement based on kernel methods. It was later on used to create future variants as well.

3.1.1 Kernel covariance - definitions

First, random vectors $\mathbf{x}$ and $\mathbf{y}$ are defined in $\mathcal{X}, \mathcal{Y}$ respectively, where $\mathcal{X} \subset R^d, \mathcal{Y} \subset R^d$.Next, the mappings ϕ_x and ϕ_y are required such that

$$\phi_x : \mathcal{X} \to \mathcal{F}_\mathcal{X}, \phi_y : \mathcal{Y} \to \mathcal{F}_\mathcal{Y},$$

$\mathcal{F}_\mathcal{X}$ and $\mathcal{F}_\mathcal{Y}$ potentially Reproducing Kernel Hilbert Spaces. The covariance matrices are then generalized and have the same form in the mapped feature spaces:

$$\mathbf{\Sigma}_{\phi_x\phi_y} = \mathbf{E}_{\phi_\mathbf{x}\phi_\mathbf{y}}(\phi_\mathbf{x}(\mathbf{x}) - \mu_{\phi_x})(\phi_\mathbf{y}(\mathbf{y}) - \mu_{\phi_y})^T) \tag{3.1.2}$$

$$\mathbf{\Sigma}_{\phi_x\phi_x} = \mathbf{E}_{\phi_\mathbf{x}\phi_\mathbf{x}}(\phi_\mathbf{x}(\mathbf{x}) - \mu_{\phi_x})(\phi_\mathbf{x}(\mathbf{x}) - \mu_{\phi_x})^T) \tag{3.1.3}$$

$$\mathbf{\Sigma}_{\phi_y\phi_y} = \mathbf{E}_{\phi_\mathbf{y}\phi_\mathbf{y}}(\phi_\mathbf{y}(\mathbf{y}) - \mu_{\phi_y})(\phi_\mathbf{y}(\mathbf{y}) - \mu_{\phi_y})^T) \tag{3.1.4}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{\phi_\mathbf{x}\phi_\mathbf{x}} & \mathbf{\Sigma}_{\phi_\mathbf{x}\phi_\mathbf{y}} \\ \mathbf{\Sigma}^T_{\phi_\mathbf{x}\phi_\mathbf{y}} & \mathbf{\Sigma}_{\phi_\mathbf{y}\phi_\mathbf{y}} \end{bmatrix}$$

Initially, we will look at $\phi_x(\mathbf{x}) = \mathbf{x}$ for simplicity. Given n iid samples $\mathbf{z} = (\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)$, we have

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & ... & \mathbf{x}_n \end{bmatrix}^T, \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & ... & \mathbf{y}_n \end{bmatrix}^T,$$

and therefore can estimate $\mathbf{\Sigma}$ as

$$\hat{\mathbf{\Sigma}}_{xy} = \frac{1}{n-1}\mathbf{X^T H Y}, \hat{\mathbf{\Sigma}}_{xx} = \frac{1}{n-1}\mathbf{X^T H X}, \hat{\mathbf{\Sigma}}_{yy} = \frac{1}{n-1}\mathbf{Y^T H Y}, \tag{3.1.5}$$

where $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$.

Next, the centred sample matrices are defined as

$$\widetilde{\mathbf{X}} = \mathbf{HX}, \widetilde{\mathbf{Y}} = \mathbf{HY},$$

noting that as $\mathbf{H}$ is idempotent and symmetric,

$$\widetilde{\mathbf{X}}^{\mathbf{T}}\widetilde{\mathbf{X}}$$

gives an estimate for $\mathbf{\Sigma}_{xx}$ (ignoring a factor of $\frac{1}{n-1}$), as

$$\widetilde{\mathbf{X}}^{\mathbf{T}}\widetilde{\mathbf{X}} = (\mathbf{HX})^{\mathbf{T}}\mathbf{HX} = \mathbf{X}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}}\mathbf{HX} = \mathbf{X}^{\mathbf{T}}\mathbf{HHX} = \mathbf{X}^{\mathbf{T}}\mathbf{HX} \propto \hat{\mathbf{\Sigma}}_{xx}. \tag{3.1.6}$$

Finally, given F_X and F_Y are RKHSs, we define uncentred Gram matrices as

$$\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathbf{T}}, \mathbf{L} = \mathbf{Y}\mathbf{Y}^{\mathbf{T}},$$

with centred alternatives

$$\widetilde{\mathbf{K}} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^{\mathbf{T}}, \widetilde{\mathbf{L}} = \widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^{\mathbf{T}}.$$

3.1.2 Kernel covariance as an eigenvalue problem

Given the previous definitions, the normalised covariance can then be defined similarly to canonical correlation by finding vectors $\alpha_i \in \mathcal{F}_\mathcal{X} : \alpha_i^T\alpha_i \leq 1, \beta_i \in \mathcal{F}_\mathcal{Y} : \beta_i^T\beta \leq 1$, projecting $\mathbf{x}, \mathbf{y}$ on them, such that γ_i, the covariance between the projections is a stationary point w.r.t. α_i, β_i. Then we can say that the stationary points γ_i of

$$cov(\alpha^{\mathbf{T}}\mathbf{x}, \beta^{\mathbf{T}}\mathbf{y}) : \|\alpha\|_{\mathcal{F}_\mathcal{X}} \leq 1, \|\beta\|_{\mathcal{F}_\mathcal{Y}} \leq 1,$$

are given by solutions to the eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \hat{\mathbf{\Sigma}}_{xy} \\ \hat{\mathbf{\Sigma}}_{xy}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \gamma_i \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}. \tag{3.1.7}$$

Proof: We wish to find vectors $\alpha_i \in \mathcal{F}_\mathcal{X} : \alpha_i^\intercal\alpha_i \leq 1, \beta_i \in \mathcal{F}_\mathcal{Y} : \beta_i^\intercal\beta_i \leq 1$ onto which x,y respectively project in a way so γ_i, the covariance between the them is a stationary point w.r.t. $\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i$. The Lagrangian is written

$$L(\boldsymbol{\alpha}, \beta, \lambda, \xi) = \mathbf{E}_{\mathbf{x},\mathbf{y}}\left((\mathbf{x}\boldsymbol{\alpha})^\top(\mathbf{y}\boldsymbol{\beta})\right) - \mathbf{E}_{\mathbf{x}}\left(\mathbf{x}\boldsymbol{\alpha}\right)^\top \mathbf{E}_{\mathbf{y}}\left(\mathbf{y}\boldsymbol{\beta}\right) - \lambda\left(\boldsymbol{\alpha}^\top\boldsymbol{\alpha} - 1\right) - \xi\left(\boldsymbol{\beta}^\top\boldsymbol{\beta} - 1\right)$$

$$= \boldsymbol{\alpha}^\top \hat{\boldsymbol{\Sigma}}_{\mathbf{xy}} \boldsymbol{\beta} - \lambda \left(\boldsymbol{\alpha}^\top \boldsymbol{\alpha} - 1 \right) - \xi \left(\boldsymbol{\beta}^\top \boldsymbol{\beta} - 1 \right).$$

In order to compute the saddle points, the partial derivatives w.r.t. $\alpha, \beta, \lambda, \xi$ are set to zero:

$$\frac{\partial L}{\partial \boldsymbol{\alpha}} = \hat{\boldsymbol{\Sigma}}_{xy} \boldsymbol{\beta} - 2\lambda \boldsymbol{\alpha} = 0$$

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \hat{\boldsymbol{\Sigma}}_{xy}^\top \boldsymbol{\alpha} - 2\xi \boldsymbol{\beta} = 0$$

Then, by multiplying the first equation by $\boldsymbol{\alpha}^T$, and the second by $\boldsymbol{\beta}^T$, we get

$$\begin{array}{ll} \boldsymbol{\alpha}^\top \hat{\boldsymbol{\Sigma}}_{xy} \boldsymbol{\beta} - 2\lambda \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = & \boldsymbol{\alpha}^\top \hat{\boldsymbol{\Sigma}}_{xy} \boldsymbol{\beta} - 2\lambda = 0 \\ \boldsymbol{\beta}^\top \hat{\boldsymbol{\Sigma}}_{xy}^\top \boldsymbol{\alpha} - 2\gamma \boldsymbol{\beta}^\top \boldsymbol{\beta} = & \boldsymbol{\beta}^\top \hat{\boldsymbol{\Sigma}}_{xy}^\top \boldsymbol{\alpha} - 2\xi = 0 \end{array},$$

giving $\lambda = \xi$. Then, writing $\gamma = 2\lambda = 2\xi$, and substituting back into the original equation for the saddle points and changing to matrix notation yields

$$\begin{bmatrix} 0 & \hat{\boldsymbol{\Sigma}}_{xy} \\ \hat{\boldsymbol{\Sigma}}_{xy}^\top & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \gamma \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \tag{3.1.8}$$

By writing $\hat{\boldsymbol{\Sigma}}_{\mathbf{xy}} = \mathbf{X^T H Y}$ (again, off by a factor of $\frac{1}{n-1}$, and making use of H being idempotent and symmetric), one can also write

$$\begin{bmatrix} \tilde{\mathbf{X}}^\mathbf{T} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Y}}^\mathbf{T} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{Y}} \beta_i \\ \tilde{\mathbf{X}} \alpha_i \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{X^T H Y} \\ \mathbf{Y^T H X} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \gamma_i \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}. \tag{3.1.9}$$

Making use of the representer theorem, we can also define

$$\boldsymbol{\alpha}_i = \tilde{\mathbf{X}}^\mathbf{T} \mathbf{c}_i, \boldsymbol{\beta}_i = \tilde{\mathbf{Y}}^\mathbf{T} \mathbf{d}_i.$$

Replacing these in the previous equation, we have

$$\begin{bmatrix} \tilde{\mathbf{X}}^\mathbf{T} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Y}}^\mathbf{T} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{Y}} \tilde{\mathbf{Y}}^\mathbf{T} \mathbf{d_i} \\ \tilde{\mathbf{X}} \tilde{\mathbf{X}}^\mathbf{T} \mathbf{c_i} \end{bmatrix} = \gamma_i \begin{bmatrix} \tilde{\mathbf{X}}^\mathbf{T} \mathbf{c}_i \\ \tilde{\mathbf{Y}}^\mathbf{T} \mathbf{d}_i \end{bmatrix} \tag{3.1.10}$$

Then, multiplying each side with $\begin{bmatrix} \tilde{\mathbf{X}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Y}} \end{bmatrix}$, the eigenvalue problem for normalized covariance can also be written using the centred Gram matrices as

$$\begin{bmatrix} \tilde{\mathbf{X}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Y}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}^{\mathbf{T}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Y}}^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^{\mathbf{T}}\mathbf{d_i} \\ \tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathbf{T}}\mathbf{c_i} \end{bmatrix} = \gamma_i \begin{bmatrix} \tilde{\mathbf{X}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Y}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}^{\mathbf{T}}\mathbf{c}_i \\ \tilde{\mathbf{Y}}^{\mathbf{T}}\mathbf{d}_i \end{bmatrix},$$

which can be simplified as

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ \tilde{\mathbf{L}}\tilde{\mathbf{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \gamma_i \begin{bmatrix} \tilde{\mathbf{K}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{L}} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix}. \tag{3.1.11}$$

Alternatively, one can apply the representer theorem earlier. We begin with the estimated covariance between the functions in the projected RKHS space

$$\hat{cov}(\phi_x(x), \phi_y(y)) = \sum_{k=1}^{n} [\phi_x(x_k) - \bar{\phi}_x(x)][\phi_y(y_k) - \bar{\phi}_y(y)]^T, \tag{3.1.12}$$

such that $||\phi_x|| = 1, ||\phi_y|| = 1$ and $\phi_{xk}(x) = \frac{1}{n}\sum_{k=1.}^{n} \phi_x(x_k)$.

Then, using the reproducing property, we can write

$$\phi_x(x) = \sum_{i=1}^{n} c_i \tilde{K}(x_i, x)$$

$$\phi_y(y) = \sum_{i=1}^{n} d_i \tilde{L}(y_i, y).$$

Substituting back into the covariance estimate this yields

$$\hat{cov}(\sum_{i=1}^{n} c_i \tilde{K}(x, x_i), \sum_{j=1}^{n} d_j \tilde{L}(y, y_j)) = \mathbf{c}^T \tilde{K} \tilde{L}^T \mathbf{d}, \tag{3.1.13}$$

subject to constraints $\mathbf{c}^T \tilde{K} \mathbf{c} = 1, \mathbf{d}^T \tilde{L} \mathbf{d} = 1$, since we have

$$||\phi_x(x)||^2 = ||\sum_{k=1}^{n} c_i \tilde{K}(x_i, x)||^2 = \langle \sum_{i=1}^{n} c_i \tilde{K}(x_i, x), \sum_{j=1}^{n} c_j \tilde{K}(x_j, x) \rangle = 1.$$

Next, in order to optimize this estimate with respect to c,d using the constraints, we obtain

the Lagrangian

$$\mathcal{L}(\mathbf{c}, \mathbf{d}) = \mathbf{c}^T \tilde{K} \tilde{L}^T \mathbf{d} - \lambda(\mathbf{c}^T \tilde{K} \mathbf{c} - 1) - \xi(\mathbf{d}^T \tilde{L} \mathbf{d} - 1).$$

To calculate the saddle points, we write $\gamma = 2\lambda = 2\xi$ and take the partial derivatives

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}} = \tilde{K} \tilde{L}^T \mathbf{d} - \gamma \tilde{K} \mathbf{c} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{d}} = \tilde{L} \tilde{K}^T \mathbf{c} - \gamma \tilde{L} \mathbf{d} = 0,$$

which in matrix notation can be written in the form of the eigenvalue problem

$$\begin{bmatrix} 0 & \tilde{K}\tilde{L}^T \\ \tilde{L}\tilde{K}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \gamma \begin{bmatrix} \tilde{K} & 0 \\ 0 & \tilde{L} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} \tag{3.1.14}$$

With this background it is then possible to show that Kernel Covariance can be used as a measure of independence.

3.1.3 Kernel covariance as a measure of independence

The kernel covariance itself is defined given RKHSs $\mathcal{F}_\mathcal{X}, \mathcal{F}_\mathcal{Y}$ with associated kernels $k(x_i - x_j), l(y_i - y_j)$ as

$$\mathcal{J}\left(\mathbf{P}_{x,y}, \mathcal{F}_\mathcal{X}, \mathcal{F}_\mathcal{Y}\right) = \sup_{f \in \widetilde{\mathcal{F}}_\mathcal{X}, g \in \widetilde{\mathcal{F}}_\mathcal{Y}} \left| \mathbf{E}_{\mathbf{x},\mathbf{y}}[f(\mathbf{x})g(\mathbf{y})] - \mathbf{E}_\mathbf{x}[f(\mathbf{x})]\mathbf{E}_\mathbf{y}[g(\mathbf{y})] \right|, \tag{3.1.15}$$

where

$$\widetilde{\mathcal{F}}_\mathcal{X} := \left\{ f \in \mathcal{F}_\mathcal{X} : \|f\|_{\mathcal{F}_\mathcal{X}} \leq 1 \right\}, \widetilde{\mathcal{F}}_\mathcal{Y} := \left\{ g \in \mathcal{F}_\mathcal{Y} : \|g\|_{\mathcal{F}_\mathcal{Y}} \leq 1 \right\}.$$

Then, using previous definitions $\phi_\mathbf{X}(\mathbf{x}), \phi_\mathbf{y}(\mathbf{y})$ according to $\mathbf{P}_{\mathbf{x},\mathbf{y}}$, the empirical kernel covariance take the form

$$\mathcal{J}_{\text{emp}}\left(\phi_x(\mathbf{x}), \phi_y(\mathbf{y}), \mathcal{F}_\mathcal{X}, \mathcal{F}_\mathcal{Y}\right) = \sup_{f \in \widetilde{\mathcal{F}}_\mathcal{X}, g \in \widetilde{\mathcal{F}}_\mathcal{Y}} \frac{1}{n-1} \left| \sum_{l=1}^{n} f\left(\mathbf{x}_l\right) g\left(\mathbf{y}_l\right) - \frac{1}{n} \left(\sum_{l=1}^{n} f\left(\mathbf{x}_l\right) \right) \left(\sum_{l=1}^{n} g\left(\mathbf{y}_l\right) \right) \right|, \tag{3.1.16}$$

where we can replace

$$f(\mathbf{x}) = \sum_{l=1}^{m} c_l k\left(\mathbf{x}, \mathbf{x}_l\right) = \sum_{l=1}^{m} c_l \mathbf{x}^T \mathbf{x}_l$$

$$g(\mathbf{y}) = \sum_{l=1}^{m} d_l k\left(\mathbf{y}, \mathbf{y}_l\right) = \sum_{l=1}^{m} d_l \mathbf{y}^T \mathbf{y}_l,$$

yielding the maximum stationary point w.r.t $\mathbf{c}, \mathbf{d}$ of the normalised covariance.

After defining the kernel covariance, it is of key interest to show that it can indicate independence. Given $f \in \tilde{F}, g \in \tilde{G}$ on the bounded sets $\mathcal{X} \subset \mathbb{R}^k, \mathcal{Y} \subset \mathbb{R}^l$, then the kernel correlation is zero if and only if $\mathbf{x}$ and $\mathbf{y}$ are independent.

First, one needs to show that the kernel covariance is zero if $\mathbf{x}$ and $\mathbf{y}$ are independent:

$$\mathcal{J}\left(\mathbf{P}_{x,y}, \mathcal{F}_{\mathcal{X}}, \mathcal{F}_{\mathcal{Y}}\right) = \sup_{f \in \widetilde{\mathcal{F}}_{\mathcal{X}}, g \in \widetilde{\mathcal{F}}_{\mathcal{Y}}} \left|\mathbf{E}_{\mathbf{x},\mathbf{y}}[f(\mathbf{x})g(\mathbf{y})] - \mathbf{E}_{\mathbf{x}}[f(\mathbf{x})]\mathbf{E}_{\mathbf{y}}[g(\mathbf{y})]\right|$$

$$= \sup_{f \in \widetilde{\mathcal{F}}_{\mathcal{X}}, g \in \widetilde{\mathcal{F}}_{\mathcal{Y}}} \left|\mathbf{E}_{\mathbf{x}}[f(\mathbf{x})]\mathbf{E}_{\mathbf{y}}[g(\mathbf{y})] - \mathbf{E}_{\mathbf{x}}[f(\mathbf{x})]\mathbf{E}_{\mathbf{y}}[g(\mathbf{y})]\right| = 0.$$

To show the converse, using the case where $\mathbb{R}^{n_x} = \mathbb{R}$ and $\mathbb{R}^{n_y} = \mathbb{R}$, let $[q_1, q_2] \subseteq \mathcal{X}$ and $[r_1, r_2] \subseteq \mathcal{Y}$ on which strictly positive function $u(x) \in \tilde{F}_{\mathcal{X}}$ and $v(y) \in \tilde{F}_{\mathcal{Y}}$ are compactly supported. Then, $u^{1/l}(x) \in \tilde{F}_{\mathcal{X}}, v^{1/l}(y) \in \tilde{F}_{\mathcal{Y}}$ for $l \geq 1$. Using limits

$$\lim_{l \to \infty} u^{1/l}(x) = \mathbb{I}_{x \in [q_1, q_2]} \qquad \textit{and} \qquad \lim_{l \to \infty} v^{1/l}(y) = \mathbb{I}_{y \in [r_1, r_2]},$$

then if the supremum is zero, then

$$\lim_{l \to \infty} \left|\mathbf{E}_{x,y}[u^{1/l}(x)v^{1/l}(y)] - \mathbf{E}_x[u^{1/l}(x)]\mathbf{E}_y[v^{1/l}(y)]\right| = 0,$$

and therefore

$$\mathbf{P}_{x,y}([q_1, q_2], [r_1, r_2]) = \mathbf{P}_x([q_1, q_2])\mathbf{P}_y([r_1, r_2]).$$

Then, as σ-algebra over

$$[q_1, q_2] \times [r_1, r_2] : q_1, q_2 \in \mathcal{X}, r_1, r_2 \in \mathcal{Y}$$

constitute the Borel sets over $\mathcal{X} \times \mathcal{Y}$,

$$\mathcal{J}(\mathbf{P}_{x,y}, \mathcal{F}_{\mathcal{X}}, \mathcal{F}_{\mathcal{Y}}) = 0$$

implies that x and y are independent.

The technical report (Gretton et al, 2003) then generalizes kernel covariance to higher dimensions, which can become very useful for structural learning. However, in higher dimensions kernel covariance only extends to pairwise independence, while ultimately, conditional independence is the goal in order to incorporate it into a learning algorithm. Nevertheless, kernel covariance could potentially be useful as foundation for future methods as it is able to measure pairwise independence.

3.1.4 Constrained Covariance and the Hilbert-Schmidt Independence Criterion

When examining kernel methods to measure independence, a key step taken is defining covariance and cross-covariance operators in RKHSs, then deriving a statistics from them that is able suited to indicate dependence between functions in these spaces (Gretton et al, 2005). An example is to use the Kernel Canonical Correlation, a regularised correlation operator derived from such operators, and using its largest singular value to test independence (Bach and Jordan, 2002).

Similarly, using the the largest singular value of the cross-covariance operator, later work on kernel methods for measuring independence (Gretton et al, 2005), Kernel Covariance was developed into Constrained Covariance (COCO). Not requiring the same regularization as using correlation, given function classes $\mathcal{F}, \mathcal{G}$ and probability measure $P_{x,y}$, taking the form

$$COCO(P_{x,y}; \mathcal{F}, \mathcal{G}) = \sup_{f \in \mathcal{F}, g \in \mathcal{G}} [cov(f(x), g(y)]). \tag{3.1.17}$$

Given independent observations $\mathbf{z} = (x_1, y_1), ..., (x_n, y_n)$, and $\mathcal{F}$ and $\mathcal{G}$ are unit balls in their vector spaces, one obtains an empirical estimate

$$COCO(\mathbf{z}; \mathcal{F}, \mathcal{G}) = \sup_{f \in \mathcal{F}, g \in \mathcal{G}} [\frac{1}{n} \sum_{i=1}^{n} f(x_i) g(y_i) - \frac{1}{n^2} \sum_{i=1}^{n} f(x_i) \sum_{j=1}^{n} g(y_j)]. \tag{3.1.18}$$

In addition, when $\mathcal{F}$ and $\mathcal{G}$ are RKHSs, and we have unit ball functions F and G, one can also write in terms of an eigenvalue problem,

$$COCO(\mathbf{z}; F, G) = \frac{1}{n} \sqrt{||\tilde{\mathbf{K}}\tilde{\mathbf{L}}||_2}, \tag{3.1.19}$$

where $||\cdot||_2$ denotes the largest singular value. Much like Kernel covariance, if F, G belong to the set of bounded continuous function, COCO is 0 if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent. An important requirement, however, for measures such as COCO and other quantities to measure independence when 0, is that the kernels are associated with RKHSs which are universal (Gretton et al, 2005). A kernel is universal if the RKHS function class $\mathcal{F}$ is dense in $\mathcal{C}(\mathcal{X})$ (Steinwart, 2001). This allows measuring independence without a density estimator, but will systematically be large, unless restricted. To obtain a more robust independence measure, the entire spectrum of the cross-covariance operator can be used (Gretton et al 2008), giving the sum of squared singular values of the cross-covariance operator. Given linear operator $\mathcal{C} : \mathcal{G} \to \mathcal{F}$, define the Hilbert-Schimdt norm of $\mathcal{C}$ as

$$||\mathcal{C}||^2_{HS} := \sum_{i=1,j=1}^{n} \langle \mathcal{C}v_i, u_j \rangle^2_{\mathcal{F}},$$

where v_i and u_j are orthonormal bases of $\mathcal{F}$ $\mathcal{G}$ respectively. Then, we have cross-covariance operator $\mathcal{C}_{xy}$, taking the form

$$\mathcal{C}_{xy} = \mathbf{E}_{\mathbf{x},\mathbf{y}}[f(\mathbf{x})g(\mathbf{y})] - \mathbf{E}_{\mathbf{x}}[f(\mathbf{x})]\mathbf{E}_{\mathbf{y}}[g(\mathbf{y})].$$

Combining these then gives the new estimator

$$HSIC(P_{x,y}; \mathcal{F}, \mathcal{G}) = ||\mathcal{C}_{xy}||^2_{HS} = ||\mathbf{E}_{\mathbf{x},\mathbf{y}}[f(\mathbf{x})g(\mathbf{y})] - \mathbf{E}_{\mathbf{x}}[f(\mathbf{x})]\mathbf{E}_{\mathbf{y}}[g(\mathbf{y})]||^2_{HS} \tag{3.1.20}$$

which was named the Hilbert-Schmidt Independence Criterion (HSIC), where $||\cdot||^2_{HS}$ is the squared Hilbert-Schmidt (HS) norm. Matching the quadratic dependence measure (Achard et al, 2003) ,this version of the method has reduced bias, and the empirical estimates are computationally cheaper, as given sample Z, one can obtain an estimate simply using the trace of centered Gram matrices

$$HSIC(Z, \mathcal{F}, \mathcal{G}) = \frac{1}{n^2} tr\tilde{\mathbf{K}}\tilde{\mathbf{L}}, \tag{3.1.21}$$

which can then be used as a test statistic. An appealing feature is that the estimate converges to the population estimate at rate $\frac{1}{\sqrt{n}}$, where n is the sample size, therefore independence tests based on HSIC do not suffer from slow learning rates (Devroye et al, 1996).

Therefore, it appears that some variant of these kernel methods using HSIC can measure independence fairly efficiently. The next step is to further examine this statistic as an inde-

pendence measurement, and aim to embed it in a structural learning algorithm.

3.1.5 The pairwise HSIC test

Given i.i.d. sample $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$, a statistical test

$$\mathcal{T}(Z) : (\mathcal{X} \times \mathcal{Y})^n \to \{0, 1\}$$

distinguishes between the null hypothesis, stating that $\mathbf{X}, \mathbf{Y}$ are independent, that is

$$\mathcal{H}_0 : \mathbf{P}_{xy} = \mathbf{P}_x \mathbf{P}_y$$

and the alternative hypothesis

$$\mathcal{H}_1 : \mathbf{P}_{xy} \neq \mathbf{P}_x \mathbf{P}_y.$$

This is performed through obtaining a test statistic, and comparing it to a threshold. In this context, using the empirical estimate

$$HSIC(Z, \mathcal{F}, \mathcal{G}) = \frac{1}{n^2} tr\tilde{\mathbf{K}}\tilde{\mathbf{L}}, = HSIC(Z),$$

it is of interest to find such threshold. Given significance level α the threshold is typically the quantile $1 - \alpha$ of some distribution. For $HSIC(Z)$, the estimate of $HSIC(P_{x,y}; \mathcal{F}, \mathcal{G})$, an asymptotic distribution under $\mathcal{H}_0$ is then required. One approach to obtain a quantile for this null distribution is through a Gamma-approximation of an infinite sum of χ^2 variables (Kankainen, 1995). That is

$$nHSIC(Z) \sim \frac{x^{\alpha-1} e^{-x/\beta}}{\beta^\alpha \Gamma(\alpha)},$$

where

$$\alpha = \frac{(\mathbf{E}(HSIC(Z))^2}{var(HSIC(Z))}$$

and

$$\beta = \frac{nvar(HSIC(Z)}{\mathbf{E}(HSIC(Z))}.$$

In order to carry out the approximation under $\mathcal{H}_0$ using the available sample, one then needs an estimate $\mathbf{E}(HSIC(Z))$ and $var(HSIC(Z))$. (Gretton et al, 2008). Using the mean elements and covariance operators

$$\mathbf{\Sigma}_{\phi_x \phi_x} = \mathbf{E}_{\phi_\mathbf{x} \phi_\mathbf{x}} (\phi_\mathbf{x}(\mathbf{x}) - \mu_{\phi_x})(\phi_\mathbf{x}(\mathbf{x}) - \mu_{\phi_x})^T)$$

under $\mathcal{H}_0$, we then have

$$\mathbf{E}(HSIC(Z)) = \frac{1}{n}(1 + ||\mu_x||^2||\mu_y||^2 - ||\mu_x||^2 - ||\mu_y||^2), \tag{3.1.22}$$

as well as

$$var(HSIC(Z)) = \frac{2(n-4)(n-5)}{(n)_4}||\mathbf{\Sigma}_{\phi_x\phi_x}||^2_{HS}||\mathbf{\Sigma}_{\phi_y\phi_y}||^2_{HS} + O(n^{-3}) \tag{3.1.23}$$

Having obtained the necessary empirical estimates then makes it possible to find the $1-\alpha$ quantile of the asymptotic distribution which can then be used as threshold to compare the test stastic $HSIC(Z)$ to. Then, it is possible to utilize HSIC as a pairwise independence test, where the data may not need be Gaussian.

In order to fully fit the need for a measure of independence used in structural learning, it is also important to account for conditional independence. While conditional versions of HSIC are possible to calculate, the asymptotic distribution is not always straightforward to approximate. However, the potential for a new direction for the thesis was considered when encountering an algorithm using HSIC independence tests for Structural Equation Modelling (SEM)(Peters et al. 2014). Named Regression with subsequent independence test (RESIT) the algorithm goes through two main phases. First, the HSIC tests are conducted – in this case, for SEMs, this is preceded by regressing each random variable on the rest, then using the residuals to measure dependence to determine the topological order, assigning the residuals that are the least dependent on the remaining variables to be a terminal node. Then, to remove superfluous edges, each node is examined, and further incoming edges are removed until the residuals are not independent anymore.

While used for Structural Equation Modelling, which may not be applicable in this context, the RESIT algorithm utilized the pairwise independence test in a way that established an initial structure by selecting edges based on some criteria and then removed superfluous edges that remained. As an important note, using these tests were also developed into an R package 'dHSIC'. This led to the initial inspiration of using HSIC tests to establish pairwise relations between random variables, and then examining the initial structure to remove edges based on Conditional Mutual Information.

3.2 Approximating Mutual Information

While Mutual Information, and by extension, Conditional Mutual Information has some desirable properties in the context of this thesis, it is often not feasible to obtain for continuous random variables. It is therefore of interest to approximate Mutual Information in some manner. This is frequently done through discretization with some type of binning scheme (Darbellay 1999). At their root, these methods partition the supports into bins, resulting in the finite sum

$$I(X,Y) \approx I_{binned}(X,Y) = \sum_{ij} p(i,j) \log \frac{p(i,j)}{p_x(i)p_y(j)},$$

where i, j denote the bins. In order to estimate $I_{binned}(X,Y)$, one can count the number of points within each bin, with a total N number of points, n_x being the number of points in the ith bin of X, n_y the number of points in the jth bin of Y, and $n(i,j)$ is the number of points in their intersection. Then, with these estimates, an approximation is possible by writing

$$p_x(i) \approx \frac{n_x(i)}{N}, \quad p_y(j) \approx \frac{n_y(j)}{N}, \quad p(i,j) \approx \frac{n(i,j)}{N}.$$

It is important to note that splitting the domain in intervals increases the bias of the estimate with the number of intervals (Tsimpiris et al, 2012). The bin sizes themselves can be optimized to have similar number of points in the intersections $n(i,j)$ for each pair of (i,j) (Fraser and Swinney, 1986). Nevertheless, approximating through binning as well as through frequencies does introduce systematic errors (Kraskov et al, 2004), which then need to be countered by corrections (Grassberger, 1988).

Another option to obtain an estimate of mutual information between two continuous variables is using a nearest neighbour approximation (Tsimpiris et al., 2012). A popular approach is the use of some version of the Kozachenko-Leonenko estimator for entropy (Kraskov et al, 2004), estimating $H(X)$ from the average distance to the k-nearest neighbor, which is then averaged over all x_i. It begins by writing

$$\hat{H}(X) = -\frac{1}{n} \sum_{i=1}^{n} \widehat{\log \mu(x_i)}, \tag{3.2.1}$$

where $\mu(x_i)$ is the mean average distance of observation i to the k-th nearest neighbor.

In the context of nearest neighbors, for calculating $\widehat{\log \mu(x_i)}$, one needs to consider $P_k(\epsilon)$, the probability distribution for the distance between x_i and its k-th nearest neighbor. Any probability $P_k(\epsilon)d\epsilon$ represents the chance of a single point within distance $r \in [\epsilon/2, \epsilon/2+d\epsilon/2]$ from x_i, as well as k-1 points closer to x_i and $n-k-1$ points further from than x_k. Then, having p_i be the mass of the ϵ with x_i as center, the trinomial formula yields

$$P_k(\epsilon)d\epsilon = k\binom{n-1}{k}\frac{dp_i(\epsilon)}{d\epsilon}p_i^{k-1}(1-p_i)^{n-k-1}. \tag{3.2.2}$$

Then, as the expectation value of $\log p_i(\epsilon)$, we also have

$$E(\log p_i) = \psi(k) - \psi(n), \tag{3.2.3}$$

where $\psi(x)$ is the digamma function, $\psi(x) = \Gamma(x)^{-1}d\Gamma(x)/dx$.

Proof. Let $X \sim Beta(\alpha, \beta)$,and the geometric mean $\ln G_X = E[\ln X]$. For a beta distribution, the expected value integral gives:

$$\begin{aligned}
\mathbb{E}[\ln X] &= \int_0^1 \ln x f(x;\alpha,\beta)dx \\
&= \int_0^1 \ln x \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}dx \\
&= \frac{1}{\mathrm{B}(\alpha,\beta)}\int_0^1 \frac{\partial x^{\alpha-1}(1-x)^{2-1}}{\partial\alpha}dx \\
&= \frac{1}{\mathrm{B}(\alpha,\beta)}\frac{\partial}{\partial\alpha}\int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \\
&= \frac{1}{\mathrm{B}(\alpha,\beta)}\frac{\partial B(\alpha,\beta)}{\partial\alpha} \\
&= \frac{\partial \ln B(\alpha,\beta)}{\partial\alpha}
\end{aligned}$$

$$= \frac{\partial \ln \Gamma(\alpha)}{\partial \alpha} - \frac{\partial \ln \Gamma(\alpha+\beta)}{\partial \alpha}$$

$$= \psi(\alpha) - \psi(\alpha+\beta)$$

Then if we have $\alpha = k, \beta = N-k$, we get $E[\ln X] = \psi(k) - \psi(N)$. This expectation is over all $n-1$ points, while x_i is fixed. One can then obtain $\widehat{\log \mu(x)}$ assuming $\mu(x)$ is constant over the volume of ϵ, which gives

$$p(\epsilon) \approx c_d \epsilon^d \mu(x) \tag{3.2.4}$$

Using these, we obtain

$$\log \mu(x_i) \approx \psi(k) - \psi(n) - dE(\log \epsilon) - \log c_d, \tag{3.2.5}$$

where d is the dimension of x, and c_d is the volume of the d-dimensional ball.

Finally, using these equations allow us to estimate entropy as

$$\hat{H}(X) = -\psi(k) + \psi(n) + \log c_d + \frac{d}{n}\sum_{i=1}^{n} \log \epsilon(i), \tag{3.2.6}$$

and therefore an estimate for the joint entropy

$$\hat{H}(X,Y) = -\psi(k) + \psi(n) + \log(c_{d_X} c_{d_Y}) + \frac{d_X + d_Y}{n}\sum_{i=1}^{n} \log \epsilon(i), \tag{3.2.7}$$

where $z_i = (x_i, y_i), d_X + d_Y = d_Z, c_{d_X} c_{d_Y} = c_d$.

When moving on to mutual information, the approximation essentially calculates the average distance from k-th nearest neighbour in the joint space of $\mathbf{X}$ and $\mathbf{Y}$, then compares it to the average number of points within the same distance in the projected marginal spaces, denoting the number of points by n_x, n_y for fixed x.

As an illustration, Figures 3.1 and 3.2 show how the distance $\epsilon(i)$ for a fixed $x(i)$ and by extension $n_{x(i)}$ and $n_{y(i)}$ can be calculated. It is possible to use a single value of $\epsilon(i)$ for both marginal projections, or use separate values for the two marginal projections, slightly changing the value for $n_{x(i)}, n_{y(i)}$. This is typically decided to reduce the difference in errors,

as they are introduced during estimating the marginal and joint entropies, and normally do not cancel out completely.

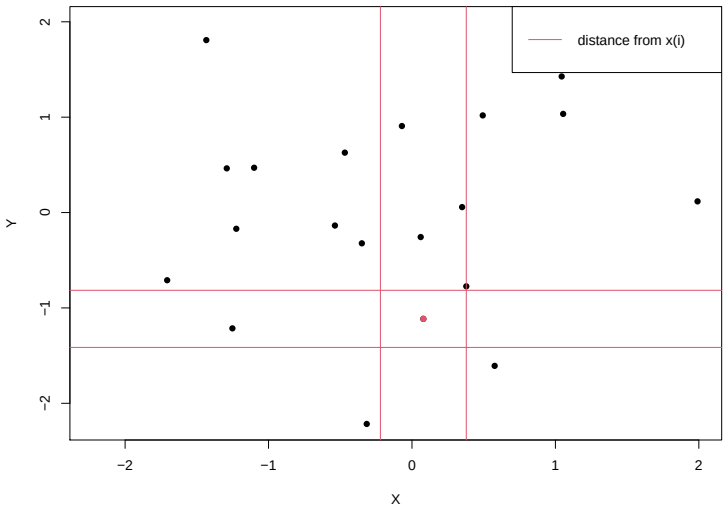


Figure 3.1: Calculating $\epsilon(i)$ in one marginal direction (x(i));$k = 1, n_x = 4, n_y = 1$

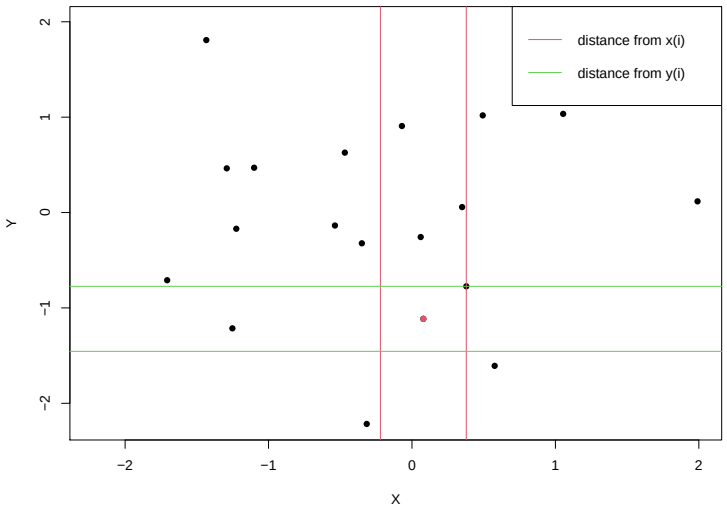


Figure 3.2: Calculating $\epsilon_x(i)$ and $\epsilon_y(i)$ for both marginal directions;$k = 1, n_x = 4, n_y = 2$

In this example, a small random sample from a normal distribution was taken. K, the nearest neighbour to calculate the distances $\epsilon, \epsilon_x(i), \epsilon_y(i)$ was chosen to be 1. Then n_x and n_y were calculated by using the distance to the nearest neighbor projected in one marginal direction

in Figure 3.1, and both marginal directions in Figure 3.2. For Figure 3.1, only the distance projected onto x informs the distance used to calculate n_x and n_y, while Figure 3.2 uses the different distances, according to their respective projections. It demonstrates how projecting the the distance in one or both marginal directions can change the value of ϵ slightly, as n_y=1 when using one distance for both dimensions, and $n_y = 2$ when both directions project their owen average distance.

Having used $\epsilon_x(i)$ in projecting in one direction as well, $n_x = 4$ in both cases. When projecting the same distance in the other dimension in the first case, we have $n_y = 1$, while on the other hand, using $\epsilon_y(i)$ to project in the second direction yields $n_y = 2$. While in this case the difference is rather minimal, the same process is then repeated for all points in the sample, potentially leading to larger differences. The average of these values are then taken, and are used as a penalty of sorts for the estimation. Both ways of calculating n_x and n_y introduces some errors which often do not cancel, but they still provide a comparison between average distance in the joint space and the marginal spaces.

Finally, we can then obtain an estimate for mutual information

$$\hat{I}(X;Y) = \psi(k) + \psi(N) - \langle(\psi(n_x + 1) + (\psi(n_y + 1))\rangle. \tag{3.2.8}$$

A property of this estimate is that much like MI, it can also be extended to higher dimensions

$$\hat{I}(X_1, ...X_m) = \psi(k) + (m-1)\psi(N) - \langle(\psi(n_{x1} + 1) + ... + (\psi(n_{x_m} + 1)]\rangle,$$

where $\langle ...\rangle$ is the average required, and therefore enables approximation of conditional mutual information between continuous variables (Tsimpiris et al,2012)

$$\hat{I}(X;Y|Z) = \psi(k) - \langle\psi[n_{xz}(i)] + \psi[n_{yz}(i)] - \psi[n_z(i)]\rangle, \tag{3.2.9}$$

where similarly, n_{xz}, n_{yz}, n_z are the number of observations within the same distance in their respective projected space. As it is relatively straightforward to calculate, the approximation is then worth considering as a potential independence measurement for a learning algorithm, at least during the conditional phase, where kernel methods are not readily available. Additionally, this approximation was shown to have lower bias as dimensionality of the increases (Vlachos and Kugiumtzis, 2010), and is less sensitive to sample size (Hu et al, 2011).

One of the main concerns regarding MI approximation through nearest neighbours methods is computation time due to finding neighbours. While the implementation is fairly straightforward, a considerable amount of time is spent searching for the neighbors, as it requires nested loops (Kraskov et al, 2004), and could be less ideal for larger datasets. However, Berrett and Samworth (2019) recently introduced a weighted version of the Kozachenko-Leonenko entropy estimator, which was developed into an R package (IndepTest - CRAN) and appears to calculate distances fairly quickly and efficiently. A downside is that initial testing shows that mutual information calculated using this approach appears to violate a fundamental property of entropy - that is, $H(X,Y) \leq H(X) + H(Y)$ -, even though it is only by a small margin.

Another issue to note is that there is no clear maximum value for Mutual Information apart from the highest possible value that can be obtained, $I(X,X)$, which is essentially $H(X)$, when the conditional entropy is 0. In broader terms, as one can define mutual information as

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X),$$

due to the symmetry of the quantity, the upper bound for MI is

$$\max I(X,Y) = \min(H(X), H(Y)).$$

Examining this may give some insight on deciding when an approximation for MI is close enough 0, therefore indicating pairwise independence, although estimating the marginal entropy for each random variable in addition to the nearest neighbour approximation could become computationally expensive. which can increase the complexity of the approximation. In addition, there is no similarly helpful guideline when it comes to Conditional MI, as the maximum possible value is even more obscure when considering three random variables. As for Conditional Mutual Information, from Equation 2.3.5, we have

$$I(X;Y|Z) = H(X,Z) + H(Y,Z) - H(X,Y,Z) - H(Z),$$

or in terms of Mutual Information,

$$I(X;Y|Z) = I(X;Y,Z) - I(X;Z),$$

this means Conditional MI is maximized when both the joint entropy $H(X,Y,Z)$ and marginal entropy $H(Z)$, or alternatively $I(X;Z)$ is as small as possible, which becomes

a complex minimization to carry out each time conditional independence is assessed. Therefore, in cases where the estimate is not clearly very close to 0, deciding whether conditional independence holds may become difficult. While normalized versions for both MI and Conditional MI have been proposed, it requires estimation of marginal entropies in the pairwise case, and some conditional entropies in the conditional case, which can make the approximation overly complex.

Overall, it seems that using the nearest neighbour method does produce an approximation for Mutual Information which is fairly intuitive, not too complex to calculate, and can be extended to finding Conditional Mutual Information, with decent computation speed – although higher dimensional data does slow the process down considerably. Therefore, it is also of interest to ensure that one such approximation that also follows all properties of entropy and mutual information is obtained, and incorporate it into a structural learning algorithm.

Chapter 4

4 The algorithm

The previous sections described the background and key methodology, focusing on the essential elements necessary. It is now time to bring them together to create a structural learning algorithm.

4.1 Pairwise phase

Starting from a complete graph, the first step is to establish pairwise independence relations. Overall, there were three potential approaches identified for doing so. Looking back at the kernel methods, while conditional independence tests for are not widely available for non-Gaussian continuous distributions, there are several kernel methods that are efficient at testing for pairwise independence. Therefore, to test each pair of variables, the pairwise test using Hilbert-Schimdt Independence Criterion (HSIC) (Peters et al, 2013) was chosen as one of the optional steps for the pairwise phase. In addition, as Mutual Information is also a potential candidate to measure independence, and the approximations for continuous variables can also be obtained, the Estimated Mutual Information was therefore also considered for use in the pairwise phase, either through ranking, or through a threshold. While the results when using HSIC tests in the pairwise phase are slightly more conservative (keeping more edges), the test indicating independence generally agrees with the result obtained using the threshold.

4.1.1 Choosing a value of K

As in most cases that make use of the nearest neighbour method, an important decision to make is the value of K. In this context, K signifies the number of nearest neighbours that are used to compare the average distance from the K-th neighbour in the joint space to the number of points within the same distance in the projected marginal spaces for each individual point. Therefore, it is important to examine the effect of choosing different values of K, as it can have an effect on the Estimated Mutual Information. To assess this, it can be useful to observe the maximum value of Mutual Information. We know that

$$I(X;Y) = H(X) - H(X|Y),$$

and therefore

$$I(X;Y) \leq H(X),$$

the entropy of X. However, as mentioned in section 2.3, calculating the entropy for a continuous variable can become too complex. Nevertheless, it may be possible to get a notion of what is close to the highest attainable value through approximating $\hat{I}(X;X)$. Then, without making changes to X, but varying the values of K the effect of the number of neighbours for the approximation can be observed more directly. Figure 4.1. illustrates these estimates for the same random variable - in this case, a sample of 100 from a standard Gaussian distribution -, but using values of one to fifteen for K.

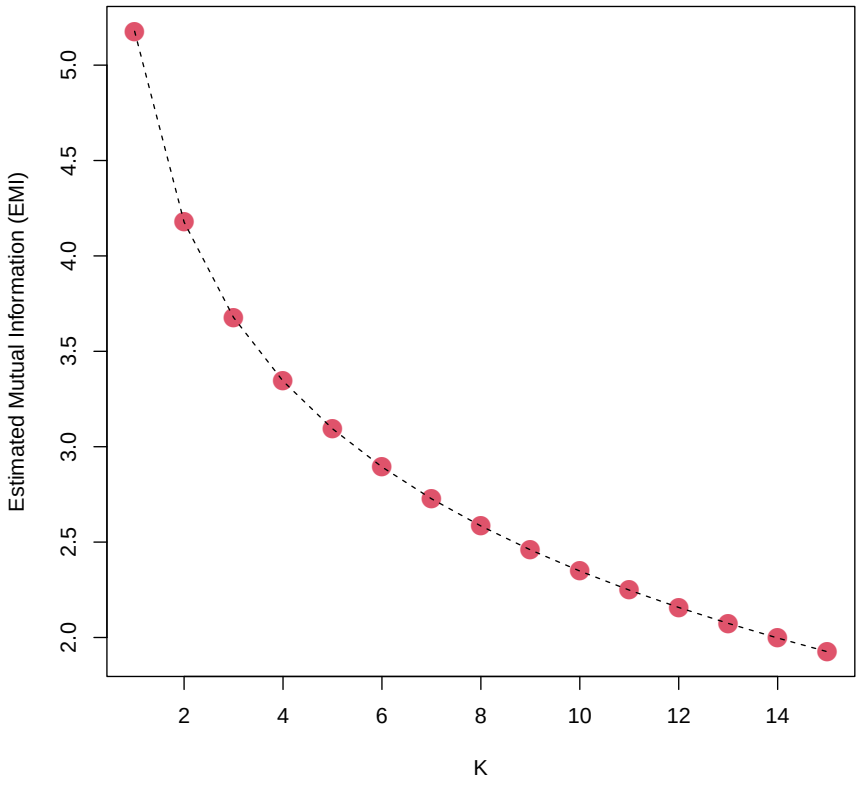


Figure 4.1: Highest observed EMI for different values of K

It seems apparent that as K increases, the estimates decline in value as well. When approximating Mutual Information through the nearest neighbour method, we have

$$\hat{I}(X;Y) = \psi(k) + \psi(n) - \langle(\psi(n_x + 1)) + (\psi(n_y + 1))\rangle.$$

A consistent decrease for the value of observed estimates can be explained by that as the value of K increases, so does the average number of points in the marginal space (which acts as the penalty during the approximation). In addition, the decrease also appears to hold for approximation of Conditional Mutual Information as well. It is therefore important to note that the choice of K may effect results using the nearest neighbour estimation and should potentially change the value of K based on the available sample size. As with most cases using nearest neighbour method, lower values of K are preferable when the sample size is small, and larger values may be preferable when the sample size is large, although it is very rare to go above a value of 10

4.1.2 Threshold or ranking

Another important factor that was considered for this method during the pairwise phase was deciding when to remove an edge. This is fairly straightforward when using the HSIC test of independence, removing edges between all random variables that the tests indicate to be independent. For the Estimated Mutual Information, this is more difficult, as a formal test is not readily available. One possibility is to find a suitable threshold by observing $\hat{I}(X;X)$ again for different values of K, and noting the percentages, as shown by Figure 4.2.

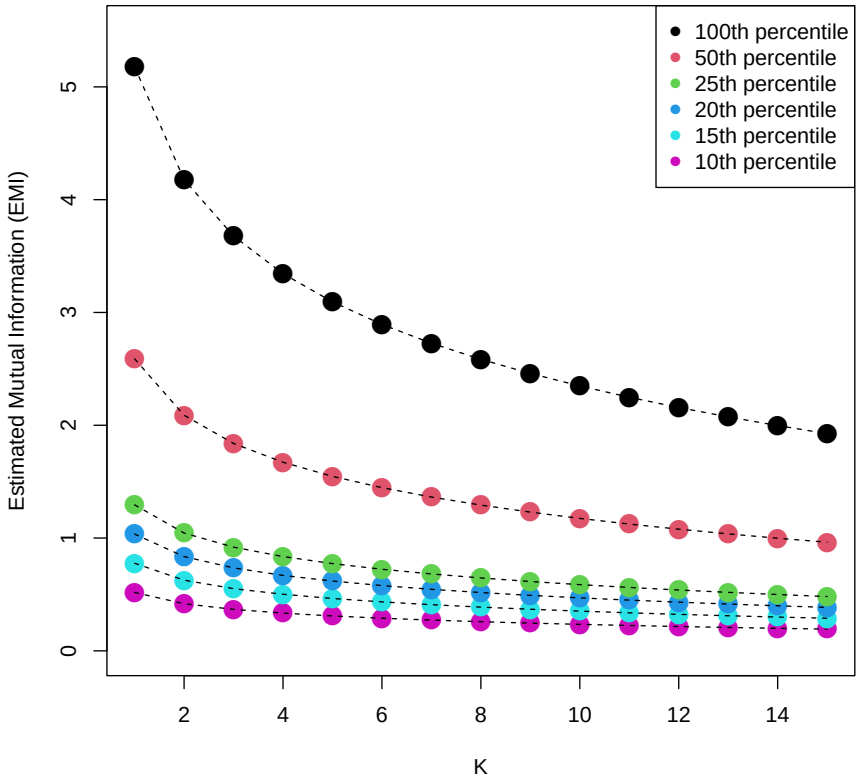


Figure 4.2: Percentages of highest observed EMI for different values of K

The figure suggests that while the value of K for this method is still important, the more the percentage of the highest observed $\hat{I}(X;X)$ is reduced, the more each curve starts to flatten out, further suggesting that a large value of K (above 8-10) may not be sensible. Therefore, when looking at the tenth percentiles, an initial threshold of 0.5 appears reasonable to consider EMI to be close enough to zero, as regardless of the value of K chosen, 0.5 is below the tenth percentile in each case. Apart from this initial value, it is also possible to manually set the threshold depending on context. Then, it becomes possible to and assign or remove edges between pairs of random variables accordingly. It is also possible to ensure that the threshold changes according to the value of K chosen, as well as conducting further checks of the highest observed estimates.

When it comes to assigning edges initially, it is also possible to approach the problem focusing more on the relevance of edges. Once the EMI is calculated for all pairs, it is straightforward to sort and rank the values for each pair. Then, even though the whole structure might not

be presented, if simplicity is more preferable, one can choose to assign edges only to some chosen number of pairs with the highest estimated amount of Mutual Information between them - while potentially maintaining a lower threshold.

4.2 Conditional phase

So far it has been established that through the same approximation, it is possible to obtain the Conditional Mutual Information, which is 0 if and only if conditional independence holds. Then, as the next step, it is required that CMI is checked where applicable. Having obtained a list of edges through the pairwise phase allows the observation of cliques. In this case, cliques of order 3 are checked, giving triplets of variables where all three nodes are connected by an edge.

A key point to consider is that while Mutual Information is symmetric, the Conditional Mutual Information changes depending on which variable is assumed as known – that is, while

$$\hat{I}(X_i; X_j|X_k) = \hat{I}(X_j; X_i|X_k),$$

$$\hat{I}(X_i; X_j|X_k) \neq \hat{I}(X_i; X_k|X_j).$$

In order to counter this issue, the approach taken is to estimate all three possible CMI for a triplet, and then checking whether the lowest one is close zero. While it was found that as the value K increases, estimates of Conditional Mutual Information also decrease, it much more difficult to find an indication what the upper bound may be for CMI. Therefore, choosing to be more conservative compared to the pairwise phase, half of the initial pairwise threshold, 0.25 is used to consider the lowest estimate of the triplet to be close enough to zero. This is then repeated for all remaining cliques of order 3, regardless which way the pairwise phase was conducted.

4.3 The algorithm

Gathering all these elements, the three combinations considered are describe below. A simple illustration is also included for the two main directions: using ranking, and using either HSIC tests or all estimated MI values for the pairwise phase. The next section then further demonstrates initial tuning through applying it to simulation data.

4.3.1 Using ranking

To use the ranking method, much like any learning algorithm, the random variables – assumed to be continuous, but not necessarily linearly related – are assigned their specific

edges. Then, the value of K needs to be specified for the nearest neighbour approximation, as well as the maximum number of edges to be assigned. Then, as EMI is calculated for each pair of variables, the pairs with the highest estimates are selected according to the specified maximum number of edges. Then, for the conditional phase, each cliques of order 3 is examined, calculating the Estimated CMI for each possibility – a pair being independent given a third variable, conditioning on each variable of the triplet –, and examining the lowest of the three. Then, if the value is deemed close enough to zero by the threshold, the edge between the pair that is conditioned on the third variable is removed. The algorithm is outlined in Algorithm 3.

Algorithm 3 Edges assigned through ranked mutual information, then eliminated through conditional mutual information

Pairwise phase

1. Take random variables X_i and assign set of vertices V_i representing them, specify k and the number of initial edges to assign e_{max}.

2. For each pair (X_i, X_j),calculate MI estimate $m_{ij} = \hat{MI}(X_i, X_j)$, where $i \neq j$

3. Sort the estimates in descending order into ranks r.

4. Assign edge e_{ij} between V_i and V_j,using the number of pairs e_{max} from r

Conditional phase

1. Obtain cliques of order 3.

2. For each triplet (X_i, X_j, X_k), calculate and sort CMI estimates $c_{ijk} = C\hat{M}I(X_i, X_j, X_k) = \hat{I}(X_i; X_j|X_k) = \psi(k) - \langle\psi[n_{X_iX_k}(i)] + \psi[n_{X_jX_k}(i)] - \psi[n_{X_k}(i)]\rangle$, as well as c_{ikj} and c_{jki}, where $i \neq j \neq k$.

3. If the lowest CMI estimate $\hat{I}(X_i; X_j|X_k)$ is close to zero, remove edge between V_i and V_j.

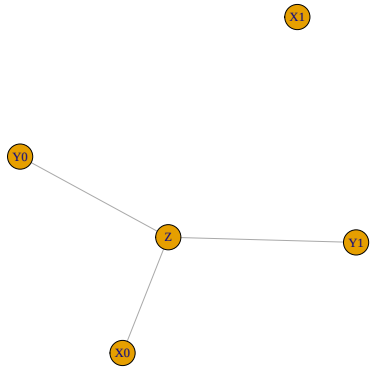


Figure 4.3: Structure by using Algorithm 3, using top 3 MI value. No cliques of order 3 were found, showing Z having a pairwise dependent relationship with X0, Y0 and Y1, X1 being independent of all other variables

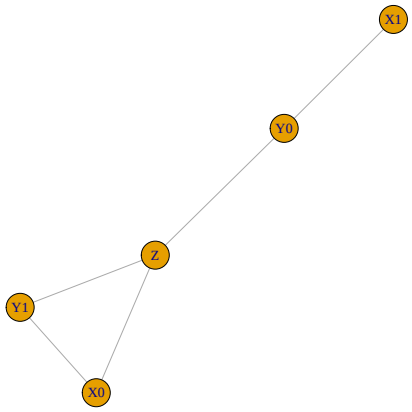


Figure 4.4: Structure by using Algorithm 3, using top 5 MI value after the pairwise phase. A link between Y0 and X1 is shown, as well as a link between Y0 and Z. A clique of order 3 between Z, Y1 and X0 was found.

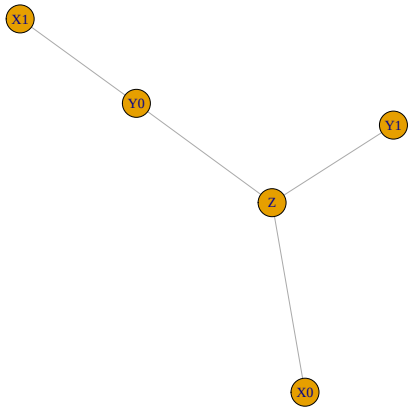


Figure 4.5: Structure by using Algorithm 3, using top 5 MI value after the conditional phase. The edge between X0 and Y1 was removed after observing CMI estimates.

The illustrations on Figure 4.3 to Figure 4.5 displays the output for the same simulated data (setup described in Section 5), using 3 and 5 as the maximum number of edges to be assigned. When choosing 3, there were no cliques of order 3 to examine, therefore ending at the pairwise phase. When choosing 5 as the maximum number of edges, there was a clique observed, and an edge was removed during the conditional phase, recognizing conditional independence relations for the triplet, leaving 4 edges.

Overall, the ranking method is almost entirely influenced by the maximum number of edges chosen. This is not specifically of interest, as the focus is on establishing a method that works well for continuous variables not in a Gaussian BN, but could potentially be useful for high-dimensional data, where the number of edges would be too high. In that case, one may choose to assign the number of edges in advance, as a means to examine and visualize the relationships that are the most relevant – potentially introducing a threshold as well to ensure a minimum amount of Mutual Information, so that the chosen edges are not completely relative to their rank.

4.3.2 Using threshold

Following the ranking method, it is also possible to use the Estimated Mutual Information through the use of a threshold. In this case, after assigning the vertices, and specifying the value of k, instead of the maximum number of edges, a threshold value is used. The main difference then is that the number of edges in the graph are not determined in advance. This can be a constant – initially using 0.5 after observing Figure 4.2 –, or can be calculated as a percentage of the highest observed EMI given the value of k. Then, once EMI is calculated for all pairs of random variables, an edge is assigned if the estimate is over the threshold. This is then followed by the conditional phase the same way as for the ranking method. This version of the algorithm is shown in Algorithm 4.

Algorithm 4

Pairwise phase

1. Take random variables X_i and assign set of vertices V_i representing them, specify k and a threshold value

2. For each pair (X_i, X_j),calculate MI estimate $m_{ij} = \hat{MI}(X_i, X_j)$, where $i \neq j$

3. If $m_{ij} = \hat{MI}(X_i, X_j)$ is greater than the threshold value, assign edge e_{ij} between V_i and V_j.

Conditional phase

4. Obtain cliques of order 3.

5. For each triplet (X_i, X_j, X_k), calculate and sort CMI estimates $c_{ijk} = C\hat{MI}(X_i, X_j, X_k) = \hat{I}(X_i; X_j|X_k) = \psi(k) - \langle\psi[n_{X_iX_k}(i)] + \psi[n_{X_jX_k}(i)] - \psi[n_{X_k}(i)]\rangle$, as well as c_{ikj} and c_{jki}, where $i \neq j \neq k$.

6. If the lowest CMI estimate $\hat{I}(X_i; X_j|X_k)$ is close to zero, remove edge between V_i and V_j.

4.3.3 Using HSIC pairwise testing

Finally, it is also possible to use kernel methods in the pairwise phase. Initially, the independence tests based on the Hilbert-Schimdt Independence Criterion are used, testing each pair of variable for independence and assigning edges accordingly. Nevertheless, this variant still needs to have the value of k specified, as the conditional phase is the same as before, and the parameter is required to estimate Conditional Mutual Information.

Figures 4.6 to 4.9 illustrate a key point for the method using a threshold and the method using HSIC tests. While HSIC tests can be more conservative than using EMI through the pairwise phase – as when using a formal test, pairwise independence is likely to become less frequent – , it is plausible that both approaches lead to same structure. Then, as the conditional phase is the same, the final output is the same as well. While this is not guaranteed, it led to a slight possible modification: allowing the option to carry out the pairwise phase for both methods at the same time, then comparing the assigned edges. If the structures are the same, the conditional phase only needs to be carried out once, otherwise it provides a point of comparison between the two methods, outlined in Algorithm 5.

Algorithm 5

Pairwise phase

1. Take random variables X_i and assign set of vertices V_i representing them, specify k and a threshold value

2. For each pair (X_i, X_j), conduct HSIC test.

3. If the obtained test statistic is larger than the critical value of the Gamma approximation, assign edge e_{ij} between V_i and V_j.

Conditional phase

4. Obtain cliques of order 3.

5. For each triplet (X_i, X_j, X_k), calculate and sort CMI estimates $c_{ijk} = C\hat{M}I(X_i, X_j, X_k) = \hat{I}(X_i; X_j|X_k) = \psi(k) - \langle\psi[n_{X_iX_k}(i)] + \psi[n_{X_jX_k}(i)] - \psi[n_{X_k}(i)]\rangle$, as well as c_{ikj} and c_{jki}, where $i \neq j \neq k$.

6. If the lowest CMI estimate $\hat{I}(X_i; X_j|X_k)$ is close to zero, remove edge between V_i and V_j.

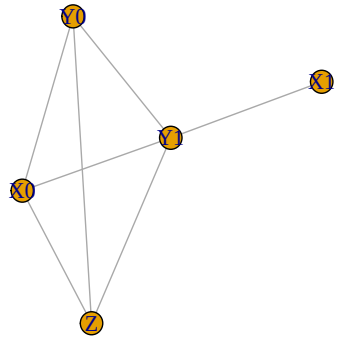


Figure 4.6: Graph obtained after pairwise phase using Algorithm 4 and 5. Both algorithms showed the same links in this phase, with cliques of order 3 between all variables except X1, which was only linked to Y1

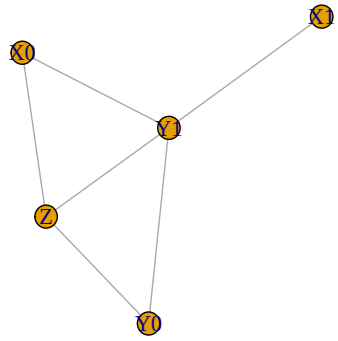


Figure 4.7: Graph obtained after checking first clique. Examining the CMI estimates of the first triplet, the link between X0 and Y0 was removed, leaving only two more cliques.

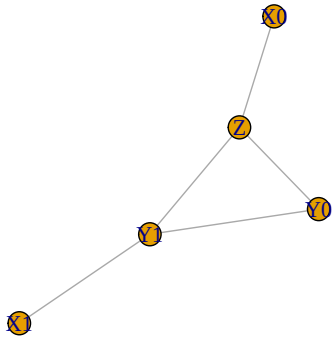


Figure 4.8: Graph obtained after checking second clique. Examining the CMI estimates of the second triplet, the link between X0 and Y1 was removed, leaving a final clique

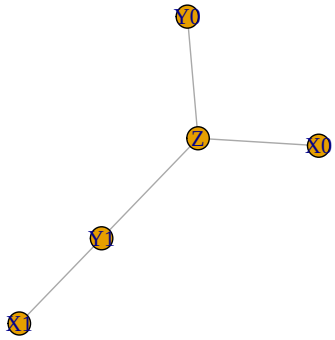


Figure 4.9: Graph obtained after checking final clique. Examining the CMI estimates of the final triplet, the link between Y1 and Y0 was removed, leaving no cliques of order 3.

Figures 4.6 to 4.9 demonstrate the process of going through the two phases. The initial graph structure is obtained by either estimating MI for all pairs of random variables, then compar-

ing it to the threshold, or by conducting an HSIC test for each pair. In this case, the pairwise phase returned the same structure. With the exception of X_1, which was only connected to one other node (Y_1),the remaining random variables formed a complete graph, resulting in four cliques of order 3. Then, each clique is examined, approximating CMI by conditioning on each of the three nodes, and choosing the lowest estimate to compare to a threshold value.

Figure 4.6 shows a fairly common occurrence, where removing one edge actually eliminates two cliques at the same time. This could raise the question of conflicting results, in a potential scenario where examining one clique may suggest removing the edge, while the looking at the other clique may not do the same. In this context, it was decided that even if such a situation occurs, having a significantly low estimate for CMI in any cliques should justify the deletion of such and edge. The following figures then visualize investigation of the remaining two cliques, each time finding that the estimate was low enough to remove further edges, resulting in the final output. This graph had no cliques of order 3 remaining and therefore give a clearer insight to the links between the 5 random variables. Without any additional context, Z having the highest degree of centrality suggests that is likely a crucial random variable in the structure. In order to provide more information on the workings of this method, and the particular structure, the next section describe the simulation which was the main testing ground for the algorithm, using MI estimates in the pairwise phase.

Chapter 5

5 The Bellot setup

5.1 Introduction

In order to assess the algorithms using MI estimators, a synthetic data setup (Bellot 2019) was implemented. Originally, the setup was focused on the hypothesis testing problem of detecting conditional dependence, with a focus on high-dimensional feature spaces. The structure is defined under two hypotheses: the null hypothesis H_0, where Z is the common cause of X and Y; and the alternative hypothesis H_1, where Y is a common effect of X and Z. Using the post non-linear noise model (noted by functions f, g, h and noise variables $\epsilon_f, \epsilon_g, \epsilon_h$), this structure was enforced as follows:

$$H_0 : X = f(A_f Z + \epsilon_f), Y = g(A_g Z + \epsilon_g)$$

$$H_1 : Y = h(A_h Z + \alpha X + \epsilon_h).$$

The functions f, g, h are specified as part of the setup, as well as the distributions of X, Y, Z. The matrix dimensions of A are such that X, Y are univariate, matrix entries as well as parameter α are generated in the interval $[0, 1]$, and lastly, the noise variables ϵ – $\epsilon_f, \epsilon_g, \epsilon_h$ being noise associated with f, g, h respectively– are 0 on average with variance 0.025. The distributions of X, Y, Z and ϵ, and the complexity of dependencies via f, g and h can then be tuned to make performance comparisons in different settings. After generating samples for each random variable under both H_0 and H_1, Figure 5.1 depicts the structure incurred by the setting using a graphical representation, showing links Z and X under H_0 (X0), Y under H_0 (Y0), Y under H_1 (Y1), as well as a link between X and Y under H_1 (X1, Y1).

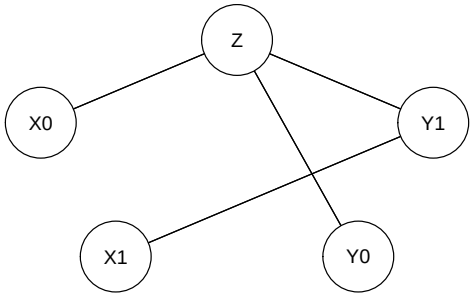


Figure 5.1: The structure enforced by the setup, Z having links with X0 and Y0, representing H_0, as well as Y_1 having links with X_1 and Z respectively, representing H_1.

The settings used to test the algorithm were Multivariate Gaussian - where f, g and h were fixed to be the identity functions - , and then some arbitrary distributions by sampling f, g and h from $[x^3, \tanh x, exp(-x)]$, while sampling Z from a Gaussian distribution. The aim then was to examine if one of the new algorithms would be able to identify this enforced structure using different examples.

5.2 Application of the algorithm

For each setting, MI estimators were calculated under H_0 and H_1, then compared to the output of the PC-algorithm – described as a constraint-based structural learning algorithm in Section 2.1.4 – by examining how closely they match Figure 5.1, the structure that is enforced by the setup. This was represented by five nodes: one for Z, then a node for X and Y under H_0 and H_1. While this is not a complete comparison with only using estimators for MI during the pairwise phase, it is informative as an initial step to compare the ranking method to using a threshold. Throughout these examples, the value of k was chosen to be 3. As current learning algorithms are still mainly limited to discrete and Gaussian data, it was expected that mutual information estimates would pick up on relationships that the PC-algorithm cannot, although it is likely that for Gaussian settings, the PC-algorithm performs better.

5.2.1 Multivariate Gaussian - Algorithm 3

For this part of the setup, the functions f,g and h are set to be the identity function. We set

$$Z \sim N(0,1), X_1 \sim N(0,1), \epsilon_{(.)} \sim N(0, 0.025).$$

Then, we have

$$X_0 = Z + \epsilon_f, Y_0 = Z + \epsilon_g$$

$$Y_1 = Z + X_1 + \epsilon_h$$

Using a sample size of 100, Figure 5.2 shows the structure learned by the PC algorithm and compare it to the ranking method using the 3 and 5 highest MI estimates to assign the edges in the pairwise phase.

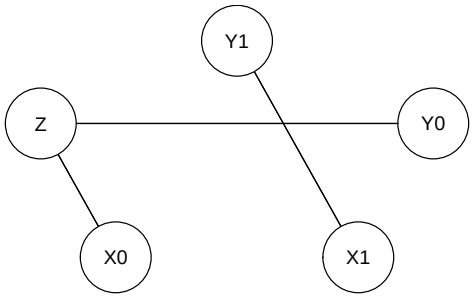


Figure 5.2: Structure learned by applying the PC algorithm, where f, g and h are the identity function. The link between Z and Y1 was not found, while the remaining edges match with Figure 5.1.

As expected, since the PC-algorithm generally performs well when used in Gaussian settings, the structure under H_0 was discovered using it. In addition, the link between X_1 and Y_1 was also discovered, but not the link between Z and Y_1. The output therefore identifies the correct structure under H_0, but not under H_1. Then, the MI estimates were calculated and sorted. Here one can decide the number of overall edges, after checking triplets for the Conditional MI estimates. Figure 5.3 shows the structure by ranking and then using the top 3 MI value, Figure 5.4 shows the same ranks, but using the top 5 MI value.

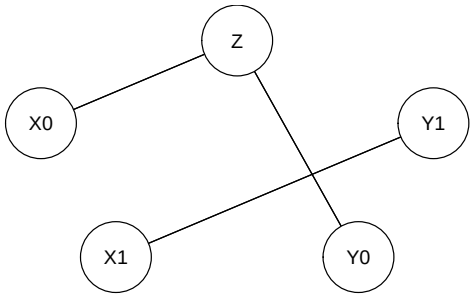


Figure 5.3: Structure learned by applying Algorithm 3 and using top 3 MI value, where f, g and h are the identity function. The link between Z and $Y1$ was not found.

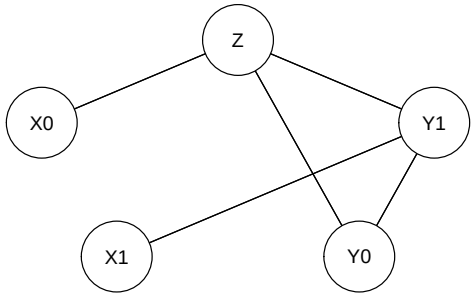


Figure 5.4: Structure learned by applying Algorithm 3 and using top 5 MI value, where f, g and h are the identity function. The link between Z and $Y1$ was found, but an additional edge was added between $Y1$ and $Y0$.

We see that the enforced structure was mostly discovered using the 3 highest MI estimates –with the same output as the PC-algorithm –, although the link between Z and Y_1 was not included, as a fourth edge was not allowed. This highlights the main limitation of choosing the number of edges allowed in advance, as this may lead to missing connections that would otherwise be picked up. In contrast, using 5 as a maximum number of edges did lead to a link between Z and Y_1, it also introduced a clique of three where no edges could be removed during the conditional phase.

While none of the outcomes were completely as hoped, it does point out the potential to choose what is more important: recognizing a link that is expected to be there at the cost of additional edges, or not recognizing a dependency, but eliminating cliques at the same time. Depending on context, it may also be of interest to find the most relevant relationships between random variables, which might justify identifying a structure using a predetermined number of edges with the highest MI estimates.

5.2.2 More complex relationships - Algorithm 3

After the Multivariate Gaussian setup, different settings were explored. We still have $\epsilon_{(.)} \sim N(0, 0.025)$ and return to sample

$$Z \sim N(0,1), X_1 \sim N(0,1).$$

Then, we have

$$X_0 = f(Z + \epsilon_f), Y_0 = g(Z + \epsilon_g)$$

$$Y_1 = h(Z + X_1 + \epsilon_h).$$

The main difference is that in the original Bellot setup, f, g and h are randomly selected from $[x^3, \tanh x, exp(-x)]$. For demonstration, Figures 5.5 shows the structure learned by applying the PC algorithm, while 5.6 and 5.7 show results of using the top 3 and 5 MI estimates respectively, when f, g and h are fixed to be $\tanh x$.

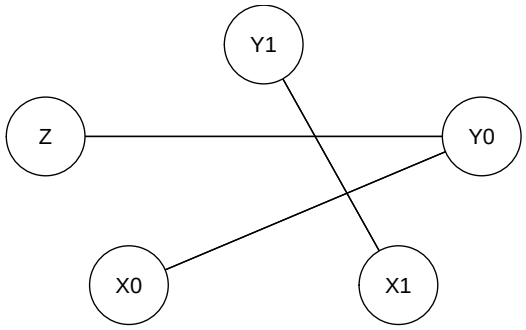


Figure 5.5: Structure learned by applying PC algorithm, where f, g and h are the $\tanh(x)$ function. The link between Z and $Y1$ was not found, and the edges between $Z, X0$ and $Y0$ are different from Figure 5.1 as well.

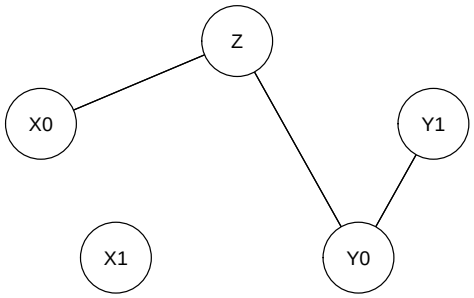


Figure 5.6: Structure learned by applying Algorithm 3 and using top 3 MI value, where f, g and h are the $\tanh(x)$ function. The edges between $Z, X0$ and $Y0$ match Figure 5.1, but an edge between $Y0$ and $Y1$ was added, with no links between $Z, X1$ and $Y1$.

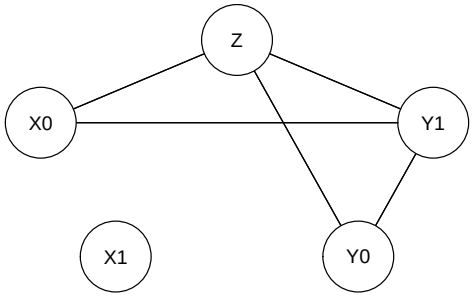


Figure 5.7: Structure learned by applying Algorithm 3 and using top 5 MI value, where f, g and h are the $\tanh(x)$ function.The edges between $Z, X0$ and $Y0$ match Figure 5.1, but an edge between $Y0$ and $Y1$, as well as between $X0$ and $Y1$ was added, $X1$ having no links with $Y1$ or Z.

Both the PC algorithm and using ranking system missed some of the enforced links, and using 5 edges for ranking did leave some cliques that did not disappear. So far, this appears to remain fairly similar even when f,g and h were left to be randomly picked. Overall, this highlights that the ranking method may not be particularly helpful with a setting with few random variables, but especially as using 3 as the maximum number of edges does not enable the method to learn the ideal (enforced) structure. While some variant might be useful for high-dimensional data, the algorithm using a threshold of 0.5 was used from here on out.

5.2.3 More complex relationships - Algorithm 4

As the first comparison, we draw a new sample where

$$Z \sim N(0,1), X_1 \sim N(0,1), \epsilon_{(.)} \sim N(0, 0.025).$$

Like before, he have

$$X_0 = f(Z + \epsilon_f), Y_0 = g(Z + \epsilon_g)$$

$$Y_1 = h(Z + X_1 + \epsilon_h).$$

with f, g and h fixed to be $\tanh x$. For this sample, the results of the PC algorithm are shown in Figure 5.8, while results of using the 0.5 threshold for EMI in the pairwise phase are shown in Figures 5.9 and 5.10.

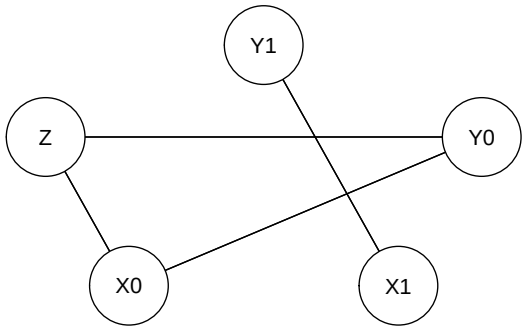


Figure 5.8: Structure learned by applying PC algorithm, where f, g and h are the $\tanh(x)$ function. The link between Z and $Y1$ was not found, and $Z, X0$ and $Y0$ formed a clique of order 3.

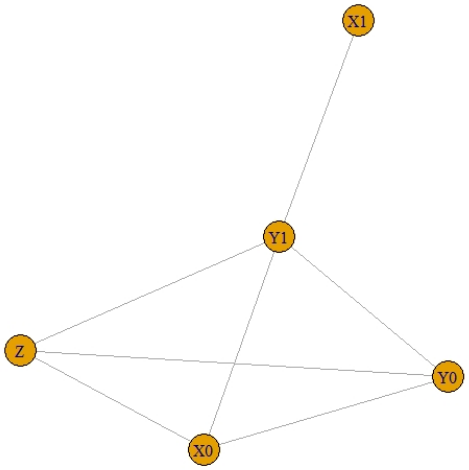


Figure 5.9: Structure learned by applying Algorithm 4, using 0.5 threshold, where f, g and h are the $\tanh(x)$ function after the pairwise phase. Several cliques of order 3 are present between variables, except for $X1$.

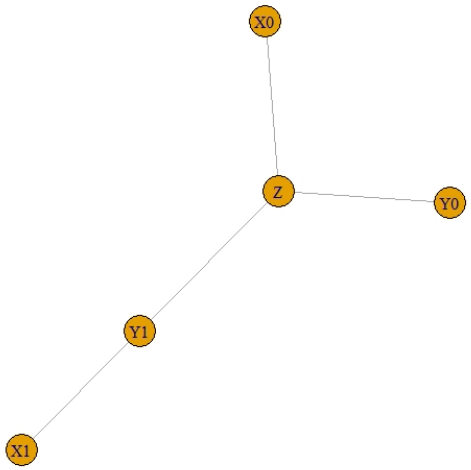


Figure 5.10: Structure learned by applying Algorithm 4, the conditional phase. For all cliques of order three an edge could be removed, matching Figure 5.1.

The PC algorithm ended up recognizing the several of the intended links. However, the dependency between Z and Y_1 was not picked up this time either. In addition, the dependencies of Z, Y_0 and X_0 are also more complex, retaining a clique of order 3. In contrast, using the threshold resulted in the ideal structure recognized, with the link between Z and Y_1 present, and the edge between X_0, Y_0 removed.

It then appears that the using MI estimates with a threshold may be preferable to the PC-algorithm when functions f, g, h are fixed to be $tanh(x)$. It is also worth of note that using this new sample, the output for the PC-algorithm was different than before. Therefore, to further investigate, the same distributions were sampled using a different seed. The results are shown in Figures 5.11 to 5.13.

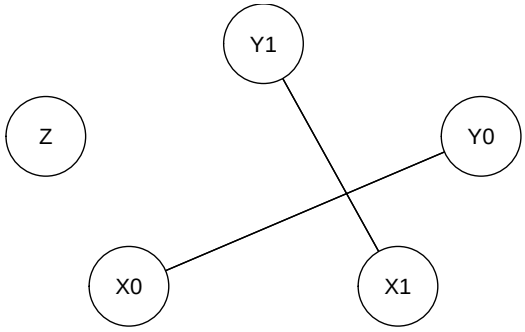


Figure 5.11: Structure learned by applying PC algorithm using a new seed, where f, g and h are the $\tanh(x)$ function. No links with Z were found

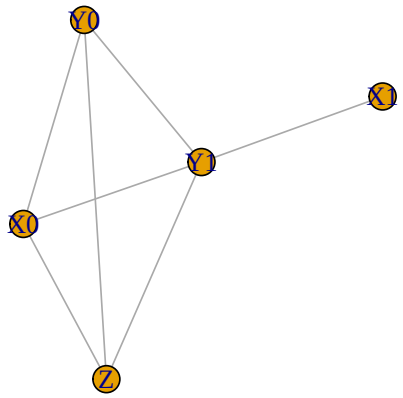


Figure 5.12: Structure learned by applying Algorithm 4, using a 0.5 threshold and a new seed, where f, g and h are the $\tanh(x)$ function after the pairwise phase. The results match with those using the previous seed.

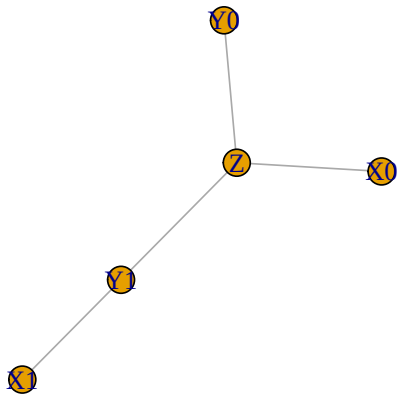


Figure 5.13: Structure learned by applying Algorithm 4, after the conditional phase. The results match with those using the previous seed.

In this case, the PC-algorithm again returned a different output after being applied to a new sample. In this case, the dependencies that are learned are somewhat expected, with the important difference that Z had no edges assigned to it, which seems to contradict the design of the setup. On the other hand, using a threshold for MI estimates in the pairwise phase returned the same output as before, which also matched the ideal, enforced structure. This seems to suggest that at the very least, when f, g and h are fixed to be $tanh(x)$, using the new method may be preferable.

As the next example, we have a similar sample where

$$Z \sim N(0,1), X_1 \sim N(0,1), \epsilon_{(.)} \sim N(0, 0.025).$$

Like before, he have

$$X_0 = f(Z + \epsilon_f), Y_0 = g(Z + \epsilon_g), Y_1 = h(Z + X_1 + \epsilon_h),$$

but while f and g are fixed to be $\tanh x$, h is now fixed to be x^3. For this sample, the results of the PC algorithm are shown in Figure 5.14, while the results of using a threshold for EMI

are shown in Figures 5.15 and 5.16.

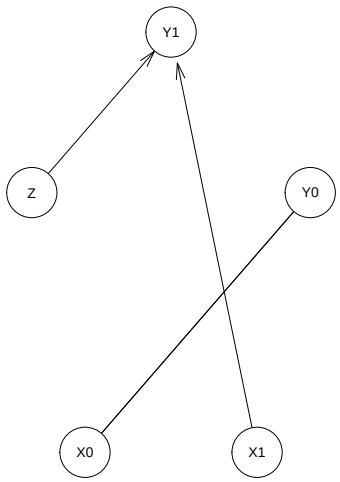


Figure 5.14: Structure learned by applying PC algorithm, where f, g are the $\tanh(x)$ function and h is the x^3 function. A link between $X0$ and $Y0$ was found instead of separate link with Z. The algorithm picked directed edges to $Y1$ from Z and $X1$.

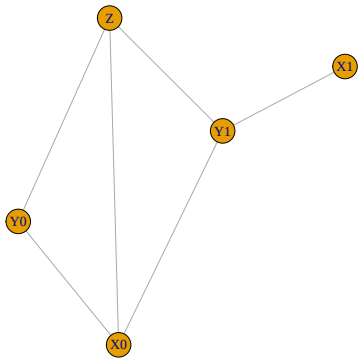


Figure 5.15: Structure learned by applying Algorithm 4, using 0.5 threshold for EMI, where f, g are the $\tanh(x)$ function and h is the x^3 function after the pairwise phase. Cliques between $Z, Y0, X0$ and $Z, Y1, X0$ were identified.

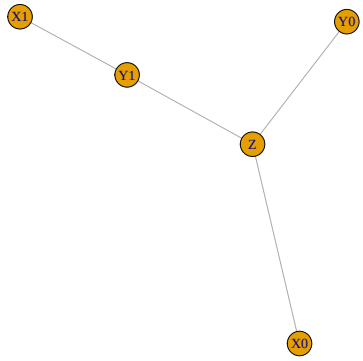


Figure 5.16: Structure learned by applying Algorithm 4, after the conditional phase. The outcome matches Figure 5.1.

In this case, we see that while the PC algorithm, as one of the few occasions, picked a few directed edges, recognising parts of the enforced structure (between Z, X1 and Y1). This is possible if the algorithm can establish strong enough dependencies . The algorithm PC algorithm only missed the link between Z and X_0, examining the cliques using the 0.5 threshold for MI estimates results in the intended output.

5.3 Repeated applications using different seeds

Although there have been a few examples where the new method appears to perform better than the PC-algorithm, it was also noted that drawing a new sample from the same distribution could lead to different results. As a further step to investigate this, the algorithm was run ten times in succession, using different samples - with 100 observations each - from the same distributions. In each instance, the remaining edges were recorded, then an overall graph would be created, indicating the number of time an edge remained by their width (number of times recorded) and color - dark blue: edge always remains; yellow: edge remains 7-9 times; green: edge remains 5-6 times; light blue: edge remains 3-4 times; orange: edge remains 1-2 times. While this often does not give a notion of each individual structure, it does highlight overall patterns across trials. Initially, the case where f,g, and h are identity functions are examined, shown in Figure 5.17 for results using the PC algorithm, and Figure 5.18 using the new method with a 0.5 threshold in the pairwise phase:

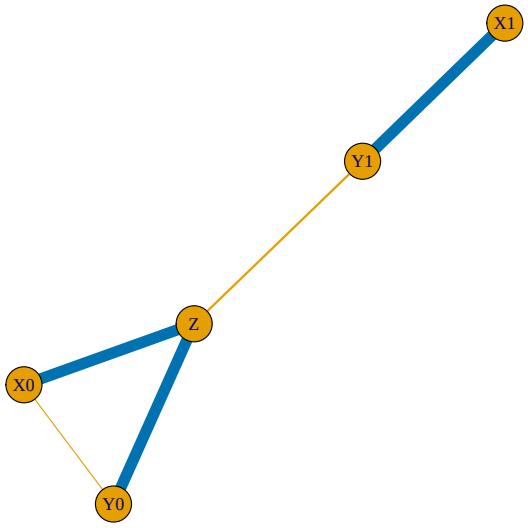


Figure 5.17: Overall structure after ten repeats using PC algorithm; f,g and h are identity functions. The link between Z and $Y1$ is often missed, while an additional link is found occasionally between $X0$ and $Y0$.

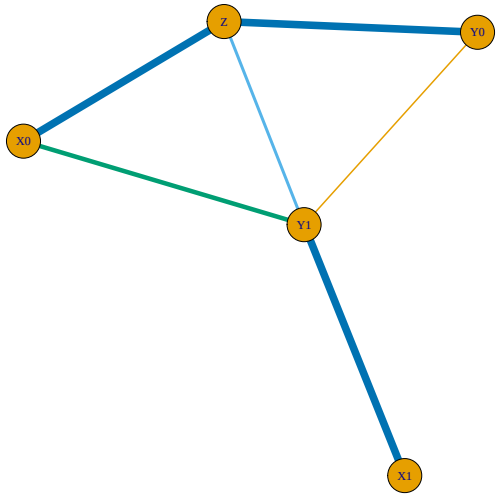


Figure 5.18: Overall structure after ten repeats using Algorithm 4; f,g and h are identity functions. The link between Z and $Y1$ are missed on occasions, with additional link between $Y1$ and $X0$ or $Y0$.

As expected, the PC-algorithm performs quite well in the first, Multivariate Gaussian case, recognizing conditional independence between X0 and Y0 most of the times, although the link between Z and Y1 are also missed most of the time. The new method picks up the a link between Z and Y1 more often, however this is often through X0. This suggests that while cliques of order 3 are rare, it does not necessarily lead to identifying the enforced structure. To further investigate, these reruns were also performed for several other configurations of $\{exp(-x), tanh(x), x^3\}$ for f,g and h. While these figures do not allow interpretation on an individual structure, it does help by reflecting on how frequently and edge is assigned throughout different trials.

Shown in Figures 5.18 and 5.19, the reruns further suggest that the new method may be better at recognizing the intended structure when f, g and h are fixed to be $tanh(x)$. The PC-algorithm was shown to never establish conditional independence relations between Z, X_0 and Y_0, and rarely linked Z and Y_1. In addition, a link between X_0 and Y_1 was recorded on occasion.

On the other hand, using a 0.5 threshold for EMI led to learning the ideal structure most of the time, with only some rare instances where instead of linking Y_1 with Z, it was linked

with X_0 instead.

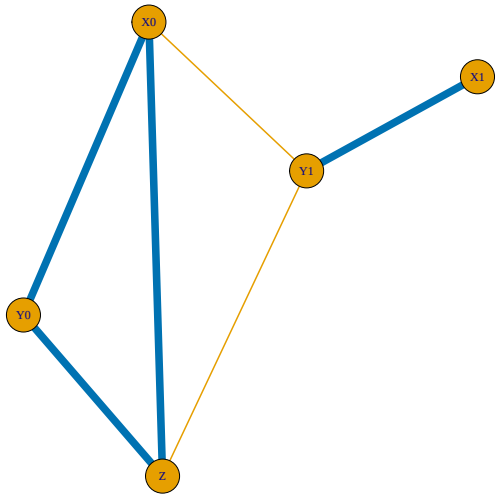


Figure 5.19: Overall structure after ten repeats using PC algorithm; f,g and h are $tanh(x)$ functions. The link between Z and $Y1$ is often missed, with occasional link found between $Y1$ and $X0$, and consistent link found between $X0$ and $Y0$.

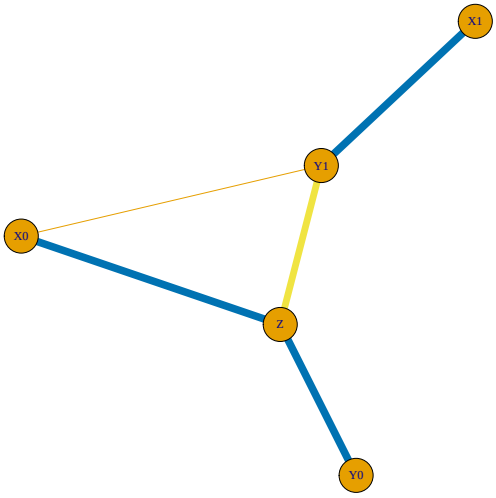


Figure 5.20: Overall structure after ten repeats using Algorithm 4; f,g and h are $tanh(x)$ functions. The outcome almost always matches Figure 5.1, occasionally finding a link between $X0$ and $Y1$.

This seems to suggest that at the very least, this method may be able to identify links by the $tanh(x)$ function fairly well. The following configuration was therefore similar, but h was chosen to be the x^3 function, the outputs shown in Figures 5.20 and 5.21.

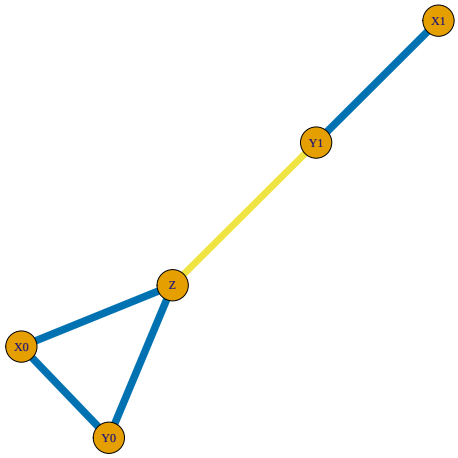


Figure 5.21: Overall structure after ten repeats using PC algorithm; f,g are $tanh(x)$ functions, and h is the x^3 function. A link between $X0$ and $Y0$ is consistently found, while the link between Z and $Y1$ are occasionally missed.

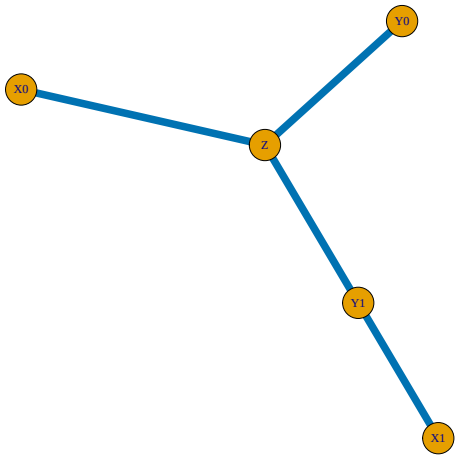


Figure 5.22: Overall structure after ten repeats using Algorithm 4; f,g are $tanh(x)$ functions, and h is the x^3 function. The outcome consistently matches Figure 5.1.

Looking at these figures, the PC-algorithm performed fairly well in this case as well. This time, the link between Z and Y_1 was discovered quite often. However, much like in previous cases, the conditional dependency was not established in the clique (Z, X_0, Y_0). The improvement in performance does appear to relate to the change in function h, which directly influences how Y_1 is drawn. After examining the output for the PC-algorithm, the structure using 0.5 threshold for EMI was obtained. The result matched the intended outcome, seemingly unaffected by the change in function h.

As a final check for using $tanh(x)$ for f and g, the next configuration fixed h to be $exp(-x)$. From previous results, the main difference is expected to be between Z and Y_1. The results for these reruns are shown in Figures 5.22 and 5.23.

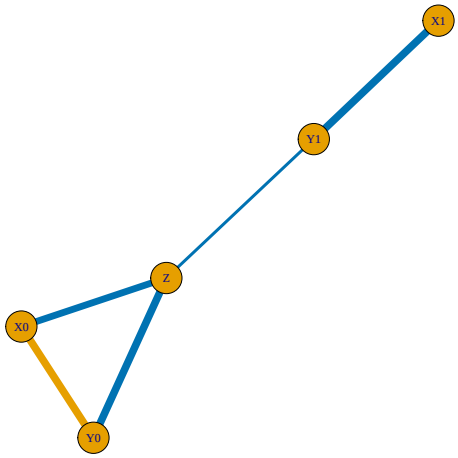


Figure 5.23: Overall structure after ten repeats using PC algorithm; f,g are $tanh(x)$ functions, and h is the $exp(-x)$ function. The link between Z and $Y1$ are often missing, and the link between $X0$ and $Y0$ often present.

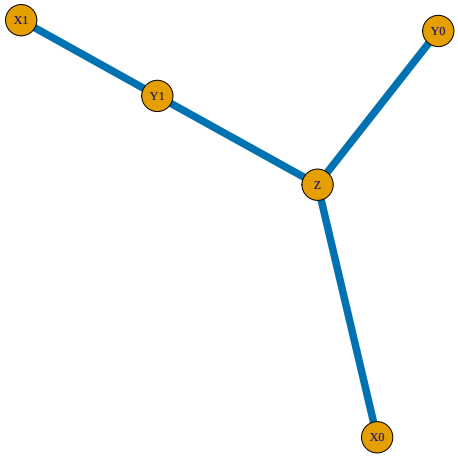


Figure 5.24: Overall structure after ten repeats using Algorithm 4; f,g are $tanh(x)$ functions, and h is the $exp(-x)$ function. The outcome consistently matches Figure 5.1.

As expected, the PC-algorithm performed well, although slightly less worse than before. The change to h appears to have resulted in the link between Z and Y_1 to be rarely recognized. However, contrary to previous configurations, there have been a few instances where the clique of order 3 had an edge removed. On the other hand, the new method appears to perform very well in these reruns as well, picking the enforced structure all ten times.

After observing these trials, the next step was to move further away from the commonality of previous runs, that is, choosing functions for f and g other than $tanh(x)$. The first of such configurations fixed f and g to be the x^3 function, while this time h was fixed to be $tanh(x)$. The results of the PC-algorithm and the method using 0.5 threshold for EMI are shown in Figures 5.24 and 5.25.

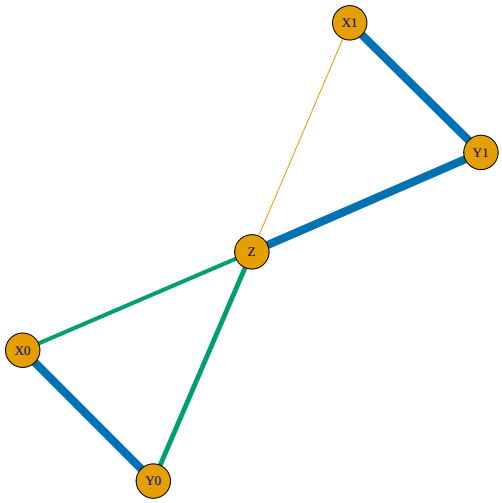


Figure 5.25: Overall structure after ten repeats using PC algorithm; f,g are x^3 functions, and h is the $tanh(x)$ function.

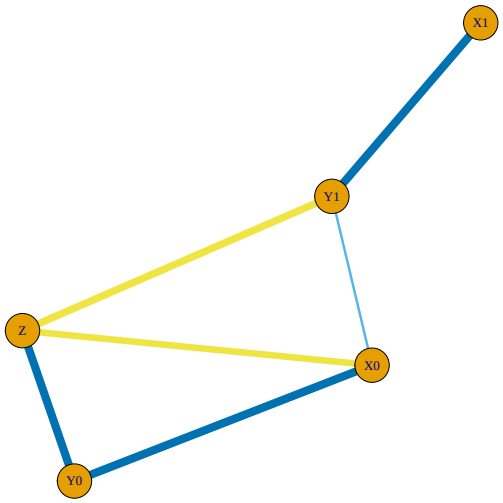


Figure 5.26: Overall structure after ten repeats using Algorithm 4; f,g are x^3 functions, and h is the $tanh(x)$ function

The results for these reruns are more difficult to interpret, as neither of the algorithms performed ideally. While these figures do not enable commenting on individual samples, there are some points to consider. First, the PC-algorithm recognized the link between Z and Y_1 every time, which is a big change to previous configurations. It also appears that the clique (Z, X_0, Y_0) is often reduced. However, the constant link is between X_0 and Y_0, which does not match the ideal structure.

On the other hand, the new method did not find the ideal structure during this trial. While the link between Z and Y_1 is present most of the time, the clique of order 3 remained just as often. It is then challenging to compare the two methods, as both of them led to outcomes different from the enforced, intended structure. It may depend on context or intent to choose which is more important to avoid: eliminating an edge that maybe should not be eliminated, or keeping an edge when it should be removed. Regardless, in these setups, it may also indicate that the new method is more suitable for certain type of relationships.

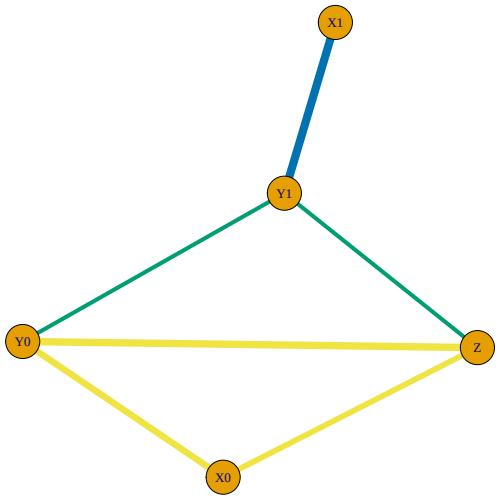


Figure 5.27: Overall structure after ten repeats using PC algorithm; f is the $exp(-x)$ function, g and h are the x^3 function

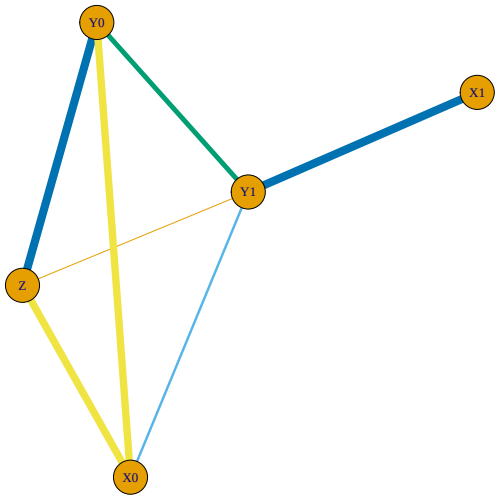


Figure 5.28: Overall structure after ten repeats using Algorithm 4; f is the $exp(-x)$ function, g and h is the x^3 function

The following configuration fixed f to be the $exp(-x)$ function, while g and h were fixed to be the x^3 function, the graphs shown in Figure 5.26 an Figure 5.27. In this case, both methods performed rather poorly compared to previous runs. The clique of order 3 for the variables under H_0 have almost always remained. For the output from the PC algorithm, the link between Z and Y_1 was picked up roughly half of the time, where the method using the 0.5 threshold for EMI did so less frequently. While it is not feasible to interpret these figures as the actual structure learned, it appears fairly clear that the PC-algorithm performed better in this case – although still not matching the ideal output.

As the next step in examining different configurations, each function was picked to be unique: f was fixed to be the $exp(-x)$ function, g to be x^3 and h to be $tanh(x)$. The output is shown in Figure 5.28 and Figure 5.29.

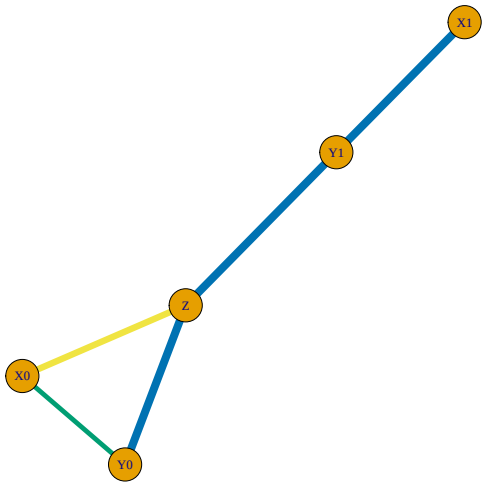


Figure 5.29: Overall structure after ten repeats using PC algorithm; f is the $exp(-x)$ function, g is the x^3 function and h is the $tanh(x)$ function. The link between Z and $X0$ are occasionally missed, often finding a link between $X0$ and $Y0$.

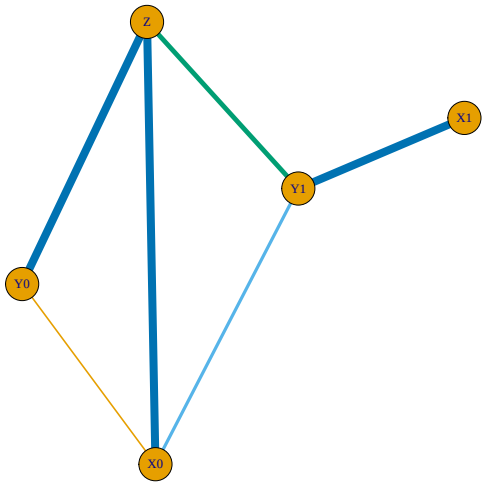


Figure 5.30: Overall structure after ten repeats using Algorithm 4; f is the $exp(-x)$ function, g is the x^3 function and h is the $tanh(x)$ function. A link between $X0$ and $Y0$ or $Y1$ are found occasionally, and the link between Z and $Y1$ is often missed.

For this configuration, the PC-algorithm performed very well yet again. The link between Z and Y_1 was always present, and the edge between X_0 and Y_0 was removed half of the time throughout the trial. While the other half kept the edge, it is also worth noting that there were a few instances where the link between Z and X_0 was not picked up. The method using the threshold seemed to perform well in a different way, and made mistakes in a different way. In this case, the clique under H_0 had the link between X and Y removed almost every time. However, the link between Z and Y_1 was missed often.

When attempting to compare the outputs, it brings back the question of where the priority lies: picking up an edge that should be present, or deleting an edge that should be removed. Nevertheless, out of these configurations, $tanh(x)$ seemed to be the most compatible with the new method. As a final check, it was used more than once yet again, but with different links: f was fixed to be $exp(-x)$, while g and h were $tanh(x)$. The results are shown in Figure 5.30 and Figure 5.31.

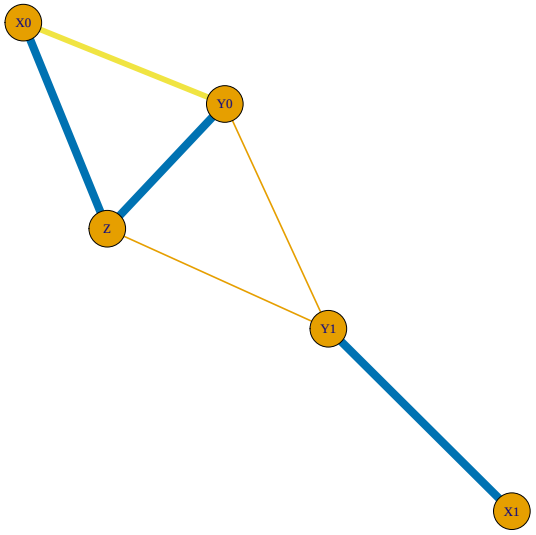


Figure 5.31: Overall structure after ten repeats using PC algorithm; f is the $exp(-x)$ function, g and h are the $tanh(x)$ functions. Repeated links between $X0$ and $Y0$ are shown, with occasional links between $Y0$ and $Y1$. The link between Z and $Y1$ is often missed.

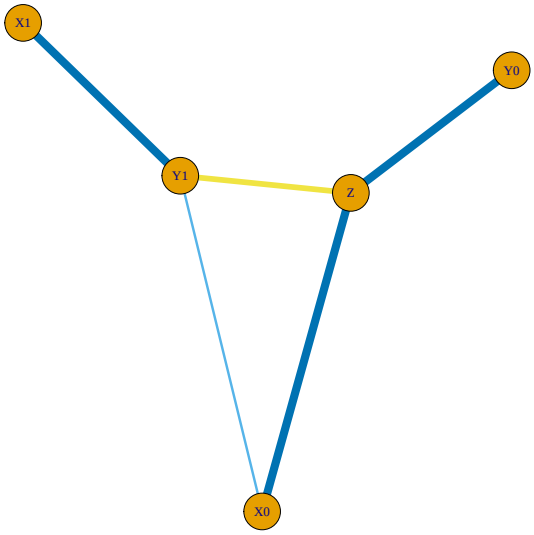


Figure 5.32: Overall structure after ten repeats using Algorithm 4; f is the $exp(-x)$ function, g and h are the $tanh(x)$ functions. The link between Z and $Y1$ is missed few times, a link between $Y1$ and $X0$ is shown a few times instead

In terms of previous points, the PC-algorithm seems to perform rather poorly in this case. The link between Z and Y_1 is rarely present, and the clique under H_0 almost always remains. While usually one of the two issues was usually present, having them both leads further from the ideal structure. On the other hand, using the 0.5 threshold for EMI appears to perform rather well in this trial. The edge between X_0 and Y_0 is always removed, and the edge between Z and Y_1 are mostly present.

As noted before, while recording the frequency of edges appearing in the graph structure does not indicate performance directly, it does help to pick up some patterns that emerge when new samples are being drawn from the same distributions. In addition, it gave the notion that the new method may perform better than the PC-algorithm in some circumstances, but at the same time there are configurations where it does worse. Potentially, these figures

could be also represented in a table using the frequency of a link appearing. However, this does not give additional insight to specific cases, or connections regarding how the presence or absence of edges are related. It also highlights the importance of clarifying priorities based on context. In some cases, it may be more important to find a specific link, even at the cost of assigning or removing some edges that were not expected.

It is also quite rare that there is a notion of the 'ideal' structure, as the lack thereof is why these methods are used in the first place. Therefore, when comparing the performance of learning algorithms, it is rarely possible to objectively deem one better, and even less likely to obtain numerical summaries. Although through simulation, certain relationships may be enforced, and therefore expected, this is usually not the case. Therefore, it may be more of interest to focus on what type of relationships we wish to detect, as some method might be more suitable than another.

Overall, the new method did show instances where it appears to perform better than the PC-algorithm, even if it did not match it in the Multivariate Gaussian case. As the ranking method does not seem to be very useful for low dimensional data, it was mostly abandoned, although it might be useful for data with far more random variables, where the key interest is finding the most relevant links (and then examine cliques). When comparing the similarities in the outputs of the PC-algorithm and the method with threshold for EMI, 0.5 appears to be reasonable value for k=3, but it might be more suitable to ensure that the threshold moves with highest observed EMI for each case, and with different values of k.

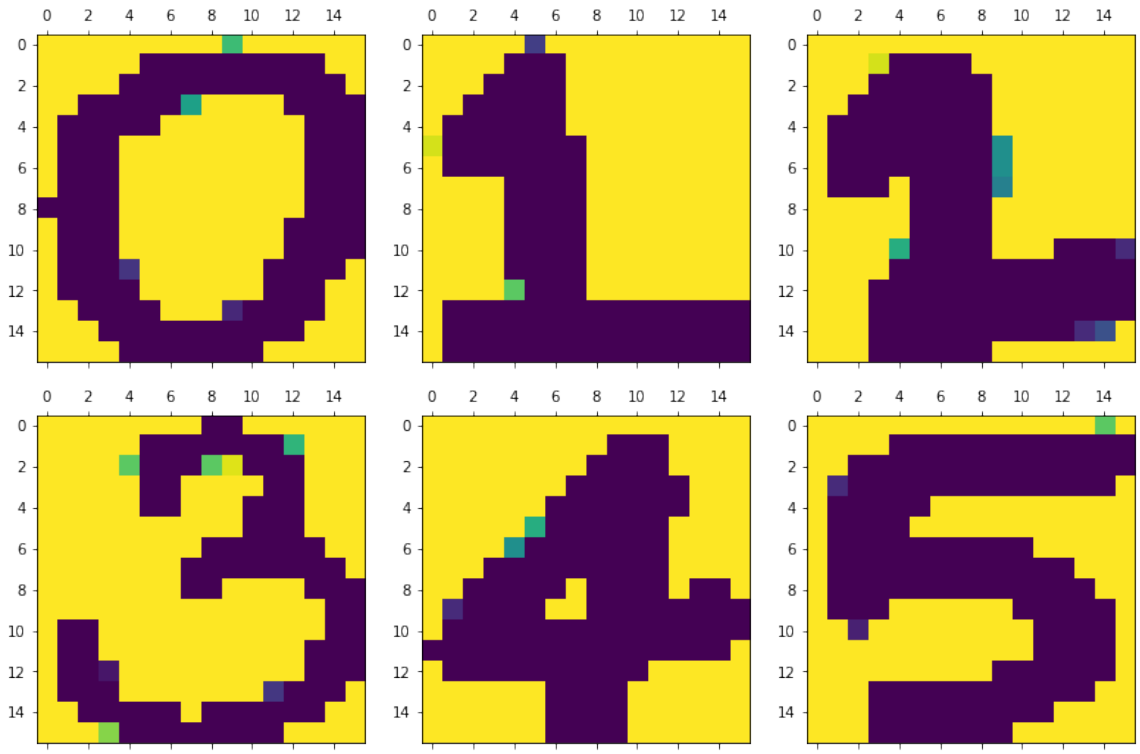
Chapter 6

6 Application - handwritten digits

6.1 Introduction

With the increasing amount of data available due to advancements in technology and collection processes, high-dimensional datasets have become fairly commonplace in a number of settings. Finding and presenting the relevant information in such a high volume of data can be difficult, which often leads to dimension reduction problems.

One example of such high-dimensional datasets is the Chars74K dataset (De Campos et al., 2009). The dataset consists of a total of 16425 Kannada characters generated by 25 volunteers and 3410 English characters, both digits and letters, generated by 55 volunteers. Therefore, it contains 55 color filtered, cropped and scaled images for digits 0 to 9. After pre-processing, each image is an observation containing 256 pixels (16 x 16), where each pixel a realisation of one dimension with additional noise. Figure 6.1 shows an example of each digit in the 16 x 16 pixel form.



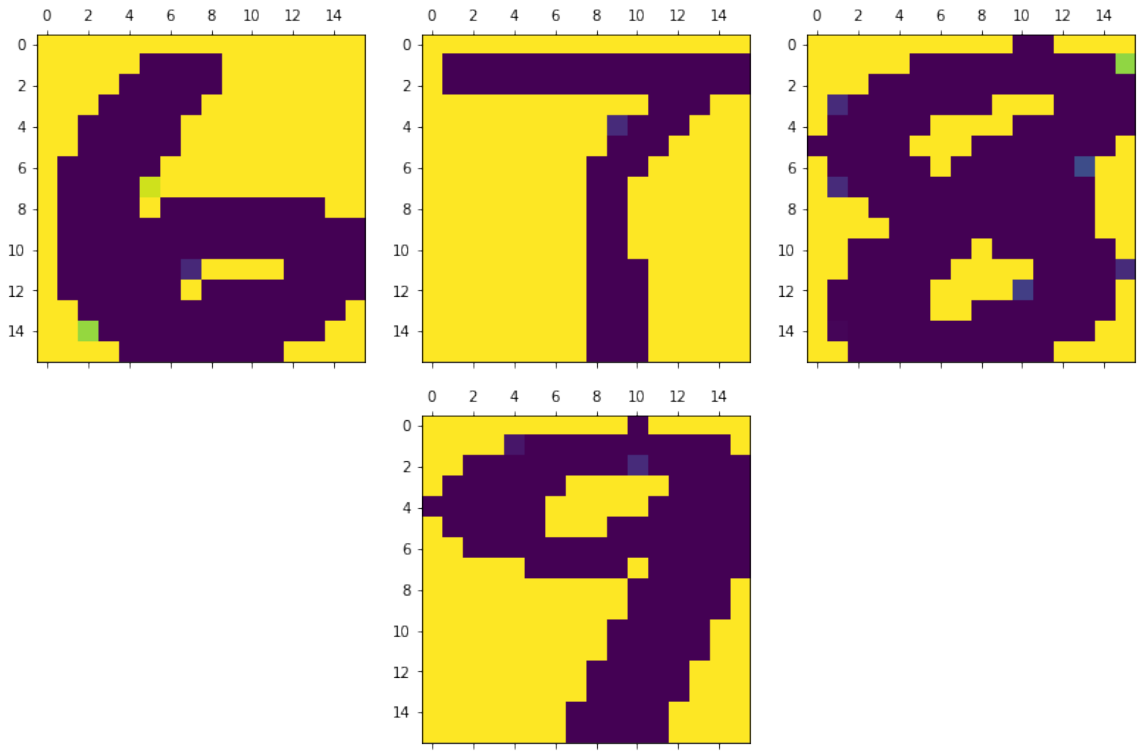


Figure 6.1: Examples of each of the digits

While some of the numbers are quite distinct, the figure shows how distinguishing them can become a more difficult task in some cases. For example, the digits 1 and 2 are fairly similar to each other, as well as the digits 5 and 6. In addition, even when the shape of these digits are not as similar, they may be mistaken as a part of another number, such as 0, 2, 3, 5 and 6 could potentially be considered part of 8 with some discrepancies in the details.

Furthermore, there is more than one way to write any number, and this is also represented in the Chars74K dataset as well. Figure 6.2 shows 3 different instances of the digit 1, and highlights how different observations could be mistaken for another digit - the first one close to the appearance of 2, and the third similar to a 7.

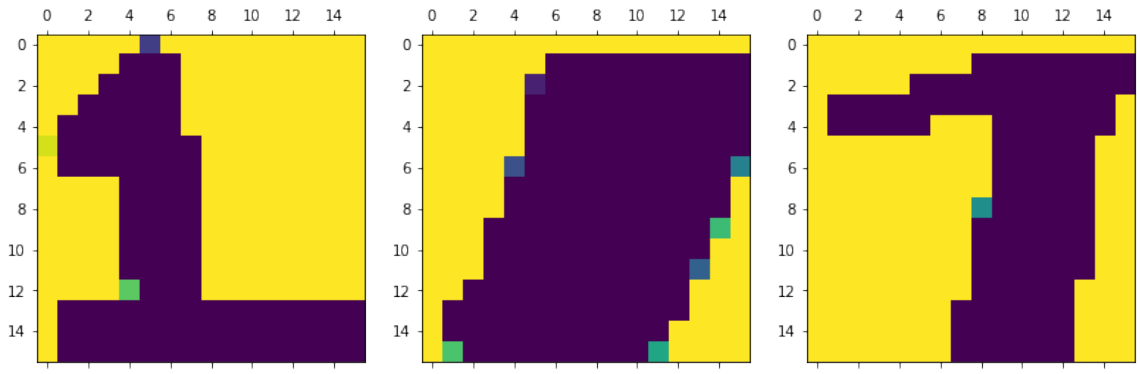


Figure 6.2: Different instances of digit 1

Typically, it is of interest if a dimension reduction tool can be used that separates the digits, making them identifiable in a lower dimensional setting. If the shape similarity of digits is used as the distance between images, the ideal dimensionality reduction tool would make digits with similar shapes close in latent space. For this setting, the digits 1, 2, 8 and 9 were chosen, resulting in 220 labelled observations. For the initial dimension reduction, Gaussian Process Latent Variable Models (GP-LVM) with 5 and 10-dimensional latent space, and Radial Basis Function (RBF) kernel were used.

It was then of interest to examine the structure of the latent space obtained from the models, potentially highlighting dimensions which contribute most towards the separation of the digits while observing the degree of separation. This is where the algorithm is used, this time using HSIC tests during the pairwise phase, noting the dependence relations as well as the degree of centrality for these dimensions.

6.2 Gaussian Process Latent Variable Model (GP-LVM)

A Gaussian Process (GP) is a collection of random variables where any point $\mathbf{x}$ is assigned a random variable $f(\mathbf{x})$, and any finite number of such variables are Multivariate Gaussian (Williams and Rasmussen, 2006). GP as a nonparametric model is a defining block of the generative Gaussian Process Latent Variable Models (GP-LVM). The two functions defining a GP are its mean function $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$, usually taking the form:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{6.1}$$

where

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})],$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x} - m(\mathbf{x}))(f(\mathbf{x}' - m(\mathbf{x}'))].$$

When defining the mean function, it is fairly common to set $m(\mathbf{x}) = \mathbf{0}$. An important step therefore is the choice of kernel function, specifying the covariance between pairs of variables through some hyperparameters. One of the most commonly used is the RBF kernel, defined as:

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \sigma_k^2 exp(-\frac{d^2}{2l^2}), \tag{6.2}$$

where d is the Euclidian distance between $\mathbf{x}$ and $\mathbf{x}'$, l is the lengthscale parameter and σ_k^2 is the variance controlling vertical variation.

Then, given observations N in D dimensions, that is $Y \in \mathbb{R}^{N \times D}$, the aim is to obtain the latent matrix $X \in \mathbb{R}^{N \times Q}$, where $Q \ll D$. X and Y are linked by vector valued function $\mathbf{f}$. Starting from noise realization, a row of Y, Y_i is obtained from

$$Y_i = f(X_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_f^2 I)$$

(noting the difference between σ_k^2 and σ^2.)

Then, placing a GP prior on $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), ..., \mathbf{f}_D(\mathbf{x}))$,

$$f_d(\mathbf{x}) \sim \mathcal{GP}(0, k_f(\mathbf{x}, \mathbf{x}')), \quad d = 1, ..., D,$$

we have independent draws from a GP for each component of $\mathbf{f}$ (Lawrence,2004). For the

link between observations (Y) and the latent function values, noted as $F^{N\times D}$ at X, we have:

$$Y_i \sim \mathcal{N}(F_i, \sigma^2 I), \quad F(X_i) \sim \mathcal{GP}(0, K_{ff}), \quad X_i \sim \mathcal{N}(\mathbf{0}, I),$$

where X_i is the ith data point and $K_{ff} = k_f(X, X)$ is defined by the kernel over pairs of rows of X. Then, denoting the dth column of Y and F as $\mathbf{y}_d$ and $\mathbf{f}_d$, kernel hyperparameters as $\boldsymbol{\theta}_f$, and assuming that the draws from the components of $\mathbf{f_d}$ are independent in the original space, the prior distribution of the latent variables can be written as

$$p(F|X, \boldsymbol{\theta}_f) = \prod_{d=1}^{D} p(\mathbf{f}_d|X, \boldsymbol{\theta}_f) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{f}_d|\mathbf{0}, K_{ff}),$$

noting that the dimensions are not independent in the latent space. Using the link between observations and the latent function we also have

$$p(Y|F) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_d|\mathbf{f}_d, \sigma^2 I).$$

Having the jointly Gaussian distribution $p(Y, F|X) = p(Y|F, X)p(F|X)$, for fixed X the distribution $p(Y|X)$ is also Gaussian:

$$p(Y|X) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_d|\mathbf{0}, K), K = K_{ff} + \sigma^2 I.$$

Since X appears within the $K_{ff} + \sigma^2 I$, it is not always possible to calculate. However, it is possible to obtain the latent matrix X is optimized through a maximum likelihood or a MAP approach. For optimal values $\hat{X}$ and $\hat{\boldsymbol{\theta}}$ this can be written as

$$\{\hat{X}, \hat{\boldsymbol{\theta}}\} = \underset{X,\boldsymbol{\theta}}{\operatorname{argmax}}\, p(Y|X, \boldsymbol{\theta})p(X), \tag{6.3}$$

which is followed by optimizing the log-likelihood $\mathcal{L} = \log\, p(Y|X)$ based on gradients. As a result, the mapping function and the latent variable positions are possible to estimate after an initial dimension reduction for X.

As part of the initial dimension reduction, it is also worth noting the option of using **automatic relevance determination (ARD)** kernels. These kernels assign scaling parameters for each dimension, noting the scaling (Neal, 2012). Low scaling manifests in fairly constant

output over changes in an input dimension, suggesting that the specific input dimension may be less relevant, as the output does not appear to be affected (Zwießele, 2017). By using ARD kernels and the scaling they provide, one can identify input dimension with low scaling. As these variables have less of an effect on the output, they can then be potentially be removed, reducing dimensionality. Figure 6.3 shows examples of different scalings for two input variable and the output. From the visual representation, the scaling can be interpreted as the frequency of movement (up and down) in the output over a range in the input. In the first case, both input dimensions affecting the output on the region, which would deem both variables relevant; in the second case, while $\mathbf{X_1}$ affects the output, $\mathbf{Y}$ appears fairly constant over $\mathbf{X_2}$; in the third case, the output is constant, suggesting that neither input variables are relevant and should consider 'switching off'.

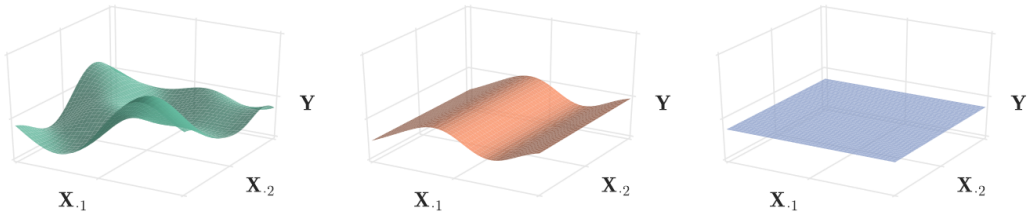


Figure 6.3: Different scalings for two input variable and the output.First case (left) shows both X_1 and Y_1 affection Y. Second case (middle) shows X_2 hardly affecting Y. Third case (right) shows neither input variables really affecting Y.

These scaling parameters can be incorporated to kernels such as the RBF kernel (Williams and Rasmussen, 2006) as a part of the lengthscale parameter l, by writing

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2_{RBF} exp(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T M(\mathbf{x} - \mathbf{x}')), \tag{6.4}$$

where

$$M = diag(\mathbf{l})^{-2}$$

In general, one may choose to reduce dimensionality further by choosing input dimensions with high scaling, especially if one observes the output to be fairly constant regardless of changes in the input. As an additional note, when ARD kernels are used, often the input dimensions are sorted based on the relevance assigned by the scaling introduced – that is, for $\mathbf{X} = (X_1, ..., X_n), X_1$ would have to most noticable effect on output $\mathbf{Y}$. Therefore, the first few input dimension are expected to be the most relevant. In this context, however, there may be variables with lower scaling that are still of interest, due to potential conditional

dependence relationships.

Overall, with a range of choices for kernel functions and their associated hyperparameters, the optimization method and the initialization of X, GP-LVMs provide a flexible approach that can be adapted to the problem at hand, with several variants available (Titsias and Lawrence, 2010). While other dimensional reduction techniques such as Principal Component Analysis could be used – leading to examination of the principal components –, for the case of the handwritten digits, the interest is not necessarily finding an ideal model or technique, but rather choosing a select few dimensions once a reduction techniques was already used. Therefore in this context, fairly straightforward GP-LVMs are used .

6.3 Application of the algorithm

6.3.1 Examining the models on handwritten digits

First, to apply the algorithm in the context of the Chars74K dataset, more specificially for the digits 0, 3, 4 and 7, where the digits are more distinct to begin with. A GP-LVM was used with RBF kernel with ARD incorporated, where the optimized latent matrix had 5 dimensions, and the variables are named according to higher scaling – $(V_1, V_2, ...V_5)$

As a following step, it was of interest to apply the algorithm on the Chars74K dataset, – as it was seen that 1 can be a difficult number to identify, as well as a lot of similarities between 8 and 9. Therefore, the digits 0, 3, 4 and 7 were picked. Similarly, a GP-LVM was used with RBF kernel with ARD incorporated, first with the optimized latent matrix having 5 dimensions, the variables named according to higher scaling – $(V_1, V_2, ...V_5)$, and then with 10.

Much like in the previous case, the initial structure after the pairwise phase was first examined, using HSIC tests in the pairwise phase, shown in Figure 6.19. Then, the conditional phase was also conducted using a value of 6 for k, shown in Figure 6.20. Then, the PC-algorithm was applied to the dataset again to examine any main differences in the output - shown in Figure 6.18.

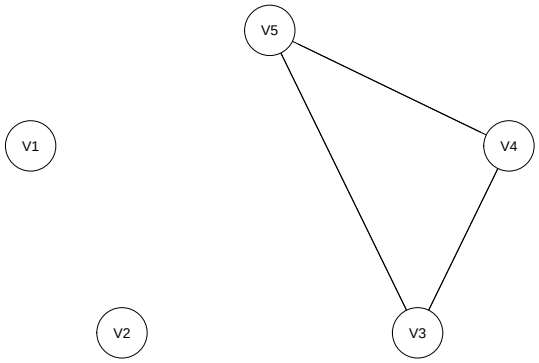


Figure 6.4: Graph obtained for digits 0, 3, 4 and 7 by using PC algorithms for 5 input dimensions

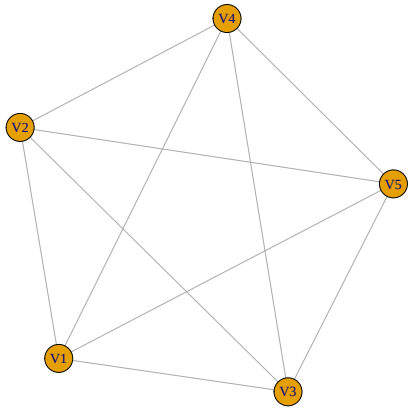


Figure 6.5: Initial graph for digits 0, 3, 4 and 7 using HSIC tests in pairwise phase

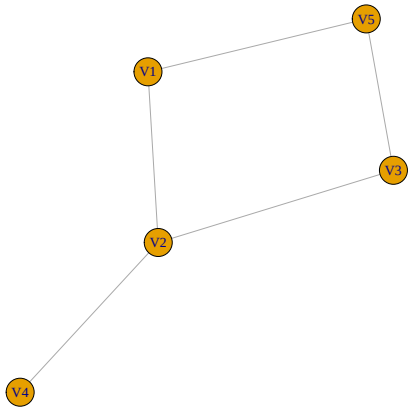


Figure 6.6: Final graph for digits 0, 3, 4 and 7 using HSIC tests in pairwise phase, k=6

Looking at the learned structure, the PC-algorithm created three subgraphs, with V_1 and V_2 being isolated nodes, as well as a clique of 3 for the remaining latent variables. On the other hand, the new method learned a complete graph in the pairwise phase, showing that the latent variables all have some connection to each other which is not necessarily linear. After the conditional phase, the graph shows a clique of order 4 between all variables except V_4, with V_2 having the highest degree of centrality, followed by the remaining nodes within the clique. Being a connected graph not containing any isolated subgraphs, the output suggests that all latent variables are related to some degree, even if not directly.

As ARD kernels are being used for this model as well, it was expected that V_1 would again be the most relevant input dimension in separating the handwritten digits. In this case, the variable with the next highest scaling, V_2 appears to agree with the output structure. These variables could potentially contribute the most to the separation of the handwritten digits, as they are related to the most dimensions, sharing a large amount of information with them and therefore explaining more of the clustering in comparison. The degree of distinguishing the handwritten digits for these two latent variables – V_1 and V_2 – using this model was also examined, shown in Figure 6.21.

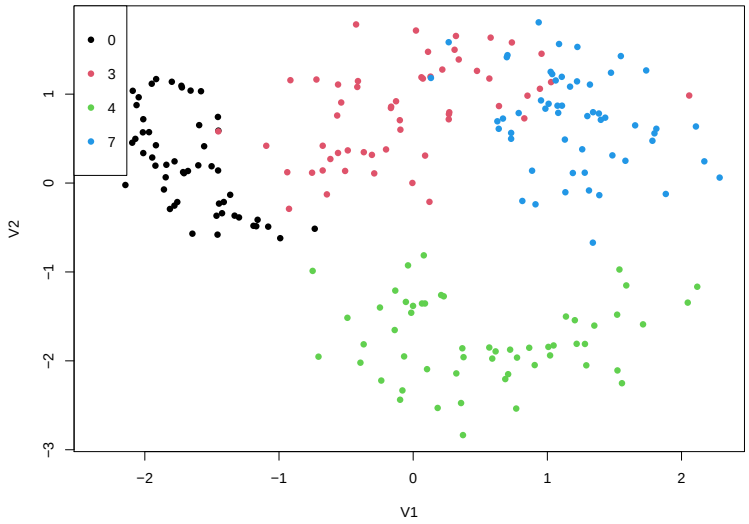


Figure 6.7: Handwritten digits 0, 3, 4 and 7 for latent dimensions V_1 and V_2

The chosen two dimensions appear to separate the numbers rather well. All the digits are relatively closely clustered, with some minimal overlap between 3 and 7, with the variance for individual digits appearing to be larger for V_1.

It was then of interest to examine separation in three dimensions as before. To choose another variable, a plausible choice would be to pick another member of the clique of 4 to see if the clusters become easier to identify. Further agreeing with the conventions of the ARD kernels, V_3 was chosen as the third vairable, as according to the scaling, the output reacts more to this input dimension. Figure 6.22 shows a snap of the handwritten digits when using latent dimensions V_1, V_2 and V_4.

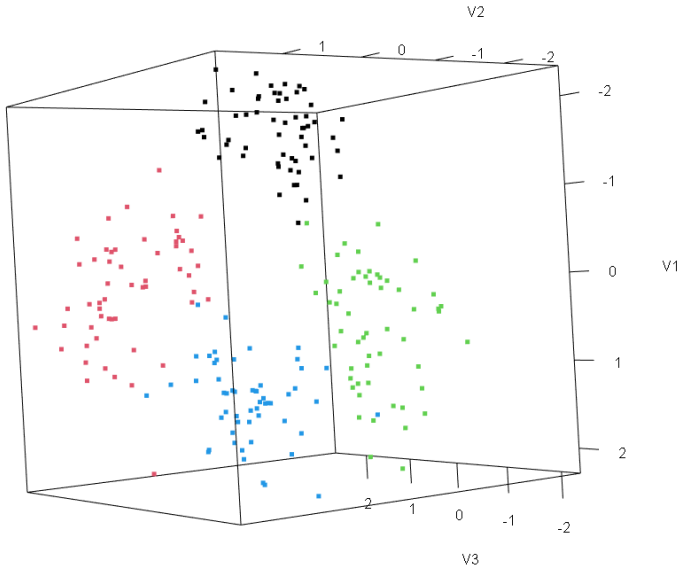


Figure 6.8: Handwritten digits 0, 3, 4 and 7 for latent dimensions V_1, V_2 and V_3

While the digits were well separated using only V_1 and V_2, adding V_3 appears to further improve the separation of the numbers, including digits 3 and 7. The previous mix with has now vanished, as while the clusters for the two digits are still close to each other, the third input dimension adds distance between them. In this three dimensional space, while not entirely simple, there does appear to be a spherical shape to each handwritten digit. In this case, the GP-LVM performed rather well, and the nodes with higher degree of centrality appear to be key contributors to the level of success, agreeing with the relevance suggested by the ARD kernel naming conventions.

In order to further assess the model performance, the silhouette widths were also calculated. For observation i, define $a(i)$ as the average Euclidean distance between point i and all points within the cluster it belongs to. For all other clusters C, calculate the average distance between i and points in C as $d(i, C)$. Then, silhouette width $s(i)$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}, \tag{6.5}$$

where $b(i)$ is the distance between the point and its nearest neighbouring cluster, $min_C(d(i, C)$. When a cluster is of size 1, $a(i)$ is not clearly defined, and the silhouette width is then typically set to 0 (Rousseeuw 1987). Ranging from -1 to 1, higher values for $s(i)$ show that in linear space, the point is close to other observation within its own cluster; lower values

represent that other clusters are close to it as well; negative values suggest that it is mainly surrounded by another cluster and should potentially be reassigned to that cluster. It is then possible to obtain average silhouette widths for each cluster, measuring how close all point within a cluster are, as well as an average of all silhouette widths, giving a sense of the performance of the model, quantifying how well the data is clustered overall.

As the next stage, the average silhouette widths were compared. The distance matrices were calculated for the case with all 5 dimension, followed by using V_1 and V_2 only, as well using V_1, V_2 and V_3. Figure 6.23 shows a silhouette plot for the clustering using all 5 variables, while Figure 6.24 shows when the distance matrix is calculated using V_1 and V_2. Finally, Figure 6.25 uses V_1, V_2 and V_3 to check the clustering:

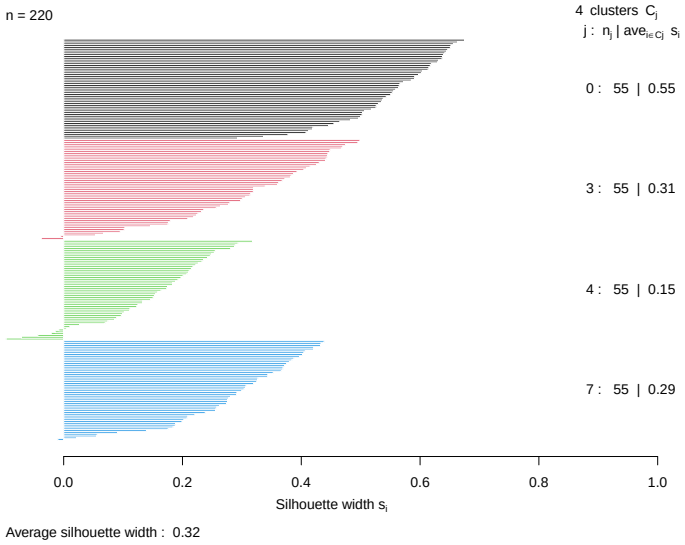


Figure 6.9: Silhouette plot for digits 0, 3, 4 and 7 using all latent dimensions to obtain distance matrix

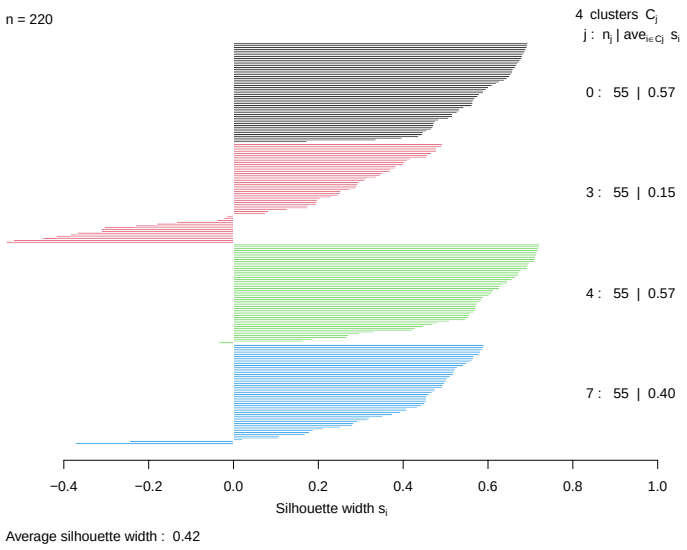


Figure 6.10: Silhouette plot for digits 0, 3, 4 and 7 using V_1 and V_2 to obtain distance matrix

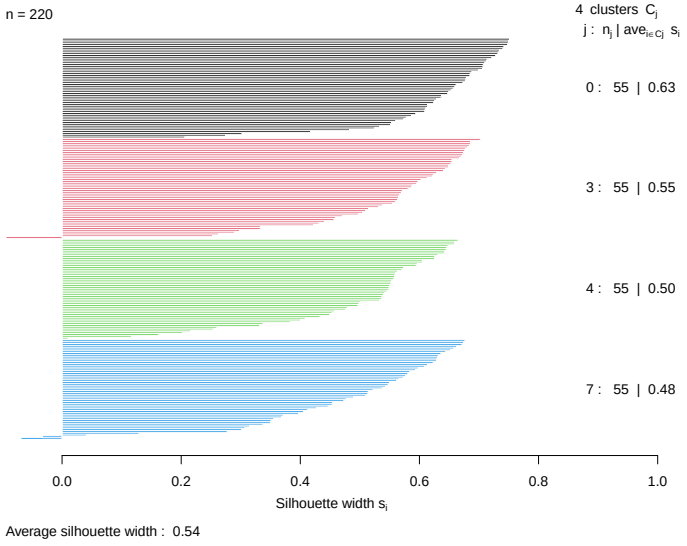


Figure 6.11: Silhouette plot for digits 0, 3, 4 and 7 using V_1, V_2 and V_3 to obtain distance matrix

The average silhouette widths for all three cases have improved considerably for digits 0, 3, 4 and 7. This was somewhat expected as these numbers are slightly more distinct to begin with, and previous figures suggested that the model performs well. Looking at each class specifically, the class with lowest values is the digit 4 when using all variables – with an average width of 0.15 –, and the digit 3 when using V_1 and V_2 only – with an average

width of 0.15. When only V_1 and V_2 are used, all silhouette widths have increased with the exception of 3, suggesting that this digit may be more difficult to separate form the rest in two dimensions. The overall average silhouette width is highest when using three variables, with a value of 0.54, while using V_1 and V_2 had an average of 0.42, and using all variables gives an average silhouette width of 0.32. While a value of 0.54 is still not overly high, it suggests a much better separation for these digits than the previous case

Overall, when looking at the numerical summaries, the GP-LVM performs rather well in separating these four digits, especially when using 3 latent input dimensions. In addition, the choice of 3 latent variables from the learned structure agreed with the relevance suggested by the ARD kernel naming conventions.

6.3.2 Increasing the number of latent dimensions

As the following step, another GP-LVM was used with 10 input dimensions for these digits as well. The initial and final graph shown in Figures 6.27 and 6.28, still using a value of 6 for k. Similarly to the model with 5 dimensions, the input dimensions seems to be fairly well connected during the pairwise phase, but the number of cliques of order 3 were reduced to 3 after the conditional phase.

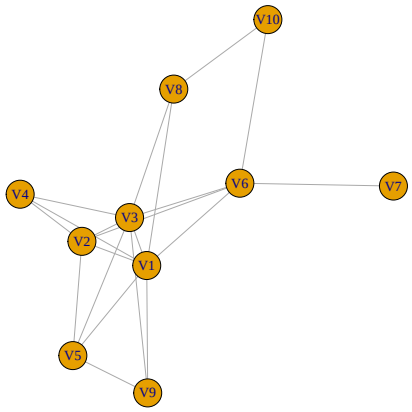


Figure 6.12: Initial graph for digits 0, 3, 4 and 7 using HSIC tests after the pairwise phase

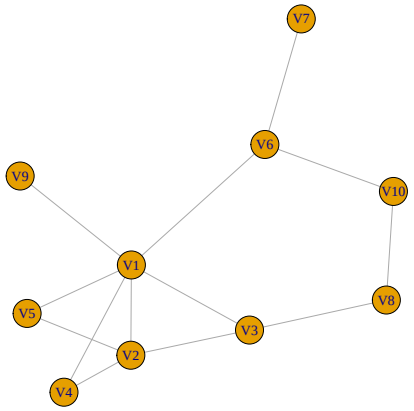


Figure 6.13: Final graph for digits 0, 3, 4 and 7 using HSIC tests after the conditional phase, k=6

After the pairwise phase, the initial graph does appear well connected. In addition, with the exception of V_7 and V_10, all nodes are part of at least one clique of order 3, most of which had an edge removed during the conditional phase. However, unlike the case with 5 input dimensions, the method was unable to establish a conditional independence relation between V_1, V_2 and V_3, V_1, V_2 and V_4 and V_1, V_2 and V_5. In addition, V_1 and V_2 both had a high degree of centrality, and being part of all the remaining cliques of order 3, the output suggests they have an important contribution in the clustering. Figure 6.29 shows the separation with V_1 and V_2 only, suggesting that the digits are well separated, with some overlap at the edges of the clusters for 3, 4 and 7. A further thing to note is a clique of order 5 remaining for variables V_1, V_3, V_8, V_11 and V_6

As V_1 and V_2 are part of all remaining cliques of order three, it was deemed best to ensure that these variables are used, with a third input variable chosen from one of the cliques – $V3, V_4$ or V_5. As ARD kernels ensure that V_3 has a higher scaling than V_4 and V_5, this was chosen as the third input variable, as shown in Figure 6.30, .

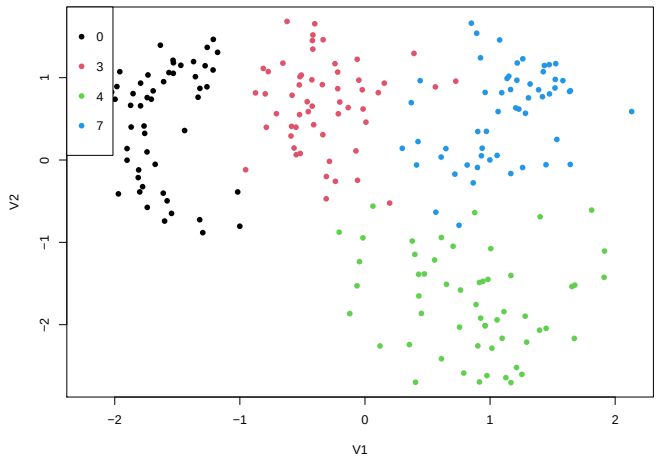


Figure 6.14: Handwritten digits 0, 3, 4 and 7 for latent dimensions V_1 and V_2

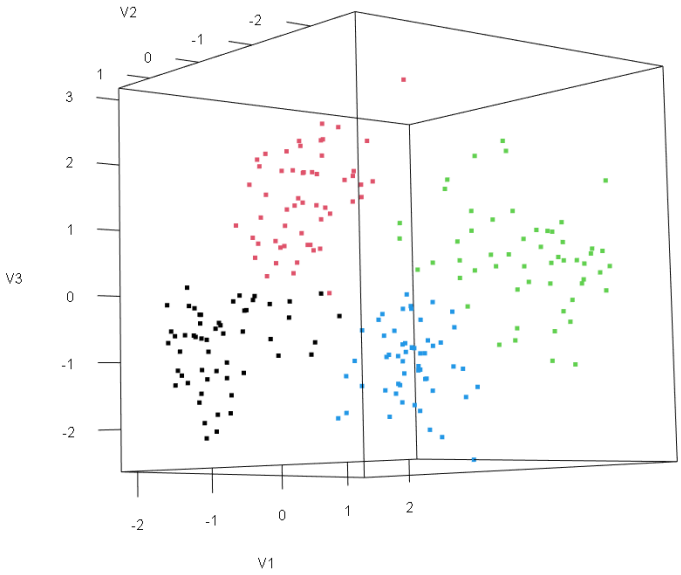


Figure 6.15: Handwritten digits 0, 3, 4 and 7 for latent dimensions V_1, V_2 and V_3

The 3D snap shows the digits to be well separated, visualizing that these three latent dimensions are key contributors for the clustering, supporting the importance of the clique, where conditional independence relations could not be established. The addition of the third variables spaced out each cluster from each other. This was followed by obtaining the silhouette widths for this model as well, shown in Figure 6.31 and Figure 6.32,

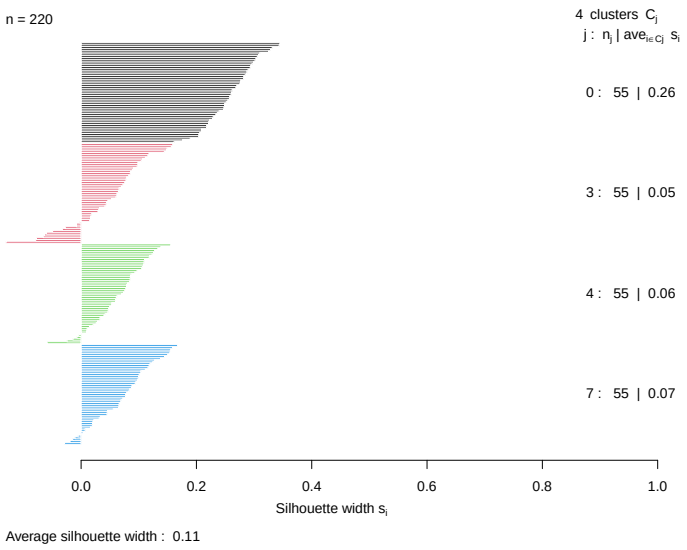


Figure 6.16: Silhouette plot for digits 0, 3, 4 and 7 using all latent dimensions to obtain distance matrix

The average silhouette widths for this model had mixed results, with a rather low value of 0.11 when the distance matrix is calculated using all 10 latent dimensions, and 0.53 when using V_1, V_2 and V_3. The average silhouette widths calculated from all input dimensions appeared very low with the exception for the cluster for 0, although there were no negative averages.

On the other hand, calculating silhouette widths by using the clique identified by applying the algorithm to obtain the distance matrix led to fairly different results. The overall average for all silhouette widths increased by a decent amount in comparison – although in general still rather low. The averages for 2, 8 and 9 have also increased, but also led to negative values for over half the observations for the digit 1, giving an average cluster width of -0.06. Therefore, it appears there is a trade-off between overall performance and performance distinguishing 1 specifically, at least in the space used to calculate silhouette widths. Then, judging by the average silhouette widths, the GP-LVM does not seem to perform well in this case either. However, the overall outcome may be better when looking at 5 latent dimensions, specifically when it comes to distinguishing 1 from the other digits.

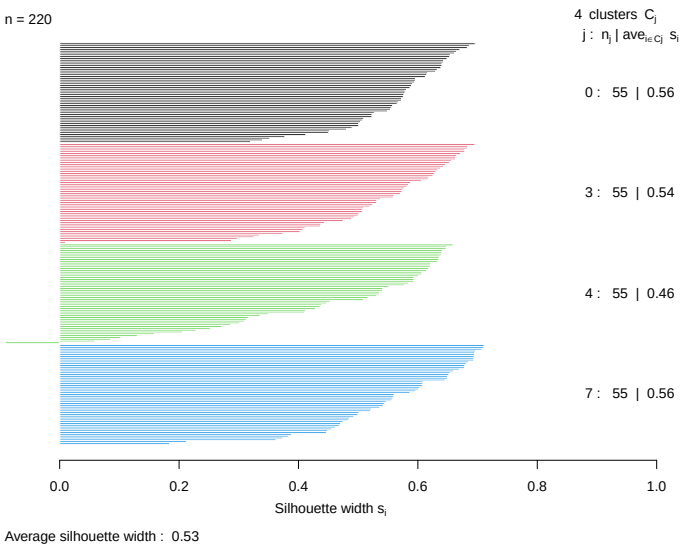


Figure 6.17: Silhouette plot for digits 0, 3, 4 and 7 using V_1, V_2 and V_3 to obtain distance matrix

While the low average silhouette width values suggest high dissimilarity for the observations, the cluster shapes suggested by the triplet observed after running the algorithm do appear to overlap less then 5 dimensional case and could potentially be used to distinguish the number 1, 2, 8 and 9 - even though the model is still far from ideal. Therefore, using the model with 10 initial latent dimensions, then having the algorithm pick the most important dimensions to visualize the clusters does seem to give improved insight on separating the handwritten digits, even when numerically, the model used is not ideal.

As a further point of interest, the learned structure did not agree with the output of the PC-algorithm. Still, utilizing it to did lead to a decent option to distinguish the digits, suggesting that using HSIC test in the pairwise phase, then checking CMI might be preferable to separate the handwritten digits. In addition, the outcome in this case supports the input variables with the highest scaling assigned by the ARD kernel, noting the clique between the first three dimensions representing the Mutual Information shared between them, and the need to keep all three input dimensions, as no conditional independence relations could be established. Furthermore, this outcome suggests that when the link between variables is not linear, and therefore less likely to be picked up by the PC-algorithm, the new method is able to learn a set of dependence relations that appeared helpful in this context, even when the GP-LVM performance may be questionable.

6.3.3 Comparison using different digits

As a following step, it was of interest to apply the algorithm on the Chars74K dataset where the numbers may not be so distinct from the start. Therefore, the digits 1, 2, 8 and 9 were picked. Similarly, a GP-LVM was used with RBF kernel with ARD incorporated, first with the optimized latent matrix having 5 dimensions, the variables named according to higher scaling – $(V_1, V_2, ...V_5)$, and then with 10.

In order to better understand the contribution of each input dimension for the handwritten digits, the initial structure was first examined, using HSIC tests in the pairwise phase, shown in Figure 6.19. Then, the conditional phase was also conducted using a value of 6 for k, shown in Figure 6.20. To follow up on previous interests, the PC-algorithm was applied to the dataset as well in order to examine any main differences in the output.

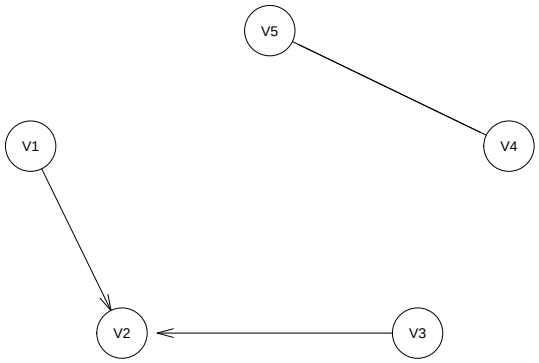


Figure 6.18: Graph obtained by using PC algorithms for 5 input dimensions

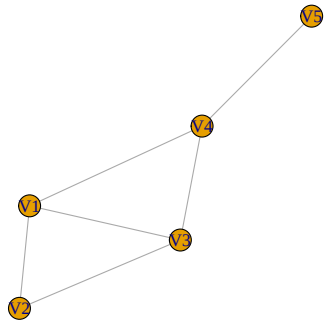


Figure 6.19: Initial graph using Algorithm 5, k=6.

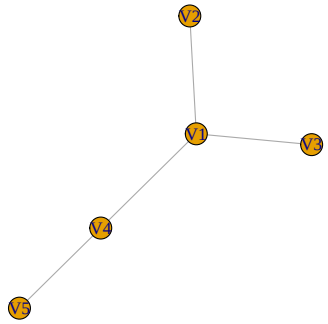


Figure 6.20: Final graph using Algorithm 5, k=6

Looking at the learned structure, the PC-algorithm seems to isolate the first three input variables from the remaining two, creating two subgraphs. On the other hand, when using the new method it appears that when only accounting for pairwise independence, the graph is well connected, suggesting that the latent variables have a fairly equal role separating 1, 2, 8 and 9 - with the potential exception of V_5. However, after the conditional phase, V_3 had its edges removed that connected it to V_2 and V_4. As a result, the random variable V_1 had the highest degree of centrality, followed by V_4, but still not creating any subgraphs.

Due to the naming convention of ARD kernels, it was expected that V_1 would be considered the most relevant input dimension in separating the handwritten digits. Still, it is worth noting that instead of the variable with the next highest scaling, V_4 appears to be a more plausible choice according to the output. These variables could potentially contribute the most to the separation of the handwritten digits, as they are related to the most dimensions, sharing a large amount of information with them and therefore explaining more of the clustering in comparison. Therefore, it is important to examine the performance of the model for these two latent variables. Figure 6.21 shows how well input dimensions V_1 and V_4 separate the digits.

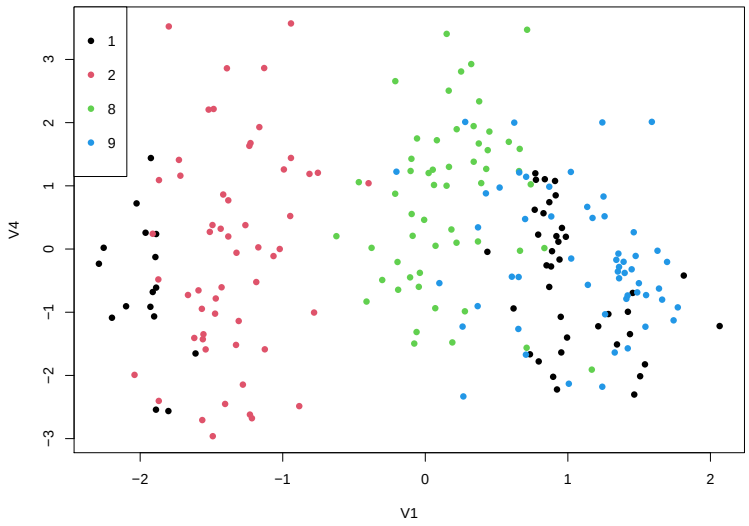


Figure 6.21: Handwritten digits 1, 2, 8 and 9 for latent dimensions V_1 and V_4

The chosen two dimensions appear to perform with mixed success. V_1 and V_4 separate the digit 2 (red), 8 (green) and 9 (blue) well, with some minimal overlap between 8 and 9 . However, performance drops when it come to 1 (black), which seems to have two separate

clusters, one of them heavily mixing with the digit 9. In addition, the variance for individual digits appears to be much larger for V_4.

For further investigation, it is also of interest to examine separation in three dimensions, as this enables identification of patterns that are not clear to see in a two-dimensional space. Therefore, another variable was added to see if the clusters become easier to identify. As the graph does not suggest a clear choice for the third variable, V_2 was chosen, as according to the scaling done by the ARD kernel, the output reacts more to this input dimension. Figure 6.22 shows a snap of the handwritten digits when using latent dimensions V_1, V_2 and V_4.

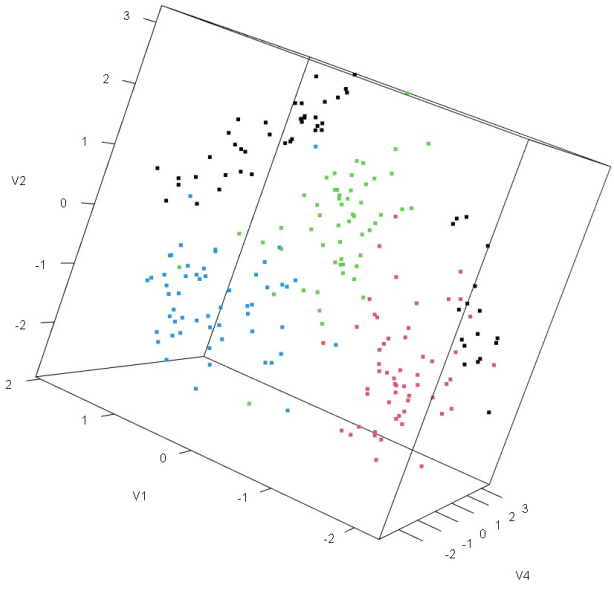


Figure 6.22: Handwritten digits 1, 2, 8 and 9 for latent dimensions V_1, V_2 and V_4

While the digit 1 still appears to be in two separate clusters, with the additional latent variable these two clusters are much better separated from the other digits. The previous mix with 9 has vanished, as the two digits are on different ends on the third input dimension. In this three dimensional space, while not entirely simple, there does appear to be a parabolic shape that could encapsulate 1. In addition, the overlaps between the clusters for 2, 8 and 9 seems to have been reduced, each of the three numbers forming their separate cluster – resembling a sphere. Although the performance of the GP-LVM may not be ideal, the nodes with higher degree of centrality appear to be key contributors to the level of success.

For the GP-LVM with 5 latent dimension, the average silhouette widths were then compared.

The distance matrices were calculated for the case with all 5 dimension, followed by using V_1 and V_4 only, as well using V_1, V_2 and V_4. Figure 6.23 shows a silhouette plot for the clustering using all 5 variables, while Figure 6.24 shows when the distance matrix is calculated using V_1 and V_4. Finally, Figure 6.25 uses V_1, V_4 and V_2 to check the clustering:

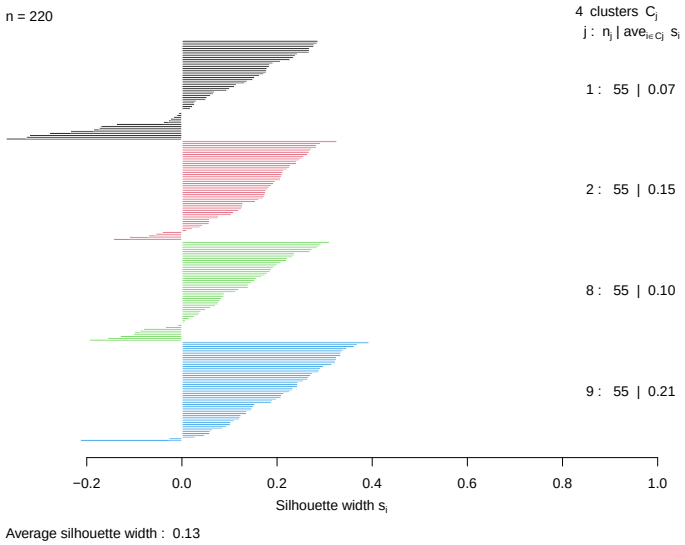


Figure 6.23: Silhouette plot using all latent dimensions to obtain distance matrix

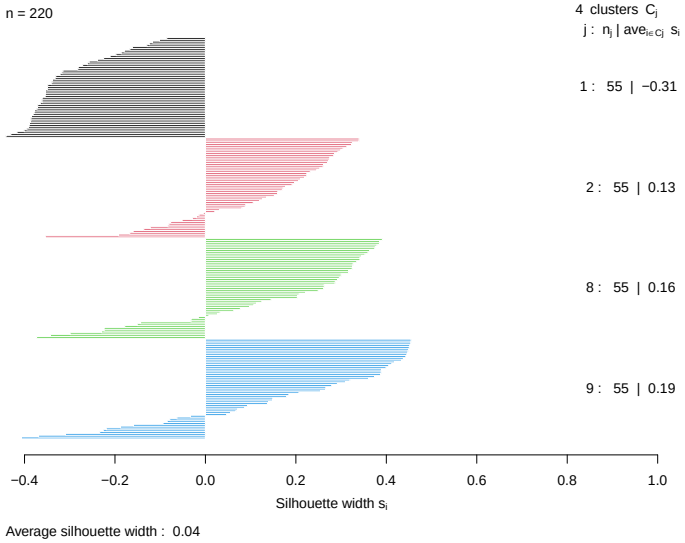


Figure 6.24: Silhouette plot using V_1 and V_4 to obtain distance matrix

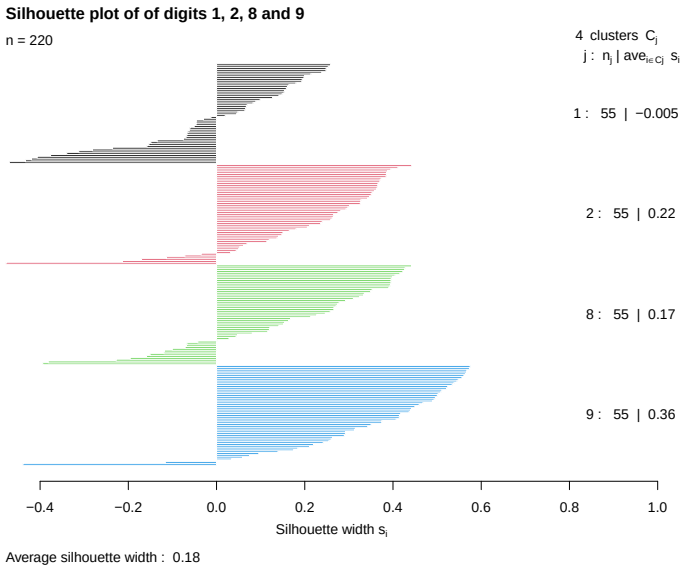


Figure 6.25: Silhouette plot using V_1, V_2 and V_4 to obtain distance matrix

The average silhouette widths for all three cases are rather low. Looking at each class specifically, the class with lowest values is the digit 1, as expected from observing the two- and three-dimensional figures indicating two separate groups for point in this cluster. When only V_1 and V_4 are used, all silhouette widths within the cluster had negative values (-0.31), suggesting that they should be reclassified – judging from previous output, merging the observations with the digits 2 and 9. The average silhouette width does improve as more latent dimensions are included, with - 0.005 when V_2 is included, and 0.07 with all 5 variables. This further supports insight that digit 1 is the most problematic number to distinguish.

While the model performs the worst when it comes to 1, the clusters for other digits also contribute to the low overall averages. Although negative silhouette widths are much more rare for the other numbers, the values are still very low and fairly close to zero, suggesting that the points within the clusters are not too similar. This is likely due to the shape of the suggested groups, where several point are close to the edges of other clusters, especially as the number of input dimensions are reduced. Using V_1 and V_4 leads to the poorest average silhouette width of 0.04, while using all latent dimension gives 0.13, and adding V_2 to V_1 and V_4 gives 0.18.

Overall, when looking at the numerical summaries, the GP-LVM does not appear to perform very well in separating these four digits. However, when observing visual representation, us-

ing V_1, the node with highest degree of centrality after running the algorithm, along with V_4 does give a decent idea of the shape of the clustering, especially after adding in a further latent dimension with high scaling linked to V_1, even with low values for silhouette widths. This suggests that while the model may not be ideal, the selection suggested by running the algorithm does provide insight to distinguish these numbers, especially when extended to three dimensions.

As the following step, another GP-LVM was used with 10 input dimensions to see how increasing the number of latent dimensions affects the separation of the digits. The initial and final graph shown in Figures 6.27 and 6.28, still using a value of 6 for k. Similarly to the previous case, the input dimensions seems to be fairly well connected during the pairwise phase, but the number of cliques of order 3 were reduced to one after the conditional phase. In addition, the PC-algorithm was again applied as well.

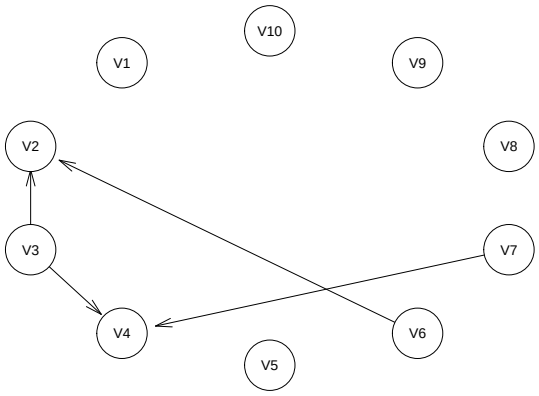


Figure 6.26: Graph obtained by using PC algorithms for 10 input dimensions

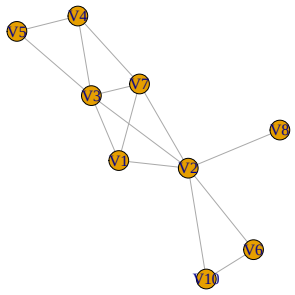


Figure 6.27: Initial graph using HSIC tests in pairwise phase

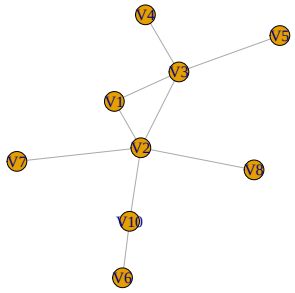


Figure 6.28: Final graph using HSIC tests in pairwise phase, k=6

As a contrast to the output using the new method, the output from the PC-algorithm show only five of the ten nodes connected, and V_1 with the highest scaling is not among them.

This supports the idea that a lot of the links between these input dimensional are present, but are not linear, and therefore may be missed.

After the pairwise phase, the initial graph does appear well connected. In addition, with the exception of V_8, all nodes are part of at least one clique of order 3, most of which had an edge removed during the conditional phase. However, unlike the case with 5 input dimensions, the method was unable to establish a conditional independence relation between V_1, V_2 and V_3. In addition, V_2 and V_3 both had high degree of centrality, suggesting an important contribution in the clustering. This suggests that even though these variables have the highest scaling – due to ARD naming conventions –, they share a lot of Mutual Information among each other, and no conditional dependence relation could be established, therefore should likely keep them together.

Furthermore, as seen in Figure 6.29, V_2 and V_3 on their own don't seem to separate the digits very well. All digits appear to overlap with one another in two dimension, although it is worth noting that unlike before, the observations of handwritten digit 1 are closer together and not necessarily in two separate clusters. It therefore appears best to ensure that the triplet from the clique of order 3 are retained together, even if V_1 is only connected to V_2 and V_3, as no conditional independence relation was established. Figure 6.30 illustrates the separation of the digits when looking at these three variables.

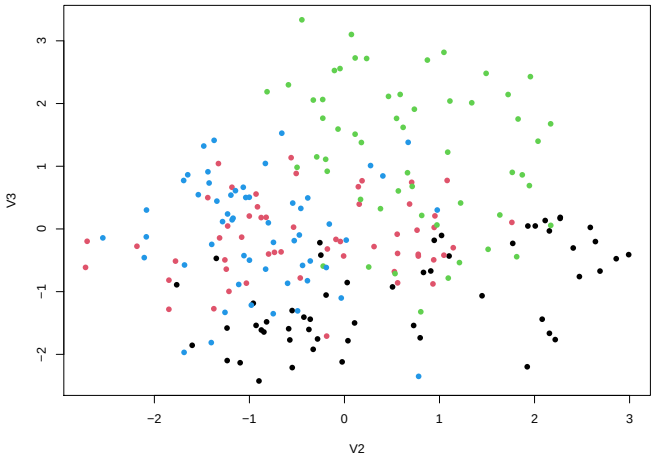


Figure 6.29: Handwritten digits 1, 2, 8 and 9 for latent dimensions V_2 and V_3

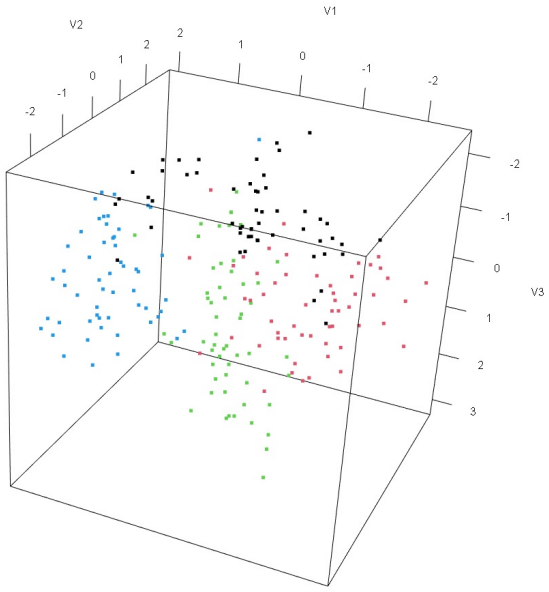


Figure 6.30: Handwritten digits 1, 2, 8 and 9 for latent dimensions V_1, V_2 and V_3

The 3D snap shows the digits to be well separated, visualizing that these three latent dimensions are key contributors for the clustering, supporting the importance of the clique, where conditional independence relations could not be established. The addition of the third variables spaced out each cluster from each other, even digit 1 to a decent degree. As the next step, the silhouette widths were examined for this model as well, shown in Figure 6.31 and Figure 6.32,

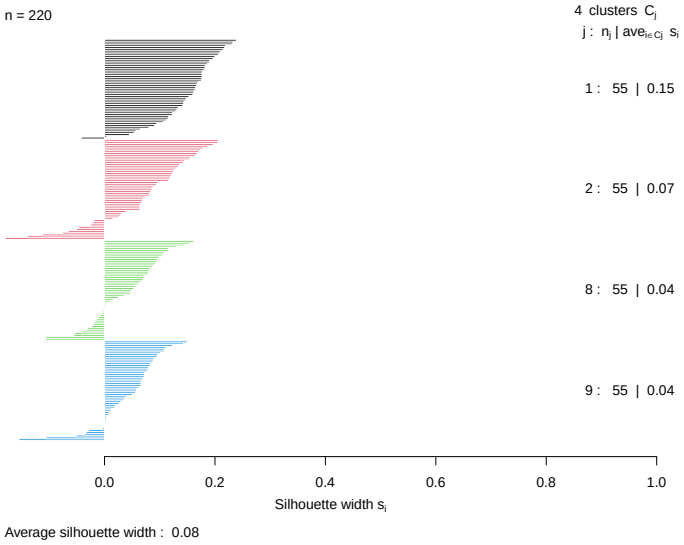


Figure 6.31: Silhouette plot using all latent dimensions to obtain distance matrix.

The average silhouette widths for this model end up low as well, with a value of 0.08 when the distance matrix is calculated using all 10 latent dimensions, and 0.25 when using V_1, V_2 and V_3. When looking at silhouette widths calculated from all input dimensions, there seems to be an improvement to the cluster 1, with far fewer negative values, and a cluster average of 0.15. While this hold for the other clusters as well, it also led to low average widths for them, suggesting that while the observations are fairly dissimilar within each of their cluster, they are in the right group.

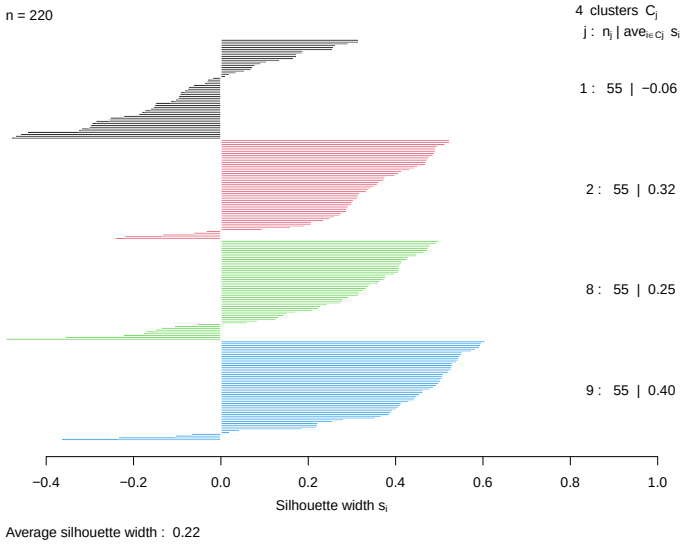


Figure 6.32: Silhouette plot using V_1, V_2 and V_3 to obtain distance matrix

On the other hand, calculating silhouette widths by using the clique identified by applying the algorithm to obtain the distance matrix led to fairly different results. The overall average for all silhouette widths increased by a decent amount in comparison – although in general still rather low. The averages for 2, 8 and 9 have also increased, but also led to negative values for over half the observations for the digit 1, giving an average cluster width of -0.06. Therefore, it appears there is a trade-off between overall performance and performance distinguishing 1 specifically, at least in the space used to calculate silhouette widths. Then, judging by the average silhouette widths, the GP-LVM does not seem to perform well in this case either. However, the overall outcome may be better when looking at 5 latent dimensions, specifically when it comes to distinguishing 1 from the other digits.

While the low average silhouette width values suggest high dissimilarity for the observations,

the cluster shapes suggested by the triplet observed after running the algorithm do appear to overlap less then 5 dimensional case and could potentially be used to distinguish the number 1, 2, 8 and 9 - even though the model is still far from ideal. Therefore, using the model with 10 initial latent dimensions, then having the algorithm pick the most important dimensions to visualize the clusters does seem to give improved insight on separating the handwritten digits, even when numerically, the model used is not ideal.

As a further point of interest, the learned structure did not agree with the output of the PC-algorithm. Still, utilizing it to did lead to a decent option to distinguish the digits, suggesting that using HSIC test in the pairwise phase, then checking CMI might be preferable to separate the handwritten digits. In addition, the outcome in this case supports the input variables with the highest scaling assigned by the ARD kernel, noting the clique between the first three dimensions representing the Mutual Information shared between them, and the need to keep all three input dimensions, as no conditional independence relations could be established. Furthermore, this outcome suggests that when the link between variables is not linear, and therefore less likely to be picked up by the PC-algorithm, the new method is able to learn a set of dependence relations that appeared helpful in this context, even when the GP-LVM performance may be questionable.

In addition to the visual representation between the input variables that can present additional insight, the learned structure also appears to lead to a favourable outcome despite model performance. Focusing on the structure of a set of random variables rather than finding the 'ideal' model could potentially be useful in other contexts as well, not simply in the case of non-linear links. The idea of finding relevant links between a set of random variables could also be useful in different dimension reduction problems as well.

Chapter 7

7 Application - vowel sounds

7.1 Introduction

The field of phonetics can be divided into three main areas: articulatory, acoustic and auditory (Reetz and Jongman, 2020). Articulatory phonetics is focused on how speech is produced, therefore concerns physiology. Auditory phonetics deals with the perception of speech and hence investigates the auditory system as well as memory. Finally, acoustic phonetics investigates characteristics of speech such as its frequency, intensity and duration, leading to examining sound waves. This means that phonetics is closely related to other fields such as linguistics, physics and psychology.

In broad terms, when speech sounds are formed they travel as sound waves, the fluctuations of air pressure often represented in an oscillogram. The graphic display of sounds can then be used to investigate what the produced sounds are in more detail. Common traits to look for in sound patterns are their amplitude, associated with loudness and whether the sound wave is periodic, repeating itself at certain intervals. Signals are also often represented in a spectogram, visualizing the the energy of sounds waves in different frequency bands across time. In this context, the main area of interest is regarding acoustic phonetics, more specifically the main characteristics of speech sounds.

Speech sounds are often characterized by the first three formant frequencies as well as their durations. Formants can be thought of as frequency peaks in the spectrum with a high energy level, especially prominent in vowels (Abhang et al, 2016). Each formant corresponds to a resonance in the vocal tract, with a formant roughly every 1000 Hz of the spectrum. It is important to note that as formant frequencies are a unique characteristic of the vocal tract, measuring them is difficult. Speech is typically picked up by a microphone of a computer and often mixes with background noise. The formant frequencies are therefore estimated.

The first formant frequency (F1) has an inverse relationship to the height of the tongue in the vowel quadrilateral. The second frequency (F2) relates to front-and backness of a vowel, high values associated with front vowels. The third formant frequency (F3) is related to the roundness of a vowel in the case a language has rounded and unrounded vowels of the same

front- or backness. In addition to vowels, approximants should also be noted in this context. While their formant positioning is close to vowels in the same articulatory place, they are closer together, indicated by lower overall energy. The study of consonants is also part of this area of study, but for current purposes, the focus remains on vowels.

Vowels are concentrations of high amplitude energy – and therefore relatively loud signals – around certain sound frequencies that are created by transmission of noise source (e.g. voicing) through relatively open vocal tract. Their main characteristic is the location of formant frequencies, determined by the shape of the vocal tract. Vowels have a specific formant frequency pattern when it comes to a speaker, or alternatively, a group that has the same vocal tract length. The key to distinguishing two vowels of a speaker is the exact location of these frequencies, associated with the different shapes the vocal tract takes when creating the sound. An additional factor that helps identifying a vowel is its duration, especially for languages that differentiate between vowels of varying length (Lehiste, 1965).

In phonetics, Keywords are often used to represent vowels sounds. John Wells (1982) classified words of the English language into 24 lexical sets on the basis of the pronunciation of the vowel of their stressed syllable in Received Pronunciation (RP; also referred to as Southern Standard British English). Each lexical set is named after a representative keyword. The five keywords used here are:

kit, dress, trap, lot, strut

For the purposes of this case, it is only necessary to recognize the keyword.

Formants are numbered from the bottom of each vowel, so that the lowest formant is always formant 1, the next is formant 2, followed by formant 3, and so on. In analysis, typically formants 1 and 2 are the most frequently used, but when possible, it includes analysis of formant 3 and 4 - which is the case in this setting. Formants will always have some level of correlation within a single vowel, as the first formant is always lower than the second, and the second is always lower than the third.

As vowel phonemes are part of a system, theory predicts numerous correlations between vowels (Foulkes et al, 2015). The vowel quadrilateral below gives a rough outline of the relative positioning Frequency 1 and 2 of the five vowels **kit, dress, trap, lot** and **strut** for Received Pronunciation (RP) (Standard Southern British English):

$$KITF1 < DRESSF1 < TRAPF1,$$

$$KITF1 < STRUTF1 < LOTF1,$$

$$KITF2 > DRESSF2 > TRAPF2 > STRUTF2/LOTF2,$$

This relations need to hold in order for words such as bit, bet, bat, bot, but to remain possible to distinguish. More accurately, vowels can be considered to have multiple sub-systems. One such sub-system is the short front vowel system consisting of **kit, dress** and **trap**. This means that in processes of sound change the same forces act on each vowel (i.e. if **trap** moves, then **dress** and **kit** must also move). An example of this can be seen in the short front vowel system of New Zealand English. The process of change caused a decrease in F1 for **trap** causing it to overlap with **dress**, as such **dress** F1 decreases overlapping with **kit**, and finally **kit** F2 decreases as it has nowhere to go on the F1 dimension. As such these vowels in New Zealand English sound like 'trep' (trap), 'driss' (dress) and 'kut' (kit).

Formants are also affected by the adjacent sounds in a syllable, also known as co-articulation. Following up /l/ sounds cause F2 to decrease in many dialects. Therefor, F2 for the vowel in 'kill' is expected to be lower than the F2 in 'kit'. Preceding /j/ sounds result in higher F2 values for the vowel that follows. Therefore, the vowel in 'yes' should have a higher F2 than 'guess'.

Overall, acoustic phonetics focus on sound waves produced by speech to examine different characteristics for the speakers. The information gained can then be applied to a number of disciplines, such as sociology, or as in the following case, forensics.

7.2 Phonetics in forensic science

7.2.1 Examining evidence in forensics

In forensic science, it is often of interest to determine whether two sets of evidence have come from a common source or not (Lindley, 1977). Typically, it relates to finding some material at a crime scene, and a similar material is also found on a suspect, which could indicate they were at the scene of the crime. While not a physical material, some sound recording of a crime may become available throughout an investigation. It can then become an important goal to identify the speaker, and therefore using knowledge gained from the acoustic phonetics perspective looking at the main characteristics of the speech can be useful.

As one of the main focus of forensic science is identification, the collected evidence requires appropriate ways to interpret it. A way to improve such interpretation is through advances in the analytical equipment and facilities that collect data from evidence found during an investigation. Another way is to improve statistical methodology that evaluates the evidence (Aitken et al, 2006). While hypothesis testing may still be used in these cases, they often do not reflect the information gained by the evidence that has been collected (Robertson and Vignaux, 1995). It is therefore preferable to use methods that take collected evidence into account when trying to identify a suspect, which can be addressed by using the likelihood ratio (Aitken and Taroni, 2004).

The likelihood ratio is used to compare the probability of relevant measurements on the evidence when a common source is assumed for evidence from the crime scene and evidence associated with the suspect to the probability assuming different sources for evidence at the crime scene and the suspect evidence. For multivariate normal data, this can then be described as a multilevel model, where the covariance structure may represent data structure in a graphical model. This may potentially reduce the number of dimensions needed to parameterise the model while dependencies are still recognised. The graphical model would estimate within-group and between-group covariances, and then use the scaled version of the inverse of them to determine dependencies. Then, the product required to calculate the joint density can be simplified by considering the learned structure, as opposed to assuming all random variables dependent, which can then be compared between evidence from a crime scene and evidence from a suspect. A likelihood ratio value above 1 supports the proposition that the evidence from the crime scene and the suspect likely likely share the same source, while a value below 1 supports the proposition that the two pieces of evidence come from

a different source. The further the obtained values are from the threshold, the stronger the evidence is considered.

In this section, it is of interest to apply the algorithm to show the dependencies in a multivariate setting, and examine if it enables a similar reduction in dimensionality through learning a structure that enables a simpler calculation of the joint density. The data used was extracted from Task 1 of the The Dynamic Variability in Speech Corpus (DyViS) database (Nolan et al., 2009). The database contains recordings of 100 male speakers of Standard Southern British English, aged 18-25, undertaking four tasks involving different speaking styles: a simulated police interview, a telephone call with an 'accomplice', a reading passage, and a set of read sentences. Task 1 is a mock police interview recording, in which the participant was forced to lie about a crime.

The five vowel keywords are formatted in the same way and saved as individual .csv files for a project by Foulkes et. al (2013-2015). Each row is a single instance of each vowel. Column 1 is speaker number, using the same 25 speakers across all vowel sounds. Column 2 is the F1 measurement, column 3 the F2 measurement, column 4 the F3 measurement and column 5 the F4 measurement. F3 and F4 have been found to be much more speaker-specific than F1 or F2, since they perform less of a role in maintaining auditory distinction between vowels. F3 is expected to be correlated with F2 to some extent; in vowels where F2 is high (e.g. KIT) F3 is also often high. F3 is also sensitive to articulatory factors such as lip rounding. The articulatory bases of F4 are less well known, although in most forensic cases F4 is unavailable because of the limitations of telephone bandpass filters. Column 6 contains the word from which the vowel measurements were extracted.

In this context, one of the aims is to examine the overall structure for all speakers when all four frequencies for each vowel sound are included. Given that there are four frequencies for five different vowel sounds, leading to a rather high dimensional dataset, it may be useful to identify a frequency that is more influential, or sufficient to distinguish between speakers while reducing dimensionality of the data that needs to be observed. In addition, it is of interest to see if the new method is able to learn a structure that may enable a simpler parameterisation for different likelihood ratios that wish to compare evidence from a crime scene and evidence from a suspect. The goal is then to retain as much of the information as possible, but through representing dependencies, address the difficulty of using high-dimensional data.

7.2.2 Graphical models and evidence evaluation

Throughout the process of comparing two sources of evidence using different likelihood ratios, one of the aspects of the problem is the factorization of likelihoods. When the random variables of interest can all be considered independent, it becomes a simple product of each marginal density. However, this is rarely the case, especially as while some variables may be pairwise independent, they may be dependent conditioned on some others. On the other hand, when no variables are assumed independent, the joint density is more complex to calculate. For a set of n random variables $\mathbf{X} = (X_1, ..., X_n)$, one can still use Bayes' theorem and the chain rule to write

$$\begin{aligned} P(\mathbf{X}) = P(X_1, ..., X_n) &= P(X_1|X_2, ..., X_n)P(X_2, ..., X_n) \\ &= P(X_1|X_2, ..., X_n)P(X_2|X_3, ..., X_n)P(X_3, ..., X_n) \\ &= P(X_1|X_2, ..., X_n)P(X_2|X_3, ..., X_n)...P(X_{n-1}|X_n)P(X_n), \end{aligned}$$

this product can become rather high dimensional as the number of random variables grow. Therefore, it is useful to understand all dependence relations within the dataset in order to simplify this product by representing the independencies that are present in the structure. Structural learning methods using graphical models are then a plausible way to identify and visualize instances where this is possible. For Bayesian Networks, as mentioned in Equation 2.1.4, once the structure is learned the joint density simplifies to

$$\mathcal{P} = P(\mathbf{X}) = \prod_{v \in V} P(X_v|X_{pa(v)}),$$

where $X_{pa(v)}$ are the parents of node v in the graph, as the Markov property and the d-separation within the graph allows considering each random variable independent of the remaining covariates once conditioned on their parent nodes. However, the different possible representations within an equivalence class can then imply different potential causal relationships, which may be unfavourable.

Another possible way to utilize graphical model is by making use of the decomposability of triangulated graphs. A simple graph G is called triangulated if every cycle of length at least four in G has a chord, that is, an edge joining two nonconsecutive vertices of the cycle. Then, these graphs can be thought of as a collection of cliques of order 3 or less, and the associated joint density can be decomposed by the product of the densities of each clique,

divided by the product of densities of separators, which separate the cliques through the global Markov property (Lauritzen, 1996)

$$f(i) = \frac{\prod_{j=1}^{k} f(i_{C_j})}{\prod_{j=2}^{k} f(i_{S_j})}, \tag{7.1}$$

where C_j are the cliques and S_j are the separators. As an example, the density represented by Figure 7.1 is triangulated and can be decomposed.

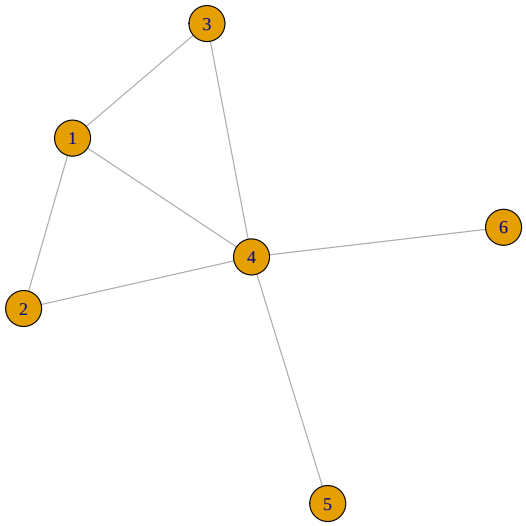


Figure 7.1: Example of a triangulated, decomposable graph

The cliques for this graph are

$$C_1 = \{1, 2, 4\}, C_2 = \{1, 3, 4\}, C_3 = \{4, 5\}, C_4 = \{4, 6\}$$

and the separators are

$$S_2 = \{1, 4\}, S_3 = \{4\}, S_4 = \{4\}.$$

Then, the joint density of the 5 random variables can be decomposed as

$$f(1,2,3,4,5) = \frac{f(1,2,4)f(1,3,4)f(4,5)f(4,6)}{f(1,4)f(4)f(4)}$$

As the example demonstrates, using decomposition of a triangulated graph, a joint density in higher dimension can then be factorized as a product of several, but lower dimensional densities. In this context, this can becomes a useful tool to compare evidence gathered in a forensic setting. Depending on the case and evidence gathered, the number of random variables that are examined to calculate a likelihood ratio can become rather high, making their joint density difficult to obtain. However, if a structure is learned with triangulated graph representing them, it may be simpler overall to use the decomposition to compare evidence. Therefore, it is of interest to examine the structure of the data obtained from the DyViS database, and investigate if using the algorithm enables the use of a decomposable density to identify speakers and compare it to the strength of the evidence obtained by a more general representation, where the likelihood ratio is calculated for each vowel keyword on the log scale.

The new algorithms were therefore applied to the data from the DyViS database. It is important to note, however, that in this context, the main goal is to identify speakers using likelihood ratios on the log scale. In order to calculate these, it is necessary to have a decomposable graph. Therefore, it may be necessary to modify the structure learned by the algorithm to ensure it is triangulated, and hence decomposable to obtain the requred likelihood ratios.

7.3 Application of the algorithm

As the first step in order to examine the relationship between the frequencies and the overall structure for the vowel sounds, the PC algorithm was initially applied to the combined dataset as a point of comparison to the new algorithm, shown in Figure 7.2. Overall, the graph is fairly well connected, often indicating relationships between vowels of the same frequency, as well as the same formants across different frequencies. While this may suggest that picking a single frequency for a vowel, or a single vowel across frequencies might be helpful to reduce the number of variables, there are a number of edges that connect different frequencies of different vowel sounds. While this may help simplify the calculation of the joint density, the factorization can become complicated due to the connected frequencies. Next, the new method was also applied, starting with HSIC tests in the pairwise phase, then using Estimated Mutual Information estimates with using a value of 5 for k.

Figure 7.3 shows the initial graph using HSIC tests in the pairwise phase, resulting in a graph with over 900 cliques of order 3. Given that the number of edges is way too high, the cliques were not further examined for the conditional phase, but it was noted that the the links between the frequencies of different vowel sounds may make it difficult to isolate a smaller number of variables. Afterwards, the case using EMI during the pairwise phase was also examined, shown in Figure 7.4 and Figure 7.5.

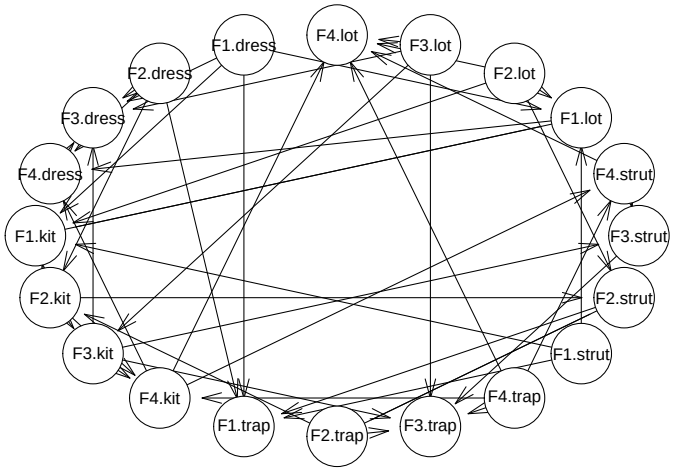


Figure 7.2: Structure learned by applying the PC algorithm. Several links are present for the same frequencies and the same vowel sounds at the same time, with a number of links that are unrelated to them.

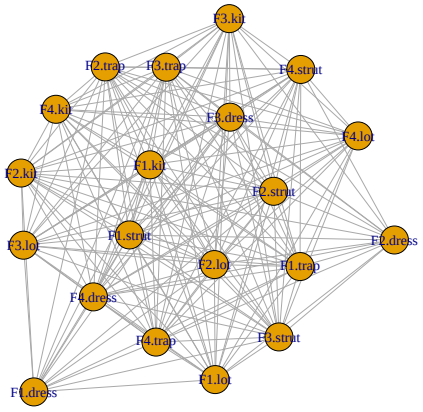


Figure 7.3: Structure learned by applying Algorithm 5 after pairwise phase. Over 900 cliques of order 3 were found between the 20 variables with no recognizable pattern, making the structure unfeasible for the conditional phase.

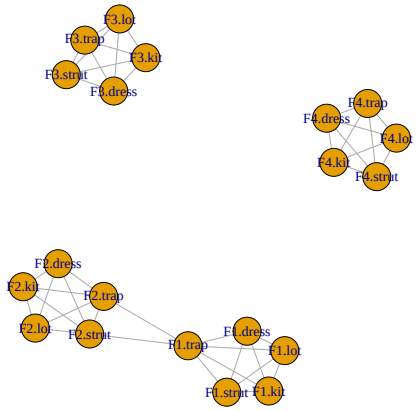


Figure 7.4: Structure learned by applying Algorithm 4 after pairwise phase, k=5. Frequency 3 and 4 formed isolated, complete subgraphs for the vowel sounds. The same is almost true for Frequency 1 and 2, with the additional clique between F1.trap, F2.trap and F2. strut.

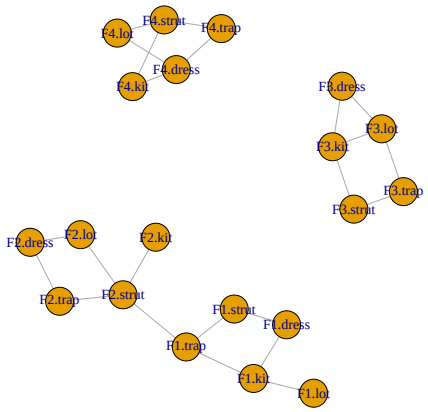


Figure 7.5: Structure learned by applying Algorithm 4 after the conditional phase, k=5. Frequency 1 and 2 remained linked through F1.trap and F2.strut, but several cliques of order 3 were eliminated for each frequency. The resulting graph is not triangulated.

As a contrast to using HSIC tests, using Mutual Information in the pairwise phase indicated all frequencies formed a complete subgraph for each formant, with the exception of a link between F1 and F2. Afterwards, the conditional phase further reduced the number of edges,

although for each vowel sound, the subgraphs remained connected, and the link between vowel sounds **trap** and **strut** remained for F1 and F2. Nevertheless, as opposed to the case with using HSIC tests initially, or the output of the PC algorithm it, there does seem to be a clearer separation between variables. The results suggest that it may be plausible to reduce the number of dimensions used to identify a speaker – potentially by choosing a keyword from each formant frequency.

In addition, the subgraphs representing the different frequencies may be helpful in simplifying likelihood ratio calculation for comparison of evidence. As for each of the frequencies every vowel keyword is connected after the conditional phase (and complete after the pairwise phase) it may be plausible to consider the joint density of each frequency, especially for F3 and F4. These frequencies are additionally considered less influential in identifying a speaker and could then justify wishing to reduce their dimension. For F1 and F2, which are connected by a single edge, it may be possible simplify the calculations needed to compare evidence, potentially by modifying the graph to be triangulated and therefore factorized in a more straightforward way. Either way, it is worth noting the dependencies for the keywords within each frequency moving forward. Overall, applying the algorithm did give some insight of the structure of the data that could be useful for forensic purposes, such as identifying a speaker.

7.4 Identifying speakers

As the next step, the data obtained from the DyViS database was used to obtain log-likelihood ratios to compare different speakers. For these calculations, it needs to be decided how the joint density of 20 random variables is examined, which for this setting could potentially considered to be Gaussian.

One approach is to consider each random variable of interest independent. While a rather naive assumption, this then requires comparisons for all 20 variables, then adding the obtained values (on the log scale). Another approach is to assume that the keywords are independent of each other. This simplifies the problem, although Figure 7.5 suggests this might not be representative, as within each frequency, the vowel sounds do appear connected. Finally, it is possible to find a triangulated graph to decompose the joint density.

While the structure learned by applying the algorithm is not decomposable, Figure 7.5 shows that the graph could be modified to be triangulated fairly easily by adding a few edges. In order to factorize the density, four additional edges were added, as shown in Figure 7.6.

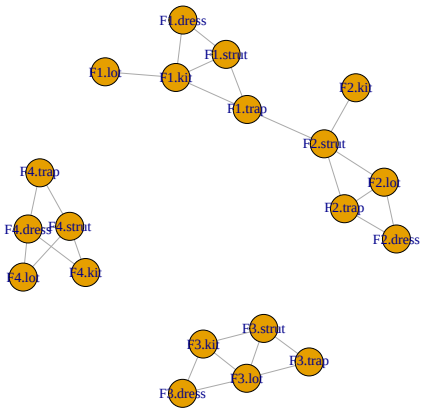


Figure 7.6: Triangulated graph based on structure learned by Algorithm 5. 4 additional edges were added one between F4.strut and F4.dress, one edge F3.strut and F3.kit, one between F2.lot and F2.trap and one F1.kit and F1.strut. The graph is now decomposable.

Then, using this representation the joint density can be factorized. For the subragph of

Frequency 4, one has

$$f(F4.dress, F4.strut, F4.trap, F4.lot, F4.kit) =$$

$$\frac{f(F4.dress, F4.strut, F4.trap)f(F4.dress, F4.strut, F4.lot)f(F4.dress, F4.strut, F4.kit)}{f(F4.dress, F4.strut)f(F4.dress, F4.strut)}.$$

For Frequency 3

$$f(F3.dress, F3.strut, F3.trap, F3.lot, F3.kit) =$$

$$\frac{f(F3.dress, F3.kit, F3.lot)f(F3.kit, F3.lot, F3.strut)f(F3.lot, F3.strut, F3.trap)}{f(F3.kit, F3.lot)f(F3.lot, F3.strut)}.$$

Then, for the subgraph of Frequencies 1 and 2, the factorization is

$$f(F1.dress, F1.strut, F1.trap, F1.lot, F1.kit, F2.dress, F2.strut, F2.trap, F2.lot, F2.kit) =$$

$$\frac{f(F2.dress, F2.lot, F2.trap)f(F2.trap, F2.lot, F2.strut)f(F2.strut, F2.kit)}{f(F2.trap, F2.kit)f(F2.strut)f(F2.strut)}$$

$$\frac{f(F2.strut, F1.trap)f(F1.trap, F1.strut, F1.kit)f(F1.kit, F1.strut, F1.dress)f(F1.kit, F1.lot)}{f(F1.kit, F1.strut)f(F1.kit)}$$

As the factorization has been obtained, some comparisons were made to examine different approaches to identifying speakers. First, 8 observations from the first and the second speakers were compared, then 4 observations from the same speaker. As the caluclations were made on the log scale, a value below 0 suggests that the speakers are different, while a value above 1 would indicate the evidence came from the same source. The lower a value below 0 is, the more likely it can be considered that the speakers are different, and the higher above 0 a value is would suggest it is more likely that the speakers are the same. Using these

values as the strength of evidence, the different approaches can be compared.

Assuming all the random variables are independent, the log-likelihood ratio was calculated to be $\mathbf{-33.58}$ for different speakers, and $\mathbf{14.34}$ when the observations came from the same speaker. While by themselves these values may not be particularly informative, it is rather clear that the ratios are not close to 0, making the evidence in support of the same- and different speaker proposition stronger. This would suggest that even with a rather naive approach, it is plausible to distinguish the speakers based on the vowel sounds.

Next, the case when the keywords are assumed to be independent were examined. When the observations compared are from different speakers, the log-likelihood ratio was $\mathbf{-30.02}$, and $\mathbf{12.39}$ when they came from the same speaker. While still far from the value of 0, both cases appeared to have slightly weaker evidence to distinguish the speakers than assuming all variables independent. It is worth noting however, that the learned structure suggests that it may not be reasonable to assume the vowel sounds independent, as they were rather well connected within each frequency.

Finally, the ratio using the factorization was also calculated. When the observations came from different speaker, the log-likelihood ratio was $\mathbf{-40.83}$, and $\mathbf{8.54}$ when the speaker was the same, both values rather different from 0. This suggest that among all the approaches, the evidence here is the strongest for – correctly – distinguishing different speakers, but the weakest recognizing the same speaker. Therefore, depending on context, it may be preferable to use this approach, especially as the justification for accounting for dependence relations could be relevant.

Overall, it appears that using log-likelihood ratios is a viable choice to identify if a sound came from two different speakers or the same one. While the results show that even when using the naive assumption that all variables are independent, there is sufficient evidence to distinguish the sources. Applying the algorithm gave some insight however, that it may be worth considering dependence relations between the random variables of interest, assuming independence less plausible. It also lead to an easy way to obtain a triangulated graph, which allowed decomposition of the density. The log-likelihood ratio using the factorization than gave stronger evidence when the source of observations were different than other approaches, although it was weaker when they came from the same source. Nevertheless, depending on

context, this approach might be useful in the future to identify speakers, while also exploring and accounting for the dependence relations in the data.

Chapter 8

8 Conclusion

The main objective of this thesis was to create a structural learning algorithm that is able to establish dependence relations that are not between discrete or Gaussian variables, and to dependencies that are not necessarily between the means. In order to achieve this, the problem was approached using graphical models, which allow the visual representation of dependence relations between a set of random variables through graphs. While there are a number of such algorithms that have been used extensively and are effective, they are somewhat limited by the type of relationships they can learn. Current methods excel at learning a network structure when the variables of interest are discrete, or if continuous, they are Gaussian. Therefore, the aim was to develop methodology that can learn the structure of data that is continuous, but not Gaussian while still representing them via graphs.

Section 2 laid out the topics relevant to the thesis. It covered an introduction to graph theory, describing key definitions, including the Markov property and how it can be used to represent independence. This was followed by the fundamentals of Bayesian Networks, how they can be useful for inference, as well as the idea of equivalence classes, which often leads to difficulties during causal inference. Some details on probabilistic and causal inference were given, underlining the need for structural learning. The PC-algorithm and the Hill-Climbing algorithm were described, demonstrating the main approaches to learn overall structure – either through constraints, such as independence tests or through scoring functions using optimization techniques – as well as the key steps to do so.

As the following step, some background on kernels were described. Some key properties of kernels are introduced, along with a few examples for their application on learning problems. Then, after describing the kernel trick and mentioning a few common kernels, dot products were then extended to a more general level. Both the case of defining a kernel or the mapping first were examined, along with some implications. The concept of Reproducing Kernel Hilbert Spaces was introduced. Finally, the importance of the choice of kernels was briefly discussed, as the right kernel may be useful to represent information about non-Gaussian continuous data.

Afterwards, another approach was considered to examine the network structure, with the

introduction of Mutual Information. Therefore, some of the basics of Information Theory were reviewed. This included a brief description of information as a measurable quantity, leading to the laws of information. Using the average of information, entropy was introduced, a key element to define Mutual Information and its potential in measuring independence. After the initial description for discrete random variables, the extension to continuous variables was presented. An appealing feature is that while potentially difficult to calculate, the framework can be used for both the pairwise and the conditional case.

Section 3 then continued on to review key methodology based on these previous topics. First, to establish pairwise independence, a plausible approach was the use of kernel methods. Therefore, an overview on Kernel Covariance is covered in terms of measuring independence, potentially even for non-Gaussian continuous settings.

Then, some later variants of the Kernel Covariance were examined and their capacity as a tool to measure independence was established. One such version using the Hilbert-Schimdt Independence Criterion, is introduced, capable of testing independence between non-Gaussian random variables. This results in a powerful potential tool to use, specifically during the pairwise phase.

While using kernel methods may enable to learn the pairwise dependence structure, and there have been great strides taken to use these methods to establish conditional independence, extending these methods to the continuous phase did not appear readily available. Therefore, it was of interest to examine whether there are other, more straightforward ways to achieve this.

Returning to information theory, it was established that for continuous random variables, it is often not plausible to calculate Mutual Information due to the complex form of the differential entropies necessary. Nevertheless, as it can normally be extended to examine conditional and not strictly linear dependencies between continuous random variables, it was of interest to see if there is a way to use it. Therefore, a way to estimate Mutual Information for continuous variables is described, when the exact measurement is not readily obtainable. The approximation of entropy using the Kozachenko-Leonenko estimator is first described and then is extended to obtain estimates for Mutual Information, as well as Conditional Mutual Information through the nearest neighbour method.

Section 4 gathered all previous building blocks in order to create a structural learning algorithm. It described the choice of using either the estimated Mutual Information, or independence tests based on the Hilbert-Schmidt Independence Criterion for the pairwise phase. For Mutual Information, this included choosing the parameters of the approximation and the approach to interpret the obtained estimates. This led on to two different variants within the pairwise phase, one based on ranking, the other on a threshold. The potential effect of different values of K on the approximation is observed, informing the choice for a threshold. As a rule of thumb, a threshold of 0.5 appeared plausible in the pairwise phase, as it appeared to be a low enough value for MI regardless of the choice of K to be considered close enough to 0. Nevertheless, the value of K and the threshold can be modified manually. Overall, this provided three options during the pairwise phase. Then, for all the three options, during the conditional phase, the approximations of conditional mutual information are used to account for the conditional dependencies. A short illustration was also added for the two main directions – using ranking, or some way to account for all relations during the pairwise phase.

Section 5 then described the main testing ground for these algorithms, a setup using simulated data. After a brief description of the simulation setting, the variants using Mutual Information estimates were applied using different examples, observing how well the method recognizes the relationships within these enforced structures. In addition, the PC-algorithm was also applied for these settings in order to compare performance, illustrated through several examples. Overall, while the PC-algorithm still performed better at recognizing dependencies when the data was Gaussian, there were several instances with non-Gaussian data when the new algorithm performed better.

As a further step, some of the examples were then repeated using different seeds, and the frequencies of specific assigned edges were recorded and compared. Overall, this led to mostly abandoning the ranking method through the pairwise phase, with potential exception to the case of having high-dimensional data and the interest in identifying only the most relevant dependencies. The findings seemed to support that while the PC-algorithm may still be the most plausible choice for Gaussian data, the new method consistently recognised dependencies better in some istances with non-Gaussian data. Part of this research was presented and published at the 4th International Conference on Statistics (Szili et al, 2022).

Section 6 then moved on to apply the algorithm to the Chars74K dataset in order to assist with distinguishing handwritten digits. After describing the data, a brief overview was given on Gaussian Processes (GPs) as well as GP-Latent Variable Models (GP-LVM). After a GP-LVM was applied to the data with 5 and 10 input variables respectively, the new method was applied to the resulting latent variables. Then, using insight gained from the network structure, a few of the random variables were selected to assess how well they separate the handwritten digits. This was performed on two sets of numbers. In addition, model performance was also assessed using silhouette plots. In both cases, using algorithm to choose the latent variables for dimension reduction did separate the handwritten digits fairly well. While model performance appeared less than ideal in the second case, the selected nodes from the network still appeared to do a decent job at differentiating the digits.

Section 7 described the application of the algorithm on a dataset containing Vowel sounds. After a brief introduction to phonetics, a common approach to comparing evidence in forensic science was described, as well as some background on factorizing decomposable models. Then, the data used is described, followed by examining the network structure for a sample of 25 speakers, ensuring that dataset was balanced. Next, a triangulated graph was created based on the learned structure, and the joint density was factorized. Then, the strength of evidence using the learned structure was compared to alternative approaches through an example, when all variables were assumed to be independent, and when the keywords were considered independent of each other.

The results of all three approaches were fairly close to each other in magnitude, potentially suggesting that using the decomposable model based on the learned structure may be better at distinguishing different speakers than recognizing if the source is the same. Regardless, it also demonstrated that as it is often the case, the simplest model – assuming independence across all 20 variables – still performed very well. Nevertheless, applying the algorithm did provide insight to the underlying structure of the data, which may be relevant in the context of phonetics.

The main goal of this thesis was to develop a structural learning algorithm that can recognize dependencies between continuous, non-Gaussian data. Using HSIC and EMI, such an algorithm was created. After the simulations and applications, it appears plausible that in certain cases, the algorithm may be a better alternative to current methodology. However, there have been a number of areas to improve identified. First, the conditional phase of the

algorithm is not fully automated as of yet. While this does allow one to monitor the change of the structure as it is learned step by step, it makes application of the algorithm more time-consuming. Next, while the threshold values for EMI in the pairwise and CMI in the continuous phase can be set manually, it would be beneficial to develop a test statistic, ideally one that can be compared to some distribution. Finally, it may be necessary to extend application of the algorithm to better understand when it may be a better alternative for structural learning.

While the resulting algorithms do have their limitations they provide a new approach to structural learning when the data is continuous but not Gaussian. Using this approach may be useful in identifying links that may have been missed using a different learning algorithm. In addition, it can be used to aid in finding more relevant variables within a data structure, as well as give additional insight to more complex underlying relationships within a dataset, which could then be used in a number of settings even when model performance may not seem ideal. Finally, as the resulting output of the algorithm is a graph, the visual representation can often be extremely helpful to give a visual presentation of a set of underlying relationships within a dataset.

9 References

P A Abhang, B W Gawali, S C Mehrotra (2016). Introduction to EEG- and Speech-Based Emotion Recognition.

S Achard, D-T Pham, C Jutten (2003). Quadratic dependence measure for nonlinear blind source separation. 4th International Conference on ICA and BSS.

S Acid, L M de Campos (2003) Searching for Bayesian Network Strutures in the Space of Restricted Acylic Partially Directed Graphs. Journal of Artificial Intelligence Research (18): 445-490.

C G G Aitken, D Lucy, G Zadora, J M Curran (2006). Evaluation of transfer evidence for three-level multivariate data with the use of graphical models. Computational Statistics and Data Analysis, vol. 50, pp. 2571-2588.

C G G Aitken, F Taroni (2004).Statistics and the evaluation of evidence for forensic scientists. Wiley, Chichester.

S-I Amari, S Wu (1999). Improving support vetor machines by modifying kernel functions. Technical report, RIKEN.

S A Andersson, D Madigan, M D Perlman (1997). On the Markov equivalence of chain graphs, undirected graphs, and acyclic digraphs. Scand. J. Statist. 24:81-102

F Bach, M I Jordan (2002). Kernel independent component analysis. Journal of Machine Learning Research 3:1-48.

A Bellot, M Schaar (2019). Conditional Independence Testing using Generative Adversarial Networks. NeurIPS

T B Berrett, R J Samworth (2019). Nonparametric independence testing via mutual information. Biometrika, Volume 106(3): 547–566.

B E Boser, I M Guyon, V Vapnik (1992). A training algorithm for optimal margin classifiers. In D. Haussler, editor, Proceeding of the 5th Annual ACM Workshop on Computational Learning Theory, pages 144-152, Pittsburg,PA. ACM Press.

T E de Campos, B R Babu, M Varma (2009). Character Recognition in Natural Images, Proceedings of the 4th International Conference on Computer Vision Theory and Applications, Lisboa.

A Carvalho (2009) Scoring functions for learning Bayesian networks.

T M Cover, J A Thomas (1991). Elements of Information Theory/ Wiley, New York.

D R Cox, N Wermuth (1994). Tests of linearity, multivariate normality and the adequacy of linear scores. Apll. Staist. 45:347-355

G A Darbellay (1999). An estimator of the mutual information based on a criterion for conditional independence. Computational Statistics and Data Analysis, 32(1): 1–17

J N Darroch, S L Lauritzen, T P Speed (1980). Markov fields and log-linear interaction models for contingency tables. Ann. Stat. 8:522-539.

L Devroye, L Györfi, G Lugosi (1996). A probabilistic theory of pattern recognition. Applications of mathematics, 31, Springer, New York.

D Edwards (1999). On model prespecification in confirmatory randomized studies. Statist. Med. 18:771-785

D Edwards, T Havránek (1987). A fast model selection procedure for large families of models. J Amer. Stat. Assoc. 82:205-213.

A M Fraser, H L Swinney (1986). Phys. Rev. A (33): 1134

K Fukumizu, A Gretton, X Sun, and B Schölkopf (2008). Kernel measures of conditional dependence. In Daphne Koller and Yoram Singer, editors, Advances in Neural Information Processing Systems 20, Cambridge, MA. MIT Press.

D Geiger, D Heckerman (1994) Learning Gaussian networks. Technical Report, Microsoft Research, Redmond, Washington, available as Technical Report MSR-TR-94-10

P Grassberger (1988). Phys. Lett. A (128): 369

W Gibbs (1902). Elementary Principles of Statistical Mechanics. Yale University Press.

M A Gómez-Villegas, P Main, P Viviani (2014). Sensitivity to evidence in Gaussian Bayesian networks using mutual information. Information Sciences (275):115–126

A Gretton,O Bousquet, A Smola, B Schölkopf (2005). Measuring Statistical Dependence with Hilbert-Schmidt Norms. Lecture Notes in Computer Science, vol 3734. Springer, Berlin.

A Gretton, R Herbrich, and A Smola (2003). The Kernel Mutual Information. Max Planck Institute for Biological Cybernetics

A Gretton, R Herbrich, A Smola, O Bousquet, B Schölkopf (2005) Kernel methods for measuring independence. Journal of Machine Learning Research (6):2075-2129.

A Gretton, K Fukumizu, C H Teo, L Song, B Schölkopf, A Smola (2008) A kernel statistical test of independence. In Advances in Neural Information Processing Systems 20 (NIPS).

A Gretton, A Smola, O Bousquet, R Herbrich, A Belitski, M Augath, Y Murayama, J Pauls, B Schölkopf, N Logothetis (2005). Kernel constrained covariance for dependence measurement, AISTATS, vol. 10

D Heckerman, D Geiger, D M Chickering (1995). Learning Bayesian networks: the combination of of knowledge and statistical data. Machine Learning, 20:197-243

T Hoffman, B Schölkopf, A Smola (2008). Kernel methods in machine learning. The Annals of Statistics, vol. 36 (3):1171-1220.

A Hyvärinen, J Karhunen, E Oja (2001). Independent Component Analysis. Wiley, New York.

Q Hu, L Zhang, D Zhang, W Pan, S An, W Pedrycz (2011). Measuring relevance between discrete and continuous features based on neighborhood mutual information. Expert Systems with Applications, 38 (9): 10737-10750

S Ihara (1993). Information Theory for Continuous Systems. World Scientific, Singapore.

M I Jordan (1999). Learning in graphical models. MIT Press.

A Kankainen (1995). Consistent Testing of Total Independence Based on the Empirical Characteristic Function. PhD thesis, University of Jyväskylä, 1995.

G S Kimeldorf, G Wahba (1971). Some results on Tchebycheffian spline functions. J. Math. Anal. Appl. 33 82–95.

A Kraskov, H Stögbauer, and P Grassberger (2004), Estimating mutual information. Phys. Rev. E 69, 066138.

D Koller D, N Friedman (2009) Probabilistic graphical models: principles and techniques. MIT Press, Cambridge.

W Lam, F Bacchus (1994). Learning Bayesian belief networks: An approach based on the MDL principle. Computational Intelligence, 10:269-293.

S L Lauritzen, N Wermuth (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. Ann. Stat. 17:31-57

S L Lauritzen (1996). Graphical Models. Clarendon Press, Oxford.

D V Lindley (1977). A Problem in Forensic Science. Biometrika, Vol 64(2) pp. 207-213.

R Nagarajan, M Scutari S. Lèbre (2013) Bayesian Networks in R: With Applications in Systems Biology. Springer.

R M Neal (2012). Bayesian learning for neural networks. Springer Science Business Media Volume 118.

F Nolan, K McDougall, G de Jong, T Hudson (2009). The DyViS database: Style-controlled recordings of 100 homogeneous speakers for forensic phonetic research. International Journal of Speech Language and the Law 16(1): pp. 31–57.

K R Parthasaranthy, K Schmidt (1972). Positive definite kernels, continuous tensor products and central limit theorems of probability theory, volume 272 of Lecture Notes in Mathematics. Springer-Verlag, Berlin.

J Pearl (1988). Probabilistic Inference in Intelligent Systems. Morgan Kaufman, San Mateo.

J Pearl (2009). Causality: models, reasoning and inference, 2nd edn. Cambridge University Press, Cambridge

J Pensar, H Nyman, J Niiranen, J Corander (2017). Marginal Pseudo-Likelihood Learning of Discrete Markov Network Structures. Bayesian analysis, 12(4), 1195-1215

F Pernkopf, R Peharz, S Tschiatschek (2014). Introduction to Probabilistic Graphical Models. Academic Press Library in Signal Processing :989-1064

J Peters, J M Mooij, D Janzing, B chölkopf (2014) Causal Discovery with Continuous Additive Noise Models. Journal of Machine Learning Research (15):2009-2053.

H Reetz, A Jongman (2020). Phonetics: Transcription, Production, Acoustics, and Perception. Wiley-Blackwell.

A Renyi (1959). On measures of dependence. Acta Mathematica Academiae Scientiarum Hungarica (10): 441–451.

J Richiari (2007). Probabilistics Models for Multi-Classifier Biometric Authentication Using Quality Measures, These no. 3954, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland.

J Rissanen (1989). Stochastic Complexity in Statistical Inquiry. World Scientific, River Edge, NJ.

B Roberston , G A Vignaux (1995). DNA evidence: wrong answers or wrong questions? B.S. Weir (Ed.), Human identification: the use of DNA markers, Kluwer Academic Publishers, Dordrecht, pp. 145-152.

P J Rousseeuw (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. Computational and Applied Mathematics (20): 53–65

B Schölkopf, A J Smola (2001). Learning with kernels : Support vector machines, regularization, optimization, and beyond. MIT Press.

B Schölkopf, A J Smola (2005). Support Vector Machines and Kernel Algorithms. Encyclopedia of Biostatistics, 5328-5335.

J Schürmann (1996). Pattern Classification: a unified view of statistical and neural approaches. Wiley, New York.

C E Shannon (1948). A mathematical theory of communication. Bell System Technical Journal (27):379–423.

C E Shannon, W Weaver (1949). The Mathematical Theory of Communication. University of Illinois Press.

P Spirtes, C Glymour, R Scheines (1991): Causal Inference. Erkenntnis (35):151-189.

P Spirtes,C Glymour,R Scheines(1993) Causation, Prediction, and Search. 10.1007/978-1-4612-2748-9.

B Szili, M Niu, T Neocleous (2022). A Structural Learning Method for Graphical Models. Proceedings of the 4th International Conference on Statistics: Theory and Applications. 10.11159/icsta22.113

J V Stone (2015). Information Theory: A Tutorial Introduction. Sebtel Press, Sheffield, England.

A Strehl, J Ghosh (2002). Cluster ensembles – a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3:583–617

M Titsias, N D Lawrence (2010). Bayesian gaussian process latent variable model. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, p: 844–851. JMLR Workshop and Conference Proceedings.

A Tsimpiris, I Vlachos, D Kugiumtzis (2012). Nearest neighbor estimate of conditional mutual information in feature selection. Expert System with Applications (39):12697-12708

V Vapnik, A Lerner (1963). Pattern recognition using generalized portrait method. Automation and Remote Control, 24:774-780.

T S Verma, J Pearl (1990) Causal Networks: Semantics and expressiveness. Uncertainty in

artificial intelligence IV, pp. 69-76. North-Holland, Amsterdam.

T S Verma, J Pearl (1991) Equivalence and synthesis of causal models. Uncertain Artif Intell (6): 255–268

I Vlachos, D Kugiumtzis (2010). Non-uniform state space reconstruction and coupling detection. Physical Review E (82), 016207.

J C Wells, 1982. Accents of English: Volume 2. Campridge University Press.

C K Williams, C E Rasmussen (2006). Gaussian processes for machine learning. MIT press Cambridge, MA, Volume 2.

S Wright (1921). Correlation and causation. J. Agric. Res. 20:557-585.

J Zvárová (1974). On measures of statistical dependence, Časopis pro pěstování matematiky 99.1: 15-29

M Zwiessele (2017). Bringing Models to the Domain: Deploying Gaussian Processes in the Biological Sciences, PhD Thesis, Sheffield University.