Tetrad Constraints Beyond The Linear Case

Using the kernel mean embedding of distributions as feature to classify the existence of tetrad constraints when they are not specified

Utrecht University

Author - Jesse Balster
Student Number - 5546354
Supervisor - Dr. T. van Ommen
Second Assessor - Dr. A.J. Feelders

December 15, 2020

Artificial Intelligence
Department of Information and Computing Sciences
Graduate School of Natural Sciences

Abstract

Tetrad constraints can tell us something about the existence and structure of latent parents shared by groups of observed variables. They are however unspecified in the case that variables are related through non-linear functions because they are defined as a vanishing constraint on the covariance matrix. Still, it might be the case that the distributions of the observed variables contain enough information to judge if a tetrad constraint holds, even if the relations in the model are non-linear. To find out of this is true, a random forest classifier is trained on the kernel mean embedding of the distributions of the observed variables. The classifier is tested against the Wishart test, a statistical test for tetrad constraints, on test data sampled from a multitude of different pure and impure measurement models. It is found that if the test and training data share the same underlying graphical structures and data generating process, then the classifier can beat the Wishart test in cases in which the tetrad constraints are not specified. But if the test data is sampled from a more complex graph than the training data, the results of the classifier degrade. A possible explanation is that the more complex graphs contain distributions that do not exist in the training graphs. Further research is needed to see how the variety of training distributions can be increased without the search space growing too large.

Contents

Chapter 1

Introduction

1.1 Background

Asking and answering causal questions is something that comes natural to humans. 'Why did I miss my bus?' Because I spent 5 minutes talking to my neighbour. 'What would happen if I just stopped writing here?' This thesis would not get finished. Our causal understanding of the world not only allows us to explain the past, it also gives us a means to reason about the future. Under the motto "representation first, discovery second"[1], Judea Pearl pioneered the representation of both Bayesian statistics and causal mechanism in graphical form [2]. Besides giving us the ability to ask and answer causal questions, the graphs also made it easier to reason about the discovery of causal relations.

A popular method to discover the causal graph underlying the observed data is by looking for constraints in the data that the structure of the graph leaves behind. Certain assumptions about the relations between the variables and their noise distributions allow for the (partial) discovery of the true causal graph. Most constraint based discovery algorithms use conditional independence constraints. They are however less effective when we are dealing with unobserved (latent) variables that have a causal effect on the variables that we do observe [3]. It might be the case two variables only seem dependent because there is a third, hidden variable, that is the cause of both.

Learning something about the variables that we do not observe is a difficult problem. There is an infinite amount of structures that we can consider when we are allowed to add extra variables. There are however signals called the tetrad constraints that shared latent parents leave in the covariance matrix of the observed children. These signals can tell us something about the existence and structure of the unobserved variables by only looking at the observed data. Their definition is however restricted to the case where relations between variables are

linear and their noise distributions are Gaussian. This makes sense since the covariance matrix defines how a group of variables is linearly related. It is however not unthinkable that the distributions of multiple variables that share a latent parent will leave a signal even if they are non-linearly related, it is just not captured in the covariance matrix. Looking for such a signal would require a more general method that can learn directly from the distributions of variables.

The last decade has seen a growing interest in causality from the machine learning community. Not only can causal information help to make machine learning algorithms more robust, machine learning theory can help the other way around by identifying the circumstances under which we can learn the causal structure of a model from data [4]. A lot of work has focussed on the special case of learning the causal direction between two variables. In this case it is believed that one variable Y is defined by a function $f(X, \epsilon)$ of the other variable X and a noise term ϵ. If no assumptions about the form of this function are made, inferring the causal direction is impossible. There is enough flexibility to make $Y = f(X, \epsilon)$ and $X = f(Y, \epsilon)$ fit the data. Assumptions such as additive noise restrict the form of f so that the causal structure between the variables can be identified.

A successful approach to learning the causal direction between two variables is the randomized causation coefficient (RCC).The RCC trains a classifier with the kernel mean embedding of the distributions of the variables as feature [5]. A problem of this method however is that it is unclear how to extend it to problems that contain rich latent hierarchies [6], something that is deemed important if we want to build artificial intelligence that is more human-like. A good first step would be to limit the amount of latent structures we need to consider by classifying the constraints that underlie the true causal graph.

1.2 Problem

There are two intertwined problems that this thesis deals with. On the one hand it tries to extend the idea of tetrad constraints in the covariance matrix to data that is non-linearly related by looking at the whole distribution of the variables. The tetrad constraints are the special four variable case of the more general notion of rank constraints on the covariance matrix. Just as conditional independence is linked to the graphical criterium of d-separation, the rank constraints on the covariance matrix are graphically represented as t-separations. Both the idea of rank constraints on the covariance matrix and its graphical interpretation t-separation can only be applied when the relations between variables are linear. So the general problem we are trying to solve is seeing if the whole distribution of non-linearly related variables contains enough information to discern different cases of rank constraints in the covariance matrix. The

graphical criterium t-separation will come in handy here because it will allow us to know when we should expect a rank constraint to hold.

On the other hand it tries to extend the idea of the RCC to latent structure discovery. It is unclear how to use the RCC to classify complex structures without adding a label for each case. To solve both these problem we will use the ideas behind the RCC to make a classifier that uses the kernel mean embedding of the observed variable distributions as feature to discern different cases of rank constraints in the data. We will then test if this method also works on cases where the idea of t-separation is undefined.

Thus, the main questions we are trying to answer is:

Can the randomized causation coefficient tell us something about the latent variable structure in the cases that the rank constraints on covariance matrices are not applicable?

To answer this question, we will first need to exactly specify how latent variable discovery works when it is framed as a classification problem. That is why the first sub-question we will answer is:

How is latent variable discovery formulated as a classification problem?

After it is clear how latent variables and structure can be discovered via a classifier using the KME of distributions as feature, we need to see how this compares to methods based on rank constraints on the covariance matrix of linearly related data. That is why the second subquestions is:

How does a classifier using the KME of distributions as feature compare to other statistical tests that test for t-separation constraints?

Chapter 2

Literature Review

2.1 Representing causal relations

Before we can discuss the discovery of causal structures we first need a way to represent them. An often used representation are directed acyclic graphs (DAGs). In a DAG nodes are connected by directed edges and there is no path that follows the direction of the edges that will visit a node twice.

A famous use of DAGs is for representing a joint distribution of random variables in the form of a Bayesian network. Bayesian networks exploit the conditional independences between variables to compactly represent their joint probability distribution. The joint distribution is factorized into conditional probabilities that represent the distribution of a variable as conditioned on its direct influences. These conditional probabilities $P(X|Y)$ are made graphical by a node X having as parents the variables in Y. The conditional independences between variables are represented in the Bayesian network graph through the concept of d-separation.

It is tempting to see the directed arrows in a Bayesian network as causal relations but this is wrong. An arrow $X \to Y$ in a Bayesian network denoting a correlation between X and Y can imply according to Reichenbachs common cause principle that X causes Y, Y causes X or there is a confounder $X \leftarrow Z \to Y$. The Bayesian network only has to represent the underlying joint distribution. Often the graph that does this is not unique, which stands in contrast to the unique graph that represent the causal structure of the data [7].

What is special about causal graphs is that they allow us to do interventions and ask counterfactual questions. An intervention fixes a variable's value so that we can observe how this changes the rest of the system. Normally this is done with a randomized controlled experiment where one variable is changed while the rest are static or random. When it is not possible to do such an experiment,

causal graphs can be used to infer the effects of interventions from observational data.

To see a DAG as a causal graph, we can assume it was induced by an underlying structural causal model (SCM) [8]. An SCM consists of two sets of variables U and V. U stands for *exogenous variables* (also called error terms) which are external to the model, their cause is not explained. V are *endogenous variables*, each endogenous variable has at least one exogenous parent. The value of each variable in V is defined as a function of its parents. If we are given a set of observed random variables $X_1 \dots X_n$, we assume that each observed variable is defined as a function:

$$X_i := f_i(PA_i, U_i),$$
where $i = 1, \dots, n$ and PA_i denotes the endogenous parents of X_i.

These functions are easily translated to a graph where each node X_i its incoming arcs are the parents PA_i and U_i. SCMs are rich in meaning by expressing the observational distribution and the interventional distribution. Although we will not try to infer the exact functions relating these variables for interventional purposes, we will need to make assumptions about the form of the functions to learn something about the causal structure that connects the variables.

2.2 Learning the structure of a causal graph

Learning a causal graph can be split into the learning the structure and learning the parameters. We will focus on learning the structure, which is about finding a set of nodes and vertexes connecting them to accurately display the underlying causal relations. Learning causal graphs has two levels of difficulty. The classical statistical problem is that the observed data will not accurately represent the population. But even if we could obtain the data truly representing the population, we would still not be sure about the causal relations. The functions in a SCM can be so expressive, that many graphical structures can be linked to an SCM that lets them induce the same observed distribution [3]. Assumptions about the graph structure, function classes and noise terms are therefore essential to limit the amount of graphs we need to consider and make the true causal structure identifiable.

2.2.1 Assumptions

There are three important assumptions that are often made about the relation between the observed data and the underlying structure generating it:

- The causal Markov condition

- The faithfulness condition
- The causal sufficiency assumption

The Markov condition states that if a distribution is Markov relative to a graph, then a d-separation in the graph implies a conditional independence in the data. This is equivalent to assuming that each variable is conditionally independent from its non-descendants given its parents. It also allows us to state that two graphs are Markov equivalent if the same set of distributions is Markovian relative to both graphs. The assumption that a distribution is Markov relative to a causal graph is called the causal Markov condition. This property is important for causal inference because it means that we only need to control for a variable's parents if we want to measure its effect with an intervention.

Faithfulness assumes that a conditional independence in the data implies that there is a d-separation in the underlying causal graph. Assuming faithfulness allows us to learn the structure of the causal graph on the basis of conditional independences in the observed data. Together the causal Markov condition and faithfulness assure there is a one-to-one correspondence to the independences that we can read from the graph and the conditional independences in the distribution. This allows us to find the class of Markov equivalent graphs, graphs that entail the same set of independences. It will however not guarantee that we can find a unique graph.

A set of variables is causally sufficient if there is no hidden common cause that is causing more than one variable in the set. A common cause C of X and Y means that there is a directed path from C to X and a direct path from C to Y, that both do not include X and Y. A common cause is also called a confounder [3]. Causal sufficiency is important because it assures us that the results of an intervention will not be muddled by unmeasurable effects. Unfortunately, the assumption is unrealistic because often it is not possible to measure every variable that might be of influence.

2.2.2 Dealing with latent variables

Letting go of causal sufficiency means we have to find a way to take the existence of latent variables into account when learning the structure of the observerd variables. Learning a DAG with latent variables is a highly non-trivial problem because of the infinite amount of options you can consider (for a specific list of problems see [9]).

A solution is to make assumptions about the structure of the latent variables, limiting the amount of cases we need to consider. A strong constraint would be to assume that the causal graph is biparte. This splits the variables

into two groups and only allows arrows from one to the other. The latent variables will be the group with outgoing arrows and the observed variables will be the group with incoming arrows. Assuming this structure will allow us to consider a lot less latent variable models, making the problem manageable. A relaxation of this constraint would be to assume that the underlying causal DAG between the observed variables is sparse, and a few hidden variables have a direct effect on many of the observed ones. Making this assumptions covers important real-world applications [10]. I will refer to graphs that have this general structure, with the addition that we also take structure between the latent variables into account, as multiple indicator models.

Multiple indicator models can be divided into a structural and measurement model. The structural model contains the connections between latent variables. The measurement model contains all other connections, which are the connections from latent variables to observed variables and the connections between observed variables [11]. If every node in the measurement model has one parent in the structural model and no parents from the measurement model we call it a pure measurement model [12]. The problem of learning the structure of latent variables consists of two steps. First, discovering the correct measurement model. Second, discovering the correct structural model.

2.2.3 Causal discovery methods

Most methods for learning causal structure fall into the categories of score based and constraint based. Score-based methods use a scoring function to find the graph that best fits the data. This is an NP-hard problem where the amount of graphs one needs to consider grows exponentially with the variables in the model. Therefore methods often rely on step by step improvements to reach a maximum. If latent variables are taken into account, searching the space of possible models becomes intractable [7].

Constraint-based methods often refers to methods that rely on independence tests. The variables in the data are tested for (conditional) independence, and the graph is made such that it captures the found independences. Independence test based methods such as the PC algorithm assume that the distribution is faithful and Markovian to the underlying graph and data is causally sufficient, allowing for the identification of the class of DAGs that are Markov equivalent to the true graph. To get the right results one needs a correct threshold to accept and reject independence tests, something that is not trivial to obtain. A similar method that deals with latent variables is FCI, which returns the class of MAGs that are equivalent in the independences that they model. The performance of these methods worsens with latent variables. When for example a lot of the observed variables share latent parents, FCI will output a very dense graph. This makes it difficult to orient the edges and thus a lot of uncertainty

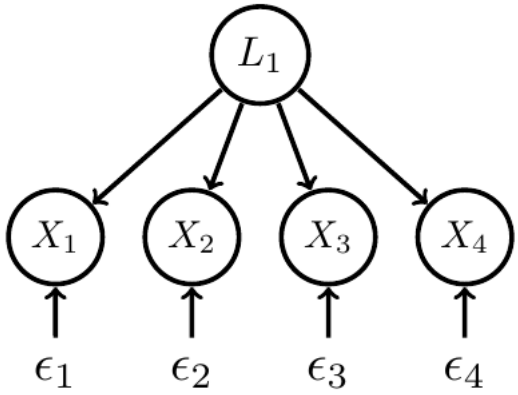


Figure 2.1: An example of a linear model satisfying three tetrad constraints [11].

is present in the final result.

The best we can get with conditional independence constraints on problems with latent variables is a partially ancestral graph that displays the uncertainty we have about the correct causal graph. This PAG will contain the correct causal graph, but it will possibly also contain a lot of uncertainty when dealing with latent variables.

2.2.4 Causal discovery based on tetrad constraints

Another kind of constraint to consider are the Tetrad constraints. They were discovered by Charles Spearman in 1928 [13]. He wanted to show that cognitive abilities are organized hierarchically and that general intelligence tops the hierarchy. So Spearman was trying to see if multiple highly correlated random variables share one latent parent (general intelligence), which is an instance of a multiple indicator model. To find the latent parent we need to assume that there is a linear model with at least 4 random variables that are functions of an unmeasured variable. An example of this model can be seen in figure 4.1. In this model every pair of observed variables is dependent if you condition on any other observed variable or set of observed variables. Thus, looking for conditional independence makes no sense. The latent parent does however leave a signal by entailing three constraints on the correlations (covariances) of the measured variables, which can be seen below.

$$\rho_{13}\rho_{24} - \rho_{14}\rho_{23} = 0 \tag{2.1}$$

$$\rho_{12}\rho_{34} - \rho_{14}\rho_{23} = 0 \tag{2.2}$$

$$\rho_{13}\rho_{24} - \rho_{12}\rho_{34} = 0 \tag{2.3}$$

An example of a model where only the first tetrad constraint is entailed is seen in figure 4.2. Important to notice is that the constraints give a general indication of the model. Some examples of adjustments that could be made to the graph in figure 4.2 while still entailing the same constraint [11]:

- Inverting the arrow between L_1 and L_2.
- Inverting the arrow between L_1 and X_2.
- Adding a shared parent L_3 for L_1 and X_3.

Each X_i in figure 4.1 is defined by the function $X_i = \lambda_{i1}L_1 + \epsilon_i$, where λ denotes the linear coefficient and ϵ the error term. For the error terms we will assume that they have a mean of zero and are independent from each other and the latent variable. So the population covariance of each observed variable is $\sigma_{ij} = \lambda_{i1}\lambda_{j1}\phi$, where ϕ stands for the variance of L_1. If one replace the sample covariances with the definition of the population covariances in the tetrad constraint equations, it can be seen that in the case of three t-separation constraints both terms in the substraction contain the same ingredients for all three equations. But in the case of the one t-separation constraint case of figure 4.2, the terms are not equal in two of the three equations because one of the terms captures the variance of both L_1 and L_2. In the ideal case we could find the tetrad constraints in the population, but due to sampling errors often not all tetrad constraints will vanish [14].

Finding tetrad constraints requires a statistical test that takes into account the average sampling error that will occur when using the sample covariance matrix. John Wishart deviced a statistical test that checks whether a non-zero tetrad constraint should or should not hold [15]. The test is based on the Wishart distribution, which captures the distribution of the covariance matrix of normally distributed variables. The Wishart test checks how likely it is that the non-zero tetrad constraints should hold, given the variance of the Wishart distribution. The Wishart test has been used in multiple causal discovery methods that build a graph based on the tetrad constraints that hold in the data [16] [17].

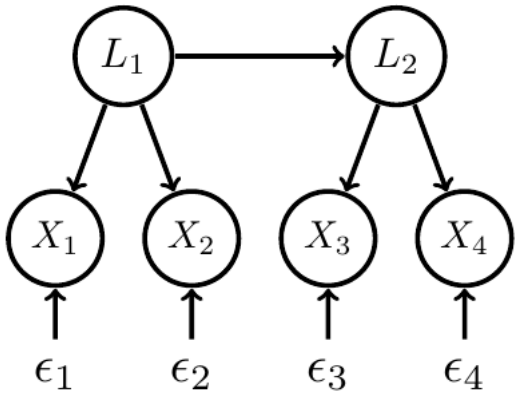


Figure 2.2: An example of a linear model satisfying one tetrad constraint [11].

2.2.5 Tetrad representation theorem and trek-separation constraints

The tetrad constraints were first linked to graphical models through the tetrad representation theorem [18]. The theorem uses the notion of choke points to define the graphical requirements for a tetrad constraint to be present. The idea of d-separation and the tetrad representation theorem were unified under the concept of trek-separation [19]. Trek-separation or t-separation gives a graphical interpretation to the existence of rank constraints in the covariance matrix of Gaussian graphical models (which are characterized by linear coefficients, continuous variables and Gaussian error terms).

A trek between (sets of) nodes X and Y is an ordered pair of directed paths (P_X, P_Y), where P_X has a sink in (meaning that it ends in) X and P_Y has a sink in Y and both paths have the same source Z. An important subtlety in the definition is that the common source Z can also be either X or Y. A set (T_X, T_Y) t-separates X from Y if every trek (P_X, P_Y) from a node in X to a node in Y, either P_X has a node in T_X or P_Y has a node in T_Y [**?**]. So if there is a directed path from X to Y, then the common source is X and any node on the path can block P_Y and thus t-separate X from Y.

In the case of d-separation it can be stated that if variable(s) C d-separates A from B, then the rank of the covariance matrix $\sum_{A\cup C, B\cup C} = \#C$. So the rank of the rows of A and C and column B and C in the covariance matrix are equal to the amount of variables in C. This is another way of saying that if we already know C then knowing A will not give us any extra information about B (or vice versa). The tetrad representation theorem states that if (T_X, T_Y) t-separates variables X and Y, then the rank of the covariance matrix is constrained as

$\sum_{X,Y} \leq r$. Here r stand for $\#T_X + \#T_Y$, where $\#T$ denotes the size of the set.

The idea of t-separation allows us to express rank constraints on the covariance matrix that can not be explained with d-separation. But these situations do not occur in Gaussian tree models [20] (of which pure measurement models are a special case). Still, we will use the term t-separation because its one-to-one correspondence with the underlying tetrad constraints it will make it easier to talk about those in the coming examples.

Interesting work has been done to extend the Tetrad representation theorem to the non-linear and cyclic case [**?**]. A brief intuition of these findings is that as long as the path from latent confounder to observed variable remains free of non-linear relations and does not belong to a cycle, then the identifiability results of the causal structure will remain the same.

But besides the above exception, both the ideas of tetrad constraints on the covariance matrix and its graphical interpretation t-separation are only defined in models with linear path coefficients. This is also because the idea relies on the covariance matrix, which measures the linear dependence between variables. When this is calculated, a lot of information contained in the original distributions is lost. It might be the case that even when the relations between variables are nonlinear, that it is still possible to discriminate between graphs that would contain different t-separations in the linear case. The next chapter will discuss a method that will allow us to learn directly from the distribution.

2.3 The kernel mean embedding of distributions

The past decade has seen a rise in the applications of the kernel mean embedding (KME) in statistical inference. There is for example a kernel based two sample test [**?**], a kernel test for independence [**?**] and a kernel test for conditional independence [21].

The general idea of the kernel mean embedding is that it maps a distribution to a point as seen in figure 2.3. If the kernel used is characteristic, the mapping becomes injective. This means that each distribution is mapped to a unique vector. These vectors thus define a metric that can be used to compare distributions.

2.3.1 Using the kernel mean embedding as a feature

Besides its use in making statistical tests, the KME can also be used as a feature for machine learning methods. Already a method has been developed for identifying the structure of a causal graph by phrasing it as a machine learning problem [5]. The method tries to infer the causal direction between two variables. For this, access is assumed to a set $\{(S_i, l_i\})_{i=1}^n$, where each S_i is

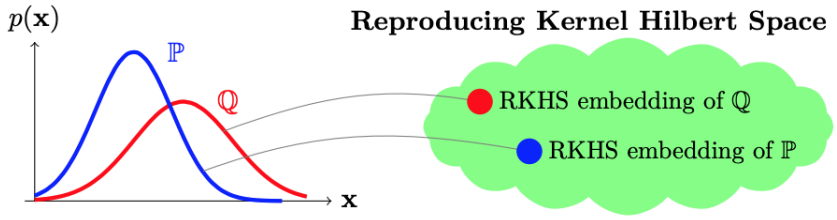


Figure 2.3: A graph showing how the kernel mean embedding maps each distribution to a point in a space [22].

a sample set drawn from a joint distribution P of the sets of random variables X_i and Y_i. The binary variable l_i indicates whether X_i causes Y_i or vice versa. Essential for this method to work is a way to featurize the observations as a probability distribution. To see how this is done, a further explanation of the kernel mean embedding is required.

First off is the kernel function k defined as $k : X \times X \to \mathbb{R}$. The function k is a similarity measure that expresses how equivalent two inputs are with a real number. A famous example of such a measure is the dot product between two vectors. To exemplify the usefulness of a similarity measure we can take a binary classification problem with data of the form $(x_1, y_1)...(x_m, y_m) \in \mathcal{X} \times \{\pm 1\}$ (which is alike to the problem we defined earlier). Given a new example x_i, a similarity measure allows us to find its label by looking at the labels of previous examples that are similar to x_i [23].

Second, there is the idea of a reproducing Kernel Hilbert space (RKHS). A RKHS is a space where each coordinate is an infinite list of numbers representing a function. What makes these spaces very useful is that they give a means to compare the similarity of functions by looking at the distances between the points in the RKHS. For a tutorial introduction of the mathematical details of the RKHS see [24].

The kernel mean embedding of the probability distribution P is defined in equation 2.4. The distribution P of random variable Z takes on values in the separable topological space $(\mathcal{Z}, \tau_z)$. The kernel mean embedding $\mu_k(P)$ is an element in $\mathcal{H}_k$, the reproducing Kernel Hilbert space (RKHS) associated with k. So the distribution P can be defined as a infinite dimensional coordinate in a space. Important about kernel mean embedding is that it is injective when a characteristic kernel is used (such as the Gaussian kernel), allowing us to have a unique representation for each probability distribution that we can compare for their similarity.

$$\mu_k(P) := \int_{\mathcal{Z}} k(z, \cdot) dP(z) \tag{2.4}$$

Often we do not have access to the true distribution P but only some samples from the distribution. So we estimate the kernel mean embedding with the *emperical mean embedding* seen in equation 2.5, where P_S stands for a sample of $\{z_i\}_{i=1}^n$ drawn from a distribution P.

$$\mu_k(P_S) := \frac{1}{n}\sum_{i=1}^{n} k(z_i, \cdot) \in \mathcal{H}_k \tag{2.5}$$

The learning of the causal direction involves featurizing each sample S into a kernel mean embedding $\mu_k(P_S)$ by using a characteristic kernel so that no information is lost. For this to work the authors make two assumptions. The first assumption is that all pairs of distributions between two variables and directions between two variables (P, l) are defined by a mother distribution M. Second, each P holds enough information to predict l.

Because the kernel mean embedding can be infinite dimensional, [5] approximates it by using random Fourier features. Random features allow us to lower the dimensionality of the input and approximate the similarities between the original inputs [**?**]. Because random features are used, the authors call the method the *randomized causation coefficient*. The approximation of the empirical mean embedding with random features gives us a low dimensional featurization of the distributions, which together with the causal direction as label can be fed into a classifier. This method has achieved the best known results on the Tübingen cause-effect pairs, a popular dataset that consists of two-variable pairs with their causal direction [25].

The method can also be extended to classify more than two variables. But extending it from a two dimensional probability distribution to an n-dimensional probability distribution makes the amount of DAGs we need to consider grow super-exponentially in n. This is why [5] suggest to keep it to the basic cases of $X \to Y$, $Y \to X$ and $X \perp\!\!\!\perp Y$ to learn causal graphs. If however, we are dealing with latent variables then there will be constraints on the marginalized graph that can not be discovered by looking at only two variables. If however we are able to discern between pairs of variables being t-separated or not, we will have an intermediate step in which we do not have to add every possible latent configuration as label but we can still say something about the latent variables. To make this work, we will assume that the kernel mean embedding will carry enough information to learn the constraints that the distributions of the variables in these sets might contain.

Previously, a Hilbert space embedding has been used to detect a hidden confounder shared by two or more variables [26]. This method is however limited to finding a confounder that takes on a finite range of values and in practice shows to be only appropriate for a confounder with a small number of values.

2.4 Theoretical difference between classifying the causal direction between two variables and classifying the tetrad constraints

The idea of training a classifier on datasets featurized with the KME to learn the causal direction between two variables is based on the idea of an asymmetry existing between cause and effect [**?**]. This assumes that the cause is causally sufficient to determine the effect and that there is independence of mechanism and input. The last assumption states that the process that generates the values of the cause variable (input) is independent from the process that transforms the values of the cause into the effect (mechanism). If we for example take the relationship between the altitude of a city and its average temperature, we would find that higher altitudes are the cause of lower temperatures. Even if we take a different landscape where the distribution of altitudes changes, the function that transforms altitude into a temperature will remain the same [3]. The asymmetry exists because the function that calculates the causal value from the effect value does depend on the distribution of the effect. So different temperature distributions would lead us to different functions that calculate the altitude from the temperature.

The main point of bringing this up is that classifying tetrad constraints is not based on this theory. The idea of tetrad constraints is that every sample dataset drawn from certain graphs will have the same vanishing points in the covariance matrix. So this relies on the causal graph leaving a fingerprint in the dataset that is invariant given different path coefficient. This is the case when the data comes from a model that has linear coefficients and standard Gaussian noise. It might be the case that t-separated sets of variables leave different signals, even in cases of non-linear coefficients and non-Gaussian noise, which can be detected in their distribution. If this is the case it, it would not be based on the assumption of independence of mechanism and input since this does not transfer to groups of variables sharing a latent parent. So just because the KME of distributions is a useful feature when classifying the causal direction between two variables does not mean that it is trivial that this will also be the case for classifying tetrad constraints.

Chapter 3

Classifying t-separation constraints

The previous chapter gave an overview of the literature surrounding my research topic and explained the ingredients necessary to devise a method that can help us answer the main question of this thesis. This chapter will explain how this method is constructed.
The outline of a method that finds the causal relations between variables is:

1. Make judgements about the causal relations between the variables in our data.

2. Combine these judgements to form a causal graph.

In the case of constraint based methods, the judgements represent the constraints that are found in the data. These constraints can then be used to form a graph. The goal of this paper is to find a new method to test for t-separation constraints in a given dataset. This will be done by featurizing tetrads of variables with the kernel mean embedding, labelling them with t-separation constraints and classifying them with a random forests classifier. The focus will thus lie on part 1 of the program. For part 2 there are multiple existing algorithms that use t-separation constraint to construct a graph [16] [17].

3.1 Kernel Mean Embedding as feature

There are three parameters that can be varied with the kernel mean embedding: which sets of variables to embed, the length of the KME vectors and the weights used for the embedding. In [5] a sample of two variables $S = (x_i, y_i)_{i=1}^{n}$ is

encoded as $v(S) = (\mu(S_x), \mu(S_y), \mu(S_{xy}))$, where μ stands for the kernel mean embedding.

There are a few different options I considered for featurizing tetrads of variables:

1. The kernel mean embedding of each of the four marginal distributions.
   $v(S) = (\mu(S_{y_1}), \mu(S_{y_2}), \mu(S_{y_3}), \mu(S_{y_4}))$
2. The kernel mean embedding of the joined distribution of the two pairs of variables that are being tested on the t-separation constraint.
   $v(S) = (\mu(S_{y_1 y_2}), \mu(S_{y_3 y_4}))$
3. The kernel mean embedding of the joint distribution of the tetrad.
   $v(S) = \mu(S_{y_1 y_2 y_3 y_4})$

The low-dimensional random approximation of the KME of a set of examples S is computed as $\mu_{k,m}(S) = \frac{2C_k}{|S|} \sum_{z \in S} (\cos(\langle w_j, z \rangle + b_j))_{j=1}^m$. Here m stands for the size of each KME vector, which should be increased until no clear improvements are made. C_k is a constant based on the fourier transformation of the kernel function k and b_j is drawn from the uniform distribution $U[0, 2\pi]$. The most important property of this function is that it approximates the Gaussian kernel.

The weights are taken from the distribution of the fourier transformation of the kernel, which in the case of a Gaussian kernel is $\mathcal{N}(0, 2\gamma I)$. γ is found by using the median heuristic, a method that is for example used in finding the bandwidth for a kernel density estimation. In other words, it adjusts the method for the variance found in the data. It can be calculated as $v = \sqrt{H/2}$, where for n random variables X, we have $H = Med\{||X_i - X_j||^2 | 1 \leq i \leq j \leq n\}$ [27]. H takes the median value of the ordered list of all differences between the random variables in our dataset.

With the median heuristic values $[0.1\gamma, \gamma, 10\gamma]$, three Gaussian kernels are estimated and summed to form one kernel mean embedding vector $\mu(S_x)$. In practice the same weights will be used for every single variable, since we can calculate the median heuristic only once for a dataset. It is however unclear if the training data needs to use the same weights as the test data. [27] seems to suggest that this is not necessary since the median heuristic is calculated for a dataset, but in the code from [5] the same weights are used for training and testing. After doing some experiments it can be concluded that not sharing the weights will lead to classifier scores that are no better than a random choice algorithm. This unfortunately leaves open how to exactly calculate the median heuristic, since it is undefined over multiple datasets. Some small experiments have lead me to cautiously conclude that the effect of choosing a different median heuristic does not strongly affect the results of the classifier.

The last thing that elicits some elaboration is the use of the Gaussian kernel function. There are at least a few other kernel functions that have the same properties as the Gaussian kernel function [22] and can be approximated by random fourier features [**?**]. There is however no general heuristic known to choose the correct kernel function, and most literature seem to suggest iterating through the different kernels to find the one that delivers the best results [**?**]. Searching for the best kernel has not been included in this paper because it is not about finding the optimal solution to a problem, but giving a proof of concept that it could work.

3.2 T-separation constraints as label

To learn tetrad constraints from the data, we will label each of the three ways to split the four variables into two sets of pairs with being t-separated or not. So if we have the variables x_1, x_2, x_3 and x_4 then we would have three ways to divide the variables into two pairs:

$$x_1x_2 \times x_3x_4,$$
$$x_1x_3 \times x_2x_4,$$
$$x_1x_4 \times x_2x_3.$$

Each of the three sets is labeled on being t-separated or not. This can easily be done by using the graphical criterium of t-separation [19] since we know the underlying causal graph that generated the data.

3.3 Random Forest Classier

For the classification of t-separation constraints we need a binary classifier because each KME embedding of two pairs of variables is either labeled as t-separated or not. Two methods suited for binary classification tasks are random forest and support vector machines. In practice the difference in score between these two methods is not statistically significant on many datasets [**?**]. Because random forests has already worked on a similar problem it is chosen over other binary classification methods [5].

Chapter 4

Experiment Setup

4.1 Goal of the experiments

The main objective of the experiments is to see how well the kernel mean embeddings fares against the Wishart test in classifying t-separation constraints on tetrads of variables. This is of course a very broad question since there is an infinite amount of settings in which we can compare the methods. To keep the results of the experiments concise, it is necessary to break down how exactly the comparison between methods will be made into various subgoals. The first step is to test the Wishart test on data generated with linear path coefficients, on which it is known to work. The second step is to test the Wishart test on data with non-linear coefficients. To limit the amount of graphs that can be tried, only two elementary graphs containing cases of 3 and 1 t-separation(s) are considered. The first subgoal of the experiments will be to find a setting in which the Wishart test works effectively and one in which it does not. This will be done by searching for a clear decline in the results of the Wishart test on non-linearly generated data.

Once established, these two settings can be used to test different ideas for optimizing the KME. The second goal of the experiments is to see how well different hyperparameters settings of the KME work and which ones should be varied when comparing it to the Wishart test.

When the best settings for the kernel mean embedding are known, they can be compared to the Wishart test. The third goal is to see which method works better on pure measurement models, so we will compare them on the two basic graphs.

Besides the two basic cases of t-separation, there are also multiple impurities that can exist in a measurement model. To see if the KME is more flexible, it will be compared against the Wishart test on data generated by three basic

graphs each containing one of the three impurities. So the fourth goal is to establish which methods works better with impure measurement data.

Lastly, both methods are run on a big impure measurement model with nonlinearly generated data and a big pure measurement model with linearly generated data. The goal of this is to test both methods on a more realistic example.

4.2 General setup for the experiments

For the experiments both training and test data need to be generated because there is a general lack of datasets of which the underlying causal graph is known. The benefit of generating data for causal discovery methods is that the causal direction labels between variables are contained in the way we generate the values of the variables. There are however, as far as I know, no general guidelines in generating data, only recurring practices, on which this section is based.

The process of generating data will consist of making a SEM and splitting it in a latent and observed part. For the training data a sparse SEM with limited noise distributions and simple coefficients is important because it will make the results of the experiments more generalizable to other problems. For the test data generation we will start with elementary SEMs that have a small amount of nodes that cover most of the cases that we will see in SEMs with many nodes.

4.3 Generating Test data

4.3.1 Graphs

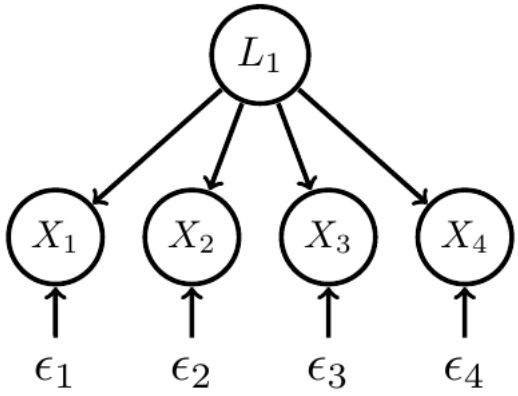


Figure 4.1: Model with three t-separations.

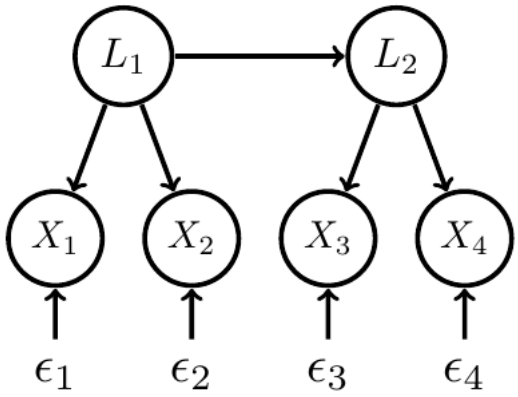


Figure 4.2: Model with one t-separation.

The most simple models for generating test data are the elementary ones that contain 3 or 1 t-separation constraints, as seen in images 4.1 and 4.2 respectively. These two graphs will give a good indication of how well a method would work if the given measurement model is pure.

Besides these graphs with pure measurement models, there are also three impurities that we can introduce that are easy to model graphically. The first one is cross construct impure. An example can be seen in 4.3, where the child X_2 of one latent parent L_1 is the parent of the child X_3 of another latent parent L_2. In this graph 0 t-separations occur between any two pairs of observed variables.

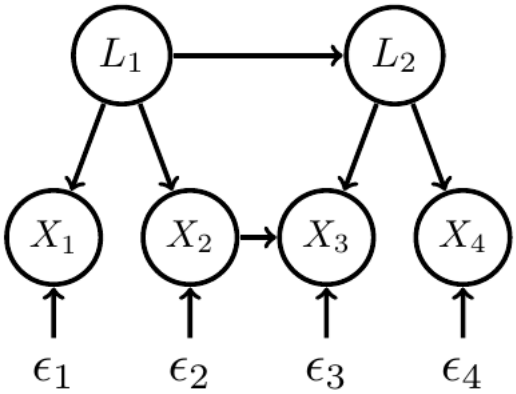


Figure 4.3: An example of a graph with a cross construct impurity.

The second one is intra construct impure. An example can be seen in 4.4, where the child X_2 of latent variable L is also the parent of another child X_3 of L. In this graph, one of the three t-separations between pairs of variables

occurs, namely between (X_2, X_3) and (X_1, X_4).

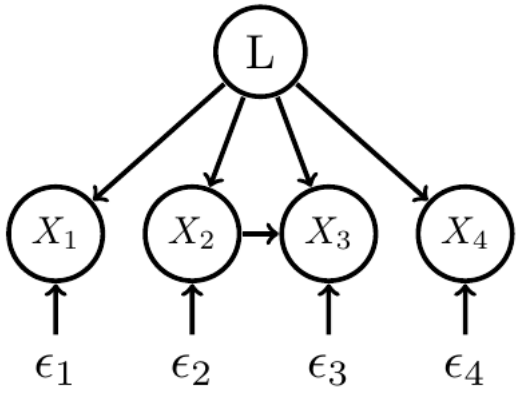


Figure 4.4: An example of a graph with an intra construct impurity.

The third one is latent measure impure. An example can be seen in 4.5, where two latent parents $L1$ and $L2$ share a child X_5. In this graph five of the fifteen possible t-separations between pairs of variables occur. This is the standard one between (X_1, X_2) and (X_3, X_4) and the four variations you get when swapping any one of these variables for X_5.

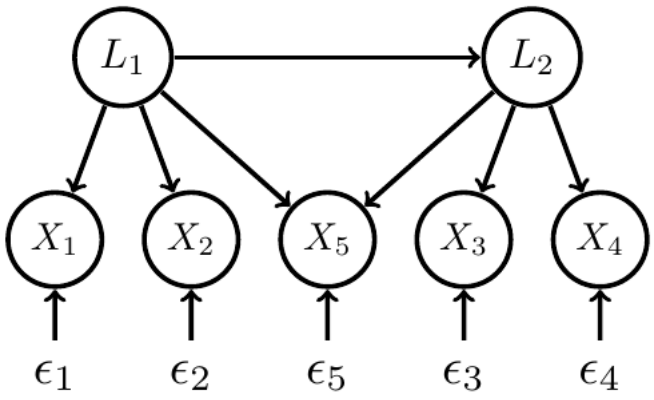


Figure 4.5: An example of a graph with an latent measure impurity.

Besides these basic cases, two larger graphs are included to make the experiments a bit more realistic. This is a large impure measurement model as seen in figure 4.6 and a large pure measurement model as seen in figure 4.7. This model consists of all the basic graphs that we have discussed thus far. We would expect the methods to perform no worse on the large graph than its worst sub-graph performance.

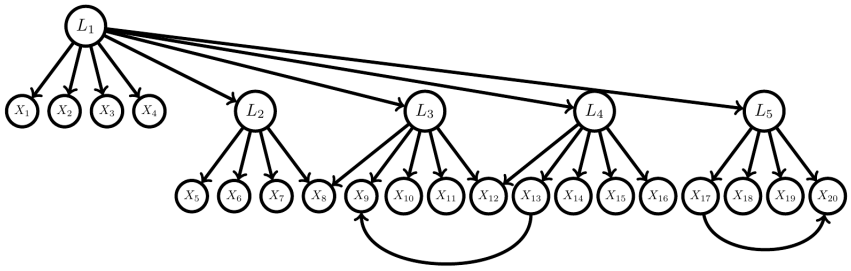


Figure 4.6: Large impure measurement model used for generating test.

The graph from figure 4.6 has five latent variables $L1 \dots L5$, where $L1$ is the parent of all other latent variables. Each latent variable has 4 unique observed children. Furthermore, the cross construct impurity of $X13 \rightarrow X9$, the intra construct impurity of $X17 \rightarrow X20$ and the latent construct impurities of $L3 \rightarrow X8$ and $L4 \rightarrow X12$ are introduced to make the model impure.

The graph in figure 4.7 contains five latent variables that each have four children.

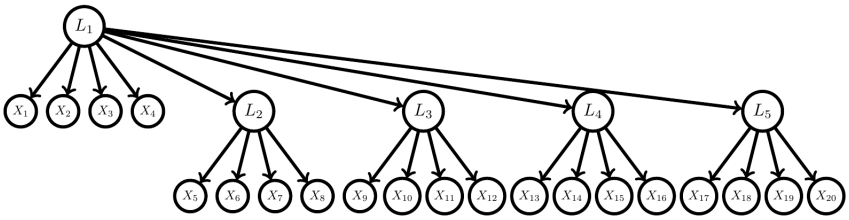


Figure 4.7: Large pure measurement model used for generating test data.

4.3.2 Noise and path coefficients

It is possible to mix linear and non-linear functions between variables within one experiment, but to keep things simple the choice of function class will only vary per experiment.

In the first setting, linear relations between latent and observed variables, we start with calculating the values of all parentless variables ($L1$) by sampling its values from a standard Gaussian distribution. Then the values of all remaining latent variables are calculated by the following procedure:

1. For each parent x of a variable, the vector of effects is calculated with equation aL_x. Here a is drawn from a uniform distribution $\mathcal{U}(0.5, 2)$.

2. All effect vectors for a single variable are added together.

3. The vector is standardized.

4. A vector of noise is sampled from a normal Gaussian and added to effect vector.

5. The vector is once again standardized.

In the non-linear case the process is repeated, but in step one the function $a(1-b)L_x + bcL_x^d$ is used. Here a is drawn from a uniform distribution $\mathcal{U}(0.5, 2)$, c from $\mathcal{U}(0.5, 2)$ and d is a random natural number ranging from 1 till 6. The degree of non-linearity can be varied by adjusting b, where 0 stands for a purely linear relation. To keep the duration of the experiments manageable, b is not varied but set to 0.1.

4.4 Training data

To generate training data we will use two simple graphs as seen in 4.1 and 4.2. Also, experiments will be done by generating training data from the two simple graphs and the three examples containing an impurity. For simplicity, I will use the same steps and settings as the test data generating process in these graphs.

4.5 Hyperparameters

So to summarize and elaborate further on the previous parts, these are the elements that can be varied in the experiments.

Both methods will be tested on different amount of samples per variable: 50, 100, 500 and 1000.

For the random forest classifier with the KME as feature:

- The combination of variables that I will use as feature: the embedding of the joint distribution of four variables, the embedding of the marginal distribution of four variables and the embedding of the two joint distributions of the pairs that are being checked for t-separation.
- The amount of different training distributions generated per graph example: 100, 500, 1000.
- The length of each KME vector: 100, 500 and 1000.
- The amount of trees in the random forests classifier: 100, 500 and 1000.

For the Wishart test the significance level will be varied: 0.001, 0.005, 0.01, 0.05 and 0.1.

4.6 Code

The Github repository containing the code of the experiments: `https://github.com/JesseBalster/RCC_tetrad_constraints`. To perform the experiments code was written in python [28]. For generating data the python packages pandas [29] and numpy [30]where used. The random forests classifier is taken from the python package scikit-learn [31]. Visualization were made with the packages matplotlib [32] and seaborn [33].

Chapter 5

Experiment Results

5.1 Experiment results format

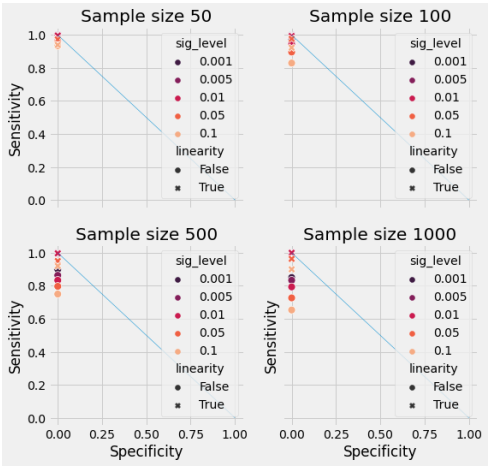
To measure how well the methods performed in the experiments, the sensitivity and specificity are shown graphically. The sensitivity is a measure that displays how many t-separations are correctly identified, where a score of 1 means that there were no false negatives. The specificity measures how many cases of no t-separation are correctly identified, where a score of 1 means that there were no false positives. Each point on the scatter plots displayed in this chapter represent the average of 100 experiments run with a certain hyper parameter setting.

5.2 The Wishart test on non-linear data

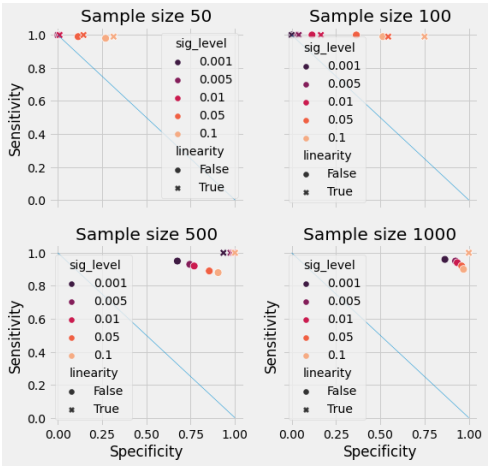
Before the results of the KME are displayed, it is worthwhile to first investigate how well the Wishart test fares on data generated from non-linear path coefficients versus data generated with linear path coefficients.

Figure 5.1 shows the results of the Wishart test on the basic graph with three t-separations and with one t-separation. Since the three t-separation case in 5.1a has no negative labels the specificity is always zero and the sensitivity equals the accuracy. Adding more samples decreases the sensitivity, meaning that the test is more inclined to classify pairs of variables as not t-separated. Also, the Wishart test declines faster on non-linear data than on linear data.

In figure 5.1b it can be seen that in the linear case the Wishart test is able to correctly classify the one existing t-separation and with increasing sample sizes it is able to correctly classify all three labels of the graph. In the non-linear case the specificity improves with sample size, but the specificity worsens. It can also be seen that increasing the significance level does not lead to a better/worse



(a) The Wishart test results on the basic graph with three t-separations.



(b) The Wishart test results on the basic graph with one t-separation.

Figure 5.1: The sensitivity and specificity of the Wishart test with different significance levels used on linearly and non-linearly generated data.

score, but to a trade-off in sensitivity and specificity.

Although the test is more robust than expected on non-linear data, the linearity or non-linearity of path coefficients still has an undeniable effect on the results.

5.3 KME hyperparameter results

	# samples	# distributions	# weights	# trees
Sensitivity	0.242	0.026	0.0079	0.0023
Specificity	0.071	0.018	0.0091	0.0006

Table 5.1: The correlation coefficients between the sensitivity and specificity and the hyperparameters of the KME.

Before discussing the results of the KME in comparison to the Wishart test, it is worthwhile to see how much the different hyperparameter influence the KME in general. All combinations of setting each hyperparamter to 100, 500 and 1000 were tried, except for the amount of samples, which was also set to 50. In table 5.1 the correlation coefficients display the effect of the hyperparameters on the sensitivity and specificity of the experiments. It can be seen that the greatest effect is achieved by increasing the amount of samples. The amount of

distributions has a smaller effect. The amount of weights and trees have very little impact on the sensitivity and specificity.

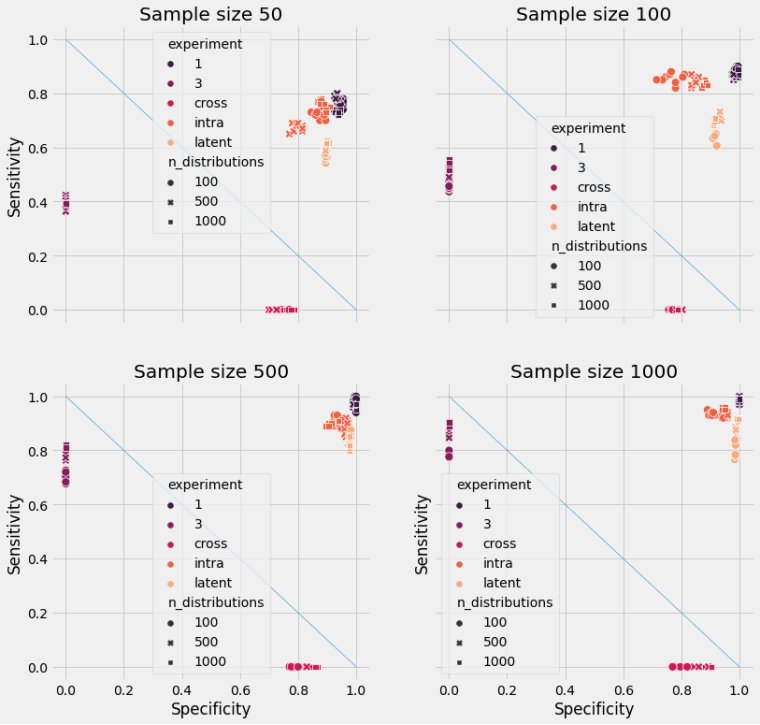
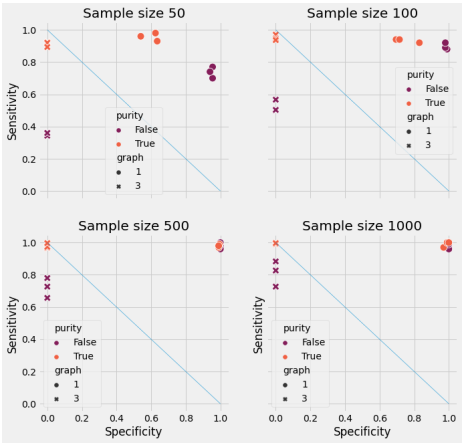


Figure 5.2: The sensitivity and specificity of the different experiments done with the KME.

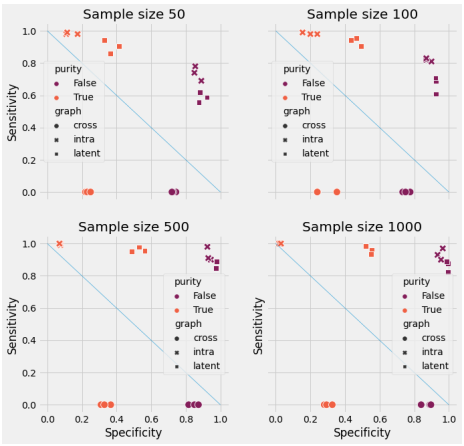
The same effects can be seen in figure 5.2, where the sample size results in the experiments being clearly grouped, and the amount of distributions explains most of the variance within a group. This is also in accordance with [5]. There it is stated that the weight vectors need to be a certain size to contain all information about the distribution, but beyond this size there are no improvements. Random forest classifiers are also known to increase accuracy with the amount of trees until a certain threshold, at which point the gains come to a halt [**?**]. It is thus clear that setting the amount of trees and the size of the weights at 500 each will give a good enough indication of how well the KME works. A final note is that in these experiments non-linear path coefficients were used, but the same results occured when using linear path coefficients.

A second question regarding the KME is how training data sampled from graphs containing an impurity will affect the results. For this experiment non-linearly generated data was used, the amount of weights and tree size were set to 500 and the amount of distributions was varied between 100, 500 and 1000.

In figure 5.3 the KME is trained on data sampled from the two basic graphs and data sampled from the two basic graphs plus the three impure graphs. As seen in 5.3a, sampling impure training data worsens the results on the two basic graphs, especially on the graph with three t-separations. It does however improve all instances of the graphs containing an impurity, as seen in figure 5.3.



(a) The KME test results on the basic graph with three and one t-separation(s).



(b) The KME test results on the graphs containing an cross construct, intra construct and latent measure impurity.

Figure 5.3: The sensitivity and specificity of the KME test used on non-linearly generate data, where the three points per class each represent the average of 100 experiments with a different distribution size.

The last experiment with the KME will be to see how different embeddings affect the results, as seen in table 5.2. Each row of the table represents a different KME embedding, as discussed in section 3.1. The data is generated with non-linear functions and the classifier is trained on a mix of pure and impure graph examples. The extreme results of the marginal embedding can be explained by it almost always deciding on no t-separation. The embedding of the two pairs is however fairly close to the embedding of the joint distribution. On the 3 t-separation case it is even better, but this can be explained by it having a higher false positive rate on any other problem. So the embedding of the joint distribution gives the overall best results. This is also the embedding the encodes the most information and needs to be calculated uniquely for every tetrad. From now on all experiments will use the joint embedding as feature for the classifier.

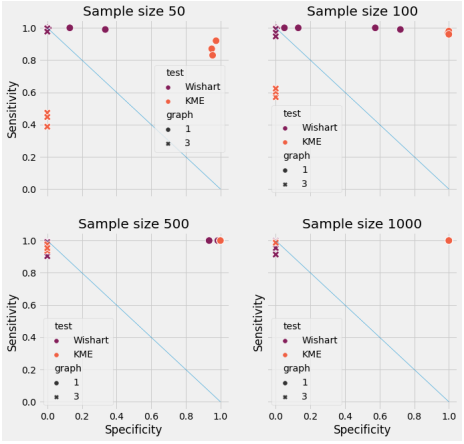
	1 t-separation	3 t-separations	cross construct impure	intra construct impure	latent measure impure
Joint	95.4%	60.3%	80.2%	89%	87.96%
Marginal	66.2%	3%	97.8%	65.8%	66.5%
Pairs	89.1%	74.8%	76.6%	59.2%	83%

Table 5.2: The accuracy of different KME embeddings.

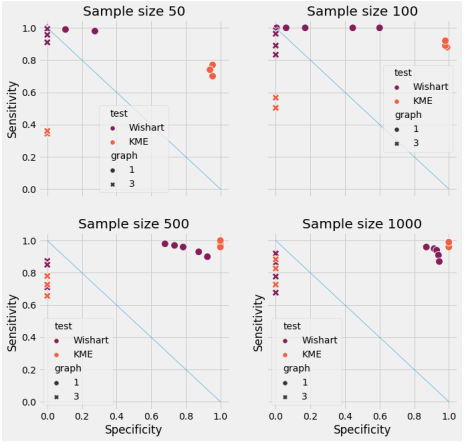
5.4 Comparing the KME with the Wishart test

For this section the amount of trees in the random forest and the size of the KME weights will each be set to 500 and the training data will be sampled from both pure and impure graph examples. The points on the graph represent the amount of distributions set at 100, 500 or 1000 for the KME and the significance level set to 0.001, 0.005, 0.01, 0.05 or 0.1 for the Wishart test.

5.4.1 The two basic graphs



(a) Graph comparing the Wishart vs the KME test on data generated with linear path coefficients.



(b) Graph comparing the Wishart vs the KME test on data generated with non-linear path coefficients.

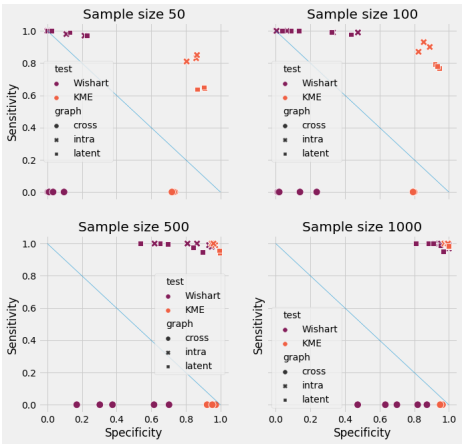
Figure 5.4: The sensitivity and specificity of the Wishart tested versus the KME on data sampled from the basic graph with one t-separation and with three t-separations. Each point represents the average of 100 runs with either a different distribution size for the KME or a different significance level for the Wishart test.

The results of the KME versus the Wishart test run on data sampled from the two graphs containing one and three t-separations can be seen in figure 5.4.

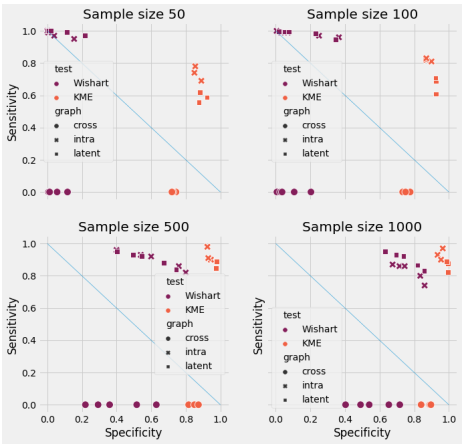
The linear case in figure 5.4a shows that for low sample sizes the Wishart test is superior in the three t-separation case, while the KME is superior in the 1 t-separation case. When the sample size increases, both methods get comparable results.

For the nonlinear case in figure 5.4b we see about the same results for the three t-separation graph, except for the Wishart test performing worse on lower sample sizes. The KME test has a good performance on the 1 t-separation graph with low sample size, and a nearly perfect performance with higher sample size. It seems that the Wishart test improves to a certain point, at which it trades in specificity for sensitivity when changing the significance level.

5.4.2 The three impure graphs



(a) Graph comparing the Wishart vs the KME test on data generated with linear path coefficients.



(b) Graph comparing the Wishart vs the KME test on data generated with non-linear path coefficients.

Figure 5.5: The sensitivity and specificity of the Wishart tested versus the KME on data sampled the three impure graphs. Each point represents the average of 100 runs with either a different distribution size for the KME or a different significance level for the Wishart test.

Results of the KME and Wishart test on the three graphs each containing a different impurity can be seen in figure 5.5. For the KME training data sampled from the two pure graph examples and three impure graph examples was used. In both the linear and non-linear case the KME has a clear advantage on the data sampled from the cross construct impure graph example. In the linear case, both methods have different advantages on the intra measure and latent construct impure graph examples at low sample sizes and become similar as the

sample size increases. In the non-linear case the KME has a clear advantage on every graph example at larger sample sizes.

5.4.3 The impure measurement model

Up until this point, the KME has been tested on data that has been sampled from the same graphs as from which the training data has been sampled. This is a simplification since normally we could not know the exact causal structures that underlies the test data. The real challenge lies in generalizing the constraints seen in simple graph examples to bigger and more realistic graphs. In figure 5.6 an experiment on data sampled from a larger impure measurement model, as seen in figure 4.6, with non-linear coefficients can be seen. For this the KME was trained on data sampled from the pure and impure graph examples and each point represent the average of 10 runs.

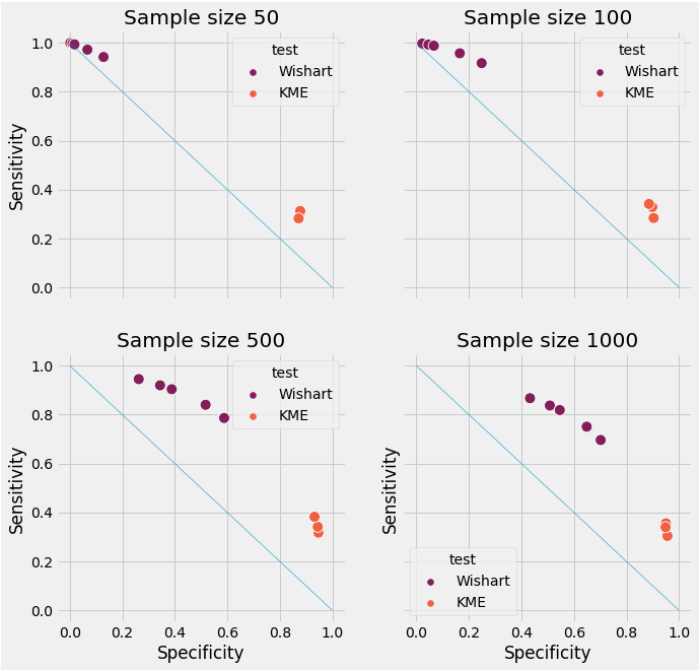


Figure 5.6: The sensitivity and specificity of the KME and Wishart test on an impure measurement model with nonlinearly generated data. Each point represents the average of 10 runs with either a different distribution size for the KME or a different significance level for the Wishart test.

Both methods seem to lie on the same frontier of the sensitivity and specifity tradeoff. In this model there were 7987 instances of no t-separation and 6547

instances of t-separation. The best accuracy of the classifier was on 1000 samples per variable and 1000 training distributions per graph example: 68.25% with a standard deviation of 2%. The best accuracy of the Wishart test was with a 1000 samples per variable and a significance level of 0.1: 69.92% with a standard deviation of 6%.

The methods lie very close, although the classifier is more stable in its results.

5.4.4 The pure measurement model

To give the results on the impure measurement model a bit more context the results on the pure measurement model (from figure 4.7) displayed in figure 5.7 will also be discussed. The tests were done on linearly generated data, which makes this a problem on which the Wishart test should get its best results. The KME was only trained on the two pure graph examples. Each point represents the average of 10 runs. Similar to the impure measurement model, it seems that the KME is stuck at a certain threshold. The Wishart test however improves to an almost perfect score with a high sample size.

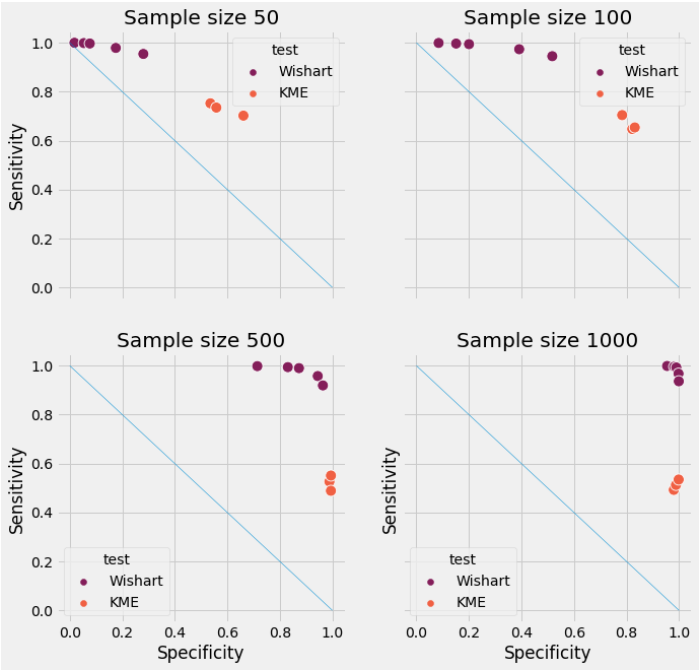


Figure 5.7: The sensitivity and specificity of the KME and Wishart test on an pure measurement model with linearly generated data. Each point represents the average of 10 runs with either a different distribution size for the KME or a different significance level for the Wishart test.

Chapter 6

Discussion

6.1 Conclusions from the experiments

The experiments have shown that the Wishart test is fairly robust when tested on non-linearly generated data, although its results do degrade. The KME based classifier is able to get similar results to the Wishart test when the test and training data share the same underlying graphical structure and data generating process. In this case, given two pairs of variables, the classifier is often able to correctly decide if the pairs are t-separated or not. The results also remain fairly stable when testing the classifier on training examples sampled from graphs containing an impurity. This is an advantage over the Wishart test, that has a worse performance on impure graphs and especially those with non-linear functions between variables. The results mean that in theory the KME should be a satisfactory feature for discerning the cases we can encounter in an impure measurement model, assuming that the faithfulness and Markov assumptions necessary for causal discovery hold and the variables are connected by a limited set of non-linear functions.

However, when we sample the training data from a different graph as the test data, the results clearly degrade. So there is a gap in what the classifier learns from the data sampled from basic graph examples and what the classifier understands about the data sampled from more complex graphs. This gap can also be observed in the results of the pure measurement model with linearly generated data. It is likely that this gap exists because the KME does not generalize well to graphs in which different kinds of distributions can occur. The difficulty with larger graphs is that the distributions that are created by larger hierarchies are non existent in the simple examples. This would also explain why increasing the amount of samples per variable and the amount of distributions per training graph barely increase the accuracy. It is not the

case that the classifier should train on more of the same distributions that already occur in the training data, but it should learn on different distributions altogether. The creation of unseen distributions by large hierarchies might also be amplified by non-linear functions, since these allow for the creation of wider variety of distributions.

This problem could be prevented by sampling the latent variables from more complex distributions, so that they could model any kind of structure that causes them. Some options for and problems with doing this are discussed in the next section.

6.2 Generating training data that is similar to the test data

Besides what was mentioned in the previous chapter, another problem of using the kernel mean embedding as a feature is that it relies on a large bag of sample sets to train the classifier. It could be the case that the amount of labeled training data is sparse or that there is none at all. In these cases it would be required to first generate training data with similar distributions as the test data before the classifier can be trained. As mentioned earlier, if the functions that generate the data are to general, they can model any distribution and it would not be possible to discern between the different underlying causal graphs.

In [5] data samples $S_i = \{(x_{ij}, y_{ij})_{j=1}^{n}\}$ are generated. The values of x are decided by sampling from a Gaussian mixture model. To calculate y, a spline $f(x)$ is fitted on the values of x as a mapping mechanism. Then, random Gaussian noise n is added to the results of the mapping function $y = f(x) + n$. Finally, each of these samples is labelled with the causal direction between the variables. The data generating process contains many hyperparameters such as how many elements to include in the spline, the amount of variables to pick from the mixture model and the variances of the sampling distributions. To decide which of these parameters to choose, a grid search is done where the paramters are found that generate data closest in distance to the test data. To measure the distance, the authors use the following formula:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i} \min_{i \leq j \leq N} \left\| v(S_i) - v(\tilde{S}_j) \right\|_2^2 \tag{6.1}$$

Here θ represents the hyperparameter settings, $v(S_i)$ represents the kernel mean embedding of a test samples set and $v(\tilde{S}_j)$ the KME of a generated sample set. The function finds the generated sample sets with the overall closest distance to the test data sets.

Applying this approach to the latent variable case, it is likely that the Gaussian mixture is able to model the possible complex hierarchy that lies

behind the latent variable. A problem with this function lies in the size of the search space when not much is known of the process that generated the test data. When increasing the amount of variables whose values need to be calculated with this process, the amount of combinations we need to try grows exponentially.

Similar work has been done to find conditional independences with a mimic and classify approach [34]. In short this entails that the test data is split into two, of which one part is used as a target for a mimicking algorithm. The mimicked data is then added together with the second part and a classifier is trained to separate these two sets. If it fails, then we know that the model that the mimicking algorithm used is the correct one. This framework might have interesting aspects that can be used to classify tetrad constraints. There are however some additional challenges when applying this to tetrad constraints, such as which models to choose as representative for the constraints.

6.3 Kernel methods versus statistical tests

Besides the difference in results of the Wishart test and Kernel mean embedding, it is also interesting to look at the difference in how they can be applied. The process of using a classifier as test is different from using a statistical test such as the Wishart test. For a statistical test it is necessary to have an idea about the distribution of a signal to know how likely it is that we should see it attain a certain value. Using a classifier with the KME as feature requires that the sampled training data is close enough to the test data and that what we are testing for is discernable from a distribution. This opens up the door for a more experimental approach to finding constraints in de data, but it does always leave the chance that better parameter optimization would have lead to the envisioned results.

The Wishart test is used both confirmatory [35] as well as exploratory [17]. Exploratory analysis entails that the observed data is tested for tetrad constraints, after which the found constraints are used to infer a class of graphs that adhere to them. It is yet to be seen if the KME can work like this since the experiments have shown that if the true graph differs from the training examples, the results degrade steeply. Confirmatory is when one has a certain graph structure in mind and compares that vanishing tetrad constraints of that graph with those in the observed data to see if it is a good fit. This could in theory work since you can train the KME based classifer on training data from the hypothetical graph.

6.4 Future research

1. The most important next step is to figure out why the score of a KME based classifier degrades on bigger graphs. A direction of further research would be to see if adding a few more graph examples as training data generators improves the results. Alternatively, one could search for the training data that lies as close as possible to the test data. For this the mimic and classify approach discussed in the previous section is an interesting line of research to find methods that allow us to mimic the test data.

2. In this thesis experimentation has been done with multiple KME embeddings. There is however a lot more possible such as adding and multiplying vectors. Would it for example work to embed the distribution of four variables separately and add them together to get the same effect as embedding the four variables together? If so, this would save a lot of computational steps.

3. Although the random forest classifier is a powerful method, it might be interesting to train a neural network on the KME. When done properly, this might be able to capture more complex patterns that emerge between the KME's of distributions.

4. A thing that is hard to decide is the ratio of noise versus incoming values. In my setup the values of a variable's parents are first normalized before the noise is added, making it an influential part of the variable's value. It could be interesting to investigate how different ratios would change the results.

5. Another problem with doing research into measurement models is that there is a lack of datasets where the causal structure is known. To further the field of causal discovery more datasets of which the causal graph is known need to be published so methods can be compared on clear benchmarks.

6. Lastly, the problem of doing causal discovery without the assumption of causal sufficiency and with the presence of non-linear continuous functions between variables is barely discussed in the literature. Work has been done on finding a confounder between two variables in this case, but methods of finding more complex latent structures are never mentioned in overview papers or books about causal discovery [3] [7]. It is however my belief that focussing on signals that might exist in the distributions of multiple observed variables can yield results that can not be obtained by the two variable case.

Chapter 7

Conclusion

In this thesis it has been attempted to solve two intertwined problems. On the one hand we tried the extend the idea of tetrad constraints to multiple indicator models with non-linear functions. On the other hand, we tried to apply the ideas of the randomized causation coefficient on latent variable discovery. To do this we trained a binary classifier on the KME of distributions to judge the existence of a t-separation constraint between two pairs of variables.

Experiments have shown that in the case that test and training data share the underlying graphical structures and data generating process the classifier is able to discern between cases of t-separation just as good or better as the Wishart test. This sparks hope that the KME of distributions contains enough information to recognize t-separation constraints in the underlying graph and can even go beyond by recognizing constraints in non-linearly generated data. Unfortunately, the results degrade when using the classifier on more realistic graphs with an increased amount of variables and larger hierarchies. The most likely cause of this is that the larger graphs contain distributions that do not occur in the more simple graphs used for sampling training data. Further research is needed to see how the training data can capture a wider variety of distributions without the search space growing too big.

Bibliography

[1] Lex Fridman. Judea Pearl: Causal Reasoning, Counterfactuals, Bayesian Networks, and the Path to AGI — AI Podcast.

[2] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

[3] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT press, 2017.

[4] Bernhard Schölkopf. Causality for Machine Learning. *arXiv:1911.10500 [cs, stat]*, December 2019.

[5] David Lopez-Paz, Krikamol Muandet, Bernhard Scholkopf, Ilya Tolstikhin, and Lopezpaz Org. Towards a Learning Theory of Cause-Effect Inference. *International Conference on Machine Learning. 2015.*

[6] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building Machines That Learn and Think Like People. *Behavioral and brain sciences 40 (2017).*, November 2016.

[7] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[8] Judea Pearl, Madelyn Glymour, and Nicholas P. Jewell. *Causal Inference in Statistics: A Primer*. John Wiley & Sons, March 2016.

[9] Thomas Richardson and Peter Spirtes. Ancestral graph Markov models. *Annals of Statistics*, 30(4):962–1030, August 2002.

[10] Benjamin Frot, Preetam Nandy, and Marloes H. Maathuis. Robust causal structure learning with some hidden variables. *arXiv:1708.01151 [stat]*, August 2018.

[11] Peter Spirtes and Clark Glymour. *Causation, Prediction, and Search*. MIT press, 1993.

[12] Ricardo Silva, Richard Scheines, Clark Glymour, and Peter Spirtes. Learning the Structure of Linear Latent Variable Models. *Journal of Machine Learning Research 7.Feb (2006): 191-246.*, page 56.

[13] Charles Spearman. Pearson's contribution to the theory of two factors. *British Journal of Psychology*, 19(1):95, 1928.

[14] Kenneth A Bollen and Kwok-fai Ting. A Tetrad Test for Causal Indicators. *Psychological methods 5.1 (2000): 3.*, page 20.

[15] John Wishart. Sampling Errors in the Theory of Two Factors. *British Journal of Psychology. General Section*, 19(2):180–187, 1928.

[16] Ricardo Silva, Richard Scheines, Clark Glymour, and Peter L. Spirtes. Learning Measurement Models for Unobserved Variables. *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence. 2002.*, October 2012.

[17] Erich Kummerfeld and Joseph Ramsey. Causal Clustering for 1-Factor Measurement Models. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1655–1664, San Francisco California USA, August 2016. ACM.

[18] Glenn Shafer. Generalization of the Tetrad Representation Theorem. *Rutgers Center for Operations Research [RUTCOR].*, 1993.

[19] Seth Sullivant, Kelli Talaska, and Jan Draisma. Trek separation for gaussian graphical models. *Annals of Statistics*, 38(3):1665–1685, 2010.

[20] Seth Sullivant. Algebraic geometry of Gaussian Bayesian networks. *Advances in Applied Mathematics*, 40(4):482–513, 2008.

[21] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schoelkopf. Kernel-based Conditional Independence Test and Application in Causal Discovery. *arXiv:1202.3775 [cs, stat]*, February 2012.

[22] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. *Kernel Mean Embedding of Distributions: A Review and Beyond*, volume 10. 2017.

[23] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 2002.

[24] Hal Daumé III. From Zero to Reproducing Kernel Hilbert Spaces in Twelve Pages or Less. Technical report, 2004.

[25] Joris M. Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing Cause from Effect Using Observational Data: Methods and Benchmarks. *Journal of Machine Learning Research*, 17(32):1–102, 2016.

[26] Eleni Sgouritsa, Dominik Janzing, Jonas Peters, and Bernhard Schoelkopf. Identifying Finite Mixtures of Nonparametric Product Distributions and Causal Inference of Confounders. *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, September 2013.

[27] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *Max Planck Institute for Intelligent Systems*, October 2018.

[28] Guido Van Rossum and Fred L. Drake Jr. *Python Tutorial*, volume 620. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[29] Wes McKinney. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010.

[30] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[31] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[32] Thomas A Caswell, Michael Droettboom, John Hunter, Eric Firing, Antony Lee, Jody Klymak, David Stansby, Elliott Sales De Andrade, Jens Hedegaard Nielsen, Nelle Varoquaux, Benjamin Root, Tim Hoffmann, Phil Elson, Ryan May, Darren Dale, Jae-Joon Lee, Jouni K. Seppänen, Damon McDougall, Andrew Straw, Paul Hobson, Christoph Gohlke, Tony S Yu, Eric Ma, Adrien F. Vincent, Steven Silvester, Charlie Moad, Jan Katins, Nikita Kniazev, Federico Ariza, and Elan Ernest. Matplotlib/matplotlib v3.1.0. Zenodo, May 2019.

[33] Michael Waskom, Olga Botvinnik, Maoz Gelbart, Joel Ostblom, Paul Hobson, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, Jordi Warmenhoven, John B. Cole, Julian De Ruiter, Jake Vanderplas, Stephan Hoyer, Cameron Pye, Alistair Miles, Corban Swain, Kyle Meyer, Marcel Martin, Pete Bachant, Eric Quintero, Gero Kunter, Santi Villalba, Brian, Clark Fitzgerald, C.G. Evans, Mike Lee Williams, Drew O'Kane, Tal Yarkoni, and Thomas Brunner. Mwaskom/seaborn: V0.11.0 (Sepetmber 2020). Zenodo, September 2020.

[34] Rajat Sen, Karthikeyan Shanmugam, Himanshu Asnani, Arman Rahimzamani, and Sreeram Kannan. Mimic and Classify : A meta-algorithm for Conditional Independence Testing. *arXiv:1806.09708 [cs, stat]*, June 2018.

[35] Kwok-fai Ting. Confirmatory Tetrad Analysis. *Sociological methodology*, 23:147–175, 1993.