eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifizialaren Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Contributions to Vine-Copula Modeling

by

Diana María Carrera Soto

Supervised by Dr. Roberto Santana and Dr. José A. Lozano

Donostia - San Sebastián, February 2022

Agradecimientos

He repasado y reescrito estas líneas varias veces, porque al leerlo nunca me pareció que expresara toda la gratitud que siento—las palabras no parecían suficientes. Incluso he derramado algunas lágrimas recordando momentos de mis años de trabajo en esta tesis: son tantas las personas a las que estoy agradecida.

Ha sido un proceso arduo (por decirlo de alguna manera), del que he aprendido mucho. He pasado por momentos difíciles y también muy felices. Ahora mismo, mientras escribo, el sentimiento que prevalece es de orgullo por todo lo que hemos conseguido. Hablo en plural, porque sin un grupo específico de personas, todo esto no habría sido posible. Deseo expresar mi más profunda gratitud a:

Mis tutores, Roberto Santana y mi 'profe', José A. Lozano, por guiarme estos años, lo cual fue mucho más allá de nuestra tesis. Nuestra colaboración fue perfecta desde el primer día, sobre todo gracias a la diversidad entre nosotros, permitiéndonos aportar diferentes puntos de vista. Roberto, siempre muy meticuloso y estructurado, me enseñaste el difícil arte de escribir un Resumen, a ser más estructurada con el trabajo y a ponerme siempre en el lugar del lector. Lozano, de ti aprendí a pensar críticamente, a enfocar los problemas desde diferentes ángulos y a ser concisa. Muchas gracias a los dos por compartir vuestros conocimientos, por ser tan pacientes y por vuestro inmenso apoyo, tanto profesional como personal.

A todos los miembros del Grupo de Sistemas Inteligentes, por ser unos compañeros maravillosos. Un grupo en el que nos apoyamos los unos a los otros, y en el que prevalece el conocimiento y el buen ambiente.

Johnny, mi profesor de Inglés, por toda tu persistencia y flexibilidad; por ayudarme con cada presentación, haciéndome repetirla una y otra vez. Sin ti, este agradecimiento seguramente estaría escrito en "Spanglish". Gracias por prestarme tu oído, fuiste mi confidente.

Letti y Ekhiñe, no se imaginan la suerte que tengo por haberlas conocido. Gracias por todos los momentos que hemos vivido juntas, mis compañeras de viaje; por cada anécdota (algunas quedarán para siempre entre nosotras). Principalmente, quiero dales las gracias por haberme acogido en nuestro SU grupo de amigos.

Itzi, Unai, Sari, y Mire por distraerme del trabajo—vital a veces. Por los viajes por carretera, las cervezas, los bailes y las artes culinarias. Sois vosotros los que me habéis convertido en el *foodie* que soy hoy.

Roger, Ingrid, Efrem, mis amigos de siempre, amigos para toda la vida. Ustedes me empujaron a embarcarme en esta aventura, sabiendo que supondría estar alejados, "que era una oportunidad increíble que no podía perder". Nuestras llamadas siempre me llenaron de alegría y de energía renovada para poder seguir adelante.

Mi extensa familia (abuelos, tíos, primos y hermano), por apoyarme en todo lo que han podido, por estar siempre pendientes, y por hacerme sentir lo orgullosos que están de mí. Ustedes son mis raíces, de donde vengo y hacia donde voy.

Juanca, mi pareja, llegaste a mi vida en la etapa más intensa de mi trabajo. Eres la persona

que estuvo en las noches de insomnio y en los días de incertidumbre y frustración. Gracias por tu amor en las buenas y en las malas.

Mi padre, el mejor del mundo. Aunque estabas lejos, te sentía tan cercano. Gracias por los consejos y las verdades, por no dejar que me rinda. Sin tu educación, no habría llegado hasta aquí.

Hermana mía, son tantas cosas las que tengo que agradecerte. Gracias por mantener esta tesis en marcha y por toda tu ayuda. Por compartir todo lo que tienes sin pensarlo dos veces. Por hacer que yo, tu hermana mayor, me sienta lo más importante. Eres mi compañera de vida.

A mi madre, a quien he dejado para el final porque sin ti esto no hubiera sido posible. Gracias por introducirme en el mundo de la ciencia, por estar detrás de cada paso que he dado, y por darlos conmigo. A falta de palabras para describir lo que significas para mí y para la realización de este trabajo final, dedico esta tesis enteramente a ti.

A mi mamá

Acknowledgments

I have gone over and rewritten this repeatedly, because when reading it, it never seemed to me that it expressed all the gratitude that I feel—words just didn't seem to be enough. I even shed a few tears remembering moments from my years of work on this thesis: there have been so many people to whom I am grateful.

It has been an arduous process (to say the least), from which I have learned plenty. I have gone through difficult times as well as very happy ones. Just now, as I write this, the prevailing feeling is pride in all that we have accomplished. I speak in plural, because without a specific group of people, all this would not have been achievable. Let my deepest gratitude go to:

My mentors, Roberto Santana and my 'profe', Jose A. Lozano, for their guidance, which went far beyond our thesis. Our collaboration was spot-on from day one, mainly thanks to the diversity among us, allowing us to bring different points of view to the table. Roberto, always very meticulous and structured, you taught me the challenging art of writing an Abstract, how to be more structured with work and to always put myself in the reader's shoes. Lozano, from you I learned how to think critically, to approach problems from every angle and to be concise. Many thanks to you both for sharing your knowledge, for being so patient and for your immense support, both professionally and personally.

The entire Intelligent Systems Group, for being wonderful companions. A group where we supported each other, and where knowledge and good vibes prevailed.

Johnny, my English teacher, for all your persistence and flexibility; for helping me with each presentation, making me repeat it over and over again. Without you, this acknowledgment would surely be written in "Spanglish". Thank you for lending me your ear, you were my confidant.

Letti and Ekhiñe, who just can't imagine how lucky I feel to have met them. Thank you for all the moments we lived together, my travel companions; for each anecdote (some will remain forever between us). Mainly, I want to thank you for having welcomed me into our YOUR group of friends.

Itzi, Unai, Sari, and Mire for distracting me from work—vital at times. For the road trips, the beers, the dancing, and the culinary arts. It is you who turned me into the foodie I am today.

Roger, Ingrid, Efrem, my friends from my hometown, friends for life. You pushed me to embark on this adventure, knowing that it would mean being apart, "that this was an incredible opportunity I couldn't miss". Our calls always filled me with joy and renewed energy to be able to carry on.

My vast family (grandparents, uncles, cousins, and brother), for supporting me in whatever they could, for always being on the lookout, and for making me feel how proud they are of me. You are my roots, from where I come and to where I am going.

Juanca, my husband, who came into my life in the most intense stage of my work. You, the person was there throughout sleepless nights and in the days of uncertainty and frustration. Thank you for your love through thick and thin.

My dad, the best in the world. Although far away, you seemed so close. For the advice and for the truths, for never letting me give up. Without your upbringing, I would not have gotten to where I am today.

My sister, there is so much I must thank you for. For keeping this thesis going and all your help. For sharing everything you have without thinking twice. For making me, your big sister, feel like the most important thing. You are my partner in crime for life.

My mother, whom I have saved for last because without you this simply would not have been possible. Thank you for introducing me to the world of science, for being behind every step I have taken, and for taking them with me. For the lack of words to describe what you mean to me and the realization of this final work, I dedicate this thesis entirely to you.

To my mother

Abstract

Regular vine-copula models (R-vines) are a powerful statistical tool for modeling the dependence structure of multivariate distribution functions. In particular, they allow modeling different types of dependencies among random variables independently of their marginal distributions, which is deemed the most valued characteristic of these models. In this thesis, we investigate the theoretical properties of R-vines for representing dependencies and extend their use to solve supervised classification problems. We focus on three research directions.

In the first line of research, the relationship between the graphical representations of R-vines and Bayesian polytree networks is analyzed in terms of how conditional pairwise independence relationships are represented by both models. In order to do that, we use an extended graphical representation of R-vines in which the R-vine graph is endowed with further expressiveness, being possible to distinguish between edges representing independence and dependence relationships. Using this representation, a separation criterion in the R-vine graph, called R-separation, is defined. The proposed criterion is used in designing methods for building the graphical structure of polytrees from that of R-vines, and vice versa. Moreover, possible correspondences between the R-vine graph and the associated R-vine copula as well as different properties of R-separation are analyzed.

In the second research line, we design methods for learning the graphical structure of R-vines from dependence lists. The main challenge of this task lies in the extremely large size of the search space of all possible R-vine structures. We provide two strategies to solve the problem of learning R-vines that represent the largest number of dependencies in a list. The first approach is a 0-1 linear programming formulation for building truncated R-vines with only two trees. The second approach is an evolutionary algorithm, which is able to learn complete and truncated R-vines. Experimental results show the success of this strategy in solving the optimization problem posed.

In the third research line, we introduce a supervised classification approach where the dependence structure of the problem features is modeled through R-vines. The efficacy of these classifiers is validated in a mental decoding problem and in an image recognition task. While R-vines have been extensively applied in fields such as economics, finance and statistics, only recently have they found their place in classification tasks. This contribution represents a step forward in understanding R-vines and the prospect of extending their use to other machine learning tasks.

Contents

Introduction

In the few last years, there has been a growing interest in modeling the dependence of multivariate distributions using copulas [97]. Thanks to the possibility of modeling different types of dependencies among random variables independently of their marginal distributions, copulas have been successfully used in applications of different domains, and these achievements have led to further advances in both theory and practice. In artificial intelligence, for instance, we can find copula-based contributions in classification [29, 41, 95, 125, 128, 135], regression [30, 98, 132] and optimization [3, 34, 38, 121, 122, 136] tasks.

Copulas are distribution functions with uniformly distributed margins [97]. The relevance of copulas in probabilistic modeling is summarized in Sklar's theorem [120], which establishes that any multivariate distribution function can be decomposed into its marginal distributions and a multivariate copula that describes the dependence structure among them.

Despite the generality of the copula-based framework, building high dimensional copulas is a challenging problem [1]. Although there is a variety of bivariate copula families that cover a wide range of different types of dependencies, the range of standard higher-variate copula families is quite limited [75]. Another drawback of these copulas is that they express the same type of dependence among random variables, which could be misleading in real-world data-driven applications where the selected variables interact differently [36].

Pair-copula constructions (PCCs) [74] overcome these disadvantages by using (conditional) bivariate copulas, called pair-copulas, as building blocks to describe a multivariate distribution. Developed in [8, 9, 84], regular vine-copula models (R-vines) organize such PCCs graphically by means of a hierarchy of nested trees with undirected edges that constitute the R-vine graph. This representation facilitates the identification of the required pair-copulas associated with the edges of the trees. Moreover, by combining pair-copulas of different families, R-vine copulas are able to model a wide variety of dependencies in multivariate data with regard to symmetry and tail dependence of the bivariate distribution [36]. Thereby, these models combine the strengths of multivariate copula modeling, namely the separation of marginal and dependence modeling as well as the diversity of pair-copula families, which can be mixed in the same PCC. Moreover, each pair-copula can be chosen independently from the others [19].

R-vines are the research object of the present thesis, which focuses on the three directions presented in the following paragraphs.

The first line of research concerns the relationship between the graphical representations of R-vines and polytrees—a particular type of Bayesian networks (BNs) [101] with only one undirected path joining any two nodes in the graph (i.e., singly connected). The possible connection between both structures is analyzed. We focus on pairwise graph separations and non-separations encoded in the R-vine graph that could correspond with the set of (un)conditional[1] pairwise (in)dependencies[2] of the dependence model associated with the polytree graph, and vice versa.

[1] (un)conditional denotes unconditional and conditional.
[2] (in)dependence denotes independence and dependence.

Graphical models (GMs) [86], are a powerful statistical tool for describing the relationships of (un)conditional (in)dependence in a set of random variables in an intuitive way using graphs. To deduce such relationships from the topology of the graph, graphical separation criteria are used. These criteria are the rules for understanding how (un)conditional (in)dependencies of a set of random variables are encoded in a graph. The definition of these concepts strongly depends on the type of graph, for instance, *D-separation* for directed graphs used in BNs, and *U-separation* for undirected graphs used in Markov networks (MNs) [27]. Since R-vines are relatively recent models, their own graphical separation criterion has not yet been defined. A definition of this concept is needed to carry out the analysis of the connection between the graphical representations of R-vines and polytrees. To this end, in this thesis a concept of graphical separation for R-vines, called R-separation, is defined.

When defining the concept of R-separation, two aspects should be taken into account: that the edges of R-vine graphs illustrate the required pair-copulas only [7], and also that the pair-copulas can represent independence relationships (e.g., through the Product copula). In order to disambiguate whether the pair-copula relationship is that of dependence or independence, we use an extended representation, in which the R-vine graph is endowed with further expressiveness with edges indicating the type of relationship they represent [83]. The proposed criterion is used in designing methods for building the graphical structure of polytrees from that of R-vines, and vice versa, and their use is illustrated through examples. Moreover, a theorem on possible correspondences between R-vine graphs and the associated R-vine copulas is presented. In addition, properties of R-separation such as symmetry, decomposition, contraction, intersection, (strong/weak) union, and (strong/weak) transitivity are analyzed.

The second line of research focuses on designing methods for learning the graphical structure of R-vines from dependence lists. Attributable to the vast search space, the task of finding the globally optimal R-vine graph for a high-dimensional dataset is challenging, since the number of possible structures grows extremely fast as the number of nodes increases. Indeed, model selection is a challenge even for eight or nine nodes [91]. This scenario makes unfeasible to evaluate all possible graphical structures and select the optimal one.

Learning the graph structure of R-vines is an active research topic and different methods have been proposed in the literature (see, among others, [1,36,42,82]). These methods are mainly focused on greedy heuristics that learn directly from data. The rationale behind these algorithms is to build R-vines that cover the strongest dependencies in the first trees, since they can be estimated with more accuracy. Non-greedy heuristics based on fit indices and the Monte Carlo tree search have been also studied in [18,28]. It is usual that the R-vine learning heuristics perform the tree selection and the estimation of the corresponding pair-copulas at the same time, i.e., the subsequent tree in the hierarchy can not be constructed until the pair-copulas of the previous tree have been selected. In order to reduce the learning cost of R-vine models, the truncation strategy proposed in [17] allows building only the first trees, assuming conditional independence in the last trees.

A different avenue is explored in this work. We design methods for learning the graph structure of R-vines from lists of (un)conditional pairwise dependence relationships (or simply, dependence lists), instead of a dataset, as has been done up to now. This is a common practice in other graphical models, such as Bayesian networks and Markov networks, where the dependence lists can be supplied by experts in the application domain [27]. Specifically, the research question we deal with is an optimization problem which aims to build R-vine graphs that incorporate the largest number of dependence relationships given in a dependence list.

Two approaches are proposed for solving the optimization problem posed. The first approach is a 0-1 linear programming formulation that builds truncated R-vine graphs with only two trees. The second approach is a genetic algorithm (GA) [71] that is able to learn complete and truncated R-vine graphs. The designed GA uses crossover and mutation operators specifically designed to

ensure that the resulting solutions are feasible. Furthermore, numerical experiments are carried out to assess the effectiveness of the designed evolutionary algorithm in solving the optimization problem posed.

The third line of research focuses on extending the use of R-vines to solve supervised classification tasks. These models are increasingly popular in classification applications, mainly because they can accommodate a wide range of complex dependencies by using copula-pairs from different families. Supervised classification is concerned with assigning a new sample (example or instance) to a class based on the predictor variables (features) of the classification problem. Typically, the algorithms utilized to perform this task use training data to model the feature distribution of each class. Relevant references of these methods are Bayesian classifiers such as Naive Bayes (NB) [44, 85] and Tree Augmented Naive Bayes (TAN) [52], which assume severe conditional independence constraints among the predictor variables.

Motivated by the flexibility of R-vines to model the dependence structure of multivariate distributions, this thesis proposes to incorporate them in the design of probabilistic classifiers. This property could be particularly useful in applications where the dependence patterns among the predictor variables are quite diverse. This is the case of the two real-world classification tasks addressed in this thesis, namely the Mind Reading Problem (MRP) [96] and the Dune Classification Problem (DCP) [5]. The potential existence of complex and diverse dependence patterns among the predictor variables in these problems encourage us to select R-vines as a promising candidate to approach them in the first place.

Based on the motivations outlined earlier, the main aim of this dissertation is to investigate theoretical properties of R-vines for representing dependencies and extend the use of these models to solve supervised classification problems.

Subsidiary aims derived from the main goal are the following:

1. To formulate a graphical separation criterion for R-vines, and to analyze the relationship between the graph representations of R-vines and polytrees.
2. To design strategies for learning the graph structure of R-vines from dependence lists.
3. To design a supervised classification approach based on R-vines.

The research contributions of this thesis are summarized as follows:

1. Definition of R-separation, a graphical separation criterion for R-vines. In addition, possible dependence maps between the R-vine graph and the associated R-vine copula, as well as different properties of R-separation are analyzed. Moreover, algorithms for building the graph structure of polytrees from that of R-vines, and vice versa, using R-separation, are designed. These algorithms allow to analyze the possible dependence maps between both models.
2. Design of optimization strategies for learning the graphical structure of R-vines from dependence lists.
3. Introduction of a classification approach where the dependence structure of the features is modeled through R-vines. The designed classifiers are successfully applied to the MRP and DCP.

This thesis is divided into four chapters as follows. First, in Chapter 1, the necessary mathematical context on copulas and R-vines is provided. Next, the contributions of this dissertation are presented. Chapter 2 studies the connection between the graphical representation of R-vines and polytrees, defines the concept of R-separation, and proposes algorithms for building the graphical structure of a polytree from an R-vine graph, and vice versa. Chapter 3 exposes a 0-1 linear programming formulation and an evolutionary algorithm for learning R-vines from dependence

lists. Chapter 4 introduces supervised classifiers based on R-vines, which then are applied to the MRP and DCP. Finally, the general conclusions of this thesis, the lines of possible future work, and the list of publications produced during this dissertation are presented.

Chapter 1

Mathematical Context

This chapter provides the mathematical framework for understanding the document, including notation and definitions. First, the concept of copula and Sklar's theorem are presented. Then, a review of pair-copula constructions (PCCs) and R-vines, which is the probabilistic graphical model derived from PCCs, is provided. We proceed by presenting some measures of dependence, examples of families of bivariate copulas, a strategy for the selection of pair-copulas, a method for R-vine model selection based on information theory metrics, and also an algorithm for learning R-vine models. This chapter closes with a section dedicated to the presentation of basic definitions on directed graphs, Bayesian networks (BNs), polytrees, and concepts on dependence maps used in the context of graphical models (GMs).

1.1 Copulas

Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be an n-dimensional random variable and $\boldsymbol{x} = (x_1, \ldots, x_n)$ be a sample of $\boldsymbol{X}$, with joint density function $f : \mathbb{R}^n \to [0, \infty)$ and cumulative distribution function $F : \mathbb{R}^n \to [0, 1]$. Furthermore, let $F_i : \mathbb{R} \to [0, 1]$, $i = 1, \ldots, n$, be the corresponding strictly increasing and continuous marginal distribution of X_i. Capital letters denote variables and lower case letters are their assignments.

Definition 1 (Copula) *An n-dimensional copula C is a distribution function $C : [0, 1]^n \to [0, 1]$ with uniformly distributed margins.*

The function C is an n-dimensional copula if and only if there exist random variables $U_1, \ldots, U_n$ such that the probability $\mathrm{P}(U_i \leq u_i) = u_i$ for $i = 1, \ldots n$ and $C(u_1, \ldots, u_n) = \mathrm{P}(U_1 \leq u_1, ..., U_n \leq u_n)$.

The relevance of copulas in probabilistic modeling is summarized in Sklar's theorem [120], which states that any multivariate distribution function F can be decomposed into its marginals F_i and a copula describing the dependence structure among them.

Theorem 1 (Sklar's Theorem) *For every joint cumulative distribution function F of random variables $X_1, \ldots, X_n$ with marginal cumulative distribution functions (cdfs) $F_1, \ldots, F_n$, there exists a copula C such that*

$$F(x_1, \ldots, x_n) = C(F_1(x_1), \ldots, F_n(x_n)). \tag{1.1}$$

If $F_1, \ldots, F_n$ are continuous, C is a unique n-dimensional copula. Conversely, given C and $F_1, \ldots, F_n$, F in Equation (1.1) is a multivariate cdf with marginals $F_1, \ldots, F_n$.

Moreover, if F is an n-dimensional distribution function with continuous marginals $F_1, \ldots, F_n$ and C satisfies Equation (1.1), then for any $\boldsymbol{u} = (u_1, \ldots, u_n) \in [0,1]^n$ we have

$$C(u_1, \ldots, u_n) = F\left(F_1^{-1}(u_1), \ldots, F_n^{-1}(u_n)\right), \tag{1.2}$$

where F_i^{-1}, $i = 1, \ldots, n$, denotes the inverse distribution function F_i defined as $F_i^{-1}(u_i) = \inf\{x_i; F_i(x_i) \geq u_i\}$, for all $u_i \in [0,1]$.

For continuous F and $F_1, \ldots, F_n$, the copula density function c of C can be obtained by partially differentiating Equation (1.1), such that

$$f(x_1, \ldots, x_n) = c(F_1(x_1), \ldots, F_n(x_n)) \cdot \prod_{i=1}^{n} f_i(x_i), \tag{1.3}$$

where $f_1, \ldots, f_n$ are the marginal densities and f the multivariate density of F.

The simplest copula is the Product copula. It appears naturally as the copula in Equation (1.1) associated to a random vector of independent variables. It can be shown that if $F(x_1, \ldots, x_n) = F_1(x_1) \cdot \ldots \cdot F_n(x_n) = C(F_1(x_1), \ldots, F_n(x_n))$, then C is defined as $C(u_1, \ldots, u_n) = u_1 \cdot \ldots \cdot u_n$ and $c(u_1, \ldots, u_n) = 1$.

A further example is the multivariate Normal copula, which is the copula of the multivariate normal distribution. In fact, the random vector $\boldsymbol{X} = (X_1, \ldots, X_n)$ is multivariate normal if and only if the univariate marginals $F_1, \ldots, F_n$ are Gaussian and the dependence structure among them is described by the Normal copula such that

$$C(u_1, \ldots, u_n) = \Phi_R\left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_n)\right),$$

where Φ_R is the standard multivariate Normal distribution function with linear correlation matrix R and Φ^{-1} is the inverse of the standard univariate Gaussian distribution function. In the bivariate case, for example, departing from Sklar's theorem, we get

$$f(x_1, x_2) = c_{12}(F_1(x_1), F_2(x_2)) \cdot f_1(x_1) \cdot f_2(x_2), \tag{1.4}$$

where

$$f_i(x_i) = \frac{1}{\sqrt{2\pi}} \cdot e^{\left\{-\frac{1}{2}x_i^2\right\}},\ i = 1, 2, \tag{1.5}$$

and

$$f(x_1, x_2) = \frac{1}{2\pi\sqrt{(1-\rho_{12}^2)}} \cdot e^{-\frac{x_1^2 + x_2^2 - 2\rho_{12}x_1x_2}{2(1-\rho_{12}^2)}}, \tag{1.6}$$

are the univariate and bivariate densities, and ρ_{12} is the linear correlation between X_1 and X_2.

Using Equation (1.4), we have that

$$\frac{f(x_1, x_2)}{f_1(x_1) f_2(x_2)} = c_{12}(F_1(x_1), F_2(x_2)), \tag{1.7}$$

and therefore

$$c_{12}(u_1, u_2) = \frac{1}{\sqrt{(1-\rho_{12}^2)}} \cdot e^{-\frac{\rho_{12}^2\left(x_1^2 + x_2^2\right) - 2\rho_{12}x_1x_2}{2(1-\rho_{12}^2)}}, \tag{1.8}$$

where $u_1 = \phi_1(x_1)$, $u_2 = \phi_2(x_2)$, and $x_1 = \phi_1^{-1}(u_1), x_2 = \phi_2^{-1}(u_2)$.

1.2 Pair-Copula Constructions

PCCs are multivariate models that factorize multivariate copula densities into (conditional) bivariate copula densities, so-called pair-copulas [8,74,84]. PCCs provide a flexible way of modeling the dependence structure of multivariate distributions, since pair-copulas of different families can be combined in the same decomposition, making it possible to capture different features such as non-linear, asymmetric dependence and tail dependence [35].

Applying the chain rule [27], any probability density function can be expressed as

$$f(x_1,\ldots,x_n) = f(x_1)\cdot f(x_2\,|x_1)\cdot f(x_3\,|x_1,x_2)\cdot\ldots\cdot f(x_n\,|x_1,\ldots,x_{n-1}). \tag{1.9}$$

Each conditional density in Equation (1.9) can be decomposed into a pair-copula and a conditional marginal density as

$$f(x_i\,|\boldsymbol{x_S}) = c_{i,k|\boldsymbol{S}_{-k}}\left(F\left(x_i\,\middle|\boldsymbol{x_{S_{-k}}}\right),F\left(x_k\,\middle|\boldsymbol{x_{S_{-k}}}\right)\right)\cdot f\left(x_i\,\middle|\boldsymbol{x_{S_{-k}}}\right), \tag{1.10}$$

for $i,k,\boldsymbol{S}\subseteq\boldsymbol{I}=\{1,\ldots,n\}\in\mathbb{N}$ disjoint subsets, and $|i|=|k|=1$. $\boldsymbol{x_S}$ denotes an arbitrary set of $\{x_1,\ldots,x_n\}\setminus x_i$ with x_k in it, x_k is one arbitrarily chosen component of $\boldsymbol{x_S}$, and $\boldsymbol{x_{S_{-k}}}=\boldsymbol{x_S}\setminus x_k$ denotes all the components from $\boldsymbol{x_S}$ excluding x_k. For instance, the second factor in (1.9) is the simplest conditional term and can be written using (1.10) as

$$f(x_2\,|x_1) = c_{1,2}\left(F(x_1),F(x_2)\right)\cdot f(x_2), \tag{1.11}$$

where $c_{1,2}$ denotes the pair-copula density for the pair of transformed variables $F(x_1)$ and $F(x_2)$. In the three-variate case, the third factor of (1.9) can be decomposed for the pair-copula density $c_{13|2}$ and treated as a bivariate density again as

$$f(x_3\,|x_1,x_2) = c_{1,3|2}\left(F(x_1\,|x_2),F(x_3\,|x_2)\right)\cdot f(x_3\,|x_2). \tag{1.12}$$

Notice that we could also have decomposed the third term for $c_{2,3|1}$ as

$$f(x_3\,|x_1,x_2) = c_{2,3|1}\left(F(x_2\,|x_1),F(x_3\,|x_1)\right)\cdot f(x_3\,|x_1), \tag{1.13}$$

where $c_{2,3|1}$ is different from $c_{1,3|2}$ in (1.12). Thus, given a specific factorization, there are different decompositions. Decomposing $f(x_3\,|x_2)$ in (1.12) further leads to

$$f(x_3\,|x_1,x_2) = c_{1,3|2}\left(F(x_1\,|x_2),F(x_3\,|x_2)\right)\cdot c_{2,3}\left(F(x_2),F(x_3)\right)\cdot f(x_3), \tag{1.14}$$

where two pair-copulas are present: $c_{2,3}$ describes the unconditional dependence of X_2 and X_3, while $c_{1,3|2}$ describes the conditional dependence of X_1 and X_3 given X_2.

Using (1.10), we can obtain similar expressions for the remaining terms of Equation (1.9) and derive a pair-copula decomposition of f that consists of marginal densities and pair-copulas.

More parsimonious models can be built by assuming conditional independencies: For any vector of variables $\boldsymbol{X_S}$ with $X_i,X_k\notin\boldsymbol{X_S}$, X_i and X_k are conditionally independent given $\boldsymbol{X_S}$ if and only if [1]

$$c_{i,k|\boldsymbol{S}}\left(F(X_i\,|\boldsymbol{X_S}),F(X_k\,|\boldsymbol{X_S})\right)=1. \tag{1.15}$$

For instance, if we consider the following three-variate pair-copula decomposition of f:

$$\begin{aligned} f(x_1,x_2,x_3) \;=\; & c_{1,2}\left(F(x_1),F(x_2)\right)\cdot c_{2,3}\left(F(x_2),F(x_3)\right) \\ & c_{1,3|2}\left(F(x_1\,|x_2),F(x_3\,|x_2)\right) \\ & \prod_{i=1}^{3} f_i(x_i) \end{aligned} \tag{1.16}$$

Assuming that X_1 and X_3 are independent given X_2, Equation(1.16) is simplified as

$$\begin{aligned} f(x_1, x_2, x_3) &= c_{1,2}(F_1(x_1), F_2(x_2)) \cdot c_{2,3}(F_2(x_2), F_3(x_3)) \\ &\quad \prod_{i=1}^{3} f_i(x_i) \end{aligned} \tag{1.17}$$

where $c_{1,3|2} = 1$.

1.2.1 Conditional Distribution Functions

The conditional distribution functions $F(x_i|\boldsymbol{x_S})$ constituting the pair-copula arguments in (1.10) can be obtained by applying recursively the formula derived in [74], written as

$$F(x_i|\boldsymbol{x_S}) = \frac{\partial C_{i,k|\boldsymbol{S}_{-k}}\left(F\left(x_i|\boldsymbol{x_{S_{-k}}}\right), F\left(x_k|\boldsymbol{x_{S_{-k}}}\right)\right)}{\partial F\left(x_k|\boldsymbol{x_{S_{-k}}}\right)}, \tag{1.18}$$

where $C_{i,k|\boldsymbol{S}_{-k}}$ is the bivariate copula to $F_{i,k|\boldsymbol{S}_{-k}}$, $\boldsymbol{x_S}$ is a random vector without the component x_i and with x_k in it, x_k is an arbitrary component of $\boldsymbol{x_S}$, and $\boldsymbol{x_{S_{-k}}}$ denotes the vector $\boldsymbol{x_S}$ excluding x_k.

For the case in which $\boldsymbol{x_S}$ is univariate, Equation (1.18) is simplified as

$$F(x_i|x_k) = \frac{\partial C_{i,k}(F(x_i), F(x_k))}{\partial F(x_k)}. \tag{1.19}$$

Moreover, when X_i and X_k are uniformly distributed, i.e., $f(x_i) = f(x_k) = 1$, $F(x_i) = x_i$ and $F(x_k) = x_k$, Equation (1.19) reduces further as

$$F(x_i|x_k) = \frac{\partial C_{i,k}(x_i, x_k)}{\partial x_k}. \tag{1.20}$$

1.3 Regular Vines

There exist many PCC decompositions in high dimensions. To organize such decompositions, a graphical model called regular vine (R-vine) is introduced in [8,9]. It involves the specification of a graph comprised of a hierarchy of $n-1$ trees $T_1, ..., T_{n-1}$, and $n(n-1)/2$ pair-copulas associated to each edge of the graph.

The following concepts follow the definitions provided in [8,84].

Definition 2 (R-vine graph) *An R-vine graph on n indexes $\{1, \ldots, n\}$ is defined by a hierarchy of trees $G = (T_1, \ldots, T_{n-1})$, where $T_j = \left(N^j, E^j\right)$ is the tree at level $j = 1, \ldots, n-1$ in the tree hierarchy, and N^j and E^j denote the node set and edge set of T_j respectively, such that:*

1. *T_1 is a tree with n nodes $N^1 = \{1, \ldots, n\}$ and a set of edges denoted by E^1.*
2. *For $j = 2, \ldots, n-1$, T_j is a tree with nodes $N^j = E^{j-1}$ and edge set E^j.*
3. *Proximity condition: For $j = 2, \ldots, n-1$, two nodes in T_j can only be adjacent (joined by an edge) if the corresponding edges in T_{j-1} share a common node.*

In an R-vine graph, the edges of T_{j-1} become nodes in T_j for $j = 2, \ldots, n-1$. Any edge of T_j joins a pair of nodes $Na, Nb \in N^j$. Each edge is identified as $e = \{i_e, k_e, \boldsymbol{S}_e\}$, where $i_e \in Na$ and $k_e \in Nb$ are single indexes of $\{1, \ldots, n\}$, $i_e \neq k_e$, $\boldsymbol{S}_e = Na \cap Nb$ is a subset of $\{1, \ldots, n\}$, and $i_e, k_e \notin \boldsymbol{S}_e$.

Definition 3 (R-vine copula) *Let G be an R-vine graph on n indexes. The n-variate copula density corresponding to G is given by*

$$\prod_{T_j \in G} \prod_{e \in E^j} c_{i_e,k_e|\boldsymbol{S}_e} \left(F_{i_e|\boldsymbol{S}_e} \left(x_{i_e} \left| \boldsymbol{x}_{\boldsymbol{S}_e} \right. \right), F_{k_e|\boldsymbol{S}_e} \left(x_{k_e} \left| \boldsymbol{x}_{\boldsymbol{S}_e} \right. \right) \right), \tag{1.21}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$, $e = \{i_e, k_e, \boldsymbol{S}_e\}$, $i_e \neq k_e$, $\boldsymbol{S}_e \subset \{1, \ldots, n\} \setminus \{i_e \cup k_e\}$, and $i_e, k_e, \boldsymbol{S}_e$ determine the variables $X_{i_e}, X_{k_e}, \boldsymbol{X}_{\boldsymbol{S}_e}$ in $\boldsymbol{X}$ respectively, and each $c_{i_e,k_e|\boldsymbol{S}_e}$ associated to the edge e comes from a set of bivariate copula densities defined as:

$$B = \left\{ c_{i_e,k_e|\boldsymbol{S}_e} \left(F_{i_e|\boldsymbol{S}_e} \left(x_{i_e} \left| \boldsymbol{x}_{\boldsymbol{S}_e} \right. \right), F_{k_e|\boldsymbol{S}_e} \left(x_{k_e} \left| \boldsymbol{x}_{\boldsymbol{S}_e} \right. \right) \right) \middle| e \in E^j, j = 1, \ldots, n-1 \right\}, \tag{1.22}$$

where the two arguments of $c_{i_e,k_e|\boldsymbol{S}_e}$ denote, respectively, the conditional distribution functions of x_{i_e} given $\boldsymbol{x}_{\boldsymbol{S}_e}$ and x_{k_e} given $\boldsymbol{x}_{\boldsymbol{S}_e}$.

Definition 4 (R-vine density) *An n-variate density function $f(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, \ldots, x_n)$, with a dependence structure represented by an R-vine copula (Definition 3) with associated R-vine graph G (Definition 2), and marginal densities $f_i(x_i)$, where $i = 1, \ldots, n$, is given by*

$$\underbrace{\underbrace{\prod_{T_j \in G} \prod_{e \in E^j} c_{i_e,k_e|\boldsymbol{S}_e} \left(F_{i_e|\boldsymbol{S}_e} \left(x_{i_e} \left| \boldsymbol{x}_{\boldsymbol{S}_e} \right. \right), F_{k_e|\boldsymbol{S}_e} \left(x_{k_e} \left| \boldsymbol{x}_{\boldsymbol{S}_e} \right. \right) \right)}_{R\text{-vine copula}} \underbrace{\prod_{i=1}^{n} f_i(x_i)}_{Marginal\ densities}}_{R\text{-vine density}}. \tag{1.23}$$

From now on, the symbol e will be omitted from the previous expressions for better readability, so we simply write $\{i, k, \boldsymbol{S}\}$ and $c_{i,k|\boldsymbol{S}}$ for denoting an edge and its associated pair-copula respectively.

As shown in [84], in R-vine models, each pair of variables X_i and X_k can occur only once as a conditioned set of only one pair-copula $c_{i,k|\boldsymbol{S}}$, which is associated with the edge $\{i, k, \boldsymbol{S}\}$ of the corresponding R-vine graph.

Throughout the dissertation, we adopt the simplifying assumption explored in [61], which states that the pair-copula $c_{i,k|\boldsymbol{S}} \left(F_{i|\boldsymbol{S}} \left(x_i \left| \boldsymbol{x}_{\boldsymbol{S}} \right. \right), F_{k|\boldsymbol{S}} \left(x_k \left| \boldsymbol{x}_{\boldsymbol{S}} \right. \right) \right)$ is independent of the conditioning variables $\boldsymbol{x}_{\boldsymbol{S}}$ except through their conditional distributions $F_{i|\boldsymbol{S}} \left(x_i \left| \boldsymbol{x}_{\boldsymbol{S}} \right. \right)$ and $F_{k|\boldsymbol{S}} \left(x_k \left| \boldsymbol{x}_{\boldsymbol{S}} \right. \right)$, which still depend on $\boldsymbol{x}_{\boldsymbol{S}}$. However, even when the simplifying assumption is not fulfilled and the PCC is not exact, it can be a good approximation of the distribution. This matter has been reviewed in further detail in [79, 92, 124].

Figure 1.1 shows a seven-dimensional R-vine graph whose associated R-vine copula is given by

$$\begin{aligned} &\underbrace{c_{1,2} \cdot c_{2,3} \cdot c_{2,5} \cdot c_{3,4} \cdot c_{3,6} \cdot c_{6,7}}_{T_1} \cdot \underbrace{c_{1,3|2} \cdot c_{2,4|3} \cdot c_{2,6|3} \cdot c_{3,5|2} \cdot c_{3,7|6}}_{T_2} \\ &\cdot \underbrace{c_{1,4|2,3} \cdot c_{1,5|2,3} \cdot c_{1,6|2,3} \cdot c_{2,7|3,6}}_{T_3} \cdot \underbrace{c_{4,5|1,2,3} \cdot c_{5,6|1,2,3} \cdot c_{1,7|2,3,6}}_{T_4} \\ &\cdot \underbrace{c_{4,6|1,2,3,5} \cdot c_{5,7|1,2,3,6}}_{T_5} \cdot \underbrace{c_{4,7|1,2,3,5,6}}_{T_6}, \end{aligned} \tag{1.24}$$

where the arguments for the bivariate copulas are omitted.

Two representative examples of R-vines are the Canonical and the Drawable vines (C-vine and D-vine respectively) [1, 84]:

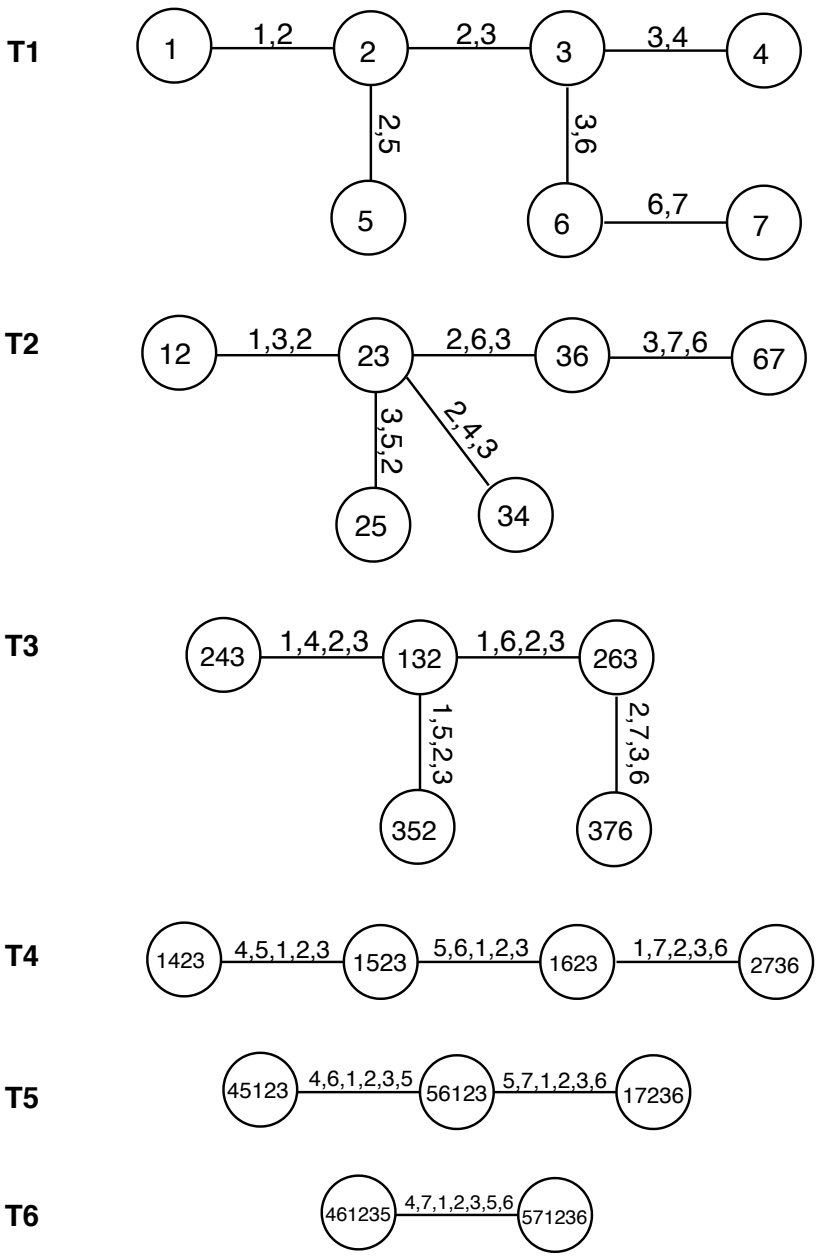


Figure 1.1: A seven-dimensional R-vine graph over the index set $\{1, \ldots, 7\}$. Edges are labeled with the indexes $i, k, \boldsymbol{S}$, comprising the edge $\{i, k, \boldsymbol{S}\}$. As edges of T_j become nodes in the next tree, nodes are labeled with $ik\boldsymbol{S}$ (displayed without commas) in T_{j+1} ; except T_1, where nodes are labeled with a single index. Each edge has a pair-copula associated with it.

- In a C-vine, the tree T_j, for $j = 1, ..., n-1$, has a node connected to the rest of the nodes, so the degree of this node is $n - j$ (see Figure 1.2).
- In a D-vine, in all the trees all nodes have a degree of at most two (see Figure 1.2).

The C-vine copula of a multivariate density $f(\boldsymbol{x})$ is written as

$$\prod_{j=1}^{n-1} \prod_{i=1}^{n-j} c_{j,j+i|i,\ldots,j-1} \left(F\left(x_j \left| x_1, \ldots, x_{j-1} \right.\right), F\left(x_{j+i} \left| x_1, \ldots, x_{j-1} \right.\right)\right), \tag{1.25}$$

and the D-vine copula of a multivariate density $f(\boldsymbol{x})$ is written as

$$\prod_{j=1}^{n-1} \prod_{i=1}^{n-j} c_{i,i+j|i+1,\ldots,i+j-1} \left(F\left(x_i \left| x_{i+1}, \ldots, x_{i+j-1} \right.\right), F\left(x_{i+j} \left| x_{i+1}, \ldots, x_{i+j-1} \right.\right)\right), \tag{1.26}$$

where j identifies the trees, while i runs over the edges in each tree.

The C-vine in Figure 1.2 has density

$$\underbrace{c_{1,2} \cdot c_{1,3} \cdot c_{1,4} \cdot c_{1,5}}_{T_1} \cdot \underbrace{c_{2,3|1} \cdot c_{2,4|1} \cdot c_{2,5|1}}_{T_2} \cdot \underbrace{c_{3,4|1,2} \cdot c_{3,5|1,2}}_{T_3} \cdot \underbrace{c_{4,5|1,2,3}}_{T_4}, \tag{1.27}$$

while the D-vine in Figure 1.2 has density

$$\underbrace{c_{1,2} \cdot c_{2,3} \cdot c_{3,4} \cdot c_{4,5}}_{T_1} \cdot \underbrace{c_{1,3|2} \cdot c_{2,4|3} \cdot c_{3,5|4}}_{T_2} \cdot \underbrace{c_{1,4|2,3} \cdot c_{2,5|3,4}}_{T_3} \cdot \underbrace{c_{1,5|2,3,4}}_{T_4}, \tag{1.28}$$

where the inputs for the bivariate copulas are omitted.

Among these PCC decompositions, R-vines have the most flexible structure as they are not affected by the particular structural constraints of C-vines and D-vines.

In Chapter 4, the D-vine and R-vine copulas are used to model the dependence structure of the features (variables) in two supervised classification tasks.

1.4 Bivariate Dependence Measures

We review two dependence measures that can be used to select and specify a regular vine-copula: Pearson's correlation and Kendall's tau coefficients, which here are given according to their definition in [43]. More details can be found in [84].

1.4.1 Pearson's Correlation

Pearson's linear correlation coefficient is a measure of the strength of a linear association between two random variables. It takes values in $[-1, 1]$.

Definition 5 (Pearson's correlation coefficient) *The Pearson's correlation coefficient of two random variables X,Y is defined as*

$$\rho(X, Y) = \frac{\operatorname{cov}(X, Y)}{\sqrt{\sigma^2(X)}\sqrt{\sigma^2(Y)}}, \tag{1.29}$$

where $\operatorname{cov}(X, Y)$ *denotes the covariance of X,Y; and* $\sigma^2(X)$ *and* $\sigma^2(Y)$ *denote the variance of X and Y respectively.*

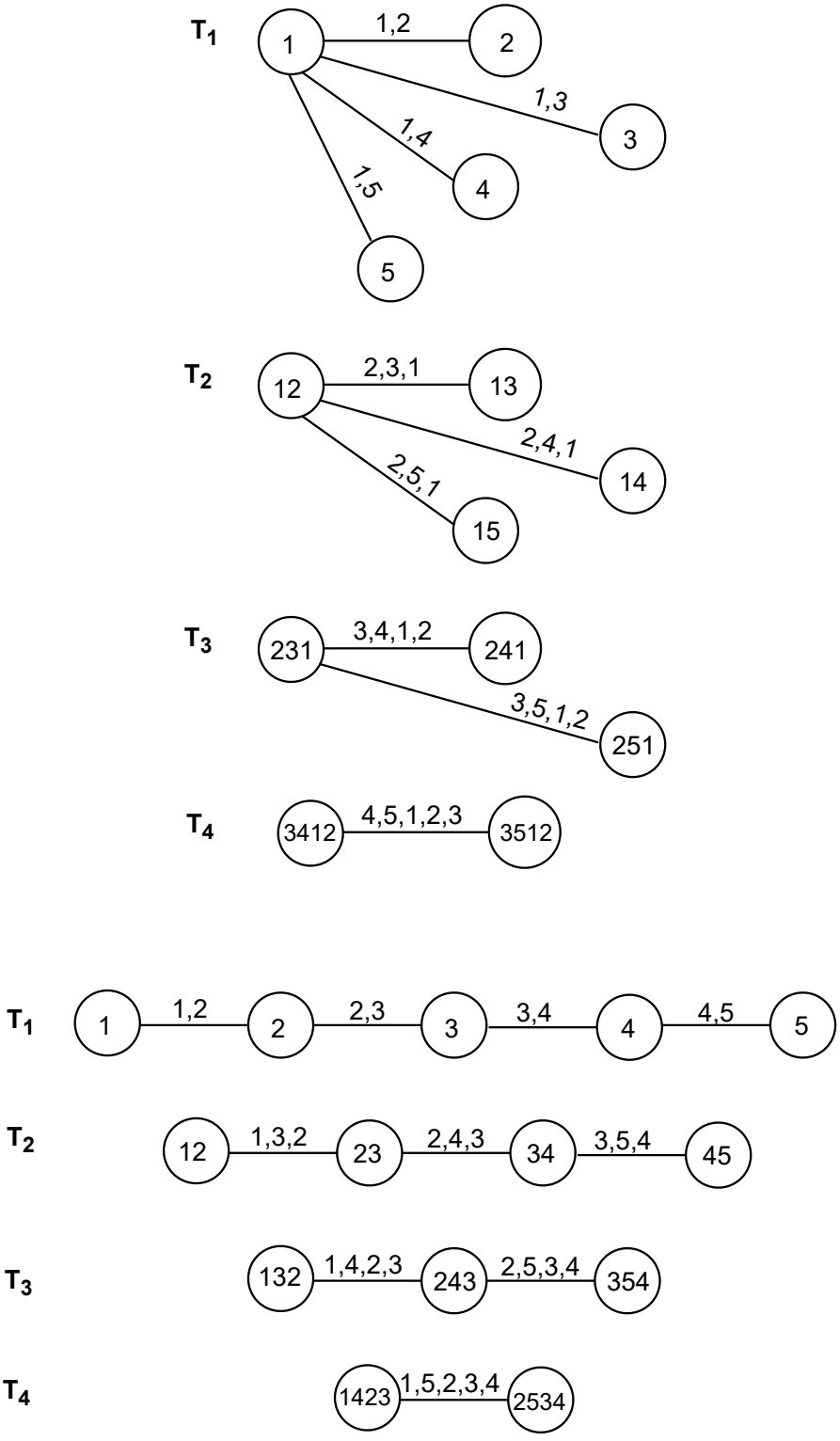


Figure 1.2: C-vine (top) and D-vine (bottom) graphs for $n = 5$. A C-vine describes the notion of the influence of one node over others, while in a D-vine each tree has a path structure.

X and Y are uncorrelated if $\rho(X,Y) = 0$. A value of $\rho(X,Y) > 0$ indicates a positive relationship, i.e., as the values of X increase/decrease, the values of Y do so as well, and vice versa; or simply put, the variables move in the same direction. A value of $\rho(X,Y) < 0$ indicates a negative relationship, i.e., as the value of one variable increases, the value of the other variable decreases. The stronger the linear association of X and Y, the closer the Pearson's correlation coefficient is to either 1 or -1, depending on whether the relationship is positive or negative respectively.

Given m pairs of observations (x_l, y_l), $l = 1, \ldots, m$ from the random vector (X,Y), the empirical Pearson's correlation $\hat{\rho}$ is calculated as

$$\hat{\rho}(X,Y) = \frac{\sum_{l=1}^{m}(x_l - \bar{x})(y_l - \bar{y})}{\sqrt{\sum_{l=1}^{m}(x_l - \bar{x})^2}\sqrt{\sum_{l=1}^{m}(y_l - \bar{y})^2}},$$

where $\bar{x} = \frac{1}{m}\sum_{l=1}^{m} x_l$ and $\bar{y} = \frac{1}{m}\sum_{l=1}^{m} y_l$ denote the empirical mean of X and Y, respectively.

There are some drawbacks from the Pearson's correlation that need particular attention when modeling dependence and marginal distributions separately. These include the fact that it depends on the marginal distributions of X and Y, and that it is not invariant under non-linear strictly increasing transformations of the variables [84].

1.4.2 Kendall's tau

The term *correlation coefficient* generally refers to measures of linear dependence between random variables (e.g., the Pearson's correlation coefficient (1.29)). The more general term *association coefficient* is used to refer to a group of dependence measures that are not restricted to be linear. The Kendall's tau correlation coefficient is among these measures.

Definition 6 (Kendall's tau coefficient) *Let (X_1, Y_1) and (X_2, Y_2) be two independent pairs of random variables. Kendall's tau is given by*

$$\tau(X,Y) = P[(X_1 - X_2)(Y_1 - Y_2) > 0] - P[(X_1 - X_2)(Y_1 - Y_2) < 0]. \tag{1.30}$$

Given m pairs of observations (x_l, y_l), $l = 1, \ldots, m$ from the random vector (X,Y), two pairs (x_i, y_i) and (x_j, y_j) of these observations are said to be concordant if $x_i < x_j$ and $y_i < y_j$ or $x_i > x_j$ and $y_i > y_j$. Similarly, they are said to be discordant if $x_i < x_j$ and $y_i > y_j$ or $x_i > x_j$ and $y_i < y_j$.

Informally, two random variables are concordant if the large values of one are usually associated with the large values of the other, or the small values of one are usually associated with the small values of the other. In Equation (1.30), $\mathrm{P}[(X_1 - X_2)(Y_1 - Y_2) > 0]$ and $\mathrm{P}[(X_1 - X_2)(Y_1 - Y_2) < 0]$ denote the probability of concordance and discordance respectively. Then, the empirical Kendall's tau $\hat{\tau}$ for the data sample is calculated as

$$\hat{\tau}(X,Y) = \frac{c - d}{c + d}, \tag{1.31}$$

where c and d denote the number of concordant and discordant pairs in the data sample, respectively. This coefficient also takes values in the interval $[-1, 1]$.

Unlike Pearson's correlation, Kendall's tau is one of the so-called rank correlation coefficients, so it does not depend directly on the values of the variables, instead it is a function of relationships between them. This coefficient is invariant by monotone increasing transformations of the variables [54].

1.5 Bivariate Copula Families

As previously discussed in Sections 1.2 and 1.3, pair-copulas are the building blocks for constructing multivariate vine-copula models. In a PCC, bivariate copulas from different families can be selected independently, providing great flexibility for dependence modeling. In particular, symmetry, tail and positive/negative dependencies as well as conditional independencies to build simple models (see Equation (1.15)) can be taken into account when modeling with PCCs.

In this section, we give an overview of five common bivariate copulas, namely Product (P), Gaussian or Normal (N), Student's t (S), Clayton (C) and Gumbel (G), following the definitions provided in [1, 43].

Bivariate Product Copula Two random variables X_1 and X_2 with continuous distribution functions F_1 and F_2 and joint distribution function F are independent if and only if $F(x_1, x_2) = F_1(x_1) \cdot F_2(x_2)$ for all $x_1, x_2 \in \mathbb{R}$. The structure of such a relationship is given by the independence or Product copula, whose distribution and density functions are defined as follows

$$C_{\mathrm{P}}(u_1, u_2) = u_1 \cdot u_2, \tag{1.32}$$

and

$$c_{\mathrm{P}}(u_1, u_2) = 1, \tag{1.33}$$

respectively. A scatter plot of bivariate independent data is shown in Figure 1.3.

Bivariate Normal Copula The distribution and density functions of the bivariate Normal copula with Pearson's correlation parameter $\rho_{1,2} \in (-1, 1)$ are given by

$$C_{\mathrm{N}}(u_1, u_2) = \Phi_{\rho_{1,2}}\left(\Phi^{-1}(u_1), \Phi^{-1}(u_2)\right), \tag{1.34}$$

and

$$c_{\mathrm{N}}(u_1, u_2) = \frac{1}{\sqrt{1-\rho_{1,2}^2}} \cdot e^{-\frac{\rho_{1,2}^2\left(x_1^2+x_2^2\right)-2\cdot\rho_{1,2}\cdot x_1\cdot x_2}{2\cdot\left(1-\rho_{1,2}^2\right)}}, \tag{1.35}$$

respectively, where $\Phi_{\rho_{1,2}}$ is the bivariate standard Normal distribution function with correlation parameter $\rho_{1,2}$, $x_1 = \Phi^{-1}(u_1)$, $x_2 = \Phi^{-1}(u_2)$, and Φ^{-1} is the inverse of the standard univariate Gaussian distribution function. The closer $\rho_{1,2}$ is to either 1 or -1, the stronger the linear positive or negative relationship between U_1 and U_2, respectively, while $\rho_{1,2} = 0$ means there is no linear relationship between them.

Bivariate Student's t Copula The distribution and density functions of the Student's t copula with parameters $\rho_{1,2} \in (-1, 1)$ and $\nu_{1,2} > 0$ are given by

$$C_{\mathrm{S}}(u_1, u_2) = t_{\rho_{1,2},\nu_{1,2}}\left(t_{\nu_{1,2}}^{-1}(u_1), t_{\nu_{1,2}}^{-1}(u_2)\right), \tag{1.36}$$

and

$$c_{\mathrm{S}}(u_1, u_2) = \frac{\Gamma\left(\frac{\nu_{1,2}+2}{2}\right)/\Gamma\left(\frac{\nu_{1,2}}{2}\right)}{\nu_{1,2}\cdot\pi\cdot dt_{\nu_{1,2}}(x_1)\cdot dt_{\nu_{1,2}}(x_2)\cdot\sqrt{1-\rho_{1,2}^2}}\left(1+\frac{x_1^2+x_2^2-2\cdot\rho_{1,2}\cdot x_1\cdot x_2}{\nu_{1,2}\cdot\left(1-\rho_{1,2}^2\right)}\right)^{-\frac{\nu_{1,2}+2}{2}}, \tag{1.37}$$

respectively, where $x_1 = t^{-1}_{\nu_{1,2}}(u_1)$, $x_2 = t^{-1}_{\nu_{1,2}}(u_2)$, Γ is the Gamma function, and $dt_{\nu_{1,2}}$ and $t^{-1}_{\nu_{1,2}}$ are the density and the inverse functions, respectively, of the standard univariate Student's t distribution function with $\nu_{1,2}$ degrees of freedom, expectation 0 and variance $\frac{\nu_{1,2}}{\nu_{1,2}-2}$.

Bivariate Clayton Copula The distribution and density functions of the bivariate Clayton copula with parameter $\delta_{1,2} \in (0, \infty)$ are given by

$$C_{\mathrm{C}}(u_1, u_2) = \left(u_1^{-\delta_{1,2}} + u_2^{-\delta_{1,2}} - 1\right)^{-\frac{1}{\delta_{1,2}}}, \tag{1.38}$$

and

$$c_{\mathrm{C}}(u_1, u_2) = (1 + \delta_{1,2}) \cdot (u_1 \cdot u_2)^{-1-\delta_{1,2}} \left(u_1^{-\delta_{1,2}} + u_2^{-\delta_{1,2}} - 1\right)^{-\frac{1}{\delta_{1,2}}-2}, \tag{1.39}$$

respectively. The larger the value of $\delta_{1,2}$, the stronger the positive relationship between U_1 and U_2, while the closer $\delta_{1,2}$ is to zero, the more independent they are.

Bivariate Gumbel Copula The distribution and density functions of the bivariate Gumbel copula with parameter $\delta_{1,2} \in [1, \infty)$ are given by

$$C_{\mathrm{G}}(u_1, u_2) = e^{-\left((-\log u_1)^{\delta_{1,2}} + (-\log u_2)^{\delta_{1,2}}\right)^{\frac{1}{\delta_{1,2}}}}, \tag{1.40}$$

and

$$\begin{aligned} c_{\mathrm{G}}(u_1, u_2) = {} & \tfrac{1}{C_{\mathrm{G}}(u_1, u_2)} \left(-\left((-\log u_1)^{\delta_{1,2}} + (-\log u_2)^{\delta_{1,2}}\right)^{\frac{2}{\delta_{1,2}}-2}\right) \\ & \cdot (\log u_1 \cdot \log u_2)^{\delta_{1,2}-1} \\ & \cdot \left(1 + (\delta_{1,2} - 1) \cdot \left((-\log u_1)^{\delta_{1,2}} + (-\log u_2)^{\delta_{1,2}}\right)^{\frac{1}{\delta_{1,2}}}\right), \end{aligned} \tag{1.41}$$

respectively. The larger the value of $\delta_{1,2}$, the stronger the positive relationship between U_1 and U_2, while $\delta_{1,2} = 1$ implies independence.

The Kendall's tau correlation coefficient can be expressed as a function of the dependence parameter of the bivariate copulas described in this section. These expressions can be inverted to obtain the value of the copula parameter from the value of Kendall's tau correlation coefficient [97] (see Section 1.4). Table 1.1 shows the relationship between Kendall's tau and the dependence parameter for the bivariate copulas described in this section.

The bivariate Product copula describes pairwise independence while the other four copulas capture different types of pairwise dependence, regarding features such as symmetry, strength in the tails, and a positive/negative relationship of the bivariate distribution. The bivariate Normal and Student's t copulas are symmetric. In particular, the Normal has neither lower nor upper tail dependence, while the Student's t has both lower and upper tail dependence. On the other hand, the bivariate Clayton and Gumbel copulas are asymmetric. In particular, the Clayton copula has lower tail dependence, but not upper, while the Gumbel has upper tail dependence, but not lower. Unlike the Normal and Student's t copulas, which can represent both positive and negative relationships, the Clayton and Gumbel copulas only capture positive relationships. A comprehensive reference about bivariate copula families can be found in [75, 97].

To get a sense of how these copulas represent different types of bivariate dependencies, Figure 1.4 shows scatter plots of samples generated from the bivariate Normal, Clayton and Gumbel copulas,

Table 1.1: Expressions of the dependence parameter of a group of bivariate copulas as a function of Kendall's tau.

Copula	Dependence Parameter	Kendall's tau
Normal	$\rho_{1,2} = \sin\left(\frac{\pi}{2}\tau\right), \rho_{1,2} \in (-1,1)$	$\tau = \frac{2}{\pi}\arcsin\left(\rho_{1,2}\right)$
Student's t	$\rho_{1,2} = \sin\left(\frac{\pi}{2}\tau\right), \rho_{1,2} \in (-1,1), \nu > 2$	$\tau = \frac{2}{\pi}\arcsin\left(\rho_{1,2}\right)$
Clayton	$\delta_{1,2} = \frac{2\tau}{1-\tau}, \delta_{1,2} > 0$	$\tau = \frac{\delta_{1,2}}{\delta_{1,2}+2}$
Gumbel	$\delta_{1,2} = \frac{1}{1-\tau}, \delta_{1,2} \geq 1$	$\tau = 1 - \frac{1}{\delta_{1,2}}$

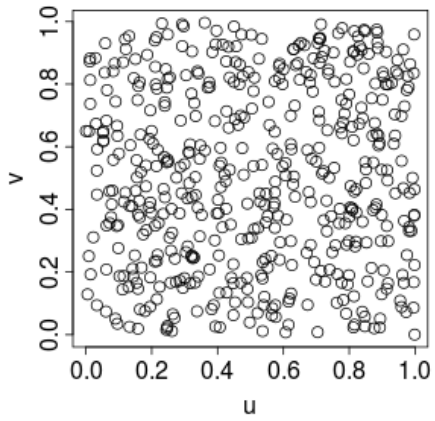


Figure 1.3: Scatter plots of 500 points sampled from the bivariate Product copula.

for different levels of dependence strength according to Kendall's tau coefficient, namely weak ($\hat{\tau} = \pm 0,25$), moderate ($\hat{\tau} = \pm 0,5$) and strong ($\hat{\tau} = \pm 8,0$).

In addition to these families, in the analysis of Chapter 4, we allow the rotated versions of the Clayton and Gumbel copulas by 90°, 180° and 270° for the modeling of negative dependence, which is not possible with their non-rotated versions (Equations (1.38) and (1.40) respectively). The dependence parameters of the rotated pair-copulas by 90° and 270° are on the negative scale (see Table 1.1). The distribution functions C_{90}, C_{180} and C_{270} of a copula C rotated by 90°, 180° and 270°, respectively, are given by

$$\begin{array}{lcl} C_{90}\left(u_1,u_2\right) & = & u_2 - C\left(1-u_1,u_2\right), \\ C_{180}\left(u_1,u_2\right) & = & u_1 + u_2 - 1 + C\left(1-u_1,1-u_2\right), \\ C_{270}\left(u_1,u_2\right) & = & u_1 - C\left(u_1,1-u_2\right). \end{array} \tag{1.42}$$

1.6 Selection of Pair-Copulas

Strategies of copula selection allow us to determine whether a copula is a suitable representation of the dependence patterns presented in a set of observations. Typically, these methods select, from a set of families, the one that best fits the data sample, using, for instance, a model selection criterion such as the Akaike Information Criterion (AIC) [2] or the Bayesian Information Criterion

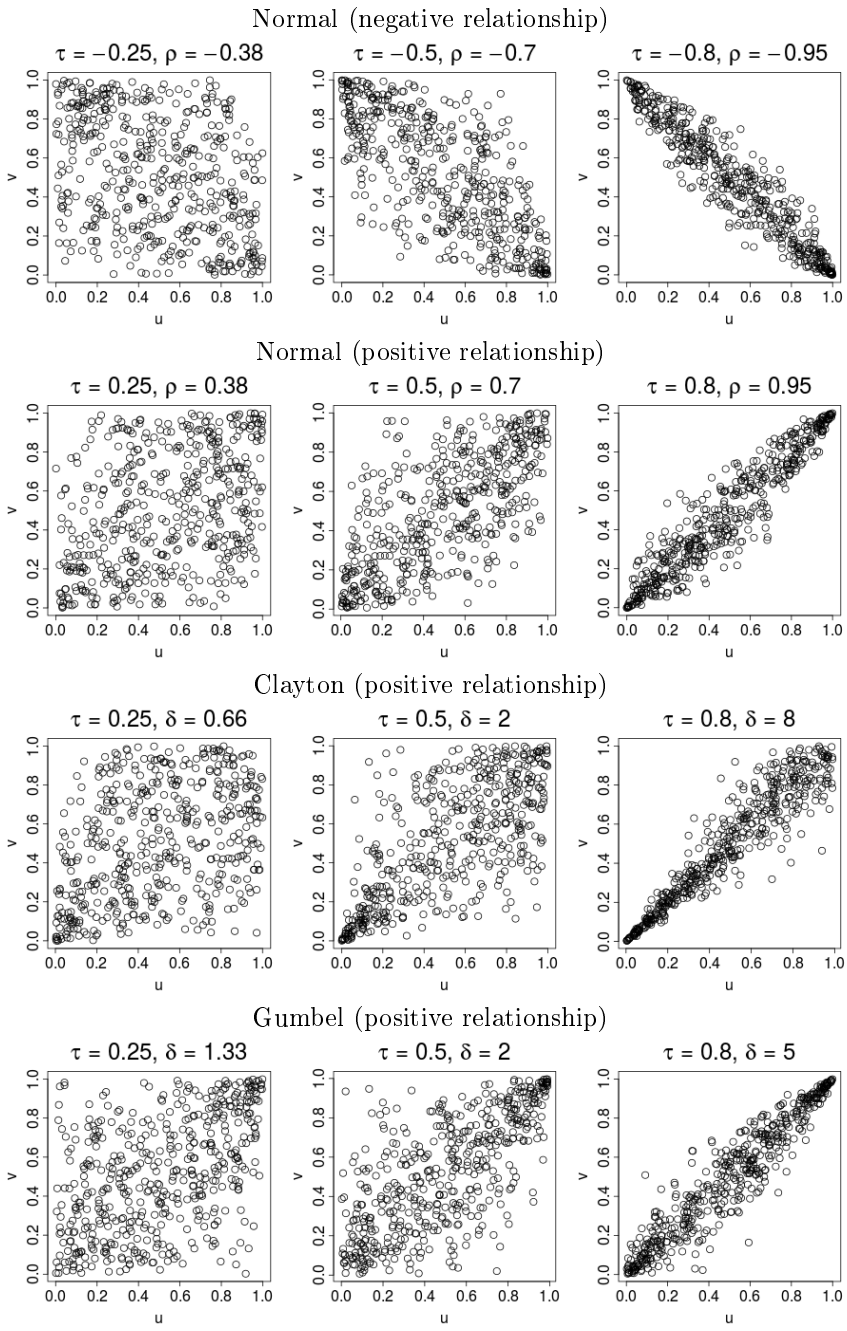


Figure 1.4: Scatter plots of 500 points sampled from the bivariate Normal, Clayton, and Gumbel copulas, respectively, for different values of the dependence parameter. The Normal copula has neither lower nor upper tail dependence, and can represent positive/negative correlations. The Clayton copula is asymmetric, positive, and lower tail dependent, but not upper. The Gumbel copula is asymmetric, positive, and upper tail dependent, but not lower. Parameter setting: weak ($\hat{\tau} = \pm 0,25$), moderate ($\hat{\tau} = \pm 0,5$), and strong ($\hat{\tau} = \pm 8,0$) positive/negative relationship.

(BIC) [117] presented in [17]. Further approaches include the goodness-of-fit test for bivariate copulas based on the Cramér-von Mises statistic[1] as proposed in [56]. The latter is the method used in this thesis for the selection of pair-copulas.

1.6.1 Goodness-of-fit Test for Bivariate Copulas

Goodness-of-fit tests for copulas allow us to assess how good a copula describes the dependence structure of a data sample. In particular, we use a goodness-of-fit test based on the Cramér-von Mises statistic. According to this test, the parametric copula family with the smallest distance to the empirical copula distribution [39] is chosen.

Definition 7 (Empirical bivariate copula) *The empirical copula distribution C_m of m observations $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m\}$, where $\boldsymbol{u}_l = (u_{l,1}, u_{l,2})$, $l = 1, \ldots, m$, of a random vector $(U_1, U_2) \in (0,1)^2$ is given by*

$$C_m(u_1, u_2) = \frac{1}{m} \sum_{l=1}^{m} \mathbf{1}_{u_{l,1} \leq u_1, u_{l,2} \leq u_2}, \tag{1.43}$$

Definition 8 (Cramér-von Mises statistic) *The Cramér-von Mises statistic S_m is given by*

$$S_m = \sum_{l=1}^{m} \left(C_m(u_{l,1}, u_{l,2}) - C_{\hat{\theta}}(u_{l,1}, u_{l,2}) \right)^2, \tag{1.44}$$

where C_m denotes the empirical copula (Equation (1.43)), and $C_{\hat{\theta}}$ is a parametric bivariate copula, where $\hat{\theta}$ is an estimation of θ using the data sample $(u_{l,1}, u_{l,2})$, $l = 1, \ldots, m$.

The Cramér-von Mises goodness-of-fit test is based on the difference between the empirical bivariate copula C_m and the estimated parametric copula $C_{\hat{\theta}}$ of the unknown pair-copula C, under the null hypothesis that C belongs to a parametric copula family C_θ with parameter θ, i.e.,

$$H_0 : C \in C_\theta \text{ versus } H_1 : C \notin C_\theta,$$

where $\theta \in \Theta$, Θ is a subset of $\mathbb{R}^r$ for an integer $r \geq 1$. Then, the parametric copula with the highest p-value is selected. Corresponding p-values can be calculated by the bootstrapping-based methods [56, 81]. However, we will use the copula with the smallest S_m assuming that, among the tested copulas, this should be the least likely to be rejected.

A test of independence can be considered a particular case of the goodness-of-fit test for copulas, where the null hypothesis is that the unknown bivariate copula C is the Product copula [40, 55]. Therefore, the Cramér-von Mises statistic can be used, replacing $C_{\hat{\theta}}(u_1, u_2)$ by $C_{\mathrm{P}}(u_1, u_2)$ in Equation (1.44). C_{P} is selected if the corresponding S_m is the smallest.

1.7 AIC and BIC for R-vine Model Selection

The flexibility of R-vines comes at the price of a rapid increase in the number of parameters with the dimension. In order to reduce the number of parameters in high-dimension applications, the truncation strategy proposed in [17] allows the creation of parsimonious models by using simple pair-copula terms in the last trees.

[1] The Cramér-von Mises criterion is a classical goodness-of-fit statistic that characterizes the distance between a cumulative distribution function F and a given empirical distribution function F_n in $\ell 2$-norm.

Definition 9 (Truncated R-vine Copula) *An R-vine copula (Definition 3) is said to be truncated at level t when all pair-copulas with the conditioning set equal to or larger than t are set to bivariate Product copulas. Then, the density of a truncated R-vine copula at level t is given by*

$$c_{tRV(t)} = \prod_{j=1}^{t} \prod_{e \in E^j} c_{i_e,k_e|\boldsymbol{S}_e} \left(F_{i_e|\boldsymbol{S}_e} \left(x_{i_e} | \boldsymbol{x}_{\boldsymbol{S}_e} \right), F_{k_e|\boldsymbol{S}_e} \left(x_{k_e} | \boldsymbol{x}_{\boldsymbol{S}_e} \right) \right). \tag{1.45}$$

If $t = 1$, the truncated R-vine copula becomes a Markov tree distribution, where independence is assumed between all pairs of variables [62].

To identify the most appropriate truncation level, a heuristic based on a statistical model selection approach via AIC or BIC is proposed in [17]. These criteria are commonly used in the comparison of nested models.

Definition 10 (AIC) *Given a set of m observations* $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, *where* $\boldsymbol{x}_l = (x_{l,1}, \ldots, x_{l,n})$, $l = 1, \ldots, m$, *the AIC for a parametric model is defined as*

$$AIC = -2 \sum_{l=1}^{m} \log L \left(\boldsymbol{x}_l; \hat{\boldsymbol{\theta}} \right) + 2k, \tag{1.46}$$

where L denotes the likelihood function for the model, and $\hat{\boldsymbol{\theta}}$ *is the vector of the k parameters of the model estimated by maximum likelihood.*

Definition 11 (BIC) *Given a set of m observations* $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, *where* $\boldsymbol{x}_l = (x_{l,1}, \ldots, x_{l,n})$, $l = 1, \ldots, m$, *the BIC for a parametric model is defined as*

$$BIC = -2 \sum_{l=1}^{m} \log L \left(\boldsymbol{x}_l; \hat{\boldsymbol{\theta}} \right) + k \log m, \tag{1.47}$$

where L denotes the likelihood function for the model, and $\hat{\boldsymbol{\theta}}$ *is the vector of the k parameters of the model estimated by maximum likelihood.*

In these kind of metrics, the first term is a measure of the goodness-of-fit, i.e., the higher the values of L, the better the model describes the sample. The second term penalizes the complexity of the model in terms of the number of parameters to be estimated, which favors the selection of models with fewer parameters [116]. A complex model will then have a good score only if the gain in terms of likelihood is high enough to justify the number of parameters used. In the BIC metric, the penalization term also includes the sample size.

From (1.46) and (1.47), the AIC and BIC of a truncated R-vine copula at level t (1.45) are given by

$$AIC\left(tRV\left(t\right)\right) = -2L_{tRV(t)} \left(\boldsymbol{x}; \hat{\boldsymbol{\theta}} \right) + 2k, \tag{1.48}$$

and

$$BIC\left(tRV\left(t\right)\right) = -2L_{tRV(t)} \left(\boldsymbol{x}; \hat{\boldsymbol{\theta}} \right) + k \log m, \tag{1.49}$$

respectively, where $L_{tRV(t)}$ denotes the log likelihood of the truncated R-vine copula density (1.45), which is given by

$$
\begin{aligned}
&L_{tRV(t)}\left(\boldsymbol{x};\hat{\boldsymbol{\theta}}\right) \\
&=\sum_{l=1}^{m}\sum_{j=1}^{t}\sum_{e\in E^j}\log c_{i_e,k_e|\boldsymbol{S}_e}\left(F_{i_e|\boldsymbol{S}_e}\left(x_{li_e}|\boldsymbol{x}_{l\boldsymbol{S}_e}\right),F_{k_e|\boldsymbol{S}_e}\left(x_{lk_e}|\boldsymbol{x}_{l\boldsymbol{S}_e}\right);\hat{\boldsymbol{\theta}}_{i_e,k_e|\boldsymbol{S}_e}\right),
\end{aligned}
\tag{1.50}
$$

where m is the number of observations $\boldsymbol{x}_l = (x_{l,1},\ldots,x_{l,n})$, $l = 1,\ldots,m$, $\hat{\boldsymbol{\theta}} = \left\{\hat{\boldsymbol{\theta}}_{i_e,k_e|\boldsymbol{S}_e} : e\in E^j, j=1,\ldots,t\right\}$, and $\hat{\boldsymbol{\theta}}_{i_e,k_e|\boldsymbol{S}_e}$ denotes the parameters of the copula density $c_{i_e,k_e|\boldsymbol{S}_e}$.

The procedure for identifying the truncation level starts by fitting the R-vine copula to the first level $t = 1$. With each iteration, t is increased by one, only stopping at a truncation level $t = t_0$ if the contribution from fitting an extra tree is not significant. To assess whether there is gain of an additionally fitted tree, we can compare the AIC (1.48) or BIC (1.49) of the two models $tRV(t)$ and $tRV(t+1)$, and the model with the lowest metric is selected.

Since $tRV(t)$ is nested within $tRV(t+1)$, the truncation heuristic only needs to calculate the contribution from the level $t+1$ to the AIC or BIC of $tRV(t+1)$, since the values of these metrics for $tRV(t)$ and $tRV(t+1)$ are equal except for the contribution from the level $t+1$. Due to the fact that all pair-copulas associated to the tree at level $t+1$ of $tRV(t)$ are Product copulas, the contribution from the level $t+1$ to the AIC or BIC of $tRV(t)$ is zero. Therefore, if the contribution from the level $t+1$ to the metric of $tRV(t+1)$ is positive, i.e., $BIC(tRV(t+1)) > 0$ (similar to AIC), the R-vine copula is truncated at level $t = t_0$. See [16] for details.

1.8 R-vine Learning

An attractive feature of the copula framework is that it provides a way to fit $f(\boldsymbol{x})$ by estimating the marginal densities $f_i(x_i)$ and the R-vine copula separately [76, 77]. This property is leveraged by the learning algorithms, which first fit the marginals and then learn the R-vine copula. The latter is accomplished by the following three closely related tasks: (i) Selection of the R-vine structure; (ii) Selection of pair-copulas; (iii) Estimation of the pair-copula parameters.

Algorithms to learn R-vines from data have mainly focused on greedy heuristics [37, 82]. Since pair-copulas can be estimated more accurately in the first trees (as the conditioning sets involve fewer variables), a natural way to proceed is by building the structure one tree at a time in a top-down approach, while trying to maximize the dependence in the first levels. Such a heuristic, proposed in [42], proceeds sequentially, starting to optimize individually the first tree, T_1, continuing with the second tree, T_2, and so on. This procedure requires that at each level the pair-copulas and their parameters are simultaneously estimated before moving on to the next level.

In this thesis, the procedure used to learn the R-vine density function (1.21) combines both the top-down R-vine learning procedure and the AIC/BIC-based truncation heuristic (described previously in Section 1.7). Here, we refer to this algorithm as Top-Down Sequential Heuristics (TDSH). A description of this algorithm is given in the next section.

1.8.1 Top-Down Sequential Greedy Heuristic

We introduce some notation before describing the TDSH in Algorithm 1.1. First, $i,k = 1,\ldots,n$ is run over indexes, $j = 1,\ldots,n-1$ over trees, and $l = 1,\ldots,m$ over observations (sample data). Further, let $\boldsymbol{D}_{\boldsymbol{X}} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m\}$ be the set of m observations (the so-called *original data*), where $\boldsymbol{x}_l = (x_{l,1},\ldots,x_{l,n})$ denotes an observation of $\boldsymbol{X} = (X_1,\ldots,X_n)$, and $\boldsymbol{D}_{\boldsymbol{U}}^j = \{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_m\}$ the set of m observations (the so-called *transformed data*) associated to the level j, where $\boldsymbol{u}_l = (u_{l,1},\ldots,u_{l,n-j+1})$ denotes an observation of $\boldsymbol{U} = (U_1,\ldots,U_{n-j+1})$.

This algorithm starts by estimating n marginal cumulative distributions $F_i(X_i)$ from the original data $\boldsymbol{D_X}$. Then, by evaluating $U_i = F_i(X_i)$, it computes the transformed observations needed to estimate the unconditional pair-copulas of the first tree T_1. To select this tree, the algorithm starts by finding the maximum spanning tree (MST) (using Prim's algorithm [32]) from the complete graph $\mathcal{G}$ over n nodes, which is the tree on all nodes that maximizes the sum of the pairwise dependence measures used as weights on the edges. Then, the pair-copulas associated to the edges of T_1 are selected and their parameters are estimated using the transformed data previously computed. New transformed observations, $F_{i|\boldsymbol{S}}$ and $F_{k|\boldsymbol{S}}$ (the arguments of the pair-copulas), are recursively computed from the pair-copulas estimated in the previous level using Equation (1.18). These observations are used as input data for the subsequent trees, which are obtained similarly by finding the MST from a connected and weighted graph $\mathcal{G}$ (usually, not complete) with those edges allowed by the proximity condition. In each level, given the pair-copulas of the previous tree, the conditional pair-copulas of the next tree are selected and estimated. This procedure iterates until a complete or truncated R-vine is learned. Steps of this algorithm are described in the following[2].

Step 1 The univariate cumulative and density functions $F_i(X_i)$ and $f_i(X_i)$ are estimated from a set of original observations $\boldsymbol{D_X}$. Two strategies can be used to select the type of margins: The simplest, but rather inaccurate, strategy selects the same family of distributions to model all margins. A more flexible method chooses the one that best fits the data among a predefined group of parametric families (e.g., Beta, Gamma) as well as an empirical distribution based on a kernel function, which allows us to prevent making assumptions about the shape of the univariate distribution. The density estimator K of an unknown density f at any given point x is given by

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right), \tag{1.51}$$

where $\{x_1, \ldots, x_m\}$ are samples of the unknown f, and $h > 0$ is a smoothing parameter called the bandwidth [14]. The Normal (or Gaussian) kernel is one of the most widely used kernels to approximate univariate data, which means that $K(x) = \Phi(x)$, where Φ is the standard Normal density function, and is expressed as

$$K(x) = e^{-\frac{(x-x_i)^2}{2h^2}}, \tag{1.52}$$

where the choice for h according to a rule-of-thumb bandwidth estimator is given by $h = \left(4\hat{\sigma}^5/3m\right)^{\frac{1}{5}} \approx 1.06\hat{\sigma}m^{-\frac{1}{5}}$, where $\hat{\sigma}$ denotes the empirical standard deviation.

Step 2 From the marginal distributions estimated at Step 1, the algorithm computes the (unconditional) observations $\boldsymbol{u}_l = (u_{l,1}, \ldots, u_{l,n})$, where $u_{l,i} = F_i(x_{l,i})$, $l = 1, \ldots, m$, and $i = 1, \ldots, n$.

Steps 3 and 8 We use the empirical Kendall's tau coefficient (see (1.30)) as a measure of pairwise dependence. This measure does not rely on any assumptions on the distributions of the variables, which is especially useful when different bivariate copulas are combined in the same decomposition. For $j = 1$, a complete graph $\mathcal{G}$ over n nodes is built. Then, for each edge $\{i, k\}$ of $\mathcal{G}$, empirical Kendall's tau $\hat{\tau}$ are computed from $\boldsymbol{D}_{\boldsymbol{U}}^1$ and assigned as weights of the edges. For the subsequent levels ($j > 1$), the graph $\mathcal{G}$ is built over $n - j + 1$ nodes for all possible conditional pairs allowed by the proximity condition. Then, in a similar way to the previous tree, for each edge $\{i, k, \boldsymbol{S}\}$ of $\mathcal{G}$, empirical Kendall's taus $\hat{\tau}$ are computed from $\boldsymbol{D}_{\boldsymbol{U}}^j$, $j = 2, \ldots, n-1$ and assigned as weights of the edges.

[2]The implementation of the TDSH is available at https://github.com/DianaCarrera

Algorithm 1.1 Steps of TDSH for learning R-vine distributions.

Input
$\boldsymbol{D_X}$ – Original observations.
$\mathcal{C}$ – Set of candidate copula families.
t – Truncation level. Possible values $1 \leq t \leq n-1$.
$t_0 = j = 1$ – Auxiliary variables.

Output
$f(X_i)$– Marginal density, $i = 1, \ldots, n$.
G – R-vine graph.
B – Set of pair-copulas associated to G.

1: Estimate F_i from $\boldsymbol{D_X}$.
2: Obtain $m \times n$ observations $\boldsymbol{D_U^j}$ for T_j by computing $F_i(X_i)$.
Compute $f_i(X_i)$.
3: Build the complete graph $\mathcal{G}$ over n nodes and compute dependence measures (e.g., $\hat{\tau}$) as weights of the edges $\{i,k\} \in \mathcal{G}$ from $\boldsymbol{D_U^j}$.
4: Select the MST T_j from the weighted graph $\mathcal{G}$.
5: For each edge $\{i,k\}$ in T_j, select $c_{i,k} \in \mathcal{C}$ and estimate its parameter $\hat{\theta}_{i,k}$ from $\boldsymbol{D_U^j}$.
if $t > j$:
6: Compute $L_{tRV(t_0)}$ (1.50) and $BIC(tRV(t_0))$ (1.49).
for $j = 2 : t$:
$t_0 = j - 1$
7: Obtain $m \times n - j + 1$ transformed observations $\boldsymbol{D_U^j}$ for T_j by computing conditional distribution functions $F_{i|k \cup \boldsymbol{S}}$ (1.18).
8: Build a graph $\mathcal{G}$ over $n - j + 1$ nodes with all possible edges that meet the proximity condition from T_{j-1} and compute dependence measures (e.g., $\hat{\tau}$) as weights of the edges $\{i,k,\boldsymbol{S}\} \in \mathcal{G}$ from $\boldsymbol{D_U^j}$.
9: Select the MST T_j from the weighted graph $\mathcal{G}$.
10: For each edge $\{i,k,\boldsymbol{S}\}$ in T_j, select $c_{i,k|\boldsymbol{S}} \in \mathcal{C}$ and estimate its parameter $\hat{\theta}_{i,k|\boldsymbol{S}}$ from $\boldsymbol{D_U^j}$.
11: Compute $L_{tRV(t_0+1)}$ and $BIC(tRV(t_0+1))$.
if $BIC(tRV(t_0+1)) - BIC(tRV(t_0)) > 0$:
12: Set all pair-copulas with $|\boldsymbol{S}| > t_0$ to Product copulas.
exit for
else $j == t$
13: Set all pair-copulas with $|\boldsymbol{S}| > t$ to Product copulas.
exit for
end if
end for
end if
return $G = \{T_1, \ldots, T_t\}$, $B = \left\{c_{i,k|\boldsymbol{S}} \text{ in } T_j, j = 1, \ldots, t\right\}$, $f_i(X_i)$

Steps 4 and 9 For R-vine selection, in each level the tree that maximizes the sum of absolute empirical Kendall's taus is built using the MST method. However, while in the first step ($j = 1$), the TDSH starts from a complete weighted graph, in subsequent steps it departs from a connected weighted graph (not complete in general) with the edges that meet the proximity condition. For the selection of a C-vine, the algorithm chooses as root node the node that maximizes the sum of absolute pairwise dependencies to this node. In the case of a D-vine, the path structure of the first tree determines the path structure of the subsequent trees. Therefore, the optimization of a D-vine graph can be formulated as the optimization of T_1, which translates into finding a Hamiltonian path that maximizes the sum of absolute pairwise dependencies. It can be transformed to a related Traveling Salesman Problem (TSP) [16]. As TSP is NP-hard, we use a heuristic called Cheapest Insertion [63, 64, 111] to find the best path of n nodes to build the first tree.

Steps 5 and 10 In general, one can select the same copula family for all the edges of the R-vine, and in this case, we only need to estimate their parameters, or select the suitable pair-copula family individually for each edge. In the latter case, we use the Cramér-von Mises statistic S_m (1.44). By comparing a parametric copula C_θ (chosen from a predefined group of candidate copula families $\mathcal{C}$) to the empirical copula C_m, the tested copula with the smallest S_m is selected. The corresponding pair-copula parameters are estimated via the Kendall's tau-based method (when the copula parameter is a scalar). The relationship of Kendall's tau and the dependence parameter of the bivariate Normal, Student's t, Clayton and Gumbel bivariate families are given in Table 1.1.

Step 7 The algorithm computes the transformed observations (i.e., conditional distribution functions) for the trees at levels $j > 2$ using the copula parameters from the previous tree and the recursive evaluation of (1.18). Notice that the only pair-copulas needed in this calculation are those already specified in the previous trees. The obtained transformed observations (also called copula data) at each level are used to compute the pairwise dependence measures as well as estimate the pair-copulas and their parameters for the subsequent level.

Steps 6, 11, 12 and 13 These steps are optional and have to do with learning truncated R-vines (Definition 9). AIC (1.48) or BIC (1.49) is computed for the two models $tRV\,(t_0)$ (Step 6) and $tRV\,(t_0 + 1)$ (Step 11). An R-vine can be truncated either at level t_0 (Step 12), which is identified by the AIC/BIC-based truncation heuristic described in Section 1.7, or at a truncation level t (Step 13), which is given as an input parameter of the TDSH.

Appendix A is a simple illustration of the TDSH using a four-variable example.

1.9 Bayesian Networks

This section provides basic definitions on directed graphs, BNs, polytrees, and some concepts on dependence maps used in the context of GMs, which are the core of Chapter 2 focused on the relationship between graphical representations of R-vines and polytrees.

1.9.1 Directed Graphs

In this section, we follow the terminology used in [6], adapting the notation to that used in this thesis in order to ensure its consistency.

Let $G = (N, E)$ be a graph with node set N and edge set E, where $V \neq \emptyset$ is a finite set and $E \subseteq \mathcal{E} = \{(v, w)\,|v, w \in N, v \neq w\}$. G is said to be complete if every pair of nodes is joined by

an edge. G is said to be totally non-connected if $E = \emptyset$. Directed graphs only contain directed edges, which are represented with the symbol "$\rightarrow$". If G contains the directed edge $v \rightarrow w$, then $(v, w) \in E$, but $(w, v) \notin E$. If $v \rightarrow w$, v is then referred to as a parent of w, and w as the child of v. Undirected graphs only contain undirected edges, which are represented with the symbol "–". If G contains the undirected edge v–w, then $(v, w), (w, v) \in E$. The skeleton of a directed graph is the undirected graph obtained by replacing all directed edges with undirected edges. A path from v_1 to v_d is a sequence of distinct nodes $v_1, \ldots, v_d \in N$, $d \geq 2$, connected by edges in G, where $(v_i, v_{i+1}) \in E$, $i = 1, \ldots, d-1$. A cycle is a path where $v_1 = v_d$. In particular, a directed path from v_1 to v_d is a directed cycle if $v_1 = v_d$. A graph containing only directed edges and without directed cycles is known as a directed acyclic graph (DAG). In DAGs, we can refer to sets of parents (pa_v), ancestors (an_v), descendants (de_v), and non-descendants (nd_v) of v, defined as follows. For $v, w \in N$, $v \neq w$:

- $pa_v = \{w \,|(w, v) \in E\}$ is the set of parents of v. Analogously, v is a child of w.
- $an_v = \{w \,|G \text{ contains a path from } w \text{ to } v\}$ is the set of ancestors of v.
- $de_v = \{w \,|G \text{ contains a path from } v \text{ to } w\}$ is the set of descendants of v.
- $nd_v = N \setminus \{v\} \cup de_v$ is the set of non-descendants of v.

1.9.2 Graphical Independence in Directed Graphs

A separation concept serves to translate topological properties over a graph to conditional independence of a distribution. The graphical independence criterion in DAGs, called D-separation, provides a semantic interpretation of a DAG in terms of the independence relationships encoded in the graph. A formal definition of this concept is given as follows.

Definition 12 (D-separation) *Let $G = (N, E)$ be a DAG. If $\boldsymbol{I}, \boldsymbol{K}, \boldsymbol{S}$ are three disjoint subsets of nodes in G, then $\boldsymbol{S}$ separates $\boldsymbol{I}$ from $\boldsymbol{K}$ or, similarly, $\boldsymbol{I}$ and $\boldsymbol{K}$ are separated given $\boldsymbol{S}$ if all the paths between any node of $\boldsymbol{I}$ and any node of $\boldsymbol{K}$ are blocked by $\boldsymbol{S}$. In other words, $\boldsymbol{S}$ separates $\boldsymbol{I}$ and $\boldsymbol{K}$, denoted as $I(\boldsymbol{I}, \boldsymbol{K} \,|\boldsymbol{S})$, if and only if along any undirected path between any node of $\boldsymbol{I}$ and any node of $\boldsymbol{K}$ there is an intermediate node A such that:*

- *Either A is a head-to-head node in the path and neither A nor its descendants are in $\boldsymbol{S}$.*

or

- *A is not a head-to-head node in the path and it is in $\boldsymbol{S}$.*

Let $G = (N, E)$ be a DAG on $n = |N|$ nodes and $\boldsymbol{I}, \boldsymbol{K}, \boldsymbol{S} \subseteq N = \{1, \ldots, n\} \in \mathbb{N}$ be pairwise disjoint sets. Furthermore, $P(\boldsymbol{X})$ denotes the probability distribution on $\boldsymbol{X}$, and $P_{\boldsymbol{I}}$, $P_{\boldsymbol{K}}$, $P_{\boldsymbol{S}}$ denote the corresponding $\boldsymbol{I}, \boldsymbol{K}, \boldsymbol{S}$-marginal distributions of P respectively. Therefore, $\boldsymbol{X}_{\boldsymbol{I}} \sim P_{\boldsymbol{I}}$, $\boldsymbol{X}_{\boldsymbol{K}} \sim P_{\boldsymbol{K}}$, and $\boldsymbol{X}_{\boldsymbol{S}} \sim P_{\boldsymbol{S}}$. If $\boldsymbol{I} = \{i\}$ (i.e., $|\boldsymbol{I}| = 1$), we write X_i and P_i. Furthermore, the conditionally independence of $\boldsymbol{X}_{\boldsymbol{I}}$ and $\boldsymbol{X}_{\boldsymbol{K}}$ given $\boldsymbol{X}_{\boldsymbol{S}}$ is denoted as $I(\boldsymbol{X}_{\boldsymbol{I}}, \boldsymbol{X}_{\boldsymbol{K}} \,|\boldsymbol{X}_{\boldsymbol{S}})$. The following properties associate conditional independencies of P with separations of G:

P is say to exhibit the local and global G-Markovian properties if

$$I\left(X_i, \boldsymbol{X}_{nd_i \setminus pa_i} \,|\boldsymbol{X}_{pa_i}\right) \; \forall i \in N, \tag{1.53}$$

and

$$I(\boldsymbol{I}, \boldsymbol{K} \,|\boldsymbol{S}) \implies I(\boldsymbol{X}_{\boldsymbol{I}}, \boldsymbol{X}_{\boldsymbol{K}} \,|\boldsymbol{X}_{\boldsymbol{S}}) \; \forall \text{ disjoint } \boldsymbol{I}, \boldsymbol{K}, \boldsymbol{S} \subseteq N, \tag{1.54}$$

respectively. If P satisfies (1.53) and (1.54), it is called G-Markovian.

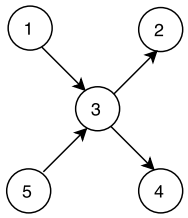


Figure 1.5: A five-dimensional polytree. According to this structure, $f(\boldsymbol{X})$ factorizes as $f(X_1) \cdot f(X_5) \cdot f(X_3 | X_1, X_5) \cdot f(X_2 | X_3) \cdot f(X_4 | X_3)$.

1.9.3 Bayesian Networks and Polytrees

Definition 13 (Bayesian Network) *A BN is a pair $(G(N,E), P)$, where G is a DAG, N denotes the set of nodes and each node represents a random variable, and E denotes the set of edges. Furthermore, $P = \{f(X_1 | \boldsymbol{X}_{pa_1}), \cdots, f(X_n | \boldsymbol{X}_{pa_n})\}$ is a set of n conditional probability distributions (one for each variable), where $\boldsymbol{X}_{pa_i}$, $i = 1, \ldots, n$, is the set of parents of X_i in G. The set P defines a probability density function given by*

$$f(\boldsymbol{X}) = \prod_{i=1}^{n} f(X_i | \boldsymbol{X}_{pa_i}). \tag{1.55}$$

A polytree is a BN for which the skeleton of the DAG is both connected and acyclic, i.e., a tree. In polytrees, there is no more than one undirected path that connects any two nodes. The number of edges in a polytree is $n-1$. Figure 1.5 shows a five-dimensional polytree and the corresponding factorization of $f(\boldsymbol{X})$. Particular types of polytrees include chains: each node has at most one parent and/or only one child, and trees: each node has only one parent. A comprehensive introduction to graph theory and graphical models can be found in [33, 86].

In a polytree, we can find different types of connections:

- A head-to-head connection is a subgraph $i \rightarrow s \leftarrow k$ in which s is a node with convergent edges, also called a head-to-head node.
- A fork connection is a subgraph $i \leftarrow s \rightarrow k$ in which s is a node with divergent edges.
- A chain connection is a subgraph $i \rightarrow s \rightarrow k$ (or $i \leftarrow s \leftarrow k$) in which the edges are directed towards the same direction.

According to the concept of D-separation (Definition 12), the head-to-head connection indicates that i and k are non-separated by s in G, since the path between i and k has s as a head-to-head node. This connection represents the conditional dependence relationship of X_i and X_k given X_s ($D(X_i, X_k | X_s)$) in the probability distribution represented by G. On the other hand, the fork and chain connections indicate that i and k are separated by s in G, since in the path between i and k, s is neither a head-to-head node nor a descendant of a head-to-head node. These connections represent the conditional independence relationship of X_i and X_k given X_s ($I(X_i, X_k | X_s)$) in the probability distribution represented by G. It is worth noticing that, $D(X_i, X_k | X_s)$ is represented by a unique head-to-head connection, while $I(X_i, X_k | X_s)$ can be represented by different connections.

1.9.4 Dependence Models

Every probabilistic model $p(\boldsymbol{x})$ has an associated dependence model. While $p(\boldsymbol{x})$ contains a complete description (quantitative and qualitative) of the relationships among random variables, the dependence model only describes its qualitative structure [27], which can be obtained using different approaches: from directed or undirected graphs, a dependence list, or a set of conditional probability functions. Here we focus on dependence models defined graphically. An advantage of using the graphical approach is that the graph defines a factorization of $p(\boldsymbol{x})$ as a product of conditional probability functions.

The following definitions, provided in [27], concern types of correspondences (mapping) between graphs and dependence models, namely I-map, D-map and P-map.

Definition 14 (Dependence model) *Any model M of a set of random variables $\boldsymbol{X}$ that allows verifying whether $\boldsymbol{X_I}$ and $\boldsymbol{X_K}$ are conditionally independent or dependent given $\boldsymbol{X_S}$, denoted as $I(\boldsymbol{X_I}, \boldsymbol{X_K} | \boldsymbol{X_S})$ and $D(\boldsymbol{X_I}, \boldsymbol{X_K} | \boldsymbol{X_S})$ respectively, for all possible disjoint subsets $\boldsymbol{X_I}, \boldsymbol{X_K}, \boldsymbol{X_S} \subseteq \boldsymbol{X}$, where $\boldsymbol{I}, \boldsymbol{K}, \boldsymbol{S} \subseteq \{1, \ldots, n\} \in \mathbb{N}$, and $|\boldsymbol{X_I}|, |\boldsymbol{X_K}| \geq 1$, is called a dependence model.*

Definition 15 (I-map) *A graph G is an independence map of a dependence model M if all separations derived from G are verified by M, that is,*

$$I(\boldsymbol{I}, \boldsymbol{K} | \boldsymbol{S})_G \implies I(\boldsymbol{X_I}, \boldsymbol{X_K} | \boldsymbol{X_S})_M . \tag{1.56}$$

Definition 16 (D-map) *A graph G is a dependence map of a dependence model M if all non-separations derived from G are verified by M, that is,*

$$D(\boldsymbol{I}, \boldsymbol{K} | \boldsymbol{S})_G \implies D(\boldsymbol{X_I}, \boldsymbol{X_K} | \boldsymbol{X_S})_M . \tag{1.57}$$

Definition 17 (P-map) *A graph G is a perfect map or faithful to a dependence model M, if every independence relationship obtained from G can also be obtained from M, and vice versa, that is,*

$$I(\boldsymbol{I}, \boldsymbol{K} | \boldsymbol{S})_G \iff I(\boldsymbol{X_I}, \boldsymbol{X_K} | \boldsymbol{X_S})_M . \tag{1.58}$$

From Definition 17, it follows that a P-map has to be simultaneously an I-map and a D-map.

An I-map G of M includes some of the independence relationships of M, but not necessarily all of them. An I-map guarantees that the separated nodes correspond to independent variables in M, but does not guarantee that connected nodes correspond to dependent variables in M. On the other hand, a D-map G of M includes some of the dependence relationships of M, but not necessarily all of them. A D-map guarantees that connected nodes correspond to dependent variables in M, but does not guarantee that the d-separated nodes correspond to independent variables in M. Moreover, it is not always possible to build a graph G that is a P-map of a given M. Nevertheless, every M has associated a trivial I-map and D-map, since any complete graph is a trivial I-map of any M, and a totally unconnected graph is a trivial D-map of any M.

Lauritzen's et al. theorem [87] relates the concept of an I-map and the factorization of $p(\boldsymbol{x})$ as follows.

Definition 18 (Factorization of probability functions) *Given a probability function $p(\boldsymbol{x})$ and a graph G, the following two conditions are equivalent:*

- $p(\boldsymbol{x})$ *factorizes according to* G.
- G *is an I-map of a dependence model* M *of* $p(\boldsymbol{x})$.

Theorem 18 states that given G, a factorization of the probability function can be found. The set of parameters that define the model can be identified thereupon either given by an expert in the problem domain or estimated from data. The set of conditional probability functions together with the assigned parameter values is known as the *complete model structure*.

Chapter 2

R-separation Criterion, Regular Vine-Copulas and Polytrees

2.1 Introduction

Graphical models (GMs) are a powerful statistical tool to model uncertain events and describe the dependence structure among random variables in an intuitive graphical way [101,129]. This chapter concerns the relationship (or connection) between graphical representations of two types of GMs: R-vines and polytrees.

There exist differences in interpreting R-vines with respect to other GMs, such as Bayesian Networks (BNs) and Markov networks (MNs), regarding the dependence structure in the distribution represented by these models. In [9], the authors claim that: "A vine is a convenient tool with a graphical representation that makes it easy to describe which conditional specifications are being made for the joint distribution." R-vines differ from BNs and MNs in that "the concept of conditional independence is weakened to allow for various forms of conditional dependence". The authors illustrate these differences using the graphical representations in Figure 2.1, which correspond to a BN, a MN, and an R-vine on three variables, from left to right respectively. While in the BN and MN models, X_1 and X_3 are conditionally independent given X_2, in the R-vine, in contrast, X_1 and X_3 are conditionally dependent given X_2.

In general, BNs have well-studied mathematical properties that have been developed throughout decades. In contrast, interest in R-vines has grown in the last few years. In the literature, the relationship between BNs and PCCs has been approached from different perspectives. In [65], a

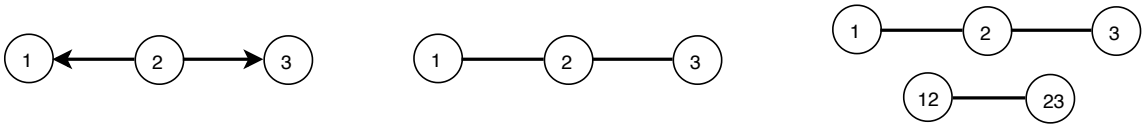


Figure 2.1: Examples of graphical representations of a BN, a MN, and an R-vine on three variables, from left to right respectively. These models highlight the differences in interpreting R-vines with respect to other graphical models. While in the BN and MN models, X_1 and X_3 are conditionally independent given X_2, in the R-vine, in contrast, X_1 and X_3 are conditionally dependent given X_2.

non-parametric BN, as an alternative to a particular subclass of R-vines, is proposed. There is also a discussion on the differences between both models as well as guidelines on when to use one or the other from a quantitative perspective. In [6, 7], a BN that uses pair-copulas of different families is built using the PCC framework. However, the procedure for learning the graph structure of the BN (i.e., the graph) requires the computation of multidimensional integration, necessary to compute the conditional distributions that are arguments of pair-copulas. In [62], the proposed approach is restricted to the subclass of R-vines, which means that there is a computationally efficient procedure for learning BNs. In [83], procedures that build a C-vine or a D-vine graph from the joint factorization corresponding to a DAG, and vice versa, are proposed.

However, the problem of verifying whether the pairwise graphical independencies found in a polytree can be represented in an R-vine, and vice versa, has not yet been answered. Here, we address this matter in both directions: to build the R-vine graph that represents the largest number of pairwise (un)conditional independencies represented in a polytree graph, and vice versa.

To accomplish this task, two challenges have to be overcome: First, in order to extract the list of separation relationships represented in the R-vine graph, an appropriate graphical separation criterion is required. However, as far as we know, such a concept has not yet been defined for R-vines. In this thesis, a concept of graphical separation for R-vines, called R-separation, is formulated. Secondly, it turns out that the edges of the R-vine graph only indicate which pair-copulas are present in the decomposition, but not whether the relationship they represent is that of independence or dependence. In order to differentiate the pair-copula relationship, we use an improved representation of the R-vine graph, in which its edges indicate the type of the relationship they represent [83].

The defined R-separation concept leads to a theorem that in turn establishes the possible correspondences or maps of (in)dependence between the R-vine graph and the associated R-vine copula. Furthermore, properties of R-separation such as symmetry, decomposition, contraction, intersection, (strong/weak) union, and (strong/weak) transitivity are analyzed.

Based on R-separation, two algorithms that build a polytree graph that encodes as many pairwise relationships as possible derived from an R-vine graph, and vice versa, are designed. These algorithms are useful as they make it clear that properties and procedures that can be applied to polytrees can be carried over to R-vines, and vice versa. The use of these algorithms is illustrated through examples.

The chapter is organized as follows. Section 2.2 introduces the R-separation criterion. Section 2.3 presents a theorem on possible dependence maps between the R-vine graph and the associated R-vine copula. Section 2.4 discusses the R-separation properties. Section 2.5 explores the relationship between graphical representations of R-vines and polytrees and Section 2.6 provides the conclusions of the chapter.

2.2 R-separation

In this section, the concept of graphical separation for R-vines is formulated. As we have already discussed in the introductory section of this chapter, in defining the R-separation concept, it should be taken into account the lack of graphical expressiveness of R-vines in the sense that, while the edges in DAGs specify conditional (in)dependence conditions (see Section 1.9.2), the edges of R-vine graphs illustrate the required pair-copulas only [7]. Similar to DAGs, the nodes in the first tree of the R-vine graph represent univariate marginal distributions of the associated R-vine copula. However, in contrast to DAGs, the R-vine graph does not have an interpretation in terms of Markov properties of its associated R-vine copula [6]. Notice that there is a one-to-one correspondence between edges and (in)dependence relationships represented in the R-vine graph.

In order to make the R-vine graph more expressive, we adopt the extended representation used

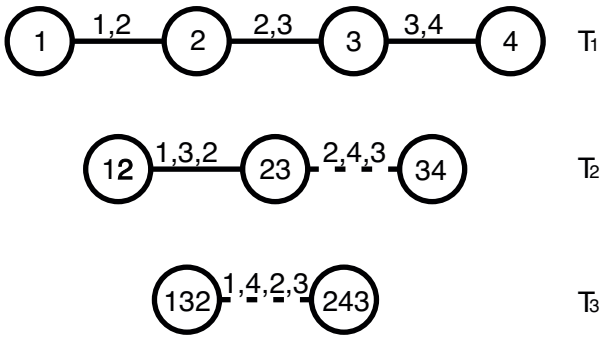


Figure 2.2: Four-dimensional R-vine to illustrate R-separation.

in [83], which modifies its graphical appearance by using two types of edges: dashed and continuous to indicate graphical separation and non-separation respectively. From now on, a discontinuous edge is denoted as $\langle i, k, \boldsymbol{S} \rangle$ and a continuous edge as $\{i, k, \boldsymbol{S}\}$.

The R-separation[1] concept formulated in this chapter works over R-vines with additional expressiveness. It is defined as follows.

Let $\boldsymbol{I} = \{1, \ldots, n\}$ be a finite set of indexes with $n \geq 2$, and $G = (T_1, T_2, \ldots, T_{n-1})$ be an R-vine graph composed of a hierarchy of $n-1$ trees T_j, where $j = 1, \ldots, n-1$ denotes the level of the tree in the hierarchy (see Definition 2). T_j has $n - j + 1$ nodes and $n - j$ edges. Moreover, $Na \subseteq \boldsymbol{I}$ and $Nb \subseteq \boldsymbol{I}$ are two nodes of the j^{th} tree with cardinality $|Na| = |Nb| = j$. Furthermore, let $i,k, \boldsymbol{S} \subseteq \boldsymbol{I}$ be disjoint subsets. If $\boldsymbol{i} = \{i\}$ and $\boldsymbol{k} = \{k\}$, which implies that $|\boldsymbol{i}| = 1$ and $|\boldsymbol{k}| = 1$, we write i and k (not in boldface) respectively.

Definition 19 (R-separation criterion) *Given an R-vine graph $G = (T_1, \ldots, T_{n-1})$ on n indexes, the disjoint subsets $i, k \subseteq \boldsymbol{I}$ are graphically separated in G by a set $\boldsymbol{S} \subseteq \boldsymbol{I} \backslash i \cup k$, $|\boldsymbol{S}| = j-1$, which is denoted as $I(i, k \,|\boldsymbol{S})$, if there exist two nodes Na and Nb in T_j with $i \in Na$ and $k \in Nb$ such that the following composite condition is met: Na and Nb are adjacent, $\boldsymbol{S} = Na \cap Nb$, and Na and Nb are joined by a dashed edge $\langle i, k, \boldsymbol{S} \rangle$.*

Based on the R-separation concept, a non-separation criterion for R-vines can be defined as follows.

Definition 20 (Non-separation concept) *Given an R-vine graph $G = (T_1, \ldots, T_{n-1})$ on n indexes, the disjoint subsets $i, k \subseteq \boldsymbol{I}$ are graphically non-separated in G by a set $\boldsymbol{S} \subseteq \boldsymbol{I} \setminus i \cup k$, $|\boldsymbol{S}| = j-1$, which is denoted as $D(i, k \,|\boldsymbol{S})$, if there exist two nodes Na and Nb in T_j with $i \in Na$ and $k \in Nb$ such that the following composite condition is met: Na and Nb are adjacent, $\boldsymbol{S} = Na \cap Nb$, and Na and Nb are joined by a continuous edge $\{i, k, \boldsymbol{S}\}$.*

Let us consider the R-vine graph shown in Figure 2.2. The set of graphical relationships derived from this R-vine graph using the R-separation criterion are listed in Tables 2.1-(a) and (b) respectively.

Notice that:

- A dashed edge $\langle i, k, \boldsymbol{S} \rangle$ denotes a separation relationship $I(i, k \,|\boldsymbol{S})$, meaning that i and k are separated by $\boldsymbol{S}$ in G.

[1] The R-separation concept represents an evolution of that presented in [26].

Table 2.1: Separation and non-separation relationships derived from the four-dimensional R-vine graph shown in Figure 2.2 using the R-separation criterion.

Dashed Edges	Separations
$\langle 2,4,3\rangle$	$I(2,4\,\vert 3)$
$\langle 1,4,2,3\rangle$	$I(1,4\,\vert 2,3)$

(a) Separations.

Continuous Edges	Non-separations
$\{1,2\}$	$D(1,2)$
$\{2,3\}$	$D(2,3)$
$\{3,4\}$	$D(3,4)$
$\{1,3,2\}$	$D(1,3\,\vert 2)$

(b) Non-separations.

- A continuous edge $\{i,k,\boldsymbol{S}\}$ denotes a non-separation relationship $D(i,k\,|\boldsymbol{S})$, meaning that i and k are non-separated by $\boldsymbol{S}$ in G.

Notice that the separation and non-separation relationships represented in G occur only between pairs of single indexes i, k. Furthermore, notice that each pair of variables occurs once as conditioned variables of only one pair-copula in the associated R-vine copula [84]. A brief discussion on this topic was previously presented in Section 1.3.

By using the R-separation and non-separation criteria, we can easily list the pairwise separation and non-separation relationships represented in the R-vine graph, which correspond to (un)conditional independence and dependence relationships in the associated R-vine copula respectively.

The usefulness of having the R-separation criterion is twofold: From a practical point of view, it facilitates the task of listing separation and non-separation relationships by merely examining the topology of the graph and the edge types. From a methodological point of view, similarly to other GMs, the R-vine is provided with its own graphical separation criterion, thus filling this gap for this class of models.

2.3 Relationship Between R-vine Graphs and R-vine Copulas

Based on the R-separation criterion, we formalize the possible dependence maps (namely I-map, D-map and P-map in (1.56)-(1.58) respectively) between the R-vine graph and its associated R-vine copula in Theorem 2. Necessary definitions for the present section are provided in Section 14.

Theorem 2 (R-vine maps) *Let G be an R-vine graph and M a set of (un)conditional (in)dependencies associated to pair-copulas involved in the R-vine copula associated to G. The following statements are true:*

(i) *G is an I-map of M, which is written as*

$$I(i,k\,|\boldsymbol{S})_G \implies I(X_i,X_k\,|\boldsymbol{X_S})_M\,. \tag{2.1}$$

(ii) *G is a D-map of M, which is written as*

$$D(i,k\,|\boldsymbol{S})_G \implies D(X_i,X_k\,|\boldsymbol{X_S})_M\,. \tag{2.2}$$

(iii) *It cannot be deduced from G whether this graph is a P-map of M.*

One of the key points in the proof of the statements (i), (ii) and (iii) has to do with the Product copula. Let M be the set of (un)conditional (in)dependencies in the R-vine copula associated to G. Every edge of G joins two nodes Na and Nb, such that $i \in Na$, $k \in Nb$, and $\boldsymbol{S} = Na \cap Nb$. Moreover, each edge corresponds to a pair-copula in M, which is determined by the indexes of the edge $i, k, \boldsymbol{S}$. Then, we have:

- If $c_{i,k|\boldsymbol{S}} = c_{\mathrm{P}i,k|\boldsymbol{S}}$, then $I(X_i, X_k | \boldsymbol{X_S})$ is true and $\langle i, k, \boldsymbol{S} \rangle$ is a dashed edge in G.
- If $c_{i,k|\boldsymbol{S}} \neq c_{\mathrm{P}i,k|\boldsymbol{S}}$, then $D(X_i, X_k | \boldsymbol{X_S})$ is true and $\{i, k, \boldsymbol{S}\}$ is a continuous edge in G.

Proof

(i) Suppose that the dashed edge $\langle i, k, \boldsymbol{S} \rangle$ joins two nodes $Na, Nb \in T_j$, $i \in Na$, $k \in Nb$, and $|\boldsymbol{S}| = j - 1$. Therefore,

$$\langle i, k, \boldsymbol{S} \rangle \in T_j \Longrightarrow I(i, k | \boldsymbol{S}) \Longrightarrow c_{i,k|\boldsymbol{S}} = c_{\mathrm{P}i,k|\boldsymbol{S}} \in M \Longrightarrow I(X_i, X_k | \boldsymbol{X_S}). \tag{2.3}$$

Using the R-separation criterion (Definition 19), it follows that G is an I-map of M, which is written as

$$I(i, k | \boldsymbol{S})_G \Longrightarrow I(X_i, X_k | \boldsymbol{X_S})_M. \tag{2.4}$$

(ii) Suppose that the continuous edge $\{i, k, \boldsymbol{S}\}$ joins two nodes $Na, Nb \in T_j$, $i \in Na$, $k \in Nb$, and $|\boldsymbol{S}| = j - 1$. Therefore,

$$\{i, k, \boldsymbol{S}\} \in T_j \Longrightarrow D(i, k | \boldsymbol{S}) \Longrightarrow c_{i,k|\boldsymbol{S}} \neq c_{\mathrm{P}i,k|\boldsymbol{S}} \in M \Longrightarrow D(X_i, X_k | \boldsymbol{X_S}). \tag{2.5}$$

Using the non-separation criterion (Definition 20), it follows that G is a D-map of M, which is written as

$$D(i, k | \boldsymbol{S})_G \Longrightarrow D(X_i, X_k | \boldsymbol{X_S})_M. \tag{2.6}$$

(iii) Although G is an I-map and a D-map of M, it is not necessarily a P-map of M, since, from the R-vine graph, it is not possible to infer (non)separations other than those represented by edges in G.

Let us illustrate this matter using the example offered in [79] for the three-dimensional R-vine graph shown in Figure 2.3. Here, $c_{1,2}$ and $c_{2,3}$ are associated to the edges $\{1, 2\}$ and $\{2, 3\}$, and specify the dependence relationships $D(X_1, X_2)$ and $D(X_2, X_3)$ in M, respectively. The third unconditional pair-copula $c_{1,3}$ can be obtained via one-dimensional integration as

$$c_{1,3}(u_1, u_3) = \int_0^1 c(u_1, v, u_3) \, \partial v. \tag{2.7}$$

However, $c_{1,3}$ is not associated with any edge in G.

□

In summary, the proof of this theorem leads us to conclude that R-vines do not allow a graphical separation concept that yields a complete independence map to be defined.

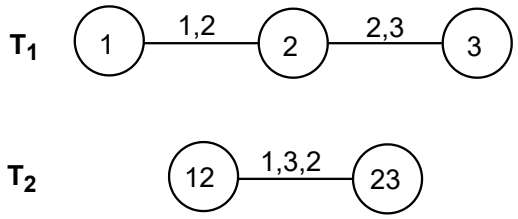


Figure 2.3: A three-dimensional R-vine graph.

2.4 Properties of R-separation

In the field of GMs, properties of conditional (in)dependence have been defined which, depending on the type of graph, can be represented graphically. The usefulness of describing these properties from the graphical perspective lies in the fact that, from the initial set of separation/non-separation relationships, it is possible to derive new ones. An in-depth discussion on conditional (in)dependence properties of graphical models is presented in [27].

Graphical properties for R-separation analyzed in this section include symmetry, decomposition, contraction, intersection, (strong/weak) union, and (strong/weak) transitivity.

To make the description of these properties easier, when it is pertinent, we illustrate them graphically in Figures 2.4-2.5. In order to facilitate the interpretation of these figures, only the right side of the implication relationship is represented. The figure legend reads as follows: Indexes belonging to $\boldsymbol{S}$ appear in bold in the node and over the edge, whereas i and k appear underlined in shaded nodes.

Let $i, k, w, \boldsymbol{S}$ be disjoint subsets of $\{1, \ldots, n\}$.

Symmetry The property of symmetry states that:

$$I\left(i, k \,|\boldsymbol{S}\right) \Longleftrightarrow I\left(k, i \,|\boldsymbol{S}\right). \tag{2.8}$$

It can be verified that R-separation meets symmetry on G, since graphical pairwise relationships do not depend on the order of the pair i and k. Notice that when $I\left(i, k \,|\boldsymbol{S}\right)$ in M, separations in G correspond to dashed edges of adjacent nodes Na and Nb, with $i \in Na$ and $k \in Nb$, or vice versa, $k \in Na$ and $i \in Nb$.

Strong transitivity The property of strong transitivity states that:

$$D\left(i, w \,|\boldsymbol{S}\right) \wedge D\left(w, k \,|\boldsymbol{S}\right) \Longrightarrow D\left(i, k \,|\boldsymbol{S}\right). \tag{2.9}$$

It can be verified that R-separation does not meet strong transitivity on G. A counter-example can be seen in Figure 2.4: In the case where $i = \{4\}$, $k = \{5\}$, $w = \{2\}$, and $S = \{3\}$, it can be verified that although $D\left(4, 2 \,|3\right)$ and $D\left(2, 5 \,|3\right)$ (on the left of (2.9)) are true by Definition 20, the edge $\{4, 5, 3\}$ associated with $D\left(4, 5 \,|3\right)$ (on the right of (2.9)) does not belong to G. In addition, the edge $\{4, 5, 3\}$ is not in G because it would create a cycle with $\{4, 2, 3\}$ and $\{2, 5, 3\}$.

Summarizing, R-vine graphs cannot represent graphical transitivity relationships. It can be verified that $D\left(i, w \,|\boldsymbol{S}\right)$ and $D\left(w, k \,|\boldsymbol{S}\right)$ are associated with the edges $\{i, w, \boldsymbol{S}\}$ and $\{w, k, \boldsymbol{S}\}$,

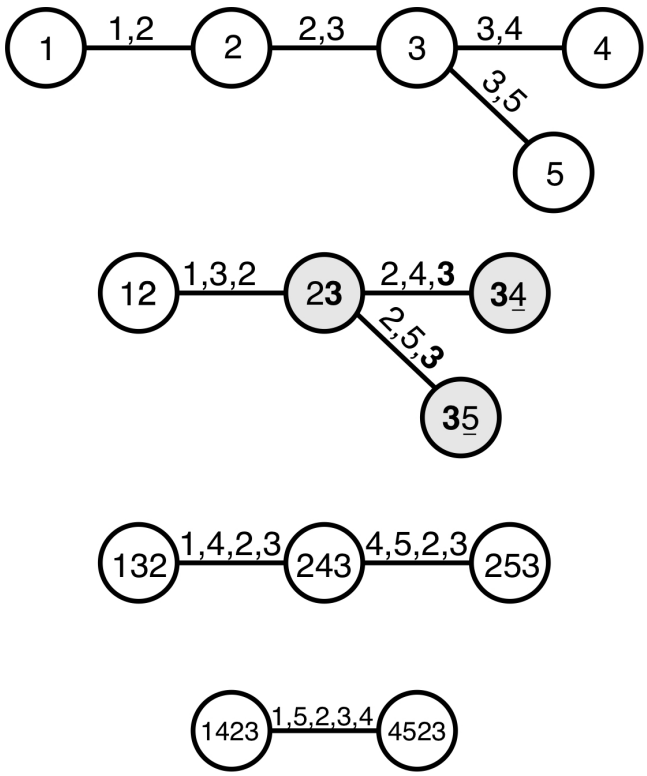


Figure 2.4: Illustration of strong transitivity violation: In the case where $i = \{4\}$, $k = \{5\}$, $w = \{2\}$, and $S = \{3\}$, it can be verified that although $D(4, 2\,|3)$ and $D(2, 5\,|3)$ (on the left of (2.9)) are true by Definition 20, the edge $\{4, 5, 3\}$ associated with $D(4, 5\,|3)$ (on the right of (2.9)) does not belong to G. In addition, the edge $\{4, 5, 3\}$ is not in G because it would create a cycle with $\{4, 2, 3\}$ and $\{2, 5, 3\}$.

respectively. Therefore, the edge $\{i, k, \boldsymbol{S}\}$ associated with $D(i, k\,|\boldsymbol{S})$ cannot be in G, since it would violate the tree restriction of having no cycles (even though it meets the proximity condition).

Weak transitivity The property of weak transitivity states that:

$$D(i, w\,|\boldsymbol{S}) \wedge D(w, k\,|\boldsymbol{S}) \Longrightarrow D(i, k\,|\boldsymbol{S}) \wedge D(i, k\,|\boldsymbol{S} \cup w)\,. \tag{2.10}$$

It can be verified that R-separation does not meet weak transitivity on G. A counter-example can be seen in Figure 2.5: In the case where $i = \{4\}$, $k = \{5\}$, $w = \{3\}$, and $S = \emptyset$, it can be verified that although $D(4, 3)$ and $D(3, 5)$ (on the left of (2.10)) are true by Definition 20, the edges $\{4, 5\}$ and $\{4, 5, 3\}$ associated with $D(4, 5)$ and $D(4, 5\,|3)$ (on the right of (2.10)), respectively, are not in G, since these non-separations would create cycles as well as the pair 4 and 5 would be involved as a conditioned set in more than one relationship.

Summarizing, it can be verified that $D(i, w\,|\boldsymbol{S})$ and $D(w, k\,|\boldsymbol{S})$ are associated with the edges $\{i, w, \boldsymbol{S}\}$ and $\{w, k, \boldsymbol{S}\}$, respectively, whereas the edges $\{i, k, \boldsymbol{S}\}$ and $\{i, k, \boldsymbol{S}, w\}$ are not in G, since they would violate the tree restriction of having no cycles and also the R-vine property that every pair i and k occurs once as a conditioned set of only one relationship.

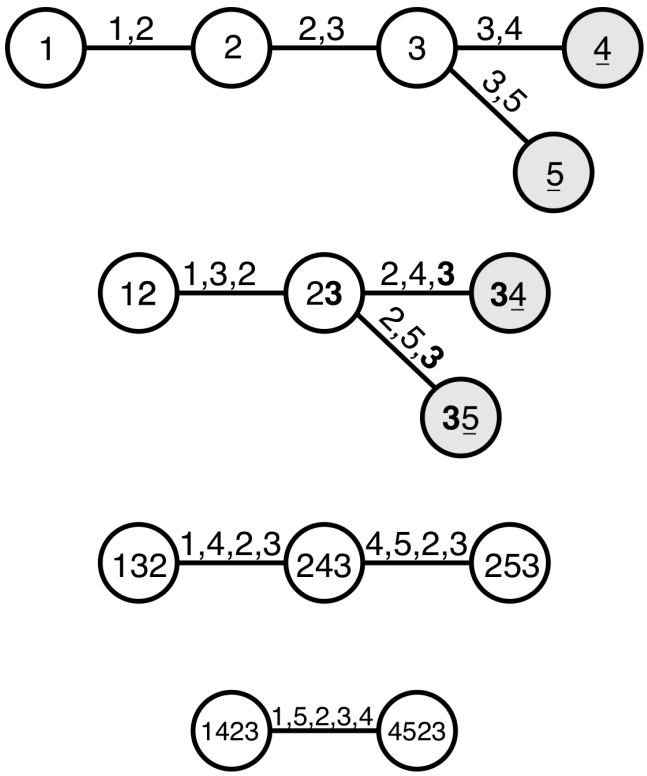


Figure 2.5: Illustration of weak transitivity violation: In the case where $i = \{4\}$, $k = \{5\}$, $w = \{3\}$, and $S = \emptyset$, it can be verified that although $D\,(4,3)$ and $D\,(3,5)$ (on the left of (2.10)) are true by Definition 20, the edges $\{4,5\}$ and $\{4,5,3\}$ associated with $D\,(4,5)$ and $D\,(4,5\,|3)$ (on the right of (2.10)), respectively, are not in G, since both they would create cycles and the pair 4 and 5 would be involved as a conditioned set in more than one relationship.

Strong union The property of strong union states that:

$$I\,(i,k\,|\boldsymbol{S}) \Longrightarrow I\,(i,k\,|\boldsymbol{S} \cup w)\,. \tag{2.11}$$

It can be verified that R-separation does not meet strong union on G. A counter-example can be seen in Figure 2.6: In the case where $i = \{1\}$, $k = \{4\}$, $w = \{5\}$, and $\boldsymbol{S} = \{2,3\}$, it can be verified that although $I\,(1,4\,|2 \cup 3)$ (on the left of (2.11)) is true, the edge $\langle 1,4,2,3,5\rangle$ associated with $I\,(1,4\,|2,3 \cup 5)$ (on the right of (2.11)) is not in G, since the pair 1 and 4 would be involved as a conditioned set in more than one relationship.

Summarizing, it can be verified that $I\,(i,k\,|\boldsymbol{S})$ is associated with the edge $\langle i,k,\boldsymbol{S}\rangle$, whereas the edge $\langle i,k,\boldsymbol{S},w\rangle$ is not in G, since it would violate the R-vine property that every pair i and k occurs once as a conditioned set of only one relationship (even though it meets the proximity condition).

The following properties cannot be verified since R-vine graphs represent relationships between pairs of single indexes only, not between sets of indexes, which is the case of the statement $I\,(i,k \cup w\,|\boldsymbol{S})$ that appears in all the properties listed in (2.12)-(2.15).

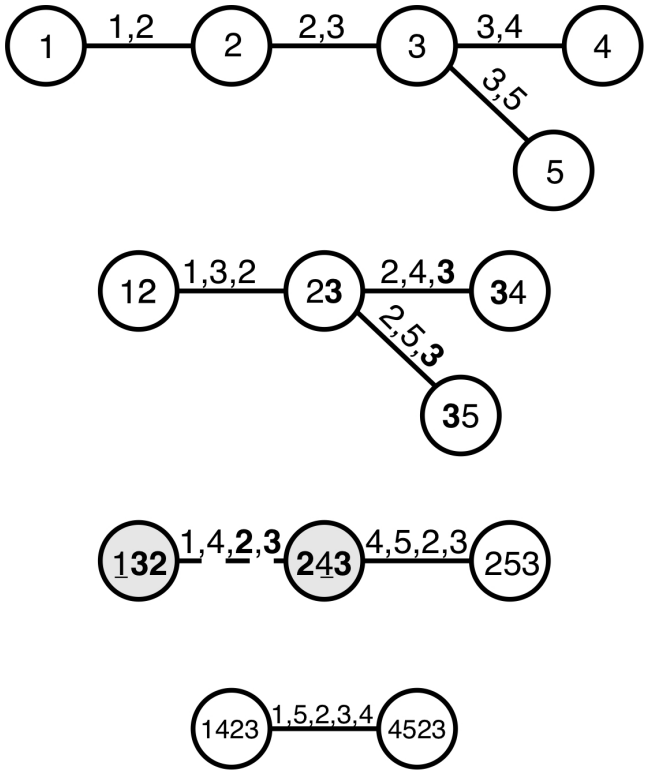


Figure 2.6: Illustration of strong union violation: In the case where $i = \{1\}$, $k = \{4\}$, $w = \{5\}$, and $\boldsymbol{S} = \{2, 3\}$, it can be verified that although $I\,(1, 4\,|2 \cup 3\,)$ (on the left of (2.11)) is true, the edge $\langle 1, 4, 2, 3, 5\rangle$ associated with $I\,(1, 4\,|2, 3 \cup 5\,)$ (on the right of (2.11)) is not in G, since the pair 1 and 4 would be involved as a conditioned set in more than one relationship.

Weak Union The property of weak union states that:

$$I\left(i, k \cup w \,|\boldsymbol{S}\right) \Longrightarrow I\left(i, w \,|\boldsymbol{S} \cup k\right) \ \wedge \ I\left(i, k \,|\boldsymbol{S} \cup w\right). \tag{2.12}$$

This property cannot be verified since R-vine graphs represent relationships between pairs of indexes only, not between sets of indexes, which is the case of $I\left(i, k \cup w \,|\boldsymbol{S}\right)_G$ on the left side in (2.12).

Decomposition The property of decomposition states that:

$$I\left(i, k \cup w \,|\boldsymbol{S}\right) \Longrightarrow I\left(i, k \,|\boldsymbol{S}\right) \ \wedge \ I\left(i, w \,|\boldsymbol{S}\right). \tag{2.13}$$

This property cannot be verified since R-vine graphs represent relationships between pairs of indexes only, not between sets of indexes, which is the case of $I\left(i, k \cup w \,|\boldsymbol{S}\right)$ on the left side in (2.13).

Contraction The property of contraction states that:

$$I\left(i, k \,|\boldsymbol{S}\right) \ \wedge I\left(i, w \,|\boldsymbol{S} \cup k\right) \Longrightarrow I\left(i, k \cup w \,|\boldsymbol{S}\right). \tag{2.14}$$

This property cannot be verified since R-vine graphs represent relationships between pairs of indexes only, not between sets of indexes, which is the case of $I\left(i, k \cup w \,|\boldsymbol{S}\right)$ on the right side in (2.14).

Intersection The property of intersection states that:

$$I\left(i, w \,|\boldsymbol{S} \cup k\right) \ \wedge \ I\left(i, k \,|\boldsymbol{S} \cup w\right) \Longrightarrow I\left(i, k \cup w \,|\boldsymbol{S}\right). \tag{2.15}$$

This property cannot be verified since R-vine graphs represent relationships between pairs of indexes only, not between sets of indexes, which is the case of $I\left(i, k \cup w \,|\boldsymbol{S}\right)$ on the right side in (2.15).

2.5 Relationship Between Graph Representations of R-vines and Polytrees

In this section, we study the relationship between graphical representations of R-vines and polytrees. The focus is on pairwise separations and non-separations encoded in one graph that correspond with the set of pairwise (in)dependencies of the dependence model M associated with the other graph. For this purpose, two algorithms are designed: One algorithm that aims to induce the R-vine graph that encodes as many pairwise relationships (either separations or non-separations) as possible derived from the polytree graph. The other algorithm achieves the same goal but in reverse, from the R-vine graph to the polytree graph. The output of these algorithms allows us to determine whether the output graph is an I-map or a D-map of the dependence model associated with the input graph.

Algorithm 2.1 Procedure for building an R-vine graph from a polytree graph.

Input
Polytree graph.
t – Number of trees. Possible values $1 \leq t \leq n-1$ (see Algorithm 1.1).

Output
$G = (T_1, \ldots, T_t)$ – R-vine graph.

1: Create L_I and L_D from the polytree graph via D-separation.
2: Build T_1 as the skeleton of the polytree graph with continuous edges $\{i, k\}$ ($\boldsymbol{S} = \emptyset$).
for $j = 2, \ldots, t$:
3: Build the graph $\mathcal{G} = \left(N^j, E^j\right)$, where $N^j = E^{j-1}$ in T_{j-1}, and E^j contains the edges allowed by the proximity condition.
4: Assign weights (one or zero) and edge types (dashed or continuous) to the edges in E^j of $\mathcal{G}$:
4.1: Set edges in E^j as dashed $\langle i, k, \boldsymbol{S}\rangle$ and assign them weight one.
for $D\left(i, k \,|\boldsymbol{S}\right)$ in L_D, where $|\boldsymbol{S}| = j - 1$:
if $\langle i, k, \boldsymbol{S}\rangle \in E^j$:
4.2: Replace the dashed edge $\langle i, k, \boldsymbol{S}\rangle$ (that it is in T_j) by the continuous edge $\{i, k, \boldsymbol{S}\}$, and the weight value is changed to zero in $\mathcal{G}$.
end if
end for
5: Create T_j from $\mathcal{G}$ using the MST method. Ties are resolved randomly.
end for
return $G = \{T_1, \ldots, T_t\}$

2.5.1 From Polytree Graphs to R-vine Graphs

Algorithm 2.1 presents the pseudo-code of a method capable of building an R-vine graph G (the target or the output graph) that encodes as many pairwise relationships as possible derived from a polytree graph (the starting or the input graph). This algorithm is described in the following.

Step 1 Create two lists L_I and L_D, which contain all pairwise separation $I\left(i, k \,|\boldsymbol{S}\right)$ and non-separation $D\left(i, k \,|\boldsymbol{S}\right)$ relationships, respectively, derived from the starting polytree graph via D-separation (Definition 12).

Step 2 Build T_1 of the target R-vine graph as the skeleton of the starting polytree graph. Notice that T_1 and the polytree skeleton have the same edge set, all of which are continuous.

Step 3 Subsequent trees of the target R-vine graph are built inside a for-loop that runs over $j = 2, \ldots, t$. Before building T_j, it is necessary to construct the graph $\mathcal{G} = \left(N^j, E^j\right)$ (usually, not complete). Nodes in N^j are the edges of the already built T_{j-1} of the target R-vine graph, while the edge set E^j contains those edges allowed by the proximity condition in T_j.

Step 4 Assigning weights to edges of $\mathcal{G}$ is carried out in two steps. At Step 4.1, all edges in E^j are set as dashed and weighted with value one by default. Next, at Step 4.2, the algorithm runs through those non-separations in L_D that could potentially be in T_j (i.e., unconditional relationships in T_1, conditional of order one in T_2, and so on) and checks whether the possible corresponding edges exist in E^j of $\mathcal{G}$. If that is the case, the dashed edge $\langle i, k, \boldsymbol{S}\rangle$ (of T_j) is replaced by the continuous

$\{i, k, \boldsymbol{S}\}$, and the weight value is set to zero in $\mathcal{G}$. When this step is completed, all edges of $\mathcal{G}$ have an assigned weight as per design they are assigned weight one by default at Step 4.1.

Step 5 The maximum spanning tree (MST) method (using Prim's algorithm [32]) chooses a MST from $\mathcal{G}$ over $n - j + 1$ nodes, which becomes the tree T_j of the target R-vine graph. It should be noted that one of the following scenarios can occur:

- First scenario: $\mathcal{G}$ is a tree. In this case, finding the MST can be omitted and $\mathcal{G}$ turns directly into T_j (see Example 1).
- Second scenario: $\mathcal{G}$ is a graph that contains a single MST (see Example 2).
- Third scenario: $\mathcal{G}$ is a graph that contains multiple MSTs. In this case, multiple R-vine graphs could potentially be generated from the same starting polytree graph, all of them encoding existing relationships in it (see Example 3).

In conclusion:

1. In the end, multiple R-vine graphs can be built from the same starting polytree graph. This is because, in polytrees, different graph connections (i.e., fork and chain) can represent the same set of separations (see Section 1.9.3).
2. In general, not all pairwise separations and non-separations encoded in the starting polytree graph can necessarily be represented in the built R-vine graph.
3. However, all pairwise separations and non-separations represented in the built R-vine graph are encoded in the starting polytree graph, since the $n(n-1)/2$ possible pairwise relationships inserted in the built R-vine graph are a subset of pairwise relationships represented in the starting polytree graph of n nodes. It is worth remembering that, while in the R-vine graph the pair i and k appears once as a conditioned set of only one relationship (either a separation or a non-separation), in contrast, in the polytree graph every pair i and k can be involved in multiple relationships with different conditioning sets.
4. Consequently, Algorithm 2.1 guarantees that the built R-vine graph is both an I-map and a D-map of the dependence model associated with the starting polytree graph.

In the following, Examples 1-3 illustrate Algorithm 2.1 through the three different scenarios referred to earlier.

Example 1 From the polytree graph to the R-vine graph (Algorithm 2.1 and Figure 2.7).

The derived L_1 and L_2 via D-separation (Step 1) from the four-dimensional polytree graph in Figure 2.7 are given by

$$\begin{aligned} L_I &= [I(1,3), I(1,4), I(2,4), \\ &\quad I(1,3|4), I(1,4|3), I(2,4|3), \\ &\quad I(1,4|2,3)]. \end{aligned}$$

$$\begin{aligned} L_D &= [D(1,2), D(2,3), D(3,4), \\ &\quad D(1,3|2), D(1,4|2), \\ &\quad D(1,2|3,4), D(1,3|2,4), D(2,3|1,4)]. \end{aligned}$$

The skeleton of the polytree graph turns into T_1 of the R-vine graph (Step 2). Therefore, in this tree, non-separations $D(1,2), D(2,3), D(3,4)$ in L_D are represented by the continuous edges

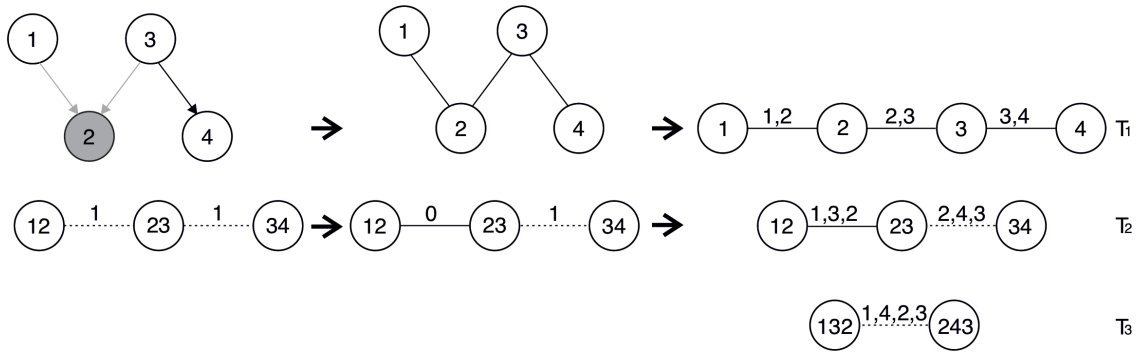


Figure 2.7: Illustration of Example 1 (Scenario 1), from the polytree graph to the R-vine graph (Algorithm 2.1): (From left to right, from top to bottom) Starting polytree graph; polytree skeleton; T_1 of the built R-vine graph encodes $D(1,2), D(2,3), D(3,4)$ in L_D; weighted graphs $\mathcal{G}$; T_2 encodes $I(2,4|3)$ in L_I and $D(1,3|2)$ in L_D, whereas T_3 encodes $I(1,4|2,3)$ in L_I respectively.

$\{1,2\}$, $\{2,3\}$ and $\{3,4\}$, whereas $I(1,3), I(1,4), I(2,4)$ in L_I cannot be represented in T_1, since the possible corresponding edges $\langle 1,3\rangle, \langle 1,4\rangle, \langle 2,4\rangle$ not in E^1 would create cycles.

Since T_1 has a chain structure, the structure of the subsequent trees is completely determined. This makes $\mathcal{G}$ a tree in itself (Step 3).

Initially, the two edges of $\mathcal{G}$, $\langle 1,3,2\rangle$ and $\langle 2,4,3\rangle$, are dashed with weight one by default (Step 4.1). Then, for $D(1,3|2), D(1,4|2)$ in L_D (the two non-separations that could eventually be represented in T_2) it is checked whether the possible corresponding edges exist in E^2. This is the case of the dashed edge $\langle 1,3,2\rangle$, which is replaced by the continuous $\{1,3,2\}$ and the weight value set to zero in $\mathcal{G}$ (Step 4.2). Consequently, T_2 encodes $D(1,3|2)$ in L_D and $I(2,4|3)$ in L_I. Here, the first scenario has taken place (Step 5). On the other hand, $I(1,3|4), I(1,4|3)$ in L_I and $D(1,4|2)$ in L_D cannot be represented in T_2 given that their possible corresponding edges $\langle 1,3,4\rangle$, $\langle 1,4,3\rangle$ and $\{1,4,2\}$, respectively, do not belong to this tree. At T_3 (the last tree of the built R-vine graph), its single dashed edge $\langle 1,4,2,3\rangle$ represents $I(1,4|2,3)$ in L_I. Therefore, $D(1,2|3,4), D(1,3|2,4), D(2,3|1,4)$ in L_D cannot be represented in T_3.

Conclusions from Example 1 Only one R-vine graph can be generated from the starting polytree graph. The built R-vine graph is both an I-map and a D-map of the dependence model M associated with the polytree graph since pairwise separations, $I(2,4|3), I(1,4|2,3)$ in L_I, and non-separations, $D(1,2), D(2,3), D(3,4), D(1,3|2)$ in L_D, represented in the built R-vine graph correspond to (in)dependence relationships of the dependence model associated with the starting polytree graph, namely $I(X_2, X_4|X_3)$, $I(X_1, X_4|X_2, X_3)$, $D(X_1, X_2)$, $D(X_2, X_3)$, $D(X_3, X_4)$, $D(X_1, X_3|X_2)$ in M.

Example 2 From the polytree graph to the R-vine graph (Algorithm 2.1 and Figure 2.8).

The derived L_1 and L_2 via D-separation (Step 1) from the four-dimensional polytree graph in Figure 2.8 are given by

$$\begin{aligned} L_I \;=\; & [I(2,3), I(2,4), I(3,4), \\ & I(2,4|1), I(3,4|1), \\ & I(2,4|1,3), I(3,4|1,3)]. \end{aligned}$$

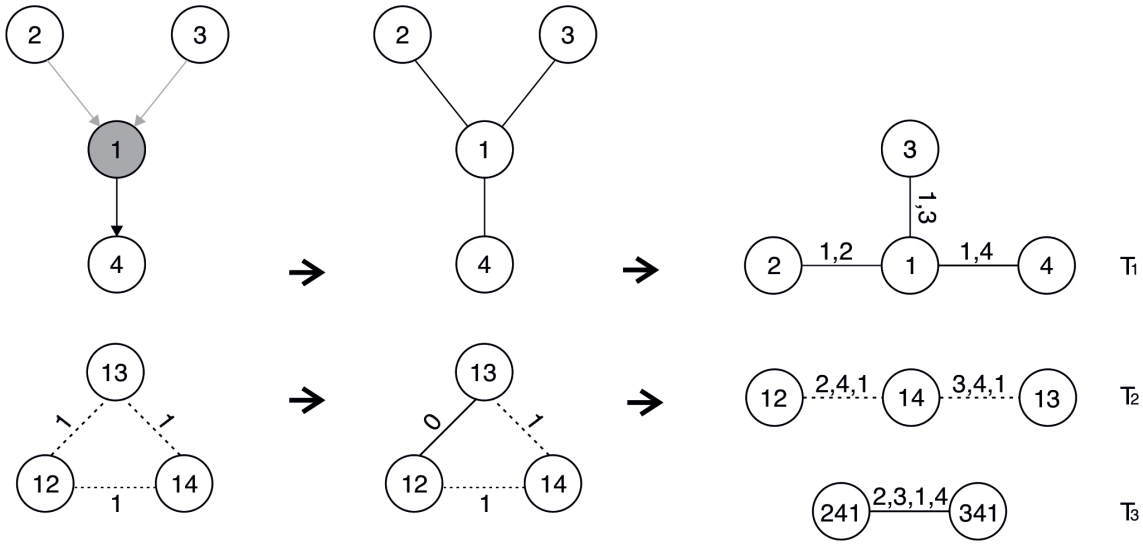


Figure 2.8: Illustration of Example 2 (Scenario 2), from the polytree graph to the R-vine graph (Algorithm 2.1): (From left to right, from top to bottom) Starting polytree graph; polytree skeleton; T_1 of the built R-vine graph encodes $D(1,2), D(1,3), D(1,4)$ in L_D; weighted graphs $\mathcal{G}$; T_2 encodes $I(2,4|1), I(3,4|1)$ in L_I, whereas T_3 encodes $D(2,3|1,4)$ in L_D respectively.

$$\begin{aligned} L_D \quad = \quad & [D(1,2), D(1,3), D(1,4), \\ & D(2,3|1), \\ & D(2,3|1,4)]. \end{aligned}$$

The skeleton of the polytree graph turns into T_1 of the R-vine graph (Step 2). Therefore, in this tree, non-separations $D(1,2), D(1,3), D(1,4)$ in L_D are represented by the continuous edges $\{1,2\}$, $\{1,3\}$ and $\{1,4\}$. Moreover, $I(2,3), I(2,4), I(3,4)$ in L_I cannot be represented in T_1, since the possible corresponding edges $\langle 2,3\rangle, \langle 2,4\rangle, \langle 3,4\rangle$ not in E^1 would create cycles.

Unlike Example 1, here $\mathcal{G}$ is a complete graph with three dashed edges, $\langle 2,3,1\rangle$, $\langle 2,4,1\rangle$ and $\langle 3,4,1\rangle$ (Step 3), with weight one by default (Step 4.1). Then, for $D(2,3|1)$ in L_D (the single non-separation that could eventually be represented in T_2) as its corresponding edge $\langle 2,3,1\rangle$ in E^2 is replaced by the continuous $\{2,3,1\}$, and the weight set to zero in $\mathcal{G}$ (Step 4.2). Notice that only two out of three edges of $\mathcal{G}$ can be inserted in T_2 given the tree restriction of having no undirected cycles.

The Prim's MST method used to build T_2 from $\mathcal{G}$ selects the dashed edges $\langle 2,4,1\rangle$ and $\langle 3,4,1\rangle$ (both having weight one) leaving out edge $\{2,3,1\}$ (having weight zero). Consequently, T_2 encodes $I(2,4|1), I(3,4|1)$ in L_I. Here, the second scenario has taken place (Step 5). On the other hand, $D(2,3|1)$ in L_D cannot be represented in T_2, because the possible corresponding edge $\{2,3,1\}$ does not belong to this tree. At T_3 (the last tree of the built R-vine graph), its single continuous edge $\{2,3,1,4\}$ represents the non-separation $D(2,3|1,4)$ in L_D (Figure 2.8) and , therefore, $I(2,4|1,3), I(3,4|1,3)$ in L_I cannot be represented in this tree.

Conclusions from Example 2 Only one R-vine graph can be generated from the starting polytree graph. The built R-vine graph is both an I-map and a D-map of the dependence model M associated with the polytree graph since pairwise separations, $I(2,4|1), I(3,4|1)$ in L_I, and non-separations, $D(1,2), D(1,3), D(1,4), D(2,3|1,4)$ in L_D, represented in the built R-vine graph correspond to (in)dependence relationships of the dependence model associ-

ated with the starting polytree graph, namely $I(X_2, X_4|X_1)$, $I(X_3, X_4|X_1)$, $D(X_1, X_2)$, $D(X_1, X_3)$, $D(X_1, X_4)$, $D(X_2, X_3|X_1, X_2)$ in M.

Example 3 From the polytree graph to the R-vine graph (Algorithm 2.1 and Figure 2.9).

Let us consider the four-dimensional polytree graph in Figure 2.9-(a). Notice that the polytree graph in this figure differs from that used in Example 2 (Figure 2.8) in that the edge joining nodes 1 and 4 has been reversed towards node 1. The present example illustrates how this seemingly small change leads to a different R-vine graph compared to that obtained in Example 2.

The derived L_1 and L_2 via D-separation (Step 1) from the four-dimensional polytree graph in Figure 2.9-(a) are given by

$$L_I = [I(2,3), I(2,4), I(3,4)].$$

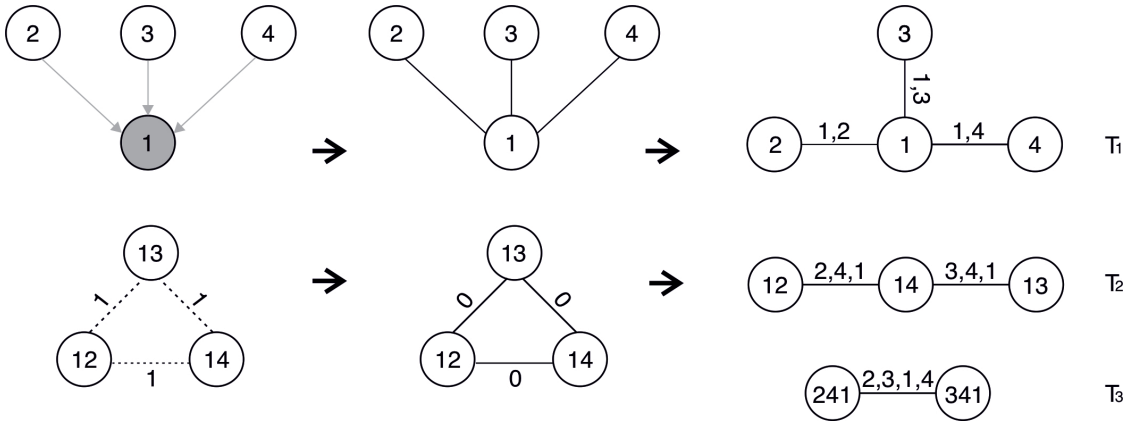
$$\begin{aligned} L_D = [&D(1,2), D(1,3), D(1,4),\\ &D(2,3|1), D(2,4|1), D(3,4|1),\\ &D(2,3|1,4), D(2,4|1,3), D(3,4|1,2)]. \end{aligned}$$

The skeleton of the polytree graph turns into T_1 of the R-vine graph (Step 2). Notice that the polytree skeleton in the present example (Figure 2.9-(a)) is the same as in Example 2 (Figure 2.8). In T_1, $D(1,2), D(1,3), D(1,4)$ in L_D are represented by the continuous edges $\{1,2\}$, $\{1,3\}$ and $\{1,4\}$. Moreover, $I(2,3), I(2,4), I(3,4)$ in L_I cannot be represented in this tree, since the possible corresponding edges $\langle 2,3\rangle, \langle 2,4\rangle, \langle 3,4\rangle$ not in E^1 would create cycles.

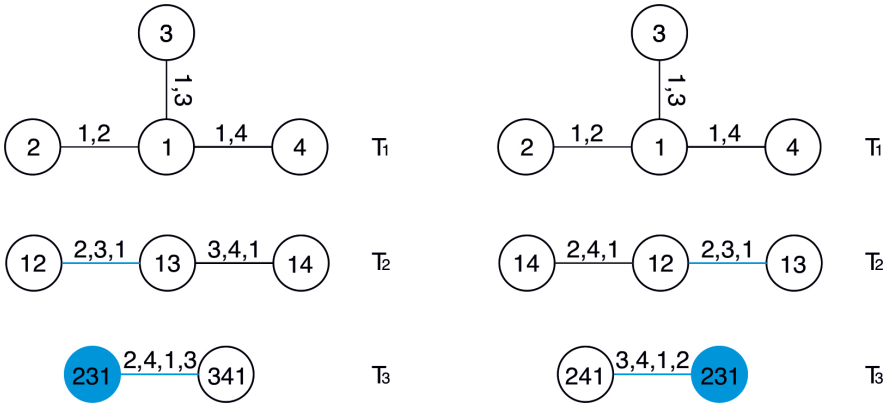
Analogously to Example 2, here $\mathcal{G}$ is a complete graph with three dashed edges, $\langle 2,3,1\rangle$, $\langle 2,4,1\rangle$ and $\langle 3,4,1\rangle$ (Step 3), with weight one by default (Step 4.1). Then, for $D(2,3|1), D(2,4|1), D(3,4|1)$ in L_D (the three non-separations that could eventually be represented in T_2) it is checked whether the possible corresponding edges exist in E^2. Being that true, the three dashed edges of $\mathcal{G}$ are replaced by the continuous $\{2,3,1\}$, $\{2,4,1\}$ and $\{3,4,1\}$, and the weight set to zero in $\mathcal{G}$ (Step 4.2). It turns out that since the three edges in $\mathcal{G}$ have the same weight value, the three trees embedded in $\mathcal{G}$ are optimal solutions of the Prim's MST algorithm (Step 5). Here, the third scenario has taken place (Step 5). Therefore, let us assume that the built T_2 has the continuous edges $\{2,4,1\}$ and $\{3,4,1\}$ corresponding to $D(2,4|1)$ and $D(3,4|1)$ in L_D, respectively. Accordingly, $D(2,3|1)$ in L_D cannot be represented in T_2, given that their possible corresponding edge $\{2,3,1\}$ does not belong to this tree. In T_3 (the last tree of the built R-vine graph), its single continuous edge $\{2,3,1,4\}$ represents $D(2,3|1,4)$ in L_D and, therefore, $D(2,4|1,3), D(3,4|1,3)$ in L_D cannot be represented in this tree.

Another two R-vine graphs (Figure 2.9-(b)) could be generated from the same starting polytree graph, in addition to the one shown in Figure 2.9-(a). In one R-vine graph, the edges of T_2 would be $\{2,3,1\}$ and $\{2,4,1\}$, and that of T_3 would be $\{3,4,1,2\}$ (all continuous), corresponding to non-separations $D(2,3|1)$, $D(2,4|1)$, $D(3,4|1,2)$ in L_D respectively. In the other R-vine graph, the edges of T_2 would be $\{2,3,1\}$ and $\{3,4,1\}$, and that of T_3 would be $\{2,4,1,3\}$ (all continuous), corresponding to non-separations $D(2,3|1)$, $D(3,4|1)$, $D(2,4|1,3)$ in L_D respectively.

Conclusions from Example 3 Three different R-vine graphs, representing the same set of relationships, can be generated from the same starting polytree graph. Algorithm 2.1 guarantees that all relationships they encode exist in the starting polytree graph. Consequently, these R-vine graphs are an I-map and a D-map of the dependence model associated with the starting polytree graph. Specifically, in the example described here, the pairwise non-separations $D(1,2), D(1,3), D(1,4), D(2,4|1), D(3,4|1), D(2,3|1,4)$ in L_D encoded in the built R-vine graph correspond to (in)dependence relationships of the dependence model associated with the starting polytree graph, namely $D(X_1, X_2)$, $D(X_1, X_3)$, $D(X_1, X_4)$, $D(X_2, X_4|X_1)$, $D(X_3, X_4|X_1)$, $D(X_2, X_3|X_1, X_4)$ in M.



(a) From the polytree graph to the R-vine graph.



(b) Another two R-vine graphs.

Figure 2.9: Illustration of Example 3 (Scenario 3). (a) From the polytree graph to the R-vine graph, using Algorithm 2.1: (From left to right, from top to bottom) Starting polytree graph; polytree skeleton; T_1 of the built R-vine graph encodes $D(1,2), D(1,3), D(1,4)$ in L_D; weighted graphs $\mathcal{G}$; T_2 encodes $D(2,4|1), D(3,4|1)$ in L_D, whereas T_3 encodes $D(2,3|1,4)$ in L_D respectively. (b) Another two R-vine graphs that could be built from the same starting polytree graph, in addition to the one shown in (a): (On the left) Edges of T_2 would be $\{2,3,1\}$ and $\{2,4,1\}$, and that of T_3 would be $\{3,4,1,2\}$ (all continuous), corresponding to non-separations $D(2,3|1)$, $D(2,4|1)$, $D(3,4|1,2)$ in L_D respectively. (On the right) Edges of T_2 would be $\{2,3,1\}$ and $\{3,4,1\}$, and that of T_3 would be $\{2,4,1,3\}$ (all continuous), corresponding to non-separations $D(2,3|1)$, $D(3,4|1)$, $D(2,4|1,3)$ in L_D respectively. What is different in these R-vine graphs with respect to the one shown in (a) is depicted in blue.

Algorithm 2.2 Procedure for building a polytree graph from an R-vine graph.

Input
R-vine graph with all edges in T_1 continuous.

Output
$G = (N, E)$ – Polytree graph.

1: Create L_I and L_D from the R-vine graph via R-separation.
  for Edges in T_j, $j \geq 2$, in the R-vine graph:
    if $\langle i, k, \boldsymbol{S} \rangle$:
1.1 Add $I(i, k \,|\boldsymbol{S})$ to L_I.
    else
1.2 Add $D(i, k \,|\boldsymbol{S})$ to L_D.
2: Build the skeleton of G as T_1 in the R-vine graph.
3: Direct the edges of the skeleton of G:
  for $I(i, k \,|\boldsymbol{S})_G$ in L_I:
    if there is at least one undirected edge in the path between i and k with nodes of $\boldsymbol{S}$ in between (without considering the edge direction):
3.1: Direct undirected edges in the path between i and k without creating head-to-head connections.
    end if
  end for
  for $D(i, k \,|\boldsymbol{S})_G$ in L_D:
3.2: Set a node of $\boldsymbol{S}$ in the path between i and k as a head-to-head if the direction of the edges related with previous separations are preserved. Edges involved in the head-to-head connection are marked as *fixed*.
  end for
  return G

2.5.2 From R-vine Graphs to Polytree Graphs

Algorithm 2.2 presents the pseudo-code of a method capable of building a polytree graph G (the target or the output graph) that encodes as many pairwise relationships as possible derived from an R-vine graph (the starting or the input graph). This algorithm is described in the following.

Step 1 Create two lists L_I and L_D, which contain all pairwise separation $I(i, k \,|\boldsymbol{S})$ and non-separation $D(i, k \,|\boldsymbol{S})$ relationships, respectively, derived from the R-vine graph via R-separation (Definition 19).

A for-loop runs through edges of the starting R-vine graph, from top to bottom in the hierarchy such that, separations associated with continuous edges are added to L_I (Step 1.1), and non-separations associated to continuous edges are added to L_D (Step 1.2).

It is worth noticing that, for building a polytree graph from an R-vine graph, it is required that all edges of T_1 are continuous, since a discontinuous edge implies a disconnection in the resulting graph and, hence, no longer singly connected (see Section 1.9.3).

Step 2 Build the skeleton of the target polytree graph as T_1 of the starting R-vine graph.

Step 3 Edge direction of the skeleton is carried out in two phases. First, at Step 3.1, the algorithm goes through separations $I(i,k|\boldsymbol{S})$ in L_I (from lower to higher order separations, regarding the size of the separating set), checking if undirected edges in the path between i and k (through nodes in $\boldsymbol{S}$) exist. If they do, undirected edges in that path are directed without creating head-to-head connections. To ensure such a requirement, edges already directed in that path can be redirected in the opposite direction, if necessary.

At Step 3.2, the algorithm goes through non-separations in L_D (from lower to higher order relationships) in order to orient the corresponding edges as follows: Edges belonging to the path between i and k with nodes of $\boldsymbol{S}$ in between (without considering the edge direction) are directed in such a way that it allows the creation of at least one head-to-head node of $\boldsymbol{S}$, only if the direction of the edges related with previous separations are preserved. Edges involved in such a head-to-head connection are marked as *fixed*, so that they cannot be redirected again.

When Step 3 is completed, all edges of the polytree skeleton become directed. As in R-vine graphs, each pair i and k appears once as a conditioned set (either as a separation or a non-separation), all skeleton paths have been traversed more than once for edge direction according to L_I and L_D.

In conclusion:

1. Building a polytree graph from an R-vine graph requires that edges of T_1 are continuous to ensure that the resulting graph is singly connected.

2. In general, multiple polytree graphs can be built from the same starting R-vine graph. This is because different polytree connections can represent the same separation set (see Section 1.9.3 for different connections in polytree graphs).

3. All separations encoded in the initial R-vine graph can be incorporated in the built polytree graph because of the following facts: Separations in the starting R-vine graph are a subset of those in the built polytree graph. Moreover, separations in L_I are inserted before non-separations in L_D, and conflicts can be resolved using the edge redirection strategy. In general, not all non-separations existing in the starting R-vine graph can be represented in the built polytree graph, and it can encode relationships not represented in the starting R-vine graph.

4. Consequently, the built polytree graph using Algorithm 2.2 is neither an I-map nor a D-map of the dependence model associated with the starting R-vine graph.

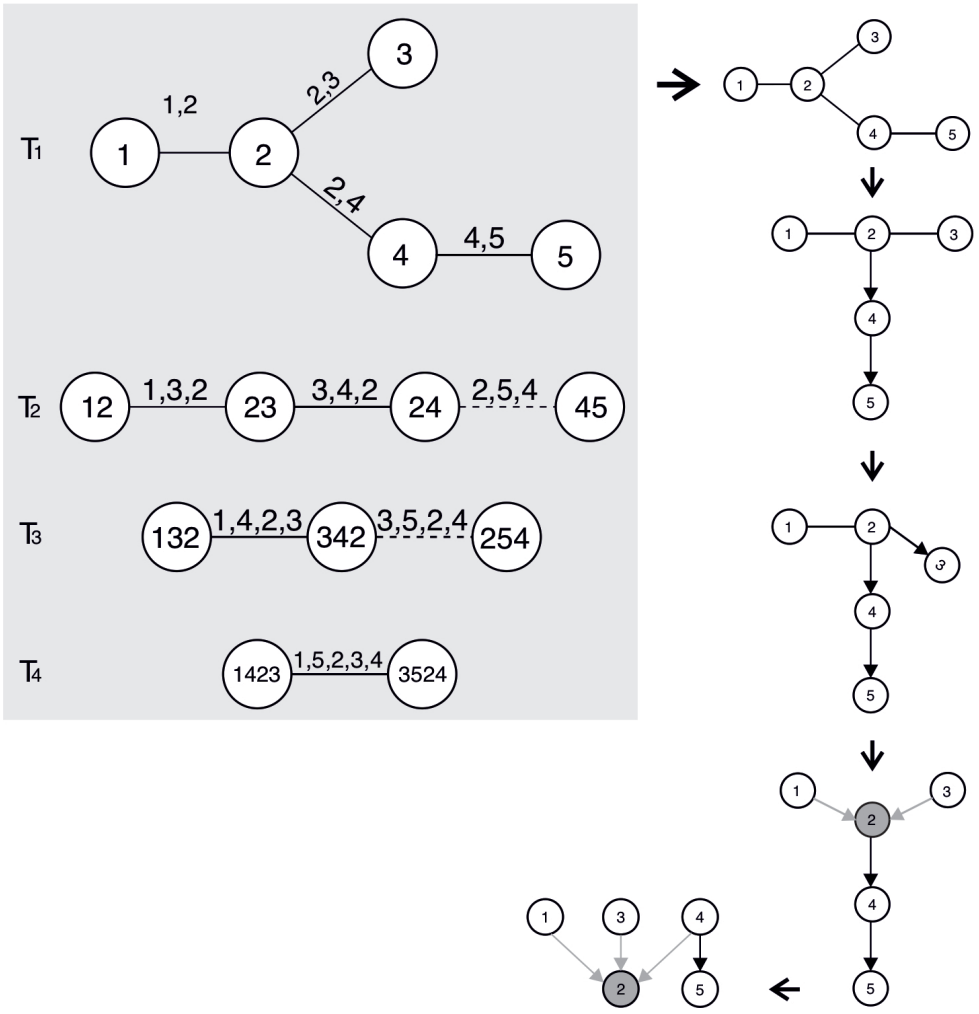
These conclusions are illustrated in Examples 4-6 using Figures 2.10-2.12 respectively. In these figures, the starting R-vine graph is framed in the gray rectangle, whereas head-to-head connections in polytree graphs appear in gray as well.

Example 4 From the R-vine graph to the polytree graph (Algorithm 2.2 and Figure 2.10-(a)).

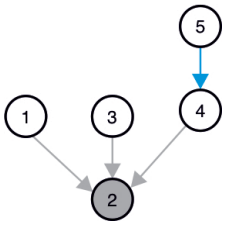
L_1 and L_2 derived via R-separation (Step 1) from the five-dimensional R-vine graph in Figure 2.10-(a) are given by

$$L_I = [I(2,5|4), I(3,5|2,4)].$$

$$\begin{aligned} L_D = [&D(1,2), D(2,3), D(2,4), D(4,5),\\ &D(1,3|2), D(3,4|2),\\ &D(1,4|2,3),\\ &D(1,5|2,3,4)]. \end{aligned}$$



(a) From the R-vine graph to the polytree graph.



(b) Another polytree graph.

Figure 2.10: Illustration of Example 7. (a) From the R-vine graph to the polytree graph, using Algorithm 2.2: (From left to right, from top to bottom) Starting R-vine graph; skeleton of the built polytree graph as T_1; edge direction for representing $I(2,5\,|4)$, $I(3,5\,|2,4)$ in L_I and $D(1,3\,|2)$, $D(3,4\,|2)$, $D(1,4\,|2,3)$ in L_D in the built polytree graph step by step. $D(1,5\,|2,3,4)$ in L_D cannot be represented in it, and $I(1,5\,|2,3,4)$ not in L_I is encoded instead. (b) Another polytree graph that could be built from the same starting R-vine graph, in addition to the one shown in (a), obtained by changing the fork connection $2 \leftarrow 4 \rightarrow 5$ to the chain connection $2 \leftarrow 4 \leftarrow 5$. What is different in this polytree graph with respect to the one shown in (a) is depicted in blue.

The first tree of the R-vine graph turns into the skeleton of the polytree graph (Step 2). Therefore, undirected edges associated to $D(1,2), D(2,3), D(2,4), D(4,5)$ in L_D are inserted in the target polytree graph, whose skeleton has the undirected edges $1-2$, $2-3$, $2-4$ and $4-5$.

Separations $I(2,5\,|4), I(3,5\,|2,4)$ in L_I are represented in the target polytree graph as $2 \rightarrow 4 \rightarrow 5$ and $3 \rightarrow 2 \rightarrow 4 \rightarrow 5$ respectively (Step 3.1). In order to represent $D(1,3\mid 2)$ in L_D, node 2 is turned into a head-to-head node in the connection $1 \rightarrow 2 \leftarrow 3$ by reversing $2 \rightarrow 3$ as $2 \leftarrow 3$. Analogously, for representing $D(3,4\,|2)$ in L_D, node 2 is turned into a head-to-head node in the connection $3 \rightarrow 2 \leftarrow 4$ by reversing $2 \rightarrow 4$ as $2 \leftarrow 4$. Likewise, for representing $D(1,4\,|2,3)$ in L_D, again, 2 becomes a head-to-head in the connection $1 \rightarrow 2 \leftarrow 4$, with no need to direct its two edges, since they were directed and marked as fixed by the two separations set before (Step 3.2). Notice that 3 is not on the path (undirected) path between 1 and 4. Finally, $D(1,5\,|2,3,4)$ in L_D cannot be represented in the built polytree graph, while the opposite $I(1,5\,|2,3,4)$ not in L_I is represented instead. Having completed the edge direction phase, the resulting polytree graph specifies separations and non-separations which do not exist in the starting R-vine graph, for instance, $I(1,5)$, $I(2,5)$, $I(3,5)$, $I(1,5\,|4)$, $I(3,5\,|4)$, $I(1,5\,|2,4)$ not in L_I and $D(1,5\,|2)$, $I(3,5\,|2)$ not in L_D respectively.

Another polytree graph could be built from the same starting R-vine graph, in addition to the one shown in Figure 2.10-(a), obtained by changing the fork connection $2 \leftarrow 4 \rightarrow 5$ to the chain connection $2 \leftarrow 4 \leftarrow 5$, as shown in Figure 2.10-(b).

Conclusions from Example 4 Two different polytree graphs, representing the same set of relationships, can be generated from the same starting R-vine graph with Algorithm 2.2. This is related with the edge direction, i.e., although the fork and chain connections are graphically different, they represent the same set of separations. Moreover, all separations derived from the starting R-vine graph can be specified in the built polytree graph, but the opposite cannot be specified. In addition, since the built polytree graph encodes separations and non-separations that do not exist in the starting R-vine graph, while leaving out a non-separation that does exist in the starting R-vine graph, the built polytree graph is neither an I-map nor a D-map of the dependence model associated with the starting R-vine graph.

Example 5 From the R-vine graph to the polytree graph (Algorithm 2.2 and Figure 2.11).

In this example, a distinctive feature of the starting R-vine graph is that all edges are continuous, representing only non-separations. The derived L_1 and L_2 via R-separation (Step 1) from the five-dimensional R-vine graph (indeed, a C-vine graph) in Figure 2.11 are given by

$$
\begin{aligned}
L_I &= \emptyset. \\
L_D &= [D(1,2), D(1,3), D(1,4), D(1,5), \\
&\quad D(2,3\,|1), D(2,4\,|1), D(2,5\,|1), \\
&\quad D(3,4\,|1,2), D(3,5\,|1,2), \\
&\quad D(4,5\,|1,2,3)].
\end{aligned}
$$

The first tree of the R-vine graph turns into the skeleton of the polytree graph (Step 2). Therefore, undirected edges associated to $D(1,2), D(2,3), D(2,4), D(2,5)$ in L_D are inserted in the target polytree graph, whose skeleton has the undirected edges $1-2$, $1-3$, $1-4$ and $1-5$.

In order to represent the first order relationships, $D(2,3\,|1)$, $D(2,4\,|1)$, $D(2,5\,|1)$ in L_D in the target polytree graph, the corresponding head-to-head connections $2 \rightarrow 1 \leftarrow 3$, $2 \rightarrow 1 \leftarrow 4$ and $2 \rightarrow 1 \leftarrow 5$ are created, where 1 is head-to-head node. In this way, the built polytree graph has all its edges directed, and the separations $D(3,4\,|1,2), D(3,5\,|1,2), D(4,5\,|1,2,3)$ in L_D encoded in it (Step 3.2). This is because they share the same head-to-head node as well as directed edges in head-to-head connections created before. Having completed the edge direction phase, the built polytree

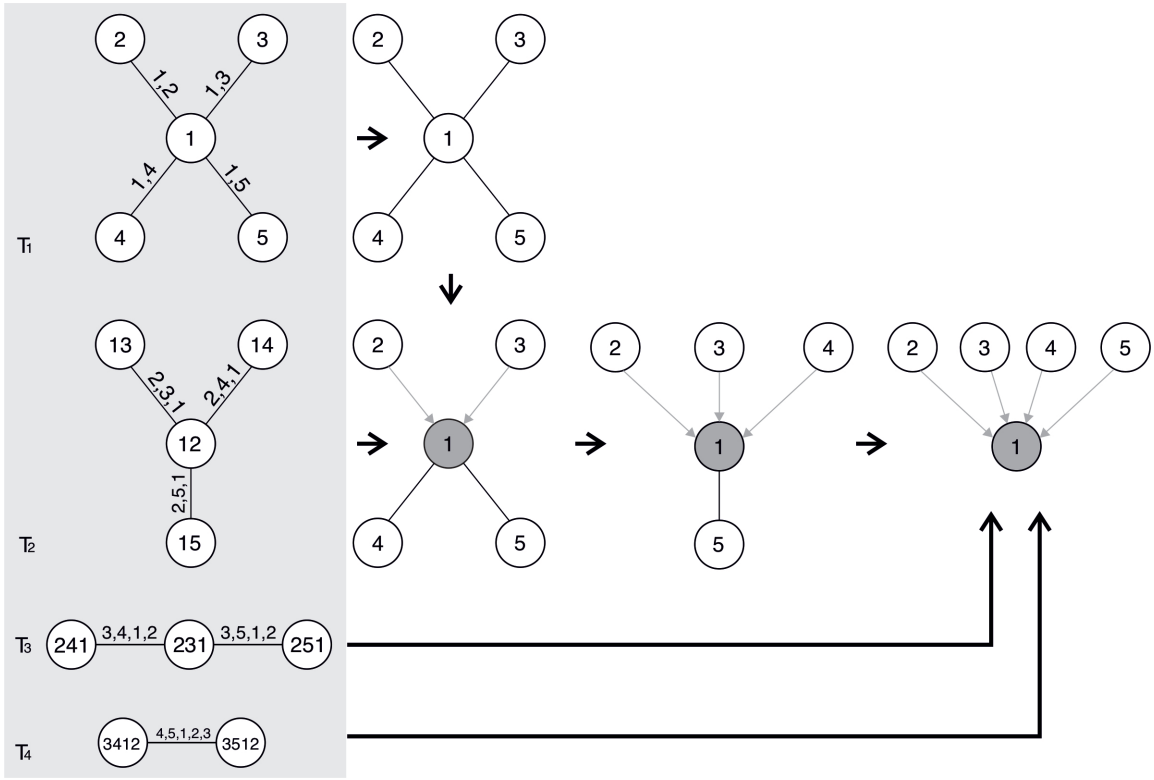


Figure 2.11: Illustration of Example 5. From the R-vine graph to the polytree graph, using Algorithm 2.2: (From left to right, from top to bottom) Starting R-vine graph; skeleton of the built polytree graph as T_1; edge direction for representing $D(2,3\,|1)$, $D(2,4\,|1)$, $D(2,5\,|1)$, $D(3,4\,|1,2)$, $D(3,5\,|1,2)$, $D(4,5\,|1,2,3)$ in L_D in the built polytree graph step by step.

graph is able to encode all non-separations derived from the starting R-vine graph (Figure 2.11), but also others that are not represented in the starting R-vine graph, for instance, $D(3,4\,|1)$, $D(3,5\,|1)$, $D(2,3\,|1,4,5)$ not in L_D.

Conclusions from Example 5 When the starting R-vine graph has a C-vine structure with all edges continuous, i.e., all relationships are of non-separation, all of them can be codified in the built polytree graph. This is because in a C-vine graph, the index that acts as the root node in T_1 belongs to the separating set of all non-separations, becoming the single head-to-head node in the built polytree graph. This particularity prevents the emergence of conflicts when orienting edges, as all can be directed towards the same head-to-head node. However, since the built polytree graph encodes other separations and non-separations which do not exist in the starting R-vine graph, the built polytree graph is neither an I-map nor a D-map of the dependence model associated with the starting R-vine graph.

Example 6 From the R-vine graph to the polytree graph (Algorithm 2.2 and Figure 2.12-(a)).

A distinctive feature of the starting R-vine graph is that it is truncated at T_1. This implies that in T_2, T_3 and T_4 all edges are dashed, representing only separations. L_1 and L_2 derived via R-separation (Step 1) from the five-dimensional R-vine graph in Figure 2.12-(a) are given by

$$\begin{aligned} L_I \;=\; & [I(2,3\,|1), I(2,4\,|1), I(2,5\,|1), \\ & I(3,4\,|1,2), I(3,5\,|1,2), \\ & I(4,5\,|1,2,3)]. \end{aligned}$$

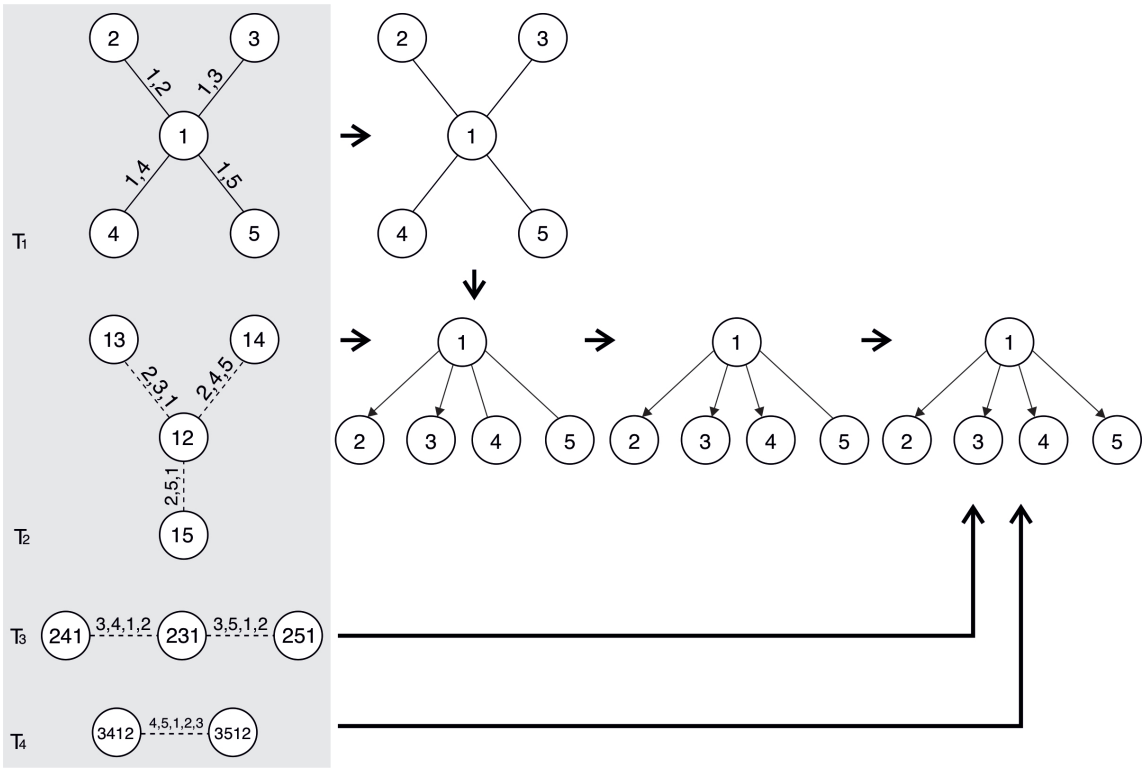
$L_D = [D(1,2), D(1,3), D(1,4), D(1,5)]$.

The first tree of the R-vine graph turns into the skeleton of the polytree graph (Step 2). Therefore, undirected edges associated to $D(1,2), D(1,3), D(1,4), D(1,5)$ in L_D are inserted in the target polytree graph, whose skeleton has the undirected edges $1-2$, $1-3$, $1-4$ and $1-5$.

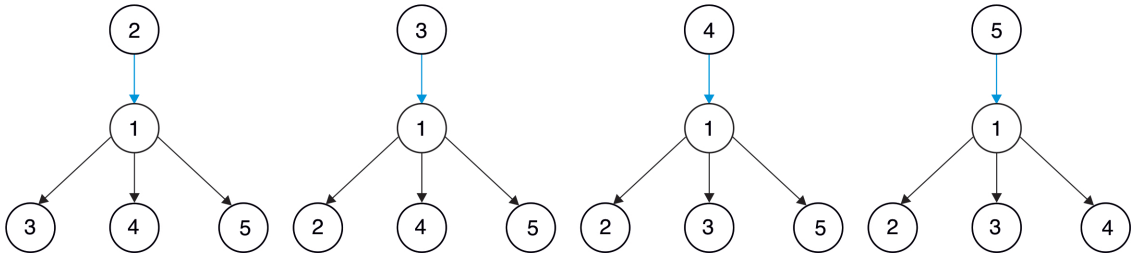
First, the separations $I(2,3\,|1), I(2,4\,|1), I(2,5\,|1)$ in L_I from T_2 are represented in the target polytree graph through the fork connections $2 \leftarrow 1 \rightarrow 3$, $2 \leftarrow 1 \rightarrow 4$ and $2 \leftarrow 1 \rightarrow 5$ respectively. The other three separations $I(3,4\,|1,2), I(3,5\,|1,2), I(4,5\,|1,2,3)$ in L_I, are encoded in the target polytree graph through three fork connections, $3 \leftarrow 1 \rightarrow 4$, $3 \leftarrow 1 \rightarrow 5$ and $4 \leftarrow 1 \rightarrow 5$ respectively, with no need to direct edges involved in them, because they are already directed (Step 3.1). When all separations of L_I have been inserted, the edge direction phase ends, since non-separations, coming only from T_1, are encoded by edges joining adjacent nodes. Having completed the edge direction phase, the polytree graph encodes all non-separations derived from the starting R-vine graph and no others, which is evidenced by the absence of head-to-head connections. On the other hand, the built polytree graph is able to encode all separations derived from the starting R-vine graph, but also others which do not exist in it, for instance, $I(2,3)$, $I(3,4)$, $I(4,5)$, $I(3,4\,|1), I(4,5\,|1)$ not in L_I.

Other four polytree graphs can be built from the same starting R-vine graph, in addition to the one shown in Figure 2.12-(a), obtained by interchanging between fork and chain connections, as shown in Figure 2.12-(b).

Conclusions from Example 6 Five different polytree graphs, representing the same set of relationships, can be generated from the same initial R-vine graph with Algorithm 2.2. Moreover, when the starting R-vine graph is truncated at T_1, all non-separations and separations derived from the starting R-vine graph are represented in the built polytree graph: Unlike Examples 4 and 5, here all non-separations come only from T_1, so that each edge of the built polytree graph represents one of them. On the other hand, all non-separations encoded in the remaining trees of the starting R-vine graph are encoded in the built polytree graph by inserting fork and chain connections, although others not in the starting R-vine graph also emerge. Consequently, since all non-separations encoded in the built polytree graph exist



(a) From the R-vine graph to the polytree graph.



(b) Another four polytree graphs.

Figure 2.12: Illustration of Example 6. (a) From the R-vine graph to the polytree graph, using Algorithm 2.2: (From left to right, from top to bottom) Starting R-vine graph; skeleton of the built polytree graph as T_1; edge direction for representing $I(3,4|1,2)$, $I(3,5|1,2)$, $I(4,5|1,2,3)$ in L_I in the built polytree graph step by step. (b) Another four polytree graphs that can be built from the same starting R-vine graph, in addition to the one shown in (a), obtained by interchanging between fork and chain connections. What is different in these polytree graphs with respect to the one shown in (a) is depicted in blue.

in the starting R-vine graph, the built polytree graph is a D-map of the dependence model associated with the starting R-vine graph. Moreover, since the built polytree graph encodes other separations which do not exist in the starting R-vine graph, the built polytree graph is not an I-map of the dependence model associated with the starting R-vine graph.

2.6 Summary

In this chapter, a graphical separation criterion for R-vines, called R-separation, has been defined. The proposed criterion facilitates the enumeration of (non-)separation relationships encoded in the R-vine graph with enhanced expressiveness by examining its topology and the edge types. The derived graphical relationships correspond to (un)conditional pairwise (in)dependence relationships in the associated R-vine copula. Moreover, from the R-separation criterion, a theorem of R-vine dependence maps is enunciated and it has been proved that every R-vine graph is an I-map and a D-map of the associated R-vine copula, but not necessarily a P-map, since from the R-vine graph, it is not possible to infer (non)separations other than those represented by its edges. Summarizing, R-vines do not allow to define a graphical separation concept that yields a complete independence map. We further analyzed different R-separation properties. Findings on this concept include the following: (i) It satisfies symmetry; (ii) it does not satisfy strong transitivity, weak transitivity nor strong union; (iii) weak union, decomposition, contraction and intersection cannot be verified, since R-vine graphs represent pairwise relationships only, and these properties involve sets of indices as the conditioned set.

Furthermore, the relationship between graphical representations of R-vines and polytrees has been analyzed. The focus has been on pairwise separations and non-separations encoded in one graph that correspond with the set of (un)conditional pairwise (in)dependencies of the dependence model associated with the other graph. For this purpose, two algorithms have been designed: One algorithm that aims to induce the R-vine graph that encodes as many pairwise relationships as possible derived from the polytree graph. The other algorithm achieves the same goal but in reverse, from the R-vine graph to the polytree graph. The former algorithm is capable of building an R-vine graph that encodes all separation and non-separation relationships derived from the starting polytree graph. Therefore, the R-vine graph is both an I-map and a D-map of the dependence model associated with the starting polytree graph. The other algorithm can produce the polytree graph that represents all separations derived from the starting R-vine graph, but not all the non-separations. In addition, graphical properties that favor the generation of multiple polytree graphs, representing the same set of separations and non-separations, have been identified. However, since the built polytree graph can represent additional non-coded relationships in the starting R-vine graph, the built polytree is neither an I-map nor a D-map of the dependence model associated with the starting R-vine graph.

Chapter 3

Learning the Graph Structure of Regular Vine-Copulas from Dependence Lists

3.1 Introduction

Typically, greedy heuristics for learning the graph structure of R-vines from data aim to maximize the strengths of the dependencies captured by the first trees using pairwise dependence measures computed from the dataset [1, 36, 42, 82]. We adopt a different approach by designing optimization strategies for building R-vine graphs from dependence lists, as opposed to computing them straightforward from the dataset. In the proposed approach, only the graph structure is built and, therefore, the designed strategies do not require the simultaneous estimation of pair-copulas, which is usual in other heuristics [1, 42].

Dependence lists are a way for building probabilistic models [27]. In some applications, these lists can be provided directly by experts in the domain of the problem being addressed [9, 27, 83]. However, it should be taken into account that the proposed method could be used to learn the graph structure of R-vines from a dependence list that is also generated from data, or using the joint factorization associated to a Bayesian network as in [83], or even derived from other GMs by applying the appropriate separation concept. Such is the case presented in Chapter 2, where the concepts of R-separation (as formulated in this thesis) and D-separation are used to extract the set of pairwise relationships encoded in R-vine graphs and polytree graphs respectively.

In the proposed approach, these lists can contain both dependence and independence relationships. However, R-vines, unlike other graphical models (e.g., Bayesian and Markov networks), can only represent explicit (un)conditional pairwise (in)dependencies. In other words, other possible (in)dependence relationships, not explicitly represented in the graph, cannot be derived from it. Therefore, and without loss of generality, we assume that the dependence list used for learning the graph structure of R-vines only contains (un)conditional pairwise dependence relationships $D\left(X_i, X_k \,|\boldsymbol{X}_{\boldsymbol{S}}\right)$, since the goal of the proposed strategies is to maximize the number of edges $\{i, k, \boldsymbol{S}\}$ corresponding to relationships in the dependence list.

Specifically, the research question we deal with is an optimization problem which aims to build the R-vine graphs that incorporate the largest number of dependence relationships given in a dependence list. This task faces several challenges. One of them concerns the (potentially) large number of possible R-vine graphs, which grows exponentially with the number of variables. Indeed,

the search space is so huge ($\binom{n}{2} \cdot (n-2)! \cdot 2^{\binom{n-2}{2}}$ for n dimensions [91]) that the brute-force search is only feasible for a few number of variables. The graphical constraints of the R-vine models as well as possible incompatibilities among the relationships belonging to the dependence list present a further challenge for the search strategies that should ensure that the generated solutions are feasible.

Two approaches are proposed for solving the optimization problem posed. The first approach is a 0-1 linear programming formulation that builds truncated R-vines with only two trees. The second approach is a genetic algorithm (GA) [71] that is able to learn complete and truncated R-vine graphs. The designed GA uses crossover and mutation operators specifically designed to ensure that the resulting solutions are feasible. Furthermore, extensive numerical experiments are carried out to assess the effectiveness of the designed evolutionary algorithm in solving the optimization problem posed.

This chapter is organized as follows. Sections 3.2 and 3.3 introduce the 0-1 linear programming and GA-based approaches. Section 3.4 describes the experimental framework and discusses numerical results in the context of the second approach, and Section 3.5 provides the conclusions of this chapter.

3.2 Linear Programming Approach

In this section, we present a 0-1 linear mathematical approach to address the problem of learning R-vine graphs that represent the largest number of dependence relationships in a dependence list. This approach works for R-vines graphs with two levels, therefore, the dependence list, denoted by L, is comprised of unconditional and order-one conditional dependence relationships of the form $D(X_i, X_k)$ and $D(X_i, X_k \mid X_s)$ respectively. Notice that in the expressions of this section, we use s instead of $\boldsymbol{S}$, since s represents a single index in all of them.

Let each possible edge in T_1 be associated to one binary variable $y_{i,k}$ and each possible edge in T_2 be associated to one binary variable $z_{i,k,s}$, where $i \neq s$, $k \neq s$, $i < k$, $i = 1, \ldots, n-1$, $k = i+1, \ldots, n$, and $s = 1, \ldots, n$. The meaning of the variables is as follows:

$$\begin{aligned} y_{i,k} &= \begin{cases} 1 & \text{if the edge } \{i,k\} \text{ is in the tree } T_1 \\ 0 & \text{otherwise} \end{cases} \\ z_{i,k,s} &= \begin{cases} 1 & \text{if the edge } \{i,k,s\} \text{ is in the tree } T_2 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.1}$$

For the dependencies in L, we assume that $\forall i,k, i<k$, since that $D(x_i, x_k) = D(x_k, x_i)$ as well as $D(x_i, x_k \,|x_s) = D(x_k, x_i \,|x_s)$.

A formal definition of the linear programming problem posed is as follows:

$$max \sum_{i,k|D(x_i,x_k)\in L} y_{i,k} + \sum_{i,k,s|D(x_i,x_k|x_s)\in L} z_{i,k,s} \tag{3.2}$$

subject to

$$y_{i,k} + \sum_{s=1, i\neq s, k\neq s}^{n} z_{i,k,s} \leq 1, \forall i,k, i<k \tag{3.3}$$

$$\sum_{i=1}^{n-1}\sum_{k=i+1}^{n} y_{i,k} = n-1 \tag{3.4}$$

$$\sum_{i,k\in M} y_{i,k} \leq |M|-1, \forall M \subseteq \{1,\ldots,n\}, i<k \tag{3.5}$$

$$\sum_{i=1}^{n-2}\sum_{k=i+1}^{n-1}\sum_{s=1}^{n} z_{i,k,s} = n-2 \tag{3.6}$$

$$\sum_{\{i,s\},\{k,s\}\in M_1, i<k} z_{i,k,s} \leq |M_1|-1, \tag{3.7}$$

$$\forall M_1 \subseteq \{\{a,b\} \mid a<b, a=1,\ldots,n-1, b=a+1,\ldots,n\}$$

$$2\cdot z_{i,k,s} \leq (y_{s,i}+y_{s,k}), \forall i,k,s, i\neq s, k\neq s, i<k \tag{3.8}$$

$$y_{i,k} \in \{0,1\}, \forall i<k \tag{3.9}$$

$$z_{i,k,s} \in \{0,1\}, \forall i\neq s, k\neq s, i<k \tag{3.10}$$

Note that $z_{i,k,s}=1$ occurs if the edge $\{i,k\}$ is not in the tree T_1 and there are two edges $\{i,s\}$ and $\{k,s\}$ in T_1. The triples (i,k,s) that satisfy $(y_{i,s}=1, y_{k,s}=1, z_{i,k,s}=1)$ serve to define the edges $\{i,k,s\}$ of T_2. This is so because these triples do not form a cycle in T_1 ($z_{i,k,s}=1$) and each triple groups two edges of T_1 that have a common node in T_1.

The objective function of the optimization problem (3.2) is subject to several constraints. (3.3) means that an edge can not be part of the tree T_1 and, at the same time, join two other edges in T_1. Moreover, (3.4), and (3.5) (together with (3.3)) guarantee that variables $y_{i,k}$ represent a tree T_1. Specifically, (3.4) means that the number of edges in T_1 is $n-1$. The inequality (3.5) enforces the resulting graph to not contain any cycles, which is, together with the enforced number of edges, also a sufficient condition for trees. On the other hand, (3.6) and (3.7) guarantee that the $z_{i,k,s}$ represent a tree T_2. Specifically, (3.6) means the number of edges in T_2 is $n-2$, and (3.7) ensures that the resulting graph does not contain any cycles. The fulfillment of the proximity condition relating T_1 and T_2 is guaranteed by (3.8), which reinforces the condition that whenever $\sum_{k=1}^{n} z_{i,k,s}=1$ then there exist two variables $y_{s,i}=1$ and $y_{s,k}=1$, which correspond to two edges with a common node in T_1. (3.9) and (3.10) mean this is a binary problem[1]. In the proposed formulation, the number of constraints of the linear program increases exponentially.

3.3 Evolutionary Approach

The evolutionary approach used for building R-vine graphs that incorporate the largest possible number of dependencies in L is based on GAs. Over the following sections, we present how R-vine graph solutions are encoded, the fitness function used to quantify the quality of each solution, and the genetic operators used to generate new solutions.

[1] More specifically, it is a 0-1 problem with linear restrictions and a linear objective function.

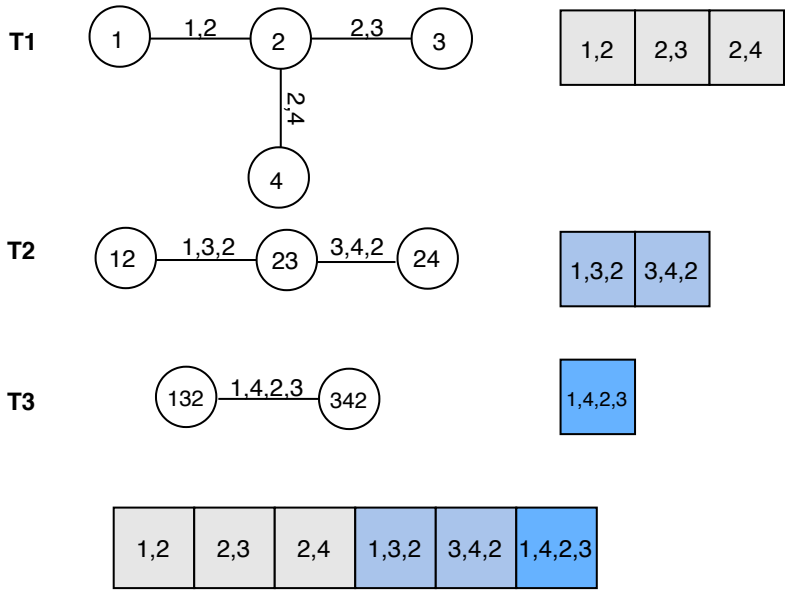


Figure 3.1: An R-vine graph solution is encoded by a list of edges, where each edge is represented by a tuple of integers, each one representing an index in I. For instance, the list of tuples associated to the R-vine graph solution of this picture, which has six edges, contains six tuples, namely $(1,2),(2,3),(2,4),(1,3,2),(3,4,2),(1,4,2,3)$.

3.3.1 Representation

In the designed GA, each solution represents an R-vine graph encoded as a list of edges, where each edge $\{i,k,\boldsymbol{S}\}$ is represented by a tuple of integers, each representing an index in I. The list of tuples consists of $n(n-1)/2$ tuples, where the first $n-1$ tuples correspond to the edges of T_1; the following $n-2$, to the edges of T_2, and so on, until the last tuple representing the one edge of T_{n-1}. Figure 3.1 shows a three-dimensional R-vine graph represented by a list of six tuples.

3.3.2 Fitness Function

Let $y_{i,k,\boldsymbol{S}}$ be a binary variable associated to an edge $\{i,k,\boldsymbol{S}\}$. The fitness function of the solution R is written by

$$f(R) = \sum_{i,k,\boldsymbol{S}|D(x_i,x_k|x_{\boldsymbol{S}})\in L} y_{i,k,\boldsymbol{S}}, \tag{3.11}$$

where

$$y_{i,k,\boldsymbol{S}} = \begin{cases} 1 & \text{if } \{i,k,\boldsymbol{S}\}\in R \\ 0 & \text{otherwise} \end{cases}$$

This is a maximization problem, as the goal of the search is to find the solutions that represent the largest number of dependence relationships given in L. Notice that the fitness function goes over the dependencies in L to count how many of them are captured by edges in R.

Algorithm 3.1 Procedure for building an initial R-vine graph solution from L in the initial population. This procedure uses the following edge-prioritization-strategy: high weights are assigned to possible edges associated with dependencies in L, and low weights to other possible edges. The term *possible edges* refers to the edges of the graph that meet the proximity condition.

Inputs
n - Number of indexes.
L - A dependence list.

Outputs
R - An initial R-vine graph solution.

for $j = 1 : n - 1$:
  if $j = 1$:
1: Build a complete graph of n nodes.
  else
2: Build a graph of $n - j + 1$ nodes that contains
    all *possible edges* from T_{j-1}.
  end if
3: Assign high weights to the edges of the graph associated with L
    and low weights to the rest of the edges.
4: Built a MST T_j from the weighted graph.
end for
return $R = (T_1, \ldots, T_{n-1})$

3.3.3 Initialization

In order to generate the initial population, R-vine graph solutions encoded as shown in Figure 3.1 have to be generated. The procedure for generating an initial R-vine graph solution is inspired by the Top-Down Greedy Heuristic (TDGH) for learning R-vines from data introduced in [42] (see Algorithm 1.1). This heuristic relies on individually optimizing the tree at each level, while trying to recover the strongest dependencies in the dataset. Departing from a complete graph, where the weights of the edges represent empirical pairwise dependence values (for instance, Kendall's tau or BIC values) the algorithm starts finding the maximum spanning tree (MST) of n nodes (which is the tree that maximizes the sum of empirical pairwise dependencies) using Prim's algorithm [105]. Subsequent trees are also built using the MST method. However, while in the first step the heuristic starts from a complete graph, in subsequent steps it departs from a graph that only contains edges that meet the proximity condition (not a complete graph in general). Such a heuristic requires that at each level the pair-copulas and their parameters are simultaneously estimated before moving on to the next tree.

Indeed, what is new in Algorithm 3.1 for generating an initial R-vine graph solution in comparison with the TDGH is threefold: (i) The R-vine graph is learned from a dependence list, not from data. (ii) It implements a strategy that assigns high weights to those possible edges associated with dependencies in L and low weights to the rest, instead of using as weights empirical pairwise dependence values. Such weights are randomly generated in $\mathbb{R}^+$. (iii) It does not require pair-copula fitting.

Algorithm 3.1 can be used to build truncated R-vine graph solutions by simply executing the *for* loop for a certain truncation level $1 \leq t \leq n - 1$.

3.3.4 Genetic Operators

We describe the selection, crossover and mutation operators designed to produce new R-vine graph solutions.

Selection

We used truncation selection where a percentage of the best solutions are selected to be parents in the next generation.

Crossover

Designing a crossover operator for R-vine graphs is a challenging task because, as previously noted, the hierarchical nature of these models implies that the structure of each tree depends on the previous one. Consequently, a modification in one tree must be propagated throughout the hierarchy, in such a way that the resulting structure is an R-vine graph.

Taking into account this peculiarity, the designed crossover method is accomplished in several steps, graphically illustrated in Figure 3.2, namely choosing, recombining, and rebuilding.

The crossover operator uses a single parameter, denoted by $d < n - 1$, which stands for the number of edges of the maximum weight subtrees extracted from each individual parent solution.

We illustrate this process for one of the two new solutions generated by the crossover method in Figure 3.2. The illustration for the second solution is analogous. In this figure, edges associated to dependencies in L are represented by solid lines; otherwise, they are represented by dashed lines. Furthermore, we set the number of variables to $n = 4$, the crossover parameter to $d = 2$, and assume the possible edges associated with dependencies in L are given by

$$\underbrace{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{3,4\}}_{T_1},\underbrace{\{2,3,1\}}_{T_2},\underbrace{\{2,4,1,3\}}_{T_3}$$

Choosing The crossover method starts by assigning weight values to the edges of the first tree T_1 of the individual solution R^A. These values are randomly generated in $\mathbb{R}^+$ and assigned in the following way: Weight values assigned to edges associated to dependencies in L are larger than those assigned to the rest of the edges. Then, the MST method chooses the subtree of maximum weight of d edges, $SubT$, from T_1 of R^A.

In Figure 3.2-(Choosing), we use $d = 2$ and assume that the edges of $SubT$ are $\{1,2\},\{1,3\}$. The nodes of $SubT$ are shaded.

Recombining The goal of this phase is to build the first tree of a new solution, R^{new}, by recombining structures from the first tree of two individual solutions R^A and R^B. We proceed as follows:

1. Consider a graph that contains the nodes and the edges that meet the following:

$$\begin{array}{l} \{i,k\} \text{ if } i \notin SubT \text{ and } k \notin SubT \\ \{i,k\} \text{ if } i \in SubT \text{ and } k \notin SubT \text{ or } i \notin SubT \text{ and } k \in SubT \\ \{i,k\} \text{ if } i \in SubT \text{ and } k \in SubT \text{ and } \{i,k\} \in SubT \end{array} \tag{3.12}$$

That is, (3.12) is a complete graph except for the sub-graph generated by the nodes in $SubT$, for which the tree structure coming from R^A is kept. The weight values of this graph are randomly generated in $\mathbb{R}^+$ and assigned in the following way:

- High weight values to the edges associated to dependencies in L that also belong to R^B. This ensures that the first tree of R^{new} inherits structures from both parent solutions, which also determine the structure of the subsequent trees of the new solution.
- Intermediate weight values to the edges associated to dependencies in L that are not in R^B.
- Low weight values to the rest of edges.

High weight values are higher than intermediate weight values, which in turn are higher than low weight values.

2. Departing from the graph with all weights assigned and fixed edges of $SubT$, the MST method finds the first tree T_1 of R^{new}.

In Figure 3.2-(Recombining), we assume that the edges of T_1 of R^{new} are $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, where the first two edges come from $SubT \in R^A$ and the last one, from R^B.

Rebuilding The subsequent $n-2$ trees of R_{new} are rebuilt sequentially similar to the previous constructions: At each level (except for the last one), a tree of maximum weight is built departing from a weighted graph that contains all possible edges using the MST method. Weight values of this graph are randomly generated in $\mathbb{R}^+$ and assigned in the following way: Weight values assigned to edges associated to dependencies in L are larger than those assigned to the rest of the edges.

In Figure 3.2-(Rebuilding), we assume that the two edges of T_2 of R^{new} are $\{2,3,1\}$, $\{2,4,1\}$ where only the first one is associated with a dependence in L. Notice that, the insertion of $\{2,4,1\}$ in T_2 prevents the edge $\{2,4,1,3\}$ (which is associated with a dependence in L) from being included in the last tree T_3 of R^{new}. The only edge of T_3 is $\{3,4,1,2\}$.

Mutation

The mutation operator uses a single parameter, h, which indicates that trees at levels lower than h, i.e., $T_1, \ldots, T_{h-1}$, are not modified; while trees that are at levels higher or equal to h, i.e., $T_h, \ldots, T_{n-1}$, are rebuilt. At each level (except for the last one), the MST method departs from a weighted graph that contains all the edges that meet the proximity condition (not complete in general). The weights of this graph are randomly generated in $\mathbb{R}^+$ and assigned in the following way: Weight values assigned to edges associated to dependencies in L are larger than those assigned to the rest of the edges. It is worth noticing that, for the mutation to have a chance of impacting the population diversity, the value of h must be less than or equal to the number of variables in the conditioning set of the highest-order dependencies in L.

Figure 3.3 illustrates the mutation operator, for $h = 2$, applied on R^{new}, the solution previously generated by the crossover method. The first tree of the mutated solution, $R^{mutated}$, is the same as the first tree of R^{new}, while the subsequent two trees are rebuilt. The edges of T_2 are $\{2,3,1\}$, $\{3,4,1\}$, where the first one is associated with a dependence in L. The only possible edge of T_3 is $\{2,4,1,3\}$, which is in turn associated with a dependence in L.

3.4 Experiments

We conduct numerical experiments to analyze the effectiveness of the evolutionary approach in solving the optimization problem posed. This section is divided in two parts: we first outline the experimental framework and then present and discuss the obtained numerical results.

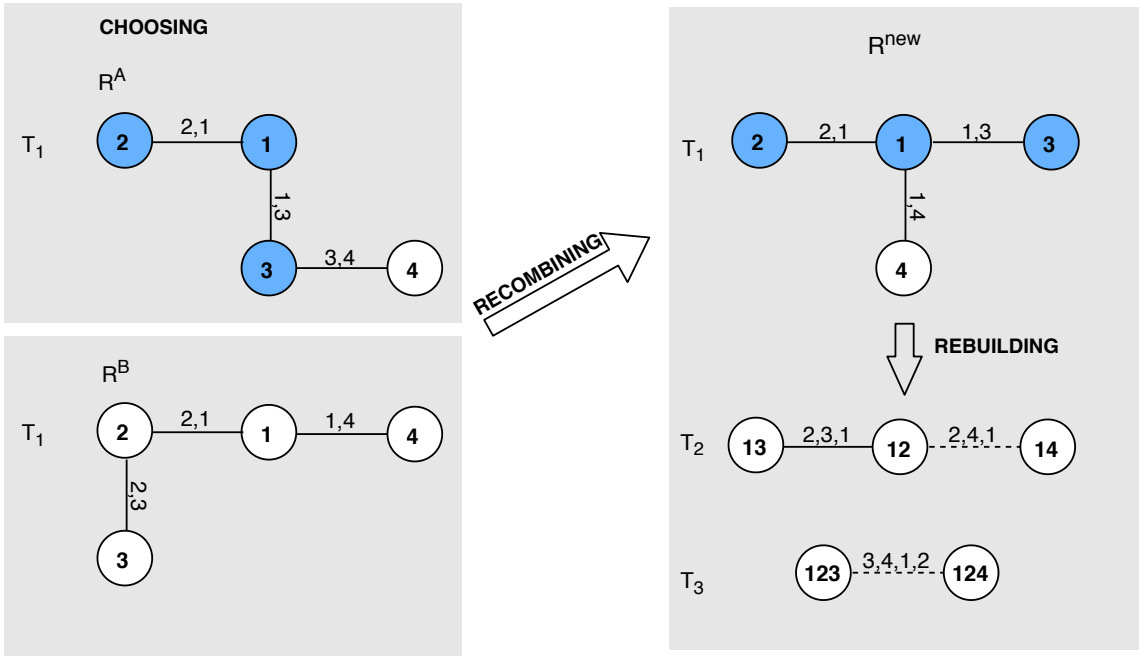


Figure 3.2: Illustration of the crossover operator. Edges associated to dependencies in L are represented by solid lines; otherwise, they are represented by dashed lines. In the choosing phase, we use $d = 2$ and assume that the edges of $SubT$ are $\{1,2\}, \{1,3\}$. The nodes of $SubT$ are shaded. In the recombining phase, we assume that the edges of T_1 of R^{new} are $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, where the first two edges come from $SubT \in R^A$, and the last one from R^B. In the rebuilding phase, the two edges of T_2 of R^{new} are $\{2,3,1\}$, $\{2,4,1\}$ where only the first one is associated with a dependence in L. Notice that, the insertion of $\{2,4,1\}$ in T_2 prevents the edge $\{2,4,1,3\}$ (which is associated with a dependence in L) from being included in the last tree T_3 of R^{new}. The only edge of T_3 is $\{3,4,1,2\}$. The fitness of the new solution is $f(R^{new}) = 4$.

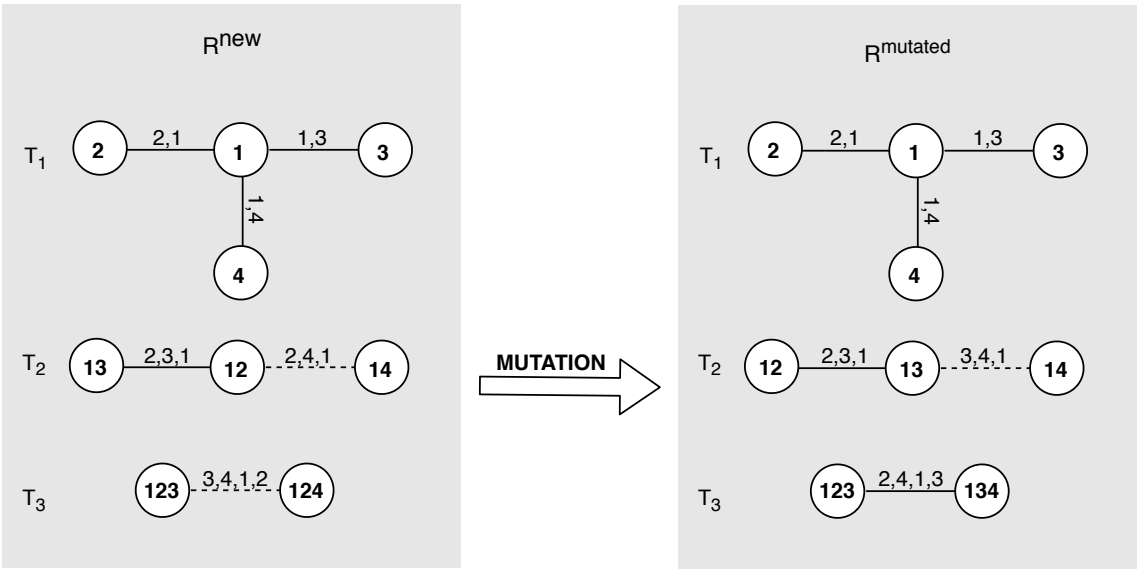


Figure 3.3: Illustration of the mutation operator, for $h = 2$, applied on R^{new}. Edges associated to dependencies in L are represented by solid lines; otherwise, they are represented by dashed lines. The first tree of the mutated solution, $R^{mutated}$, is the same as the first tree of R^{new}, while the subsequent two trees are rebuilt: The edges of T_2 are $\{2,3,1\}$, $\{3,4,1\}$, where the first one is associated with a dependence in L. The only possible edge of T_3 is $\{2,4,1,3\}$, which is in turn associated with a dependence in L. The fitness of the mutated solution is $f\left(R^{mutated}\right) = 5$. In this example, the solution generated by the mutation is better than the original solution, with fitness $f\left(R^{new}\right) = 4$.

3.4.1 Experimental Framework

Since there are no antecedents on the use of GAs in the optimization problem addressed here, a benchmark or baseline method (BM) is designed in order to evaluate the effectiveness of the designed GA compared to the BM. The BM is similar to the method that generates the solutions of the initial population in the designed GA, where no genetic operators are involved, hence it can be seen as a fairly basic procedure to solve the optimization problem posed.

In order to evaluate the algorithms, we propose two types of problems based on two types of dependence lists: feasible and unfeasible, according to whether the dependencies in L are or not compatible with an R-vine graph. To generate these lists, the following two steps are carried out:

Step 1 Build an R-vine graph: T_1 is obtained from a complete weighted graph over n nodes, where the weights of its edges are values randomly generated in $\mathbb{R}^+$. Departing from this graph, T_1 is found using the MST method. Subsequent trees $T_2, T_3, \ldots$ are also built using the MST method on graphs over $n-1, n-2, \ldots$ nodes, respectively, containing the edges that meet the proximity condition, which are labeled with weight values randomly generated in $\mathbb{R}^+$.

Step 2 Create a (feasible or unfeasible) dependence list using the R-vine graph built in the previous step as follows:

Feasible dependence list A subset of the dependence relationships represented by the edges of the R-vine graph is chosen and included in L. An optimal solution is the R-vine graph that encodes all the relationships contained in the feasible dependence list, and the optimal fitness value is the number of dependencies in L.

It is worth noticing that, if the feasible list contains the full set of dependence relationships that a complete R-vine graph can represent, then the optimal solution corresponds, precisely, to that graph, and the optimal fitness value is $n(n-1)/2$. In this case, the optimal solution can be recovered by the BM. However, it should be pointed out that in real-world applications (especially in high dimensions), the complete dependence list is not only unlikely to be available, but actually not necessary. In addition to the fact that for building simple models, such as truncated ones, it is enough that the dependence list contains a few relationships associated with the first trees in the R-vine hierarchy.

Unfeasible dependence list These lists can be generated in different ways. For instance, by randomly generating tuples, mixing lists from different R-vine graphs, or combining the two previous strategies. Therefore, the lists created in these ways can have more than $n(n-1)/2$ dependencies. When working with unfeasible lists, both the optimal solution and the optimal fitness value are unknown, which prevents us from accurately assessing how good the best generated solution is.

To alleviate this scenario, we propose a strategy that provides a lower bound for the optimal fitness value. It consists of joining in L two feasible lists created from two different original R-vine graphs. Both graphs must have the same dimension and the same number of trees. Each of these lists contains a percentage of dependencies represented as edges by the corresponding R-vine graph. The lower bound of the optimal fitness value is the number of relationships in the longer dependence list.

We set $n = 50, 100, 150$ variables. To ensure a fair comparison between the algorithms, parameter values for the genetic operators are selected as follows: The crossover parameter d, which indicates the size of the sub-graph extracted from the first tree of one parent solution, is set as a ratio

according to the size of that first tree. The crossover ratio is denoted by *dratio*. The mutation parameter h is set as a ratio of the $n-1$ trees that comprise the R-vine graph solutions. The mutation ratio is denoted by *hratio*. After preliminary experiments, we set $dratio = 30, 50, 70\%$ with respect to the number of edges of the first tree of the R-vine graph solution (a small, medium and large crossover ratio); and $hratio = 50, 70\%$ with respect to the number of trees of the R-vine graph solution. We use elitism, where the best two solutions of each generation are guaranteed a place in the next generation. Additional parameters for the GA are $maxGen = 100$ as the maximum number of generations; and $N = 100$ as the population size.

The optimization algorithm stops when it reaches at least one of the following stopping criteria: The optimum is found; the best fitness function value found over 10 successive generations remains constant; and the maximum number of evaluations is reached, which is given by $maxGen \cdot N$.

The number of independent runs of the GA for the same dependence list and parameter setting is 30. The metrics used to evaluate the performance of the GA are the average of the fitness value of the best solutions found in each run expressed as a percentage of the number of dependencies in L ($\%Fitness$); the average of the total number of function evaluations required to reach the best solution found in each run (numEvals); the number of times the optimum is found in all the runs ($numOpt$) when the dependence list is feasible; and the number of times the lower bound is either reached or exceeded in all the runs ($numBnd$) when the dependence list is unfeasible.

The BM is run 10000 times for the same dependence list. Notice that we have set the number of evaluations performed by the BM equal to the maximum number of evaluations that could be performed by the GA. The metric used to evaluate the effectiveness of the BM is the average of the fitness value of solutions built in each execution expressed as a percentage of the number of dependencies in L ($\%Fitness$). The metrics $numEvals$, $numOpt$, and $numBnd$ used to evaluate the effectiveness of the GA do not apply to the BM. Instead, we report how many times the BM builds the optimal R-vine graph solution.

3.4.2 Numerical Results

In this section, we focus on three main topics: First, we analyze the effectiveness of the proposed evolutionary approach through a comparison between the BM and the GA. In a second step, we explore possible scenarios that allow the GA to converge to sub-optimal solutions and, finally, we analyze the robustness of the GA.

3.4.2.1 Comparison Between GA and BM

We begin with a comparison between the GA and the BM. In the present experiments, the feasible dependence list contains 40% of all dependencies represented by the corresponding original complete R-vine graph. The unfeasible dependence list contains two feasible lists created with 40% of all dependencies represented by the corresponding original complete R-vine graph.

Tables 3.1 and 3.2 show the results obtained with feasible and unfeasible lists respectively. In all experiments, the GA converges to optimal or near-optimal solutions, while the BM stays further away from optimal solutions, which is more remarkable if the dependence list is unfeasible. Figure 3.4 illustrates this behavior through convergence curves of the GA for the feasible and unfeasible dependence lists. In both cases, the average $\%Fitness$ increases with each new generation for each value of n.

Table 3.1: Comparison between the BM and the GA using the feasible dependence list for each value of n. For the GA, three different values for the crossover parameter d corresponding to $dratio = 30, 50, 70\%$ respectively, and two values for the mutation parameter h corresponding to $hratio = 50, 70\%$ are used.

Feasible dependence list						
	BM	GA				
n	$\%Fitness$	d	h	$\%Fitness$	$numEvals$	$numOpt$
50	$50,18 \pm 4,15$	15	25	$96,83 \pm 2,90$	$7722 \pm 2,05$	25
			35	$95,92 \pm 2,97$	$7963 \pm 2,66$	24
		25	25	$98,12 \pm 2,37$	$6998 \pm 2,52$	27
			35	$98,31 \pm 2,11$	$6890 \pm 2,45$	27
		35	25	$96,60 \pm 2,76$	$7872 \pm 2,96$	25
			35	$96,42 \pm 2,88$	$7643 \pm 2,99$	26
100	$46,01 \pm 4,37$	30	50	$96,20 \pm 2,84$	$8224 \pm 3,47$	24
			70	$95,12 \pm 3,12$	$8270 \pm 3,48$	24
		50	50	$97,81 \pm 3,04$	$8101 \pm 3,22$	26
			70	$97,53 \pm 3,27$	8031 ± 2.15	25
		70	50	$94,70 \pm 3,33$	$8331 \pm 3,60$	23
			70	$95,32 \pm 3,65$	$8410 \pm 3,58$	24
150	$42,41 \pm 4,89$	45	75	$94,46 \pm 3,87$	$8865 \pm 4,11$	22
			105	$93,27 \pm 4,02$	$8891 \pm 4,22$	21
		75	75	$96,09 \pm 3,95$	$8677 \pm 3,91$	24
			105	$97,30 \pm 3,91$	$8654 \pm 4,00$	25
		105	75	$93,80 \pm 4,22$	$9022 \pm 4,33$	21
			105	$95,25 \pm 4,03$	$8947 \pm 4,14$	23

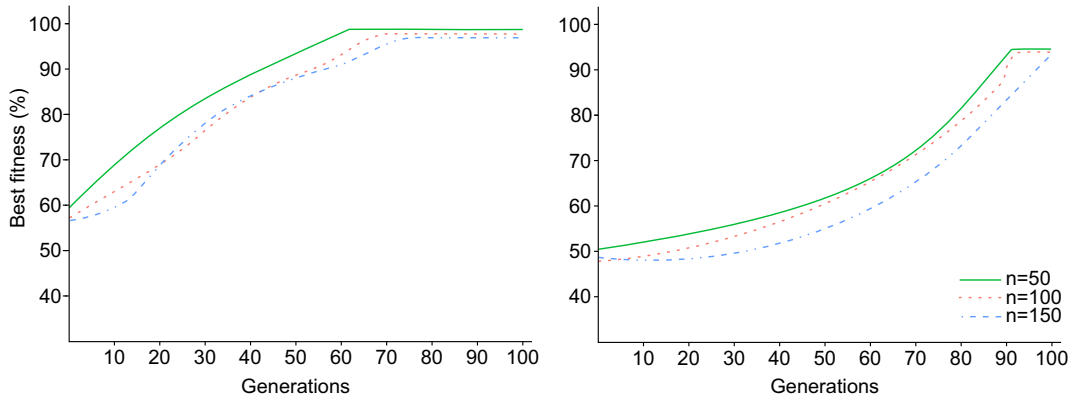


Figure 3.4: Convergence curves of average of the best fitness value expressed in percent in each generation using the feasible (left panel) as well as unfeasible dependence list (right panel) for each value of n, $dratio = 50\%$, and $hratio = 50\%$.

Table 3.2: Comparison between the BM and the GA using the unfeasible dependence list for each value of n. For the GA, three different values for the crossover parameter d corresponding to $dratio = 30, 50, 70\%$ respectively, and two values for the mutation parameter h corresponding to $hratio = 50, 70\%$ are used.

Unfeasible dependence list						
	BM	GA				
n	$\%Fitness$	d	h	$\%Fitness$	$numEvals$	$numBnd$
50	$38,78 \pm 4,59$	15	25	$94.37 \pm 3,75$	$8999 \pm 4,27$	24
			35	$94,94 \pm 3,14$	$8972 \pm 3,22$	24
		25	25	$96,03 \pm 4,26$	$8767 \pm 4,13$	26
			35	$95,13 \pm 4,32$	$8754 \pm 3,77$	25
		35	25	$95,61 \pm 4,16$	$8553 \pm 3,04$	24
			35	$94,23 \pm 4,11$	$8772 \pm 3,16$	25
100	$37,11 \pm 4,82$	30	50	$94,18 \pm 3,41$	$8990 \pm 4,44$	23
			70	$93,74 \pm 3,57$	$8866 \pm 4,06$	23
		50	50	$95,45 \pm 3,39$	$8964 \pm 4,07$	25
			70	$94,58 \pm 3,69$	$8943 \pm 4,15$	24
		70	50	$94,81 \pm 4,02$	$8876 \pm 3,89$	24
			70	$93,22 \pm 3,94$	$8893 \pm 4,00$	25
150	$35,32 \pm 4,94$	45	75	$93,22 \pm 4,10$	$9344 \pm 4,67$	22
			105	$93,16 \pm 4,23$	$9567 \pm 4,21$	21
		75	75	$94,01 \pm 3,80$	$9245 \pm 4,43$	23
			105	$93,76 \pm 3,87$	$9486 \pm 4,04$	22
		105	75	$93,99 \pm 3,94$	$9533 \pm 4,17$	23
			105	$92,04 \pm 4,15$	$9579 \pm 4,87$	21

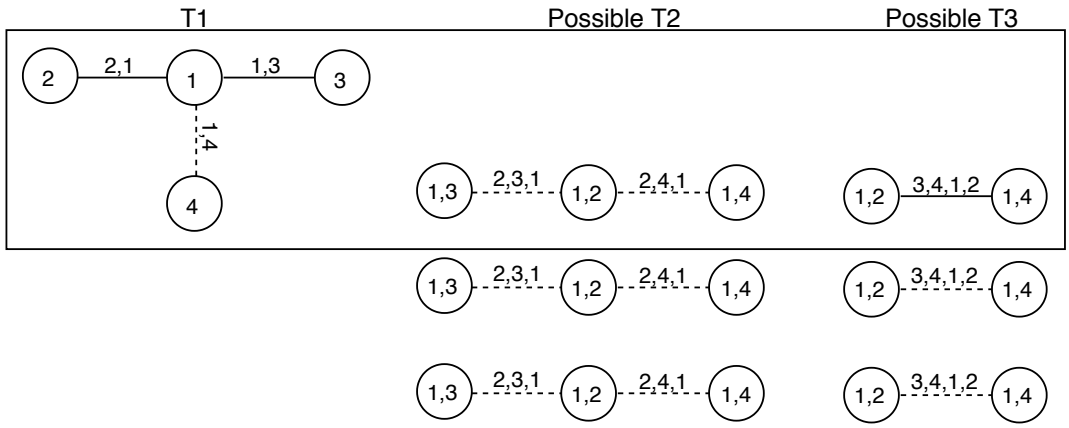


Figure 3.5: Example of a scenario that leads to the evolutionary algorithm being trapped in a sub-optimal local solution. Suppose that $L = [D(X_1, X_2), D(X_1, X_3), D(X_3, X_4 | X_1, X_2)]$ and $n = 4$. Then, prioritized edges associated to dependencies in L are $\{1, 2\}$ and $\{1, 3\}$ for the first level, and $\{3, 4, 1, 2\}$ for the third level, respectively. Since edges $\{1, 2\}$ and $\{1, 3\}$ are assigned high weights, they are inserted in T_1. Suppose that, to complete this tree, $\{1, 4\}$ is also inserted. To build the tree at the second level, T_2, the procedure departs from a graph that contains three possible edges, namely $\{2, 3, 1\}$, $\{2, 4, 1\}$, $\{3, 4, 1\}$. These edges receive low weights as not one is associated with L. Among the three possible trees at the second level, only the tree with the edges $\{2, 3, 1\}$ and $\{2, 4, 1\}$ ensures that the tree at the third level, T_3, represents the prioritized edge $\{3, 4, 1, 2\}$. The optimal solution is framed in the gray rectangle. Discontinuous lines symbolize the prioritized edges and dashed lines, the non-prioritized ones.

3.4.2.2 Analysis of GA Behavior

A question that arises is why sometimes the optimization algorithm converges to sub-optimal solutions using the same dependence list and parameter setting. We address this question in the following lines. Because the construction of R-vine graphs is implicitly sequential, i.e., the construction of a tree depends on the structure of the previous one, it might happen that possible edges not associated to dependencies in L are inserted in the current tree in order to complete its construction. The insertion of those edges may prevent the representation of dependencies in L in subsequent trees, since they are incompatible with the R-vine graph solution. If the dependence list is unfeasible, the previous scenario becomes more evident, since, by definition, these lists contain dependence relationships that are incompatible among themselves with an R-vine graph.

Figure 3.5 illustrates a scenario that leads to the GA being trapped in a sub-optimal solution. Suppose $L = [D(X_1, X_2), D(X_1, X_3), D(X_3, X_4 | X_1, X_2)]$ and $n = 4$. Then, prioritized edges associated to dependencies in L are $\{1, 2\}$ and $\{1, 3\}$ for the first level, and $\{3, 4, 1, 2\}$ for the third level, respectively. Since edges $\{1, 2\}$ and $\{1, 3\}$ are assigned high weights, they are inserted in T_1. Suppose that, to complete this tree, $\{1, 4\}$ is also inserted. To build the tree at the second level, T_2, the procedure departs from a graph that contains three possible edges, namely $\{2, 3, 1\}$, $\{2, 4, 1\}$, $\{3, 4, 1\}$. These edges receive low weights as none is associated with L. Among the three possible trees at the second level, only the tree with the edges $\{2, 3, 1\}$ and $\{2, 4, 1\}$ ensures that the tree at the third level, T_3, represents the prioritized edge $\{3, 4, 1, 2\}$. The optimal solution is framed in the gray rectangle.

Figure 3.6 is based on the experiment reported in Table 3.1, where a feasible dependence list

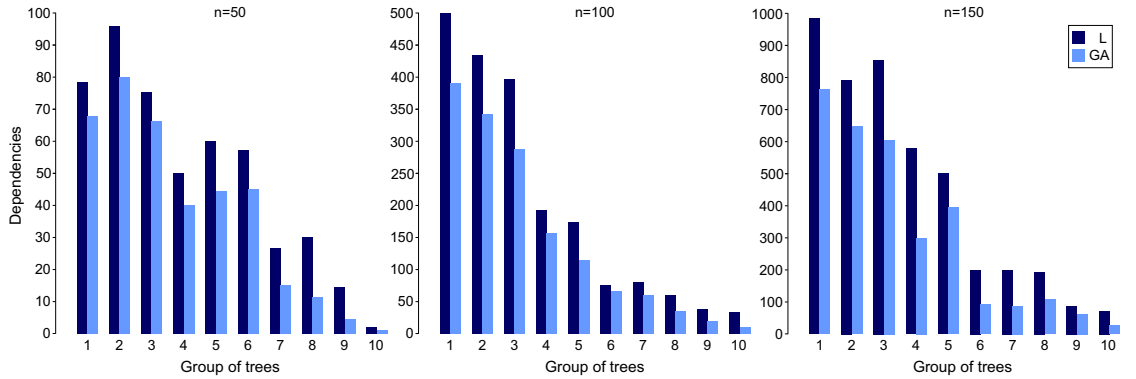


Figure 3.6: Bar-plot for each value of n, $dratio = 50\%$, $hratio = 50\%$, and feasible dependence list. In this plot, only executions where the GA does not converge to an optimal solution are taken into account. The bars stand for several trees as follows: For $n = 50$, each bar accounts for five consecutive trees, except the last bar, which accounts for the last four trees that correspond to 50-dimensional R-vine graphs; for $n = 100$, each bar accounts for ten consecutive trees, except the last bar, which accounts for the last nine trees that correspond to 100-dimensional R-vine graphs; for $n = 150$, each bar accounts for fifteen consecutive trees, except the last bar, which accounts for the last fourteen trees that correspond to 150-dimensional R-vine graphs. This plot shows that the dependencies in L not represented in the generated sub-optimal solutions are distributed in all the trees.

was used. In this plot, only executions where the GA does not converge to an optimal solution are considered. This figure shows a bar-plot for each value of n. The bars group several R-vine trees as detailed in the caption of the plot. Dark-blue bars account for dependencies in L, whereas light-blue bars correspond to $\%Fitness$ of the best sub-optimal solutions found by the GA. This plot shows that those relationships in the dependence list not represented in the sub-optimal solutions are distributed throughout all the trees.

It is expected that the more complete the dependence list is, the easier it is for the optimization algorithm to find optimal solutions. Figure 3.7 illustrates this behavior. The box-plots of this figure show the distribution of the best fitness achieved by the BM (light-blue) and the GA (dark-blue) for different groups. These groups (on the x-axis) indicate the percentage of completeness of the feasible dependence list, which ranges from 10% to 90% with a step of 10 units. From left to right, each figure corresponds to $n = 50, 100, 150$ respectively. Unlike the BM, the GA is highly effective in generating optimal and near-optimal solutions, regardless of the degree of completeness of the dependence list. Nevertheless, we notice that the more complete the list is, the easier it is for the algorithm to accommodate the dependencies of the list. We attribute this advantage to the designed crossover and mutation operators, capable of generating better solutions from generation to generation.

3.4.2.3 Robustness of GA

A fact worth mentioning is that the designed GA is able to achieve optimal or near-optimal solutions with different parameter settings, working with both feasible and unfeasible dependence lists. As shown in Tables 3.1 and 3.2, the GA parameters changed in the experiments are $dratio = 30, 50, 70\%$ (crossover ratio) and $hratio = 50, 70\%$ (mutation ratio). In summary, the proposed GA is robust

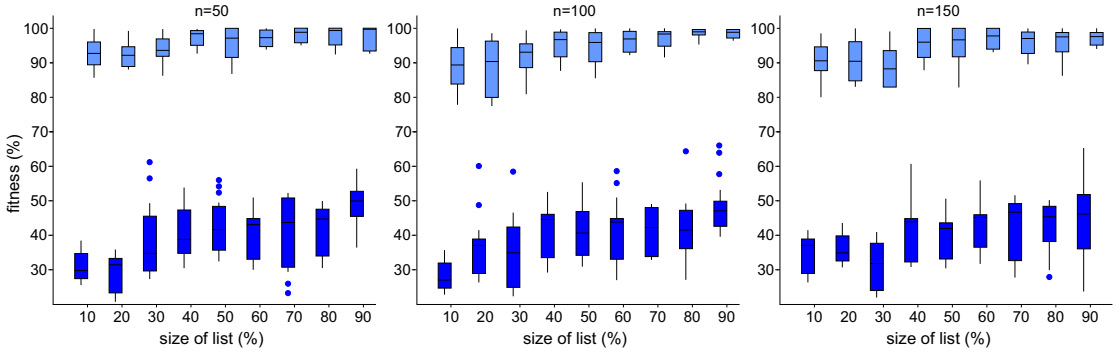


Figure 3.7: Box-plots that show the distribution of the best fitness achieved by the BM (light-blue) and the GA (dark-blue) for different groups. These groups (on the x-axis) indicate the percentage of completeness of the feasible dependence list, which ranges from 10% to 90% with a step of 10. From left to right, each figure corresponds to $n = 50, 100, 150$ respectively. These plots show that the GA is highly effective in generating optimal and near-optimal solutions, regardless of the degree of completeness of the dependence list.

and reliable. Furthermore, it is worth pointing out that the higher the n, the more complex the optimization problem, due to an exponential growth of the search space. As a consequence, this translates into a slight increase in the number of evaluations required to achieve similar fitness values.

3.5 Summary

With the aim of designing methods for learning the graph structure of R-vines from dependence lists, two approaches have been proposed. The first approach is a 0-1 linear programming formulation for building truncated R-vine graphs with only two trees. The second approach consists of a GA, which is able to learn complete and truncated R-vine graphs. A further distinctive feature of the proposed evolutionary approach is that it uses crossover and mutation operators specifically designed to ensure that the generated R-vine graphs are feasible. The designed operators are effective in generating valid and good solutions when working with both feasible and unfeasible dependence lists. Furthermore, this method further fosters a synergy between global and local optimization mechanisms in the sense that, while the MST method produces locally optimized trees, the engineered genetic operators modify those trees in such a way as to generate better global solutions. Experimental results endorse the success of the designed GA in finding R-vine graphs that incorporate the largest number of dependence relationships given in a dependence list. They reveal that although the GA does not guarantee optimal solutions, it is highly effective in producing optimal or near optimal solutions.

Chapter 4

Classification Based on Regular Vine-Copulas

4.1 Introduction

Due to the ability of R-vines to capture complex dependence structures, these models are increasingly gaining prominence in the area of artificial intelligence. In this chapter, we introduce the most general class of regular vine-copulas in the framework of supervised classification in order to provide classifiers with the strengths of these models.

Previous works on the use of copula functions as classifiers have been limited to a few papers [22, 45, 112, 115, 123]. In [45], a graphical model called Copula Bayesian Network is introduced and applied to several supervised classification benchmark problems. This model integrates the copula and Bayesian network frameworks, capturing the multivariate dependence structure through a set of local copula densities associated with the nodes of the network. The BIC is used to select the copula that best fits the data between the Normal and Clayton to model the local copulas. In [112], a classifier based on three-variate Normal copulas is used for classifying when a color pixel in an image belongs to either the foreground or the background class.

The goal of supervised classification is to assign a new instance to a label based on its features. Usually, the classification methods based on probabilistic models perform this task by learning the distributions of features for each class from a labeled dataset [44, 52, 85]. Particularly, the classification approach proposed in this thesis consists of estimating a regular vine-copula model for each class, from the corresponding training samples, to then assign the label with the highest probability to each new sample.

The valuable properties of regular vine-copulas to model the dependence of multivariate distributions motivate us to select them to address two real-world classification problems in which complex and diverse patterns of dependence between variables might arise. The first application called Mind Reading Problem (MRP) [96] belongs to the class of mental signal classification problems. In particular, the MRP consists of inferring which type of video a subject has watched from multiple time series recorded from different brain regions. The second application, called Dune Classification Problem (DCP) [5], is an image recognition task whose goal is to detect the presence or absence of sand dunes from remotely sensed images of the surface of Mars using gradient histogram features extracted from the images.

It is worth noticing that, through the MRP and the DCP applications, we show the usefulness of vine-copulas in the modeling of high dimensional problems. The first papers reporting R-vine-based applications addressed problems with a small number of variables, for instance [1, 122]. However,

recent research has reported R-vines results on larger problems, for instance, 448 variables in [23], 400 in [93], and 96 in [94], among others. In the present thesis, the dimension of the MRP is 408 [25] and of the DCP is 180 [24].

This chapter is organized in three parts: Section 4.2 outlines the vine-copula classification approach proposed in this dissertation. In Sections 4.3 and 4.4, the designed vine-copula-based classifiers are tested by their application in MRP and DCP, respectively. Section 4.5 provides the conclusion of the chapter.

4.2 Regular Vine-Copula Classification Approach

When using probabilistic models for supervised classification, a possible approach is to learn a model for each class of the problem. This is the approach used in this thesis.

Using Bayes's rule, the classifier learns one model $f(\boldsymbol{x}|k)$ for each class $k \in \{k_1, \ldots, k_K\}$ from the corresponding set of labeled training observations of a feature vector $\boldsymbol{X} = (X_1, \ldots, X_n)$. K denotes the number of labels. The probability of the unlabeled observation $\boldsymbol{x} = (x_1, \ldots, x_n)$ of being assigned to the class k is expressed as

$$p(k|\boldsymbol{x}) \propto f(\boldsymbol{x}|k) \cdot p(k), \tag{4.1}$$

where $p(k \mid \boldsymbol{x})$ is the (posterior) probability of k given $\boldsymbol{x}$, and $p(k)$ is the (prior) probability for k.

The learned models are used to predict the most likely class $k^* \in \{k_1, \ldots, k_K\}$ of the unlabeled observation $\boldsymbol{x}$, which is determined by choosing the label with the highest probability. This decision rule is formulated as

$$k^* = \underset{k \in \{k_1, \ldots, k_K\}}{\operatorname{argmax}} f(\boldsymbol{x}|k) \cdot p(k). \tag{4.2}$$

Assuming that $f(\boldsymbol{x}|k)$ in (4.1) is codified by R-vines defined in (1.23), we have

$$f(\boldsymbol{x}|k) = \prod_{T_j \in G} \prod_{\{i,k,\boldsymbol{S}\}} c_{i,k|\boldsymbol{S}} \left(F_{i|\boldsymbol{S}}(x_i|\boldsymbol{x}_{\boldsymbol{S}}), F_{k|\boldsymbol{S}}(x_k|\boldsymbol{x}_{\boldsymbol{S}})|k \right) \cdot \prod_{i=1}^{n} f_i(x_i|k) \tag{4.3}$$

For C-vine and D-vine classifiers, the dependence structure of $f(\boldsymbol{x}|k)$ is codified by the corresponding expressions in (1.25) and (1.26).

4.2.1 Regular Vine-Copula Strategies

In the design of classification strategies based on regular vine-copulas, among the most relevant factors that should be taken into account are the type of vine-copula model (e.g., C-vine, D-vine, R-vine), the diversity of copulas to be fitted (e.g., Normal, Clayton, Gumbel), the types of marginal distributions (e.g., Gaussian, Beta, Gamma), and the number of trees of each model (the structure can be complete or truncated). According to this spectrum of choices, in order to facilitate the study of the properties of these models to deal with classification tasks, we define two vine-copula classification strategies, namely unmixed or homogeneous, and heterogeneous. The latter is further divided into partially-mixed and fully-mixed strategies.

Unmixed or homogeneous Regular vine-copula models comprising the classifiers of this group use a single bivariate copula family and a single type of univariate distribution to model the pair-copulas and marginals in (4.3) respectively.

Partially-mixed heterogeneous Allow for more flexible classifiers than those described above since pair-copulas of different families are combined in the same model. However, the marginals are of a single type of distribution.

Fully-mixed heterogeneous Among the heterogeneous classifiers, fully-mixed are the most flexible since their regular-vine-copula models combine pair-copulas of different families and marginals of different types of distributions.

These classification approaches are evaluated in a comprehensive numerical study in the MRP and DCP applications using both D-vines and R-vines. As the graph structure of R-vines is more expressive than that of D-vines, we also assess the impact of this characteristic in the effectiveness of the tested classifiers.

We emphasize that when we refer to the number of trees of a regular vine-copula model, we are referring to whether or not the model is truncated. For instance, we say that a D-vine or R-vine of n variables has $t < n - 1$ trees when in the subsequent levels only Product pair-copulas are fitted.

4.3 Application to the Mind Reading Problem

This section presents the research carried out on how the properties of regular vine-copula models can be exploited to create successful probabilistic classifiers through their application in the MRP.

4.3.1 Description of the MRP

MRP consists of decoding the original stimuli (e.g., seen, heard) received by a subject from the analysis of his/her brain signals. The type of stimuli as well as the methods used to register the subject's brain activity may differ, e.g., electroencephalography (EEG) vs. magnetoencephalography (MEG) [96]. Nevertheless, a common element on different types of recording of brain signals is the high number of features involved and the high variability of the data.

The particular publicly available MRP benchmark produced by the Mind Reading Challenge Competition consists of inferring the type of video stimulus shown to the subject from MEG brain signals (recorded from a single subject in two experimental sections at different time-points). A successful brain decoding classifier should be able to recognize which type of video (among the five possible) the subject has watched by analyzing the brain signals. All videos were presented without audio, and five different types of stimuli were used:

- Class 1 (c1): Artificial Screen savers showing animated shapes or text.
- Class 2 (c2): Nature Clips from nature documentaries showing natural scenery such as mountains or oceans.
- Class 3 (c3): Football Clips taken from European football matches of "La Liga" in Spain.
- Class 4 (c4): Mr. Bean Clip from the episode "Mind the baby, Mr. Bean" of the Mr. Bean television series.
- Class 5 (c5): Chaplin Clip from the "Modern Times" feature film.

MRP is considered a challenging problem for the classification methods. The brain data are highly noisy by virtue of the variability of brain signals recorded in different sessions (in particular, two different days) which immediately implies the variability between the distribution of the training

and test data. The main consequence of this scenario is that the classifiers learned using the training data may have a low prediction accuracy on the test data. Another source of difficulty is the dimension of the problem, which has 408 features.

In this experimental benchmark, the training and test data are taken from two different sessions: 677 training examples were recorded the first day and 653 examples the second day; 50 samples from the second day will be used for the training to facilitate the modeling of possible variations in the data distribution between 2 the days.

ML algorithms have been used to decode the information contained in the brain signals and recognize a stimulus received by the subject or an intended action (e.g., decide between right and left movements) [89]. From the ML point of view, the MRP is a multi-class classification problem with five classes. This problem is considerably difficult due to the highly noisiness of the brain data, which corresponds to a set of multivariate time series with a large number of variables. Moreover, there is also a variability between the distribution of the training and test data, a problem usually known as *covariate shift* [118].

The mind decoding problem involves different multi-disciplinary approaches, from the conception of the experiments and a preliminary analysis of the mental mechanisms implicated, to the decision on the type of signals to be investigated and the recording devices to employ. In this thesis, we approach the MRP from a pure ML perspective, in which the goal is to address the classification problems involved in the MRP. However, in order to understand the peculiar features of this problem, we provide the sufficient background on the necessary experimental conditions and also a brief review of previous works on classification approaches to mind decoding and related problems. Special attention is given to classification methods that were previously applied to the data used in the numerical study of this thesis.

Much effort has been devoted to solve the mind decoding problem in the context of research on brain-computer interfaces (BCIs) [88,131]. BCIs are devices conceived to translate electrical signals into commands without the need for motor intervention. Different voluntary and involuntary mental processes can serve as a basis for implementing BCIs, but in most of the cases a decoding component is required to decode the stimulus received by the subject or his intended command. As part of this decoding component, classification algorithms are implemented. A variety of classification algorithms have been used to analyze brain data in the context of BCI applications [89]. They include Linear Discriminant Classifiers (LDA) [53], Support Vector Machines (SVMs) [12, 106], Neural Networks (NNs) [66], and other classification methods [99, 107, 109, 114]. For a survey of classification algorithms applied to BCI, [89] can be consulted.

Until recently, an obstacle limiting the advance of research on MRPs was the lack of available brain recording databases where ML approaches could be tested. However, in the past few years a number of brain decoding or mind reading competitions have provided the needed benchmarks to evaluate and develop more accurate classifiers for this problem. In particular, the Mind Reading Challenge Competition [80] allowed the comparison of nine different approaches to solve this challenging classification task.

Now, we briefly discuss the three best approaches presented in this competition. Huttunen et.al. [73] presented an algorithm based on regularized logistic regression. The authors use the mean and the slope of the selected features. Samples were weighted such that data from the second day had a higher weight in the cost function used to learn the classifier. Also, to estimate the error, large computational resources were employed. The training data were split in two parts, and this was done in hundreds of ways. For each split, a dedicated processor was assigned to the k-fold cross validation task. The computation was done on a grid of computers.

An ensemble of three types of classifiers is used in [113]: regularized multi-logistic regression, regression trees, and an affinity propagation [50] based classifier. The authors also proposed the combination of the classifiers learned from different types of features extracted from raw data,

channel correlations, mutual information between channels, and channel interaction graphs. The computation of this large number of features is a costly process. In [113], it is not discussed whether the classification accuracy obtained by the ensemble approach is due to the synergy among the different types of features used or to the combination of different classifiers.

The approach proposed in [78] used a representation based on power features from the filtered MEG signals. In addition, linear one-versus-all logistic regression-based classifiers were applied for dimensionality reduction of the MEG gradiometer channel space. Nonlinear Gaussian process (GP) multi-class classifier with a squared exponential covariance function [108] was applied to predict the class from the features. Although the GP approximation is suitable for theoretical analysis, the learning process can be computationally demanding and sensitive to numerical instabilities. However, these aspects of the solution were not discussed in [78].

4.3.2 Adding Flexibility to D-vine Classifiers

In the first application of the proposed vine-copula classification approach, we focus only on D-vine structures. According to the classification strategies introduced in Section 4.2.1, we define eight types of homogeneous and heterogeneous D-vine classifiers.

It is worth remembering that the learning algorithm for D-vines only requires finding the path of n nodes that maximizes all pairwise Kendall's tau values used as weights for T_1, as the rest of the structure is completely determined by this tree. To further simplify the model building, the truncation level is fixed (given as an input parameter of the learning algorithm), instead of being determined by the AIC/BIC model selection strategy presented in Section 1.7. In addition, all the marginal distributions of the proposed classifiers are Gaussian.

Next, we describe the individual characteristics of designed D-vine classifiers to approach the MRP.

Unmixed or homogeneous

- D-vine-P-g is the simplest D-vine classifier of this group. It uses only bivariate Product (P) copulas. Therefore, $f(\boldsymbol{x}|k) = \prod_{i=1}^{n} f_i(x_i|k)$ (see (4.3)).
- D-vine-t1-N, D-vine-t2-N, D-vine-t3-N: These classifiers build D-vine models with the same number of trees—one, two or three, denoted as t1, t2, and t3 respectively. They use only bivariate Normal (N) copulas.

Partially-mixed heterogeneous

- D-vine-t1-Sel, D-vine-t2-Sel, D-vine-t3-Sel: As in the homogeneous classifiers, these classifiers build D-vine models with the same number of trees, i.e., one, two or three, denoted as t1, t2, and t3 respectively. Pair-copulas are selected (Sel) individually from a set of candidate bivariate copula families.
- het-D-vine: This classifier differs from the previous ones in that the number of trees of the models comprising this classifier can be different.

We notice that the D-vine models comprising these classifiers are learned using Algorithm 1.1.

4.3.3 Experiments

We empirically investigate the classification approaches described in Section 4.3.2 through classifiers based on D-vine models in the solution of the MRP. First, we present a description of the

experimental framework, explaining in detail the classification scenarios considered and how the training and test data are selected. Then, we present a comparison between different variants of D-vine classifiers introduced previously in Section 4.3.2. Furthermore, we compare the proposed D-vine-copula approach with other classification approaches applied to MRP.

4.3.3.1 Experimental Framework

As previously mentioned, in the experimental benchmark for the MRP, the training and test data were recorded from two different sessions. One important question is to determine whether the potential variations between the distribution of the training and test sets affects the classification accuracy. Therefore, we considered two different scenarios for the validation of the classifiers:

Training 1 The training dataset contain examples recorded only the first day.

Training 2 The training dataset are mixed, containing the 677 examples recorded the first day plus 50 examples of the second day.

According to the probabilistic classification approach proposed in this thesis (see Section 4.2.1), to address the MRP, we assume that $f(\boldsymbol{x}|k)$ (4.1) is codified by D-vine-copulas (1.26). This means that the dependence structure between the problem features and the target class is expressed by a set of D-vine-copula models. Therefore, as the MRP challenge has five stimulus categories, one D-vine copula per category is learned from the corresponding training data using the TDSH procedure (see Algorithm 1.1). As a result, a single D-vine classifier comprises five D-vine copulas of 408 variables.

The set of candidate bivariate copulas used by the four heterogeneous D-vine classifiers are Product, Normal, Clayton and Gumbel. Both homogeneous and heterogeneous classifiers use only Gaussian marginals.

Furthermore, taking into account possible differences between the distribution training and testing datasets, we carry out a five-fold cross validation (CV) [110] with 30 repetitions using the training dataset in order to highlight these differences. In this context, we will evaluate the performance of the classifiers at two different steps:

- At the validation step, a five-fold cross validation is carried out using only the training dataset.
- At the classification step, a D-vine copula classifier is learned from the training dataset; then, the instances of test dataset are classified using the D-vine classifier.

The metric accuracy, computed from the confusion matrices, is used to evaluate the global and per class performance of the tested classifiers.

4.3.3.2 Comparison Between Training Datasets

In this experiment, we assess the effectiveness of D-vine classifiers when trained with the Training 1 and Training 2 datasets. As to be expected, the classifiers trained with the Training 2 dataset achieve higher classification accuracy values than those trained with the Training 1 dataset. These results are shown in Table 4.1 where the highest accuracy values achieved by each classifier at the classification step are highlighted in bold. One possible explanation for this behavior is that, by including data from the second day, the classifiers learn those dependence patterns that remain unchanged across sessions. Furthermore, the number of available samples is larger in the Training 2 dataset, so this additional information may contribute to improve the classification accuracy. From here on, the following experiments are performed just for the Training 2 dataset.

Table 4.1: Comparison between D-vine classifiers in the Training 1 and Training 2 scenarios.

	Accuracy			
	Training 1 Dataset		Training 2 Dataset	
Classifier	Validation Step	Classification Step	Validation Step	Classification Step
D-vine-P	62,00	49,15	62,71	**52,22**
D-vine-t1-N	72,36	46,09	71,81	**48,85**
D-vine-t2-N	81,40	52,52	79,98	**59,57**
D-vine-t3-N	82,07	52,67	79,79	**62,02**
D-vine-t1-Sel	61,06	48,08	68,16	**52,67**
D-vine-t2-Sel	76,72	53,44	73,47	**60,79**
D-vine-t3-Sel	77,40	54,21	75,03	**62,63**

4.3.3.3 Heterogeneous vs. Homogeneous

Here, we test whether D-vine-based classifiers can recognize the dependence patterns that arise in MRP. A simple visual inspection of pairwise scatter plots (Figures 4.1-4.5) for 10 arbitrary features (from a total of 408) from all datasets reveals a great variety of pairwise patterns: positive/negative relationships as well as weak/strong dependence strength, according to the small and large absolute values of Kendall's tau (located in the upper-triangle of the pair-matrix plots).

The global and per class accuracy values achieved by the tested homogeneous and heterogeneous classifiers at validation and classification steps for the Training 2 dataset are displayed in Table 4.2. The highest accuracy achieved in each class is highlighted in bold. The numerical results show that, essentially, all classifiers performed well. Nevertheless, at the validation step, the accuracy of the tested classifiers is found to be higher than in the classification step, which indicates possible overfitting.

The effectiveness of (partially-mixed) heterogeneous classifiers (they use Product, Normal, Clayton and Gumbel copulas) is clearly higher than those based exclusively on Normal copulas. The covariate shift has a more negative effect on Normal classifiers than on the heterogeneous ones. Since the latter combine copulas from different families, they are better equipped to capture the variations in the data by the covariate shift phenomena. We also observe that the accuracy increases with the number of trees. In fact, at the validation and classification steps, D-vine-t3-Sel achieves the highest accuracy among all tested classifiers. It is also the most accurate in three classes: c2, c4, c5. Another observation is that classifiers whose D-vine models use more trees (allowing them to capture a larger number of dependencies) perform better when compared to classifiers of the same group.

An analysis of these results suggests that the better the D-vine models represent the dependence patterns that emerge in the MRP via combining copula families in a larger number of trees, the better they are able to distinguish among the different classes. Thus, we would prefer heterogeneous classifiers rather than homogeneous ones, even when learning them involves additional computational cost.

4.3.3.4 Mix of Different D-vines in a Classifier

So far, we have modeled the dependence structure between the problem features and the target class by building a set of D-vine models, which use the same families of bivariate copulas and have the same number of trees (i.e., all five models, one per class, are truncated at the same level).

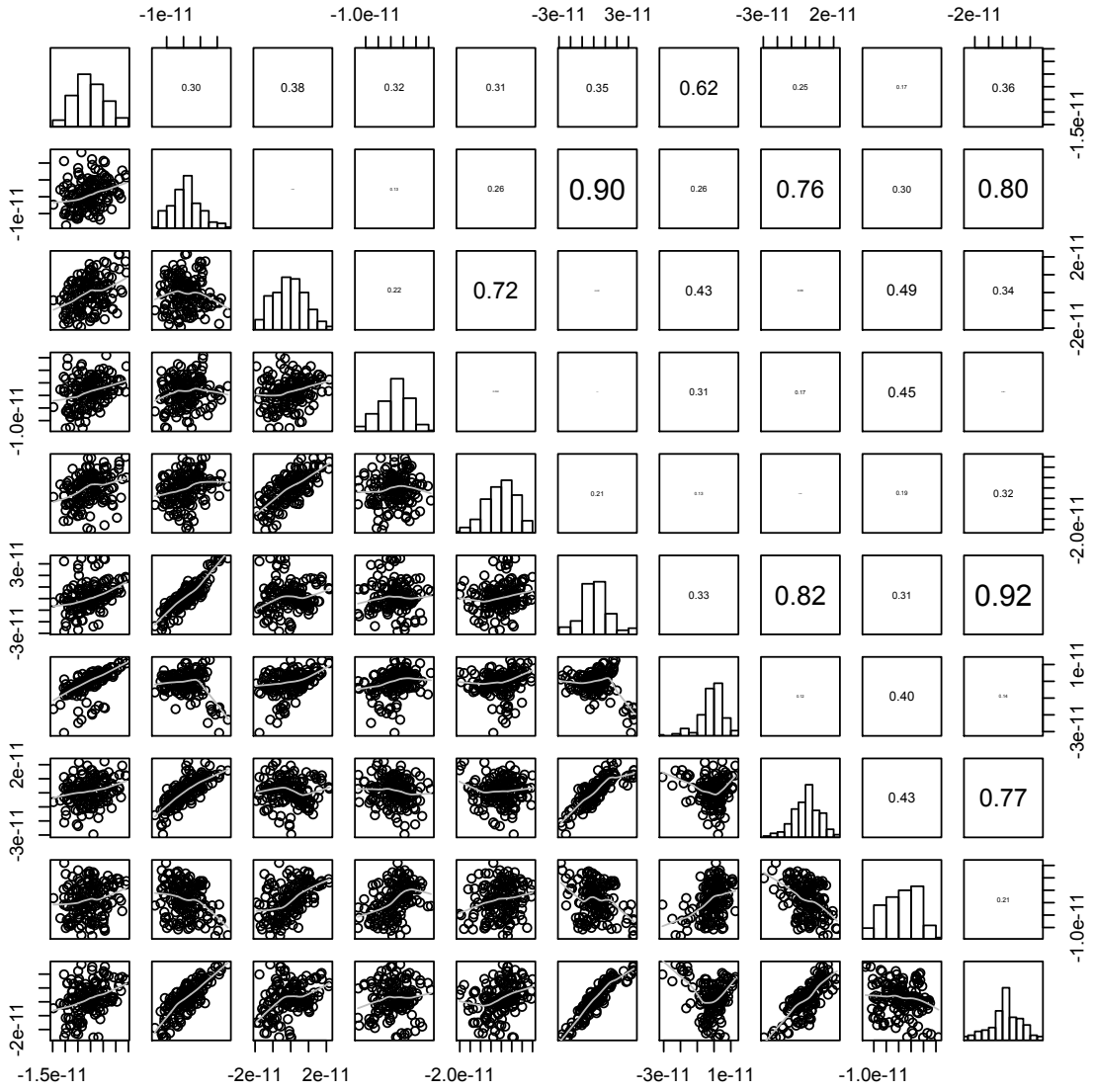


Figure 4.1: Pairwise scatter plot of 10 arbitrary features from a total of 408 from the sample data of the class c1 (Artificial).

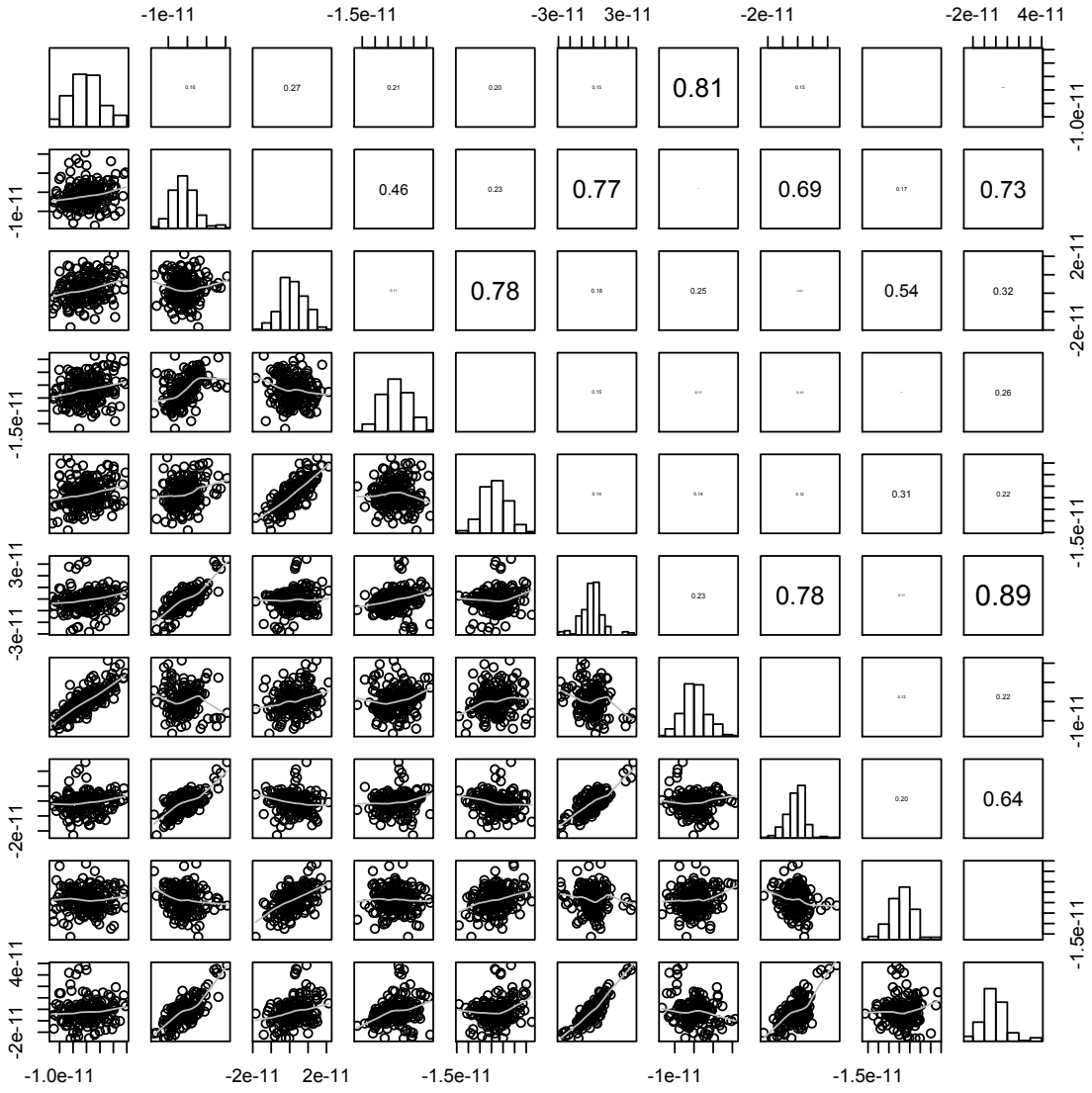


Figure 4.2: Pairwise scatter plot of 10 arbitrary features from a total of 408 from the sample data of the class c2 (Nature).

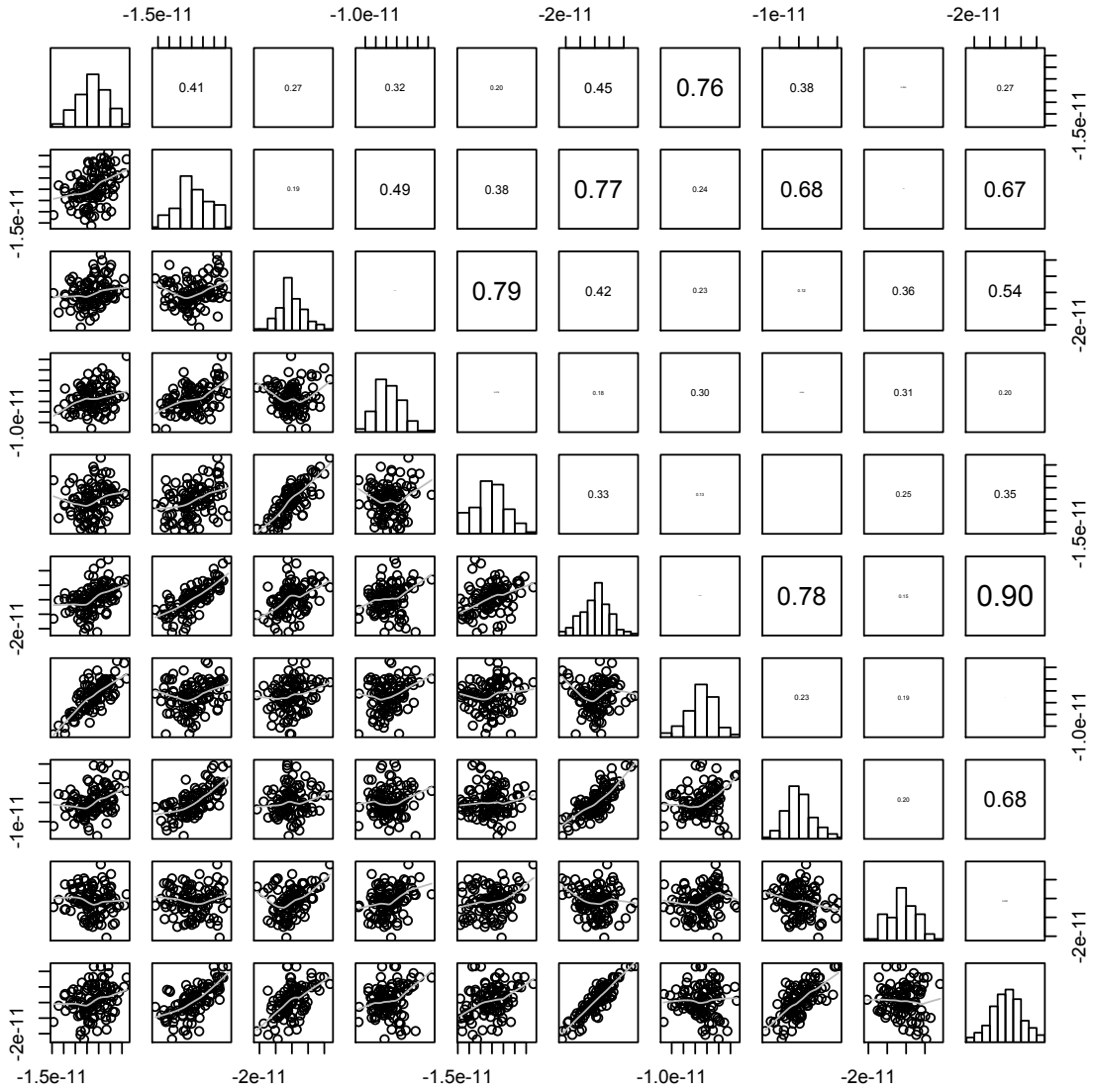


Figure 4.3: Pairwise scatter plot of 10 arbitrary features from a total of 408 from the sample data of the class c3 (Football).

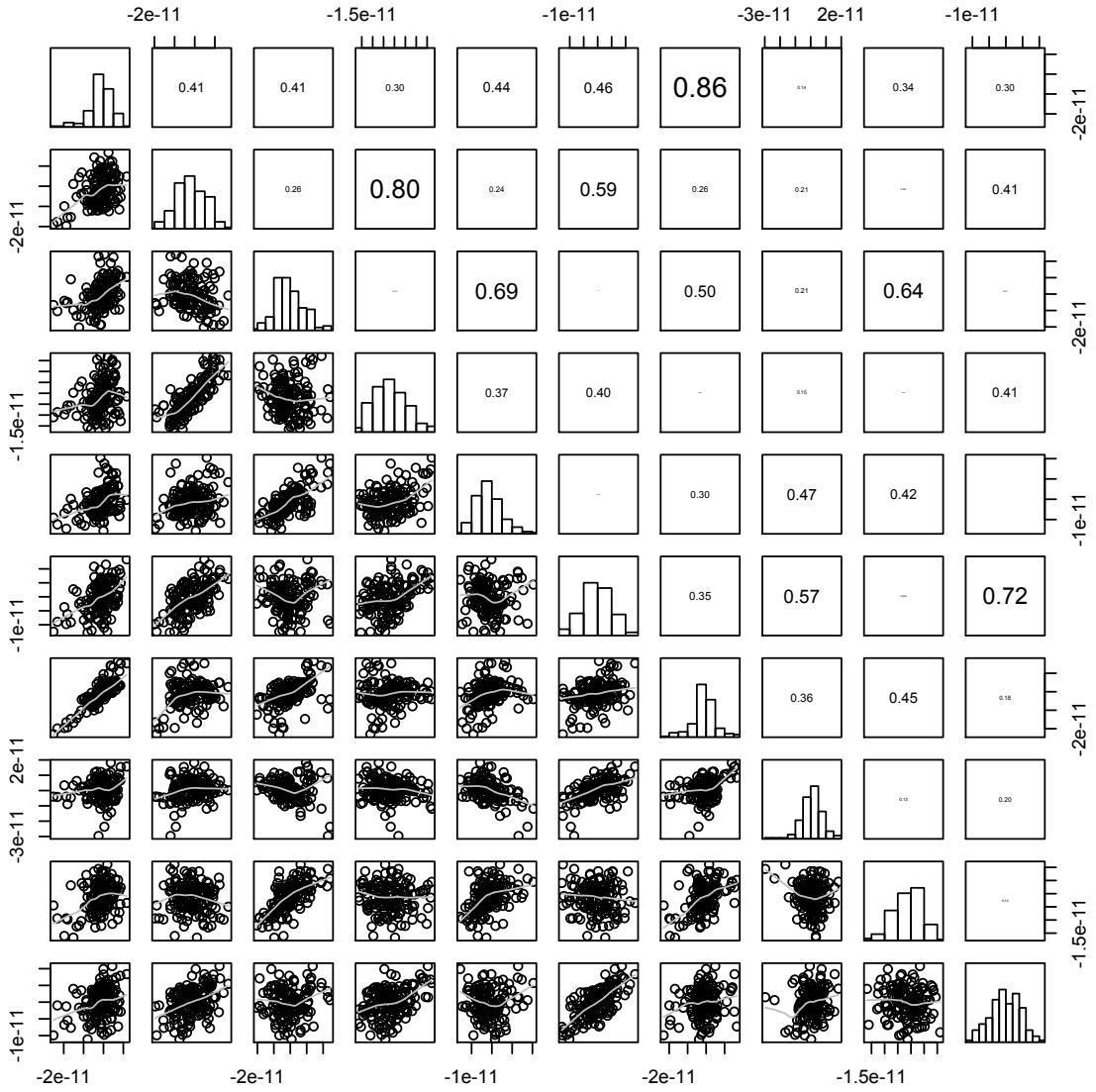


Figure 4.4: Pairwise scatter plot of 10 arbitrary features from a total of 408 from the sample data of the class c4 (Mr. Bean).

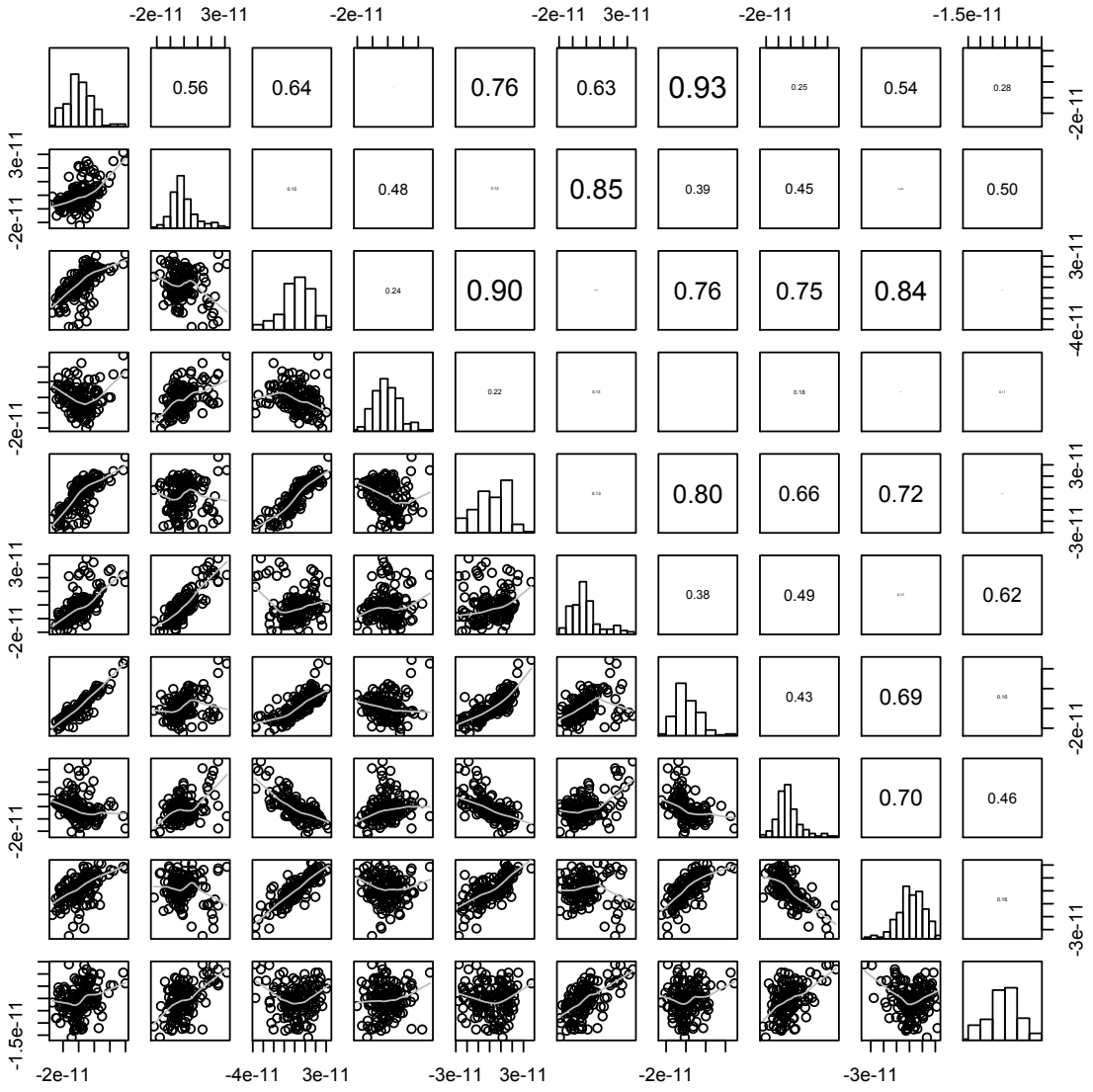


Figure 4.5: Pairwise scatter plot of 10 arbitrary features from a total of 408 from the sample data of the class c5 (Chaplin).

Table 4.2: Global and class accuracy at the validation and classification steps in the Training 2 dataset. c1 - Artificial, c2 - Nature, c3 - Football, c4 - Mr. Bean, c5 - Chaplin classes.

Classifier	Accuracy					
	c1	c2	c3	c4	c5	Global
Validation Step						
D-vine-P	79,81	53, 71	57, 10	35, 01	87, 94	62,71
D-vine-t1-N	58, 33	67, 08	62, 61	86, 72	84, 31	71, 81
D-vine-t2-N	64, 50	79, 72	**72, 61**	**94, 31**	88, 79	**79, 98**
D-vine-t3-N	**67, 50**	**83, 19**	66, 90	92, 41	**88, 96**	79, 79
D-vine-t1-Sel	60, 00	61, 80	50, 23	85, 68	83, 10	68, 16
D-vine-t2-Sel	59, 30	78, 19	50, 71	92, 75	86, 37	73, 47
D-vine-t3-Sel	59, 33	81, 52	52, 61	96, 20	85, 51	75, 03
Classification Step						
D-vine-P	**76, 70**	35, 76	47, 05	16, 80	82, 40	52, 22
D-vine-t1-N	62, 66	47, 01	49, 01	14, 40	68, 80	48, 85
D-vine-t2-N	54, 66	**64, 90**	47, 05	44, 00	84,80	59, 57
D-vine-t3-N	50, 00	64, 23	50, 98	59, 20	85,60	62, 02
D-vine-t1-Sel	55, 33	47, 68	**51, 96**	32, 00	76, 80	52, 67
D-vine-t2-Sel	51, 33	60, 92	36, 27	68, 00	84, 80	60, 79
D-vine-t3-Sel	48, 66	**64, 90**	35, 29	**76, 80**	**86, 40**	**62, 63**

Table 4.3: Confusion matrices at the classification step in the Training 2 dataset. c1 - Artificial, c2 - Nature, c3 - Football, c4 - Mr. Bean, c5 - Chaplin classes.

	D-vine-t1-N					D-vine-t2-N					D-vine-t3-N				
Class	c1	c2	c3	c4	c5	c1	c2	c3	c4	c5	c1	c2	c3	c4	c5
c1	94	18	29	4	5	82	34	23	6	5	75	38	24	8	5
c2	53	71	17	5	5	33	98	8	7	5	27	97	10	12	5
c3	31	17	50	3	1	32	18	48	3	1	31	13	52	5	1
c4	22	17	41	18	27	11	12	17	55	30	7	4	9	74	31
c5	13	0	22	4	86	4	2	7	6	106	2	2	6	8	107
	D-vine-t1-Sel					D-vine-t2-Sel					D-vine-t3-Sel				
Class	c1	c2	c3	c4	c5	c1	c2	c3	c4	c5	c1	c2	c3	c4	c5
c1	83	19	33	8	7	77	34	11	19	9	71	40	11	21	7
c2	44	72	22	6	7	26	92	6	21	6	20	98	2	25	6
c3	34	9	53	5	1	32	20	37	12	1	32	19	36	12	3
c4	26	5	23	40	31	5	3	0	85	32	4	1	0	96	24
c5	9	1	11	8	96	1	1	2	15	106	1	1	2	13	108
	D-vine-P														
Class	c1	c2	c3	c4	c5										
c1	115	8	24	2	1										
c2	72	54	18	5	2										
c3	46	4	48	2	2										
c4	35	7	45	21	17										
c5	7	0	13	2	103										

Table 4.4: Global accuracy and confusion matrix of het-D-vine in the Training 2 dataset.

Class	het-D-vine	Accuracy
c1	D-vine-t2-N	61,33
c2	D-vine-t3-Sel	52,98
c3	D-vine-t3-N	54,90
c4	D-vine-t3-Sel	64,00
c5	D-vine-t2-N	88,00
	het-D-vine	**64,01**

Confusion matrix of het-D-vine					
Class	c1	c2	c3	c4	c5
c1	92	16	24	12	6
c2	34	80	16	16	5
c3	34	4	56	7	1
c4	12	0	12	80	21
c5	5	1	7	2	110

Table 4.5: Global accuracy of the best four teams at Mind Reading Challenge and het-D-vine in the Training 2 dataset.

Rank	Team	Accuracy
1	Huttunen et.al.	68,0
2	het-D-vine	64,1
3	Santana et.al.	63,2
4	Jylanki et.al.	62,8
5	Tu et.al.	62,2

However, this approach does not necessarily have to be the most appropriate. Notice, for example, that D-vine-t3-Sel is far more accurate than D-vine-t1-Sel (62,63% vs. 52,67%). In contrast, D-vine-t1-Sel reaches better accuracy than D-vine-t3-Sel in two of the five classes, namely c1 (55,33% vs. 48,66%) and c3 (51.96% vs. 35.29%). This information can be found in Table 4.2. The global and per class accuracy values of the tested classifiers are highlighted in bold.

This information can be seen in Table 4.2 which shows the global and class accuracy values of the tested D-vine-based classifiers. These values are calculated from the confusion matrices shown in Table 4.3. This behavior suggests a more flexible approach to classifier design that combines different D-vine models, regarding the pair-copula families as well as the number of trees they use. This is exactly the rationale of the heterogeneous classifier het-D-vine presented in Section 4.3.2.

In the experiment of this section, we tested all the classifiers resulting from combining all the possible configurations used in the construction of classifiers analyzed in the previous section (see Table 4.2). The model combination yielding the highest classification accuracy, denoted as het-D-vine in Table 4.4, is as follows: Mixed D-vine-copulas with three trees for c1 and c2, and Normal D-vine-copulas with two trees for c1 and c5, and three for c3. We point to the fact that model combination of het-D-vine allows the accuracy of the best heterogeneous classifier to be to outperformed, namely D-vine-t3-Sel (see Table 4.2). Such a result suggests that the dependence structure of brain signals appears to be quite complex across classes.

4.3.3.5 Comparison with Other Algorithms

We compare het-D-vine with the four best algorithms at the Mind Reading Challenge Competition according to the global accuracy. In Table 4.5, we can see that het-D-vine ranks second compared to previous algorithms. We notice that such an accuracy has been obtained without any further processing of the data or applying more elaborated strategies for D-vine model selection as those used in previous works (see Section 4.3.1).

4.4 Application to the Dune Classification Problem

This section extends the classification approach presented in Section 4.3 in two significant ways as follows: (i) In order to extend the representation capabilities of D-vine classifiers, instead of D-vines, we use R-vines. (ii) In order to gain efficiency in the building of classifiers without compromising its effectiveness, instead of using a different R-vine graph for each class, we learn a single structure for all the classes, while the pair-copulas are estimated from the corresponding dataset. Both aspects are discussed in more detail in section 4.4.1.

Here, two learning methods that guarantee the same structure for all the classes of an R-vine classifier are introduced. These methods are evaluated through their application in the DCP.

4.4.1 Description of the DCP

Aeolian features are those produced by the action of the wind on a given surface. They are not only found on Earth but have also been reported for other planets and satellites [48, 127]. Analyzing the characteristics of these features can provide information about the current or past atmospheric circulation patterns on the planet; providing therefore relevant information about the atmospheric conditions in the area.

Dunes are the most frequent aeolian features on the Martian surface, and their study contributes to the understanding of the interactions between the atmosphere and the surface of the planet, the way the climate has evolved throughout the history of Mars and how it currently is [60, 130].

In recent years, the need to process large volumes of remotely sensed images has greatly boosted research into the use of automated methods for feature and change detection of structures on planetary surfaces and several works on the analysis of remote sensed imagery have been reported [11, 13, 47, 72, 119].

A geological classification scheme of sand dunes was proposed in [90] for terrestrial examples, mostly based on field-work. So far, the dunes identified on the Martian surface have been classified according to that scheme, and, although most of them fit into the main types, there are some undefined morphologies not known to occur on Earth [68].

Examples of the great diversity of types of Martian dunes can be seen in Figure 4.6: barchan, barchanoid, transverse, dome, linear, and star, being among the most representative [68]. From this, the multitude of factors that affect the visual aspect of dune fields become clear (e.g., constituents, size, shape and density, association to seasonal advance and withdrawal of ice cover, and angle of illumination, among others) and also the need to design classification strategies capable of detecting the presence of dunes on images.

Recently, supervised learning approaches based on image analysis and pattern recognition techniques have been applied to detect sand dunes on remotely sensed images of the surface of Mars. In particular, SVMs [126] and Boosting [137] algorithms were applied to features derived from gradient analysis made at each pixel of the images [4]. However, although a diversified image dataset was used containing examples from both hemispheres of Mars, the dunes present were mainly of the barchan and barchanoid types. In a subsequent work, SVMs and Random Forest (RF) [20] were evaluated for this problem with a set of high spatial resolution images including all types of Martian dunes [5].

Inspired by [5], in this thesis we deal with the dune detection problem, which consists of identifying the presence or absence of sand dunes from remotely sensed images of the surface of Mars, whatever their scale. The set of features extracted from these images describes both the directional and periodic characteristics of the dunes, regardless of the diversity of Martian dune types. In this thesis, we use the same methodology and feature representation introduced in that work, but we adopt a regular vine-copula based approach.

We also modify the approach developed in Section 4.3.2 in two main ways:

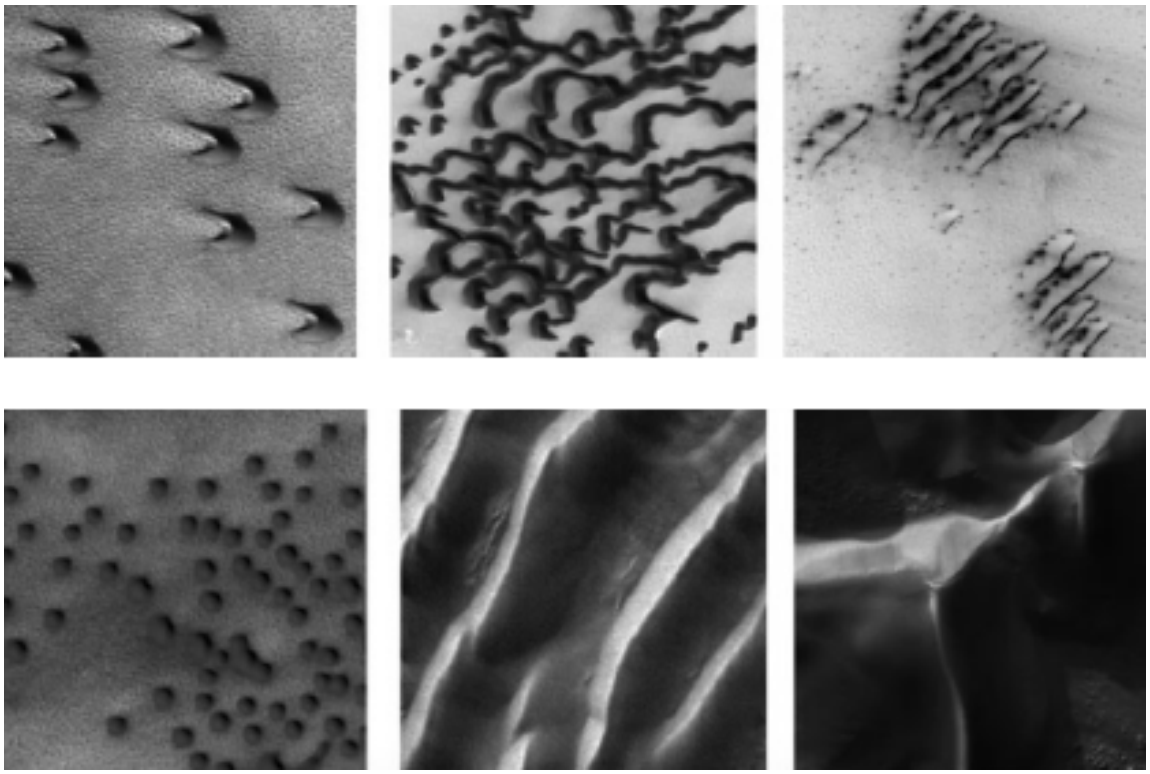


Figure 4.6: Examples of the diversity of Martian sand dunes (the side of each square image is $2500m$): From left to right and top to bottom: barchan, barchanoid, transverse, dome, linear, and star. Image credits: NASA/JPL/MSSS.

1. The classification approach utilized in Section 4.3.2 is based on D-vine distributions. In these models, the node trees are organized in a specific and fixed order, in such a way that each tree has a path structure. The structure of the D-vine is more restrictive than the R-vine structure, which can limit their capacity to produce accurate factorizations of distributions.

2. The classifiers designed to address the MRP (see Section 4.3.2) learn a different D-vine graph for each class of the MRP. This characteristic also represents a limitation in terms of the interpretability and the potential use of the classifier to reveal insights about the significant pairwise dependencies as well as the variability of them across classes.

In a probabilistic classification approach, it is common to assume the structure of the models to be the same among classes. This is the case of the NB, which assumes attributes are independent of each other, given the class; and more remarkably the TAN, which employs a tree structure where each attribute only depends on the class and one other attribute as parent nodes. In both classifiers, there is a unique tree structure shared among the classes, although it is learned with a different method to the one we propose here. Sharing structure sacrifices some accuracy but the task of identifying which pairwise dependencies are those that can help to characterize the classes, and the one of assessing how the strength and shape of the bivariate dependencies changes among the classes, are eased.

Although this constraint leads to a lack of flexibility of the model, and therefore deteriorates the performance of the classifiers, the experimental results show that the impact is not too severe due to the fact that R-vine classifiers with shared structure still keep a high degree of flexibility owing to the use of pair-copulas of different families that are estimated from the data of the corresponding class. This strategy also contributes to reducing the cost of construction of R-vine classifiers because only a single structure is built for all classes instead of one for each class.

4.4.2 Learning R-vine Classifiers with a Common Structure

Algorithm 1.1 describes the modeling scheme used to learn the distributions of an R-vine classifier. In this scheme, for each class, an R-vine distribution is learned from samples (data) aiming to get the best fit to the data. However, higher accuracy has a cost, the price that must be paid to build more accurate models is an increase of the cost of the estimation procedure. Learning an R-vine distribution for each class may not only impact the cost of estimation, but, since each class has its own model, also makes it more difficult to identify the most informative features and relevant pairwise dependence patterns of the problem. This is particularly true for problems with a high number of variables. In order to alleviate these disadvantages, we propose two different methods that allow a common R-vine structure for all classes to be learned. This constraint limits the flexibility of the learned distributions and therefore impacts the performance of the classifiers. However, this influence could be mitigated because the copulas corresponding to the vines learned in each class are selected using the respective data. Therefore, the difference in the distributions between classes is captured by these copulas.

Let us describe the proposed methods, namely CS1 and CS2 respectively (CS stands for Common Structure). These methods learn a structure that is shared by all classes, instead of learning a structure for each class. From now on, this last case is called DS method (DS stands for Different Structure). Whereas the DS method executes Algorithm 1.1 for each class independently, CS1 and CS2 implement a modification of this algorithm, since some steps require information coming from all classes.

Let us introduce the necessary notation. Let $k \in \{k_1, \ldots, k_K\}$ be the set of labels, where K is the total number of labels. Indices $l = 1, \ldots, m^k$, $i = 1, \ldots, n$, and $j = 1, \ldots, n-1$ respectively run over the samples tagged with the label k, variables, and trees of a regular vine-copula model.

$\boldsymbol{D}_{\boldsymbol{X}}^k = \{\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_{m^k}^k\}$ denotes an $m^k \times n$ matrix storing the original observations labeled with k, and $\boldsymbol{x}_l^k = \left(x_{l,1}^k, \ldots, x_{l,n}^k\right)$ is an observation of $\boldsymbol{X} = (X_1, \ldots, X_n)$. Similarly, $\boldsymbol{D}_{\boldsymbol{U}}^k = \{\boldsymbol{u}_1^k, \ldots, \boldsymbol{u}_{m^k}^k\}$ denotes an $m^k \times (n-j+1)$ matrix storing the transformed copula data labeled with k, and $\boldsymbol{u}_l^k = \left(u_{l,1}^k, \ldots, u_{l,n-j+1}^k\right)$ is a sample of $\boldsymbol{U}^k = \left(U_1^k, \ldots, U_{n-j+1}^k\right)$.

4.4.2.1 Method CS1

The CS1 method consists of learning K regular vine-copula models with a common structure (using Algorithm 1.1), where each tree T_j is learned from a single dataset denoted as $\boldsymbol{all.D_U} = \left\{\boldsymbol{D}_{\boldsymbol{U}}^1, \ldots, \boldsymbol{D}_{\boldsymbol{U}}^K\right\}$. This dataset is obtained by combining K different datasets $\boldsymbol{D}_{\boldsymbol{U}}^k$, $k \in \{k_1, \ldots, k_K\}$, which contain transformed data (obtained by (1.18)) labeled with k. However, pair-copulas at the j^{th} level of the k^{th} regular vine-copula model are estimated from the corresponding copula data.

This method proceeds as follows. At each level, it creates a matrix $\boldsymbol{W}$ that contains Kendall's tau values computed from $\boldsymbol{all.D_U}$ for all possible pairs of variables (one variable less at each new level). Next, a single MST that maximizes the absolute values of previously computed Kendall's taus is found. Then, pair-copulas are selected individually from the corresponding copula data $\boldsymbol{D}_{\boldsymbol{U}}^k$. This algorithm is executed tree-by-tree until all trees are built and their pair-copulas are estimated. The output of CS1 is K regular vine-copula models that share a common structure. Figure 4.7 shows a diagram describing the steps of the CS1 method.

When CS1 works together with the BIC-based truncation strategy, the best truncation level of the regular vine-copula models can be reached at different levels. Therefore, the question that arises is how to build a common structure for all the models of a classifier. To achieve this goal, we propose the following strategy: Suppose we have two classes, namely c1 and c2, and that for each class the truncation level is reached in the trees T_2 and T_5 respectively. Therefore, the single structure created with CS1 has five trees. Then, in the last two trees of c1 (i.e., T_4 and T_5) only Product copulas are fitted, whereas in the other class, pair-copulas are selected normally from the corresponding copula data.

4.4.2.2 Method CS2

Similar to CS1, the method CS2 learns K R-vine distributions (using the Algorithm 1.1) with a common structure, but in a different way. CS2 consists of learning a MST from a single pairwise weight matrix, denoted as $\boldsymbol{W.sum}$. This matrix is computed as follows: Firstly, for each class k, a matrix of absolute Kendall's taus, denoted as $\boldsymbol{W}^k$, is computed in each level of the R-vine structure from the corresponding copula data $\boldsymbol{D}_{\boldsymbol{U}}^k$. Then, the matrix $\boldsymbol{w.sum}$ is obtained by element-wise addition of all $\boldsymbol{W}^k$ matrices such that $\boldsymbol{W.sum} = \sum_{k=1}^{K} \left|\boldsymbol{W}^k\right|$. Then, for the current level, a single MST is built using $\boldsymbol{W.sum}$ as the weight matrix. For each class, pair-copulas are selected individually from the corresponding copula data $\boldsymbol{D}_{\boldsymbol{U}}^k$. CS2 is executed tree-by-tree, until all trees are built and their pair-copulas estimated. The final R-vine structure is shared by all classes. The output of CS2 is K R-vine distributions that share a common tree structure. Figure 4.8 shows a diagram describing the steps of the CS2 method. Similarly to CS1, CS2 can also work together with the BIC-based truncation strategy to create a common structure.

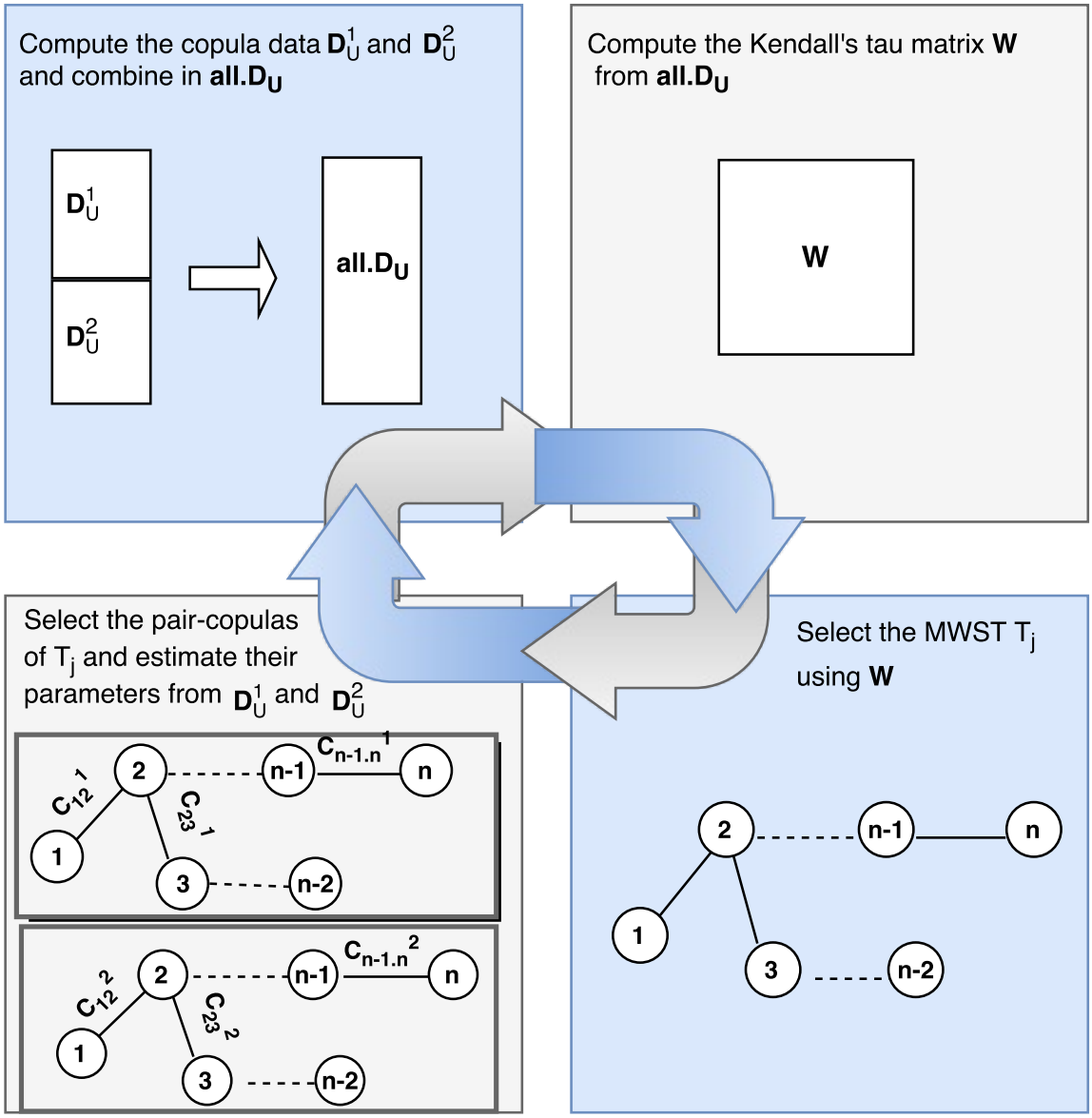


Figure 4.7: Diagram illustrating how CS1 works. It learns a single structure that is shared by all the classes. This method consists of learning K R-vine distributions with a common structure, where each tree T_j is learned from a single dataset (denoted as $\boldsymbol{all.D_U}$). This dataset contains K copula data obtained when the R-vine distribution for each class k is learned.

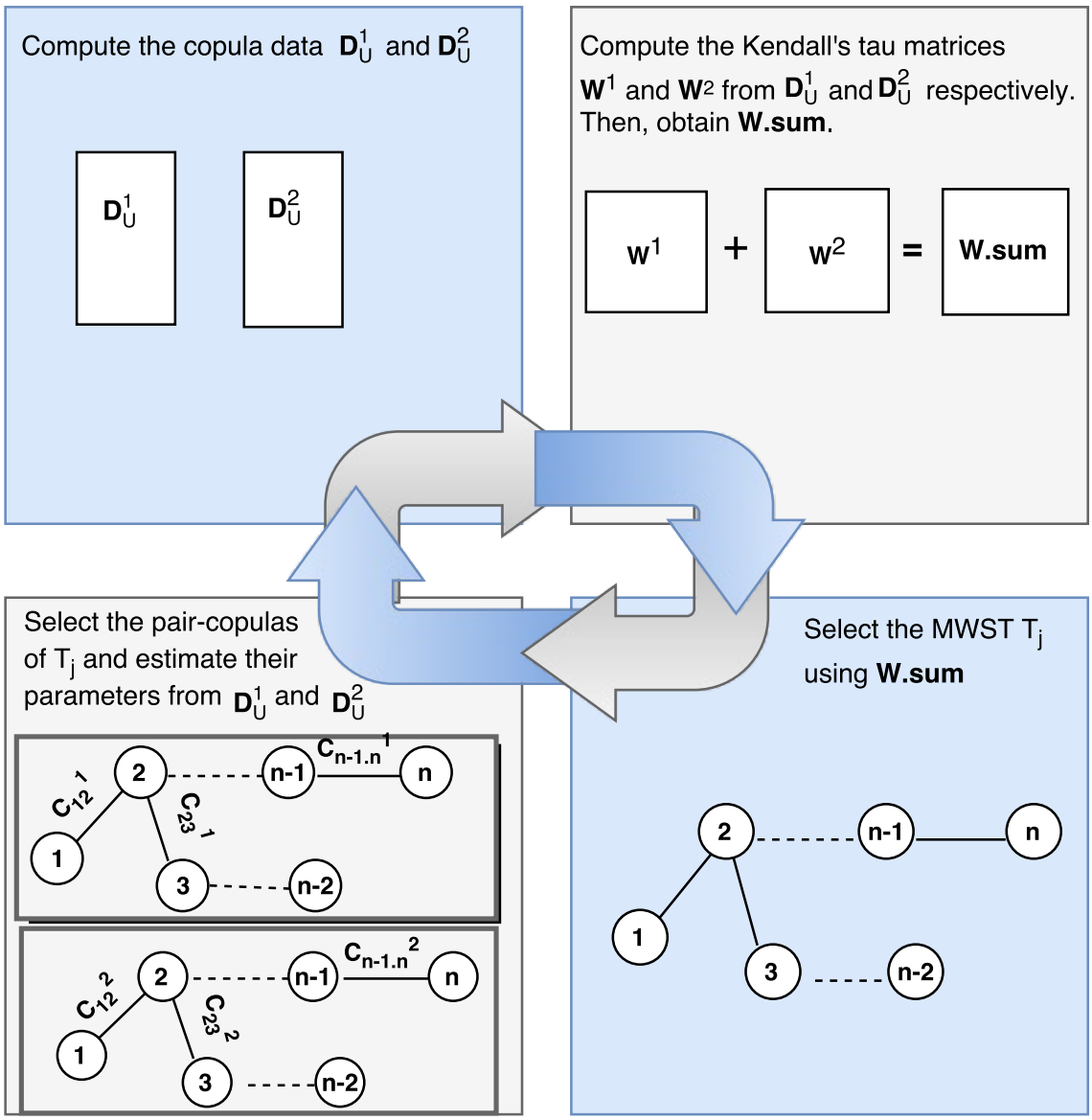


Figure 4.8: Diagram illustrating how CS2 works. In particular, CS2 consists of learning a MST from a single pairwise weight matrix, denoted as $\boldsymbol{W.sum}$. For each class, pair-copulas are selected individually from the corresponding copula data $\boldsymbol{D}_U^k$. The output of CS1 is K R-vine distributions that share a common tree structure.

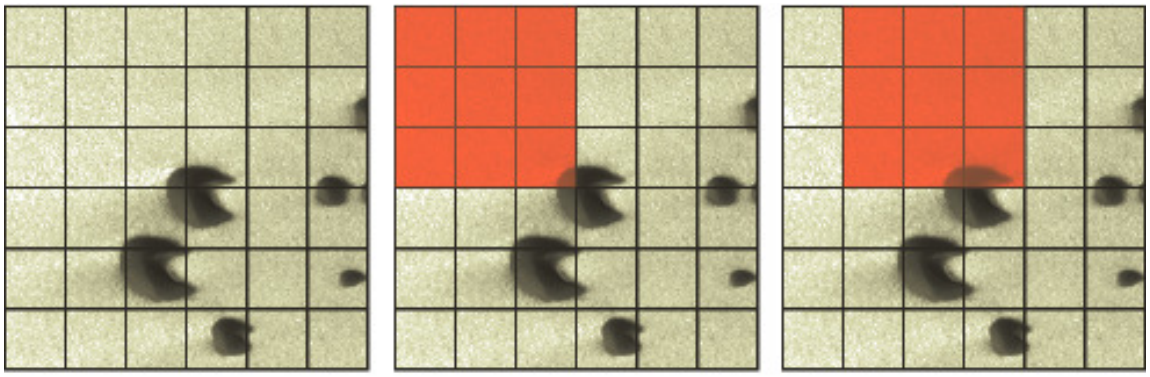


Figure 4.9: Example of the image analysis: Tiling an image in (left) cells of 40×40 pixels and (center) blocks of 3×3 cells; (right) block displacement with overlapping to the right.

4.4.3 Methodology for DCP

The methodology adopted for the automatic detection of sand dunes on images is based on the classification of small square regions of the image called cells. The task of extracting and analyzing local information (image features) is done throughout a regular grid of tiled image. The image is divided into cells with 40×40 pixels from which features are extracted (see Figure 4.9-(left)). The way of computing the vectors of features is explained later in Section 4.4.4.

The size of each cell is the same for all images. To increase the invariance to specific factors such as illumination and shadowing, an aggregation of the local features is performed within larger regions of 3×3 cells, called blocks (i.e., one block has nine cells), which are the detection windows (see Figure 4.9-(center)). To analyze the complete image, this block window is moved along the entire image grid with an overlapping between adjacent blocks equal to one cell side (see Figure 4.9-(right)).

Each cell, represented by a vector of features extracted in the block that contains this cell in the center, is classified as dune or non-dune. The labeling is carried out using the R-vine classifiers proposed in section 4.2.

4.4.4 Feature Extraction

We consider features based on the image gradient $g(x) \in R^2$ computed at each pixel x of the image. The gradient vector is characterized by the magnitude $|g(x)|$ and phase $\phi(x)$. These features are appropriate to detect the patterns presented by sand dunes, since they describe the directional and periodic characteristics of the dunes [4]. In particular, the phase and magnitude histograms try to capture the typical edge structure of the local shape of a dune with a controlled degree of invariance to local geometric and radiometric factors.

The phase histogram associated to the k^{th} cell C^k is given by

$$h_i^k = \sum_{x \in C^k} b_i(\phi(x))$$

where $b_i(\phi) = \begin{cases} 1, & \text{if } \phi \in i^{th} \text{ bin (of the histogram)} \\ 0, & \text{otherwise} \end{cases}$

The magnitude histogram associated to the k^{th} cell C^k is given by

$$\tilde{h}_i^k = \sum_{x \in C^k} \tilde{b}_i \left(|g\left(x\right)| \right)$$

where $\tilde{b}_i \left(|g| \right) = \begin{cases} 1, & \text{if } |g| \in i^{th} \text{ bin (of the histogram)} \\ 0, & \text{otherwise} \end{cases}$

The 180 features computed for each image block, constituting nine cells, result from 81 features for the phase (from 9 bins per cell, using an angular interval of 20°), and 99 features for the magnitude (from 11 bins per cell, using four unit intervals between a minimum of 0 and a maximum of 40).

A normalization step is performed globally for each image and for each individual feature in order to have the features in $[0, 1]$.

4.4.5 Image Database

The image analysis is conducted on two databases with a total of 230 MOC-NA (Mars Orbiter Camera Narrow Angle) images: One database has 160 images representative of the major types of Martian dunes identified in [68] (see Figure 4.6); and the other has 70 images containing other geomorphological structures that can be confounded with dunes, such as channels, crater rims, and textured terrain, among others.

Each original gray-scale image of the first database (see Figure 4.10-(left)) has associated its ground-truth image, which was obtained by manually delineating the contours of the dunes therein contained, indicating the dune and not-dune regions (see Figure 4.10-(center)).

In order to compare the ground-truth with the result produced by the classifier, both images had to be in the same format, such that the ground-truth images were tiled in the same fashion as the original ones (Figure 4.10-(right)). In this image there are three types of cells: dune, in green; non-dune, in yellow; and unclassified, in gray. To assign one of these labels to a cell, the area of the block (from which the cell is the center) that is occupied by ground-truth dune was computed: If this area is higher than 30% of the number of pixels of the block, the cell is considered a dune (in green); if it is less than 10%, the cell is non-dune (in yellow); if the area is between 10 and 30%, the cell is not classified as any decision is considered ambiguous (in gray). In the 230 images, there are a total of 370019 cells, of which 112029 belong to the dune class and 257990 cells to the non-dune class. The ambiguous cells have been removed from the study.

4.4.6 Adding Flexibility to D-vine and R-vine Classifiers

The R-vine classification approaches used to address the DCP are presented as an evolution of D-vine-based approaches applied to the MRP. Indeed, the classifiers we design here use both D-vines and R-vines. Another feature incorporated in the present study is that the number of trees used by the models of a classifier is not fixed in advance, but is determined by the BIC-based model selection strategy presented in Section 1.7.

Next, we describe the individual characteristics of the designed D-vine and R-vine classifiers to approach the DCP.

Unmixed or homogeneous

- R-vine-P-g: This classifier uses only Product (P) copulas and Gaussian (g) marginals. It assumes that the variables are not correlated. Obviously, in this case, the graphical structure is totally irrelevant.

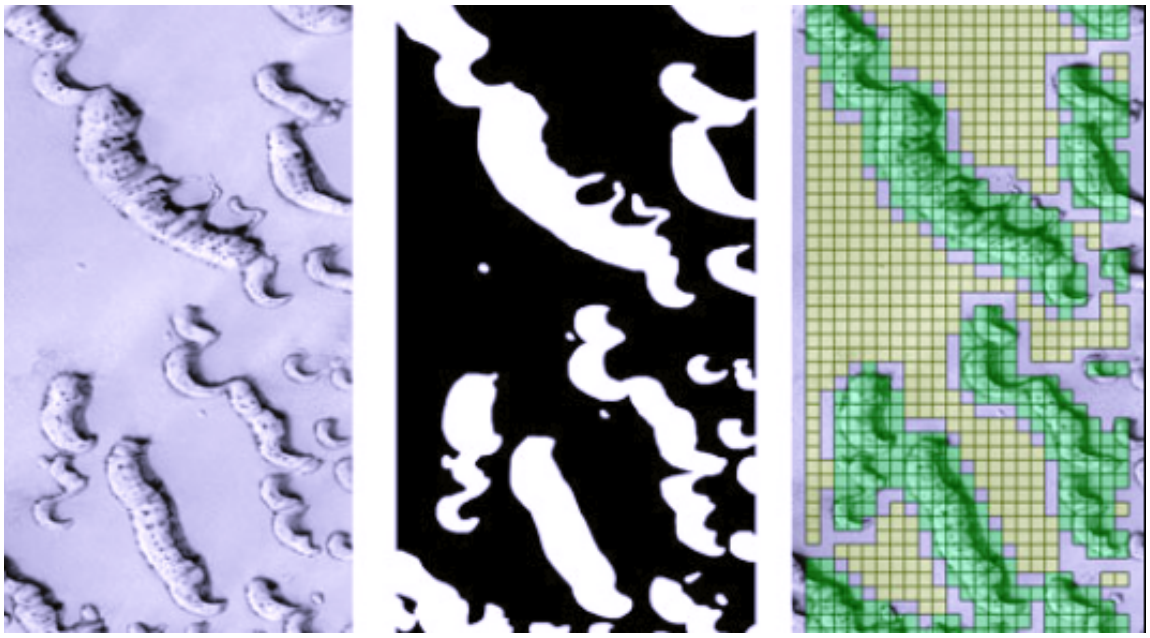


Figure 4.10: Image preparation: (right) original gray-scale image MOC-NA E18-00494; (center) manually drawn binary ground-truth; (right) tiling of the ground-truth in cells, where the dune cells are in green, non-dune cells are in yellow, and gray cells are unclassified. Image credits: NASA/JPL/MSSS. This figure is available in color online at wileyonlinelibrary.com/journal/espl.

- D-vine-t*-t*-N-g and R-vine-t*-t*-N-g: These classifiers build D-vine and R-vine models respectively. They use only bivariate Normal (N) copulas and Gaussian (g) marginal distributions.

Partially-mixed heterogeneous

- D-vine-t*-t*-Sel-g and R-vine-t*-t*-Sel-g: These classifiers build D-vine and R-vine models respectively. They select (Sel) and fit bivariate copulas of different families, whereas all the marginal distributions are Gaussian (g).

Fully-mixed heterogeneous

- D-vine-t*-t*-Sel-sel and R-vine-t*-t*-Sel-sel: These classifiers build D-vine and R vine models respectively. They use bivariate copulas of different families, which are selected (Sel) from a set of candidate copulas. Marginals are also selected (sel) from different distributions.

In the name of the designed classifiers, the first and the second 't*' denote the number of trees of the model learned from the dune and non-dune datasets respectively. The symbol '*' is the mask for the number of trees. Indeed, 't*' is the truncation level of the regular vine-copula model. However, since in a truncated model at the j^{th} level all pair-copulas in the subsequent levels are Product, in practice, these trees are not built. For instance, the classifier name 'R-vine-t2-t5-Sel-g' means that the R-vine models learned from the dune and non-dune datasets have two and five trees respectively. In other words, these models have been truncated at the second and fifth levels respectively. Furthermore, both models select (Sel) and fit pair-copulas from different families, whereas the marginals are all Gaussian (g).

For the selection of pair-copulas, we use the strategy based on the Cramér-von Mises statistic (1.44), which, among the set of candidate copulas, selects the one with the smallest distance to the bivariate empirical copula (see Section 1.6.1).

4.4.7 Experiments

In this section, through numerical experiments, we evaluate the classification approaches presented previously in Section 4.2. Here, the classifiers use both D-vine and R-vine models. We first analyze the performance of several classifiers, which differ from each other in the number of trees, the pair-copula families, and the types of univariate distributions used to model gradient histogram features (see Section 4.4.4). We then analyze how the behavior of classifiers is affected when they use the same structure for the two problem classes (see Section 4.4.2). Furthermore, we carry out comparisons of regular vine-copula classifiers and several state-of-the-art approaches.

4.4.7.1 Experimental Framework

To approach the DCP, we use a greater variety of copula types and marginals compared to that in the MRP, because using a shared structure requires greater diversity of copula families and marginal distributions.

Regarding the copula families, (partially-mixed and fully-mixed) heterogeneous approaches fit Product, Normal, Student's t, Clayton, Gumbel and rotated by 90°, 180° and 270° Clayton and Gumbel pair-copulas. These copulas describe different features of the bivariate dependence (see Section 1.5). Obviously, there is a trade-off between the flexibility provided by the use of different copulas and the computational cost of learning vine-copulas selecting a large set of copula types. For the DCP applications addressed in this thesis, we have assumed that a greater modeling capacity is better even at the expense of a higher computation cost.

Regarding the marginal families, fully-mixed heterogeneous approaches use four parametric distributions, namely Gaussian, Student's t, Beta and Gamma as well as univariate empirical distributions based on Normal kernels (see Equations (1.51) and (1.52).

As explained in Section 4.4.3, the methodology for DCP is based on cells extracted from images of the surface of Mars. The prior probabilities of being a dune and non-dune cell are approximately $0,3$ and $0,7$ respectively.

We conduct a five-fold CV with 30 repetitions. Each fold has around 74003 cells of which around 22405 belong to the dune class and 51598 belong to the non-dune class.

The metrics used to assess the performance of the designed classifiers are both the classification accuracy and area under the ROC curve (AUC) [15, 46, 104]. Moreover, we also provide statistical comparisons via the Kruskal-Wallis and post-hoc Dunn statistical tests [31], which allow us to establish the statistical significance of the results according to AUC.

4.4.7.2 Data Exploration

The assumption that underlies the application of regular vine-copula classifiers to the DCP is that the univariate distributions of the features as well as the patterns of correlation among them are different in both classes. To test this assumption, we explore the statistical properties of the dune and non-dune datasets through visual analytic tools.

To begin with, we analyze graphically the distributions of some of the variables in the two classes. Figure 4.11 shows symmetric violin plots [70] of 6 (out of the 180) features arbitrarily chosen: X_81, X_84, and X_96 belong to the set of magnitude features and X_100, X_109, and X_153 belong to the set of phase features. The violin plot is a Normal kernel density plot that is rotated and placed on each side to show the distribution shape of the data, where the vertical axis represents the values of x and the horizontal axis is the density of x. Values in the wider sections (in the bottom part) of the violin are more probable than those in the narrower sections. From these charts, we observe that none of these variables are normally distributed as well as that the shape of the distribution is different in each class: In X_100, X_109, X_153 and X_96, the non-dune class has wider sections than the dune class. However, in X_81 and X_84 the opposite occurs, the dune class being that which has the widest sections. Moreover, the widest section appears in opposite positions in each class, i.e., the highest probability is on the zero value in the dune class, and on the one value in the non-dune class. A final remark with regard to this topic, is that to properly model non-Gaussian features such as those shown by the variables X_81 and X_96, it could be convenient to use alternative univariate parametric families other than the Gaussian (e.g., Beta, Gamma) as well as the univariate empirical distributions smoothed with Normal kernel functions (see Equations (1.51) and (1.51)).

Next, we graphically assess the shape of the feature dependence in the dune and non-dune sample data in Figures 4.12 and 4.13 respectively. In these figures, the copula data is transformed to have standard Normal margins over the same six features used in the violin charts of Figure 4.11. They show scatter plots along with pairwise Pearson's correlation values above the diagonal and contour plots with standard Normal margins below the diagonal. The bivariate contour plots differ from dispersed and elliptical shapes (which characterize the pairwise independence and linear dependence, respectively) showing the need for different copulas other than the Normal to better model the diversity of types of dependencies presented in the data. Furthermore, these pair plots also reveal that both the strength and the types of dependence for the same pair are quite different in the dune and non-dune sample data.

In summary, the exploratory data analysis performed in this section shows that both the marginal behavior of the gradient features and the dependence patterns between them are different across the dune and non-dune sample data. These findings further justify the use of regular vine-copula models to design probabilistic classifiers.

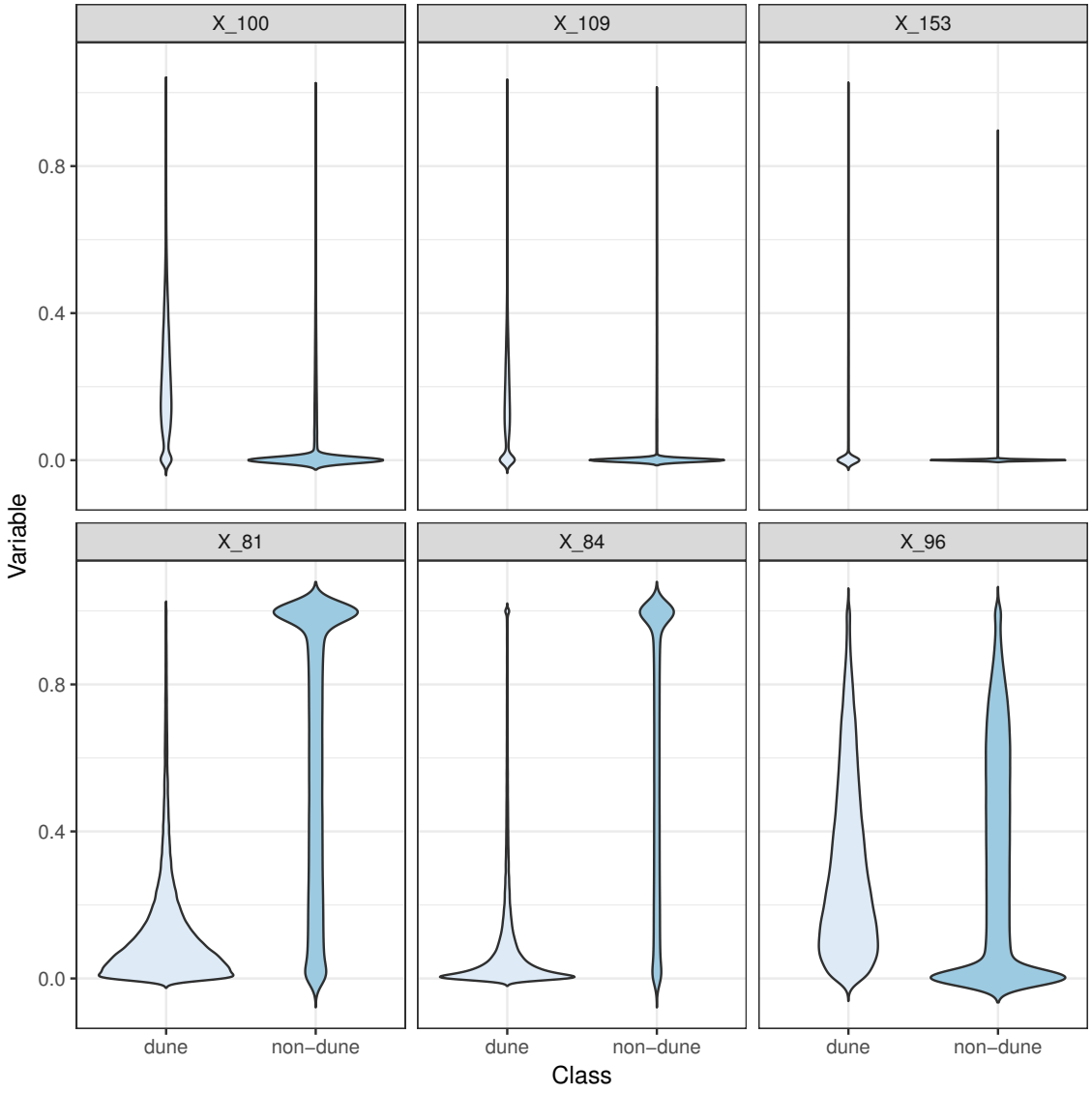


Figure 4.11: Symmetric violin plots with the distribution of 6 (of the 180) features arbitrarily chosen using the data of the respective class: X_81, X_84, and X_96 belong to the set of magnitude features and X_100, X_109, and X_153 belong to the set of phase features. We observe that none of these variables are normally distributed as well as that the shape of the distribution is different in each class: In X_100, X_109, X_153 and X_96, the non-dune class has wider sections than the dune class. However, in X_81 and X_84 the opposite occurs, the dune class being that which has the widest sections. Moreover, the widest section appears in opposite positions in each class, i.e., the highest probability is on the zero value in the dune class, and on the one value in the non-dune class. In summary, the existence of non-Gaussian margins (for instance, X_81 and X_96) justifies the use of empirical margins, which do not assume any specific distributional shape of the variables.

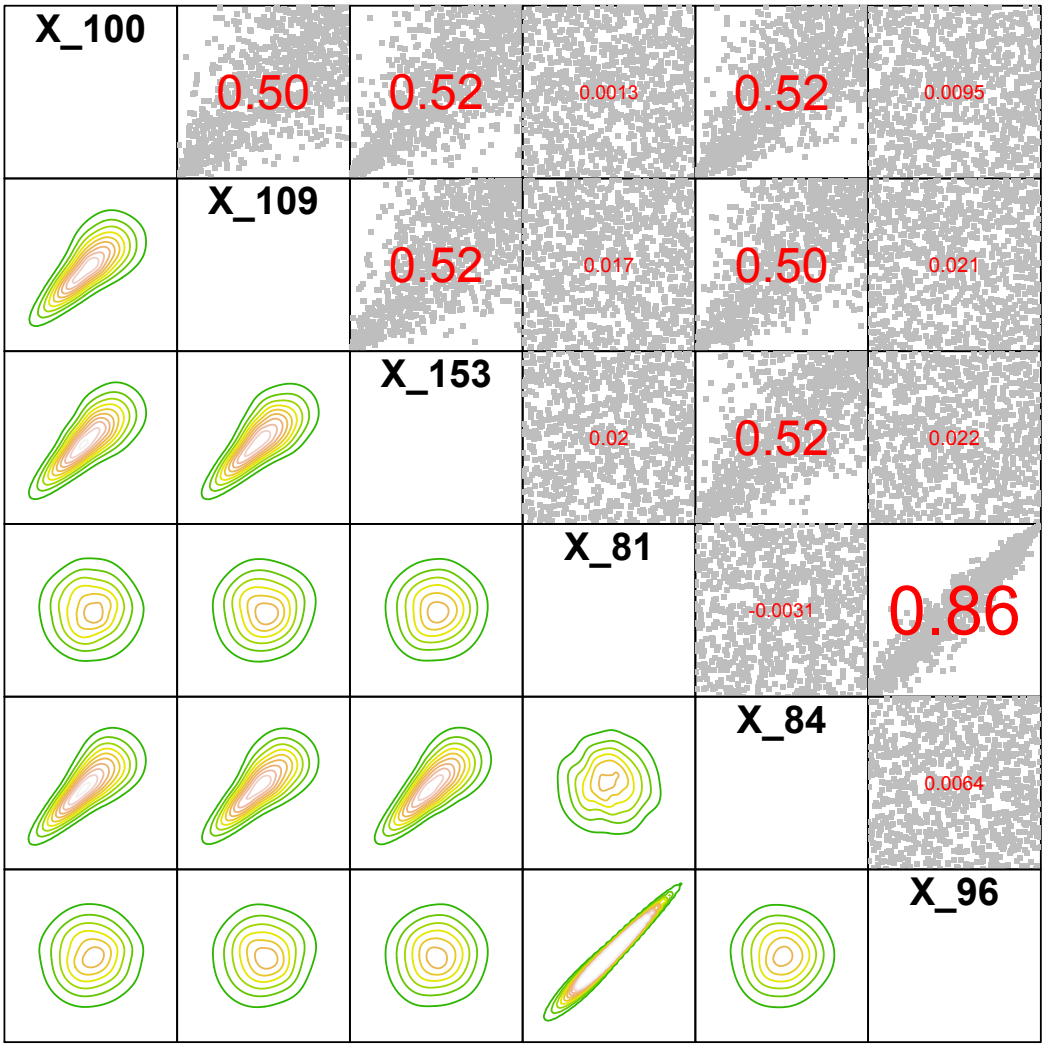


Figure 4.12: Pair plot over 6 (out of the 180) features arbitrarily chosen from the dune sample data with scatter plots above the diagonal and contour plots with standard Normal margins below the diagonal. We observe that the bivariate contour plots differ from dispersed and elliptical shapes (which characterize the pairwise independence and linear dependence, respectively) showing the need for different copulas other than the Normal to better model the diversity of types of dependencies presented in the data.

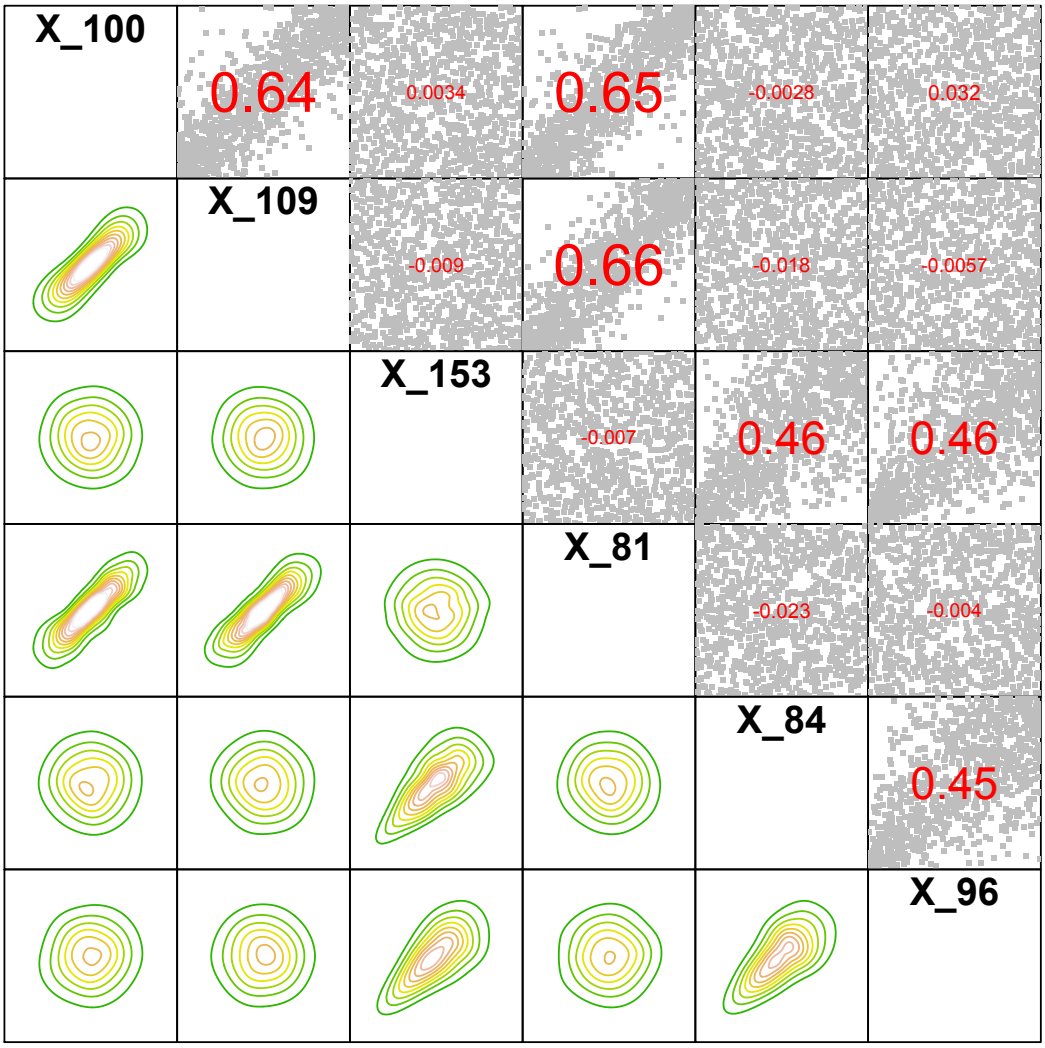


Figure 4.13: Pair plot over 6 (out of the 180) features arbitrarily chosen from the non-dune sample data with scatter plots above the diagonal and contour plots with standard Normal margins below the diagonal. As in the pair plot for the dune data (Figure 4.12), here we observe that the bivariate contour plots differ from dispersed and elliptical shapes (which characterize the pairwise independence and linear dependence, respectively) showing the need for different copulas other than the Normal to better model the diversity of types of dependencies presented in the data.

4.4.7.3 Analysis of D-vine and R-vine Classifiers

In the previous section, we presented evidence that the dune and non-dune datasets are different to each other in terms of the statistical characteristics of both magnitude and phase histogram features as well as the pairwise correlations between them. This evidence supports the working hypothesis of the experiments of this section that the better the vine-copula models capture the distribution of dune and non-dune datasets, the better the predictive ability of the classifiers.

To test the working hypothesis, we compare the regular vine-copula approaches presented in Section 4.4.6. Let us start by analyzing the performance of the unmixed classifiers, namely R-vine-P-g, D-vine-t2-t1-N-g and R-vine-t1-t1-N-g. Accuracy and AUC values obtained by them in the DCP are shown in Table 4.6. The two classifiers that use Normal copulas outperform the one that only uses Product copulas, which does not take into account the dependence structure of the problem. Furthermore, when comparing the heterogeneous classifiers (which combine copulas from different families) with those using exclusively Normal copulas, the former achieve higher accuracy and AUC values than the latter. This finding suggests the presence of asymmetric pairwise dependence structures that are better modeled by the different versions of Gumbel and Clayton copulas used in these experiments.

Now, we analyze the effect of marginal distributions in improving the functioning of (homogeneous and heterogeneous) D-vine and R-vine classifiers (see Table 4.6). We can appreciate that fully-mixed classifiers, which encode marginals of different distributions, outperform the partially-mixed classifiers, which assume that all the features follow a Gaussian distribution. We explain this result through the variable X_84, whose shape is similar to other variables of DCP. Figure 4.14 shows the Beta, Gamma and Normal density curves estimated from the sample data of this variable in order to visually check how these curves resemble the smoothed empirical. For the dune class (left panel), we can appreciate that the Gamma and smoothed empirical curves overlap each other such that they are almost indistinguishable, whereas the Beta and Gaussian distributions provide a very poor fit. For the non-dune class (right panel), the best fit is achieved with the Beta distribution, its curve is the one that most resembles the smoothed empirical curve, whereas the Gamma and Gaussian distributions produce a poor fit.

Summarizing this section, we can say that, among the tested classifiers, R-vine-t5-t7-Sel-sel is not only the most accurate, it is also the most flexible. A look at the differences in the frequencies of marginal and copula families between the R-vine-copulas learned from the dune and non-dune sample data provides an insight into how the different data distributions are captured by the R-vine-copulas of this classifier:

- For the dune class, it chooses (in ascending order) 9 Student's t, 10 Gaussian, 38 smoothed empirical, 48 Beta, and 75 Gamma margins; and 98 Rotated Clayton 90°, 102 Rotated Clayton 270°, 109 Gumbel, 117 Student's t, 135 Normal, 151 Product, and 173 Clayton, pair-copulas, and copulas (there are 885 edges in an R-vine with 5 trees and 180 variables).
- For the non-dune class, it chooses (in ascending order) 12 Student's t, 13 Gaussian, 28 smoothed empirical, 46 Gamma, and 81 Beta margins; and 96 Rotated Clayton 270°, 113 Rotated Clayton 90°, 184 Product, 197 Gumbel, 199 Student's t, 202 Normal, and 241 Clayton pair-copulas (there are 1232 edges in an R-vine with 7 trees and 180 variables).

From here on, the following experiments are performed only for the heterogeneous regular vine-copula classifiers.

4.4.7.4 Using a Common R-vine Structure

The proposed R-vine-based classification strategy requires that the learned R-vine distributions of the classes be different from each other in order that the classifier can distinguish which class a

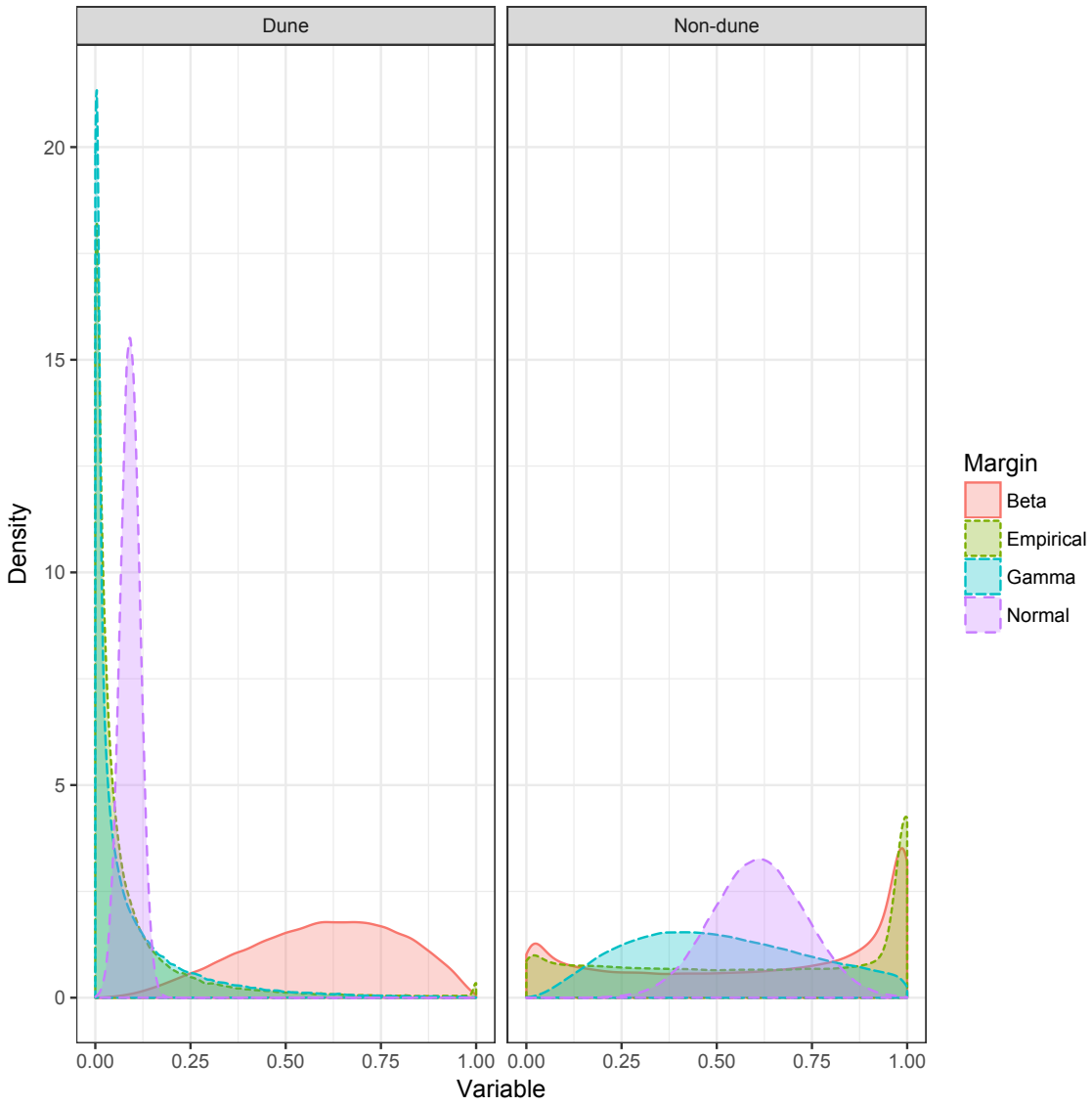


Figure 4.14: Comparison of Beta, Gamma and Gaussian margins with the empirical distribution for the same variable in the dune and non-dune classes.

Table 4.6: Classification accuracy and AUC results in the DCP with (unmixed) homogeneous and (partially-mixed and fully-mixed) heterogeneous D-vine and R-vine-based classifiers learned with the DS method.

Type of Classifier	Classifier	Accuracy	AUC
Unmixed	R-vine-P-g	81,0	76,1
	D-vine-t2-t1-N-g	85,7	86,5
	R-vine-t1-t1-N-g	87,3	90,1
Partially-mixed	D-vine-t4-t2-Sel-g	89,8	91,3
	R-vine-t3-t4-Sel-g	92,6	94,0
Fully-mixed	D-vine-t4-t4-Sel-sel	92,1	92,5
	R-vine-t5-t7-Sel-sel	95,2	98,8

sample belongs to. When using the CS1 and CS2 methods, the linked features are the same in both classes but not the pair-copulas, which are selected and estimated from the corresponding data. We believe that these methods make it easier to interpret and identify which pairwise dependencies are those that contribute to characterize the classes and to assess how the pairwise dependencies change among them.

The price to be paid for learning an R-vine structure per class is that the models can not be directly compared in terms of the selected pair-copulas, because the variables that determine the most important dependence patterns may be different in each class. A classifier with a shared structure is more amenable for identification of the differences between classes. That said, here we investigate how the performance of R-vine classifiers is affected if only a single structure is estimated for all classes. In this approach, only the structure is common for the R-vine distributions of both classes, whereas the pair-copulas and their parameters are selected individually from data of the corresponding class.

Let us begin with the discussion of the results. Table 4.7 presents the accuracy and AUC statistics for the partially-mixed and fully-mixed classifiers learned with the CS1 and CS2 methods (we use the cross validation methodology presented in Section 4.4.7.1). We can appreciate that these results are coherent with those presented in Section 4.4.7.3: The classifiers that combine different types of pair-copulas and margins achieve higher accuracies. However, the most interesting result comes from comparing DS-based classifiers (see Table 4.6) with those learned using CS1 and CS2. Comparing classifiers with the best performance learned with DS, CS1 and CS2, we can see that the accuracy and AUC of the DS-based R-vine-t5-t7-Sel-sel is 95.2% and 98,8%, whereas with CS1-based R-vine-t3-t3-Sel-sel the accuracy decreases around 4,2% and the AUC decreases around 6,4%, and with CS2-based R-vine-t4-t3-Sel-sel the accuracy decreases around 3,6% and the AUC decreases around 5,6% .

Figures 4.15-4.17 account for such behavior through the most accurate R-vine classifiers built with the methods of the three learned strategies. Figure 4.15 displays box plots of Kendall's tau values associated to the edges of the first tree learned with CS1 (left), CS2 (center) and DS (right) for the dune and non-dune classes respectively. Each box plot is drawn from 179 values of Kendall's tau (there are 179 edges in the first tree of an R-vine of 180 variables). It is important to clarify that in the cases of the CS1 and CS2 box plots, the Kendall's tau values are estimated for the edges of the common structure, using, however, the copula data of the respective class. From this figure, we see that the absolute maximum, the third quartile, the median, the first quartile and the minimum of the dune and non-dune box plots for the DS reached higher absolute Kendall's tau values than in the other four box-plots. It is easy to see that R-vine classifiers that learn the structure of each class can more freely accommodate the strongest dependencies than those that use a common structure.

Table 4.7: Classification accuracy and AUC results in the DCP with (partially-mixed and fully-mixed) heterogeneous D-vine and R-vine classifiers using the methods CS1 and CS2.

Method	Type of Classifier	Classifier	Accuracy	AUC
CS1	Partially-mixed	D-vine-t2-t2-Sel-g	85,6	90,2
		R-vine-t2-t3-Sel-g	88,4	91,3
	Fully-mixed	D-vine-t3-t2-Sel-sel	89,9	87,2
		R-vine-t3-t3-Sel-sel	**91,2**	**92,4**
CS2	Partially-mixed	D-vine-t3-t2-Sel-g	87,3	89,3
		R-vine-t3-t2-Sel-g	90,8	89,2
	Fully-mixed	D-vine-t3-t3-Sel-sel	89,2	89,8
		R-vine-t4-t3-Sel-sel	**91,7**	**93,2**

Figure 4.16 confirms this clear trend in favor of the DS method. The x-axis represents the edges belonging to the first tree learned with CS1 (circle), CS2 (triangle) and DS (square). The edges are arranged in descending order according to the absolute Kendall's tau values computed from the copula data of the dune class; the y-axis represents the Kendall's tau value associated to the edges in the x-axis. This figure is made only for the dune class since for the non-dune class the behavior is similar. The DS method has more freedom than CS1 and CS2 to include a greater number of strong dependencies. However, with CS1 and CS2, the strong edges for one class may be out of the tree since they are weak edges for the other class. The restriction of common structure together with that of being a tree prevents the insertion of strong edges that are replaced by weak ones. This behavior leads to an increase in the number of Product copulas fitted, as can be seen in the stack graph of Figure 4.17, which shows the bivariate copula families selected by each method in the first tree for both classes. The DS method selects the smallest number of Product copulas and the largest number of Clayton copulas for both classes.

The use of a single structure can facilitate the interpretation of R-vine classifiers since the set of dependencies explicitly represented is the same for all classes. This means that it is possible to compare, for each edge of the tree, which copula families are learned for this edge for the two classes. When the copula family coincides in the two trees, the strength of the dependence can help to characterize and interpret the differences between classes. In Figure 4.18, we show the copula families assigned to the 20 strongest edges of the first tree found by CS1, CS2, and DS. On one hand, we can see that the DS-based classifier fits a large number of Clayton copulas. In fact, in these edges most of the fitted copulas belong to this family (9 and 11), whereas only 2 Normal copulas are fitted in the dune and non-dune classes respectively. Conversely, the classifiers that share a common structure fit more Normal copulas (12 and 13 with CS1, and 9 and 8 with CS2 in the dune and non-dune classes respectively) than the classifier that uses different structures at the same time being the most accurate.

In summary, although the constraint of a common structure can limit the flexibility of the model and therefore impact the accuracy of the classifiers, the experiments show that the impact is not severe, which could be explained by the fact that R-vine classifiers with a shared structure still keep a high degree of flexibility thanks to the use of copulas from different families.

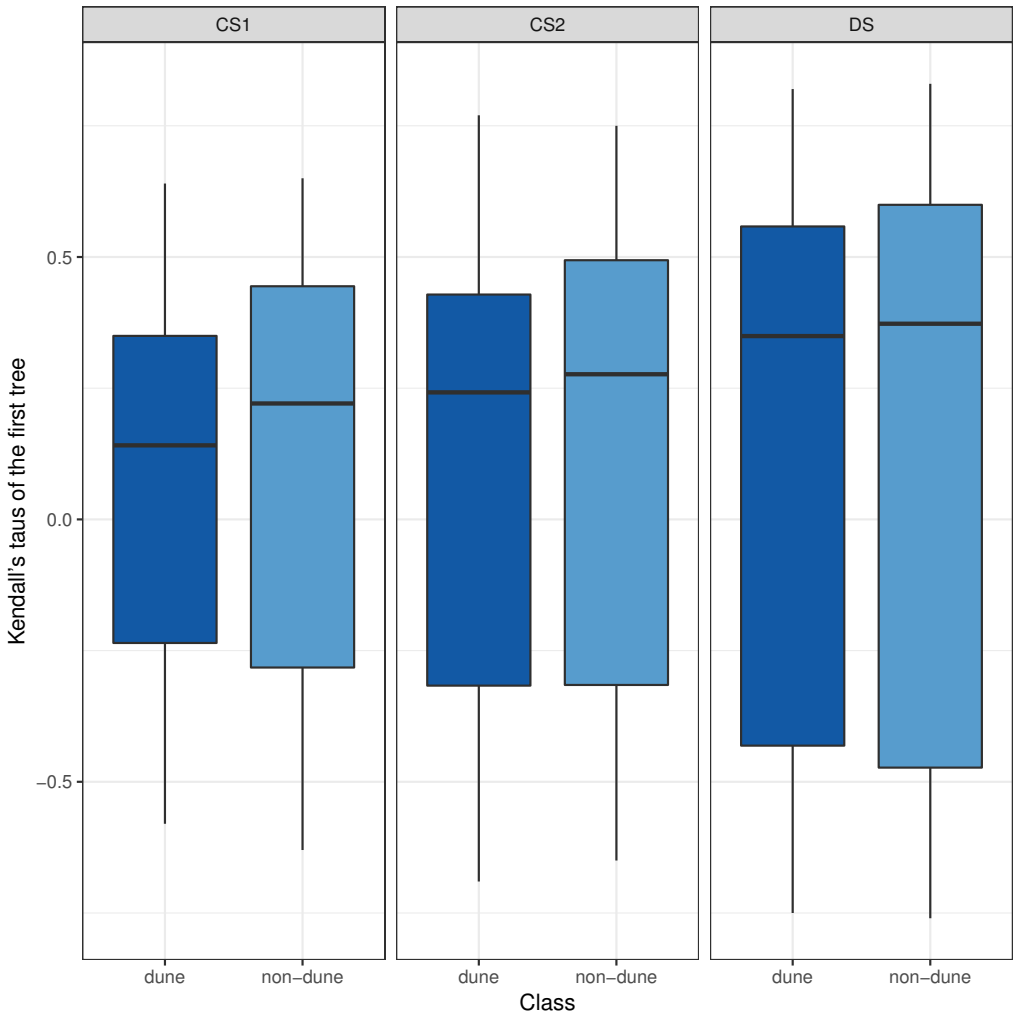
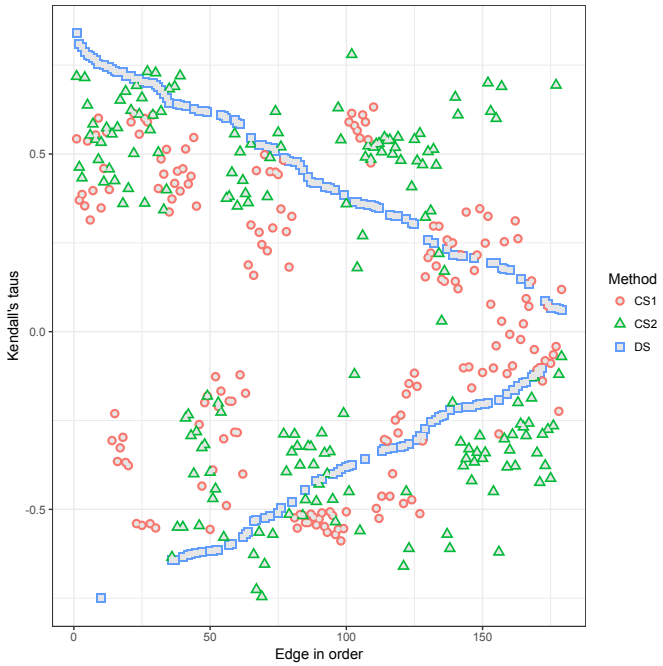
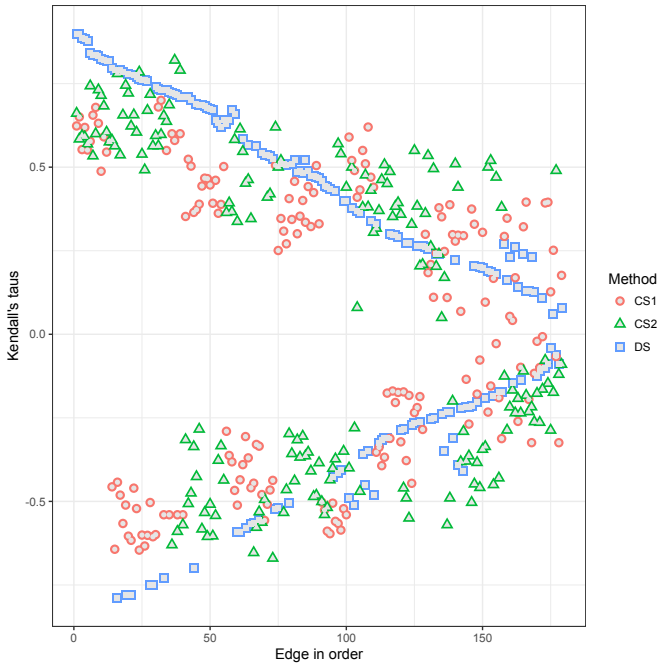


Figure 4.15: Box plots of empirical Kendall's tau values computed from the dune (dark blue) and non-dune sample data (light blue). These values are the weights associated to the edges of the first tree built with the methods CS1 (left), CS2 (center) and DS (right).



(a) Dune class.



(b) Non-dune class.

Figure 4.16: Scatter plots of empirical Kendall's tau values computed from the dune and non-dune sample data. These values are the weights associated to the 179 edges of the first tree built with the methods CS1 (red), CS2 (green) and DS (blue). In the x-axis, the edges are arranged in descending order according to the absolute empirical Kendall's tau values.

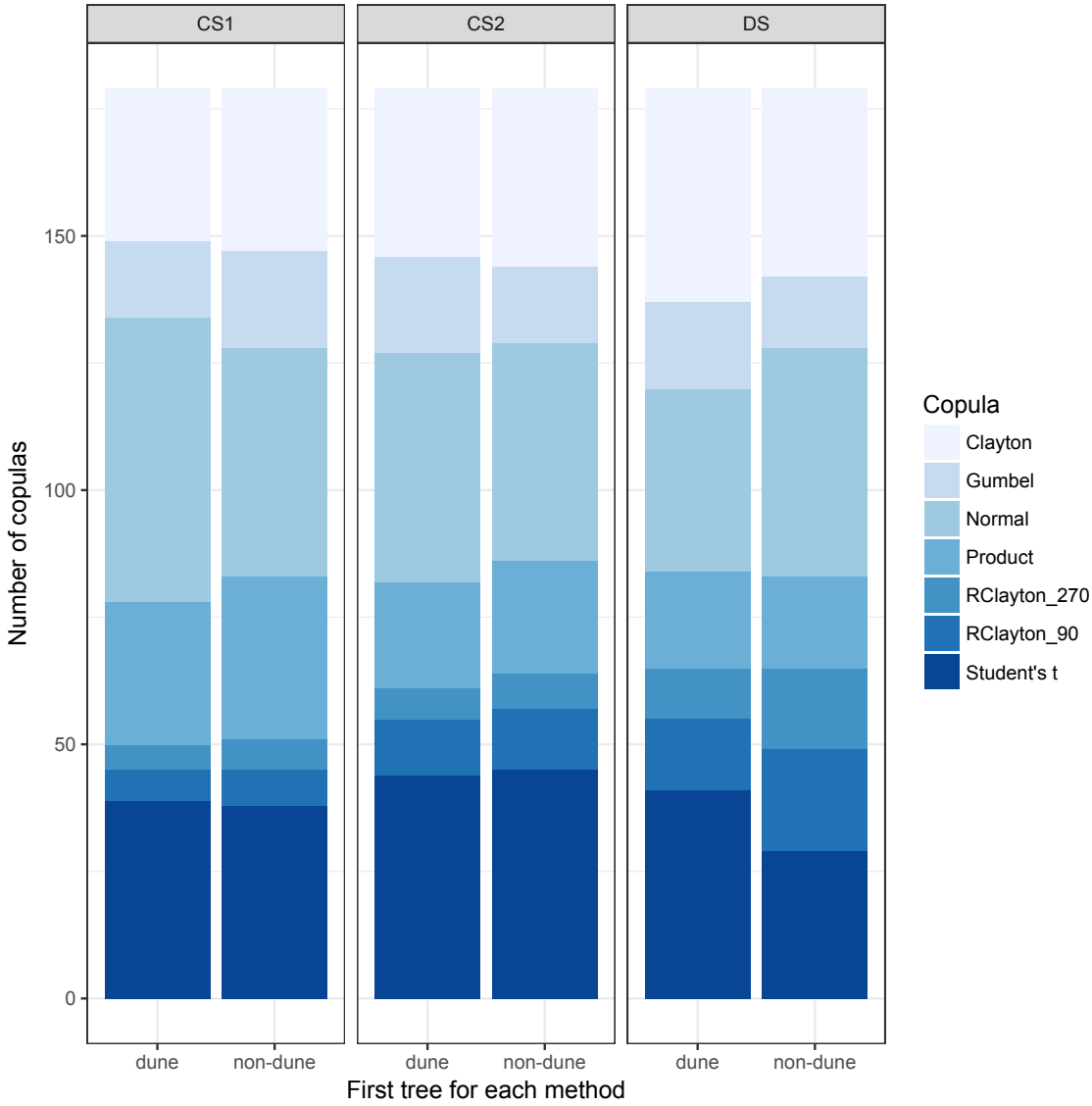


Figure 4.17: Number of Product, Normal, Student's t, Clayton, Gumbel, and rotated (by 90°, 180° and 270°) Clayton and Gumbel copulas fitted in the first tree built with the methods CS1, CS2, and DS. These copulas are estimated from the dune and non-dune sample data.

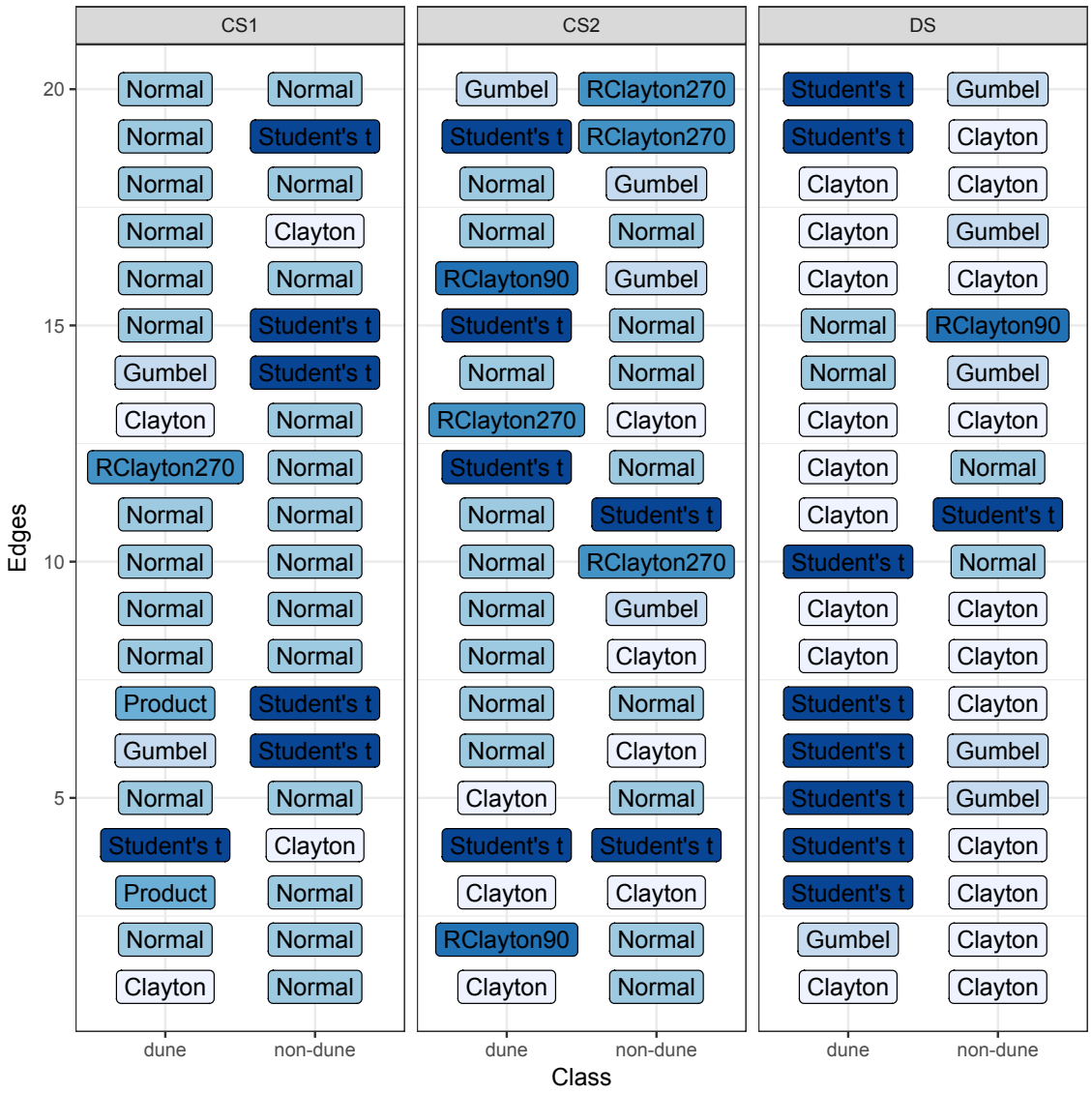


Figure 4.18: Types of copulas fitted in the 20 strongest edges of the first tree built with the methods CS1, CS2, and DS for the dune and non-dune classes.

4.4.7.5 Comparison with Other Algorithms

We assess the performance of D-vine and R-vine classifiers in comparison with 10 supervised learning classification algorithms. We here use the '`scikitlearn`'[1], a widely used ML library in Python. They are the following:

- GB - Gradient Boosting [51].
- RF - Random Forest [20].
- SVM - Support Vector Machines [103].
- LR - Logistic Regression [133].
- LDA - Linear Discriminant Analysis [49].
- EDT - Extra Decision Trees [57].
- KNN - K Nearest Neighbors [10].
- NN - Multilayer Perceptron Neural Network [69].
- GNB - Gaussian Naive Bayes [134].
- DT - Decision Tree [21].

These algorithms cover commonly applied approaches to classification tasks including linear and non-linear classifiers, tree-based classifiers, ensemble classifiers, and distance-based classifiers. Some of these algorithms are able to capture non-linear associations between the variables, while others incorporate regularization techniques. For more information on these algorithms, see [67].

For the optimization of the hyperparameters of each classifier, we first split the set of hyperparameters in two groups, those that have a strong influence on the results of the algorithm and those with a low relevance according to the suggestions given in [100]. Then, for each algorithm we perform a grid search to optimize the first group of hyperparameters using the cross validation methodology presented in Section 4.4.7.1. The second group of hyperparameters (those with low relevance) were set to the default values in the scikit-learn implementation of the algorithms. The optimized hyperparameters and the corresponding best values obtained via grid search are shown in Table 4.8.

For comparison purposes, we focus on the most flexible variants of D-vine and R-vine classifiers built with the DS, CS1, and CS2 methods, namely R-vine-t5-t7-Sel-sel (with DS), R-vine-t3-t3-Sel-sel (with CS1), and R-vine-t4-Sel-sel (with CS2). We renamed them, in short, R-vine-DS, R-vine-CS1, and R-vine-CS2 respectively. The same notation applies to D-vine-t3-t3-Sel-sel or D-vine-DS in short.

Numerical comparisons according to the accuracy and AUC are given in Table 4.9 (in increasing order according to AUC). In order to assess the statistical significance of the observed differences in algorithm performance, we use the Kruskal-Wallis statistical test on the AUC's values to determine whether all the groups originate from the same distribution. If the null hypothesis is rejected (p-value$< 0,05$), a post-hoc test is applied to all the sample data pairs, looking for differences between them. The results of the Dunn test used for the pairwise comparison of 14 algorithms are shown in Figure 4.19. Vertical lines stand for the algorithms. Horizontal lines mean that there are no statistical significant differences among algorithms that cut. On the contrary, the differences among

[1] https://github.com/scikit-learn/scikit-learn [102]

Table 4.8: Classification algorithms, the optimized hyperparameters and their best values obtained via grid search. Both the method names (in the first column) and the hyperparameter names (in the second column) correspond to their implementations in the 'scikitlearn' library.

Algorithm	Hyperparameter	Best Values
GB	`n_estimators`: Number of decision trees.	500
`GradientBoostingClassifier()`	`learning_rate`: Shrinks the contribution of each tree.loss	0.1
	`loss`: Loss function to be optimized.	`'deviance'`
	`max_depth`: Maximum depth of the individual trees.	11
	`max_features`: Number of features to consider when looking for the best split.	`'log2'`
RF	`n_estimators`: Number of trees in the forest.	500
`RandomForestClassifier()`	`criterion`: Function to measure the quality of a split.	`'entropy'`
	`max_depth`: Maximum depth of the individual trees.	11
	`max_features`: Number of features to consider when looking for the best split.	`'auto'`
SVM	`C`: Penalty parameter of the error term (regularization).	1.0
`SVC()`	`kernel`: Kernel type	`'rbf'`
	`gamma`: Kernel coefficient for the `'rbf'`, `'poly'` and `'sigmoid'` `'sigmoid'` kernels.	0.1
	`degree`: Degree for the `'poly'` kernel.	3
	`coef0`: Independent term for the `'poly'` and `'sigmoid'` kernels.	10
LR	`penalty`: L1 or L2 penalization.	`'l2'`
`LogisticRegression()`	`C`: Loss function to be optimized.	1.5
	`fit_intercept`: If the intercept is added to the decision function.	`True`
LDA	`solver`: Solver to use.	`'svd'`
`LinearDiscriminantAnalysis()`	`shrinkage`: Shrinkage parameter.	`None`
EDT	`n_estimators`: Number of trees in the forest.	1000
`ExtraTreesClassifier()`	`criterion`: Function to measure the quality of a split.	'entropy'
	`max_depth`: Maximun depth of the individual trees.	11
	`max_features`: Number of features to consider when looking for the best split.	'log2'
KNN	`n_neighbors`: Number of neighbors to use.	15
`KNeighborsClassifier()`	`weights`: Function to weight the neighbors' votes.	`'distance'`
NN	`hidden_layer_sizes`: The i^{th} element of the tuple represents the number of neurons in the i^{th} hidden layer.	$(50,)$
`MLPClassifier()`	`activation`: Activation function for the hidden layer.	'logistic'
GNB	No parameters.	
`GaussianNB()`		
DT	`criterion`: Function to measure the quality of a split.	'entropy'
`DecisionTreeClassifier()`	`max_depth`: Maximun depth of the tree.	11
	`max_features`: Number of features to consider when looking for the best split.	'none'

Table 4.9: Numerical comparisons of R-vine classifiers with other approaches with respect to the accuracy and AUC in the DCP. The compared R-vine classifiers belong to the fully-mixed group and are learned with the DS, CS1 and CS2 methods. Classifiers are ranked in descending order according to AUC.

Rank	Classifier	Accuracy	AUC
1	R-vine-DS	$95,2$	$98,8$
2	R-vine-CS2	$92,1$	$92,5$
3	D-vine-DS	$91,7$	$93,2$
4	R-vine-CS1	$91,2$	$92,4$
5	GB	$90,7$	$91,7$
6	RF	$90,0$	$91,0$
7	SVM	$89,0$	$90,2$
8	LR	$90,2$	$90,1$
9	LDA	$90,4$	$89,8$
10	EDT	$90,6$	$89,2$
11	KNN	89.4	$88,3$
12	NN	$88,7$	$85,3$
13	GNB	$87,8$	$81,2$
14	DT	$89,2$	$79,1$

algorithms are statistically significant if there is no horizontal line that cuts the vertical algorithm line.

Among all compared algorithms, R-vine-DS reaches the highest accuracy ($96,4\%$) and AUC ($98,8\%$), which is a remarkable performance. It is followed closely, first, by the other regular vine-copula classifiers, namely R-vine-CS2, D-vine-DS, and R-vine-CS1. We notice that these classifiers are the ones that obtain the highest AUC. These results are statistically significant in relation to the particular instances of algorithms tested in the DCP. It is also remarkable that classifiers based on common structure strategies, CS1 and CS2, are in the top positions.

A final remark is that these results confirm that the methodology based on gradient histograms, combined with ML algorithms, is a good approach to deal with the DCP. These features adequately describe the characteristics of the dune and non-dune images, allowing the algorithms to discriminate between the two classes.

4.5 Summary

We have introduced a classification approach where the dependence structure of the problem is modeled through R-vines. In this scheme, a vine-copula model is built for each class from the given samples. Then, a new instance is assigned to the class with the highest probability among the learned models.

We have recognized three types of classifiers according to their complexity: A simpler (homogeneous) one that uses a single family of pair-copulas and marginals and where the number of trees is the same for all the R-vines that comprise the classifier. A more general (partially-heterogeneous) scenario allows the use of pair-copulas and marginals from different families, and this feature defines the classifiers of medium complexity. The ideal (fully heterogeneous) scenario is that in which the models combine different types of pair-copulas and marginals, and the required number of trees is identified by means of statistical model selection strategies

The performance of R-vines as classifiers has been experimentally validated in a mental decoding

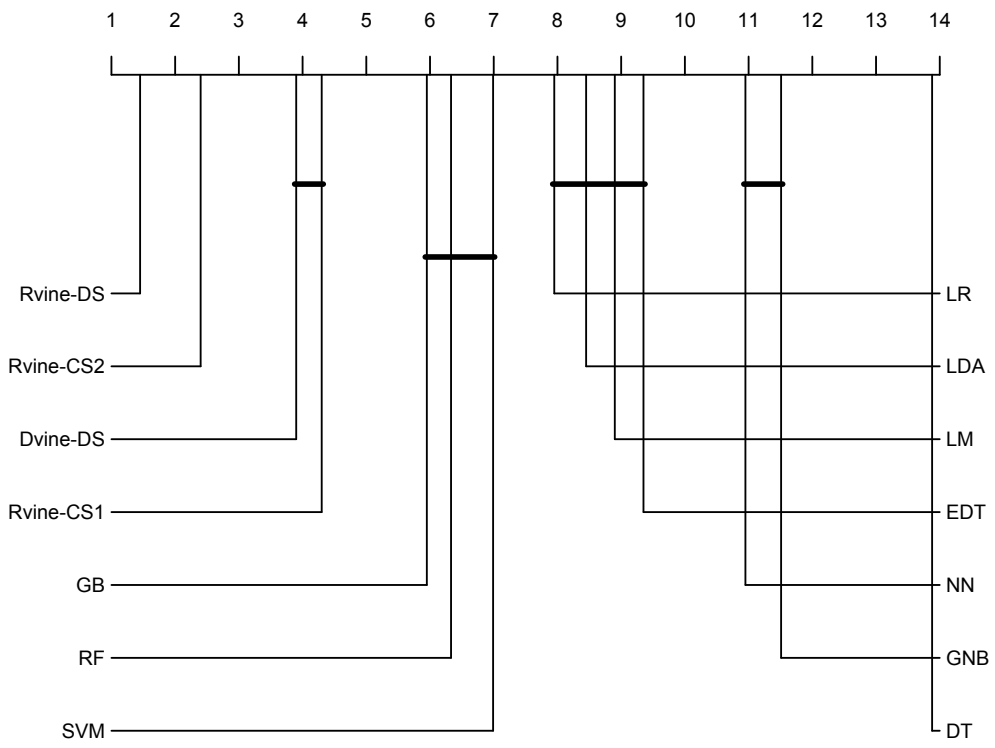


Figure 4.19: Statistical comparisons of tested classification approaches based on the Kruskal-Wallis and post-hoc Dunn statistical tests according to AUC.

problem and in an image recognition task. In the former, we found out that, although brain signals have a diverse and complex dependence structure, the designed D-vine classifiers are able to discover them. Their ability to accurately decode brain signals, even under the effect of covariate shift, underpins the feasibility of the proposed approach.

The representation capability of D-vine classifiers is extended by mean of R-vines in the dune classification problem. To aid the interpretation of R-vine classifiers, two methods that allow the construction of a common graphical structure for all the classes of this image recognition problem have been designed and evaluated. Although this strategy may prevents the representation of important dependencies, the designed classifiers are still effective, since the use of different copula families (which are learned from the corresponding dataset), can compensate for the structural constraints. In effect, the numerical results show that the regular vine-based classifiers with a common structure outperform the traditionally applied classifiers in the dune classification problem, which shows that, despite the incorporation of constraints in their structure, they remain competitive.

In addition, experimental results confirm that the designed classifiers are robust across a variety of scenarios. They further reveal that these models nicely capture the difference among the distributions learned from sample data of different classes as well as the fact that the better the distribution of each class is approximated, the more accurate the classification is. Moreover, the proposed regular vine-copula classification approach can successfully deal with high-dimensional multi-class problems.

Conclusions

This research aimed to investigate the theoretical properties of R-vines for representing dependencies and to extend the use of these models to solve supervised classification problems. Towards achieving this goal, we first have introduced a graphical separation criterion for R-vines with improved graphical expressiveness, and studied the relationship between the graphical representations of R-vines and polytrees. Thereafter, we have presented approaches for learning R-vine structures that incorporate the largest number of dependencies given in a list. Furthermore, a classification approach, where the dependence of the features is modeled through R-vines, has been introduced and applied in MRP and DCP. Hereafter, we summarize the contributions made during this dissertation in further detail.

First, a graphical separation criterion for R-vines, called R-separation, has been defined. The proposed criterion facilitates the enumeration of (non-)separation relationships encoded in the R-vine graph with enhanced expressiveness by examining its topology and the edge types. The derived graphical relationships correspond to (un)conditional pairwise (in)dependence relationships in the associated R-vine copula. Moreover, from the R-separation criterion, a theorem of R-vine dependence maps is enunciated and it has been proved that every R-vine graph is an I-map and a D-map of the associated R-vine copula, but not necessarily a P-map, since from the R-vine graph, it is not possible to infer (non)separations other than those represented by its edges. Summarizing, R-vines do not allow to define a graphical separation concept that yields a complete independence map. We further analyzed different R-separation properties. Findings on this concept include the following: (i) It satisfies symmetry; (ii) it does not satisfy strong transitivity, weak transitivity nor strong union; (iii) weak union, decomposition, contraction and intersection cannot be verified, since R-vine graphs represent pairwise relationships only, and these properties involve sets of indices as the conditioned set.

Furthermore, the relationship between graphical representations of R-vines and polytrees has been analyzed. The focus has been on pairwise separations and non-separations encoded in one graph that correspond with the set of (un)conditional pairwise (in)dependencies of the dependence model associated with the other graph. For this purpose, two algorithms have been designed: One algorithm that aims to induce the R-vine graph that encodes as many relationships as possible derived from the polytree graph. The other algorithm achieves the same goal but in reverse, from the R-vine graph to the polytree graph. The former algorithm is capable of building an R-vine graph that encodes all separation and non-separation relationships derived from the starting polytree graph. Therefore, the R-vine graph is both an I-map and a D-map of the dependence model associated with the starting polytree graph. The other algorithm can produce the polytree graph that represents all separations derived from the starting R-vine graph, but not all the non-separations. In addition, graphical properties that favor the generation of multiple polytree graphs, representing the same set of separations and non-separations, have been identified. However, since the built polytree graph can represent additional non-coded relationships in the starting R-vine graph, the built polytree is neither an I-map nor a D-map of the dependence model associated with

the starting R-vine graph.

With the aim of designing methods for learning the graph structure of R-vines from dependence lists, two approaches have been proposed. The first approach is a 0-1 linear programming formulation for building truncated R-vine graphs with only two trees. The second approach consists of a GA, which is able to learn complete and truncated R-vine graphs. A further distinctive feature of the proposed evolutionary approach is that it uses crossover and mutation operators specifically designed to ensure that the generated R-vine graphs are feasible. The designed operators are effective in generating valid and good solutions when working with both feasible and unfeasible dependence lists. Furthermore, this method further fosters a synergy between global and local optimization mechanisms in the sense that, while the maximum spanning tree method produces locally optimized trees, the engineered genetic operators modify those trees in such a way as to generate better global solutions. Experimental results endorse the success of the designed GA in finding R-vine graphs that incorporate the largest number of dependence relationships given in a dependence list. They reveal that although the GA does not guarantee optimal solutions, it is highly effective in producing optimal or near optimal solutions.

Aiming to extend the use of R-vines to solve supervised classification tasks, this thesis presents an approach where the feature dependence structure is modeled by means of pair-copulas through D-vines and R-vines. The effectiveness of R-vines as classifiers has been experimentally validated in a mental decoding problem and in an image recognition task. In the former, the numerical simulations show that the D-vine classification approach has a competitive performance compared to the four best classification methods presented at the Mind Reading Challenge Competition 2011. The obtained results also suggest that the proposed regular vine-copula classification approach is able to successfully deal with high-dimensional multi-class problems.

To aid the interpretation of R-vine based classifiers, methods that allow the construction of a common graphical structure for all the classes of the dune classification problem have been designed and evaluated. Although this strategy may restrict the flexibility of the learned distributions and, therefore, impact the performance of the classifiers, these remain competitive, since the use of different copula families (which are learned from the corresponding dataset), can compensate for the structural constraints. One could anticipate that the use of regular vine-copula classifiers that use a common structure for all classes is more advantageous in multi-class classification problems since, at each level of the R-vine, the algorithm only has to build one maximum spanning tree instead of having to do so for as many trees as number of classes. Moreover, the proposed regular vine-copula classification approach can produce classifiers that balance their accuracy and complexity when dealing with a wide variety of feature distributions and dependence patterns.

Experimental results confirm that the designed classifiers are robust across a variety of scenarios, especially those where the learned R-vines combine different types of pair-copulas and marginals, and the number of trees is identified by means of statistical model selection strategies. They further reveal that these models nicely capture the difference among the distributions learned from sample data of different classes as well as the fact that the better the distribution of each class is approximated, the more accurate the classification is.

To better meet the outlined aims, two `R` libraries, called '`rvclass`'[2] and '`VinecopulaedasExtra`'[3], have been designed and implemented. These libraries extend the functionalities of '`vines`'[4] and '`copulaedas`'[5], two `R` packages widely used in the numerical experiments of this research. In the comparisons with different existing algorithms, the '`scikitlearn`' Python library was used.

Future research directions derived from this dissertation are presented thereafter. As discussed earlier, this dissertation establishes a graphical separation criterion for R-vines. An ongoing

[2] https://github.com/DianaCarrera/rvclass
[3] https://github.com/DianaCarrera/VinecopulaedasExtra
[4] https://CRAN.R-project.org/package=vines [59]
[5] https://CRAN.R-project.org/package=copulaedas [58]

topic demanding future work is to extend the study on the connection between the graphical representations of R-vines and more general Bayesian networks by using the respective graphical separation criterion.

Regarding the learning of graphical structure of R-vines from dependence lists, a question that arises here is how the number of relationships in the dependence list as well as their distribution at each level of the R-vine graph influence the complexity of the optimization problem posed. In particular, how does the search space reduce with respect to the number of R-vines as the information drawn from the dependence list increases. It is also interesting to conduct an experimental study of the 0-1 linear programming approach proposed in this thesis in order to assess its effectiveness in finding R-vine graphs that incorporate the largest number of dependence relationships given in a dependence list.

Although we have focused on the straightforward application of vine-copulas to the MRP and DCP, there are several research lines that could be investigated to further assess the potential of vine copulas as classifiers. Within this realm of interest, a general question arises on how to use a priori information about the problem to learn the vine-copulas more accurately. For example, whether the information about the variables (groups, types, etc.) in the classification problem could be translated into effective constraints (e.g., fewer dependencies to search) when constructing the vines in order to reduce the learning complexity of the model at an equal or better precision. Another possibility is to consider the characteristics of the copula models that are used when performing the feature selection. Specifically, to not only consider the discriminatory power of the features, but also the goodness-of-fit test of pair-copulas.

Another question left unanswered is how to modify the R-vine learning algorithms in order to mitigate the impact of the covariate shift in classification accuracy. The rationale behind CS1 and CS2 is that one can determine and easily evaluate how R-vine distributions with the same structure modify the types of copulas and their parameters when adapting to class distributions. A future research line here would be to determine further strategies on how to modify the learning methods in order to increase the interpretability of the problem with at least the same accuracy. Future investigation can address common structure-based strategies in multi-class classification problems.

We have modeled pairwise dependencies with parametric bivariate copulas that describe a wide range of dependence structures of sample data. Worthy of comprehensive testing are the (smoothed) empirical copulas to model complex forms of dependence that cannot be captured by any parametric copula. We believe that a more accurate modeling of the pair-copulas may lead to an increase in the predictive ability of R-vine classifiers.

Publications

The research work carried out during this thesis has produced the following publications:

- D. Carrera, R. Santana, and J. A. Lozano. Learning regular vine-copulas from a dependence list. *Swarm and Evolutionary Computation*, 2021. (Under review)
- D. Carrera, L. Bandeira, R. Santana, and J. A. Lozano. Detection of sand dunes on Mars using a regular vine-based classification approach. *Knowledge-Based Systems*, 163:858-874, 2019.
- D. Carrera, R. Santana, and J. A. Lozano. The relationship between graphical representations of regular vine-copulas and polytrees. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 678-690. Springer, 2018.
- D. Carrera, R. Santana, and J. A. Lozano. Vine-copula classifiers for the mind reading problem. *Progress in Artificial Intelligence*, 5(4):289-305, 2016.

`R` packages published on GitHub:

- D. Carrera, R. Santana, and J. A. Lozano. rvclass: Implementation of supervised classification algorithms based on regular vine-copulas, 2018. *https://github.com/DianaCarrera/rvclass*
- D. Carrera, R. Santana, and J. A. Lozano. VinecopulaedasExtra: Implementation of extra-functions for copulaedas and rvclass, 2018. *https://github.com/DianaCarrera/VinecopulaedasExtra*

Appendix: A Simple Illustration of the TDSH

We illustrate a simplified version of the TDSH (Algorithm 1.1) using a four-variable example, where $\boldsymbol{X} = (X_1, X_2, X_3, X_4)$.

1. Estimate the univariate cumulative and density functions $F_i(X_i)$ and $f(X_i)$ from the original observations $\boldsymbol{D_X}$ defined over $\boldsymbol{X}$, such that we have

$$\begin{array}{rcl} X_1 & \sim & F_1, \\ X_2 & \sim & F_2, \\ X_3 & \sim & F_3, \\ X_4 & \sim & F_4. \end{array}$$

2. Obtain the transformed observations $\boldsymbol{D_U}$ by evaluating the unconditional distribution functions $F_i(X_i)$ estimated at Step 1, such that we have

$$\begin{array}{rcl} u_1 & := & F_1(x_1), \\ u_2 & := & F_2(x_2), \\ u_3 & := & F_3(x_3), \\ u_4 & := & F_4(x_4). \end{array}$$

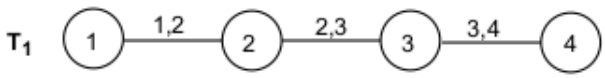
3. Compute the Kendall's tau value for each pair of variables from a set transformed observations $\boldsymbol{D_U}$ defined over $\boldsymbol{U} = (U_1, U_2, U_3, U_4)$ obtained at Step 2, such that we have

	u_1	u_2	u_3	u_4
u_1		$0,8$	$0,4$	$0,2$
u_2			$0,7$	$0,5$
u_3				$0,6$
u_4				

4. Build the MST T_1 that maximizes the sum of the absolute Kendall's tau values obtained at Step 3. Then, from a predefined group of candidate copula families, select the copula with the smallest Cramér-von Mises statistic (1.44). Afterwards, compute the parameters of the selected copulas using the relationship between Kendall's tau and the dependence parameter of the corresponding bivariate copula (see Table 1.1). Three unconditional pair-copulas, namely

$$\begin{array}{l} c_{1,2}(u_1, u_2), \\ c_{2,3}(u_2, u_3), \\ c_{3,4}(u_3, u_4), \end{array}$$

are associated to the edges of the first tree as shown in the following figure:



5. Obtain conditional copula data $\boldsymbol{D_U}$ (needed for T_2) by evaluating conditional distribution functions using the bivariate copulas fitted in T_1, such that we have

$$\begin{aligned} F\left(x_1 \left| x_2 \right.\right) &= \frac{\partial C_{1,2}(F(x_1),F(x_2))}{\partial F(x_2)} = \frac{\partial C_{1,2}(u_1,u_2)}{\partial u_2}, \\ F\left(x_2 \left| x_3 \right.\right) &= \frac{\partial C_{2,3}(F(x_2),F(x_3))}{\partial F(x_3)} = \frac{\partial C_{2,3}(u_2,u_3)}{\partial u_3}, \\ F\left(x_3 \left| x_4 \right.\right) &= \frac{\partial C_{3,4}(F(x_3),F(x_4))}{\partial F(x_4)} = \frac{\partial C_{3,4}(u_3,u_4)}{\partial u_4}. \end{aligned}$$

The derivation of these expressions for the Normal, Student's t, Clayton and Gumbel copulas can be found in [1].

6. Compute the Kendall's tau values for all possible possible edges that meet the proximity condition from the conditional copula data $\boldsymbol{D_U}$ defined over $\boldsymbol{U} = \left(F\left(X_1 \left| X_2 \right.\right), F\left(X_2 \left| X_3 \right.\right), F\left(X_3 \left| X_4 \right.\right)\right)$, such that we have

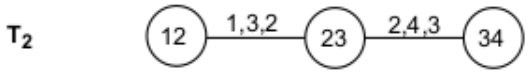
	$F\left(x_1 \left\| x_2 \right.\right)$	$F\left(x_2 \left\| x_3 \right.\right)$	$F\left(x_3 \left\| x_4 \right.\right)$
$F\left(x_1 \left\| x_2 \right.\right)$		$0,6$	NA
$F\left(x_2 \left\| x_3 \right.\right)$			$0,5$
$F\left(x_3 \left\| x_4 \right.\right)$			

where NA means that the Kendall's tau value is not being computed (the nodes {12} and {34} are not joined by an edge).

7. Build the tree T_2 that maximizes the sum of absolute Kendall's taus and fit two conditional pair-copulas, namely

$$\begin{aligned} &c_{1,3|2}\left(F\left(x_1 \left| x_2 \right.\right), F\left(x_3 \left| x_2 \right.\right)\right), \\ &c_{2,4|3}\left(F\left(x_2 \left| x_3 \right.\right), F\left(x_4 \left| x_3 \right.\right)\right), \end{aligned}$$

which are associated to the edges of the second tree as shown in the following figure:



8. Obtain conditional copula data $\boldsymbol{D_U}$ (needed for T_3) by evaluating conditional distribution functions using the bivariate copulas fitted in T_2, such that we have

$$\begin{aligned} F\left(x_1 \left| x_2, x_3 \right.\right) &= \frac{\partial C_{1,3|2}(F(x_1|x_2),F(x_3|x_2))}{\partial F(x_3|x_2)}, \\ F\left(x_2 \mid x_3, x_4\right) &= \frac{\partial C_{2,4|3}(F(x_2|x_3),F(x_4|x_3))}{\partial F(x_4|x_3)}. \end{aligned}$$

9. Build the tree T_3 and fit the conditional pair-copula

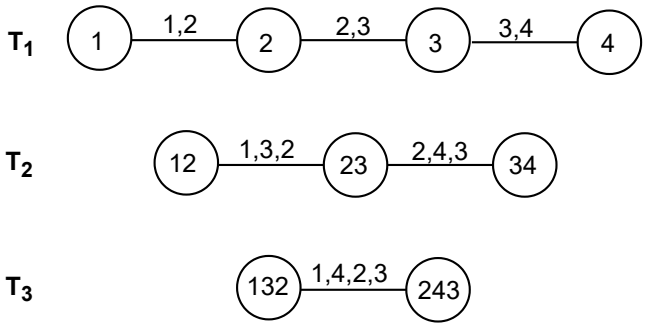
$$c_{1,4|2,3}\left(F\left(x_1 \left| x_2, x_3 \right.\right), F\left(x_4 \left| x_2, x_3 \right.\right)\right),$$

which is associated to the single edge of the last tree as shown in the following figure:

10. The result is a four-dimensional R-vine density function with six pair-copulas written as

$$f(x_1, x_2, x_3, x_4) = c_{1,2} \cdot c_{2,3} \cdot c_{3,4} \cdot c_{1,3|2} \cdot c_{2,4|3} \cdot c_{1,4|2,3} \cdot \prod_{i=1}^{4} f_i(x_i).$$

The corresponding R-vine graph is shown in the following figure:



Bibliography

[1] K. Aas, C. Czado, A. Frigessi, and H. Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2):182–198, 2009.

[2] H. Akaike. A new look at statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[3] E. Allevi, L. Boffino, M. E. De Giuli, and G. Oggioni. Analysis of long-term natural gas contracts with vine copulas in optimization portfolio problems. *Annals of Operations Research*, 274(1):1–37, 2019.

[4] L. Bandeira, J. S. Marques, J. Saraiva, and P. Pina. Automated detection of Martian dune fields. *IEEE Geoscience and Remote Sensing Letters*, 8(4):626–630, 2011.

[5] L. Bandeira, J. S. Marques, J. Saraiva, and P. Pina. Advances in automated detection of sand dunes on Mars. *Earth Surface Processes and Landforms*, 38(3):275–283, 2013.

[6] Alexander Bauer and Claudia Czado. Pair-copula Bayesian networks. *Journal of Computational and Graphical Statistics*, 25(4):1248–1271, 2016.

[7] Alexander Bauer, Claudia Czado, and Thomas Klein. Pair-copula constructions for non-Gaussian DAG models. *Canadian Journal of Statistics*, 40(1):86–109, 2012.

[8] T. Bedford and R. M. Cooke. Probability density decomposition for conditionally dependent random variables modeled by vines. *Annals of Mathematics and Artificial Intelligence*, 32(1):245–268, 2001.

[9] T. Bedford and R. M. Cooke. Vines – a new graphical model for dependent random variables. *The Annals of Statistics*, 30(4):1031–1068, 2002.

[10] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[11] M. A. Bishop. Nearest neighbor analysis of mega-barchanoid dunes, Ar Rub'al Khali, sand sea: The application of geographical indices to the understanding of dune field self-organization, maturity and environmental change. *Geomorphology*, 120(3):186–194, 2010.

[12] B. Blankertz, G. Curio, and K. R. Muller. Classifying single trial EEG: Towards brain computer interfacing. *Advances in Neural Information Processing Systems*, 1:157–164, 2002.

[13] M. C. Bourke, N. Lancaster, L. K. Fenton, E. J. R. Parteli, J. R. Zimbelman, and J. Radebaugh. Extraterrestrial dunes: An introduction to the special issue on planetary dune systems. *Geomorphology*, 121(1):1–14, 2010.

[14] A. W. Bowman and A. Azzalini. *Applied Smoothing Techniques for Data Analysis*. New York: Oxford University Press Inc, 1997.

[15] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[16] E. C. Brechmann. Truncated and simplified regular vines and their applications. Diploma thesis, University of Technology, Munich, Germany, 2010.

[17] E. C. Brechmann, C. Czado, and K. Aas. Truncated regular vines in high dimensions with application to financial data. *Canadian Journal of Statistics*, 40(1):68–85, 2012.

[18] E. C. Brechmann and H. Joe. Truncation of vine copulas using fit indices. *Journal of Multivariate Analysis*, 138:19–33, 2015.

[19] E. C. Brechmann and U. Schepsmeier. Modeling dependence with C- and D-Vine copulas: The R package CDVine. *Journal of Statistical Software*, 52(3):1–27, 2013.

[20] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[21] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. CRC press, 1984.

[22] NJ. B. Brunel, J. Lapuyade-Lahorgue, and W. Pieczynski. Modeling and unsupervised classification of multivariate hidden Markov chains with copulas. *IEEE Transactions on Automatic Control*, 55(2):338–349, 2009.

[23] D. Carrera. Modelado de dependencias con vines basados en cópulas Bernstein, 2012. Bachelor thesis, 2012. University of Havana. (In Spanish).

[24] D. Carrera, L. Bandeira, R. Santana, and J. A. Lozano. Detection of sand dunes on Mars using a regular vine-based classification approach. *Knowledge-Based Systems*, 163:858–874, 2019. https://link.springer.com/article/10.1007

[25] D. Carrera, R. Santana, and J. A. Lozano. Vine copula classifiers for the mind reading problem. *Progress in Artificial Intelligence*, 5(4):289–305, 2016.

[26] D. Carrera, R. Santana, and J. A. Lozano. The relationship between graphical representations of regular vine copulas and polytrees. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 678–690. Springer, 2018.

[27] E. Castillo, J. M. Gutiérrez, and A. S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer-Verlag, 1997.

[28] B. Chang, S. Pan, and H. Joe. Vine copula structure learning via Monte Carlo tree search. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 353–361, 2019.

[29] Y. Chen. A copula-based supervised learning classification for continuous and discrete data. *Journal of Data Science*, 14(4):769–782, 2016.

[30] R. M. Cooke, H. Joe, and B. Chang. Vine regression. *Resources for the Future Discussion Paper*, pages 15–52, 2015.

[31] G. Corder and D. Foreman. *Nonparametric Statistics: A Step-by-Step Approach*. Wiley & Son, 2009. 2nd Edition.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2009.

[33] G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks*. Springer, New York, 2003.

[34] A. Cuesta-Infante, R. Santana, J. I. Hidalgo, C. Bielza, and P. Larrañaga. Bivariate empirical and n-variate archimedean copulas in estimation of distribution algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 1355–1362, 2010.

[35] C. Czado. Pair-copula constructions of multivariate copulas. In P. Jaworski, F. Durante, W. K. Härdle, and T. Rychlik, editors, *Copula Theory and Its Applications*, volume 198 of *Lecture Notes in Statistics*, pages 93–109. Springer, Berlin, Heidelberg, 2010.

[36] C. Czado, E. C. Brechmann, and L. Gruber. Selection of vine copulas. In *Copulae in mathematical and quantitative finance*, pages 17–37. Springer, 2013.

[37] C. Czado, S. Jeske, and M. Hofmann. Selection strategies for regular vine copulae. *Journal de la Société Française de Statistique*, 154(1):174–191, 2013.

[38] H. D. de Mello Junior, L. Marti, A. V. A. da Cruz, and M. M. B. R. Vellasco. Evolutionary algorithms and elliptical copulas applied to continuous optimization problems. *Information Sciences*, 369:419–440, 2016.

[39] P. Deheuvels. La fonction de dépendance empirique et ses propriétés. un test non paramétrique d'indépendance. *Bulletins de l'Académie Royale de Belgique*, 65(1):274–292, 1979.

[40] P. Deheuvels. An asymptotic decomposition for multivariate distribution-free tests of independence. *Journal of Multivariate Analysis*, 11(1):102–113, 1981.

[41] R. S. Dhawal and L. Chen. A copula based method for the classification of fish species. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 11(1):29–45, 2017.

[42] J. Dissmann, E. C. Brechmann, C. Czado, and D. Kurowicka. Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59:52–69, 2013.

[43] J. F. Dissmann. Statistical inference for regular vines and application. Diploma thesis, University of Technology, Munich, Germany, 2010.

[44] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proc. 13th Intl. Conf. Machine Learning*, pages 105–112, 1996.

[45] G. Elidan. Copula Bayesian networks. In *Advances in Neural Information Processing Systems*, pages 559–567, 2010.

[46] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2010.

[47] L. K. Fenton and R. K. Hayward. Southern high latitude dune fields on Mars: Morphology, aeolian inactivity, and climate change. *Geomorphology*, 121(1):98–121, 2010.

[48] L. K. Fenton, A. D. Toigo, and Mark I. R. Aeolian processes in Proctor crater on Mars: Mesoscale modeling of dune-forming winds. *Journal of Geophysical Research: Planets*, 110(E6), 2005.

[49] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[50] B. J. Frey and D. Dueck. Mixture modeling by affinity propagation. In *Proceedings of the 2005 Conference Advances in Neural Information Processing Systems 18, NIPS*, pages 379–386. MIT Press, 2006.

[51] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

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

[53] D. Garrett, D.A. Peterson, C.W. Anderson, and M.H. Thaut. Comparison of linear, nonlinear, and feature selection methods for EEG signal classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):141–144, 2003.

[54] C. Genest and A. C. Favre. Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of Hydrologic Engineering*, 12(4):347–368, 2007.

[55] C. Genest, J. F. Quessy, and B. Remillard. Asymptotic local efficiency of Cramér-von Mises tests for multivariate independence. *The Annals of Statistics*, 35:166–191, 2007.

[56] C. Genest, B. Remillard, and D. Beaudoin. Goodness-of-fit tests for copulas: A review and a power study. *Insurance: Mathematics and Economics*, 44:199–213, 2009.

[57] P. Geurts, D. Ernst, and L Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[58] Y. González-Fernández and M. Soto. *copulaedas: Estimation of Distribution Algorithms Based on Copulas*, 2015. R package version 1.4.2, https://CRAN.R-project.org/package=copulaedas.

[59] Y. González-Fernández and M. Soto. *vines: Multivariate Dependence Modeling with Vines*, 2016. R package version 1.1.5, https://CRAN.R-project.org/package=vines.

[60] R. Greeley, R. O. Kuzmin, and R. M. Haberle. Aeolian processes and their effects on understanding the chronology of Mars. *Space Science Reviews*, 96(1):393–404, 2001.

[61] I. H. Haff, K. Aas, and A. Frigessi. On the simplified pair-copula construction — simply useful or too simplistic? *Journal of Multivariate Analysis*, 101:1296–1310, 2010.

[62] I. H. Haff, K. Aas, A. Frigessi, and V. Lacal. Structure learning in Bayesian networks using regular vines. *Computational Statistics & Data Analysis*, 101:186–208, 2016.

[63] M. Hahsler and K. Hornik. Tsp – Infrastructure for the Traveling Salesperson Problem. *Journal of Statistical Software*, 23(2):1–21, 2007.

[64] M. Hahsler and K. Hornik. *TSP: Traveling Salesperson Problem (TSP)*, 2020. R package version 1.1-10.

[65] A. M. Hanea. Non-parameteric Bayesian belief nets versus vines. pages 281–303, 2011.

[66] E. Haselsteiner and G. Pfurtscheller. Using time-dependent neural networks for EEG classification. *IEEE Transactions on Rehabilitation Engineering*, 8(4):457–463, 2000.

[67] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2009.

[68] R. K. Hayward, K. F. Mullins, L. K. Fenton, T. M. Hare, T. N. Titus, M. C. Bourke, A. Colaprete, and P. R. Christensen. Mars global digital dune database and initial science results. *Journal of Geophysical Research: Planets*, 112(E11), 2007.

[69] G. E. Hinton. Connectionist learning procedures. Artificial Intelligence, 40 1-3: 185–234, 1989. Reprinted in J. Carbonell, editor. *Machine Learning: Paradigms and Methods", MIT Press*, 1990.

[70] J. L. Hintze and R. D. Nelson. Violin plots: a box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

[71] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[72] C. H. Hugenholtz and T. E. Barchyn. Spatial analysis of sand dunes with a new global topographic dataset: new approaches and opportunities. *Earth surface processes and landforms*, 35(8):986–992, 2010.

[73] H. Huttunen, J. P. Kauppi, and J. Tohka. Regularized logistic regression for mind reading with parallel validation. *Proceedings of ICANN/PASCAL2 Challenge: MEG Mind-Reading*, pages 20–24, 2011.

[74] H. Joe. Families of m-variate distributions with given margins and $m(m-1)/2$ bivariate dependence parameters. In L. Rüschendorf, B. Schweizer, and M. D. Taylor, editors, *Distributions with Fixed Marginals and Related Topics*, pages 120–141, 1996.

[75] H. Joe. *Multivariate Models and Dependence Concepts*. Chapman & Hall, 1997.

[76] H. Joe. Asymptotic efficiency of the two-stage estimation method for the copula-based models. *Journal of Multivariate Analysis*, 94(2):401–419, 2005.

[77] H. Joe and J .J. Xu. The estimation method of inference functions for margins for multivariate models. Technical Report 166, University of British Columbia, 1996.

[78] P. Jylänki, J. Riihimäki, and A. Vehtari. Multi-class Gaussian process classification of single trial MEG based on frequency specific latent features extracted with binary linear classifiers. pages 31–34, 2011.

[79] M. Killiches, D. Kraus, and C. Czado. Examination and visualisation of the simplifying assumption for vine copulas in three dimensions. *Australian & New Zealand Journal of Statistics*, 59(1):95–117, 2017.

[80] A. Klami, P. Ramkumar, S. Virtanen, L. Parkkonen, R. Hari, and S. Kaski. ICANN/PASCAL2 challenge: MEG mind reading Overview and results. In A. Klami, editor, *Proceedings of ICANN/PASCAL2 Challenge: MEG Mind Reading*, Aalto University Publication series SCIENCE + TECHNOLOGY, pages 3–19. Aalto University, 2011.

[81] I. Kojadinovic, J. Yan, and M. Holmes. Fast large-sample goodness-of-fit tests for copulas. *Statistica Sinica*, 21:841–871, 2011.

[82] D. Kurowicka. Optimal truncation of vines. In *Dependence Modeling: Vine Copula Handbook*, pages 233–247. World Scientific, 2010.

[83] D. Kurowicka and R. M. Cooke. The vine copula method for representing high dimensional dependent distributions: Application to continuous belief nets. In *Proceedings of the Winter Simulation Conference*, volume 1, pages 270–278. IEEE, 2002.

[84] D. Kurowicka and R. M. Cooke. *Uncertainty Analysis with High Dimensional Dependence Modelling*. John Wiley & Sons, 2006.

[85] P. Langley, W. Iba, K. Thompson, et al. An analysis of Bayesian classifiers. In *AAAI*, volume 90, pages 223–228, 1992.

[86] S. L. Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[87] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer. Independence properties of directed Markov fields. *Networks*, 20(5):491–505, 1990.

[88] M. A. Lebedev and M. A. L. Nicolelis. Brain-machine interfaces: Past, present and future. *TRENDS in Neurosciences*, 29(9):536–546, 2006.

[89] F. Lotte, M. Congedo, A. Lecuyer, F. Lamarche, and B. Arnaldi. A review of classification algorithms for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 4:R1–R13, 2007.

[90] E. D. McKee. Introduction to a study of global sand seas. In McKee E. D., editor, *Copulae in mathematical and quantitative finance*, pages 1–19. University Press of the Pacific, 1979.

[91] O. Morales-Napoles, R. Cooke, and D. Kurowicka. About the number of vines and regular vines on n nodes. *Institutional Repository. http://resolver.tudelft.nl/uuid:912abf55-8112-48d2-9cca-323f7f6aecc7*, 2010.

[92] T. Mroz, S. Fuchs, and W. Trutschnig. How simplifying and flexible is the simplifying assumption in pair-copula constructions–analytic answers in dimension three and a glimpse beyond. *Electronic Journal of Statistics*, 15(1):1951–1992, 2021.

[93] D. Müller and C. Czado. Representing sparse Gaussian DAGs as sparse Rvines allowing for non-Gaussian dependence. *Journal of Computational and Graphical Statistics*, 2017.

[94] T. Nagler, C. Bumann, and C. Czado. Model selection in sparse high dimensional vine copula models with application to portfolio risk. *arXiv:1801.09739l*, 2018.

[95] T. Nagler and C. Czado. Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas. *Journal of Multivariate Analysis*, 151:69–89, 2016.

[96] T. Naselaris, K. N. Kay, S. Nishimoto, and J. L. Gallant. Encoding and decoding in fMRI. *Neuroimage*, 56(2):400–410, 2011.

[97] R. B. Nelsen. *An Introduction to Copulas*. Springer, 2 edition, 2006.

[98] H. Noh, A. E. Ghouch, and T. Bouezmarni. Copula-based regression estimation and inference. *Journal of the American Statistical Association*, 108(502):676–688, 2013.

[99] E. Olivetti, S. M. Kia, and P. Avesani. MEG decoding across subjects. In *Proceedings of the 2014 International Workshop on Pattern Recognition in Neuroimaging*, pages 1–4, 2014.

[100] R. S. Olson, W. La Cava, Z. Mustahsan, A. Varik, and J. H. Moore. Data-driven advice for applying machine learning to bioinformatics problems. *arXiv:1708.05070*, 2017.

[101] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[102] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[103] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[104] D. M. W. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[105] R. C. Prim. Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36(6):1389–1401, 1957.

[106] A. Rakotomamonjy and V. Guigue. BCI competition III: Dataset II-ensemble of SVMs for BCI P300 speller. *Biomedical Engineering, IEEE Transactions on*, 55(3):1147–1154, 2008.

[107] H. Ramoser, J. Muller-Gerking, and G. Pfurtscheller. Optimal spatial filtering of single trial EEG during imagined hand movement. *Rehabilitation Engineering, IEEE Transactions on*, 8(4):441–446, 2000.

[108] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[109] J. W. Rieger, C. Reichert, K. R. Gegenfurtner, T. Noesselt, C. Braun, H. J. Heinze, R. Kruse, and H. Hinrichs. Predicting the recognition of natural scenes from single trial MEG recordings of brain activity. *Neuroimage*, 42(3):1056–1068, 2008.

[110] J. D. Rodríguez, A. Perez, and J. A. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):569–575, 2010.

[111] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

[112] R. Salinas-Gutiérrez, A. Hernández Aguirre, M. J. J. Rivera-Meraz, and E. R. Villa Diharce. Supervised probabilistic classification based on gaussian copulas. In *Advances in Soft Computing - 9th Mexican International Conference on Artificial Intelligence, MICAI 2010, Pachuca, Mexico, November 8-13, 2010, Proceedings, Part II*, pages 104–115, 2010.

[113] R. Santana, C. Bielza, and P. Larrañaga. An ensemble of classifiers approach with multiple sources of information. In A. Klami, editor, *Proceedings of ICANN/PASCAL2 Challenge: MEG Mind Reading*, Aalto University Publication series SCIENCE + TECHNOLOGY, pages 25–30. Aalto University, 2011.

[114] R. Santana, C. Bielza, and P. Larrañaga. Regularized logistic regression and multi-objective variable selection for classifying MEG data. *Biological Cybernetics*, 106(6-7):389–405, 2012.

[115] S. Sathe. A novel Bayesian classifier using copula functions. *arXiv preprint cs/0611150*, 2006.

[116] U. Schepsmeier. Maximum likelihood estimation of c-vine pair-copula constructions based on bivariate copulas from different families, 2010.

[117] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[118] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[119] S. Silvestro, L. K. Fenton, D. A. Vaz, N. T. Bridges, and G. G. Ori. Ripple migration and dune activity on Mars: Evidence for dynamic wind processes. *Geophysical Research Letters*, 37(20), 2010.

[120] A. Sklar. Fonctions de repartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Universite de Paris*, 8:229–231, 1959.

[121] M. Soto, Y. González-Fernández, and A. Ochoa. Modeling with copulas and vines in estimation of distribution algorithms. *Investigación Operacional*, 36(1):1–23, 2015. http://rev-inv-ope.univ-paris1.fr/spip.php?rubrique17.

[122] M. Soto, A. Ochoa, Y. González-Fernández, Y. Milanés, A. Alvarez, D. Carrera, and E. Moreno. Vine estimation of distribution algorithms with application to molecular docking. In S. Shakya and R. Santana, editors, *Markov Networks in Evolutionary Computation*, volume 14 of *Adaptation, Learning, and Optimization*, pages 209–225. Springer, 2012. ISBN 978-3-642-28899-9.

[123] Y. Stitou, N. Lasmar, and Y. Berthoumieu. Copulas based multivariate gamma modeling for texture classification. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1045–1048. IEEE, 2009.

[124] J. Stoeber, H. Joe, and C. Czado. Simplified pair copula constructions - limitations and extensions. *Journal of Multivariate Analysis*, 119:101–118, 2013.

[125] Y. Sun, A. Cuesta-Infante, and K. Veeramachaneni. Learning vine copula models for synthetic data generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5049–5057, 2019.

[126] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[127] D. A Vaz, P. T. K. Sarmento, M. T. Barata, L. K. Fenton, and T. I. Michaels. Object-based dune analysis: Automated dune mapping and pattern characterization for Ganges Chasma and Gale crater, Mars. *Geomorphology*, 250:128–139, 2015.

[128] B. Wang, Y. Sun, T. Zhang, T. Sugi, and X. Wang. Bayesian classifier with multivariate distribution based on D-vine copula model for awake/drowsiness interpretation during power nap. *Biomedical Signal Processing and Control*, 56:101686, 2020.

[129] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*, volume 19. John Wiley & Sons, 2009.

[130] S. A. Wilson and J. R. Zimbelman. Latitude-dependent nature and physical characteristics of transverse aeolian ridges on Mars. *Journal of Geophysical Research: Planets*, 109(E10), 2004.

[131] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, 2002.

[132] L. Yang, E. W. Frees, and Z. Zhang. Nonparametric estimation of copula regression models with discrete outcomes. *Journal of the American Statistical Association*, 115(530):707–720, 2020.

[133] H. F Yu, F. L. Huang, and C. J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1):41–75, 2011.

[134] H. Zhang. The optimality of Naive Bayes. *American Association for Artificial Intelligence (www. aaai. org)*, 2004.

[135] Y. Zhang, X. Wang, D. Liu, C. Li, Q. Liu, Y. Cai, Y. Yi, and Z. Yang. Joint probability-based classifier based on vine copula method for land use classification of multispectral remote sensing data. *Earth Science Informatics*, 13(4):1079–1092, 2020.

[136] W. Zheng, X. Ren, N. Zhou, D. Jiang, and S. Li. Mixture of d-vine copulas for chemical process monitoring. *Chemometrics and Intelligent Laboratory Systems*, 169:19–34, 2017.

[137] Z. H. Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC Press, 2012.