Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis Econometrics & Management Science

The Discrete-Choice Copula Bayesian Network Model

Author

Yorn Sven Hoogendoorn

ID: 385665

Supervisor

Anna A. Kiriliouk

Second assessor

Dennis Fok

August the 20th, 2018

Abstract

Bayesian network classifiers can be combined with copulas to accurately model continuous random variables. However, applications of such frameworks often restrict the graph structure or the type of variables that may be included. We develop the predictive copula Bayesian network (PCBN), a probabilistic classification framework which allows one to model continuous, ordinal discrete and categorical variables explicitly, while allowing variables to be correlated via copulas. We also introduce an algorithm which constructs a PCBN based on (predictive) scores, without assuming tree-like structures. We apply the framework to both simulated and empirical travel mode data and show that PCBNs can accurately model the data, especially when the data is highly correlated.

Contents

1 Introduction

Statistical classification is a task that arises in many fields of research. In statistical classification, we are interested in creating a classifier function that can assign some category based on a set of observations. If we have labelled observations, then we may use *supervised learning*. Otherwise, we are confined to *unsupervised learning*. Unsupervised classification is also known as *clustering* in machine learning. On the other hand, supervised classification is often simply known as *classification*.

Classifiers can be created in many ways. One way of creating a classifier is by making use of *graphical models*, a type of probabilistic model for which a graph encodes the conditional dependence structures between random variables. Graphical models have the advantage of having a graphical representation that can be used to illustrate the problem.

Bayesian networks (BNs) are a type of graphical model in which the underlying graph is restricted to be a directed acyclic graph (DAG). A BN over a set of variables allows the joint probability of these variables to be decomposed into local terms. More specifically, the joint probability is a product of local conditional distributions. BNs have applications in various fields such as ecology, medicine and finance (Pourret, Naïm, & Marcot, 2008). A classifier that uses a BN is called a *Bayesian network classifier* or BN classifier.

Although BNs are flexible, most applications using continuous data are limited to assuming underlying Gaussian distributions. Copulas however, allow one to model multivariate continuous distributions more easily. The use of copulas is rooted in Sklar's Theorem, which states that any multivariate cumulative distribution function (CDF) can be written as a function of its marginals (Sklar, 1959). This function is called the copula, see Nelsen (2007) for a comprehensive introduction to copulas.

Using copulas in BNs results in a framework that can outperform competing models in both unsupervised (Elidan, 2010) and supervised (Elidan, 2012) learning applications. However, the applicability and predictive performance in classification problems can be improved. For instance, the framework created by Elidan (2012) does not allow ordinal discrete nodes to be modelled explicitly. Moreover, continuous nodes are not allowed to

have discrete children.

In this thesis, we aim to improve the predictive performance of copula-based BNs. We develop a framework which allows one to use continuous, ordinal discrete and categorical random variables and we create an algorithm that learns its structure automatically from the data.

This thesis is organised as follows. In Section 2, we elaborate on relevant literature regarding copulas and Bayesian networks. Then, we formally introduce the classification problem in Section 3 and we elaborate on the goals of this thesis. Next, we introduce the existing frameworks that will be used in our framework. Section 5 introduces the predictive copula Bayesian networks (PCBN), as well as the algorithms that can be used to construct these networks. Section 6 describes the simulation design that we use to assess the framework and Section 7 elaborates on the results. Then, we apply the framework to an empirical case in Section 8. Lastly, we briefly summarise the results and give a conclusion in Section 9.

The simulation results indicate that we can increase predictive accuracy by correctly modelling ordinal variables. The empirical application shows that PCBNs can greatly outperform competing models in terms of predictive accuracy.

2 Literature

Bayesian networks (BNs) have been used in various academic fields such as ecology, medicine and finance (Pourret et al., 2008). One of the first uses of a BN classifier appeared in Duda and Hart (1973). Now, this classifier is known as the Naïve Bayes (NB) classifier due to its strong or naïve independence assumptions. It assumes that the class variable is influenced by explanatory variables, whereas the explanatory variables do not influence each other. Although this is often unrealistic, the NB classifier performs surprisingly well (N. Friedman, Geiger, & Goldszmidt, 1997). J. H. Friedman (1997) shows that, even though the bias of the NB classifier may be the high, its low variance often results in good performance.

Although naïve Bayes performs well, the independence assumptions are often too strong. An extension of NB that relaxes this independence assumption is the so-called tree-augmented naïve Bayes (TAN) classifier, originally proposed by N. Friedman et al. (1997). TAN classifiers contain the NB structure, but allow some dependence between the explanatory variables. N. Friedman et al. also propose constructing a Bayesian network for each value attained by the dependent variable. They call this a *Bayesian multinet*.

In BNs, continuous explanatory variables are often assumed to come from an underlying Gaussian distribution. John and Langley (1995) propose non-parametric kernel density estimation for naïve Bayes as an alternative to assuming Gaussian distributions. The idea of John and Langley can be applied to TAN-like classifiers as well, but it would require the estimation of multivariate densities, which is not an easy task in general.

However, copulas offer a solution. A copula is a multivariate cumulative distribution function (CDF) that has uniform marginals. Copulas are unique on their domain when all marginals are continuous. Copulas have been incorporated in various models, e.g. logistic regressions applied to dental data (Nikoloulopoulos & Karlis, 2008), binary logits with spatial dependency (Bhat & Sener, 2009) and multivariate GARCH models (Jondeau & Rockinger, 2006). Copulas even find applications in cosmology (Scherrer, Berlind, Mao, & McBride, 2009).

There exists a unique copula for a multivariate distribution when its marginal distributions are continuous. When some marginals are discrete, the existence of a unique copula is not guaranteed. Besides this, concordance measures are no longer independent of the margins and estimates of Kendall's τ, a popular non-parametric dependence measure, are biased (Genest & Nešlehová, 2007). There are several solutions to this problem. Nešlehová (2007) proposes a transformation method so that count data is made continuous. This guarantees the existence of a unique copula.

Copulas have been used in BNs. In unsupervised learning, Elidan (2010) uses copula constructions for continuous data. Bauer and Czado (2016) use elaborate constructions of pair-copulas to model continuous BNs. In supervised learning, Elidan (2012) uses conditional copulas. His framework does not allow continuous nodes to have discrete children, however.

3 Problem and Goals

This section introduces the main problem mathematically and concludes with the goals that this thesis aims to reach. The main problem of this thesis is to increase the applicability and performance of copula-based Bayesian networks. In order to introduce this problem mathematically, we start with the general classification problem and follow with the out-sample performance problem. We end with the goals of this thesis.

3.1 Problem Definition

Let $X_1, \ldots, X_n$ be n random variables with realisations $x_1, \ldots, x_n$ respectively and let Y be a categorical random variable with sample space $\mathcal{K} \equiv \text{Ran}(Y)$ with $|\mathcal{K}| = k \geq 2$. Now, let $h : \mathcal{X} \to \mathcal{K}$ be a classifier function on the domain $\mathcal{X} = \text{Dom}(X_1, \ldots, X_n)$. That is, we try to create a function that can *predict* the value of the dependent categorical variable Y based on the n explanatory variables $X_1, \ldots, X_n$. We have a binary classification problem for $k = 2$ and a multinomial classification problem for $k > 2$.

In *probabilistic classification*, the classification rule is based on the calculation of k conditional probabilities $\text{P}[Y = y | X_1 = x_1, \ldots, X_n = x_n]$. If the decision rule is to choose the value y that maximises this expression, then we can construct a probabilistic classifier h^{prob} as

$$h^{\text{prob}}(x_1, \ldots, x_n) \equiv \hat{y} \in \arg\max_{y \in \mathcal{K}} \text{P}[Y = y | X_1 = x_1, \ldots, X_n = x_n]. \tag{1}$$

Note that $h^{\text{prob}} \neq \varnothing$ as there is at least one $y \in \mathcal{K}$ that maximises the conditional probability. Equation (1) forms the basis of the main classification methods in this paper.

3.1.1 Out-sample Performance Problem

Say we have some data set $\mathcal{D} = \left\{ \left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M} \right\}$ where $\mathcal{M} = \{1, \ldots, d\}$ is a set of $d \in \mathbb{N}^+$ unique indices. Each realisation $y^{(l)} \in \mathcal{K}$ corresponds to the categorical random variable Y and the n realisations $x_1^{(l)}, \ldots, x_n^{(l)}$ correspond to $X_1, \ldots, X_n$ respectively for $l \in \mathcal{M}$. Now, let $\mathcal{D}^{\text{in}} = \left\{ \left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M}^{\text{in}} \right\}$ be the in-sample set with $\left| \mathcal{M}^{\text{in}} \right| = d^{\text{in}} \in \mathbb{N}^+$ indices. Similarly, let $\mathcal{D}^{\text{out}} = \left\{ \left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M}^{\text{out}} \right\}$ be the in-sample set with $|\mathcal{M}^{\text{out}}| = d^{\text{out}} \in \mathbb{N}^+$ indices. Furthermore, let $\mathcal{M}^{\text{in}}, \mathcal{M}^{\text{out}} \subset \mathcal{M}$ with $\mathcal{M}^{\text{in}} \cap \mathcal{M}^{\text{out}} = \emptyset$.

Now, we define some loss function $L(\hat{z}, z)$ that measures how well we assign categories with estimate $\hat{z}$ and actual value z. For instance, we can define 0-1 loss using Iverson brackets as $L_{0\text{-}1}(\hat{z}, z) \equiv [\hat{z} \neq z]$. This evaluates to 0 if the estimate is equal to actual value and evaluates to 1 if it is not. If we denote $\mathcal{H}$ as the set of possible classifiers under consideration, then we can determine an optimal classifier $h^{\text{in}*}$ as

$$h^{\text{in}*} \in \arg\min_{h \in \mathcal{H}} \frac{1}{d^{\text{in}}} \sum_{l \in \mathcal{M}^{\text{in}}} L\left(h\left(x_1^{(l)}, \ldots, x_n^{(l)}\right), y^{(l)}\right). \tag{2}$$

Note that the division by d^{in} does not influence the optimal classifier $h^{\text{in}*}$. We assume that it is possible to find a best in-sample classifier. Thus, we assume $h^{\text{in}*} \neq \varnothing$. Determining $h^{\text{in}*}$ via (2) yields a classifier, out of a set of possible classifiers, that minimises a chosen loss function for the in-sample set. However, this classifier may perform poorly in our out-sample set $\mathcal{D}^{\text{out}}$. Concretely, this means that there might be some classifier $h^{\text{in}} \neq h^{\text{in}*}$ with both $h^{\text{in}}, h^{\text{in}*} \in \mathcal{H}$ for which we have

$$\sum_{l \in \mathcal{M}^{\text{out}}} L\left(h^{\text{in}}\left(x_1^{(l)}, \ldots, x_n^{(l)}\right), y^{(l)}\right) < \sum_{l \in \mathcal{M}^{\text{out}}} L\left(h^{\text{in}*}\left(x_1^{(l)}, \ldots, x_n^{(l)}\right), y^{(l)}\right).$$

Let $h^{\text{out}*}$ denote the best out-sample classifier under the same loss function L where we assume that $h^{\text{out}*} \neq \varnothing$. Then, $h^{\text{out}*}$ is given by (3).

$$h^{\text{out}*} \in \arg\min_{h \in \mathcal{H}} \frac{1}{d^{\text{out}}} \sum_{l \in \mathcal{M}^{\text{out}}} L\left(h\left(x_1^{(l)}, \ldots, x_n^{(l)}\right), y^{(l)}\right) \tag{3}$$

Now, the question that arises is: *How can we find a classifier h^{out} that has a good out-sample prediction if we can only use $\mathcal{D}^{\text{in}}$?* Although it could be possible that $h^{\text{in}*} = h^{\text{out}*}$, it is also possible that $h^{\text{in}*}$ has a very poor out-sample performance. This might be the case when $h^{\text{in}*}$ has overfit the data.

3.1.2 Main Problem of This Thesis

The optimal classifiers $h^{\text{in}*}$ and $h^{\text{out}*}$ from equations (2) and (3) respectively are defined for some set of classifiers $\mathcal{H}$. So far, this set has not been specified. In this research, we restrict our search to copula-based Bayesian network classifiers. The main question that guides this research is:

How can we increase the applicability and predictive performance of copula-based Bayesian network classifiers?

3.2 Goals

The main goal of this thesis is to increase the applicability and predictive performance of copula-based Bayesian networks to find a $h^{\text{in}*}$ which is close to $h^{\text{out}*}$. Elidan (2012) uses copulas in Bayesian networks classifiers and illustrates that this can greatly improve the performance of Bayesian network classifiers. However, his approach has some restrictions. Firstly, count data or ordinal discrete data cannot be modelled effectively, because the copulas in his framework assume that the explanatory variables are continuous. Secondly, continuous nodes cannot have discrete children, which might hinder the predictive performance of the model.

We develop a framework that models ordinal discrete variables *explicitly* in order to boost the predictive performance of copula-based Bayesian networks in these settings. We also develop a greedy algorithm that uses predictive scores on a validation set to learn a network. We call this algorithm the greedy predictive copula Bayesian network (PCBN). Due to the greedy nature of the algorithm, it is intuitive to create several PCBNs. We develop a procedure, which we call the copula spider, that creates several PCBNs. This thesis aims to reach the following goals:

1. Develop a framework that allows continuous, ordinal discrete and categorical explanatory variables to modelled explicitly, while simultaneously allowing continuous nodes to have ordinal discrete or categorical children

2. Develop an algorithm that constructs networks according to this framework. The networks, developed with this algorithm, should have good out-sample predictive performances that are comparable to, if not better than, competing models

4 Existing Frameworks

This section introduces two key frameworks on which my framework will be built. We start by introducing our notational convention. Then, we introduce Bayesian Networks (BNs) and explain how they can be used in supervised classification settings. Lastly, we elaborate on the copula framework and explain the most important copumetric results and concepts, as they are an integral part of the predictive copula Bayesian networks.

4.1 Notational Convention

Throughout this thesis, we maintain the following conventions:

1. Random variables are denoted in upper case, e.g. $X_1, \ldots, X_n$. Their corresponding realisations are denoted in lower case, e.g. $x_1, \ldots, x_n$.

2. Vectors are denoted in boldface. Upper-case bold letters are used for vectors of random variables, e.g. $\mathbf{X} = (X_1, \ldots, X_n)$. Lower case letters in boldface are used for vectors with realisations, e.g. $\mathbf{x} = (x_1, \ldots, x_n)$.

3. Tuples of length n with elements of the same type are denoted in lower-case letters in bold, but n-tuples with elements of different types are denoted in calligraphic upper-case letters. This applies to the graphs, e.g. graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$.

4. Sets are denoted with calligraphic letters such as $\mathcal{N} = \{1, \ldots, n\}$, except for the set of all non-negative integers $\mathbb{N}$, set of all positive integers $\mathbb{N}^+$, set of all real numbers $\mathbb{R}$ and extended set of all real numbers $\overline{\mathbb{R}}$.

5. Superscript between parentheses denotes the instance. For example, d realisations of X_i are denoted as $x_i^{(1)}, \ldots, x_i^{(d)}$.

6. General cumulative distribution functions (CDFs) are denoted with F where the subscript denotes what the CDF is defined over. E.g., a joint CDF over $\mathbf{X}$ is denoted as $F_{\mathbf{X}}$. A probability function is denoted with f, e.g. $f_{\mathbf{X}}$. Unless otherwise specified, CDFs and probability functions contain continuous, ordinal discrete and categorical random variables.

A list of symbols is included in the appendix, see Appendix C.

4.2 Bayesian Networks

A Bayesian network (BN) is a graphical model that is used to model a joint probability distribution over random variables by making use of independence properties. See Lauritzen (1996) for an elaborate introduction to this topic. One of the major defining features of a BN is its underlying graph structure. The other properties follow naturally from this graph.

4.2.1 Mathematical Notion of Bayesian Networks

The backbone of a Bayesian network is its directed acyclic graph (DAG). The definition of a DAG is given by Definition 4.2.1. Relevant graph and set theory, such as the definition of a path, has been included in Appendix A.1.

Definition 4.2.1 (Directed acyclic graph (DAG))**.** *A graph is an ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ of a set of nodes $\mathcal{N} = \{1, \ldots, n\}$ containing n elements and a set of arcs $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$. This graph is a directed acyclic graph (DAG) if the following conditions hold:*

1. *All arcs are directed. That is, $(i, j) \in \mathcal{A} \implies (j, i) \notin \mathcal{A}$ for all $i, j \in \mathcal{N}$ such that $i \neq j$.*
2. *The graph does not contain any cycles. That is, for every path from node i to node j, denoted as $i \to \ldots \to j$, we have that $i \neq j$. This means that there are also no self-cycles. Thus, $(i, i) \notin \mathcal{A}$ for all $i \in \mathcal{N}$.*

As a result of 1 and 2, we have $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$.

In a Bayesian network, the nodes represent random variables and directed arcs encode how each node is conditionally *independent* from all other nodes. The formal definition of a Bayesian network is given in Definition 4.2.2.

Definition 4.2.2 (Bayesian network (BN))**.** *Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a DAG as defined in Definition 4.2.1 with a set of nodes $\mathcal{N} = \{1, \ldots, n\}$ and set of directed arcs $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$. Let there be n random variables $X_1, \ldots, X_n$.*

Then, a Bayesian network (BN) is a tuple $\mathcal{B} = (\mathcal{G}, \Theta)$ in which the following is satisfied for every node $i \in \mathcal{N}$:

$$f_{X_i|\mathbf{Nd}_{X_i}}(x_i|\mathbf{nd}_{X_i}) = f_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}_{X_i}\right) \tag{4}$$

where $\mathbf{Nd}_{X_i}$ *is the vector of random variables that are non-descendants of* X_i *with corresponding realisation vector* $\mathbf{nd}_{X_i}$, $\mathbf{Pa}_{X_i}$ *is the vector of random variables that are parents of* X_i *with corresponding realisation vector* $\mathbf{pa}_{X_i}$, f *is a conditional probability function that might be a conditional pdf, pmf or categorical distribution function depending on* X_i *and* Θ *is the set of conditional probability density functions.*

Note that for every $i \in \mathcal{N}$ *we have* $\mathcal{N}_{\mathrm{pa}(i)} \subseteq \mathcal{N}_{\mathrm{nd}(i)}$. *That is, the set of nodes that are parents of* i *is always a subset of, or equal to, the set of non-descendants of* i. *Equation* (4) *essentially says that every random variable* X_i *is* independent *of its non-descendants* given *the realisations of the parents of* X_i. *If* X_i *is categorical then we may write* $\mathrm{P}\left[X_i = x_i | \mathbf{Nd}_{X_i} = \mathbf{nd}_{X_i}\right] = \mathrm{P}\left[X_i = x_i | \mathbf{Pa}_{X_i} = \mathbf{pa}_{X_i}\right]$.

The joint probability can be decomposed into local terms as given in Corrolary 4.2.2.1.

Corollary 4.2.2.1 (Factorisation of the joint probability function)**.** *Let* $\mathcal{B} = (\mathcal{G}, \Theta)$ *be a BN as given in Definition 4.2.2. Then, the joint probability function can be factorised as*

$$f_{\mathbf{X}}(\mathbf{x}) = \prod_{i \in \mathcal{N}} f_{X_i | \mathbf{Pa}_{X_i}}\left(x_i | \mathbf{pa}_{X_i}\right) \tag{5}$$

where $\mathbf{Pa}_{X_i}$ *is the vector of random variables that are parents of* X_i *with corresponding realisation vector* $\mathbf{pa}_{X_i}$.

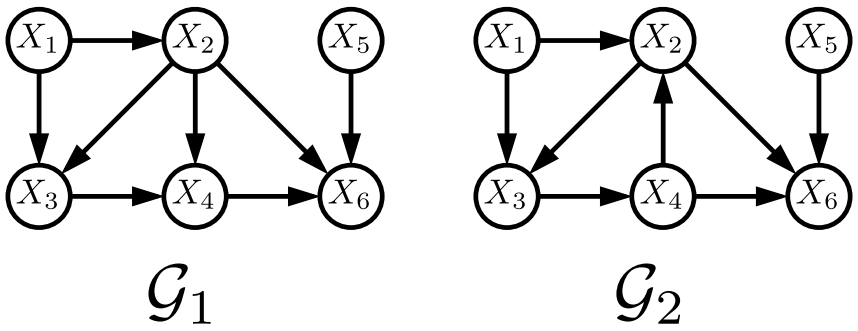

Figure 1: Example of two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. $\mathcal{G}_1$ is a directed acyclic graph (DAG) as given by Definition 4.2.1, but $\mathcal{G}_2$ is not due to the existence of the cycle $X_2 \to X_3 \to X_4 \to X_2$.

An example of a BN is given by $\mathcal{G}_1$ of Figure 4.2.1. Using (5), the joint probability of

$X_1, \ldots, X_6$ is given by:

$$f_{X_1,\ldots,X_6}(x_1,\ldots,x_6) = f_{X_1}(x_1) f_{X_2|X_1}(x_2|x_1) f_{X_3|X_1,X_2}(x_3|x_1,x_2) f_{X_4|X_2,X_3}(x_4|x_2,x_3) \\ \times f_{X_6|X_2,X_4}(x_6|x_2,x_4) f_{X_5}(x_5) f_{X_6|X_2,X_4,X_5}(x_6|x_2,x_4,x_5).$$

It is important to realise that the BNs do not encode *complete dependence* statements. To see this, consider $f_{X_2,X_5}(x_2,x_5)$ of $\mathcal{G}_1$ of Figure 4.2.1. We have that $f_{X_1,X_5}(x_1,x_5) = f_{X_1}(x_1) f_{X_5}(x_5)$ as nodes are independent of their non-descendants given their parents. However, we do *not* necessarily have $f_{X_1,X_5|X_6}(x_1,x_5|x_6) = f_{X_1|X_6}(x_1|x_6) f_{X_5|X_6}(x_5|x_6)$. That is, we cannot determine if that relation holds given the provided graph. Determining whether nodes are conditionally independent can be done by assessing if nodes are *d*-separated which can be inferred from the underlying graph, see Pearl (1988).

4.2.2 Bayesian Networks as Classifiers

If Y is a categorical random variable that we wish to predict with explanatory variables $X_1, \ldots, X_n$, then we can use Bayesian network classifier. A probabilistic classifier is given by (1). In the case of BN classifiers, we make use of the following:

$$f_{Y|X_1,\ldots,X_n}(y|x_1,\ldots,x_n) = \frac{f_{Y,X_1,\ldots,X_n}(y,x_1,\ldots,x_n)}{f_{X_1,\ldots,X_n}(x_1,\ldots,x_n)} \propto f_{Y,X_1,\ldots,X_n}(y,x_1,\ldots,x_n).$$

We denote $f_{Y|X_1,\ldots,X_n}(y|x_1,\ldots,x_n)$ as $f_{Y|\mathbf{X}}(y|\mathbf{x})$. Then, we model $f_{Y,\mathbf{X}}(y,\mathbf{x})$ with (5) which gives

$$f_{Y,\mathbf{X}}(y,\mathbf{x}) = f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) \prod_{i\in\mathcal{N}} f_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}_{X_i}\right) \tag{6}$$

where $y \in \mathcal{K}$, $\mathbf{Pa}_Y$ and $\mathbf{Pa}_{X_i}$ refer to the vector of random variables that are parents of Y and X_i respectively and $\mathbf{pa}_Y$ and $\mathbf{pa}_{X_i}$ are their corresponding realisation vectors.

If all random variables in (6) are categorical, then we could estimate the conditional probabilities by creating conditional probability tables. The conditional probabilities could be estimated by counting co-occurrences. Although this strategy could work for categorical explanatory variables, it does not work when some explanatory variables are continuous. Often, continuous variables are assumed to follow a Gaussian distribution to ease compu-

tations. Intuitively, if the distributions are not Gaussian in reality, then this assumption makes the model misspecified. Now, it is possible to incorporate non-parametric kernel density estimation in BNs, as shown by John and Langley (1995). However, this becomes difficult when we need to estimate multivariate distributions.

Section 4.3 introduces copulas, which help in modelling multivariate distributions. Incorporating copulas in BNs allows for richer BN classifiers. Examples of the naïve Bayes, tree-augmented naïve Bayes and generalised Bayesian network classifier are included in Appendix A.5.

4.3 Copula Framework

Estimating and modelling multivariate distributions is still a challenging task to this day. Fortunately, copulas can be used to help with this task. A copula is some multivariate cumulative distribution function (CDF) C that has uniform marginals. The use of copulas is rooted in Sklar's Theorem given by Sklar (1959), which states that any multivariate CDF $F_{\mathbf{X}}$ of $\mathbf{X} = (X_1, \ldots, X_n)$ can be written as copula function $C_{\mathbf{X}}(F_{X_1}(x_1), \ldots, F_{X_n}(x_n))$ that links the marginal CDFs $F_{X_1}, \ldots, F_{X_n}$. The copula function $C_{\mathbf{X}}$ captures the dependence structure of the multivariate CDF and $C_{\mathbf{X}}$ is unique if all marginal CDFs F_{X_i} are continuous.

With Sklar's Theorem, estimating multivariate distributions can be done by estimating the marginal CDFs separately from the copula function. If a copula is chosen in such a way that it can model complex dependency structures, then the burden is shifted to estimating the copula parameters rather than the multivariate CDF. Sklar's Theorem is given in Theorem 4.3.1.

Given the CDF $F_{\mathbf{X}}$ in (7) of Theorem 4.3.1, we can derive a joint probability distribution $f_{\mathbf{X}}$. The relation between $f_{\mathbf{X}}$ and the copula is given in Theorem 4.3.2. This relation is valid when some marginals are continuous and others are ordinal discrete. Corollary 4.3.2.1 gives the probability density function (pdf) when all marginals are continuous and Corollary 4.3.2.2 gives the probability mass function (pmf) when all marginals are discrete.

Theorem 4.3.1 (Sklar's Theorem (Sklar, 1959))**.** *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *denote a vector of* n *random variables with corresponding realisations* $x_i \in \overline{\mathbb{R}}$ *for* $i = 1, \ldots, n$*. Now, let* $F_{\mathbf{X}}(x_1, \ldots, x_n) = \mathrm{P}[X_1 \leq x_1, \ldots, X_n \leq x_n]$ *denote the multivariate cumulative distribution function of* $\mathbf{X}$ *with marginal cdfs* $F_{X_i}(x_i) = \mathrm{P}[X_i \leq x_i]$*. Then, there is an* n*-dimensional copula* $C_{\mathbf{X}}$ *such that*

$$F_{\mathbf{X}}(x_1, \ldots, x_n) = C_{\mathbf{X}}(F_{X_1}(x_1), \ldots, F_{X_n}(x_n)). \tag{7}$$

$C_{\mathbf{X}}$ *is unique on the cartesian product of the ranges of the marginal cdfs, i.e.* $\mathrm{Ran}(F_1) \times \ldots \times \mathrm{Ran}(F_n)$*. If all marginal cdfs* F_{X_i} *are continuous, then* $C_{\mathbf{X}}$ *is unique.*

Proof. The proof is elaborate, see Sklar (1996). □

Theorem 4.3.2 (Relation Joint Probability Distributrion and Copulas)**.** *Let* $\mathcal{N}^{\mathrm{ncat}} = \{1, \ldots, n^{\mathrm{ncat}}\}$ *be a set of* $n^{\mathrm{ncat}} \in \{2, 3, \ldots\}$ *indices corresponding to vector of non-categorical random variables* $\mathbf{X}^{\mathrm{ncat}} = (X_1, \ldots, X_{n^{\mathrm{ncat}}})$ *with realisations* $\mathbf{x}^{\mathrm{ncat}} = (x_1, \ldots, x_{n^{\mathrm{ncat}}})$*. Moreover, let* $\mathcal{N}^{\mathrm{cont}} = \{i_1, \ldots, i_n\} \subseteq \mathcal{N}^{\mathrm{ncat}}$ *and* $\mathcal{N}^{\mathrm{disc}} = \{j_1, \ldots, j_n\} \subseteq \mathcal{N}^{\mathrm{ncat}}$ *be sets of indices corresponding continuous and discrete random variables respectively, where* $\mathcal{N}^{\mathrm{cont}} \cap \mathcal{N}^{\mathrm{disc}} = \emptyset$*. We denote* $|\mathcal{N}^{\mathrm{cont}}| = n^{\mathrm{cont}}$ *and* $|\mathcal{N}^{\mathrm{cont}}| = n^{\mathrm{disc}}$*.*

Now, let $F_{X_k}(x_k)$ *denote a marginal CDF of* X_k *evaluated in* x_k *for* $k \in \mathcal{N}^{\mathrm{ncat}}$*, let* $f_{X_i}(x_i)$ *denote a continuous marginal pdf of* X_i *evaluated in* x_i *for* $i \in \mathcal{N}^{\mathrm{cont}}$ *and let* $F_{X_j}\left(x_j^-\right) \equiv \lim_{z \to x_j^-} F_{X_j}(z)$ *for* $j \in \mathcal{N}^{\mathrm{disc}}$*. Without loss of generality, let the first* n^{cont} *arguments of the copula* $C_{\mathbf{X}^{\mathrm{ncat}}}$ *be associated with realisations of continuous random variables. Then, the following relation holds:*

$$\begin{aligned} & f_{\mathbf{X}^{\mathrm{ncat}}}\left(\mathbf{x}^{\mathrm{ncat}}\right) \\ &= \prod_{i \in \mathcal{N}^{\mathrm{cont}}} f_{X_i}(x_i) \sum_{\left(F_{X_j}\left(x_j^*\right)\right)_{j \in \mathcal{N}^{\mathrm{disc}}} \in \mathcal{U}} (-1)^{\sum_{j \in \mathcal{N}^{\mathrm{disc}}} \left[F_{X_j}\left(x_j^-\right) = F_{X_j}\left(x_j^*\right)\right]} \\ &\quad \times D_{1,\ldots,n^{\mathrm{cont}}} C_{\mathbf{X}^{\mathrm{ncat}}}\left(F_{X_{i_1}}(x_{i_1}), \ldots, F_{X_{i_n}}(x_{i_n}), F_{X_{j_1}}\left(x_{j_1}^*\right), \ldots, F_{X_{j_n}}\left(x_{j_n}^*\right)\right) \end{aligned} \tag{8}$$

where $\mathcal{U} = \prod_{j \in \mathcal{N}^{\mathrm{disc}}} \left\{F_{X_j}\left(x_j^-\right), F_{X_j}(x_j)\right\}$ *is the* n^{disc}*-ary Cartesian product of all* n^{disc} *discrete realisations and their left limits with the Cartesian product containing* $|\mathcal{U}| = 2^{n^{\mathrm{disc}}}$ n^{disc}*-tuples* $\left(F_{X_{j_1}}\left(x_{j_1}^*\right), \ldots, F_{X_{j_n}}\left(x_{j_n}^*\right)\right)$*. We use Euler's differential notation, meaning*

that we take partial derivatives of C with respect to the first n^{cont} arguments first. Then, *we evaluated that function for specific values of the arguments.*

Proof. First we employ Sklar's Theorem as seen in Theorem 4.3.1. Then, (8) follows from taking n^{cont} times the partial derivative of the continuous marginal cdfs and $2^{n^{\text{disc}}}$ differences to account for the discrete marginal cdfs. □

Corollary 4.3.2.1. *Given the notation in Theorem 4.3.2, let $n^{\text{cont}} = n$. This means that $\mathcal{N}^{\text{disc}} = \emptyset$ and $\mathcal{N}^{\text{cont}} = \mathcal{N}^{\text{ncat}}$. Thus, all non-categorical marginal CDFs F_{X_i} are continuous with $i \in \mathcal{N}^{\text{ncat}}$. Then,* (8) *equals:*

$$D_{1,\ldots,n} C_{\mathbf{X}^{\text{ncat}}} \left(F_{X_1}(x_1), \ldots, F_{X_n}(x_n) \right) \prod_{i \in \mathcal{N}} f_{X_i}(x_i). \tag{9}$$

Proof. This follows directly from (8) as $n^{\text{cont}} = n$, thus $n^{\text{disc}} = 0$ and $\mathcal{N}^{\text{disc}} = \mathcal{U} = \emptyset$. □

Corollary 4.3.2.2. *Given the notation in Theorem 4.3.2, let $n^{\text{disc}} = n$. This means that $\mathcal{N}^{\text{cont}} = \emptyset$ and $\mathcal{N}^{\text{disc}} = \mathcal{N}^{\text{ncat}}$. Thus, all non-categorical marginal CDFs F_{X_i} are ordinal discrete with $i \in \mathcal{N}^{\text{disc}}$. Then,* (8) *equals:*

$$\sum_{\left(F_{X_j}\left(x_j^*\right)\right)_{j \in \mathcal{N}} \in \mathcal{U}} (-1)^{\sum_{j \in \mathcal{N}} \left[F_{X_j}\left(x_j^-\right) = F_{X_j}\left(x_j^*\right)\right]} C_{\mathbf{X}^{\text{ncat}}} \left(F_{X_1}\left(x_1^*\right), \ldots, F_{X_n}\left(x_n^*\right) \right). \tag{10}$$

Proof. This follows directly from (8) as $n^{\text{disc}} = n$, thus $n^{\text{cont}} = 0$ and $\mathcal{N}^{\text{cont}} = \emptyset$. □

We employ a probabilistic definition of the copula $C_{\mathbf{X}}$. See Appendix A.2 for an analytical definition of a copula. For an elaborate introduction to copulas, see Nelsen (2007).

Elidan (2012) uses a conditional variant of (9) to derive his copula-based BN. If one uses this equation when some marginals are in fact discrete, then the model is misspecified. Intuitively, (8) has to be used in order to accurately model mixed marginals.

5 Predictive Copula Bayesian Network

In this section, we introduce the predictive copula Bayesian network (PCBN) and give various algorithms used by this framework. We start with the graph structure of PCBNs and follow with the parametrisation of the joint probability function. Then, we elaborate on the three estimation techniques and explain when they are applicable. After that, we explain how predictions can be made with PCBNs. Using the preceding sections, we give a greedy algorithm that is used to construct a PCBN from the data. Finally, we introduce the copula spider, a technique for which several PCBNs are needed.

5.1 Graph Structure

The goal of this thesis is to find a good probabilistic classification function h^{prob} given by (1). We do this by creating a Bayesian network $\mathcal{B} = (\mathcal{G}, \Theta)$ as given in Definition 4.2.2 to model $f_{Y,\mathbf{X}}(y, \mathbf{x})$ given by (6). This formula depends on both $f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y)$ and $f_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}_{X_i}\right)$ for $i \in \mathcal{N}$. However, not all $f_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}_{X_i}\right)$ need to be evaluated in order to determine $\hat{y}$. Thus, not all DAGs $\mathcal{G}$ need to be considered when the data is complete. In order to see this, let $\mathcal{N}_{\text{de}(Y)}$, $\mathcal{N}_{\text{ch}(Y)}$ and $\mathcal{N}_{\text{nd}(Y)}$ denote the sets of nodes that are descendants, children and non-descendants of Y respectively. A child of Y is a direct descendant of Y, thus $\mathcal{N}_{\text{ch}(Y)} \subseteq \mathcal{N}_{\text{de}(Y)}$. Note that $\mathcal{N}_{\text{de}(Y)} \cap \mathcal{N}_{\text{nd}(Y)} = \mathcal{N}_{\text{ch}(Y)} \cap \mathcal{N}_{\text{nd}(Y)} = \emptyset$ as a node cannot be a descendant and non-descendant at the same time.

Now, we can rewrite the decomposition of the joint probability function $f_{Y,\mathbf{X}}(y, \mathbf{x})$ as:

$$
\begin{aligned}
& f_{Y,\mathbf{X}}(y, \mathbf{x}) \\
& = f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) \prod_{i \in \mathcal{N}} f_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}_{X_i}\right) \\
& = f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) \prod_{i^+ \in \mathcal{N}_{\text{nd}(Y)}} f_{X_{i^+}|\mathbf{Pa}_{X_{i^+}}}\left(x_{i^+}|\mathbf{pa}_{X_{i^+}}\right) \\
& \quad \times \prod_{i^- \in \mathcal{N}_{\text{de}(Y)}} f_{X_{i^-}|\mathbf{Pa}_{X_{i^-}}}\left(x_{i^-}|\mathbf{pa}_{X_{i^-}}\right) \\
& \propto f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) \prod_{i^- \in \mathcal{N}_{\text{de}(Y)}} f_{X_{i^-}|\mathbf{Pa}_{X_{i^-}}}\left(x_{i^-}|\mathbf{pa}_{X_{i^-}}\right) \\
& \propto f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) \prod_{j \in \mathcal{N}_{\text{ch}(Y)}} f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right). \qquad (11)
\end{aligned}
$$

Thus, we only need to consider two types of relations when we want to determine $\hat{y}$. The first type is that of the dependent variable Y and its parents $\mathbf{Pa}_Y$. The second type is that of Y, its children and their parents. Intuitively, $f_{X_{i^+}|\mathbf{Pa}_{X_{i^+}}}\left(x_{i^+}|\mathbf{pa}_{X_{i^+}}\right)$ does not depend on Y for $i^+ \in \mathcal{N}_{\mathrm{nd}(Y)}$ as the parent vector of each non-descendant $\mathbf{Pa}_{X_{i^+}}$ cannot contain Y. Thus, the relations between the non-descendants and their parents does not need to be considered. Moreover, $f_{X_{i^-}|\mathbf{Pa}_{X_{i^-}}}\left(x_{i^-}|\mathbf{pa}_{X_{i^-}}\right)$ does not depend on Y for $i^- \in \mathcal{N}_{\mathrm{de}(Y)} \setminus \mathcal{N}_{\mathrm{ch}(Y)}$ as the parent vector of every non-direct descendant of Y cannot have Y as one of its parents. If it did, it would be a child of Y. Note that (11) needs to be normalised if we wish to output proper probabilities. (11) is referred to as the *Markov blanket* of Y (Pearl, 1988).

The expression for $f_{Y,\mathbf{X}}$ given by (11) forms the basis of PCBNs. Note that this expression is richer than both tree-augmented and Bayesian-augmented Naïve Bayes classifiers, as the parents $\mathbf{Pa}_{X_j}$ of X_j do not necessarily need to have Y as a parent. Moreover, Y is allowed to have parents as well.

5.2 Parametrisation

The decomposition of the joint density of the copula-based Bayesian network is given by (11). The idea of the parametrisation of $f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right)$ is to condition on *all* categorical variables. Let $\mathbf{Pa}_{X_j}^{\mathrm{ncat}}$ denote the vector containing all parent variables of X_j that are continuous or ordinal discrete and let $\mathbf{Pa}_{X_j}^{\mathrm{cat}}$ denote the vector containing all parent variables of X_j that are categorical. Let $\mathbf{pa}_{X_j}^{\mathrm{ncat}}$ and $\mathbf{pa}_{X_j}^{\mathrm{cat}}$ denote the corresponding realisation vectors of $\mathbf{Pa}_{X_j}^{\mathrm{ncat}}$ and $\mathbf{Pa}_{X_j}^{\mathrm{cat}}$ respectively. Note that $\mathbf{Pa}_{X_j}^{\mathrm{cat}}$ contains Y. Now, we model the probability function $f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right)$ with (12).

$$f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right) = \frac{f_{X_j,\mathbf{Pa}_{X_j}^{\mathrm{ncat}}|\mathbf{Pa}_{X_j}^{\mathrm{cat}}}\left(x_j,\mathbf{pa}_{X_j}^{\mathrm{ncat}}|\mathbf{pa}_{X_j}^{\mathrm{cat}}\right)}{f_{\mathbf{Pa}_{X_j}^{\mathrm{ncat}}|\mathbf{Pa}_{X_j}^{\mathrm{cat}}}\left(\mathbf{pa}_{X_j}^{\mathrm{ncat}}|\mathbf{pa}_{X_j}^{\mathrm{cat}}\right)} \tag{12}$$

If X_j is categorical, then we model the probability function $f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right)$ with (13).

$$f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right) = \frac{f_{\mathbf{Pa}_{X_j}^{\mathrm{ncat}}|X_j,\mathbf{Pa}_{X_j}^{\mathrm{cat}}}\left(\mathbf{pa}_{X_j}^{\mathrm{ncat}}|x_j,\mathbf{pa}_{X_j}^{\mathrm{cat}}\right) f_{X_j|\mathbf{Pa}_{X_j}^{\mathrm{cat}}}\left(x_j|\mathbf{pa}_{X_j}^{\mathrm{cat}}\right)}{f_{\mathbf{Pa}_{X_j}^{\mathrm{ncat}}|\mathbf{Pa}_{X_j}^{\mathrm{cat}}}\left(\mathbf{pa}_{X_j}^{\mathrm{ncat}}|\mathbf{pa}_{X_j}^{\mathrm{cat}}\right)} \tag{13}$$

Using the parametrisation of (12) and (13), discrete nodes can have continuous children and vice versa. Note that $f_{X_j|\mathbf{Pa}_{X_j}^{\text{cat}}}\left(x_j|\mathbf{pa}_{X_j}^{\text{cat}}\right) = \mathrm{P}\left[X_j = x_j|\mathbf{Pa}_{X_j}^{\text{cat}} = \mathbf{pa}_{X_j}^{\text{cat}}\right]$ in (13). If we need to model a joint conditional distribution, then we use (8).

The final term that needs to be modelled is $f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y)$. For this conditional probability function, we use the following:

$$f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) = \frac{f_{Y,\mathbf{Pa}_Y}(y,\mathbf{pa}_Y)}{f_{\mathbf{Pa}_Y}(\mathbf{pa}_Y)} \propto f_{Y,\mathbf{Pa}_Y}(y,\mathbf{pa}_Y)\,.$$

Note that $f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right)$ is not proportional to $f_{X_j,\mathbf{Pa}_{X_j}}\left(x_j,\mathbf{pa}_{X_j}\right)$ as $f_{\mathbf{Pa}_{X_j}}\left(\mathbf{pa}_{X_j}\right)$ depends on Y.

The idea of the parametrisation of $f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y)$ is to condition on all categorical variables, similar to the parametrisation of $f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right)$. Let $\mathbf{Pa}_Y^{\text{ncat}}$ denote the vector containing all parent variables of Y that are continuous or ordinal discrete and let $\mathbf{Pa}_Y^{\text{cat}}$ denote the vector of all categorical parent variables of Y. The corresponding realisation vectors of $\mathbf{Pa}_Y^{\text{ncat}}$ and $\mathbf{Pa}_Y^{\text{cat}}$ are $\mathbf{pa}_Y^{\text{ncat}}$ and $\mathbf{pa}_Y^{\text{ncat}}$ respectively. Now, we rewrite $f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y)$ as

$$f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) \propto f_{\mathbf{Pa}_Y^{\text{ncat}}|Y,\mathbf{Pa}_Y^{\text{cat}}}\left(\mathbf{pa}_Y^{\text{ncat}}|y,\mathbf{pa}_Y^{\text{cat}}\right) f_{Y,\mathbf{Pa}_Y^{\text{cat}}}\left(y,\mathbf{pa}_Y^{\text{cat}}\right). \tag{14}$$

Note that $f_{Y,\mathbf{Pa}_Y^{\text{cat}}}(y,\mathbf{pa}_Y^{\text{cat}}) = \mathrm{P}\left[Y = y,\mathbf{Pa}_Y^{\text{cat}} = \mathbf{pa}_Y^{\text{cat}}\right]$. If Y does not have any parents, then (14) reduces to $f_Y(y) = \mathrm{P}[Y = y]$.

5.2.1 Decomposition Algorithm

If we want to determine the (conditional) probability functions associated with node j given set of nodes $\mathcal{N}$, then we use either (13), (12) or (14). In order to use these equations, we need to determine the relevant random variables. This has been implemented in Algorithm 1, called GETVARIABLES. This procedure returns the random variable Z_j associated with node $j \in \{0,\ldots,n\}$, meaning that Z_j is either the dependent variable Y or the explanatory variable X_j, and four ordered vectors of random variables $\mathbf{Z}^{\text{ncat}}, \mathbf{Z}^{\text{cat}}, \mathbf{Z}^{\text{ncat}-}$ and $\mathbf{Z}^{\text{cat}-}$. Here, ncat and cat indicate that the vector contains solely non-categorical and categorical random variables respectively. A minus indicates that Z_j is not included,

whereas $\mathbf{Z}^{\text{ncat}}$ and $\mathbf{Z}^{\text{cat}}$ might include Z_j if the type matches.

Using these relevant random variables, we can get all (conditional) probability functions associated with node j given set of nodes $\mathcal{N}$. This has been implemented in Algorithm 2, called DECOMPOSEFUNCTION. In line 4 of Algorithm 2 we determine the set $\mathcal{Z}^{\text{cat}}$. This set contains all possible realisations $\mathbf{z}^{\text{cat}}$. If $j = 0$, meaning that $Z_j = Y$, then we use (14) as seen in line 6. If $j \neq 0$, meaning that $Z_j \neq Y$, then we use (13) if Z_j is a categorical random variable as seen in lines 9 to 12. Note that we also need to determine the set $\mathcal{Z}^{\text{cat}-}$ containing all possible realisations $\mathbf{z}^{\text{cat}-}$. Lastly, as seen in line 14, we use (12) if $j \neq 0$ and Z_j is a non-categorical random variable.

Algorithm 1: GETVARIABLES$(\mathcal{D}, j, \mathcal{N})$

Input: data $\mathcal{D} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}\right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ unique indices, index $j \in \{0, \ldots, n\}$, set of nodes to use $\mathcal{N} \subseteq \{0, \ldots, n\}$ such that $j \notin \mathcal{N}$

Result: random variable Z_j associated with node j, ordered vectors $\mathbf{Z}^{\text{ncat}}$ and $\mathbf{Z}^{\text{cat}}$ containing non-categorical and categorical random variables respectively and possible including Z_j if the type matches, and $\mathbf{Z}^{\text{ncat}-}$ and $\mathbf{Z}^{\text{cat}-}$ containing non-categorical and categorical random variables respectively and possible excluding Z_j

1 **Procedure:** GETVARIABLES$(\mathcal{D}, j, \mathcal{N})$
2 $\quad Z_j \leftarrow$ random variable associated with j;
3 $\quad \mathcal{N} \leftarrow \mathcal{N} \cup \{j\}$;
4 $\quad \mathcal{N}^{\text{ncat}} \leftarrow \{i : i \in \mathcal{N}, \text{type of } i \text{ is non-categorical}\}$;
5 $\quad \mathcal{N}^{\text{cat}} \leftarrow \mathcal{N} \setminus \mathcal{N}^{\text{ncat}}$;
6 $\quad \mathbf{Z}^{\text{ncat}} \leftarrow$ ordered vector of all $|\mathcal{N}^{\text{ncat}}|$ random variables associated with $\mathcal{N}^{\text{ncat}}$;
7 $\quad \mathbf{Z}^{\text{cat}} \leftarrow$ ordered vector of all $|\mathcal{N}^{\text{cat}-}|$ random variables associated with $\mathcal{N}^{\text{cat}}$;
8 $\quad \mathbf{Z}^{\text{ncat}-} \leftarrow$ ordered vector of all $|\mathcal{N}^{\text{ncat}} \setminus \{j\}|$ random variables associated with $\mathcal{N}^{\text{ncat}} \setminus \{j\}$;
9 $\quad \mathbf{Z}^{\text{cat}-} \leftarrow$ ordered vector of all $|\mathcal{N}^{\text{cat}} \setminus \{j\}|$ random variables associated with $\mathcal{N}^{\text{cat}} \setminus \{j\}$;
10 **return** $(Z_j, \mathbf{Z}^{\text{ncat}}, \mathbf{Z}^{\text{cat}}, \mathbf{Z}^{\text{ncat}-}, \mathbf{Z}^{\text{cat}-})$

Algorithm 2: DECOMPOSEFUNCTION$(\mathcal{D}, j, \mathcal{N})$

Input: data $\mathcal{D} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}\right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ unique indices, index $j \in \{0, \ldots, n\}$, set of nodes to use $\mathcal{N} \subseteq \{0, \ldots, n\}$ such that $j \notin \mathcal{N}$

Result: set of probability functions $\mathcal{P}$

Procedure: DECOMPOSEFUNCTION$(\mathcal{D}, j, \mathcal{N})$

- $(Z_j, \mathbf{Z}^{\text{ncat}}, \mathbf{Z}^{\text{cat}}, \mathbf{Z}^{\text{ncat}-}, \mathbf{Z}^{\text{cat}-}) \leftarrow$ GETVARIABLES$(\mathcal{D}, j, \mathcal{N})$;
- $n^{\text{cat}} \leftarrow$ number of variables in $\mathbf{Z}^{\text{cat}}$;
- $\mathcal{Z}^{\text{cat}} \leftarrow \text{Ran}\left(Z_1^{\text{cat}}\right) \times \ldots \times \text{Ran}\left(Z_{n^{\text{cat}}}^{\text{cat}}\right)$; set of all n^{cat}-tuples
- **if** $j = 0$ **then**
  - $\mathcal{P} \leftarrow \bigcup_{\mathbf{z}^{\text{cat}} \in \mathcal{Z}^{\text{cat}}} \left\{f_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}}}\left(\mathbf{z}^{\text{ncat}}|\mathbf{z}^{\text{cat}}\right)\right\} \cup \left\{f_{\mathbf{Z}^{\text{cat}}}\left(\mathbf{z}^{\text{cat}}\right)\right\}$; see (14)
- **else**
  - **if** Z_j is a categorical random variable **then**
    - $n^{\text{cat}-} \leftarrow$ number of variables in $\mathbf{Z}^{\text{cat}-}$;
    - $\mathcal{Z}^{\text{cat}-} \leftarrow \text{Ran}\left(Z_1^{\text{cat}-}\right) \times \ldots \times \text{Ran}\left(Z_{n^{\text{cat}-}}^{\text{cat}-}\right)$; set of all $n^{\text{cat}-}$-tuples
    - $\mathcal{P} \leftarrow \bigcup_{\mathbf{z}^{\text{cat}-} \in \mathcal{Z}^{\text{cat}-}} \left\{f_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}-}}\left(\mathbf{z}^{\text{ncat}}|\mathbf{z}^{\text{cat}-}\right)\right\} \cup \left\{f_{Z_j|\mathbf{Z}^{\text{cat}-}}\left(z_j|\mathbf{z}^{\text{cat}-}\right)\right\}$;
    - $\mathcal{P} \leftarrow \mathcal{P} \cup \bigcup_{\mathbf{z}^{\text{cat}} \in \mathcal{Z}^{\text{cat}}} \left\{f_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}}}\left(\mathbf{z}^{\text{ncat}}|\mathbf{z}^{\text{cat}}\right)\right\}$; see (13)
  - **else**
    - $\mathcal{P} \leftarrow \bigcup_{\mathbf{z}^{\text{cat}} \in \mathcal{Z}^{\text{cat}}} \left\{f_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}}}\left(\mathbf{z}^{\text{ncat}}|\mathbf{z}^{\text{cat}}\right)\right\} \cup \left\{f_{\mathbf{Z}^{\text{ncat}-}|\mathbf{Z}^{\text{cat}}}\left(\mathbf{z}^{\text{ncat}-}|\mathbf{z}^{\text{cat}}\right)\right\}$; see (12)
  - **end**
- **end**

return $\mathcal{P}$

5.3 Parameter Estimation

The key functions to estimate are given by either (12) or (13) for each X_j and (14) for Y and its parents. Assume that we have $\mathcal{D} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}\right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| = d \in \mathbb{N}^+$ indices. Now, we can estimate these probabilities. Essentially, the following three situations arise:

1. Continuous or ordinal discrete kernel density estimation
2. Constructing (conditional) probability tables with counting
3. Parameter estimation of a given copula and the estimation of continuous and or discrete CDFs.

The following sections explain these estimation procedures and explain when they arise.

5.3.1 Kernel Density Estimation

Let X_j be a continuous or ordinal discrete random variable with d realisations $x_j^{(l)}$ for $l \in \mathcal{M}$ and let f_{X_j} be the probability function of X_j. Rosenblatt (1956) and Parzen (1962) show that we can estimate f_{X_j} with the kernel density estimator of X_j given by

$$\hat{f}_{X_j}(x_j|h) = \frac{1}{dh} \sum_{l \in \mathcal{M}} K\left(x_j, x_j^{(l)}, h\right) \tag{15}$$

where $K(\cdot)$ is a kernel function and h is the bandwidth parameter and $x_j^{(l)}$ are realisations from the data $\mathcal{D}$. h can be estimated from the data itself by means of cross validation. The unconditional kernel density estimator is given by (15), but we frequently need to estimate conditional distributions. Let $\mathbf{Z}$ be a vector of categorical random variables with corresponding realisation vector $\mathbf{z}$. Then, we could partition the data for all $\mathbf{z}$ such that we estimate $f_{X_j|\mathbf{Z}}$ by

$$\hat{f}_{X_j|\mathbf{Z}}\left(x_j|\mathbf{z}, \hat{h}_{X_j|\mathbf{z}}\right) \equiv \frac{1}{d\hat{h}_{X_j|\mathbf{z}}} \sum_{l \in \mathcal{M}:\mathbf{z}^{(l)}=\mathbf{z}} K\left(x_j, x_j^{(l)}, \hat{h}_{X_j|\mathbf{z}}\right) \tag{16}$$

where $\hat{h}_{X_j|\mathbf{z}^{\text{cat}}}$ is either estimated or chosen beforehand. Note that the bandwidth parameter is different for each possible realisation $\mathbf{z}$. Using (16) means that we estimate several *unconditional* densities only. Although conditional density estimation with bandwidth selection exists, see Hall, Racine, and Li (2004), it is computationally demanding.

If X_j is continuous, then one could consider the Gaussian kernel given by

$$K\left(x_j, x_j^{(l)}, h\right) = (2\pi)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(\frac{x_j - x_j^{(l)}}{h}\right)^2\right\}. \tag{17}$$

Now, let X_j be an ordinal discrete random variable with $\text{Ran}(X_1) = \{0, 1, \ldots, k_j - 1\}$ for $k_j \in \mathbb{N}^+$. Then, following the suggestion of Aitchison and Aitken (1976), we could use the kernel given by

$$K\left(x_j, x_j^{(l)}, h\right) = \frac{k_j!}{\left|x_j - x_j^{(l)}\right|!\left(k_j - \left|x_j - x_j^{(l)}\right|\right)!} h^{\left|x_j - x_j^{(l)}\right|}(1-h)^{k_j - \left|x_j - x_j^{(l)}\right|}. \tag{18}$$

The Aitchison and Aitken kernel reduces to a frequency estimator when $h \to 0$. Using a frequency estimator, instead of (18), may give unreliable results when we have few observations.

Kernel density estimation always arises when we need to estimate a conditional probability function of exactly *one* non-categorical random variable given either one or more categorical random variables. This happens in the following situations:

1. The dependent variable Y has exactly *one* non-categorical parent. This means that kernel density estimation can be used on $f_{\mathbf{Pa}_Y^{\text{ncat}}|Y,\mathbf{Pa}_Y^{\text{cat}}}\left(\mathbf{pa}_Y^{\text{ncat}}|y,\mathbf{pa}_Y^{\text{cat}}\right)$ in (14).

2. Explanatory variable X_j is a *non-categorical* child of Y that does not have any non-categorical parents. This means that kernel density estimation can be used on $f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j|\mathbf{pa}_{X_j}\right)$.

3. Explanatory variable X_j is a child of Y that has exactly *one* non-categorical parent. This means kernel density estimation can be used on $f_{\mathbf{Pa}_{X_j}^{\text{ncat}}|\mathbf{Pa}_{X_j}^{\text{cat}}}\left(\mathbf{pa}_{X_j}^{\text{ncat}}|\mathbf{pa}_{X_j}^{\text{cat}}\right)$ in (12) or (13). Additionally, if X_j is categorical, then kernel density estimation can be used on $f_{\mathbf{Pa}_{X_j}^{\text{ncat}}|X_j,\mathbf{Pa}_{X_j}^{\text{cat}}}\left(\mathbf{pa}_{X_j}^{\text{ncat}}|x_j,\mathbf{pa}_{X_j}^{\text{cat}}\right)$ in (13).

We use continuous and ordinal discrete kernel density estimation when the non-categorical random variable is continuous or ordinal discrete respectively. Kernel density estimation may also arise when we model multivariate densities, see Section 5.3.3.

5.3.2 Probability Tables and Counting

The construction of (conditional) probability tables arises when we need to estimate a probability that solely contains categorical random variables. This happens in the following situations:

1. The estimation of $f_{Y|\mathbf{Pa}_Y}\left(y|\mathbf{pa}_Y\right)$ always requires the construction of a probability table. We do this for $f_{Y,\mathbf{Pa}_Y^{\text{cat}}}\left(y,\mathbf{pa}_Y^{\text{cat}}\right)$ in (14).

2. Explanatory variable X_j is a *categorical* child of Y. This means that a conditional probability table needs to be constructed for $f_{X_j|\mathbf{Pa}_{X_j}^{\text{cat}}}\left(x_j|\mathbf{pa}_{X_j}^{\text{cat}}\right)$ in (13).

The probabilities can be calculated by means of counting:

$$\hat{f}_{Y,\mathbf{Pa}_Y^{\text{cat}}}\left(y, \mathbf{pa}_Y^{\text{cat}}\right) \equiv \frac{1}{d} \sum_{l \in \mathcal{M}} \left[y = y^{(l)}\right] \left[\mathbf{pa}_Y^{\text{cat}} = \mathbf{pa}_Y^{\text{cat},(l)}\right] \tag{19}$$

and

$$\hat{f}_{X_j|\mathbf{Pa}_{X_j}^{\text{cat}}}\left(x_j|\mathbf{pa}_{X_j}^{\text{cat}}\right) \equiv \frac{\sum_{l \in \mathcal{M}} \left[x_j = x_j^{(l)}\right] \left[\mathbf{pa}_{X_j}^{\text{cat}} = \mathbf{pa}_{X_j}^{\text{cat},(l)}\right]}{\sum_{m \in \mathcal{M}} \left[\mathbf{pa}_{X_j}^{\text{cat}} = \mathbf{pa}_{X_j}^{\text{cat},(m)}\right]}. \tag{20}$$

It is possible that (19) equals 0 for some combinations of y and $\mathbf{pa}_Y^{\text{cat}}$. Moreover, (20) is undefined for some combinations of x_j and $\mathbf{pa}_{X_j}^{\text{cat}}$. In these cases, one might set the (conditional) probability arbitrarily to a small value ε to avoid having $f_{Y,\mathbf{X}}(y, \mathbf{x}) = 0$ for some y and $\mathbf{x}$. Intuitively, if some combination of variables does not occur it does not necessarily mean that the probability is zero. Strictly speaking, other combinations of (19) and (20) need to be rescaled when some combinations are set to ε instead of 0.

5.3.3 Copula Parameter Estimation

Equation (8) describes the relation between a joint probability distribution and a copula. Copula parameter estimation is needed for every probability function that contains at least *two* non-categorical random variables. Such probability functions arise in the following situations:

1. The dependent variable Y has at least *two* non-categorical parents. We model $f_{\mathbf{Pa}_Y^{\text{ncat}}|Y,\mathbf{Pa}_Y^{\text{cat}}}\left(\mathbf{pa}_Y^{\text{ncat}}|y, \mathbf{pa}_Y^{\text{cat}}\right)$ in (14) with (8) and a copula function C. We do this for each possible $y \in \mathcal{K}$ and $\mathbf{pa}_Y^{\text{cat}}$.

2. Explanatory variable X_j is a *non-categorical* child of Y that has *at least one* non-categorical parent. We model $f_{X_j,\mathbf{Pa}_{X_j}^{\text{ncat}}|\mathbf{Pa}_{X_j}^{\text{cat}}}\left(x_j, \mathbf{pa}_{X_j}^{\text{ncat}}|\mathbf{pa}_{X_j}^{\text{cat}}\right)$ with (8) and a copula function C. We do this for each possible $\mathbf{pa}_{X_j}^{\text{cat}}$.

3. Explanatory variable X_j is a child of Y that has *more than one* non-categorical parents. We model $f_{\mathbf{Pa}_{X_j}^{\text{ncat}}|\mathbf{Pa}_{X_j}^{\text{cat}}}\left(\mathbf{pa}_{X_j}^{\text{ncat}}|\mathbf{pa}_{X_j}^{\text{cat}}\right)$ in (12) or (13) with (8) and a copula function C. We do this for each possible $\mathbf{pa}_{X_j}^{\text{cat}}$. Additionally, if X_j is categorical, then (8) and a copula function can also be used on $f_{\mathbf{Pa}_{X_j}^{\text{ncat}}|X_j,\mathbf{Pa}_{X_j}^{\text{cat}}}\left(\mathbf{pa}_{X_j}^{\text{ncat}}|x_j, \mathbf{pa}_{X_j}^{\text{cat}}\right)$ in (13) for each possible x_j and $\mathbf{pa}_{X_j}^{\text{cat}}$.

Equation (8) requires the estimation of continuous and/or discrete CDFs. We also need kernel density estimation for each continuous random variable in the joint probability. Moreover, the chosen copula C contains an unknown parameter θ. As a result, (8) depends on θ as well. In order to obtain estimates $\hat{\theta}$ given realisations of categorical random variables, we consider a likelihood-based approach.

Given a Bayesian network $\mathcal{B}$, the joint probability can be determined by (11). An expression for the likelihood of Θ given the data $\mathcal{D}$ is given by:

$$\mathcal{L}\left(\Theta|\mathcal{D}\right) \propto \prod_{l \in \mathcal{M}} f_{Y|\mathbf{Pa}_Y}\left(y^{(l)}|\mathbf{pa}_Y^{(l)}\right) \prod_{j \in \mathcal{N}_{\mathrm{ch}(Y)}} f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j^{(l)}|\mathbf{pa}_{X_j}^{(l)}\right).$$

Then, the log-likelihood is given by (21).

$$\log \mathcal{L}\left(\Theta|\mathcal{D}\right) \propto \sum_{l \in \mathcal{M}} \log f_{Y|\mathbf{Pa}_Y}\left(y^{(l)}|\mathbf{pa}_Y^{(l)}\right) + \sum_{l \in \mathcal{M}} \sum_{j \in \mathcal{N}_{\mathrm{ch}(Y)}} \log f_{X_j|\mathbf{Pa}_{X_j}}\left(x_j^{(l)}|\mathbf{pa}_{X_j}^{(l)}\right) \tag{21}$$

Maximising the log-likelihood can be done by maximising the log probability functions of (21) individually.

Let $\mathbf{X}^{\mathrm{ncat}}$ and $\mathbf{Z}$ be vectors of non-categorical and categorical random variables respectively, with $\mathbf{X}^{\mathrm{ncat}}$ containing at least two random variables. Moreover, let $\mathbf{x}^{\mathrm{ncat},(l)}$ and $\mathbf{z}^{(l)}$ denote realisation vectors l of $\mathbf{X}^{\mathrm{ncat}}$ and $\mathbf{Z}$ respectively with $l \in \mathcal{M} : \mathbf{z}^{(l)} = \mathbf{z}$. Now, the parameter (vector) estimate $\hat{\theta}_{\mathbf{z}}$ can be determined by (22) where $f_{\mathbf{X}^{\mathrm{ncat}}|\mathbf{Z}}\left(\mathbf{x}^{\mathrm{ncat},(l)}|\mathbf{z}\right)$ can be modelled by (8).

$$\hat{\theta}_{\mathbf{z}}^{*} \in \arg\max_{\theta} \left\{ \sum_{l \in \mathcal{M}:\mathbf{z}^{(l)}=\mathbf{z}} \log f_{\mathbf{X}^{\mathrm{ncat}}|\mathbf{Z}}\left(\mathbf{x}^{\mathrm{ncat},(l)}|\mathbf{z},\theta\right) \right\} \tag{22}$$

However, (22) requires expressions for $F_{X_1|\mathbf{Z}}, \ldots, F_{X_n|\mathbf{Z}}$ but these are, in fact, unknown. Fortunately, we can use the inference function for margins (IFM) approach proposed by Joe and Xu (1996). Here, we estimate the univariate margins first. Then, we estimate the multivariate distribution using the estimates for the univariate margins. In this case, applying IFM means we estimate $F_{X_1|\mathbf{Z}}, \ldots, F_{X_n|\mathbf{Z}}$ first. Using estimates $\hat{F}_{X_1|\mathbf{Z}}, \ldots, \hat{F}_{X_n|\mathbf{Z}}$

we obtain $\hat{\theta}_{\mathbf{z}^{\text{cat}}}$ with equation (23)

$$\hat{\theta}_{\mathbf{z}} \in \arg\max_{\theta} \left\{ \sum_{l \in \mathcal{M}: \mathbf{z}^{(l)} = \mathbf{z}} \log \hat{f}_{\mathbf{X}^{\text{ncat}}|\mathbf{Z}} \left(\mathbf{x}^{\text{ncat},(l)} | \mathbf{z}, \theta \right) \right\} \tag{23}$$

where $\log \hat{f}_{\mathbf{X}^{\text{ncat}}|\mathbf{Z}} \left(\mathbf{x}^{\text{ncat},(l)} | \mathbf{z}, \theta \right)$ is given by (24) using a notation similar to that in Theorem 4.3.2.

$$\begin{aligned}
&\log \hat{f}_{\mathbf{X}^{\text{ncat}}|\mathbf{Z}} \left(\mathbf{x}^{\text{ncat},(l)} | \mathbf{z}, \theta \right) \\
&= \sum_{\left(\hat{F}_{X_j|\mathbf{Z}} \left(x_j^{*,(l)} | \mathbf{z} \right) \right)_{j \in \mathcal{N}^{\text{disc}}} \in \mathcal{U}} (-1)^{\sum_{j \in \mathcal{N}^{\text{disc}}} \left[\hat{F}_{X_j|\mathbf{Z}} \left(x_j^{-,(l)} | \mathbf{z} \right) = \hat{F}_{X_j|\mathbf{Z}} \left(x_j^{*,(l)} | \mathbf{z} \right) \right]} \\
&\quad \times D_{1,\ldots,n^{\text{cont}}} C_{\mathbf{X}^{\text{ncat}}|\mathbf{Z}} \left(\hat{F}_{X_{i_1}|\mathbf{Z}} \left(x_{i_1}^{(l)} | \mathbf{z} \right), \ldots, \hat{F}_{X_{i_n}|\mathbf{Z}} \left(x_{i_n}^{(l)} | \mathbf{z} \right), \right. \\
&\quad \left. F_{X_{j_1}|\mathbf{Z}} \left(x_{j_1}^{*,(l)} | \mathbf{z} \right), \ldots, F_{X_{j_n}|\mathbf{Z}} \left(x_{j_n}^{*,(l)} | \mathbf{z} \right) | \theta \right) + \sum_{i \in \mathcal{N}^{\text{cont}}} \log \hat{f}_{X_i|\mathbf{Z}} \left(x_i^{(l)} | \mathbf{z} \right)
\end{aligned} \tag{24}$$

Note that the summation of conditional densities $\hat{f}_{X_j|\mathbf{Z}}$ does not depend on θ. Therefore, it is not needed in (23). Having obtained $\hat{\theta}_{\mathbf{z}}$, the estimated function for $f_{\mathbf{X}^{\text{ncat}}|\mathbf{Z}} \left(\mathbf{x}^{\text{ncat}} | \mathbf{z} \right)$ is given (25).

$$\begin{aligned}
&\hat{f}_{\mathbf{X}^{\text{ncat}}|\mathbf{Z}} \left(\mathbf{x}^{\text{ncat}} | \mathbf{z}, \hat{\theta}_{\mathbf{z}} \right) \\
&\equiv \sum_{\left(\hat{F}_{X_j|\mathbf{Z}} \left(x_j^{*} | \mathbf{z} \right) \right)_{j \in \mathcal{N}^{\text{disc}}} \in \mathcal{U}} (-1)^{\sum_{j \in \mathcal{N}^{\text{disc}}} \left[\hat{F}_{X_j|\mathbf{Z}} \left(x_j^{-} | \mathbf{z} \right) = \hat{F}_{X_j|\mathbf{Z}} \left(x_j^{*} | \mathbf{z} \right) \right]} \\
&\quad \times D_{1,\ldots,n^{\text{cont}}} C_{\mathbf{X}^{\text{ncat}}|\mathbf{Z}} \left(\hat{F}_{X_{i_1}|\mathbf{Z}} \left(x_{i_1} | \mathbf{z} \right), \ldots, \hat{F}_{X_{i_n}|\mathbf{Z}} \left(x_{i_n} | \mathbf{z} \right), \right. \\
&\quad \left. F_{X_{j_1}|\mathbf{Z}} \left(x_{j_1}^{*} | \mathbf{z} \right), \ldots, F_{X_{j_n}|\mathbf{Z}} \left(x_{j_n}^{*} | \mathbf{z} \right) | \hat{\theta}_{\mathbf{z}} \right) + \sum_{i \in \mathcal{N}^{\text{cont}}} \log \hat{f}_{X_i|\mathbf{Z}} \left(x_i | \mathbf{z} \right)
\end{aligned} \tag{25}$$

Note that we assume that $\hat{\theta}_{\mathbf{z}}$ in (23) exists. It is also important to realise that $\hat{\theta}_{\mathbf{z}}$ is determined by maximising a likelihood score.

Encoding a categorical random variable as an ordinal discrete random variable might obscure the true underlying dependencies. To see this, consider a categorical random variable X_1 with $\text{Ran}\left(X_1\right) = \{\text{a,b,c}\}$, a continuous random variable X_2 and some value $y^* \in \mathcal{K}$ of Y. For $Y = y^*$, let $X_1 = \text{a}$ be associated with high values of X_2, $X_1 = \text{b}$ be associated with low values of X_2 and $X_1 = \text{c}$ be associated with average values of X_2.

Using that X_1 is categorical means that we can use kernel density estimation to model X_2 given X_1 and Y. This would accurately detect the relation between X_1 and X_2 for $Y = y^*$. However, encoding X_1 as ordinal discrete such that a = 1, b = 2 and c = 3 might obscure that relation. Now, we use a copula function to model X_1 and X_2 given Y. Depending on the copula, we may not detect the dependence properly. For instance, the Clayton, Frank and Plackett copula given in Table 2 cannot model this accurately. If we used a = 1, b = 3 and c = 2 instead, then we might have detected a negative correlation with these copulas.

5.3.4 Estimation Algorithm of Local Probability Functions

An estimation algorithm of all functions belonging to node j given set of nodes $\mathcal{N}$ is given by Algorithm 3, called ESTIMATELOCAL. In line 2, we get a set $\mathcal{P}$ containing all functions that need to be estimated for j given $\mathcal{N}$. Then, we estimate all functions $f \in \mathcal{P}$. Every f is given in such a way that the estimation techniques can be applied immediately. Note that f might be conditioned on several categorical random variables.

We use kernel density estimation when f contains exactly one non-categorical random variable as seen in line 6 to 7. Copula parameter estimation is used when f contains at least two non-categorical random variables as seen in line 9 to 18. Note that we save all kernel density estimates and CDF estimates. We use multiple copulas $C \in \mathcal{C}$ when the set of copula names $\mathcal{C}$ contains more than one name. Using the same C is achieved when $|\mathcal{C}| = 1$. Lastly, we use (conditional) probability tables when f contains categorical random variables only, as seen in line 20 to 23.

Algorithm 3: estimateLocal$(\mathcal{D}, j, \mathcal{N}, \mathcal{C})$

Input: data $\mathcal{D} = \left\{ \left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M} \right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ unique indices, index $j \in \{0, \ldots, n\}$, set of nodes to use $\mathcal{N} \subseteq \{0, \ldots, n\}$ such that $j \notin \mathcal{N}$ and a set of copula names $\mathcal{C}$

Result: set of estimated probability functions $\hat{\Theta}$

Procedure: estimateLocal$(\mathcal{D}, j, \mathcal{N}, \mathcal{C})$

- $\mathcal{P} \leftarrow$ decomposeFunction$(\mathcal{D}, j, \mathcal{N})$; get all functions that need to be estimated for j given $\mathcal{N}$
- $\hat{\Theta} \leftarrow \emptyset$;
- **foreach** $f \in \mathcal{P}$ **do**
  - **if** f contains exactly one non-categorical random variable **then**
    - $\hat{h} \leftarrow$ bandwidth parameter determined or estimated beforehand;
    - $\hat{f} \leftarrow$ kernel density estimation using $\hat{h}$ and $\mathcal{D}$ (see Section 5.3.1, (16));
    - $\hat{\Theta} \leftarrow \hat{\Theta} \cup \left\{ \hat{f} \right\}$;
  - **else if** f contains more than one non-categorical random variable **then**
    - **foreach** continuous random variable $X_i \in f$ **do**
      - $\hat{h} \leftarrow$ bandwidth parameter determined or estimated beforehand;
      - $\hat{f}_{X_i} \leftarrow$ kernel density estimation using $\hat{h}$ and $\mathcal{D}$ via kernel density estimation (see Section 5.3.1, (16));
      - $\hat{h} \leftarrow$ bandwidth parameter determined or estimated beforehand;
      - $\hat{F}_{X_i} \leftarrow$ kernel density estimation using $\hat{h}$ and $\mathcal{D}$ via kernel density estimation (see Section 5.3.1, (16));
      - $\hat{\Theta} \leftarrow \hat{\Theta} \cup \left\{ \hat{f}_{X_i} \right\} \cup \left\{ \hat{F}_{X_i} \right\}$; needed for predictions
    - **end**
    - **foreach** $C \in \mathcal{C}$ **do**
      - $\hat{\theta} \leftarrow$ copula parameter estimation using all $\hat{F}_{X_i}, C$ and $\mathcal{D}$ (see Section 5.3.3, (23));
      - $\hat{f} \leftarrow$ function given by (25) using all $\hat{F}_{X_i}$, all $\hat{f}_{X_i}, C$ and $\hat{\theta}$;
      - $\hat{\Theta} \leftarrow \hat{\Theta} \cup \left\{ \hat{f} \right\}$;
    - **end**
  - **else**
    - **if** f is conditioned on random variables **then**
      - $\hat{f} \leftarrow$ conditional probability estimation using $\mathcal{D}$ (Section 5.3.2, (20));
    - **else**
      - $\hat{f} \leftarrow$ (joint) probability estimation using $\mathcal{D}$ (Section 5.3.2, (19));
    - **end**
    - $\hat{\Theta} \leftarrow \hat{\Theta} \cup \left\{ \hat{f} \right\}$;
  - **end**
- **end**
- **return** $\hat{\Theta}$

5.4 Predicting with PCBNs

Algorithm 3 details how all probability functions belonging to a node j given its parents can be estimated. These estimated functions can be used to give local predictions. That is, given (new) data $\mathcal{D}$ we can calculate $\hat{f}_{Z_j|\mathbf{Pa}_{Z_j}}$ where Z_j is the random variable associated with node j and $\mathbf{Pa}_{Z_j}$ is the vector of random variables that are parents of Z_j. As the actual realisation of Y is unknown, we need to evaluate the estimated function $\hat{f}_{Z_j|\mathbf{Pa}_{Z_j}}$ for all $Y = y^*$. Then, $\hat{f}_{Z_j|\mathbf{Pa}_{Z_j}}$ tells us *how likely* it is that we observe z_j given the parent realisations including y^*. This has been implemented in Algorithm 4 with data $\mathcal{D} = \left\{ \left(x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M} \right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ unique indices.

Given a valid DAG $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ as described in Section 5.1, we can get sets of estimated probability functions $\hat{\Theta}_j$ by calling estimateLocal on each node j given its parents. Then, we can construct a PCBN $\hat{\mathcal{B}} = \left(\mathcal{G}, \hat{\Theta} \right)$ where $\hat{\Theta} = \bigcup_j \hat{\Theta}_j$ is a union of all sets $\hat{\Theta}_j$. Now, we can get values $\hat{f}_{Y,\mathbf{X}} \left(y^*, \mathbf{x}^{(l)} \right)$ for $l \in \mathcal{M}$ by calling predictLocal on Y and all nodes with at least one parent. This has been implemented in Algorithm 5, called predictWithPCBN.

It is important to note that the outcomes $\hat{p}_{y^*}^{(l)}$ of Algorithm 5 are *not necessarily* probabilities. In order to get a probability, one needs to normalise over all $y \in \mathcal{K}$ as shown in (26). However, $\hat{p}_{y^*}^{(l)}$ can be used when a decision such as (1) is chosen, as the normalisation does not influence the outcome.

$$\hat{p}_{y^*}^{\text{proper},(l)} = \frac{\hat{p}_{y^*}^{(l)}}{\sum_{y \in \mathcal{K}} \hat{p}_y^{(l)}} \tag{26}$$

Algorithm 4: predictLocal$(\mathcal{D}, j, \mathcal{N}, \Theta, y^*)$

Input: data $\mathcal{D} = \left\{ \left(x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M} \right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ indices where m_1 and m_m are the first and last index of $\mathcal{M}$ respectively, index $j \in \{0, \ldots, n\}$, set of nodes to use $\mathcal{N} \subseteq \{0, \ldots, n\}$ such that $j \notin \mathcal{N}$, set of estimated distributions Θ and guess $y^* \in \mathcal{K}$ corresponding to random variable Y

Result: $|\mathcal{M}|$-tuple with outcomes $\hat{\ell}_{y^*}^{(m_1)}, \ldots, \hat{\ell}_{y^*}^{(m_m)}$ of evaluated functions where we assume $y = y^*$

Procedure: predictLocal$(\mathcal{D}, j, \mathcal{N}, \Theta, y^*)$
- $(Z_j, \mathbf{Z}^{\text{ncat}}, \mathbf{Z}^{\text{cat}}, \mathbf{Z}^{\text{ncat}-}, \mathbf{Z}^{\text{cat}-}) \leftarrow$ getVariables$(\mathcal{D}, j, \mathcal{N})$;
- **foreach** $l \in \mathcal{M}$ **do**
  - $\left(\mathbf{z}^{\text{ncat},(l)}, \mathbf{z}^{\text{ncat}-,(l)} \right) \leftarrow$ vectors of relations corresponding to $\mathbf{Z}^{\text{ncat}}$ and $\mathbf{Z}^{\text{ncat}-}$ respectively, with all realisations coming from $\left(x_1^{(l)}, \ldots, x_n^{(l)} \right)$;
  - $\mathbf{z}^{\text{cat},(l)} \leftarrow$ vector of relations corresponding to $\mathbf{Z}^{\text{cat}}$ respectively, with all realisations coming from $\left(x_1^{(l)}, \ldots, x_n^{(l)} \right)$ and $y = y^*$;
  - **if** $j = 0$ **then**
    - $\ell_{y^*}^{(l)} \leftarrow \hat{f}_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}}} \left(\mathbf{z}^{\text{ncat},(l)} | \mathbf{z}^{\text{cat},(l)} \right) \hat{f}_{\mathbf{Z}^{\text{cat}}} \left(\mathbf{z}^{\text{cat},(l)} \right)$; see (14)
  - **else**
    - **if** Z_j is a categorical random variable **then**
      - $\mathbf{z}^{\text{cat}-,(l)} \leftarrow$ vector of relations corresponding to $\mathbf{Z}^{\text{cat}-}$ respectively, with all realisations coming from $\left(x_1^{(l)}, \ldots, x_n^{(l)} \right)$ and $y = y^*$;
      - $\ell_{y^*}^{(l)} \leftarrow \hat{f}_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}-}} \left(\mathbf{z}^{\text{ncat},(l)} | \mathbf{z}^{\text{cat}-,(l)} \right) \hat{f}_{Z_j|\mathbf{Z}^{\text{cat}-}} \left(z_j^{(l)} | \mathbf{z}^{\text{cat}-,(l)} \right)$;
      - $\ell_{y^*}^{(l)} \leftarrow \ell_{y^*}^{(l)} \left(\hat{f}_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}}} \left(\mathbf{z}^{\text{ncat},(l)} | \mathbf{z}^{\text{cat},(l)} \right) \right)^{-1}$; see (13)
    - **else**
      - $\ell_{y^*}^{(l)} \leftarrow \hat{f}_{\mathbf{Z}^{\text{ncat}}|\mathbf{Z}^{\text{cat}}} \left(\mathbf{z}^{\text{ncat},(l)} | \mathbf{z}^{\text{cat},(l)} \right) \left(\hat{f}_{\mathbf{Z}^{\text{ncat}-}|\mathbf{Z}^{\text{cat}}} \left(\mathbf{z}^{\text{ncat}-,(l)} | \mathbf{z}^{\text{cat},(l)} \right) \right)^{-1}$; see (12)
    - **end**
  - **end**
- **end**
- **return** $\left(\hat{\ell}_{y^*}^{(m_1)}, \ldots, \hat{\ell}_{y^*}^{(m_m)} \right)$

Algorithm 5: predictWithPCBN$(\mathcal{D}, \mathcal{B})$

Input: data $\mathcal{D} = \left\{ \left(x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M} \right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ indices where m_1 and m_m are the first and last index of $\mathcal{M}$ respectively and PCBN $\mathcal{B} = (\mathcal{G}, \Theta)$ with DAG $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with set of nodes $\mathcal{N} \subseteq \{0, \ldots, n\}$ and set of arcs $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$, and set of estimated distributions Θ containing the range $\mathcal{K}$ of Y as it should contain the probabilities $\mathrm{P}[Y = y]$ for all possible y

Result: set with $k = |\mathcal{K}|$ $|\mathcal{M}|$-tuples with outcomes $\hat{p}_{y^*}^{(m_1)}, \ldots, \hat{p}_{y^*}^{(m_m)}$ with $y^* \in \mathcal{K}$. Normalising the outcomes over per $l \in \mathcal{M}$

Procedure: predictWithPCBN$(\mathcal{D}, \mathcal{B})$

- $\mathcal{N}^{\text{ch}} \leftarrow \{j : (i, j) \in \mathcal{A}, i \in \mathcal{N}\} \cup \{0\}$; set containing Y and all children with at least one parent
- **foreach** $y^* \in \mathcal{K}$ **do**
  - $\left(\hat{p}_{y^*}^{(m_1)}, \ldots, \hat{p}_{y^*}^{(m_m)} \right) \leftarrow 1$; preallocation of evaluated $\hat{f}_{Y,\mathbf{X}} \left(y^*, \mathbf{x}^{(l)} \right)$ for $l \in \mathcal{M}$
  - **foreach** $j \in \mathcal{N}^{\text{ch}}$ **do**
    - $\mathcal{N}_{\text{pa}(j)} \leftarrow \{p : (p, j) \in \mathcal{A}\}$; set of all parents of j
    - $\left(\hat{\ell}_{y^*}^{(m_1)}, \ldots, \hat{\ell}_{y^*}^{(m_m)} \right) \leftarrow$ predictLocal$\left(\mathcal{D}, j, \mathcal{N}_{\text{pa}(j)}, \Theta, y^* \right)$;
    - **foreach** $l \in \mathcal{M}$ **do**
      - $\hat{p}_{y^*}^{(l)} \leftarrow \hat{p}_{y^*}^{(l)} \hat{\ell}_{y^*}^{(l)}$; multiply with evaluated local probability function given its parents
    - **end**
  - **end**
- **end**

return $\left\{ \left(\hat{p}_{y^*}^{(m_1)}, \ldots, \hat{p}_{y^*}^{(m_m)} \right) : y^* \in \mathcal{K} \right\}$

5.5 Structure Learning

Unless provided by previous research or expert knowledge, the underlying graph of a Bayesian network is unknown and has to be inferred from the data. We developed a greedy algorithm that starts with an empty network and adds arcs iteratively. The algorithm consists of the following six steps:

1. Do a baseline estimation of $f_Y(y)$ and calculate the score. This serves as the benchmark score
2. Select an arc out of all possible arcs, favouring arcs connected to Y
3. Estimate a PCBN with the newly added arc on the *estimation data*
4. Calculate the score on the *validation data*

5. Assess the addition and update the sets of arcs, nodes and estimated probability functions appropriately

6. Go to 2 if the list of possible child nodes $\mathcal{N}^*$ is not empty and the maximum number of nodes $n^{\max}$ has not been reached

The greedy algorithm is given by Algorithm 6, called GREEDYPCBN. Section 5.5.1 to 5.5.4 explain all steps in detail. An implementation that calls GREEDYPCBN several times, called the copula spider, is introduced in Section 5.6.

5.5.1 Baseline Estimation

We start with an empty network and get the benchmark estimates in line 2 of GREEDYPCBN by calling GETBASEVALUES. These estimates are simply $\hat{f}_Y(y)$ for all $y \in \mathcal{K}$. Thus, we only use the dependent variable Y. GETBASEVALUES returns seven objects that will be used throughout the algorithm: $\mathcal{N}^*$, the set of nodes that can receive parents; $\hat{\mathcal{N}}$, the current set of nodes; $\hat{\mathcal{A}}$, the current set of arcs; $\mathcal{A}^{\text{unev}}$, the set of unevaluated arcs; $\mathcal{A}^{\text{forb}}$, the set of forbidden arcs; $\hat{\Theta}$, the set of estimated probability functions; and s, the score associated with the PCBN.

5.5.2 Selecting an Arc

Using the set of nodes that can receive parents and the set of unevaluated arcs, we can select an arc (i, j) that will be added to the network. We select a child j and a parent i in lines 2 and 3 of GETARC. We return $(\varnothing, j)$ if the child j cannot receive a possible parent. Then, j will be removed from the list of nodes $\mathcal{N}^*$ as seen in line 6 of GREEDYPCBN. We also return $(0, j)$ or $(i, 0)$ if either the child j or the parent i corresponds to the dependent variable Y.

If $i \neq 0$, $j \neq 0$ and $i \neq \varnothing$, then we return (i, j) when all $(0, i), (i, 0), (0, j)$ and $(j, 0)$ have been assessed. If $(0, j)$ has not been evaluated, then we evaluate this addition instead. If this addition does not turn out to be successful, then we remove j from the set of nodes that can receive parents. Therefore, we do not have to check if $(j, 0)$ has been evaluated. If $(0, j)$ has been evaluated but $(0, i)$ has not, then we return $(0, i)$ instead. If both $(0, j)$ and $(0, i)$ have been evaluated whereas $(i, 0)$ has not, then we evaluate $(i, 0)$ instead. Otherwise, we evaluate (i, j).

Algorithm 6: greedyPCBN $\left(\mathcal{D}^{\text{est}}, \mathcal{D}^{\text{val}}, \mathcal{N}, \mathcal{C}, \texttt{score}, n^{\max}, n^{\text{pamax}}\right)$

Input: estimation data $\mathcal{D}^{\text{est}} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}^{\text{est}}\right\}$ and validation data $\mathcal{D}^{\text{val}} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}^{\text{val}}\right\}$ where $\mathcal{M}^{\text{est}}$ and $\mathcal{M}^{\text{val}}$ are sets of $|\mathcal{M}^{\text{est}}| \in \mathbb{N}^+$ and $\left|\mathcal{M}^{\text{val}}\right| \in \mathbb{N}^+$ unique indices respectively, set of nodes to use $\mathcal{N} \subseteq \{1, \ldots, n\}$, set of copula names $\mathcal{C}$, a scoring function `score`, maximum number of nodes $n^{\max} \in \mathbb{N}^+$ and maximum number of parents $n^{\text{pamax}} \in \mathbb{N}$

Result: PCBN $\hat{\mathcal{B}} = \left(\hat{\mathcal{G}}, \hat{\Theta}\right)$

Procedure: greedyPCBN$\left(\mathcal{D}^{\text{est}}, \mathcal{D}^{\text{val}}, \mathcal{N}, \mathcal{C}, \texttt{score}, n^{\max}, n^{\text{pamax}}\right)$

- $\left(\mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}, \hat{\Theta}, s^{\text{best}}\right) \leftarrow$ getBaseValues$(\mathcal{D}^{\text{est}}, \mathcal{N})$;
- **while** $\left|\hat{\mathcal{N}}\right| > n^{\max}$ **and** $\mathcal{N}^* \neq \emptyset$ **do**
  - $(i, j) \leftarrow$ getArc$(\mathcal{N}^*, \mathcal{A}^{\text{unev}})$;
  - **if** $j = \varnothing$ **then**
    - $\mathcal{N}^* \leftarrow \mathcal{N}^* \setminus \{i\}$;
    - **next**
  - **end**
  - $\mathcal{N}_{\text{pa}(j)} \leftarrow \left\{p : (p, j) \in \hat{\mathcal{A}}\right\} \cup \{i\}$; select all parents of j
  - $\hat{\Theta}^{\text{new}} \leftarrow$ estimateLocal$\left(\mathcal{D}^{\text{est}}, j, \mathcal{N}_{\text{pa}(j)}, \mathcal{C}\right) \cup \hat{\Theta}$;
  - $\left(\hat{\mathcal{N}}^{\text{new}}, \hat{\mathcal{A}}^{\text{new}}\right) \leftarrow \left(\hat{\mathcal{N}} \cup \{i, j\}, \hat{\mathcal{A}} \cup \{(i, j)\}\right)$;
  - $s^{\text{new}} \leftarrow \texttt{score}\left(\mathcal{D}^{\text{val}}, \left(\left(\hat{\mathcal{N}}^{\text{new}}, \hat{\mathcal{A}}^{\text{new}}\right), \hat{\Theta}^{\text{new}}\right)\right)$;
  - **if** $s^{\text{new}} > s^{\text{best}}$ **then**
    - $\left(\hat{\mathcal{N}}, \hat{\mathcal{A}}, \hat{\Theta}, s^{\text{best}}\right) \leftarrow \left(\hat{\mathcal{N}}^{\text{new}}, \hat{\mathcal{A}}^{\text{new}}, \hat{\Theta}^{\text{new}}, s^{\text{new}}\right)$;
    - $\left(\mathcal{N}^*, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}\right) \leftarrow$goodUpdate$\left(i, j, \mathcal{N}, \mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}, n^{\text{pamax}}\right)$
  - **else**
    - $\left(\mathcal{N}^*, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}\right) \leftarrow$badUpdate$\left(i, j, \mathcal{N}, \mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}\right)$
  - **end**
- **end**
- $\hat{\mathcal{G}} \leftarrow \left(\hat{\mathcal{N}}, \hat{\mathcal{A}}\right)$;
- $\hat{\mathcal{B}} \leftarrow \left(\hat{\mathcal{G}}, \hat{\Theta}\right)$;

return $\hat{\mathcal{B}}$

Algorithm 7: getBaseValues($\mathcal{D}, \mathcal{N}, \texttt{score}, n^{\max}, n^{\text{pamax}}$)

Input: estimation data $\mathcal{D} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}\right\}$ where $\mathcal{M}$ is of $|\mathcal{M}| \in \mathbb{N}^+$ unique indices, set of nodes to use $\mathcal{N} \subseteq \{1, \ldots, n\}$, scoring function `score`, maximum number of nodes $n^{\max}$ and maximum number of parents per node n^{pamax}

Result: set of nodes $\mathcal{N}^* \leftarrow \mathcal{N} \cup \{0\}$, set of current nodes $\hat{\mathcal{N}}$, set of current arcs $\hat{\mathcal{A}}$, set of all unevaluated arcs $\mathcal{A}^{\text{unev}}$, set of all forbidden arcs $\mathcal{A}^{\text{forb}}$, set of estimated probability functions $\hat{\Theta}$ and score s

Procedure: getBaseValues($\mathcal{D}, \mathcal{N}, \texttt{score}, n^{\max}, n^{\text{pamax}}$)

$\left(\mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}\right) \leftarrow (\mathcal{N} \cup \{0\}, \{0\}, \emptyset)$;

$\left(\mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}\right) \leftarrow \left(\{(i,j) : i, j \in \mathcal{N}, i \neq j\}, \{(i,i) : i \in \mathcal{N}\}\right)$;

$\hat{\Theta} \leftarrow \left\{\hat{f}_Y(y) : y \in \text{Ran}(Y)\right\}$;

$s \leftarrow \texttt{score}\left(\mathcal{D}, \left(\left(\hat{\mathcal{N}}, \hat{\mathcal{A}}\right), \hat{\Theta}\right)\right)$;

if $n^{\max} = 1$ **or** $n^{\text{pamax}} = 0$ **then**

  $\mathcal{N}^* \leftarrow \emptyset$;

end

return $\left(\mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}, \hat{\Theta}, s\right)$

Note that we *do not* have to check if arc $(j, 0)$ has been evaluated, whereas we *do* need to check if arc $(i, 0)$ has been evaluated. This is because checked if $(0, j)$ had been evaluated. If it has, it should mean that Y is a parent of node j. Otherwise j should not have been in the node set $\mathcal{N}$. This line of reasoning does not apply to i. After all, i does not come the node set $\mathcal{N}$.

It is important to realise that getArc is more likely to return arcs from Y to a random variable than vice versa. We do this to ensure that non-complex PCBNs are more likely to be evaluated first. For instance, if Y already had a continuous parent X_1, then adding another continuous parent X_2 means that we have to use copulas, estimated CDFs and pdfs to model the dependencies. Adding Y to X_2 instead means that we only make use of kernel density estimation.

Algorithm 8: GETARC($\mathcal{N}, \mathcal{A}$)

Data: set of nodes $\mathcal{N} \subseteq \{0, \ldots, n\}$ set of arcs
$\mathcal{A} \subseteq \{(i,j) : (i,j) \in \{0,\ldots,n\} \times \{0,\ldots,n\}, i \neq j\}$

Result: selected arc (i,j) with parent $i \in \{0,\ldots,n\}$ or $i = \varnothing$ and child $j \in \mathcal{N}$

1 **Procedure:** GETARC($\mathcal{N}, \mathcal{A}$)
2 $\quad j \leftarrow j \in \mathcal{N}$; select a child
3 $\quad i \leftarrow i \in \mathcal{N} : (i,j) \in \mathcal{A}$; select a parent
4 $\quad$ **if** $i = \varnothing$ **or** $i = 0$ **or** $j = 0$ **then**
5 $\quad\quad$ **return** (i,j)
6 $\quad$ **else if** $(0,j) \in \mathcal{A}$ **then**
7 $\quad\quad$ $i \leftarrow 0$; relationship X_j and Y unknown, estimate X_j with parent Y instead
8 $\quad$ **else if** $(0,i) \in \mathcal{A}$ **then**
9 $\quad\quad$ $j \leftarrow i$; the parent becomes the child
10 $\quad\quad$ $i \leftarrow 0$; estimate the relationship of X_i and Y instead
11 $\quad$ **else if** $(i,0) \in \mathcal{A}$ **then**
12 $\quad\quad$ $j \leftarrow 0$; estimate the relationship of X_i and Y instead
13 **return** (i,j)

5.5.3 Evaluating Selection

The third step of the algorithm is to evaluate the addition of (i,j). First, we estimate the new set of probability functions as seen in line 10 of GREEDYPCBN by calling ESTIMATE-LOCAL on j and its parents, including new parent i. Then, we calculate the new score s^{new} on the validation data $\mathcal{D}^{\text{val}}$ using the new PCBN consisting of the estimated set of nodes $\hat{\mathcal{N}}^{\text{new}}$, estimated set of arcs $\hat{\mathcal{A}}^{\text{new}}$ and estimated set of probability functions $\hat{\Theta}^{\text{new}}$. The algorithm does not depend on a particular scoring function. An example for `score` is the 0-1-score function given by Algorithm 9. Using 0-1SCORE is equivalent to adding arcs based on validation accuracy or hit rate on the validation data. It also equivalent to using a 0-1-loss function multiplied by -1, as $[\hat{y} = y] = 1 - [\hat{y} \neq y]$. A good score function produces a higher score s when the prediction is deemed better. This, of course, depends on the prediction task. If a certain misclassification is deemed worse than other misclassifications, then `score` should reflect that.

Calculating the score on validation data, rather than estimation data, aims to prevent overfitting. However, GREEDYPCBN does not require nor assume that $\mathcal{D}^{\text{val}} \neq \mathcal{D}^{\text{est}}$. Overfitting can also be prevented by choosing penalised scores.

Algorithm 9: 0-1Score$(\mathcal{D}, \mathcal{B})$

Input: data $\mathcal{D} = \left\{ \left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M} \right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ unique indices where m_1 and m_m are the first and last index of $\mathcal{M}$ respectively and PCBN $\mathcal{B} = (\mathcal{G}, \Theta)$ with DAG $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with set of nodes $\mathcal{N} \subseteq \{0, \ldots, n\}$ and set of arcs $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$, and set of estimated distributions Θ containing the range $\mathcal{K}$ of Y as it should contain the probabilities $\mathrm{P}[Y = y]$ for all possible y

Result: score s

1 **Procedure:** 0-1Score$(\mathcal{D}, \mathcal{B})$

2 $\quad \mathcal{D}^{\text{val}} \leftarrow \left\{ \left(x_1^{(l)}, \ldots, x_n^{(l)} \right) : l \in \mathcal{M} \right\}$; remove Y

3 $\quad \left\{ \left(\hat{p}_{y^*}^{(m_1)}, \ldots, \hat{p}_{y^*}^{(m_m)} \right) : y^* \in \mathcal{K} \right\} \leftarrow$ predictWithPCBN$\left(\mathcal{D}^{\text{val}}, \mathcal{B}\right)$; get predictions

4 $\quad s \leftarrow 0$ **foreach** $l \in \mathcal{M}$ **do**

5 $\quad\quad \hat{y}^{(l)} \leftarrow \hat{y}^{(l)} \in \arg\max_{y* \in \mathcal{K}} \hat{p}_{y^*}^{(l)}$; see (1)

6 $\quad\quad s \leftarrow s + \left[\hat{y}^{(l)} = y^{(l)}\right]$; 1 if we made a good prediction, zero otherwise: score behaves similar to a 0-1-loss function multiplied by -1

7 $\quad$ **end**

8 **return** s

5.5.4 Implementing Update

The final step of greedyPCBN is to update the network based on our new score s^{new} as seen in line 14 to 19. We use goodUpdate and badUpdate if the new score is greater or worse than the current score of the network respectively.

The first step of goodUpdate is to remove (i, j) from the set of unevaluated arcs, as it has just been evaluated. Then, we must ensure that child j cannot become a parent of certain nodes. Specifically, if i cannot become a parent of nodes $p_1, \ldots, p_n$, then j should also not be allowed to become a parent of these nodes. This means that j cannot become a parent of i, as i cannot become a parent of i.

If the child j is the dependent variable, then its parent i should not be allowed to get any parents itself. Then, we also remove i from the set of possible children $\mathcal{N}^*$. This has been implemented in line 7 to 9. Lastly, we remove j from the set of possible children if its number of parents exceed the specified maximum n^{pamax}.

Algorithm 10: goodUpdate$\left(i, j, \mathcal{N}, \mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}, n^{\text{pamax}}\right)$

Input: indices $i, j \in \{0, \ldots, n\}$, set of nodes $\mathcal{N} \subseteq \{0, \ldots, n\} : i, j \in \mathcal{N}$, sets of nodes $\mathcal{N}^*, \hat{\mathcal{N}} \subseteq \mathcal{N}$, sets of arcs $\hat{\mathcal{A}}, \mathcal{A}^{\text{unev}} \subset \mathcal{N} \times \mathcal{N}$, set of arcs $\mathcal{A}^{\text{forb}} \subseteq \mathcal{N} \times \mathcal{N}$ and maximum number of nodes $n^{\text{pamax}} \in \mathbb{N}$

Result: updated $\mathcal{N}^*, \mathcal{A}^{\text{unev}}$ and $\mathcal{A}^{\text{forb}}$

Procedure: goodUpdate$\left(i, j, \mathcal{N}, \mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}, n^{\text{pamax}}\right)$

- $\mathcal{N} \leftarrow \mathcal{N} \cup \{0\}$; all possible nodes
- $\mathcal{A}^{\text{unev}} \leftarrow \mathcal{A}^{\text{unev}} \setminus \{(i, j)\}$;
- $\mathcal{A}^{\text{allf}} \leftarrow \left\{(j, p) : (i, p) \in \mathcal{A}^{\text{forb}}, p \in \mathcal{N}\right\}$; if i is not allowed to go to p then j should also not be allowed to go to p, note that $(i, i) \in \mathcal{A}^{\text{forb}}$
- $\left(\mathcal{A}^{\text{forb}}, \mathcal{A}^{\text{unev}}\right) \leftarrow \left(\mathcal{A}^{\text{forb}} \cup \mathcal{A}^{\text{allf}}, \mathcal{A}^{\text{unev}} \setminus \mathcal{A}^{\text{allf}}\right)$;
- **if** $j = 0$ **then**
  - $\mathcal{A}^{\text{allf}} \leftarrow \{(p, i) : p \in \mathcal{N}\}$; X_i goes to Y, meaning X_i should not get any parents
  - $\mathcal{N}^* \leftarrow \mathcal{N}^* \setminus \{i\}$; remove i from possible set of children
  - $\left(\mathcal{A}^{\text{forb}}, \mathcal{A}^{\text{unev}}\right) \leftarrow \left(\mathcal{A}^{\text{forb}} \cup \mathcal{A}^{\text{allf}}, \mathcal{A}^{\text{unev}} \setminus \mathcal{A}^{\text{allf}}\right)$;
- **end**
- $\mathcal{A}_{\text{to}(j)} \leftarrow \left\{(p, j) : (p, j) \in \hat{\mathcal{A}}, p \in \mathcal{N}\right\}$; set of arcs to j
- **if** $\left|\mathcal{A}_{\text{to}(j)}\right| = n^{\text{pamax}}$ **then**
  - $\mathcal{A}^{\text{unev}} \leftarrow \mathcal{A}^{\text{unev}} \setminus \{(p, j) : p \in \mathcal{N}\}$; remove all unevaluated $p \rightarrow j$
  - $\mathcal{A}^{\text{forb}} \leftarrow \mathcal{A}^{\text{forb}} \cup \left(\{(p, j) : p \in \mathcal{N}\} \setminus \mathcal{A}_{\text{to}(j)}\right)$; forbid influx from other random variables to j
  - $\mathcal{N}^* \leftarrow \mathcal{N}^* \setminus \{j\}$;
- **end**

return $\left(\mathcal{N}^*, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}\right)$

The first step of badUpdate is to remove (i, j) from the set of unevaluated arcs, identical to the first step of goodUpdate. If the parent X_i is in fact Y, then we remove j from the set of possible child nodes. We do this as adding $Y \rightarrow X_j$ does not increase the score, meaning that X_j cannot receive any parents. If X_j is in fact Y and if the arc (j, i) has been evaluated, then we remove i as neither $X_i \rightarrow Y$ nor $Y \rightarrow X_i$ will be used. Lastly, we remove j from the set of possible child nodes when there is not a single arc (i, j) left to evaluate.

An arc (i, j) that does not increase the score s may at some point in future, when other random variables have been added, increase the score. This might happen when random variables jointly depend on Y. A possible solution is to make use of a TABU list to make certain arcs unevaluated. Another solution, that is not guaranteed to add arc (i, j), is

to rerun the procedure. It is possible that (i, j) increases the score when other random variables have been added.

Algorithm 11: BADUPDATE$\Big(i, j, \mathcal{N}, \mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}\Big)$

Input: indices $i, j \in \{0, \ldots, n\}$, set of nodes $\mathcal{N} \subseteq \{0, \ldots, n\} : i, j \in \mathcal{N}$, sets of nodes $\mathcal{N}^*, \hat{\mathcal{N}} \subseteq \mathcal{N}$, set of arcs $\hat{\mathcal{A}}, \mathcal{A}^{\text{unev}} \subset \mathcal{N} \times \mathcal{N}$ and set of arcs $\mathcal{A}^{\text{forb}} \subseteq \mathcal{N} \times \mathcal{N}$

Result: updated $\mathcal{N}^*, \mathcal{A}^{\text{unev}}$ and $\mathcal{A}^{\text{forb}}$

Procedure: BADUPDATE$\Big(i, j, \mathcal{N}, \mathcal{N}^*, \hat{\mathcal{N}}, \hat{\mathcal{A}}, \mathcal{A}^{\textbf{unev}}, \mathcal{A}^{\textbf{forb}}\Big)$

- $\mathcal{N} \leftarrow \mathcal{N} \cup \{0\}$; all possible nodes
- $\mathcal{A}^{\text{unev}} \leftarrow \mathcal{A}^{\text{unev}} \setminus \{(i, j)\}$;
- **if** $i = 0$ **then**
  - $\mathcal{N}^* \leftarrow \mathcal{N}^* \setminus \{j\}$; $Y \rightarrow X_j$ was unsuccesful, meaning X_j cannot become a child of any variable
  - $\mathcal{A}^{\text{allf}} \leftarrow \{(p, j) : p \in \mathcal{N}\}$; X_j should not get any parents
  - $\big(\mathcal{A}^{\text{forb}}, \mathcal{A}^{\text{unev}}\big) \leftarrow \big(\mathcal{A}^{\text{forb}} \cup \mathcal{A}^{\text{allf}}, \mathcal{A}^{\text{unev}} \setminus \mathcal{A}^{\text{allf}}\big)$;
- **else if** $j = 0$ **and** $(j, i) \notin \mathcal{A}^{\text{unev}}$ **then**
  - $\mathcal{N}^* \leftarrow \mathcal{N}^* \setminus \{i\}$; $X_i \rightarrow Y$ was unsuccesful and $Y \rightarrow X_i$ has been checked already, meaning X_i cannot become a child of any variable
  - $\mathcal{A}^{\text{allf}} \leftarrow \{(p, i) : p \in \mathcal{N}\}$; X_i should not get any parents
  - $\big(\mathcal{A}^{\text{forb}}, \mathcal{A}^{\text{unev}}\big) \leftarrow \big(\mathcal{A}^{\text{forb}} \cup \mathcal{A}^{\text{allf}}, \mathcal{A}^{\text{unev}} \setminus \mathcal{A}^{\text{allf}}\big)$;
- $\mathcal{A}^{\text{unev}}_{\text{to}(j)} \leftarrow \{(p, j) : (p, j) \in \mathcal{A}^{\text{unev}}, p \in \mathcal{N}\}$; set of unevaluated arcs to j
- **if** $\left|\mathcal{A}^{\text{unev}}_{\text{to}(j)}\right| = 0$ **then**
  - $\mathcal{N}^* \leftarrow \mathcal{N}^* \setminus \{j\}$; make sure j cannot be selected as a child
- **end**

return $\big(\mathcal{N}^*, \mathcal{A}^{\text{unev}}, \mathcal{A}^{\text{forb}}\big)$

5.6 Copula Spider

Applying the greedy algorithm GREEDYPCBN merely once has two major drawbacks. First of all, the procedure might get stuck in local optima due to the greedy nature of the algorithm. This means that the returned PCBN does not necessarily have good predictive performance. Secondly, the algorithm should ideally not depend on a particular choice for the validation data. A solution to these problems is to run GREEDYPCBN several times while simultaneously varying the validation data, which we implemented in Algorithm 12. We call this algorithm COPULASPIDER, as it creates several networks which can be seen as elaborate webs which all connect the variables to one another.

Algorithm 12: copulaSpider $\left(\mathcal{D}, n^{\text{web}}, n^{\text{run}}, \texttt{split}, \mathcal{N}, \mathcal{C}, \texttt{score}, n^{\max}, n^{\text{pamax}}\right)$

Input: data $\mathcal{D} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}\right\}$ where $\mathcal{M}$ is a set of $|\mathcal{M}| \in \mathbb{N}^+$ unique indices, number of PCBNs to create $n^{\text{web}} \in \mathbb{N}^+$, number of reruns per PCBN $n^{\text{run}} \in \mathbb{N}^+$, function `create`, parameters $\mathcal{N}, \mathcal{C}$, `score`, $n^{\max}$ and n^{pamax} as given in greedyPCBN

Result: n^{web} PCBNs $\hat{\mathcal{B}}_1, \ldots, \hat{\mathcal{B}}_{n^{\text{web}}}$

Procedure: copulaSpider$\left(\mathcal{D}, n^{\text{web}}, n^{\text{check}}, \texttt{split}, \mathcal{N}, \mathcal{C}, \texttt{score}, n^{\max}, n^{\text{pamax}}\right)$

- $\left(\hat{\mathcal{B}}_1, \ldots, \hat{\mathcal{B}}_{n^{\text{web}}}\right) \leftarrow (\emptyset, \ldots, \emptyset)$;
- **foreach** $i \in \left\{1, \ldots, n^{\text{web}}\right\}$ **do**
  - $\left(\mathcal{D}^{\text{est}}, \mathcal{D}^{\text{val}}\right) \leftarrow \texttt{create}(\mathcal{D}, \ldots)$;
  - $s^{\text{best}} \leftarrow 0$;
  - **foreach** $j \in \{1, \ldots, n^{\text{run}}\}$ **do**
    - $\hat{\mathcal{B}}_{i,j} \leftarrow$greedyPCBN$\left(\mathcal{D}^{\text{est}}, \mathcal{D}^{\text{val}}, \mathcal{N}, \mathcal{C}, \texttt{score}, n^{\max}, n^{\text{pamax}}\right)$;
    - $s^{\text{new}} \leftarrow \texttt{score}\left(\mathcal{D}^{\text{val}}, \hat{\mathcal{B}}_{i,j}\right)$;
    - **if** $s^{\text{new}} >^{\text{best}}$ **then**
      - $\left(\hat{\mathcal{B}}_i, s^{\text{best}}\right) \leftarrow \left(\hat{\mathcal{B}}_{i,j}, s^{\text{new}}\right)$; choose the PCBN with the highest validation accuracy
    - **end**
  - **end**
- **end**

return $\hat{\mathcal{B}}_1, \ldots, \hat{\mathcal{B}}_{n^{\text{web}}}$

The spider requires some function `create` that creates estimation and validation data. If we want $\mathcal{D}^{\text{est}} \cap \mathcal{D}^{\text{val}} = \emptyset$, then we could construct the estimation and validation as $\mathcal{D}^{\text{est}} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M} \setminus \mathcal{M}_i\right\}$ and $\mathcal{D}^{\text{val}} = \left\{\left(y^{(l)}, x_1^{(l)}, \ldots, x_n^{(l)}\right) : l \in \mathcal{M}_i\right\}$ respectively, for each $i \in \left\{1, \ldots, n^{\text{web}}\right\}$. Without loss of generality, let $\mathcal{M} = \{1, \ldots, d\}$ where $d \in \mathbb{N}^+$. Then, using $\mathcal{M}_i = \{\lfloor (i-1)m \rfloor + 1, \ldots, \lfloor im \rfloor\}$ where $m = d\left(n^{\text{web}}\right)^{-1}$ means we construct $\mathcal{D}^{\text{est}}$ and $\mathcal{D}^{\text{val}}$ as if we are doing cross-validation. For instance, if $d = 1000$ and $n^{\text{web}} = 4$, then we have $\mathcal{M}_1 = \{1, \ldots, 250\}$, $\mathcal{M}_2 = \{251, \ldots, 500\}$, $\mathcal{M}_3 = \{501, \ldots, 750\}$ and , $\mathcal{M}_4 = \{751, \ldots, 1000\}$.

6 Simulation Design

The newly introduced predictive copula Bayesian network (PCBN) allows ordinal discrete random variables to be modelled *explicitly*. My simulation design aims to illustrate the performance gains that can be reached by modelling ordinal discrete variables explicitly.

The simulation algorithm is given by Algorithm 13. We create one continuous explanatory variable X_1, two ordinal discrete random variables X_2 and X_3 and a categorical dependent variable Y with $\text{Ran}(Y) = \{1, 2, 3, 4\}$. Two thirds of the simulated realisations belong to in-sample data $\mathcal{D}^{\text{in}}$ and one third belongs to the out-sample data $\mathcal{D}^{\text{out}}$.

We simulate X_1 and X_2 so that they follow a bivariate Clayton distribution with parameter $\theta = 2\tau\,(1-\tau)^{-1}$ in line 4 to 7. Note that τ can be interpreted as Kendall's tau. To simulate values from the Clayton copula, we use the conditional distribution method as described in Embrechts, Lindskog, and McNeil (2001). We also simulate X_2 and X_3 from a bivariate Clayton copula with the same parameter θ. Therefore, X_2 is correlated with both X_1 and X_3. More information about the Clayton copula is included in Appendix A.4.1.

Table 1: Parameter settings for each explanatory variable X_1, X_2 and X_3 per category of Y belonging to the simulation algorithm given by Algorithm 13. The table should be read as follows: $1 - X_1|Y = 1 \sim \text{Weibull}(1.0, 1.5)$ and $X_1|Y = 2 \sim \text{Weibull}(1.5, 1.0)$

	Variable	1	2	3	4
Reflected	X_1	yes			yes
	X_2			yes	yes
	X_3	yes		yes	
Scale	X_1	1.0	1.5	1.0	1.5
Shape	X_1	1.5	2.0	1.5	2.0

We transform some $X_i|Y = y$ to $1 - X_i|Y = y$ for specific combinations of i and y. That way, we have both positive and negative dependencies between variables per y. We transform all (possibly reflected) X_1 to a Weibull scale as shown in Table 1. We transform X_2 and X_3 to $\text{Ran}(X_2) = \text{Ran}(X_3) = \{1, 2, 3, 4, 5\}$ as seen in line 16 and 17 of Algorithm 13.

Algorithm 13: Simulation design

Input: number of data points to simulate $d \in \{4, 5, \ldots\}$ and correlation measure $\tau \in (0, 1)$

Result: in-sample data $\mathcal{D}^{\text{in}}$ and out-sample data $\mathcal{D}^{\text{out}}$

Procedure:

$\theta \leftarrow 2\tau\left(1-\tau\right)^{-1}$; calculates θ for bivariate Clayton using τ as Kendall's tau

$c \leftarrow \left\lfloor \frac{d}{4} \right\rfloor$; minimum number of observations per category

$\left(z^{(1)}, \ldots, z^{(2d)}\right) \leftarrow z^{(1)}, \ldots, z^{(2d)} \in (0, 1)$; generate $2d$ uniform values

$\left(x_1^{(1)}, \ldots, x_1^{(d)}\right) \leftarrow x_1^{(1)}, \ldots, x_1^{(d)} \in (0, 1)$; generate d uniform values

$\left(x_2^{(1)}, \ldots, x_2^{(d)}\right) \leftarrow \left(\left(\left(x_1^{(l)}\right)^{-\theta}\left(z^{(l)}\right)^{\theta(1+\theta)^{-1}-1}+1\right)^{-\theta^{-1}}\right)_{l\in\{1,\ldots,d\}}$; now each pair $x_1^{(l)}$ and $x_2^{(l)}$ d comes from a bivariate Clayton distribution with parameter θ, see e.g. Embrechts et al. (2001)

$\left(x_3^{(1)}, \ldots, x_3^{(d)}\right) \leftarrow \left(\left(\left(x_2^{(l)}\right)^{-\theta}\left(z^{(l+d)}\right)^{\theta(1+\theta)^{-1}-1}+1\right)^{-\theta^{-1}}\right)_{l\in\{1,\ldots,d\}}$;

$\left(x_1^{(1)}, \ldots, x_1^{(c)}\right) \leftarrow \left(\left(-\log x_1^{(1)}\right)^{\frac{2}{3}}\right)_{l\in\{1,\ldots,c\}}$; to Weibull

$\left(x_1^{(c+1)}, \ldots, x_1^{(2c)}\right) \leftarrow \left(\frac{3}{2}\left(-\log\left(1-x_1^{(1)}\right)\right)^{\frac{1}{2}}\right)_{l\in\{c+1,\ldots,2c\}}$;

$\left(x_1^{(2c+1)}, \ldots, x_1^{(3c)}\right) \leftarrow \left(\left(-\log\left(1-x_1^{(1)}\right)\right)^{\frac{2}{3}}\right)_{l\in\{2c+1,\ldots,3c\}}$;

$\left(x_1^{(3c+1)}, \ldots, x_1^{(d)}\right) \leftarrow \left(\frac{3}{2}\left(-\log x_1^{(1)}\right)^{\frac{1}{2}}\right)_{l\in\{3c+1,\ldots,d\}}$;

$\left(x_2^{(2c+1)}, \ldots, x_2^{(d)}\right) \leftarrow \left(1-x_2^{(l)}\right)_{l\in\{2c+1,\ldots,d\}}$; invert

$\left(x_3^{(1)}, \ldots, x_3^{(c)}, x_3^{(2c+1)}, \ldots, x_3^{(3c)}\right) \leftarrow \left(1-x_3^{(l)}\right)_{l\in\{1,\ldots,c,2c+1,\ldots,3c\}}$;

foreach $l \in \{1, \ldots, d\}$ **do**

$y^{(l)} \leftarrow 1^{[l\in\{1,\ldots,c\}]}2^{[l\in\{c+1,\ldots,2c\}]}3^{[l\in\{2c+1,\ldots,3c\}]}4^{[l\in\{3c,\ldots,d\}]}$; dependent variable

$x_2^{(l)} \leftarrow 1^{\left[x_2^{(l)}\leq 0.1\right]}2^{\left[x_2^{(l)}\in(0.1,0.25]\right]}3^{\left[x_2^{(l)}\in(0.25,0.45]\right]}4^{\left[x_2^{(l)}\in(0.45,0.75]\right]}5^{\left[x_2^{(l)}>0.75\right]}$;

$x_3^{(l)} \leftarrow 1^{\left[x_3^{(l)}\leq 0.1\right]}2^{\left[x_3^{(l)}\in(0.1,0.25]\right]}3^{\left[x_3^{(l)}\in(0.25,0.45]\right]}4^{\left[x_3^{(l)}\in(0.45,0.75]\right]}5^{\left[x_3^{(l)}>0.75\right]}$;

end

$\mathcal{M} \leftarrow \mathcal{M} \subset \{1, \ldots, d\} : |\mathcal{M}| = \left\lfloor \frac{2d}{3} \right\rfloor$;

$\mathcal{D}^{\text{in}} = \left\{\left(y^{(l)}, x_1^{(l)}, x_2^{(l)}, x_3^{(l)}\right) : l \in \mathcal{M}\right\}$; random selection of two thirds

$\mathcal{D}^{\text{out}} = \left\{\left(y^{(l)}, x_1^{(l)}, x_2^{(l)}, x_3^{(l)}\right) : l \in \{1, \ldots, d\} \setminus \mathcal{M}\right\}$;

return $\left(\mathcal{D}^{\text{in}}, \mathcal{D}^{\text{out}}\right)$

Table 1 shows that X_1 behaves differently for each possible $y \in \{1, 2, 3, 4\}$. This is not the case for either X_2 nor X_3. After all, $X_2|Y = 1$ and $X_2|Y = 2$ behave in an identical manner and cannot be distinguished from one another. This is also the case for $X_2|Y = 3$ compared to $X_2|Y = 4$, $X_3|Y = 1$ compared to $X_3|Y = 3$ and $X_3|Y = 2$ compared to $X_3|Y = 4$. Assessing X_2 and X_3 jointly allows one to identify Y more accurately.

Both $X_2|Y = 3$ and $X_3|Y = 3$ are reflected, meaning that have a positive Pearson correlation. However, $X_2|Y = 2$ and $X_3|Y = 2$ also exhibit positive dependence. The difference between the former and the latter pair can be seen when one examines tail dependence. As both X_2 and X_3 come from a bivariate Clayton distribution, $X_2|Y = 2$ and $X_3|Y = 2$ have lower tail dependence Nelsen (2007) which means that the reflected pair $X_2|Y = 3$ and $X_3|Y = 3$ has upper tail dependence.

As seen in line 16 and 17 of Algorithm 13, not all values of X_2 and X_3 have an equal chance of appearing. The proportion of categories of Y is evenly spread, however. Section 7 elaborates on the numerical results obtained from running this simulation procedure.

7 Numerical Results

This sections implements the simulation design described in Section 6 and shows the advantage of modelling ordinal discrete variables explicitly. We use the copulaSpider approach described in Section 5.6. Table 2 shows the cumulative distribution functions (CDFs) of the copulas that we use.

Table 2: Bivariate cumulative distribution functions $C_{U_1,U_2}(u_1, u_2|\theta)$ of the strict Clayton, strict reflected $R_{1,2}$ Clayton (also known as the survival Clayton copula), Frank and Plackett copula. The appendix contains derivations of their mixed derivatives

	CDF	Appendix
Clayton	$\left(u_1^{-\theta} + u_2^{-\theta} - 1\right)^{-\theta^{-1}}$	A.4.1
$R_{1,2}$ Clayton	$u_1 + u_2 - 1 + \left((1-u_1)^{-\theta} + (1-u_2)^{-\theta} - 1\right)^{-\theta^{-1}}$	-
Frank	$-\frac{1}{\theta}\log\left(1 + \frac{\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}{e^{-\theta}-1}\right)$	A.4.2
Plackett	$\frac{1+(\theta-1)(u_1+u_2)-\left((1+(\theta-1)(u_1+u_2))^2-4u_1u_2\theta(\theta-1)\right)^{\frac{1}{2}}}{2(\theta-1)}$	A.4.3

We use the term reflection rather than rotation due to multiple definitions of rotation that exist in the literature. Information regarding reflection of n-variate copulas, as well as the reflection operator, can be found in Appendix A.3. We provide and prove a general formula for any reflected n-variate copula, see Theorem A.3.4. The $R_{1,2}$ Clayton copula follows immediately from that theorem.

Both the Frank and the Plackett copula can model positive and negative dependence. They also exhibit radial symmetry (Nelsen, 2007), meaning that the $R_{1,2}$ reflection results in the same copula. Therefore, reflections of these copulas do not need to be considered. This is not the case for the Clayton copula, however. The Clayton copula has lower-tail dependence (Nelsen, 2007). This means that the $R_{1,2}$ reflection has upper-tail dependence and can be considered a distinct copula. Moreover, the strict Clayton copula cannot model negative dependence. Modelling negative dependence with the strict Clayton copula can be done by adding a rotation in one of its arguments. For instance, both the R_1 Clayton and R_2 Clayton copula can model negative dependence. A different specification of the Clayton copula allows one to model negative dependence in a limited manner. As this is restrictive, we do not consider this specification.

We implemented all algorithms in R (R Core Team, 2018). The key algorithm greedyPCBN requires a function `score`. We use `score` = 0-1Score, which predicts with the constructed PCBN per arc under consideration. This score function ensures that a higher score corresponds to a better predictive accuracy, without assuming that certain misclassifications are worse than others. A naïve or memory-efficient implementation would be to re-predict the total probabilities each time. However, this is unnecessary as many functions do not change per iteration of greedyPCBN. Therefore, we save all predicted values per PCBN while it is being constructed. Although this saves a lot of time, it is not memory efficient. However, we find that the memory in use never exceeded 100MB for the cases examined.

To estimate the densities and distributions, we use the np package in by Hayfield and Racine (2008) which implements estimation based on Li and Racine (2003). The np allows me to use the Gaussian and Aitchison kernel given by 17 and 18 respectively. The bandwidth is calculated by means of cross-validation. The np package also implements distribution estimation which we use to estimate CDFs. Although conditional density and distribution estimation appears in np, the run times are substantial. Due to the fact that greedyPCBN requires many densities and distributions to be estimated, we do not use these techniques.

The copula parameters are estimated with the method of Brent (1973). This is a derivative-free method that requires an interval. We use several start and end points and choose the value that maximises the log-likelihood function. Other optimisation methods, such as L-BFGS, were ill-behaved. This may happen when the copula likelihood function is very flat.

We use Algorithm 13 for two scenarios. For the first scenario, we simulate 25 data sets using $n = 3000$ and $\tau = 0.8$, meaning that each in-sample and out-sample data set consists of 2000 and 1000 realisations respectively. A τ of 0.8 means the variables are highly correlated. The data sets contain three explanatory variables X_1, X_2 and X_3 and one dependent variable Y that can attain four different values.

We use a simple logit specification with an intercept and individual-specific X_1, X_2 and

X_3 as a benchmark model, see Train (2009) for more information. We use the MLOGIT by Croissant (2018) which uses MAXLIK (Henningsen & Toomet, 2011) for its maximum-likelihood estimator. We estimate the logit model on in-sample data and use the estimated model to predict values of the out-sample data. Then, we calculate the accuracy on both the in-sample and out-sample data, where the accuracy is defined as the number of correctly classified observations divided by the total number of observations. The logit model has an in-sample accuracy of 0.34 with a standard deviation of 0.06 and an out-sample accuracy of 0.32 with a standard deviation of 0.06.

The PCBNs allows ordinal discrete variables to be modelled appropriately. That is, we do not have model the discrete variables as if they were continuous. We use the cross-validation COPULASPIDER as described in Section 5.6 with number of webs $n^{\text{web}} = 4$, number of runs per web $n^{\text{run}} = 3$, `score` = 0-1SCORE and maximum number of parents $n^{\text{pamax}} = 2$. We choose $n^{\text{pamax}} = 2$ as copula functions become more complicated when the number of random variables increases. Moreover, one-parameter copula functions become less appropriate. After all, there is only one parameter that models the dependency of these random variables.

The specification means that we partition each in-sample data set into an estimation set and a validation set four times. The estimation set is used for the estimation of the parameters, whereas the validation set is used to determine whether or not the network has predictive power. We construct a PCBN three times per estimation set and use the network with the highest accuracy on the validation set. Using this procedure, we end up with $n^{\text{web}} = 4$ different networks. We use two different strategies to determine how predictions are made. The first strategy is to *select the network with the highest in-sample accuracy*. The second strategy is to make predictions with each network and use *popularity voting* to make the final prediction. That is, we choose a value $y \in \{1, 2, 3, 4\}$, per realisation, that is most often predicted by the individual networks for that realisation.

Table 3: Discrepancy between the correctly-specified cross-validation copulaSpiders (Regular) and the spiders assuming that all variables are continuous (Continuous). This table shows the mean and standard deviation of in-sample and out-sample accuracy of the spiders using $n^{\text{web}} = 4$, $n^{\text{run}} = 3$, `score` = 0-1Score, $n^{\max} = \infty$, $n^{\text{pamax}} = 2$, the network with the highest in-sample accuracy is reported and used for the out-sample prediction, for 5 different copulas on 25 data sets created by Algorithm 13 using $n = 3000$ and $\tau = 0.8$. The names refer to the copula forms used, where R^* Clayton means that all reflections of the Clayton copula are used

		Clayton	$R_{1,2}$ Clayton	R^* Clayton	Frank	Plackett
Regular	in	0.80 (0.02)	0.79 (0.01)	0.80 (0.02)	0.81 (0.01)	0.81 (0.01)
	out	0.80 (0.02)	0.79 (0.02)	0.80 (0.02)	0.81 (0.02)	0.81 (0.02)
Continuous	in	0.78 (0.02)	0.74 (0.05)	0.78 (0.02)	0.79 (0.01)	0.77 (0.03)
	out	0.78 (0.02)	0.73 (0.05)	0.77 (0.03)	0.79 (0.02)	0.77 (0.03)

Table 3 shows the accuracies for the strict Clayton, strict reflected $R_{1,2}$ Clayton, Frank and Plackett copula spiders using the the networks with the highest in-sample accuracy. The results next to Regular indicate that correctly-specified spiders are used. The results next to Continuous indicate that the spider has modelled all discrete variables as if they were continuous. We also use a specification R^* that chooses, for each possible copula that may occur, the best reflection of the Clayton copula. Thus, we estimate all reflections and choose the reflection that maximises the likelihood score.

Table 3 clearly shows that the accuracies of the correctly-specified spiders are higher, although the differences are small. The average accuracies of the miss-specified Clayton copula are 0.02 percentage points lower. The differences vary across copulas, however. For instance, correctly specifying the Plackett copula leads to an average accuracy increase of about 0.04 percentage points. The accuracies of the correctly-specified copulas are similar, with the Frank and Plackett copula having the highest accuracy.

Table 4: Discrepancy between the correctly-specified cross-validation copulaSpiders (Regular) and the spiders assuming that all variables are continuous (Continuous). This table shows the mean and standard deviation of in-sample and out-sample accuracy of the spiders using $n^{\text{web}} = 4$, $n^{\text{run}} = 3$, `score` = 0-1Score, $n^{\max} = \infty$, $n^{\text{pamax}} = 2$, using popularity voting to determine the final prediction, for 5 different copulas on 25 data sets created by Algorithm 13 using $n = 3000$ and $\tau = 0.8$.

		Clayton	$R_{1,2}$ Clayton	R^* Clayton	Frank	Plackett
Regular	in	0.77 (0.07)	0.77 (0.04)	0.78 (0.03)	0.78 (0.04)	0.79 (0.04)
	out	0.76 (0.07)	0.77 (0.04)	0.78 (0.03)	0.77 (0.05)	0.78 (0.04)
Continuous	in	0.71 (0.08)	0.71 (0.07)	0.73 (0.07)	0.75 (0.06)	0.73 (0.06)
	out	0.71 (0.08)	0.71 (0.08)	0.73 (0.07)	0.75 (0.06)	0.73 (0.06)

Table 4 shows the accuracies for the same copulas using popularity voting. There are two major differences between Table 3 and 4. First, popularity voting seems to increase the standard deviation in this case. This happens when either one or more networks are significantly worse than the others. Increasing the number of reruns n^{run} does not necessarily mitigate this problem, as it could be that the estimation or validation sample is dissimilar from the in-sample and out-sample data. Secondly, the average accuracies are lower. This is due to some accuracies being significantly lower from the others. For instance, the average in-sample accuracy of the Clayton copula is 0.77, whereas its lowest in-sample accuracy is 0.55.

Due to the higher standard deviations, we cannot claim that the correctly-specified copulas have significantly higher average accuracies. However, the accuracies of miss-specified copulas compared to correctly-specified copulas are lower in actuality.

For the second scenario, we simulate 25 data sets using $n = 3000$ and $\tau = 0.4$. Using the logit model as a benchmark, we obtain an average in-sample accuracy of 0.34 with a standard deviation of 0.03 and an average out-sample accuracy of 0.32 with a standard deviation of 0.02. The logit has the same average accuracies in both scenarios, whereas the standard deviation is lower in the latter scenario. Intuitively, the logit model is hindered by strongly correlated explanatory variables.

We use the Plackett COPULASPIDER on the simulated data, similar to before. Table 5 shows that, compared to scenario 1, the spider is hindered by the lower correlation. This is not surprising, as the framework uses the correlation between random variables to make its predictions. In scenario two, the correlation between variables is more difficult to detect which makes the model perform worse. As before, the correctly-specified copula spider performs slightly better than the miss-specified counterpart. However, the results are very similar. In actuality, the correctly-specified Plackett spider almost always outperforms the miss-specified one.

Table 5: Mean and standard deviation of in-sample and out-sample accuracy of the correctly-specified cross-validation Plackett COPULASPIDERS (Reg.), the spiders assuming that all variables are continuous (Cont.) using $n^{\text{web}} = 4$, $n^{\text{run}} = 3$, `score` = 0-1SCORE, $n^{\max} = \infty$, $n^{\text{pamax}} = 2$. The second and fourth spider (pop) make use of popularity voting to determine the final prediction, whereas the other spiders use the web with the highest in-sample accuracy for the out-sample. The spiders ran on 25 data sets created by Algorithm 13 using $n = 3000$ and $\tau = 0.4$.

	Reg. Plackett	Reg. Plackett (pop)	Cont. Plackett	Cont. Plackett (pop)
in	0.56	0.55	0.55	0.53
	(0.02)	(0.03)	(0.02)	(0.04)
out	0.55	0.54	0.54	0.52
	(0.03)	(0.03)	(0.03)	(0.04)

8 Empirical Application

To illustrate that the framework has predictive power in empirical settings, we use the ModeChoice as given in the Ecdat package (Croissant, 2016). This data has been analysed by Greene (2003), which allows us to compare the predictive copula Bayesian networks to tried models.

The data contains 210 observations belonging to 210 individuals who chose a particular mode of travel given some mode-specific characteristics. The possible choices were *air*, *bus*, *car* or *train*. The mode-specific characteristics are a terminal waiting time *ttme*, in-vehicle costs *invc*, in-vehicle travel time *invt* and some generalised costs *gc*. The terminal waiting time for *car* is zero. Then, we also have the household income *hinc* belonging to the individual. There is also a variable *psize*, described in (Croissant, 2016) as the party size of mode chosen. This variable should not be used as it is endogenous.

We randomly order the data and do a 3-fold cross validation. We use both a logit model and nested logit model as given in Greene (2003), see Train (2009) for elaborate introduction to (nested) logit models. Both logit models are estimated on the in-sample data and are used to predict on the out-sample data. Note that each in-sample data set contains solely 140 observations. We use the cross-validation Frank and Plackett copulaSpider as described in Section 5.6 with number of webs $n^{\text{web}} = 2$, number of runs per web $n^{\text{run}} = 5$, `score` = 0-1Scoreand maximum number of parents $n^{\text{pamax}} = 2$. As the in-sample contains 140 observations, each estimation and validation sample contains 70 observations. We construct a PCBN five times per estimation set and use the network with the highest accuracy on the validation set. Thus, we end up with $n^{\text{web}} = 2$ different networks. Then, we select the network with the highest in-sample accuracy and use that for the out-sample prediction. As the terminal waiting time *ttme* for *car* is zero, it is not used by the copula spiders.

Table 6: Mean and standard deviation of the in-sample and out-sample accuracy of the 3-fold cross-validation results on ModeChoice data of Croissant (2016), using the Plackett and Frank copulaSpider with $n^{\text{web}} = 2$, $n^{\text{run}} = 5$, `score` = 0-1Score, $n^{\max} = \infty$, $n^{\text{pamax}} = 2$. The spiders use the web with the highest in-sample accuracy for the out-sample predictions. The logit specifications used are given in Greene (2003)

	Plackett PCBN	Frank PCBN	Nested Logit	Logit
in	1.00	1.00	0.70	0.70
	(0.00)	(0.00)	(0.04)	(0.08)
out	0.99	0.98	0.69	0.67
	(0.01)	(0.02)	(0.06)	(0.05)

Table 6 shows the average accuracies of all four models. It is clear that the Plackett and Frank spiders outperform the logit models in terms of predictive accuracy. The average in-sample accuracy is 1.00 for both copula spiders and the average out-sample accuracy is almost 1.00.

The reported in-sample accuracy of the regular logit model is higher than what we may expect from (Greene, 2003). Greene estimates a logit model on the entire data set. Calculating the in-sample hit rate from Table 23.25 gives an accuracy of 0.52, but this is not correct. Greene reports $\beta_G = -0.15501$, but this should actually be -0.01550. Using the correct value for β_G, we would obtain an in-sample hit rate of 0.69. Table 6 shows that the nested logit specification performs slightly better than the simple logit, but the differences are minimal.

Table 7 shows the number of times that each variable appears in the three networks for the Frank and Plackett copulaSpider. The terminal waiting time appears most often in all webs of both the Frank and Plackett spider. Therefore, *ttme* could be considered the most important variable. The spiders also consistently pick up a correlation between the terminal waiting time of bus and train that helps to predict the choice mode.

In-vehicle travel time appears to be less important than terminal waiting time. Interestingly, the in-vehicle travel time of *air* is used only once by the Frank spider and it is not used at all by the Plackett spider. A possible explanation for this is that the terminal waiting time is considered to be bothersome, whereas the in-vehicle travel time could be neutral. Another possibility is that the terminal waiting time of *air* dominates the choice

mode decision. Lastly, the generalised costs of car are always used by the spiders. It appears that the greatest prohibition of taking the car are in fact the costs of the car.

Table 7: Number of times that each variable appears in the three networks for the Frank and Plackett copulaSpider. Choice refers to dependent variable which always appears in each network and bus&train refers to arc between ttme for bus and ttme for train

		Frank	Plackett
choice	all	3	3
hinc	all	2	2
ttme	air	3	3
	train	3	3
	bus	3	3
	bus&train	2	3
invt	air	1	-
	train	1	2
	bus	2	3
	car	1	2
invc	air	2	3
	train	1	2
	bus	2	1
	car	1	-
gc	air	1	2
	train	1	-
	bus	1	2
	car	3	3

Importantly, the choice of the copulas under consideration does not seem to alter the predictive performance noticeably. This is similar to the results found in Section 7.

Figure 8 shows the resulting graphs for the Plackett copulaSpider. Here, all arcs of the same colour make a directed acyclic graph. Figure 8 shows that the Plackett spider picks up correlations between the terminal waiting times. The in-vehicle travel times are occasionally modelled by copulas as well. Interestingly, the spiders do not pick up direct dependencies between the in-vehicle travel times and terminal waiting times.

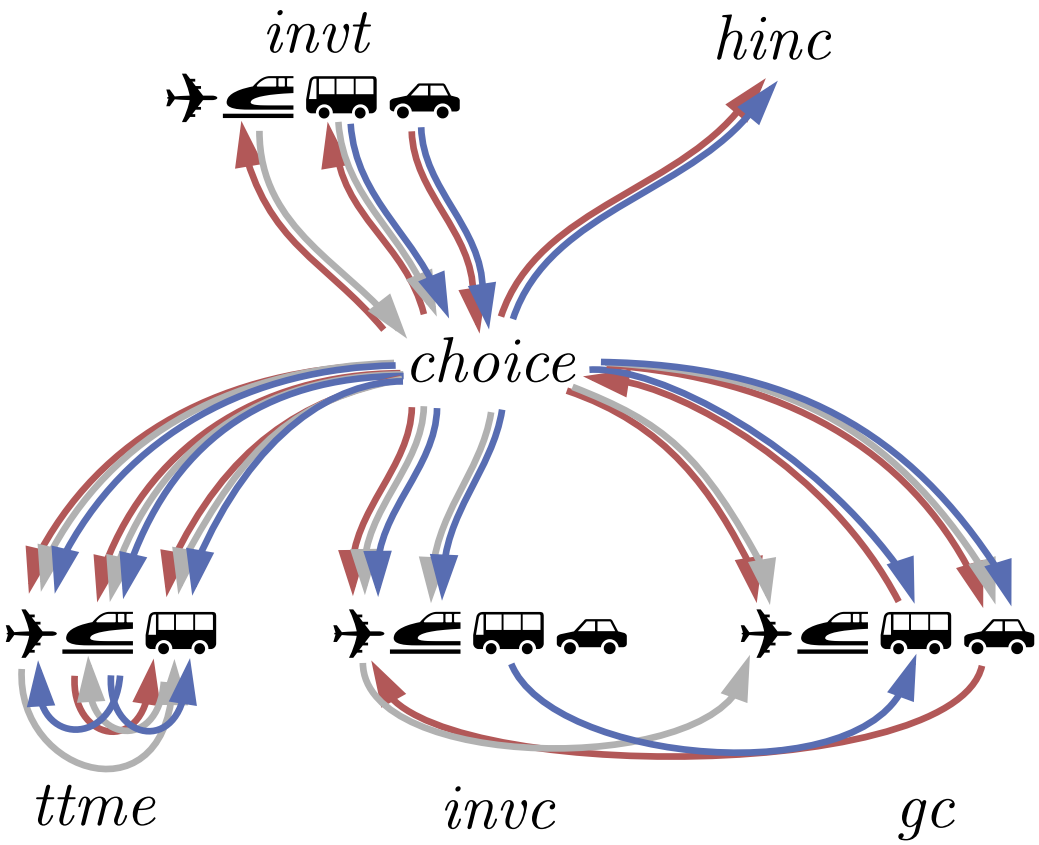

Figure 2: Resulting graphs of the Plackett copulaSpider where all arcs of the same colour make one graph. The variable names, in combination with the icons, correspond to random variables and the arrows correspond to directed arcs

Occasionally, the Plackett spider picks up a correlation between in-vehicle costs and generalised costs that helps predict the mode.

9 Conclusion and Discussion

The goal of this thesis is to increase the applicability and predictive performance of copula-based Bayesian network classifiers. These models fall under probabilistic classification, meaning that we model a joint distribution where the actual classification is based on a decision rule. As we can only use in-sample data, it is not trivial to get a classifier with good out-sample predictive power.

We set out specific goals that this thesis aims to reach. The first goal was to develop a framework that allows continuous, ordinal discrete and categorical explanatory variables to be modelled accurately. That is, we should not have to pretend that ordinal variables are continuous. Moreover, the framework should allow continuous nodes to have categorical children. We successfully developed a framework, called the predictive copula Bayesian network (PCBN), that reaches the first goal. Therefore, the applicability of copula-based Bayesian networks has been increased.

If one supplies a graph, then the framework can estimate the model associated with that graph. However, there are possibly many graphs that one could supply. We developed a greedy algorithm that constructs a PCBN from the data. This algorithm uses an estimation set to determine the parameters and a validation set to assess whether nodes and arcs may be added to the network. This approach reduces the chances of over fitting the data which would have hindered the out-sample performance. We use a technique, which we call the copula spider, that creates several PCBNs from the data.

In order to assess the predictive capabilities and performance gains of my approach, we consider both simulated and empirical data. We use the strict Clayton, strict reflected Clayton, Frank and Plackett copulas. We also use a variant which assesses all possible reflections of the Clayton copula. The simulation design shows that correctly modelling the explanatory variables leads to increases in both in-sample and out-sample accuracy. These gains become more apparent when the data is strongly correlated. Although the copulas behave differently, the specification of the copula does not influence the predictive performance noticeably.

We use travel mode data as an empirical application. The empirical case clearly illustrates that PCBNs, constructed using the proposed copula spider approach, can greatly outperform competing models. We find that the Frank and Plackett PCBNs can attain a near-perfect hit rate of 1. Importantly, the choice of the copula does not seem to alter the predictive accuracy by much.

Although the merits of my methods have been shown, we restricted the number of parent nodes to not exceed 2. Further research could be done to assess the PCBNs when the number of parents is allowed to be greater than two. Easily differentiable copulas would help ensure that the optimisation methods work. We also did not consider copulas that have multiple unknown parameters. Further research could be conducted to see if these could more accurately model data.

A possible extension of the PCBN might be to model a multivariate dependent variable rather than a univariate dependent variable. Then, the model only has to consider the Markov blanket of all dependent variables, where the dependent variables may be linked in some way. Another extension is to let the dependent variable be ordinal discrete rather than categorical. Further research could be conducted to see if the PCBNs can be applied to different settings.

References

Aitchison, J., & Aitken, C. G. (1976). Multivariate binary discrimination by the kernel method. *Biometrika*, *63*(3), 413–420.

Bauer, A., & Czado, C. (2016). Pair-copula Bayesian networks. *Journal of Computational and Graphical Statistics*, *25*(4), 1248–1271.

Bhat, C. R., & Sener, I. N. (2009). A copula-based closed-form binary logit choice model for accommodating spatial correlation across observational units. *Journal of Geographical Systems*, *11*(3), 243–272.

Brechmann, E., & Schepsmeier, U. (2013). CDVine: Modeling dependence with C-and D-vine copulas in R. *Journal of Statistical Software*, *52*(3), 1–27.

Brent, R. P. (1973). *Algorithms for minimization without derivatives*. Prentice-Hall.

Clayton, D. G. (1978). A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence. *Biometrika*, *65*(1), 141–151.

Cook, R. D., & Johnson, M. E. (1981). A family of distributions for modelling non-elliptically symmetric multivariate data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 210–218.

Croissant, Y. (2016). Ecdat: Data sets for econometrics [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=Ecdat` (R package version 0.3-1)

Croissant, Y. (2018). mlogit: Multinomial logit models [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=mlogit` (R package version 0.3-0)

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. John Wiley & Sons.

Elidan, G. (2010). Copula Bayesian networks. In *Advances in neural information processing systems* (pp. 559–567).

Elidan, G. (2012). Copula network classifiers (CNCs). In *Artificial intelligence and statistics* (pp. 346–354).

Embrechts, P., Lindskog, F., & McNeil, A. (2001). Modelling dependence with copulas. *Rapport technique, Département de mathématiques, Institut Fédéral de Technologie de Zurich, Zurich*.

Frank, M. J. (1979). On the simultaneous associativity of $F(x, y)$ and $x + y - F(x, y)$. *Aequationes Mathematicae*, *19*(1), 194–226.

Friedman, J. H. (1997). On bias, variance, 0/1loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*(2-3), 131–163.

Genest, C., & Nešlehová, J. (2007). A primer on copulas for count data. *ASTIN Bulletin: The Journal of the IAA*, *37*(2), 475–515.

Greene, W. H. (2003). *Econometric analysis*. Pearson Education India.

Hall, P., Racine, J., & Li, Q. (2004). Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, *99*(468), 1015–1026.

Hayfield, T., & Racine, J. S. (2008). Nonparametric econometrics: The np package. *Journal of Statistical Software*, *27*(5). Retrieved from `http://www.jstatsoft.org/v27/i05/`

Henningsen, A., & Toomet, O. (2011). maxLik: A package for maximum likelihood estimation in R. *Computational Statistics*, *26*(3), 443-458. Retrieved from `http://dx.doi.org/10.1007/s00180-010-0217-1` doi: 10.1007/s00180-010-0217-1

Joe, H., & Xu, J. J. (1996). *The estimation method of inference functions for margins for multivariate models* (Tech. Rep. No. 166). University of British Columbia, Department of Statistics.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the eleventh conference on uncertainty in artificial intelligence* (pp. 338–345).

Jondeau, E., & Rockinger, M. (2006). The copula-garch model of conditional dependencies: An international stock market application. *Journal of international money and finance*, *25*(5), 827–853.

Lauritzen, S. L. (1996). *Graphical models* (Vol. 17). Clarendon Press.

Li, Q., & Racine, J. (2003). Nonparametric estimation of distributions with categorical and continuous data. *Journal of Multivariate Analysis*, *86*(2), 266–292.

Nelsen, R. B. (2007). *An Introduction to Copulas*. Springer Science & Business Media.

Nešlehová, J. (2007). On rank correlation measures for non-continuous random variables.

Journal of Multivariate Analysis, *98*(3), 544–567.

Nikoloulopoulos, A. K. (2015). A mixed effect model for bivariate meta-analysis of diagnostic test accuracy studies using a copula representation of the random effects distribution. *Statistics in Medicine*, *34*(29), 3842–3865.

Nikoloulopoulos, A. K., & Karlis, D. (2008). Multivariate logit copula model with an application to dental data. *Statistics in Medicine*, *27*(30), 6393–6406.

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, *33*(3), 1065–1076.

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

Plackett, R. L. (1965). A class of bivariate distributions. *Journal of the American Statistical Association*, *60*(310), 516–522.

Pourret, O., Naïm, P., & Marcot, B. (2008). *Bayesian networks: A practical guide to applications* (Vol. 73). John Wiley & Sons.

R Core Team. (2018). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, *27*, 832–837.

Scherrer, R. J., Berlind, A. A., Mao, Q., & McBride, C. K. (2009). From finance to cosmology: The copula of large-scale structure. *The Astrophysical Journal Letters*, *708*(1), L9.

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, *8*, 229–231.

Sklar, A. (1996). Random variables, distribution functions, and copulas: a personal look backward and forward. *Lecture notes-monograph series*, 1–14.

Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

A Technical Supplement

This section of the Appendix contains background information,theoretical results and theorems relevant to the framework that are either deemed *required knowledge* or *supplementary conclusions* that supersede the main goals of this paper.

A.1 Graph and Set Theory

The mathematical definition of the n-ary Cartesian product employed in this thesis is given by Definition A.1.1. Note that the result is either an empty set, a set with elements or a set with n-tuples.

Definition A.1.1 (n-ary Cartesian product)**.** *Let there be n sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$ and let $\mathcal{N} = \{1, \ldots, n\}$ Then, the n-ary Cartesian product is the set of all ordered n-tuples given by*

$$\mathcal{X}_1 \times \ldots \times \mathcal{X}_n \equiv \{(x_1, \ldots, x_n) : x_i \in \mathcal{X}_i,\ \forall i \in \mathcal{N}\} \tag{27}$$

The definition of a path is given by Definition A.1.2. Using this definition, we can define define relations between nodes in a DAG. Four relations are included in Definition A.1.2.

Definition A.1.2 (Path)**.** *Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a graph $\mathcal{G}$ with set of nodes $\mathcal{N} = \{1, \ldots, n\}$ containing n elements and a set of arcs $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$. A path from node n_1 to n_k is a sequence of nodes $n_1, \ldots, n_k$ with $n_i \in \mathcal{N}$ for $i \in \{1, \ldots, k\}$, $(n_j, n_{j+1}) \in \mathcal{A}$ for $j \in \{1, \ldots, k-1\}$ and $n_i \neq n_l$ for all $i, l \in \{1, \ldots, k\}$ and $i \neq l$.*

A path from node n_1 to n_k is denoted as $n_1 \to \ldots \to n_k$.

With the definition of a path as given in Definition A.1.2, we can define a directed acyclic graph (DAG).

Definition A.1.3 (Relations between nodes)**.** *Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a direct acyclic graph $\mathcal{G}$ with a set of nodes $\mathcal{N}$ and a set of arcs $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$. Let $i, j \in \mathcal{A}$ with $i \neq j$, then:*

1. *i is called a* parent *of j if there is a direct path $i \to j$*
2. *j is called a* child *of i if there is a direct path $i \to j$*
3. *j is called a* descendant *of i if there is some path $i \to \ldots \to j$*
4. *j is called a* non-descendant *of i if there are not any paths $i \to \ldots \to j$*

A.2 Analytical Definition of a Copula

A copula over n uniform random variables can be written as $C_{\mathbf{U}}(u_1, \ldots, u_n)$ as $F_{U_i}(u_i) = u_i$. The analytical definition of an n-dimensional copula based on Nelsen (2007) is given in Definition A.2.1.

Definition A.2.1 (n-dimensional copula)**.** *Let $\mathcal{N} = \{1, \ldots, n\}$ be a set of $n \in \mathbb{N}^+$ indices. Let $\mathbf{U} = (U_1, \ldots, U_n)$ be a vector of n uniform random variables with corresponding realisation vector $\mathbf{u} = (u_1, \ldots, u_n)$ where each $u_i \in [0, 1]$ for $i \in \mathcal{N}$. Then, a copula is a function $C : [0, 1]^n \to [0, 1]$ that satisfies the following properties:*

1. *Let some $u_i = 0$ for $i \in \mathcal{N}$. Then, $C_{\mathbf{U}}(\mathbf{u}) = 0$.*

2. *Denote $\mathcal{N}^* = \mathcal{N} \setminus \{i\}$ as the set of indices where one index $i \in \mathcal{N}$ is missing. Let $u_{i^*} = 1$ for all $i^* \in \mathcal{N}^*$. Then, $C_{\mathbf{U}}(\mathbf{u}) = u_i$.*

3. *Let $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{W} = \{w_1, \ldots, w_n\}$ be two sets with n elements such that $v_i, w_i \in [0, 1]$ and $v_i \leq w_i$ for $i \in \mathcal{N}$. Additionally, $v_k \leq v_{k+1}$ and $w_k \leq w_{k+1}$ for all $k \in \mathcal{N} \setminus \{n\}$. Now, denote $\mathcal{U} = \{v_1, w_1\} \times \ldots \times \{v_n, w_n\}$ as the n-ary Cartesian product of all pairs $\{v_i, w_i\}$. Then, we have the following restriction:*

$$\sum_{\mathbf{u}^* \in \mathcal{U}} (-1)^{\sum_{i \in \mathcal{N}} [v_i = u_i^*]} C_{\mathbf{U}^*}(\mathbf{u}^*) \geq 0$$

where $\mathbf{u}^*$ is an n-tuple of $\mathcal{U}$, u_i^* is the i^{th} value of $\mathbf{u}^*$, $\sum_{i \in \mathcal{N}} [v_i = u_i^*]$ is a summation of Iverson brackets where $[v_i = u_i^*] = 1$ if the condition is true, and 0 if the condition is false. Note that $(-1)^{\sum_{i \in \mathcal{N}} [v_i = u_i^*]} = 1$ if $v_i = u_i^*$ is true for an even number of times, and -1 if $v_i = u_i^*$ is true for an odd number of times.

Essentially, this implies that C has to be n-increasing.

A.3 Reflected Copulas

Some copulas, such as the strict Clayton copula, can only model positive dependence between variables. In order to model negative dependence with such copulas, one can *reflect* the copula with respect to a hyperplane. Reflection is often called *rotation* for bivariate copulas, see for instance Brechmann and Schepsmeier (2013) and Nikoloulopoulos (2015).

However, the exact definition of rotation differs in the literature. Consider a bivariate copula $C_{U_1,U_2}(u_1,u_2)$ and let $C^{90}_{U_1,U_2}(u_1,u_2)$ denote the copula notated by 90 degrees. Then, Brechmann and Schepsmeier (2013) uses $C^{90}_{U_1,U_2}(u_1,u_2) = u_2 - C_{U_1,U_2}(1-u_1,u_2)$. Yet, Nikoloulopoulos (2015) uses $C^{90}_{U_1,U_2}(u_1,u_2) = u_1 - C_{U_1,U_2}(u_1,1-u_2)$.

The concept of rotation becomes more difficult to grasp when the number of dimensions increases. Therefore, we use the term reflected copula. We give a definition for the reflected copula in Definition A.3.1. We introduce the reflection operator (with respect to a hyperplane) in Definition A.3.2.

A.3.1 n-variate Reflected Copulas

Definition A.3.1 (Reflection of n-variate copulas with respect to a hyperplane)**.** *Let $U_1,\ldots,U_n$ be n uniform random variables with realisations $u_1,\ldots,u_n$. Denote the n-variate copula as $C_{U_1,\ldots,U_n}(u_1,\ldots,u_n) = \mathrm{P}[U_1 \leq u_1,\ldots,U_n \leq u_n]$.*

Now, let some $U_i^* = 1-U_i$ with $i \in \{1,\ldots,n\}$. Then, we call $C_{U_1,\ldots,U_i^*,\ldots,U_n}(u_1,\ldots,u_n) = \mathrm{P}[U_1 \leq u_1,\ldots,U_i^* \leq u_i,\ldots,U_n \leq u_n]$ the reflected copula in U_i with respect to the hyperplane $u_i = \frac{1}{2}$. We call this the reflected copula in U_i as well.

Definition A.3.2 (Reflection operator)**.** *Let $U_1,\ldots,U_n$ be n uniform random variables with realisations $u_1,\ldots,u_n$. Denote the n-variate copula as $C_{U_1,\ldots,U_n}(u_1,\ldots,u_n)$. Then, the reflection operator on the i^{th} argument with respect to the hyperplane $u_i = \frac{1}{2}$ is denoted as $R_i C_{U_1,\ldots,U_n}(u_1,\ldots,u_n) \equiv C_{U_1,\ldots,U_i^*,\ldots,U_n}(u_1,\ldots,u_n)$ which is the reflected copula in U_i^* as given by Definition A.3.1. A sequence of reflection operators $R_j \cdots R_k$ is denoted as $R_{j,\ldots,k}$ where $j,k \in \{1,\ldots,n\}$.*

A bivariate copula reflected in both of its arguments gives a particular expression. Nelsen (2007) calls this expression the *survival copula*. He shows that $R_{1,2}C_{U_1,U_2}(u_1,u_2) = u_1 + u_2 - 1 + C_{U_1,U_2}(1-u_1,1-u_2)$. We give the definition of a joint survival copula in Definition A.3.3.

Definition A.3.3 (Joint survival copula)**.** *Let $U_1,\ldots,U_n$ be n uniform random variables with realisations $u_1,\ldots,u_n$. Denote the n-variate copula as $C_{U_1,\ldots,U_n}(u_1,\ldots,u_n)$. Then, $R_{1,\ldots,n}C_{U_1,\ldots,U_n}(u_1,\ldots,u_n)$ is called the joint survival copula.*

One may wonder if any joint survival copula can be expressed by their non-reflected counterparts. Not only is this possible for joint survival copulas, it is possible for *any* reflected copula. We show this in Theorem A.3.4. The relation between n-variate joint survival copulas and their non-reflected counterparts follows immediately from Theorem A.3.4. This relation is given by Corollary A.3.4.1.

Theorem A.3.4 (Relation reflected copula and regular copula)**.** *Let* $\mathcal{N} = \{1, \ldots, n\}$ *be a set of* $n \in \{2, 3, \ldots\}$ *indices and* $C_{(U_i)_{i\in\mathcal{N}}}\left((u_i)_{i\in\mathcal{N}}\right) \equiv C_{U_1,\ldots,U_n}(u_1, \ldots, u_n)$ *be an* n*-variate copula. Let* $\mathcal{R} \subseteq \mathcal{N}$ *be a set of all indices to be reflected and* $\mathcal{M} = \mathcal{N} \setminus \mathcal{R}$ *as the set of indices that will not be reflected. Let* $R_{(r)_{r\in\mathcal{R}}}$ *denote the reflection operator as defined in Definition A.3.2. Then, the reflected copula in* $(U_r)_{r\in\mathcal{R}}$ *is given by:*

$$
\begin{aligned}
&R_{(r)_{r\in\mathcal{R}}} C_{(U_i)_{i\in\mathcal{N}}}\left((u_i)_{i\in\mathcal{N}}\right) \\
&= [\mathcal{M} = \emptyset]\left(-(n-1) + \sum_{i\in\mathcal{N}} u_i\right) + [|\mathcal{M}| = 1]\,(u_j)_{j\in\mathcal{M}} + [|\mathcal{M}| > 1]\, C_{(U_j)_{j\in\mathcal{M}}}\left((u_j)_{j\in\mathcal{M}}\right) \\
&\quad - [|\mathcal{M}| > 0] \sum_{r\in\mathcal{R}} C_{(U_j)_{j\in\{r\}\cup\mathcal{M}}}\left(\left(u_i^{[j\neq r]}\,(1-u_j)^{[j=r]}\right)_{j\in\{r\}\cup\mathcal{M}}\right) \\
&\quad + \sum_{r=2}^{|\mathcal{R}|}(-1)^r \sum_{\mathcal{R}^*\subseteq\mathcal{R}:|\mathcal{R}^*|=r} C_{(U_k)_{k\in\mathcal{R}^*\cup\mathcal{M}}}\left(\left(u_k^{[k\notin\mathcal{R}^*]}\,(1-u_k)^{[k\in\mathcal{R}^*]}\right)_{k\in\mathcal{R}^*\cup\mathcal{M}}\right). \qquad (28)
\end{aligned}
$$

Note that we do not mean that the copula is defined over tuples nor that it takes a tuple as its argument.

Proof. We have that $R_i C(\cdot)$ transforms the i^{th} variable from $U_i \to U_i^* = 1 - U_i$. Now, we use $\mathrm{P}\left[U_i^* \leq u_i\right] = \mathrm{P}\left[1 - U_i \leq u_i\right] = \mathrm{P}\left[U_i > u_i\right]$. Thus, we have that $R_{r_1,\ldots,r_n} C_{U_1,\ldots,U_n}(u_1, \ldots, u_n) = \mathrm{P}\left[U_{r_1} > 1 - u_{r_1}, \ldots, U_{r_n} > 1 - u_{r_n}, U_{j_1} \leq u_{j_1}, \ldots, U_{j_n} \leq u_{j_n}\right]$. We rewrite this probability by using the inclusion-exclusion principle. Let there be n events

$E_1 \ldots, E_n$ Then, the inclusion-exclusion principle for probabilities states that

$$\begin{aligned}
\mathrm{P}\left[\bigcup_{i\in\mathcal{N}} E_i\right] &= \sum_{i\in\mathcal{N}} \mathrm{P}\left[E_i\right] - \sum_{i,j\in\mathcal{N}:i<j} \mathrm{P}\left[E_i \cap E_j\right] \\
&\quad + \sum_{i,j,k\in\mathcal{N}:i<j<k} \mathrm{P}\left[E_i \cap E_j \cap E_k\right] - \ldots + (-1)^{n-1}\mathrm{P}\left[\bigcap_{i\in\mathcal{N}} E_i\right] \\
&= \sum_{i\in\mathcal{N}} (-1)^{i-1} \sum_{\mathcal{M}\subseteq\mathcal{N}:|\mathcal{M}|=i} \mathrm{P}\left[\bigcap_{j\in\mathcal{M}} E_j\right]
\end{aligned}$$

where we need to have that $\mathcal{N} = \{1, \ldots, n\}$. We want to find an expression for $\mathrm{P}\left[\bigcap_{r\in\mathcal{R}} \{U_r > 1 - u_r\} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right]$. Let $\mathcal{S}$ denote the sample space. Then, by using De Morgan's laws we have:

$$\begin{aligned}
&\mathrm{P}\left[\bigcap_{r\in\mathcal{R}} \{U_r > 1 - u_r\} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right] \\
&= \mathrm{P}\left[\left(\mathcal{S} \setminus \bigcup_{r\in\mathcal{R}} \{U_r \leq 1 - u_r\}\right) \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right] \\
&= \mathrm{P}\left[\left(\mathcal{S} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right) \setminus \left(\bigcup_{r\in\mathcal{R}} \left(\{U_r \leq 1 - u_r\} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right)\right)\right] \\
&= \mathrm{P}\left[\mathcal{S} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right] - \mathrm{P}\left[\bigcup_{r\in\mathcal{R}} \left(\{U_r \leq 1 - u_r\} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right)\right] \\
&= [\mathcal{M} = \emptyset] + (1 - [\mathcal{M} = \emptyset])\,\mathrm{P}\left[\bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right] - \mathrm{P}\left[\bigcup_{r\in\mathcal{R}} \left(\{U_r \leq 1 - u_r\} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right)\right]
\end{aligned}$$

where we use that the complement of the event $\{U_r > 1 - u_r\}$ equals $\{U_r \leq 1 - u_r\}$. We can use a subtraction of probabilities as $\bigcup_{r\in\mathcal{R}} \{U_r \leq 1 - u_r\} \subset \mathcal{S}$ and thus $\bigcup_{r\in\mathcal{R}} \{U_r \leq 1 - u_r\} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\} \subset \mathcal{S} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}$. Note that $\mathrm{P}\left[\mathcal{S} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right] \neq \mathrm{P}\left[\bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right]$ for $\mathcal{M} = \emptyset$. For $\mathcal{M} = \emptyset$ we have that $\mathrm{P}\left[\mathcal{S} \bigcap_{j\in\mathcal{M}} \{U_j \leq u_j\}\right] = \mathrm{P}\left[\mathcal{S}\right] = 1 = [\mathcal{M} = \emptyset]$ Now, we use the inclusion-exclusion

principle so that

$$
\begin{aligned}
&\mathrm{P}\left[\bigcap_{r\in\mathcal{R}}\{U_r > 1-u_r\}\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right]\\
&=[\mathcal{M}=\emptyset]+(1-[\mathcal{M}=\emptyset])\,\mathrm{P}\left[\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right]-\mathrm{P}\left[\bigcup_{r\in\mathcal{R}}\left(\{U_r\le 1-u_r\}\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right)\right]\\
&=[\mathcal{M}=\emptyset]+(1-[\mathcal{M}=\emptyset])\,\mathrm{P}\left[\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right]\\
&\quad-\sum_{r=1}^{|\mathcal{R}|}(-1)^{r-1}\sum_{\mathcal{R}^*\subseteq\mathcal{R}:|\mathcal{R}^*|=r}\mathrm{P}\left[\bigcap_{r^*\in\mathcal{R}^*}\left(\{U_{r^*}\le 1-u_{r^*}\}\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right)\right]\\
&=[\mathcal{M}=\emptyset]+(1-[\mathcal{M}=\emptyset])\,\mathrm{P}\left[\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right]\\
&\quad+\sum_{r=1}^{|\mathcal{R}|}(-1)^{r}\sum_{\mathcal{R}^*\subseteq\mathcal{R}:|\mathcal{R}^*|=r}\mathrm{P}\left[\bigcap_{r^*\in\mathcal{R}^*}\left(\{U_{r^*}\le 1-u_{r^*}\}\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right)\right]
\end{aligned}
$$

The joint probabilities can be written in terms of copulas, whereas we can use $\mathrm{P}\left[U_r\le u_r\right]=u_r$ for univariate probabilities. Thus:

$$
\begin{aligned}
&\mathrm{P}\left[\bigcap_{r\in\mathcal{R}}\{U_r > 1-u_r\}\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right]\\
&=[\mathcal{M}=\emptyset]+(1-[\mathcal{M}=\emptyset])\,\mathrm{P}\left[\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right]\\
&\quad-\sum_{r\in\mathcal{R}}\mathrm{P}\left[\{U_r\le 1-u_r\}\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right]\\
&\quad+\sum_{r=2}^{|\mathcal{R}|}(-1)^{r}\sum_{\mathcal{R}^*\subseteq\mathcal{R}:|\mathcal{R}^*|=r}\mathrm{P}\left[\bigcap_{r^*\in\mathcal{R}^*}\left(\{U_{r^*}\le 1-u_{r^*}\}\bigcap_{j\in\mathcal{M}}\{U_j\le u_j\}\right)\right]\\
&=[\mathcal{M}=\emptyset]+[|\mathcal{M}|=1]\,(u_j)_{j\in\mathcal{M}}+[|\mathcal{M}|>1]\,C_{(U_j)_{j\in\mathcal{M}}}\left((u_j)_{j\in\mathcal{M}}\right)\\
&\quad-[\mathcal{M}=\emptyset]\sum_{r\in\mathcal{R}}(1-u_r)-[|\mathcal{M}|>0]\sum_{r\in\mathcal{R}}C_{(U_j)_{j\in\{r\}\cup\mathcal{M}}}\left(\left(u_i^{[j\neq r]}\,(1-u_j)^{[j=r]}\right)_{j\in\{r\}\cup\mathcal{M}}\right)\\
&\quad+\sum_{r=2}^{|\mathcal{R}|}(-1)^{r}\sum_{\mathcal{R}^*\subseteq\mathcal{R}:|\mathcal{R}^*|=r}C_{(U_k)_{k\in\mathcal{R}^*\cup\mathcal{M}}}\left(\left(u_k^{[k\notin\mathcal{R}^*]}\,(1-u_k)^{[k\in\mathcal{R}^*]}\right)_{k\in\mathcal{R}^*\cup\mathcal{M}}\right)
\end{aligned}
$$

Note that $\mathcal{R}^*\subseteq\mathcal{R}:|\mathcal{R}^*|=1$ is equivalent with $r\in\mathcal{R}$.

Now we can use that $[\mathcal{M}=\emptyset]-[\mathcal{M}=\emptyset]\sum_{r\in\mathcal{R}}(1-u_r)=[\mathcal{M}=\emptyset]\left(-(|\mathcal{R}|-1)+\sum_{r\in\mathcal{R}}u_r\right)$.

Note that this expression is non-zero only when $\mathcal{M} = \emptyset$. When that is the case, we have $\mathcal{R} = \mathcal{N}$ and $|\mathcal{R}| = n$ which concludes the proof. □

Corollary A.3.4.1 (Relation joint survival copula and regular copula)**.** *Let* $R_{1,\dots,n}C_{U_1,\dots,U_n}(u_1,\dots,u_n)$ *be a joint survival copula. Then we have that*

$$\begin{aligned} &R_{1,\dots,n}C_{U_1,\dots,U_n}(u_1,\dots,u_n) \\ &= -(n-1) + \sum_{k\in\mathcal{N}} u_k + \sum_{i\in\mathcal{N}:i>1} (-1)^i \sum_{\mathcal{R}^*\subseteq\mathcal{N}:|\mathcal{R}^*|=i} C_{(U_j)_{j\in\mathcal{R}^*}}\left((1-u_j)_{j\in\mathcal{R}^*}\right). \end{aligned} \tag{29}$$

Note that we do not mean that the copula is defined over tuples nor that it takes a tuple as its argument.

Proof. This follows immediately from Theorem A.3.4. We have that $\mathcal{R} = \mathcal{N} = \{1,\dots,n\}$ and $\mathcal{M} = \emptyset$. Therefore, $[\mathcal{M} = \emptyset] = 1$, $[|\mathcal{M}| > 0] = 0$, $[|\mathcal{M}| = 1] = 0$ and $[|\mathcal{M}| > 1] = 0$. □

A.3.2 2- and 3-variate Reflected Copulas

Often, applications use bivariate and occasionally trivariate copulas. Corollary A.3.4.2 and A.3.4.2 show, respectively, the relations of reflected bivariate and trivariate copulas to the regular copula.

Corollary A.3.4.2 (Relation bivariate reflected copula and regular copula)**.** *Let* $C_{U_1,U_2}(u_1,u_2)$ *be a bivariate copula. Then, we have the following relations for the reflections:*

$$R_1C_{U_1,U_2}(u_1,u_2) = u_2 - C_{U_1,U_2}(1-u_1,u_2) \tag{30}$$

$$R_{1,2}C_{U_1,U_2}(u_1,u_2) = -1 + u_1 + u_2 + C_{U_1,U_2}(1-u_1,1-u_2) \tag{31}$$

$$R_2C_{U_1,U_2}(u_1,u_2) = u_1 - C_{U_1,U_2}(u_1,1-u_2) \tag{32}$$

Proof. This follows immediately from Theorem A.3.4. For R_1 we have $\mathcal{R} = \{1\}$ and $\mathcal{M} = \{2\}$. For $R_{1,2}$ we have $\mathcal{R} = \{1,2\}$ and $\mathcal{M} = \emptyset$. Lastly, for R_2 we have $\mathcal{R} = \{2\}$ and $\mathcal{M} = \{1\}$. □

Corollary A.3.4.3 (Relation trivariate reflected copula and regular copula)**.** *Let* $C_{U_1,U_2,U_3}(u_1,u_2,u_3)$ *be a trivariate copula. Then, we have the following relations for the*

reflections:

$$
\begin{aligned}
R_1 C_{U_1,U_2,U_3}(u_1,u_2,u_3) &= C_{U_2,U_3}(u_2,u_3) - C_{U_1,U_2,U_3}(1-u_1,u_2,u_3)\\
R_{1,2} C_{U_1,U_2,U_3}(u_1,u_2,u_3) &= u_3 - C_{U_1,U_3}(1-u_1,u_3) - C_{U_2,U_3}(1-u_2,u_3)\\
&\quad + C_{U_1,U_2,U_3}(1-u_1,1-u_2,u_3)\\
R_{1,2,3} C_{U_1,U_2,U_3}(u_1,u_2,u_3) &= -2 + u_1 + u_2 + u_3 + C_{U_1,U_2}(1-u_1,1-u_2)\\
&\quad + C_{U_1,U_3}(1-u_1,1-u_3) + C_{U_2,U_3}(1-u_2,1-u_3)\\
&\quad - C_{U_1,U_2,U_3}(1-u_1,1-u_2,1-u_3)\\
R_{1,3} C_{U_1,U_2,U_3}(u_1,u_2,u_3) &= u_2 - C_{U_1,U_2}(1-u_1,u_2) - C_{U_2,U_3}(u_2,1-u_3)\\
&\quad + C_{U_1,U_2,U_3}(1-u_1,1-u_2,u_3)\\
R_2 C_{U_1,U_2,U_3}(u_1,u_2,u_3) &= C_{U_1,U_3}(u_1,u_3) - C_{U_1,U_2,U_3}(u_1,1-u_2,u_3)\\
R_{2,3} C_{U_1,U_2,U_3}(u_1,u_2,u_3) &= u_1 - C_{U_1,U_2}(u_1,1-u_2) - C_{U_1,U_3}(u_1,1-u_3)\\
&\quad + C_{U_1,U_2,U_3}(u_1,1-u_2,1-u_3)\\
R_3 C_{U_1,U_2,U_3}(u_1,u_2,u_3) &= C_{U_1,U_2}(u_1,u_2) - C_{U_1,U_2,U_3}(u_1,u_2,1-u_3)
\end{aligned}
$$

Proof. This follows immediately from Theorem A.3.4. For R_1 we have $\mathcal{R} = \{1\}$ and $\mathcal{M} = \{2,3\}$. For $R_{1,2}$ we have $\mathcal{R} = \{1,2\}$ and $\mathcal{M} = \{3\}$. Lastly, for $R_{1,2,3}$ we have $\mathcal{R} = \{1,2,3\}$ and $\mathcal{M} = \emptyset$. The other configurations follow similarly. □

A.3.3 Argument-Addition Operator

One can conveniently determine the formula of some higher-variate reflected copulas in terms of non-reflected copulas by adding arguments to lower-variate copulas. We introduce the *argument-addition operator* which is given by Definition A.3.5. With the argument-addition operator, Theorem A.3.6 defines the relation between reflected copulas and their non-reflected counterparts differently.

Definition A.3.5 (Argument-addition operator)**.** *Let $U_1,\ldots,U_n$ be n uniform random variables with realisations $u_1,\ldots,u_n$. Denote the n-variate copula as $C_{U_1,\ldots,U_n}(u_1,\ldots,u_n)$. Then, the argument-addition (AA) operator that adds i arguments to a copula is defined as*

$A_{1,\ldots,i} C_{U_1,\ldots,U_n}(u_1,\ldots,u_n) \equiv C_{U_1,\ldots,U_n,U_{n+1},\ldots,U_{n+i}}(u_1,\ldots,u_n,u_{n+1},\ldots,u_{n+i})$. The AA operator on some $u_j \in [0,1]$ belonging to uniform random variable U_j is defined as $A_{1,\ldots,i} u_j \equiv$

$C_{U_j,U_{j+1},\ldots,U_{j+i}}(u_j, u_{j+1}, \ldots, u_{j+i})$. *Adding a named argument* u_i *can be denoted as* A_{u_i}. *Note that adding the same argument twice is undefined.*

Theorem A.3.6 (Argument-addition operator and reflected copulas)**.** *Let* $\mathcal{N} = \{1, \ldots, n\}$ *be a set of* $n \in \{2, 3, \ldots\}$ *indices and* $C_{(U_i)_{i\in\mathcal{N}}}\left((u_i)_{i\in\mathcal{N}}\right) \equiv C_{U_1,\ldots,U_n}(u_1, \ldots, u_n)$ *be an* n*-variate copula. Let* $\mathcal{R} \subseteq \mathcal{N}$ *be a set of all indices to be reflected and* $\mathcal{M} = \mathcal{N} \setminus \mathcal{R}$ *as the set of indices that will not be reflected. Let* $R_{(r)_{r\in\mathcal{R}}}$ *denote the reflection operator as defined in Definition A.3.2 and let* A *denote the argument-addition operator as defined in reflected copula in A.3.5. Then, the following relation holds:*

$$R_{(r)_{r\in\mathcal{R}}} C_{(U_i)_{i\in\mathcal{N}}}\left((u_i)_{i\in\mathcal{N}}\right) = A_{(u_j)_{j\in\mathcal{M}}} R_{(r)_{r\in\mathcal{R}}} C_{(U_r)_{r\in\mathcal{R}}}\left((u_r)_{r\in\mathcal{R}}\right). \tag{33}$$

That is, we reflect an $|\mathcal{R}|$*-variate copula in all of its arguments and add the other arguments afterwards.*

Proof. It is apparent that $A_{(u_j)_{j\in\mathcal{M}}} C_{(U_r)_{r\in\mathcal{R}}}\left((u_r)_{r\in\mathcal{R}}\right) = A_{(u_j)_{j\in\mathcal{M}}} \mathrm{P}\left[\bigcap_{r\in\mathcal{R}}\{U_r \le u_r\}\right] = \mathrm{P}\left[\bigcap_{r\in\mathcal{R}}\{U_r \le u_r\}\bigcap_{j\in\mathcal{M}}\{U_j \le u_j\}\right] = C_{(U_i)_{i\in\mathcal{N}}}$. Similarly, we have
$A_{(u_j)_{j\in\mathcal{M}}} R_{(r)_{r\in\mathcal{R}}} C_{(U_r)_{r\in\mathcal{R}}}\left((u_r)_{r\in\mathcal{R}}\right) = A_{(u_j)_{j\in\mathcal{M}}} \mathrm{P}\left[\bigcap_{r\in\mathcal{R}}\{U_r > 1 - u_r\}\right]$
$= \mathrm{P}\left[\bigcap_{r\in\mathcal{R}}\{U_r > 1 - u_r\}\bigcap_{j\in\mathcal{M}}\{U_j \le u_j\}\right] = R_{(r)_{r\in\mathcal{R}}} C_{(U_i)_{i\in\mathcal{N}}}\left((u_i)_{i\in\mathcal{N}}\right)$ which concludes the proof. □

The AA-operator itself is, like the reflection operator, commutative. Moreover, we have $A_{u_i} R_{u_j} = R_{u_j} A_{u_i}$ if and only if $u_i \neq u_j$. If $u_i = u_j$, then $A_{u_i} R_{u_j}$ is not defined as it implies that the variable is reflected and then added, even though it needs to exist when the reflection takes place. Adding a variable that is already defined is an undefined operation.

A.3.4 Derivatives of Reflected Copulas

Theorem A.3.7 shows that the derivative of *any* n-variate reflected copula solely depends on the (generally mixed) derivative of n-variate non-reflected copulas.

Theorem A.3.7 (n^{th}-order derivatives of n-variate reflected copulas)**.** *Using the notation in Theorem A.3.4, the relation for* n^{th}*-order mixed derivative of the* n*-variate reflected*

copula is given by:

$$\begin{aligned}
&\sum_{\left(u_k^*\right)_{k\in\mathcal{N}^{\text{disc}}}\in\mathcal{U}} (-1)^{\sum_{k\in\mathcal{N}^{\text{disc}}}\left[u_k^- = u_k^*\right]} D_{(j)_{j\in\mathcal{N}^{\text{cont}}}} R_{(r)_{r\in\mathcal{R}}} C_{(U_i)_{i\in\mathcal{N}}}\left(\left(u_j\right)_{j\in\mathcal{N}^{\text{cont}}}, \left(u_k^*\right)_{k\in\mathcal{N}^{\text{disc}}}\right) \\
&= \sum_{\left(u_k^*\right)_{k\in\mathcal{N}^{\text{disc}}}\in\mathcal{U}} (-1)^{\sum_{k\in\mathcal{N}^{\text{disc}}}\left[u_k^- = u_k^*\right] + |\mathcal{R}| + \left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|} D_{(j)_{j\in\mathcal{N}^{\text{cont}}}} \\
&\quad \times C_{(U_i)_{i\in\mathcal{N}}}\left(\left(u_i^{\left[i\in\mathcal{M}\cap\mathcal{N}^{\text{cont}}\right]}\left(1-u_i\right)^{\left[i\in\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right]}\left(u_i^*\right)^{\left[i\in\mathcal{M}\cap\mathcal{N}^{\text{disc}}\right]}\left(1-u_i^*\right)^{\left[i\in\mathcal{R}\cap\mathcal{N}^{\text{disc}}\right]}\right)_{i\in\mathcal{N}}\right)
\end{aligned} \tag{34}$$

where $\mathcal{U} = \prod_{k\in\mathcal{N}^{\text{disc}}}\left\{u_k^-, u_k\right\}$ *is the* n^{disc}*-ary Cartesian product of all* n^{disc} *discrete realisations and their left limits with the Cartesian product containing* $|\mathcal{U}| = 2^{n^{\text{disc}}}$ n^{disc}*-tuples* $\left(u_k^*\right)_{k\in\mathcal{N}^{\text{disc}}}$*,* $\mathcal{R}$ *is the set of indices corresponding to random variables to be reflected and* $\mathcal{M} = \mathcal{R}\setminus\mathcal{N}$*. We use Euler's differential notation, meaning that we take partial derivatives of* C *with respect to the first* n^{cont} *arguments* first *and then evaluate the obtained expression.* R *refers to the reflection operator given by Definition A.3.2.*

Proof. This follows from (28) in Theorem A.3.4. Intuitively, if some U_i is continuous for $i\in\mathcal{N}$, then the partial derivatives of the terms *not containing* u_i are zero. Moreover, if some U_j is discrete, then taking differences of terms *not* containing u_j results in zeros. Therefore, the only term that could be non-zero is the term containing *all* variables. This term is preceded by $(-1)^{|\mathcal{R}|}$, see (28). Note that $D_1R_1C_{U_1,\ldots} = -D_1C_{U_1,\ldots}$. Using Leibniz notation, that means:

$$\frac{\partial C_{U_1,\ldots}}{\partial u_1} = \frac{\partial\left(1-u_1\right)}{\partial u_1}\frac{\partial C_{U_1,\ldots}}{\partial\left(1-u_1\right)} = -\frac{\partial C_{U_1,\ldots}}{\partial\left(1-u_1\right)}.$$

The first argument of $D_1R_1C_{U_1,\ldots}$ is in fact $1-u_1$. Thus, we have to multiply the derivative of the copula with $(-1)^{\left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|}$. □

Corollary A.3.7.1 (2nd-order derivatives of continuous bivariate reflected copulas)**.** *Let* $C_{U_1,U_2}\left(u_1,u_2\right)$ *be a bivariate copula. Let both* U_1 *and* U_2 *be continuous uniform random*

variables. Then, the 2nd-order derivatives of the reflected copulas are given by:

$$
\begin{aligned}
D_{1,2}R_1C_{U_1,U_2}(u_1,u_2) &= D_{1,2}C_{U_1,U_2}(1-u_1,u_2)\\
D_{1,2}R_{1,2}C_{U_1,U_2}(u_1,u_2) &= D_{1,2}C_{U_1,U_2}(1-u_1,1-u_2)\\
D_{1,2}R_2C_{U_1,U_2}(u_1,u_2) &= D_{1,2}C_{U_1,U_2}(u_1,1-u_2)
\end{aligned}
$$

Proof. This follows immediately from (34) in Theorem A.3.7. In all cases we have $\mathcal{N}^{\text{disc}} = \emptyset$. For $D_{1,2}R_1C_{U_1,U_2}$ and $D_{1,2}R_2C_{U_1,U_2}$ we have $|\mathcal{R}| = |\mathcal{R} \cap \mathcal{N}^{\text{cont}}| = 1$. Thus, $(-1)^{|\mathcal{R}|+\left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|} = 1$. For $D_{1,2}R_{1,1}C_{U_1,U_2}$ we have $|\mathcal{R}| = |\mathcal{R} \cap \mathcal{N}^{\text{cont}}| = 2$ and thus $(-1)^{|\mathcal{R}|+\left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|} = 1$ as well. □

Corollary A.3.7.2 (2nd-order derivatives of mixed bivariate reflected copulas)**.** *Let $C_{U_1,U_2}(u_1,u_2)$ be a bivariate copula. Let U_1 be a continuous uniform random variable and let U_2 be discrete uniform. Then, the 2nd-order mixed derivatives are given by:*

$$
\begin{aligned}
&D_1R_1C_{U_1,U_2}(u_1,u_2) - D_1R_1C_{U_1,U_2}\left(u_1,u_2^-\right)\\
&= D_1C_{U_1,U_2}(1-u_1,u_2) - D_1C_{U_1,U_2}\left(1-u_1,u_2^-\right)\\
&D_1R_{1,2}C_{U_1,U_2}(u_1,u_2) - D_1R_{1,2}C_{U_1,U_2}\left(u_1,u_2^-\right)\\
&= D_1C_{U_1,U_2}\left(1-u_1,1-u_2^-\right) - D_1C_{U_1,U_2}(1-u_1,1-u_2)\\
&D_1R_2C_{U_1,U_2}(u_1,u_2) - D_1R_2C_{U_1,U_2}\left(u_1,u_2^-\right)\\
&= D_1C_{U_1,U_2}\left(u_1,1-u_2^-\right) - D_1C_{U_1,U_2}(u_1,1-u_2)
\end{aligned}
$$

Proof. This follows immediately from (34) in Theorem A.3.7. In all cases we have $\mathcal{N}^{\text{disc}} = \{2\}$. For $D_1R_1C_{U_1,U_2}$ we have $|\mathcal{R}| = |\mathcal{R} \cap \mathcal{N}^{\text{cont}}| = 1$. Thus, $(-1)^{\left[u_2=u_2^-\right]+|\mathcal{R}|+\left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|} = 1$ and $(-1)^{\left[u_2^-=u_2^-\right]+|\mathcal{R}|+\left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|} = -1$. We see opposite signs for the other two mixed derivatives. For $D_1R_{1,2}C_{U_1,U_2}$ we have $|\mathcal{R}| = 2$ and $|\mathcal{R} \cap \mathcal{N}^{\text{cont}}| = 1$. Lastly, for $D_1R_2C_{U_1,U_2}$ we have $|\mathcal{R}| = 1$ and $|\mathcal{R} \cap \mathcal{N}^{\text{cont}}| = 0$. Thus, for both $D_1R_{1,2}C_{U_1,U_2}$ and $D_1R_2C_{U_1,U_2}$ we have $(-1)^{\left[u_2=u_2^-\right]+|\mathcal{R}|+\left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|} = -1$ and $(-1)^{\left[u_2^-=u_2^-\right]+|\mathcal{R}|+\left|\mathcal{R}\cap\mathcal{N}^{\text{cont}}\right|} = 1$. □

Corollary A.3.7.3 (2nd-order derivatives of discrete bivariate reflected copulas)**.** *Let $C_{U_1,U_2}(u_1,u_2)$ be a bivariate copula. Let both U_1 and U_2 be discrete uniform random*

variables. Then, the 2nd-order derivatives of the reflected copulas are given by:

$$
\begin{aligned}
&R_1 C_{U_1,U_2}(u_1,u_2) - R_1 C_{U_1,U_2}\left(u_1^-,u_2\right) - R_1 C_{U_1,U_2}\left(u_1,u_2^-\right) + R_1 C_{U_1,U_2}\left(u_1^-,u_2^-\right) \\
&= -C_{U_1,U_2}(1-u_1,u_2) + C_{U_1,U_2}\left(1-u_1^-,u_2\right) \\
&\quad + C_{U_1,U_2}\left(1-u_1,u_2^-\right) - C_{U_1,U_2}\left(1-u_1^-,u_2^-\right) \\
&R_{1,2} C_{U_1,U_2}(u_1,u_2) - R_{1,2} C_{U_1,U_2}\left(u_1^-,u_2\right) - R_{1,2} C_{U_1,U_2}\left(u_1,u_2^-\right) + R_{1,2} C_{U_1,U_2}\left(u_1^-,u_2^-\right) \\
&= C_{U_1,U_2}(1-u_1,1-u_2) - C_{U_1,U_2}\left(1-u_1^-,1-u_2\right) \\
&\quad - C_{U_1,U_2}\left(1-u_1,1-u_2^-\right) + C_{U_1,U_2}\left(1-u_1^-,1-u_2^-\right) \\
&R_2 C_{U_1,U_2}(u_1,u_2) - R_2 C_{U_1,U_2}\left(u_1^-,u_2\right) - R_2 C_{U_1,U_2}\left(u_1,u_2^-\right) + R_2 C_{U_1,U_2}\left(u_1^-,u_2^-\right) \\
&= -C_{U_1,U_2}(u_1,1-u_2) + C_{U_1,U_2}\left(u_1^-,1-u_2\right) \\
&\quad + C_{U_1,U_2}\left(u_1,1-u_2^-\right) - C_{U_1,U_2}\left(u_1^-,1-u_2^-\right)
\end{aligned}
$$

Proof. This follows immediately from applying the reflection operator to each of the terms. It also follows from (34) in Theorem A.3.7. In all cases we have $\mathcal{N}^{\text{disc}} = \{1,2\}$ and $|\mathcal{R} \cap \mathcal{N}^{\text{cont}}| = 0$. For both $R_1 C_{U_1,U_2}$ and $R_2 C_{U_1,U_2}$ we have $|\mathcal{R}| = 1$. Thus, we have $(-1)^{[u_1=u_1^-]+[u_2=u_2^-]+|\mathcal{R}|} = (-1)^{[u_1^-=u_1^-]+[u_2^-=u_2^-]+|\mathcal{R}|} = -1$ and $(-1)^{[u_1^-=u_1^-]+[u_2=u_2^-]+|\mathcal{R}|} = (-1)^{[u_1=u_1^-]+[u_2^-=u_2^-]+|\mathcal{R}|} = 1$. We have the exact opposite for $R_{1,2} C_{U_1,U_2}$ which concludes the proof. □

A.4 Bivariate Copulas and Derivatives

This section of the appendix shows the derivation of (mixed) derivatives for the bivariate strict Clayton, Frank and Plackett copula. The discrete derivative of any bivariate copula $C_{U_1,U_2}(u_1,u_2)$ is given by:

$$
C_{U_1,U_2}(u_1,u_2) - C_{U_1,U_2}(u_1^-,u_2) - C_{U_1,U_2}(u_1,u_2^-) + C_{U_1,U_2}(u_1^-,u_2^-) \tag{35}
$$

where we define $u_i^- \equiv \lim_{z \to u_i^-} z$ for all $i \in \{1,2\}$.

A.4.1 Clayton Copula

The strict bivariate Clayton copula first appears in Clayton (1978). It also called the Cook-Johnson copula after Cook and Johnson (1981). It is an Archimedean copula, see

e.g. Nelsen (2007). The Clayton copula is given in Table 2:

$$C_{U_1,U_2}(u_1,u_2|\theta) = \left(u_1^{-\theta} + u_2^{-\theta} - 1\right)^{-\theta^{-1}}$$

with $\theta > 0$. This copula becomes the co-monotonic copula when $\theta \to \infty$ and the independence copula when $\theta \to 0$. The strict Clayton copula cannot model negative dependence. The reflected R_1 and R_2 Clayton copulas can model negative dependence however. The relation between θ and Kendall's tau is given by $\tau = \theta(\theta+2)^{-1}$ where τ is Kendall's tau. Assuming that U_1 is continuous, the partial derivatives with respect to u_1 is given by:

$$D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) = u_1^{-\theta-1}\left(u_1^{-\theta} + u_2^{-\theta} - 1\right)^{-\theta^{-1}-1}.$$

If U_2 is continuous, the derivative with respect to u_1 and u_2 is given by:

$$\begin{aligned} D_{u_1,u_2}C_{U_1,U_2}(u_1,u_2|\theta) &= -\theta\left(-\theta^{-1}-1\right)u_2^{-\theta-1}u_1^{-\theta-1}\left(u_1^{-\theta}+u_2^{-\theta}-1\right)^{-\theta^{-1}-2} \\ &= (\theta+1)\,u_2^{-\theta-1}u_1^{-\theta-1}\left(u_1^{-\theta}+u_2^{-\theta}-1\right)^{-\theta^{-1}-2}. \end{aligned} \tag{36}$$

If U_2 is discrete, the derivative with respect to u_1 and u_2 is given by:

$$\begin{aligned} &D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) - D_{u_1}C_{U_1,U_2}(u_1,u_2^-|\theta) \\ &= u_1^{-\theta-1}\left(\left(u_1^{-\theta}+u_2^{-\theta}-1\right)^{-\theta^{-1}-1} - \left(u_1^{-\theta}+\left(u_2^-\right)^{-\theta}-1\right)^{-\theta^{-1}-1}\right). \end{aligned} \tag{37}$$

where we define $u_2^- \equiv \lim_{z\to u_2^-} z$. If both U_1 and U_2 are uniform discrete random variables, then the discrete derivative is simply given by (35) using the formula of the Clayton copula.

We often work with logarithms of copula derivatives. The logarithm of (36) is given by

$$\begin{aligned} \log D_{u_1,u_2}C_{U_1,U_2}(u_1,u_2|\theta) &= \log(\theta+1) + (-\theta-1)(\log u_1 + \log u_2) \\ &\quad + \left(-\theta^{-1}-2\right)\log\left(u_1^{-\theta}+u_2^{-\theta}-1\right). \end{aligned} \tag{38}$$

The logarithm of (37) is given by:

$$\begin{aligned}
&\log\left(D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) - D_{u_1}C_{U_1,U_2}(u_1,u_2^-|\theta)\right) \\
&= (-\theta-1)\log u_1 + \log\left(\left(u_1^{-\theta}+u_2^{-\theta}-1\right)^{-\theta^{-1}-1} - \left(u_1^{-\theta}+\left(u_2^-\right)^{-\theta}-1\right)^{-\theta^{-1}-1}\right). \quad (39)
\end{aligned}$$

A.4.2 Frank Copula

The bivariate Frank copula first appeared in Frank (1979) who shows that the copula is radially symmetric. The Frank copula appears in Table 2 and is given by:

$$C_{U_1,U_2}(u_1,u_2|\theta) = -\frac{1}{\theta}\log\left(1+\frac{\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}{e^{-\theta}-1}\right)$$

with $\theta \in \mathbb{R}\setminus\{0\}$. This copula becomes the co-monotonic copula when $\theta\to\infty$, the counter-monotonic copula when $\theta\to-\infty$ and the independence copula when $\theta\to 0$. Let U_1 be a continuous uniform random variable. Then, the first-order derivative of this bivariate copula with respect to u_1 is given by:

$$\begin{aligned}
&D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) \\
&= -\frac{1}{\theta}\left(1+\frac{\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}{e^{-\theta}-1}\right)^{-1}\frac{\left(e^{-\theta u_2}-1\right)}{\left(e^{-\theta}-1\right)}\cdot -\theta e^{-\theta u_1} \\
&= e^{-\theta u_1}\left(\frac{e^{-\theta}-1}{e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}\right)\frac{\left(e^{-\theta u_2}-1\right)}{\left(e^{-\theta}-1\right)} \\
&= \frac{e^{-\theta u_1}\left(e^{-\theta u_2}-1\right)}{e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}.
\end{aligned}$$

Then, if U_2 is continuous, we can derive the second-order derivative. We take the derivative with respect to $e^{-\theta u_2}-1$ and multiply by $-\theta e^{-\theta u_2}$, as this reduces the number of chain rules that we would have to apply otherwise. The derivative is given by (40). If U_2 is discrete, then the derivative is given by (41) where we define $u_2^- \equiv \lim_{z\to u_2^-} z$. The logarithm of (40) and (41) are given by (43) and (44) respectively.

$$
\begin{aligned}
&D_{u_1,u_2}C_{U_1,U_2}(u_1,u_2|\theta) \\
&= -\theta e^{-\theta u_2}\frac{\partial}{\partial\left(e^{-\theta u_2}-1\right)}\left(\frac{e^{-\theta u_1}\left(e^{-\theta u_2}-1\right)}{e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}\right) \\
&= -\theta e^{-\theta u_2}\left(\frac{e^{-\theta u_1}\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)}{\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)^2}\right. \\
&\qquad \left. -\frac{e^{-\theta u_1}\left(e^{-\theta u_2}-1\right)\left(e^{-\theta u_1}-1\right)}{\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)^2}\right) \\
&= -\theta e^{-\theta(u_1+u_2)}\left(\frac{e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}{\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)^2}\right. \\
&\qquad \left. -\frac{\left(e^{-\theta u_2}-1\right)\left(e^{-\theta u_1}-1\right)}{\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)^2}\right) \\
&= -\theta e^{-\theta(u_1+u_2)}\left(\frac{e^{-\theta}-1}{\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)^2}\right) \qquad (40)
\end{aligned}
$$

$$
\begin{aligned}
&D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) - D_{u_1}C_{U_1,U_2}(u_1,u_2^-|\theta) \\
&= \frac{e^{-\theta u_1}\left(e^{-\theta u_2}-1\right)}{e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)} - \frac{e^{-\theta u_1}\left(e^{-\theta u_2^-}-1\right)}{e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2^-}-1\right)} \\
&= \frac{e^{-\theta u_1}}{\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2^-}-1\right)\right)} \\
&\quad \times\left(\left(e^{-\theta u_2}-1\right)\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2^-}-1\right)\right)\right. \\
&\qquad \left.-\left(e^{-\theta u_2^-}-1\right)\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)\right) \\
&= \frac{e^{-\theta u_1}\left(e^{-\theta}-1\right)\left(e^{-\theta u_2}-e^{-\theta u_2^-}\right)}{\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2^-}-1\right)\right)} \qquad (41) \\
& \qquad (42)
\end{aligned}
$$

$$\begin{aligned}\log D_{u_1,u_2}C_{U_1,U_2}(u_1,u_2|\theta) &= \log\left(\theta\left(1-e^{-\theta}\right)\right) - \theta\left(u_1+u_2\right) \\ &\quad - \log\left(\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)^2\right) \\ &= \log|\theta| + \log\left|1-e^{-\theta}\right| - \theta\left(u_1+u_2\right) \\ &\quad - 2\log\left|\left(e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right)\right| \end{aligned} \tag{43}$$

(43) defined for $\theta \in \mathbb{R} \setminus \{0\}$. Note that we cannot use $\log\left(\theta\left(1-e^{-\theta}\right)\right) = \log\theta + \log\left(1-e^{-\theta}\right)$ without using complex numbers, as $\log\theta$ is complex for $\theta < 0$. We can circumvent this problem by using absolute values. After all, if $\theta < 0$ then $1-e^{-\theta} < 0$ and if $\theta > 0$ then $1-e^{-\theta} > 0$.

$$\begin{aligned}&\log\left(D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) - D_{u_1}C_{U_1,U_2}(u_1,u_2^-|\theta)\right) \\ &= -\theta u_1 + \log\left|e^{-\theta}-1\right| + \log\left|e^{-\theta u_2} - e^{-\theta u_2^-}\right| \\ &\quad - \log\left|e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2^-}-1\right)\right| \\ &\quad - \log\left|e^{-\theta}-1+\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)\right| \end{aligned} \tag{44}$$

A.4.3 Plackett Copula

The Plackett copula arises from the work of Plackett (1965) and appears in Table 2. It is given by:

$$\begin{aligned}&C_{U_1,U_2}(u_1,u_2|\theta) \\ &= \frac{1+(\theta-1)\left(u_1+u_2\right) - \left((1+(\theta-1)(u_1+u_2))^2 - 4u_1u_2\theta(\theta-1)\right)^{\frac{1}{2}}}{2(\theta-1)}\end{aligned}$$

with $\theta > 0, \theta \neq 1$. This copula becomes the co-monotonic copula when $\theta \to \infty$, the counter-monotonic copula when $\theta \to 0$ and the independence copula when $\theta \to 1$. Let U_1 be a continuous uniform random variable. Then, the first-order derivative of the bivariate

Plackett copula with respect to u_1 is given by:

$$
\begin{aligned}
&D_{u_1} C_{U_1,U_2}(u_1, u_2|\theta) \\
&= \frac{1}{2} - \frac{\left((1+(\theta-1)(u_1+u_2))^2 - 4u_1u_2\theta(\theta-1)\right)^{-\frac{1}{2}}}{4(\theta-1)} \\
&\quad \times \left(2(\theta-1)\left(1+(\theta-1)(u_1+u_2)\right) - 4u_2\theta(\theta-1)\right) \\
&= \frac{1}{2} - \frac{1}{2}\left((1+(\theta-1)(u_1+u_2))^2 - 4u_1u_2\theta(\theta-1)\right)^{-\frac{1}{2}} \\
&\quad \times \left(1+(\theta-1)(u_1+u_2) - 2u_2\theta\right) \\
&= \frac{1}{2} - \frac{1}{2}\left((1+(\theta-1)(u_1+u_2))^2 - 4u_1u_2\theta(\theta-1)\right)^{-\frac{1}{2}} \\
&\quad \times \left(1+\theta(u_1-u_2) - u_1 - u_2\right)
\end{aligned}
$$

Let U_2 be a continuous uniform random variable. In order to derive the second-order derivative of the continuous Plackett copula, we need some intermediary results. We rewrite $(1+(\theta-1)(u_1+u_2))^2 - 4u_1u_2\theta(\theta-1)$ to the expression given in (45). Then, we multiply this with $\theta+1$ and obtain the expression given in (46). We rewrite $(1-\theta(u_1-u_2)-u_1-u_2)\left(1+\theta(u_1-u_2)-u_1-u_2\right)$ and obtain the expression given in (47). Multiplying this with $\theta-1$ gives (48). Finally, adding (46) and (48) gives (49), which is used in the derivation of the derivative. The derivative is given by (**??**).

$$
\begin{aligned}
&(1+(\theta-1)(u_1+u_2))^2 - 4u_1u_2\theta(\theta-1) \\
&= 1 + 2(\theta-1)(u_1+u_2) + (\theta-1)^2(u_1+u_2)^2 - 4u_1u_2\theta(\theta-1) \\
&= 1 - 2u_1 - 2u_2 + 2\theta(u_1+u_2) + (\theta^2-2\theta+1)(u_1^2+2u_1u_2+u_2^2) \\
&\quad - 4u_1u_2\theta^2 + 4u_1u_2\theta \\
&= \theta^2\left(u_1^2 - 2u_1u_2 + u_2^2\right) + 2\theta\left(u_1+u_2-u_1^2-u_2^2\right) \\
&\quad + u_1^2 + 2u_1u_2 + u_2^2 - 2u_1 - 2u_2 + 1
\end{aligned} \tag{45}
$$

$$
\begin{aligned}
&(\theta+1)\left((1+(\theta-1)(u_1+u_2))^2-4u_1u_2\theta(\theta-1)\right)\\
&=\theta^3\left(u_1^2-2u_1u_2+u_2^2\right)+2\theta^2\left(u_1+u_2-u_1^2-u_2^2\right)\\
&\quad+\theta\left(u_1^2+2u_1u_2+u_2^2-2u_1-2u_2+1\right)\\
&\quad+\theta^2\left(u_1^2-2u_1u_2+u_2^2\right)+2\theta\left(u_1+u_2-u_1^2-u_2^2\right)\\
&\quad+u_1^2+2u_1u_2+u_2^2-2u_1-2u_2+1\\
&=\theta^3\left(u_1^2-2u_1u_2+u_2^2\right)+\theta^2\left(2u_1+2u_2-u_1^2-u_2^2-2u_1u_2\right)\\
&\quad+\theta\left(2u_1u_2-u_1^2-u_2^2+1\right)\\
&\quad+u_1^2+2u_1u_2+u_2^2-2u_1-2u_2+1
\end{aligned}
\tag{46}
$$

$$
\begin{aligned}
&(1-\theta(u_1-u_2)-u_1-u_2)\,(1+\theta(u_1-u_2)-u_1-u_2)\\
&=(1-u_1-u_2)^2-\theta^2\,(u_1-u_2)^2\\
&=1-u_1-u_2-u_1+u_1^2+u_1u_2-u_2+u_1u_2+u_2^2\\
&\quad-\theta^2\left(u_1^2-2u_1u_2+u_2^2\right)\\
&=-\theta^2\left(u_1^2-2u_1u_2+u_2^2\right)+u_1^2+u_2^2+2u_1u_2-2u_1-2u_2+1
\end{aligned}
\tag{47}
$$

$$
\begin{aligned}
&(\theta-1)\,(1-\theta(u_1-u_2)-u_1-u_2)\,(1+\theta(u_1-u_2)-u_1-u_2)\\
&=-\theta^3\left(u_1^2-2u_1u_2+u_2^2\right)+\theta\left(u_1^2+u_2^2+2u_1u_2-2u_1-2u_2+1\right)\\
&\quad+\theta^2\left(u_1^2-2u_1u_2+u_2^2\right)-\left(u_1^2+u_2^2+2u_1u_2-2u_1-2u_2+1\right)
\end{aligned}
\tag{48}
$$

$$
\begin{aligned}
&(\theta-1)\left(1-\theta(u_1-u_2)-u_1-u_2\right)\left(1+\theta(u_1-u_2)-u_1-u_2\right)\\
&\quad+(\theta+1)\left(1+(\theta-1)(u_1+u_2)\right)^2-4u_1u_2\theta(\theta-1)\\
&=\theta^3\left(u_1^2-2u_1u_2+u_2^2\right)+\theta^2\left(2u_1+2u_2-u_1^2-u_2^2-2u_1u_2\right)\\
&\quad+\theta\left(2u_1u_2-u_1^2-u_2^2+1\right)\\
&\quad+u_1^2+2u_1u_2+u_2^2-2u_1-2u_2+1\\
&\quad-\theta^3\left(u_1^2-2u_1u_2+u_2^2\right)+\theta\left(u_1^2+u_2^2+2u_1u_2-2u_1-2u_2+1\right)\\
&\quad+\theta^2\left(u_1^2-2u_1u_2+u_2^2\right)-\left(u_1^2+u_2^2+2u_1u_2-2u_1-2u_2+1\right)\\
&=\theta^2\left(2u_1+2u_2-4u_1u_2\right)+\theta\left(4u_1u_2-2u_1-2u_2+2\right)\\
&=2\theta\left(\theta\left(u_1-2u_1u_2+u_2\right)+\left(2u_1u_2-u_1-u_2+1\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
&D_{u_1,u_2}C_{U_1,U_2}(u_1,u_2|\theta)\\
&=\frac{1}{4}\left(\left(1+(\theta-1)(u_1+u_2)\right)^2-4u_1u_2\theta(\theta-1)\right)^{-\frac{3}{2}}\\
&\quad\times\left(2(\theta-1)\left(1+(\theta-1)(u_1+u_2)\right)-4u_1\theta(\theta-1)\right)\\
&\quad\times\left(1+\theta(u_1-u_2)-u_1-u_2\right)\\
&\quad-\frac{1}{2}\left(\left(1+(\theta-1)(u_1+u_2)\right)^2-4u_1u_2\theta(\theta-1)\right)^{-\frac{1}{2}}(-\theta-1)\\
&=\frac{1}{2}\left(\left(1+(\theta-1)(u_1+u_2)\right)^2-4u_1u_2\theta(\theta-1)\right)^{-\frac{3}{2}}\\
&\quad\times\left((\theta-1)\left(1-\theta(u_1-u_2)-u_1-u_2\right)\right.\\
&\quad\times\left(1+\theta(u_1-u_2)-u_1-u_2\right)\\
&\quad\left.+(\theta+1)\left(\left(1+(\theta-1)(u_1+u_2)\right)^2-4u_1u_2\theta(\theta-1)\right)\right)\\
&=\frac{\theta\left(\theta\left(u_1-2u_1u_2+u_2\right)+\left(2u_1u_2-u_1-u_2+1\right)\right)}{\left(\left(1+(\theta-1)(u_1+u_2)\right)^2-4u_1u_2\theta(\theta-1)\right)^{\frac{3}{2}}}
\end{aligned}
\tag{49}
$$

Now assume that U_2 is discrete. Then, the mixed derivative is given by (50).

$$\begin{aligned}
&\left(D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) - D_{u_1}C_{U_1,U_2}(u_1,u_2^-|\theta)\right) \\
&= \frac{1}{2}\left(\frac{1+\theta(u_1-u_2^-)-u_1-u_2^-}{\left(\left(1+(\theta-1)(u_1+u_2^-)\right)^2-4u_1u_2^-\theta(\theta-1)\right)^{\frac{1}{2}}}\right. \\
&\quad \left. - \frac{1+\theta(u_1-u_2)-u_1-u_2}{\left((1+(\theta-1)(u_1+u_2))^2-4u_1u_2\theta(\theta-1)\right)^{\frac{1}{2}}}\right)
\end{aligned} \tag{50}$$

The logarithm of $D_{u_1,u_2}C_{U_1,U_2}(u_1,u_2|\theta)$ is given by (51)

$$\begin{aligned}
&\log D_{u_1,u_2}C_{U_1,U_2}(u_1,u_2|\theta) \\
&= \log\theta + \log\left(\theta\left(u_1-2u_1u_2+u_2\right)+\left(2u_1u_2-u_1-u_2+1\right)\right) \\
&\quad - \frac{3}{2}\log\left(\left(1+(\theta-1)(u_1+u_2)\right)^2-4u_1u_2\theta(\theta-1)\right)
\end{aligned} \tag{51}$$

The logarithm of (50) is given by (52).

$$\begin{aligned}
&\log\left(D_{u_1}C_{U_1,U_2}(u_1,u_2|\theta) - D_{u_1}C_{U_1,U_2}(u_1,u_2^-|\theta)\right) \\
&= \log\left(\frac{1+\theta(u_1-u_2^-)-u_1-u_2^-}{\left(\left(1+(\theta-1)(u_1+u_2^-)\right)^2-4u_1u_2^-\theta(\theta-1)\right)^{\frac{1}{2}}}\right. \\
&\quad \left. - \frac{1+\theta(u_1-u_2)-u_1-u_2}{\left((1+(\theta-1)(u_1+u_2))^2-4u_1u_2\theta(\theta-1)\right)^{\frac{1}{2}}}\right) - \log 2
\end{aligned} \tag{52}$$

A.5 Common Bayesian Network Classifiers

Section 4.2.2 explains how Bayesian Networks can be used as classifiers. If Y is a categorical dependent variable and $\mathbf{X} = (X_1, \ldots, X_n)$ is a vector of n random variables, then we need to model the joint probability function $f_{Y,\mathbf{X}}(y,\mathbf{x})$ by (6):

$$f_{Y,\mathbf{X}}(y,\mathbf{x}) = f_{Y|\mathbf{Pa}_Y}\left(y|\mathbf{pa}_Y\right)\prod_{i\in\mathcal{N}} f_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}_{X_i}\right)$$

where $y \in \mathcal{K}$, $\mathbf{Pa}_Y$ and $\mathbf{Pa}_{X_i}$ refer to the vector of random variables that are parents of Y and X_i respectively and $\mathbf{pa}_Y$ and $\mathbf{pa}_{X_i}$ are their corresponding realisation vectors.

A commonly used BN classifier is the naïve Bayes (NB) classifier. Here, the dependent variable Y does not have any parents, meaning $f_{Y|\mathbf{Pa}_Y}(y|\mathbf{pa}_Y) = f_Y(y)$. Moreover, all explanatory variables are assumed to be independent *and* are only allowed to have Y as their parent, meaning that $f_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}_{X_i}\right) = f_{X_i|Y}(x_i|y)$. A graphical representation of an NB classifier with three explanatory variables is given in Figure A.5.

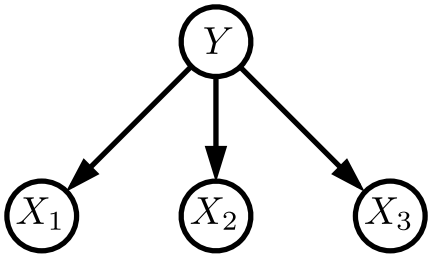

Figure 3: Example of a graph structure used by a Naïve Bayes (NB) classifier with three explanatory variables X_1, X_2, X_3. Each explanatory variable must have an incoming arc from the dependent variable Y. Arcs between the explanatory variables are not allowed. The graph should be directed acyclic.

The joint probability function belonging to the random variables in Figure A.5 is given by:

$$f_{Y,X_1,X_2,X_3}(y, x_1, x_2, x_3) = f_Y(y) f_{X_1|Y}(x_1|y)\, f_{X_2|Y}(x_2|y)\, f_{X_3|Y}(x_3|y)$$

Another BN classifier is the tree-augmented naïve Bayes (TAN) classifier. The underlying DAG belonging to a TAN classifier is an NB structure with the addition of directed arcs between the explanatory variables. An example is given in Figure A.5.

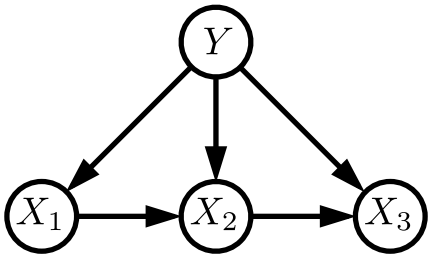

Figure 4: Example of a graph structure used by a Tree-augmented Naïve Bayes (TAN) classifier with three explanatory variables X_1, X_2, X_3. Each explanatory variable must have an incoming arc from the dependent variable Y. Arcs between the explanatory variables are allowed. The graph should be directed acyclic.

The joint probability function belonging to the random variables in Figure A.5 is given by:

$$f_{Y,X_1,X_2,X_3}(y, x_1, x_2, x_3) = f_Y(y) f_{X_1|Y}(x_1|y)\, f_{X_2|Y,X_1}(x_2|y, x_1)\, f_{X_3|Y,X_2}(x_3|y, x_2)$$

A generalised BN classifier does not necessarily contain a naïve Bayes structure. An example is given in Figure A.5. The joint probability belonging to the random variables in Figure A.5 is given by:

$$f_{Y,X_1,X_2,X_3}(y,x_1,x_2,x_3) = f_{X_3}(x_3)\, f_{Y|X_3}(y|x_3)\, f_{X_1|Y}(x_1|y)\, f_{X_2|Y,X_1}(x_2|y,x_1)$$

Note that $f_{X_3}(x_3)$ does not need to be determined when we are trying to predict values for Y.

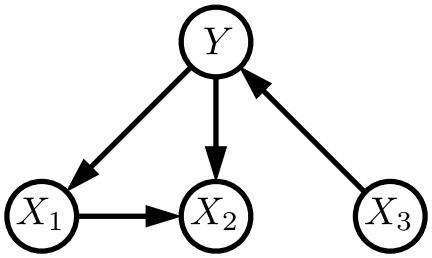

Figure 5: Example of a graph structure used by a generalised Bayesian Network classifier with three explanatory variables X_1, X_2, X_3. Arcs can go from the explanatory variables to Y and vice versa. Arcs between the explanatory variables are allowed. The graph should be directed acyclic.

B Examples

B.1 Example for Theorem 4.3.2

The following serves as an illustration of Theorem 4.3.2. We will derive the joint probability function for some multivariate distribution. Let $X_1 \sim$ Weibull, $X_4 \sim$ Pareto and X_2 and X_3 be discrete random variables. According to Sklar's Theorem, see Theorem 4.3.1, there exists a 4-dimensional copula such that

$$F_{X_1,X_2,X_3,X_4}(x_1,x_2,x_3,x_4) = C_{X_1,X_2,X_3,X_4}\left(F_{X_1}(x_1), F_{X_2}(x_2), F_{X_3}(x_3), F_{X_4}(x_4)\right).$$

This copula is not uniquely determined by its marginals, as F_{X_2} and F_{X_3} are discrete. However, the joint pdf f_{X_1,X_2,X_3,X_4} can still be derived according to Theorem 4.3.2, see (8).

We will use the same notation as in Theorem 4.3.2. Now, we have $\mathcal{N}^{\text{cont}} = \{1,4\}$, $\mathcal{N}^{\text{disc}} = \{2,3\}$ and $\mathcal{N} = \mathcal{N}^{\text{cont}} \cup \mathcal{N}^{\text{disc}} = \{1,4,2,3\}$. Evidently, $\mathcal{N}^{\text{cont}} \cap \mathcal{N}^{\text{disc}} = \emptyset$, $n^{\text{cont}} = n^{\text{disc}} = 2$

and $n = 4$. The n^{disc}-ary Cartesian product $\mathcal{U}$ is given by: $\mathcal{U} = \left\{F_{X_2}\left(x_2^-\right), F_{X_2}\left(x_2\right)\right\}$ $\times \left\{F_{X_3}\left(x_3^-\right), F_{X_3}\left(x_3\right)\right\}$. Thus we have that $\mathcal{U}$ contains the following four 2-tuples: $\left(F_{X_2}\left(x_2^-\right), F_{X_3}\left(x_3^-\right)\right)$, $\left(F_{X_2}\left(x_2^-\right), F_{X_3}\left(x_3\right)\right)$, $\left(F_{X_2}\left(x_2\right), F_{X_3}\left(x_3^-\right)\right)$ and $\left(F_{X_2}\left(x_2\right), F_{X_3}\left(x_3\right)\right)$.

Note that we can write $\left(F_{X_2}\left(x_2^*\right), F_{X_3}\left(x_3^*\right)\right) \in \mathcal{U}$ for $\left(F_{X_j}\left(x_j^*\right)\right)_{j\in\mathcal{N}^{\text{disc}}} \in \mathcal{U}$ as seen in (8). Then, we can rewrite joint distribution function of $\mathbf{X}$ as:

$$f_{\mathbf{X}}(\mathbf{x}) = \sum_{\left(F_{X_2}\left(x_2^*\right), F_{X_3}\left(x_3^*\right)\right)\in\mathcal{U}} (-1)^{\left[F_{X_2}\left(x_2^-\right)=F_{X_2}\left(x_2^*\right)\right]+\left[F_{X_3}\left(x_3^-\right)=F_{X_3}\left(x_3^*\right)\right]} \times$$
$$D_{1,4}C_{U_1,U_2,U_3,U_4}\left(F_{X_1}\left(x_1\right), F_{X_4}\left(x_4\right), F_{X_2}\left(x_2^*\right), F_{X_3}\left(x_3^*\right)\right) f_{X_1}\left(x_1\right) f_{X_4}\left(x_4\right).$$

By evaluating the Iverson brackets, we have the following expression for the joint pdf:

$$\begin{aligned} f_{\mathbf{X}}(\mathbf{x}) = f_{X_1}\left(x_1\right) f_{X_4}\left(x_4\right) \big(& D_{1,4}C_{U_1,U_2,U_3,U_4}\left(F_{X_1}\left(x_1\right), F_{X_4}\left(x_4\right), F_{X_2}\left(x_2^-\right), F_{X_3}\left(x_3^-\right)\right) \\ & - D_{1,4}C_{U_1,U_2,U_3,U_4}\left(F_{X_1}\left(x_1\right), F_{X_4}\left(x_4\right), F_{X_2}\left(x_2^-\right), F_{X_3}\left(x_3\right)\right) \\ & - D_{1,4}C_{U_1,U_2,U_3,U_4}\left(F_{X_1}\left(x_1\right), F_{X_4}\left(x_4\right), F_{X_2}\left(x_2\right), F_{X_3}\left(x_3^-\right)\right) \\ & + D_{1,4}C_{U_1,U_2,U_3,U_4}\left(F_{X_1}\left(x_1\right), F_{X_4}\left(x_4\right), F_{X_2}\left(x_2\right), F_{X_3}\left(x_3\right)\right) \big) \end{aligned}$$

One can use any multivariate copula to construct a joint For instance, consider the 4-variate Clayton copula with parameter θ given by

$$C_{U_1,U_2,U_3,U_4}\left(u_1,u_2,u_3,u_4|\theta\right) = \left(u_1^{-\theta} + u_2^{-\theta} + u_3^{-\theta} + u_4^{-\theta} - 3\right)^{-\theta^{-1}}.$$

See Nelsen (2007) for more information. The second order derivative of this copula with respect to u_1 and u_4 is given by:

$$(1+\theta)u_1^{-(\theta+1)}u_4^{-(\theta+1)}\left(u_1^{-\theta} + u_2^{-\theta} + u_3^{-\theta} + u_4^{-\theta} - 3\right)^{-\left(\frac{1+2\theta}{\theta}\right)}$$

which can be used in $f_{\mathbf{X}}(\mathbf{x})$.

C List of Symbols

The following list of symbols is consistent throughout this thesis.

$\mathcal{A}$ set of arcs

A argument-addition (AA) operator, see Appendix A.3.3

$\mathcal{B}$ Bayesian network (BN), see Definition 4.2.2

C n-dimensional copula function, see Definition A.2.1

D Euler's differential notation

$\mathcal{D}$ data, usually containing d realisations

$F_{\mathbf{X}}$ joint cumulative distribution function over vector with random variables $\mathbf{X}$

F_{X_i} cumulative distribution function over random variable X_i

$f_{\mathbf{X}}$ joint probability function over vector with random variables $\mathbf{X}$

f_{X_i} probability function over random variable X_i

$\mathcal{G}$ graph, often directed acyclic. see Definition 4.2.1

$\mathcal{K}$ set of all k possible realisations of Y

$\mathcal{N}$ set of nodes

$\mathbb{N}$ set of natural numbers *including* 0: $\{0, 1, 2, \ldots\}$

$\mathbb{N}^+$ set of natural numbers *excluding* 0: $\{1, 2, \ldots\}$

$\emptyset$ the empty set $\{\}$

$\varnothing$ empty variable

P probability measure

$\mathbf{Pa}_{X_j}$ parent vector of X_j, containing zero or more random variables

$\mathbf{Pa}_{X_j}^{\text{cat}}$ parent vector of X_j containing only categorical random variables *including* Y

$\mathbf{Pa}_{X_j}^{\text{ncat}}$ parent vector of X_j containing only non-categorical random variables

R reflection operation, see Appendix A.3.2 $\mathbb{R}$ set of real numbers $(-\infty, \infty)$

$\mathbb{R}^d$ d-dimensional set of real numbers

$\overline{\mathbb{R}}$ extended set of real numbers $\mathbb{R} \cup \{-\infty, \infty\}$

$\overline{\mathbb{R}}^d$ d-dimensional extended set of real numbers

u_i^- defined as $\lim_{z \to u_i^-} z$

$\mathbf{X}$ vector with random variables

X_i random variable i

x_i realisation of X_i

Y dependent random variable, categorical

y realisation of Y

$\hat{y}$ estimate of Y

Θ set of (cumulative) probability distributions

θ parameter of single-parameter copula function C

$\hat{\theta}$ estimate of θ

$[a = b]$ Iverson brackets, evaluates to 1 if a = b and 0 otherwise.