CM Flows - Copula Density Estimation with Normalizing Flows

Leonie Bossemeyer

Master of Science
Data Science
School of Informatics
University of Edinburgh
2020

Abstract

Copulas describe the dependence structure between random variables. The accurate estimation of copulas is vital to modeling flexible multivariate distributions and to describing dependencies between variables when working with non-Gaussian data. This thesis implements and evaluates a non-parametric copula estimation technique based on Normalizing Flows, a family of generative models that have recently shown success in efficient and exact density estimation [1]. Wiese et al. [2] propose Copula and Marginal Generative Flows (CM Flows), a bivariate copula estimation model that utilizes Normalizing Flows to estimate copulas given joint distribution samples. I implement CM Flows using deep dense sigmoid flows (DDSFs) and real-valued non-volume preserving (RealNVP) transformations. Additionally, I extend CM Flows to estimate multivariate copulas using regular vine trees (R-Vines). All models are then evaluated in a simulation study using the Archimedean copulas Clayton, Frank and Gumbel. The results show that bivariate CM Flows perform well on the Clayton and Frank copula, achieving a Jensen-Shannon (JS) divergence of under 0.1 for both copulas and outperforming a parametric model under false copula family assumptions. On the Gumbel copula, the estimated copula shows a larger JS divergence of 0.22. For multivariate copulas, the CM Flows only achieve a JS divergence between 0.31 and 0.66 depending on the dataset. Bivariate CM Flows are thus not always a suitable copula estimation technique when the underlying distribution is unknown, as it performs poorly on the Gumbel copula, but can outperform the parametric model on the Clayton and Frank copula datasets. Overall, CM Flows are shown to be a promising technique that can extend the success of Normalizing Flows to the field of copula estimation.

Acknowledgements

I would like to thank my supervisor Dr. Arno Onken sincerely for his continued support and guidance throughout the dissertation period, and for proposing such a fun and interesting project.
I would also like to thank my fellow students for continuing their spirit of encouragement and support online after Appleton Tower was closed and we could not sit together on the seventh floor anymore.
Finally, I would like to thank my family for their love and support and my friends for making Edinburgh feel like home while the world was going silent.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Leonie Bossemeyer*)

Table of Contents

Chapter 1

Introduction

This thesis implements and explores non-parametric copula estimation using Normalizing Flows. To motivate the topic, I first describe the need for copulas in statistical modeling and explain why non-parametric copula estimation is advantageous over parametric copula estimation in empirical settings. I then motivate the use of Normalizing Flows for copula estimation and finally explain the research questions and outline of the project.

1.1 Copulas in statistical modeling

A copula is a mathematical tool that describes the dependency structure between two or more random variables. Any joint distribution can be separated into its marginals and its copula, and the copula can in turn be used to combine the marginal distributions into a joint distribution [3].

The use of copulas can benefit statistical modeling in two ways: dependency modeling and multivariate joint distribution modeling [4]. For dependency modeling, copulas offer the benefit of capturing the full dependency structure between variables, rather than just modeling linear dependencies, such as the commonly used correlation coefficient [5]. For multivariate modeling, they offer an increased flexibility in two aspects: first, they can be used to separate the modeling of the marginal distributions from the modeling of the dependency structure. When using a known multivariate distribution for modeling, both dependency structure and marginal distributions are determined by the choice of multivariate distribution. Second, the independent modeling of each marginal allows the use of univariate distributions, which offer a greater variety than multivariate distributions [6].

Examples of copulas being used for both purposes are numerous: to study of dependence of two time series of financial data [7], to model the dependence between input factors of feasibility studies for development plans [8], to predict asset prices [9] or to predict wind and solar power generation given weather predictions [10].

1.2 Copula estimation

To be able to use a copula for dependency modeling or joint distribution modeling, it has to first be estimated. Unlike the marginal and joint distributions, which are directly observable, the copula is the hidden dependency structure between marginal distributions [11]. A simple way to estimate a copula is to choose a specific copula family and then fit the parameters to the given dependency structure [12]. However, when working in empirical settings, the copula family is usually unknown and can only be assumed.

One example of this occurred in what led up to the financial crisis of 2008: a copula function approach by Li [13] had become the canonical tool for modeling the time until default of financial products, such as asset swap spreads and bond prices. Given only measurements of the marginal distributions of the product components derived from market information, the tool modeled a joint distribution, allowing complex risks to be modeled from attainable data. However, the formula makes the critical assumption that the dependence structure follows a Gaussian copula [13]. Gaussian copulas have a low dependence in the tails, meaning that they predict a low interaction between the variables for unlikely events [14]. This introduced a false sense of security when joining financial products, since the accumulated risk was underestimated, and has been partially blamed for the financial crisis [15].

To avoid making false assumptions on modeled distributions, non-parametric methods can be used. Placing no assumption on the family of the copula or the marginals, they offer a greater generality of the found results. Popular non-parametric density estimation methods include kernel-based methods [16, 17] and neural networks [1, 18, 19]. For copula estimation, kernel-based methods have been well explored [11, 20], while literature on copula estimation using neural networks remains sparse. This thesis focuses on a type of neural network that has recently gained popularity for density estimation [1]: Normalizing Flows.

1.3 Normalizing Flows for density estimation

Normalizing Flows are bijective functions that explicitly model the input distribution, allowing for efficient and exact sampling and density evaluation [1]. Being bijective, their loss function is simply the negative log likelihood. They are more straightforward to train than other generative models such as Generative Adversarial Networks and Variational auto-encoders, which experience phenomena such as mode collapse, posterior collapse, vanishing gradients and training instability [1, 21, 22].

To utilize the success of Normalizing Flows in density estimation, Wiese et al. [2] propose Copula and Marginal Generative Flows (CM Flows), a method to estimate copula and marginal from a joint distribution using a combination of multiple Normalizing Flows. Given bivariate joint distribution samples, their model uses one flow to estimate each marginal and project the marginals onto a normal distribution. Then, an additional flow is used to learn the distribution of a copula-like structure from these normal distributions. After transformation, they can then generate samples from the copula. This approach allows non-parametric copula estimation given only joint samples, and utilizes the density estimation strength of Normalizing Flows. However, Wiese et al. [2] did not provide a detailed explanation of how CM Flows could be constructed, and did not show a simulation study for copula estimation. In this thesis, I use the concept of CM Flows and implement them, evaluating performance on synthetic data.

1.4 Multivariate copulas

Wiese et al. [2] also only describe CM Flows for bivariate copula estimation, rather than multivariate copula estimation. Multivariate copulas are needed when modeling multivariate data. For bivariate copulas, a large variety of copula families exist, covering many types of dependency structures. For multivariate copulas, the defined copula families are less flexible. To increase flexibility, bivariate copulas can be combined into multivariate copulas using a graphical structure of nested trees called vine copulas [23, 24]. Regular vines (R-Vines) are a subgroup of vine copulas that restrict the complexity of the nested copulas [25]. To find a fitting R-Vine, sequential algorithms can be employed, which estimate the R-Vine tree by tree. In this thesis, I use a sequential R-Vine estimation algorithm by Dißmann et al. [25], employing bivariate CM Flows for bivariate copula estimation within each tree.

1.5 Research questions and project description

This thesis first investigates whether CM Flows are a suitable method for non-parametric bivariate copula estimation. Since CM Flows are built upon the Normalizing Flows deep dense sigmoid flows (DDSFs) and real-valued non-volume preserving (RealNVP) transformations, I first test their ability to estimate diverse marginal distributions and copula distributions. I then implement bivariate CM Flows in PyTorch [26] and evaluate them using a simulation study of the three Archimedean copulas Clayton, Frank and Gumbel [3]. The main evaluation metrics are the Jensen-Shannon (JS) divergence between the true and predicted copula and the uniformity of marginals. A parametric copula estimator which uses a copula family assumption different from the true copula family functions as a baseline for the bivariate CM Flows. The results on DDSF and RealNVP show that they are able to capture the given distributions well, showing a low JS divergence to all datasets. The results show that CM Flows are only partially suitable for non-parametric bivariate copula estimation: while delivering satisfactory results on the Clayton and Frank copula, showing a JS divergence below 0.1 for both copula estimates and outperforming the baseline results, their performance on the Gumbel shows a JS divergence of 0.2, which is larger than the baseline JS divergence.

The second research question is whether CM Flows can be combined with R-Vines to provide multivariate copula estimation. To test this, I implement multivariate CM Flows using the R-Vine selection algorithm published by Dißmann et al. [25], and replacing the parametric copula estimation with bivariate CM Flows in each tree. As part of the multivariate CM Flow, a conditional copula estimator is needed, which I implement using the RealNVP. To evaluate the model, I first test the conditional copula estimator using the Clayton, Frank and Gumbel copula and then test the multivariate CM Flows using four dimensional datasets of different copula types. The evaluation metric is again the JS divergence between the true and predicted copula. The results on the conditional copula flow show that it is able to capture conditional copula densities well. The results on multivariate CM Flows however indicate, that multivariate CM Flows are currently not able to estimate multivariate copulas very accurately, showing a large divergence between estimate and ground truth.

This thesis is structured as follows: chapter 2 provides the necessary background on copulas and Normalizing Flows and introduces the relevant literature. Chapter 3 describes the methodology and evaluation techniques. Chapter 4 introduces the exper-

iments used to answer the research questions. Chapter 5 shows the results. Chapter 6 discusses the results and chapter 7 concludes. In each of chapter 3, 4 and 5, the bivariate CM Flow is treated first, including their components marginals flows and copula flows, followed by the multivariate CM.

Chapter 2

Background

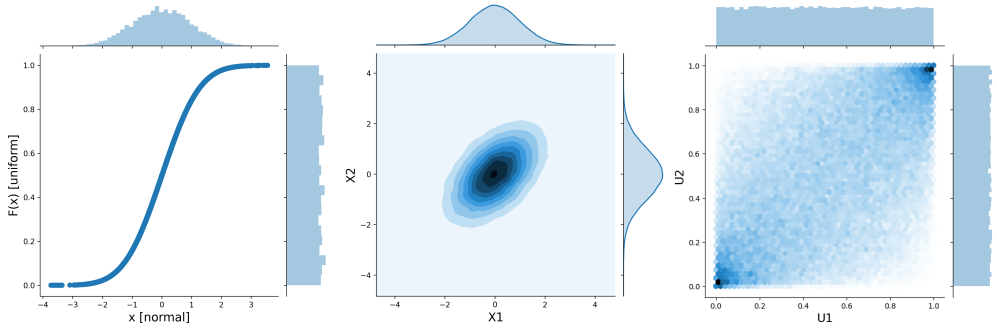
This chapter first provides definitions for the most important concepts used in copula theory and gives an overview over copula estimation methods. Then, density estimation using Normalizing Flows is explained, including an explanation of coupling flows, such as RealNVP [27], and autoregressive flows, such as DDSF [28]. Finally, the multivariate copula construction using bivariate copulas is demonstrated.

2.1 Copulas

Copulas describe the dependency structure between random variables. The copula distribution is defined as the cumulative distribution function (CDF) $C(u_1,...,u_d) = P(U_1 \leq u_1,...,U_d \leq u_d)$ of a random vector on the hypercube $[0,1]^d$ with uniform marginals over [0,1], where $U_i \sim U_{[0,1]}$, $i = 1,...,d$ and $C(u_1,...,u_d) : [0,1]^d \to [0,1]$, $d \in \mathbb{N}$. The copula probability density function (PDF), or copula density, follows as $c(u_1,...,u_d) = \partial^d C(u_1,...,u_d)/\partial u_1 ... \partial u_d$.

Panel (c) of figure 2.1 illustrates the copula of a bivariate normal joint distribution. It can be revealed using a **probability integral transform**: for any distribution X with CDF F_X, $F_X(X)$ is uniformly distributed [29]. This can be seen for the standard normal distribution in panel (a) of figure 2.1. When transforming the marginals of the normal joint distribution in panel (b), the copula remains, as seen in panel (c).

Sklar's theorem [3] formalizes how copulas connect the marginal and joint distributions: Let $X = (X_1,...,X_d)$ be a d-dimensional random vector with CDF F_X that has the marginal CDFs $F_1,...,F_d$, denote f_X the joint PDF and $f_1,...,f_d$ the marginal PDFs.



(a) Probability Integral Transform (b) Joint Distribution (c) Copula

Figure 2.1: Panel (a) shows the probability integral transform of a standard normal distribution: the normal distribution (top side) is transformed into a uniform distribution (right side) using the standard normal CDF. Panel (b) shows a bivariate normal distribution which is transformed to a copula (panel (c)) using probability integral transform on both marginals (top and right side).

Then, there exists a copula C such that

$$F_X(x_1,...,x_d) = C(F_1(x_1),...,F_d(x_d)), x_i \in \mathbb{R} \tag{2.1}$$

and

$$f_X(x_1,...,x_d) = c(F_1(x_1),...,F_d(x_d)) \prod_{i=1}^{d} f_i(x_i) \tag{2.2}$$

if the PDF f_X exists. Thus, the joint PDF f_X can be separated into the copula density c and the product of marginal densities $f_i, i = 1,...,d$ [3]. This allows for an interpretation of the copula as the part of the joint density function that is independent from the marginals and captures the dependencies between the variables [30]. When the marginals are continuous, the copula is unique [3].

One important class of copulas are the **Archimedean copulas**, such as the Clayton, Frank and Gumbel copula. They are easily constructed, and can display a large variety of dependency structures. By construction, they are defined by just one parameter, θ [3]. See table A.1 for definitions of the generating function for common Archimedean copulas [3] and figure 5.2 (second row) for a illustration of the Clayton, Frank and Gumbel copula.

One common method to generate samples (u,v) from a copula distribution C is the **conditional distribution method**. It relies on first sampling u from a uniform distribution, and then sampling v from the conditional distribution of the copula given

u, C_u. It can be shown that C_u is just the partial derivative of the copula density with respect to u, $C_u = \frac{\partial C(u,v)}{\partial u}$, and that the inverse, c_u^{-1} exists. (u,v) can then be sampled by first sampling two independent standard uniform random variables u and t, and then finding v by $v = c_u^{-1}(t)$. (u,v) is then a sample from the copula distribution C [31].

To find the copula of a joint distribution, the copula density can be estimated given samples from the distribution. **Copula estimation methods** can be divided into parametric methods, which place assumptions on both the marginal and copula distributional families, semi-parametric methods, which place assumptions only on the distribution of the copula, and non-parametric methods, which do not place assumptions on either. Examples for parametric copula estimation include maximum-likelihood estimation by Oakes [12], and the computationally more efficient inference function for margins [32]. Semi-parametric estimation techniques are proposed by Genest et al. [33], Chen and Fan [34] and Rémillard [35].

A common method for **non-parametric copula estimation** is kernel-based density estimation. Prominent examples include the local linear kernel estimator by Chen and Fan [34], the mirror-reflection kernel estimator by Gijbels and Mielniczuk [36] and the transformation estimator by Fermanian et al. [37]. However, these estimators are not always suitable for copula estimation since they show boundary bias and are inconsistent under unbounded densities [37, 38]. Fermanian et al. [37] propose improved versions of the three kernel density estimators, which are less biased in the tails. Geenens [39] propose a *kernel-type* copula density estimator, which transforms the copula density into normal distributions and combine the kernel density estimation with local likelihood density estimation methods. Their performance depends, however, on the correct selection of a smoothing parameter [39].

2.2 Normalizing Flows

Normalizing Flows are a family of generative models that allow for both sampling and efficient and exact density evaluation [1]. They are based on the idea that a representation of the target distribution should be easy to model. A normalizing flow transforms a simple probability distribution into a more complex distribution by a sequence of invertible and differentiable functions [1]. Figure 2.2 illustrates the inference and sampling process for Normalizing Flows: the flow $f : Z \to X$ is trained by applying the inverse of the flow, $f^{-1}(x) : X \to Z$ to the data X and evaluating the negative log-likelihood of the fit on a simple prior Z. The density of the sample $x \in X$ can be

evaluated using the **change of variable theorem**, where it becomes the product of the density $f^{-1}(x)$ in the prior distribution and the change in volume introduced by the inverse flow transformation:

$$p_X(x) = \left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right| p_Z(f^{-1}(x)) \tag{2.3}$$

This formula is only tractable for bijections which are easily invertible and where the calculation of the determinant of the Jacobian is feasible. Bijections that confirm with these two attributes are called Normalizing Flows. To find these kinds of bijections, it is important to note that the composition of invertible functions is itself invertible and that the Jacobian of the composition is just the product of the Jacobians of the invertible functions. Thus, arbitrarily complex bijective functions can be created by stacking bijective functions with a tractable Jacobian determinant [1].

Through this approach, new families of distributions can be fitted by choosing a prior density and a number of invertible and differentiable transformations. Samples from this new density can then be created by sampling from the prior distribution and transforming the samples using $f : Z \to X$. The density of the new samples can be directly computed via equation 2.3 [1]. Normalizing Flows were first used for density estimation by Rippel and Adams [40] and Laurence et al. [41] and popularized by Dinh et al. [27], who introduced non-linear independent component estimation (NICE).

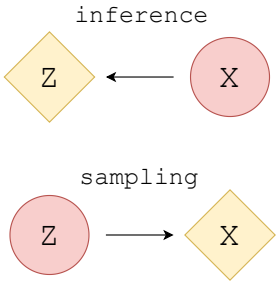


Figure 2.2: Inference and sampling in Normalizing Flows. For inference, the data X is projected onto a normal distribution, Z. For sampling, samples from a normal distribution are transformed into samples from the data distribution.

There are many types of Normalizing Flows, such as element-wise flows, linear flows and planar and radial flows. However, the most common type, and the type used in CM Flows, are coupling flows, such as the RealNVP used in the copula flow, and autoregressive flows, such as the DDSF used in marginal flows [1].

2.2.1 Coupling flows

Coupling flows contain coupling layers, which are one approach of keeping a tractable Jacobian, while allowing the use of a deep neural network for estimation. Coupling layers are a building block of the flow that causes the Jacobian to be triangular, making the determinant easily computable. In a coupling layer, the data x is split into two partitions, x^A, x^B, where d-dimensional x^A is unchanged, and x^B is transformed by an invertible function $g : \mathbb{R}^d \to \mathbb{R}^d$ [1]:

$$\begin{aligned} y^A &= x^A \\ y^B &= g(x^A; m(x^B))) \end{aligned} \tag{2.4}$$

The function $m(x^B)$ can be any function, including a deep neural network [1]. The Jacobian then takes the form [1]:

$$\frac{\partial y}{\partial x} = \begin{bmatrix} I_d & 0 \\ y^B/x^A & y^B/x^B \end{bmatrix} \tag{2.5}$$

There are different kinds of coupling layers, such as

$$\text{additive: } g(x^A, m(x^B)) = x^A + m(x^B) \tag{2.6}$$

$$\text{multiplicative: } g(x^A, m(x^B)) = x^A \times m(x^B) \tag{2.7}$$

$$\text{affine: } g(x^A, m_1(x^B), m_2(x^B)) = x_1 \times m_1(x^B) + m_2(x^B) \tag{2.8}$$

NICE uses affine coupling layers, with $m(x^B)$ being a rectified linear unit multilayer perceptron [27]. Real-valued Non-Volume Preserving (RealNVP) transformations [42] are an extension of NICE, which uses affine coupling layers, a scaling function as $m_1(x^B)$ and a shifting function as $m_2(x^B)$. Both of these are rectified convolutional networks [42]. In RealNVP, the split of x is performed using a binary mask, which is determined by either a checkerboard pattern or channel-wise masking [42].

2.2.2 Autoregressive flows

In autoregressive flows, such as the deep dense sigmoid flows (DDSFs) [28], each entry of the output $\mathbf{y} = g(\mathbf{x})$ depends on the previous entries of the input:

$$y_t = g(x_t | m_t(x_{1:t-1})). \tag{2.9}$$

By design, the Jacobian of this transformation is triangular, so the determinant is the product of the entries on the diagonal of the Jacobian. Masked autoregressive flows

(MAFs) [43] are autoregressive flows that compute equation 2.9 in one forward pass, using appropriate masks. The computation of the inverse is more challenging, and is performed sequentially [1]. An alternative approach is the inverse autoregressive flow (IAF) [44] in which each entry of $\mathbf{y}$ is conditional of the previous entries in $\mathbf{y}$. For the IAF, the forward computation is computationally expensive, while the computation of the inverse is computationally cheap. Thus, IAFs are more efficient if fast sampling is needed, and MAFs are more efficient for fast density estimation [43]. Several autoregressive flows have been proven to be able to learn any density to any precision given enough data and capacity, the so-called universal property [28, 1, 45].

In neural autoregressive flows (NAFs), introduced by Huang et al. [28], the coupling function $g : \mathbb{R}^d \to \mathbb{R}^d$ is a deep neural network. Huang et al. [28] showed that a neural network with only non-negative weights and strictly monotone activation functions is bijective, making it suitable as a coupling function. They proposed deep sigmoid flows (DSFs) and deep dense sigmoid flows (DDSFs), which are both neural networks with sigmoid layers and logit units, and non-negative weights. While deep sigmoid flows (DSFs) have a single hidden linear layer with sigmoid units, DDSFs can have more. To qualify the non-negativity restriction on the neural network weight, Wehenkel and Louppe [46] propose unconstrained monotonic neural networks (UMNN). In UMNNs, a strictly positive (or negative) function is modeled with a neural network and then integrated numerically. UMNN require fewer parameters than NAFs for a similar performance, making them more suitable for high-dimensional data [1].

2.3 Regular Vine Copulas

As seen in section 2.1, copulas are defined over two as well as more dimensions. However, while there is a large variety of parametric two dimensional copulas that are flexible enough to capture a large variety of copula densities, the choice of multivariate parametric copulas for modeling higher dimensional dependencies remains relatively limited [47]. Instead of using one multivariate parametric copula, bivariate copulas can be combined in so-called pair-copula constructions [24]. Given a vector $\mathbf{X} = (X_1, ..., X_n)$ with joint distribution $f(x_1, ..., x_n)$, the joint distribution can be factorized uniquely up to variable order:

$$f(x_1, ..., x_n) = f_n(x_n) \cdot f(x_{n-1}|x_n) \cdot ... \cdot f(x_1|x_2, ..., x_n) \tag{2.10}$$

From Sklar's theorem (equation 2.1) for densities in the bivariate case follows:

$$f(x_1|x_2) = c_{1,2}(F_1(x_1), F(x_2)) \cdot f_1(x_1) \tag{2.11}$$

and, adding a conditional x_3:

$$\Rightarrow f(x_1|x_2,x_3) = c_{1,2|3}(F_1(x_1|x_3), F(x_2|x_3)) \cdot f_1(x_1|x_3) \tag{2.12}$$

$$\Rightarrow f(x_1|x_2,x_3) = c_{1,2|3}(F_1(x_1|x_3), F(x_2|x_3)) \cdot c_{1,2}(F_1(x_1), F(x_2)) \cdot f_1(x_1) \tag{2.13}$$

for conditional densities [24].

In this manner, each density distribution on the RHS of equation 2.10 can be decomposed into its copula-marginal pair, using:

$$f(x|v) = c_{x,v_j|v_{-j}}(F(x|v_{-j}, F(v_j|v_{-j}) \cdot f(x|v_{-j}) \tag{2.14}$$

with v being a d-dimensional vector, where v_j is one component of v, and v_{-j} is v excluding v_j [24]. Note that this decomposition is not unique, which will become relevant in the selection of the R-Vine.

To find the conditional CDFs in equation 2.13, the copula distribution can be used [48]:

$$F(x|v) = \frac{\partial C_{x,v_j|v_{-j}}(F(x|v_j), F(v_j|v_{-j}))}{\partial F(v_j|v_{-j})} \tag{2.15}$$

Vine trees are a method to keep track of the dependencies between different copulas and inputs. Introduced by Bedford and Cooke [23] they are a graphical model that uses a nested set of trees which are defined by the distributions on their nodes and the copulas on their edges. The first tree's edges are the nodes of the second tree, and the second tree's edges are the nodes of the third tree, and so on. Regular vine trees are a subset of vine trees were two edges in tree j are only joined to an edge tree $j+1$ if these edges share a common node in tree j. Formally, they are a set of trees $\mathcal{V} = (T_1, ..., T_{d-1})$ where:

1. Each tree $T_j = (N_j, E_j)$ is connected, N_j being the nodes and E_j the edges.
2. T_1 is a tree with node set $N_1 = 1, ..., d$ and edge set E_1.
3. For $j \geq 2$, T_j is a tree with node set $N_j = E_{j-1}$ and edge set E_j.

4. For $j = 2, ..., d-1$ and $a, b \in E_j$ with $a = (a_1, a_2)$ and $b = (b_1, b_2)$ it must hold that there is one common node betewen the edges: $\#(a \cap b) = 1$ (proximity condition) [25].

The R-Vine tree structure becomes an R-Vine copula by defining the relationship between the trees using copulas: for every edge, the copula between the nodes is defined, and the nodes of the next tree are generated using the conditional copula as in equation 2.15 [49, 50]. R-Vine copulas specifically have the advantage that the copula needed for the nodes in tree $j+1$ is already present in tree j and thus does not need to be recomputed [51].

Searching for the optimal R-Vine is difficult, as there exist $\frac{n!}{2} \cdot 2^{\binom{n-2}{2}}$ possible R-vines for an n dimensional multivariate copula [52]. Thus, sequential algorithms have been developed to estimate the R-Vine tree by tree. Since the first trees are estimated with the highest precision, Dißmann et al. [25] estimate the R-vine from the first to the last tree, maximizing the dependence that is represented in the copulas within each tree. They maximize the dependence by finding the maximum spanning tree using Kendall's τ of each node-pair as a dependency measure [25]. For each tree, the choice of copula families and parameter estimation occurs simultaneously before moving on to the next tree [25]. [25]'s algorithm is the most common in practical applications [51]. [53] estimates R-Vines from the last tree to the first, minimizing dependence within the tree. A comparison by Czado et al. [54] found neither of the approaches to be favorable for all tested datasets. Bayesian approaches have also been developed for R-Vine selection [55], but are out of the scope of this thesis.

Chapter 3

Methodology

This section introduces the model and evaluation measures for both the bivariate CM Flow and the multivariate CM Flow.

3.1 Bivariate CM Flows

3.1.1 Model

Normalizing Flows generally model the distribution of the input dataset. Thus, when training a normalizing flow directly on a joint distribution, it would reproduce this joint distribution. For CM Flows, we are interested in modeling the copula of the input joint distribution, which requires a unique architecture of combined flows, namely two marginal flows and a copula flow.

The **marginal flows** are deep dense sigmoidal flows (DDSFs) [28], $m_{\theta_i}^{(i)} : \mathbb{R} \to \mathbb{R}, i = 1,2$ with a normally distributed prior. The outputs of each marginal flow are concatenated to form the bivariate marginal flow $m_\theta : \mathbb{R}^2 \to \mathbb{R}^2$:

$$m_\theta(x_1,x_2) = \left[m_{\theta_1}^{(1)}(x_1), m_{\theta_2}^{(2)}(x_2)\right]^T \tag{3.1}$$

with $\theta = (\theta_1, \theta_2)$ signifying the parameters of the marginal flows.

The **copula flow** uses a RealNVP [27] $h_\eta : \mathbb{R}^2 \to \mathbb{R}^2$ for density estimation and a transformation function Φ to project the prediction onto the unit square, with η signifying the parameters of the model. Plausible candidates for this function are the sigmoid function, as in Wiese et al. [2], and the Gaussian CDF.

The **CM Flow** is defined as the concatenation of the bivariate marginal flow and the copula flow: $g_{\theta,\eta}(u) = m_\theta \circ h_\eta(x)$ for $x \in \mathbb{R}^2$, $g : \mathbb{R}^2 \xrightarrow{\tilde{h}} \mathbb{R}^2 \xrightarrow{m} \mathbb{R}^2$ [2]. The copula

is extracted using

$$\begin{aligned} h : \mathbb{R}^2 &\to [0,1]^2 \\ x &\mapsto \Phi \circ \tilde{h}_\eta(x) \end{aligned} \tag{3.2}$$

with $x \sim N(0,1)$. Figure 3.1 shows an overview of the construction of CM Flows: N is normally distributed two-dimensional random noise, C is the copula, C_N is a copula-like structure with normal marginals, M_1 and M_2 are the marginals. As mentioned in section 2, Normalizing Flows are trained by feeding the data into the inverse function and maximizing the log likelihood of the outputs on a prior distribution. Here, both DDSF and RealNVP are trained on a standard normal prior. Thus, m_1^{-1} and m_2^{-1} project M_1 and M_2 onto a standard normal distribution, respectively (upper diagram). $\tilde{h}$ takes this two-dimensional distribution as input and projects it onto a two-dimensional normal distribution. To generate copula samples (lower diagram), $h(x) = \Phi(\tilde{h}(x))$ projects the random noise x onto the copula space $[0,1]^2$.

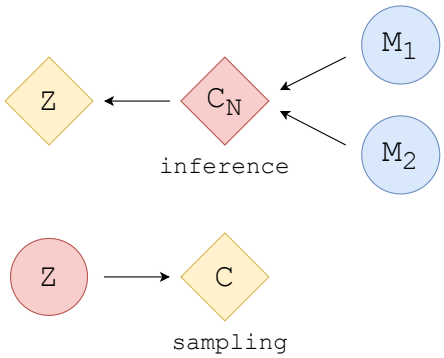


Figure 3.1: CM Flows Construction. For inference, the marginals M_1 and M_2 are projected onto normal distributions using the marginals flows, resulting in C_N. The copula flow then projects C_N onto a bivariate normal distribution Z. For sampling, the bivariate normal distribution Z is projected onto C_N and transformed using Φ.

My PyTorch [26] implementation of marginal flows is based on the NAF Github repository by Chin-Wei Huang[1], the implementation of copula flows is based on the PyTorch-flows Github repository by Ilya Kostrikov [2]. Bivariate CM Flows are also implemented in PyTorch, incorporating marginal flows and copula flows into one model. Regarding the training architecture of the CM Flows, three main possibilities emerge:

[1]NAF Github repository, `https://github.com/CW-Huang/NAF`, accessed 16. Aug. 2020

[2]PyTorch-flows Github repository, `https://github.com/ikostrikov/PyTorch-flows`, accessed 16. Aug. 2020

- **1-step model:** the two marginal flows and the copula flows are trained using a single forward pass, where the concatenated output of the marginal flows is the input to the copula flow. The backward step for the loss of each marginal flow is performed independently, retaining the gradients. The loss for the copula flow is then computed using the retained gradient of the marginal flows. One optimizer step is performed for the whole CM Flow. The best epoch is chosen by the lowest sum of losses on the validation set.

- **2-step model:** the marginal flows are trained independently first, finding the best parameters by choosing the lowest validation set loss. Then, the joint dataset is transformed using the trained marginal flows. The copula flow is then trained on the transformed dataset.

- **Pre-trained 1-step model:** The marginals flows and copula are first trained independently using the 1-step model. Then, the found weights are used to initialize the 1-step model.

Which of these architectures is most effective will be explored in sections 5.1.3 and 6.2.

3.1.2 Evaluation

While the CM flows are trained only on samples from the joint distribution, I evaluate them using known copula densities. The copula flow's objective is to minimize the difference between the true copula (C_1, C_2) and the estimated copula $(\tilde{C_1}, \tilde{C_2})) = (h_{\eta,1}(U), h_{\eta,2}(U))$ and to achieve a uniform distribution in the marginals of the copula, $U \sim U([0,1]^2)$. The similarity between the predicted and true copula is evaluated using a Monte Carlo approximation of the Jensen-Shannon (JS) divergence:

$$
\begin{aligned}
D_{JS}(p||q) =& \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2}) \\
=& \frac{1}{2}\Big[\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + \int q(x) \log \frac{2q(x)}{p(x)+q(x)}\Big] \\
=& \frac{1}{2}\Big[\mathbb{E}_{p(x)}[\log \frac{2p(x)}{p(x)+q(x)}] + \mathbb{E}_{q(x)}[\log \frac{2q(x)}{p(x)+q(x)}]\Big] \\
\approx& \frac{1}{2}\Big[\frac{1}{N}\sum_{i=1}^{N} \log \frac{2p(x_i)}{p(x_i)+q(x_i)} + \frac{1}{M}\sum_{j=1}^{M} \log \frac{2q(y_j)}{p(y_j)+q(y_j)}\Big], x_i \sim p(x), y_j \sim p(y)
\end{aligned}
\tag{3.3}
$$

with p being the predicted copula distribution, and q the true copula distribution. $D_{KL}(p|q)$ is the KL-divergence between p and q. Since the copula flow density only captures a copula-like structure with normal marginals, the density is approximated using gaussian kernels. The uniformity of the marginals of the copula is assessed with two metrics:

$$T(i,n) = \frac{1}{n} \sum_{k=1,\dots,n} \left| \log \mathbb{P}(\tilde{C}_i \in A_k) + \log n \right| \tag{3.4}$$

and

$$M(i,n) = \max_{k=1,\dots,n} \left| \log \mathbb{P}(\tilde{C}_i \in A_k) + \log n \right| \tag{3.5}$$

with $A_k = [(k-1)/n, k/n], k = 1, \dots, n$, which are both approximated using Monte-Carlo:

$$\mathbb{P}(\tilde{C}_i \in A_k) \approx \frac{1}{m} \sum_{j=1,\dots,m} \mathbb{1}^{x_j \in A_k}, x_j \sim \tilde{C}_i \tag{3.6}$$

Since DDSFs do not provide a sampling method, the marginal flows are evaluated using the average point-wise JS divergence over a grid.

3.2 Multivariate CM Flows

3.2.1 Model

The estimation of R-Vines follows a similar principle as the estimation of CM Flows. First, each marginal is transformed to a standard normal distribution using a marginal flow. The estimation of the R-Vine with uniform marginals then follows [25]:

1. The input samples are transformed into a tree structure using a maximum spanning tree algorithm that maximizes the Kendall's tau between each sample pair using Prim's algorithm [56].

2. For each edge of this tree, a conditional copula flow is estimated between the samples on each node, using a copula flow each.

3. The conditional copula is then used to generate nodes of a new graph for each edge of the current tree.

4. Steps 2 and 3 are repeated until the resulting graph has only one node.

The conditional copula is estimated by a RealNVP [27] $h_{\eta,a} : \mathbb{R} \to \mathbb{R}$, with $a \in \mathbb{R}$ serving as conditional input. It is the same model as the unconditional copula flow in the bivariate CM Flow, where one marginal is given as conditional input while the other marginal is the target data. This means that during sampling and density estimation, the conditional input information is available. Dinh et al. [27] show that RealNVPs work well when conditioned on classes, the performance of the conditional copula flow with continuous conditional inputs will be explored in section 5.2.1. On an edge $e = (e_1, e_2)$, with nodes e_1 and e_2, the conditional node is chosen as the node that also connects to one or more other edges. The other node is the unconditional node on which the conditional copula flow is trained. The new node from this edge is then transformed using the conditional flow $h_{\eta,a}(b)$.

For copula sampling, samples from a d-dimensional standard normal distribution are transformed step-by-step: beginning at the smallest tree, for each node, the corresponding dimension of the standard normal distribution samples is transformed using the inverse transformation of the respective conditional copula flow: $h_{\eta,u_i}(u_j) = \hat{u_j}$, where u_j is the transformed marginal and u_i is the conditional marginal for the respective conditional copula.

The model is implemented using PyTorch [26] for the marginal flows and copula flows and the Python library NetworkX [57].

3.2.2 Evaluation

The sampled copula is evaluated using the same procedure as for the bivariate flows: by comparing the JS divergence between the target copula and the estimated copula when training on joint samples. The JS divergence is approximated using Monte-Carlo approximation, as in equation 3.3.

Chapter 4

Experiments

This section explains which experiments are performed in order to evaluate both bivariate and multivariate CM Flows, including a description of the generated datasets.

4.1 Bivariate CM Flows

The effectiveness of CM Flows depends on three major factors: the ability of the marginal flow to model the marginal distributions, the ability of the copula flow to model the copula distribution and the effective training of the CM Flow. Thus, I first perform experiments on the copula flows and marginal flows individually, to assess their suitability as parts of the CM Flow. Then, I investigate which CM Flow architecture performs best by evaluating the marginal fit and the uniformity of copula marginals. After choosing the 1-step architecture, I evaluate the JS divergence of the estimated copula to the true copula distribution and compare the result to the JS divergence achieved by a parametric model with false copula family assumptions.

For all experiments, the dataset consists of 10,000 observations, and is split into training, validation and test set using a $80\%/10\%/10\%$ split. For an accurate estimation of the JS divergence, additional 100,000 test set samples are generated. The models are trained using the ADAM optimizer with a learning rate of $1e{-}04$ and otherwise default parameters [58]. Each model is trained for 50 epochs, using batch size 100. The best epoch is chosen by lowest validation error.

4.1.1 Univariate density estimation using marginal flows

The marginal flow is evaluated on five different distributions: Gaussian, Gaussian mixture distribution, uniform distribution, gamma distribution and lognormal distribution[1]. To find hyperparameters that work well on all tested distributions, I perform a simple grid-search over the suggested hyperparameters by Huang et al. [28] [2] In addition, I perform a random search [59] over a larger search space. Optimization hyperparameters such as the learning rate and weight decay are not evaluated, since they highly depend on the dataset and are not in focus here. They will thus remain constrant throughout the experiments. Given the best hyperparameters, I then further analyze the predicted distributions using point-wise JS divergence.

4.1.2 Copula estimation using copula flows

The copula flow is evaluated on its ability to fit samples from the Clayton, Frank and Gumbel copula, respectively. Copula samples are generated using the conditional distribution method, and then expanded to $\mathbb{R}$ using Φ^{-1}, which is the logit function or the inverse Gaussian CDF. Similar to the marginal flow experiments, I first perform a grid search over the hyperparameters of the flow, following suggested hyperparameters by Dinh et al. [42] and then an additional random search on a larger search space [3].

4.1.3 Joint distribution estimation using bivariate CM Flows

To find out which architecture is most effective for the CM Flow, I perform experiments using joint distributions with Clayton, Frank and Gumbel copulas and bimodal gaussian marginal distributions. The joint samples are generated by first sampling from the respective copula distribution and then transforming each of the marginals to a gaussian mixture marginal. The marginals are transformed by randomizing the origin Gaussian distribution of each sample and transforming it via inverse probability transform.

In each experiment, the marginal and copula flow use the hyperparameters found earlier: the marginal flow is kept at 10 flow layers and 1 hidden layer with 128 hid-

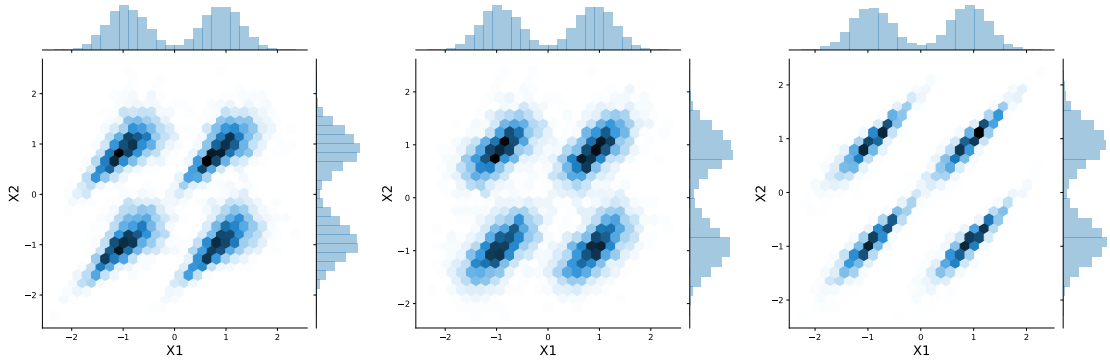
[1] The parameters for the distributions are: 1. $N(-2,3)$ 2. Gaussian mixture containing 50% $N(2,2)$ and 50% $N(12,2)$ 3. $U(-1,3)$ 4. $\Gamma(\alpha = 5)$ 5. $LN(0,-1)$

[2] The tested hyperparameters include: 1. Flow layers: 5 and 10, 2. Number of hidden layers: 1 and 2, 3. Number of hidden units: 64 and 128, 4. Number of sigmoid units: 64 and 128.

[3] The tested hyperparameters are the type of transformation function Φ (sigmoid function or gaussian CDF), the number of invertible blocks (4, 8 or 16) and the number of hidden units (32, 64, 128)

den units, with 16 sigmoid units and 2 sigmoid layers. The copula flow is kept at 8 invertible blocks and 32 hidden units.

To compare CM Flows to a parametric model under false assumptions, I also perform three experiments of a parametric copula fit, using false copula assumptions. The fit is performed using the *copulae* package, based on *scipy.optimize.minimize*. The dataset is the same as for the CM Flows, with Gaussian mixture marginals and the Clayton, Frank and Gumbel copula, respectively. For the Clayton copula, a Gumbel copula was assumed, for the Frank and Gumbel copula a Clayton copula was assumed. The CM Flow is evaluated on the three copula types, using the Gaussian mixture marginals in every experiment. Figure 4.1 shows the input datasets.



(a) Clayton (b) Frank (c) Gumbel

Figure 4.1: Input dataset for bivariate CM Flows experiments, with a Clayton, Frank and Gumbel copula respectively and Gaussian mixture marginals. The marginals are shown at the top and right side of each plot.

4.2 Multivariate CM Flows

The successful estimation of R-Vines depends on the success of the copula flows in estimating both the conditional and unconditional copula, on the marginal flow on estimating the marginal distributions, and on the tree selection algorithm to select suitable trees within the R-Vine. Due to the focus of this thesis on CM Flows, I omit an exploration of different tree selection algorithms and refer to Dißmann et al. [25] and Czado et al. [54]. The ability of marginals flows to estimate the marginal distributions, and of copula flows to estimate unconditional copulas is shown by the experiments in section 4.1. What remains is an evaluation of the estimation of conditional copulas with the copula flow and finally the evaluation of R-Vines on different datasets.

4.2.1 Conditional copula flows for conditional copulas

To test the copula flows ability to estimate conditional copulas, I perform three experiments similar to the ones on unconditional copulas in section 4.1. For Clayton, Frank and Gumbel copula respectively, I train the copula flow on samples of one marginal of the copula, given samples of the other marginal as conditional inputs. I then evaluate the JS divergence of the predicted copula to the true copula.

4.2.2 Multivariate joint distribution estimation with R-vines

To evaluate the performance of the R-Vine estimation algorithm, I estimate the R-Vine for four different datasets. Each dataset has the same R-Vine structure and marginal distributions, but different copula distributions. The first three datasets contain only Clayton, Frank and Gumbel copulas, respectively. The fourth dataset is a mixed dataset, containing both Clayton, Frank and Gumbel copulas. Figure 4.2 illustrates the R-Vine informing the datasets, table 4.1 shows the copula type for each bivariate copula in the R-vine. All datasets have Gaussian mixture marginals, as in the CM Flows experiments. The multivariate CM Flow is evaluated upon its ability to sample the correct R-Vine copula, using JS divergence.

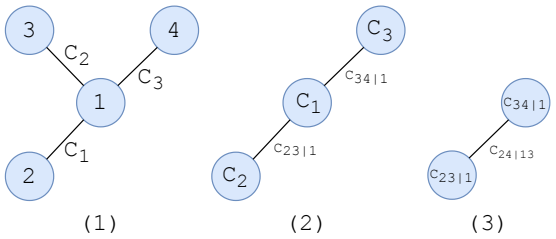


Figure 4.2: R-Vine mixed dataset structure with four dimensions, specified by three trees.

	BIVARIATE COPULA
CLAYTON	$C_1, C_{23\|1}$
FRANK	$C_2, C_{34\|1}$
GUMBEL	$C_3, C_{24\|13}$

Table 4.1: Bivariate copulas for mixed R-Vine dataset. Clayton copula uses $\theta = 2$, Frank and Gumbel $\theta = 5$.

Chapter 5

Results

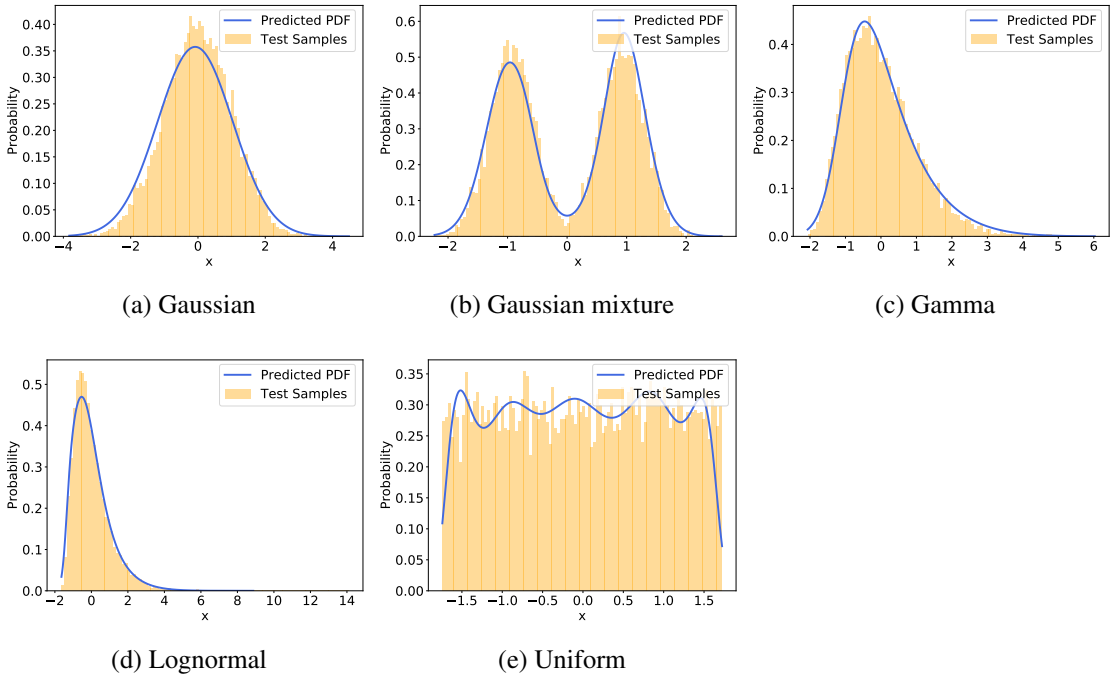
5.1 Bivariate CM Flows

5.1.1 Marginal flows

I first perform the grid and random search for the marginal flows to find the best hyperparameters for the five types of univariate distributions. As a non-parametric tool, the CM Flows should perform well without knowledge of the underlying dataset distribution. The hyperparameters can thus not be tuned to a single distribution. Rather, it is important to find a set of hyperparameters that works well for most distributions. The grid search on Gaussian, Gaussian mixture, uniform, gamma and lognormal distribution reveals that 2 sigmoid layers, 5 flow layers, 1 hidden layer, 128 hidden units and 16 sigmoid units are the best hyperparameters for the majority of distributions. The random search achieves lower NLL results on some distributions, but does not find a similar hyperparameter setting that works best for all distributions. I will thus use the grid search results for further analysis and for the CM Flows. Detailed results are reported in chapter B.1 in the appendix.

Given these hyperparameters, I evaluate the density estimation of marginal flows on the distributions using the average point-wise JS divergence of the predicted and true distributions. As expected given Huang et al. [28], the DDSF fits the distributions well and shows a very low JS divergence to the true distributions, as shown in table 5.1. For all distributions, the JS divergence remains below 0.05, indicating a good fit. Figure 5.1 supports this assertion: the DDSF shows a good density estimation on Gaussian mixture, gamma and lognormal distribution, with a slight overfit on the uniform distribution and a slight underfit on the Gaussian and lognormal distribution.

Generally, the DDSF appears to be suitable as a marginal flow.



(a) Gaussian (b) Gaussian mixture (c) Gamma

(d) Lognormal (e) Uniform

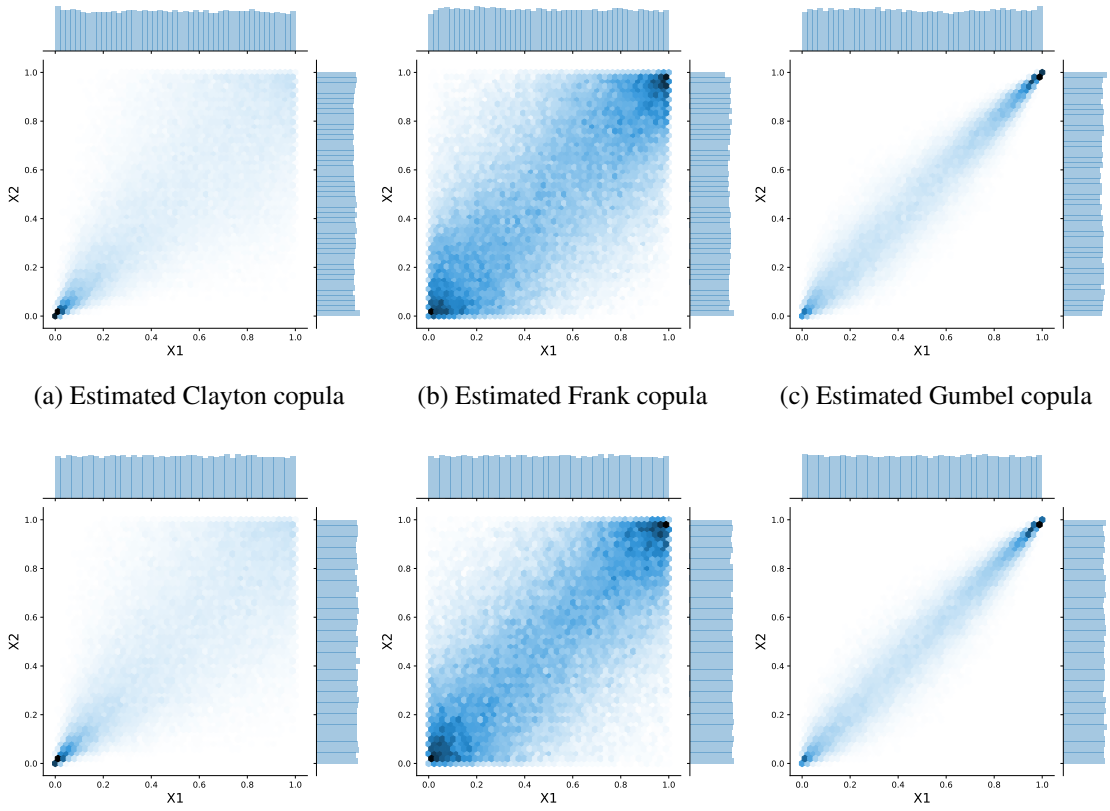
Figure 5.1: Density estimations of the trained marginal flow (blue) on a grid in comparison to samples from the test sets (yellow). Each dataset is normalized.

5.1.2 Copula flows

As with the marginal flows, a grid search and random search is performed first to find the most suitable hyperparameters for the distributions. Again, the goal is not to find hyperparameters tuned to a single copula distribution, but ones that work well on most tested copula types. The grid search finds that the Gaussian CDF and 32 hidden units are favorable for each copula, and 8 invertible blocks achieve a lower validation NLL for the Clayton and Gumbel copula. The random search also finds that the Gaussian CDF transformation is the better transformation. Details on both searches are found in chapter B.1 in the appendix. Thus, I will use a Gaussian CDF transformation, 32 hidden units and 8 invertible blocks for further analysis and the CM Flows.

Using these hyperparameters, the copula flow is evaluated on the Clayton, Frank and Gumbel copula. Given Dinh et al. [42]'s results, I expect the RealNVP to perform well on the copula samples. Table 5.1 shows that this expectation is met: the marginals of each copula appear uniform and show a low T and M metric, with values below 6e−05 for the T metric on each copula and below 8e−04 for the M metric. The copula

flow fits the Clayton copula best, with a JS divergence of only 0.04, then the Frank copula with a divergence of 0.047 and finally the Gumbel copula with a JS divergence of 0.053. Figure 5.2 shows almost no visible differences between the true and estimated copulas. Generally, the divergence is low and the marginals are uniform, showing that the RealNVP is suitable as a copula flow.



(a) Estimated Clayton copula (b) Estimated Frank copula (c) Estimated Gumbel copula

(d) True Clayton copula (e) True Frank copula (f) True Gumbel copula

Figure 5.2: The first row shows samples from the trained copula flows for each copula type. For comparison, the second row shows samples from the true copulas. The first marginal is depicted on top of the graph, the second marginal on the right side.

5.1.3 CM Flows

To find out which architecture of the CM Flow performs best, I evaluate the divergence of the true and predicted marginals of the joint distribution for the three copula types using Gaussian mixture marginals. It is not obvious which architecture will perform best: on the one hand, the 2-step model can be expected to train in a more stable way, since the marginal flows are first trained fully before transforming the data as input to

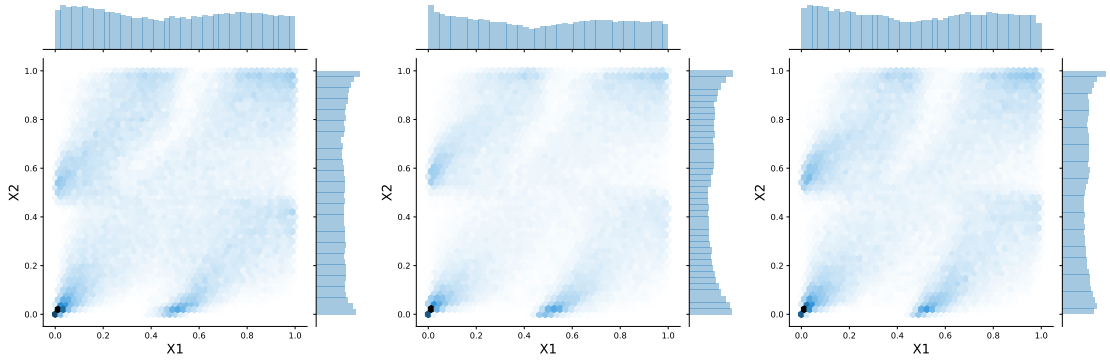
MARGINAL FLOW	$\text{JSD}(\widehat{X},X)$				
GAUSSIAN	3.12e−02				
GAUSSIAN MIXTURE	2.37e−02				
UNIFORM	2.0e−03				
GAMMA	3.71e−02				
LOGNORMAL	4.34e−02				
COPULA FLOW	$\text{JSD}(\widehat{C},C)$	T(1,25)	T(2,25)	M(1,25)	M(2,25)
CLAYTON	4.04e−02	4.96e−05	4.78e−05	8.6e−05	7.74e−04
FRANK	4.77e−02	5.25e−05	4.73e−05	1.68e−04	1.01e−04
GUMBEL	5.27e−02	3.60e−05	3.72e−05	1.31e−04	7.53e−05
COND. COPULA FLOW	$\text{JSD}(\widehat{C},C)$		T(2,25)		M(2,25)
CLAYTON	4.01e−02		6.63e−04		1.01e−04
FRANK	4.38e−02		6.64e−04		8.75e−04
GUMBEL	4.32e−02		6.66e−04		1.12e−03

Table 5.1: Marginal flow, copula flow and conditional copula flow validation set results on multiple datasets. Clayton copula uses $\theta = 2$, Frank and Gumbel $\theta = 5$. $\text{JSD}(\widehat{X},X)$ is the average point wise JS divergence between the true and estimated distribution over a grid. $\text{JSD}(\widehat{C},C)$ describes the JS divergence between the true copula and the predicted samples of the flow. T and M measure the uniformity of the marginals, as described in equations 3.4 and 3.5.

the copula flows. On the other hand, the 1-step model might result in a more robust copula flow, since it trains on outputs of different training stages of the marginals flows. The pretrained 1-step model has the benefit of initializing with the trained marginal and copula flows, but runs the danger of overfitting.

Regarding the marginal fit as seen in table 5.2, the 1-step and pre-trained 1-step model perform similarly, with a JS divergence between 0.07 and 0.09. The 2-step model performs much better, with a JS divergence between 0.03 and 0.04. This indicated a better fit to both marginals with the 2-step flow, which can be expected to result in a better estimation of the copula as well. Figure 5.3 partly supports this claim. The predicted copula of the 2-step CM Flow seems more concentrated at the lower tail, as in the ground-truth Clayton copula. However, the marginals of the 2-step CM Flow appear less uniform.

To investigate the uniformity of the marginals further, I evaluate the T and M metric for each architecture and copula type. Table 5.2 shows that the 1-step CM Flow produces predicted copula samples with much more uniform marginals than the other architecture types, showing a T metric between 5e-05 and 9e-05 compared to above 0.001 for the 2-step and pre-trained 1-step model. The choice between the 2-step and 1-step model is thus a choice between the fit of the marginals of the joint distribution and the uniformity of the predicted copula marginals. Both are important for a good fit: the marginals need to be fit to be transformed into uniform marginals, while the predicted copula marginals need to be uniform in order to result in a copula. However, the marginal fit is only an intermediate goal and thus less important than the copula fit. This means that for further investigation, I will use the 1-step model. Note that the multivariate CM Flow can not be trained in 1-step because of the sequential estimation of R-vine trees and will use the 2-step model.



(a) 1-step CM Flow (b) 2-step CM Flow (c) Pre-trained 1-step CM Flow

Figure 5.3: Comparison of the Clayton copula estimate of the CM Flow architectures. The 1-step CM Flow trains both marginal and copula flows at the same time, the 2-step CM Flow trains the marginal flows first and then trains the copula flow on the marginal flow outputs. The 1-step CM Flow with pre-training first trains the CM Flow as a 2-step model and then retrains it as a 1-step model. The marginal distributions are given at the top and right side of each plot.

It now remains to see how well the CM Flow performs on the three datasets. Table 5.3 shows that the JS divergence between predicted and true copula is low for the Clayton and Frank copula, being between 0.08 and 0.01. For the Gumbel copula, the JS divergence is higher, at 0.2. The marginals appear to be relatively uniform, with a T metic between 0.001 and 0.002 and an M metric between 0.01 and 0.03 for each dataset. Curiously, the marginals appear less uniform than in the experiment in

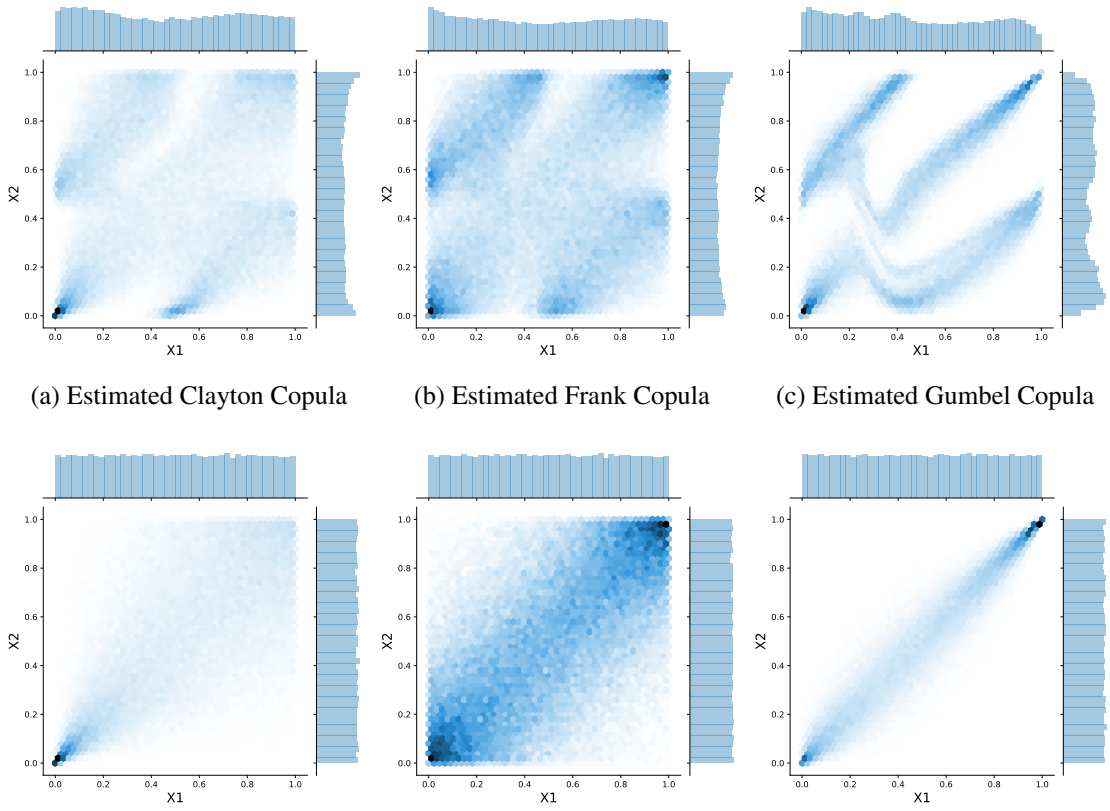
1-STEP	JSD($\widehat{X}_1, X_1$)	JSD($\widehat{X}_2, X_2$)	T(1,25)	T(2,25)	M(1,25)	M(2, 25)
CLAYTON	8.55e−02	8.51e−02	6.11e−05	5.41e−05	1.21e−04	7.53e−05
FRANK	8.55e−02	8.51e−02	5.07e−05	6.17e−05	1.36e−04	1.45e−05
GUMBEL	7.88e−02	7.90e−02	8.51e−05	7.70e−05	1.59e−04	1.64e−05
2-STEP						
CLAYTON	3.13e−02	3.62e−02	1.60e−03	1.85e−03	3.47e−03	3.47e−05
FRANK	3.04e−02	3.53e−02	1.4e−03	1.2e−03	2.43e−03	2.43e−03
GUMBEL	3.75e−02	3.48e−02	1.64e−03	1.5e−03	3.14e−03	2.8e−03
PRE-TRAINED 1-STEP						
CLAYTON	8.08e−02	8.09e−02	1.55e−03	1.74e−03	2.61e−03	3.47e−03
FRANK	8.08e−02	8.09e−02	1.65e−03	1.34e−03	2.97e−03	3.13e−03
GUMBEL	8.14e−02	8.14e−02	1.64e−03	1.46e−03	3.43e−03	2.61e−03

Table 5.2: Validation set evaluation results for the CM Flow trained on joint distribution samples. Clayton copula uses $\theta = 2$, Frank and Gumbel $\theta = 5$. JSD($\widehat{X}_1, X_1$) and JSD($\widehat{X}_2, X_2$) are the point-wise JS divergence between the true and estimated marginal distribution using a grid on the marginal flow. The T and M metrics are defined as in equation 3.4 and 3.5.

table 5.2, which might be related to the random seed. Figure 5.4 shows that the CM Flow places most of the weight in the correct tails, but shows patterns in the second and fourth quadrant that are not present in the underlying copula. The unexpected pattern most likely comes from the bimodal marginal distributions of the input dataset, which are not projected perfectly onto a normal distribution by the marginal flows. The marginals appear uniform for the Clayton and Frank copula and more noisy for the Gumbel copula.

To investigate the transformation of the marginals, I take a closer look at the outputs of the marginal flows. Ideally, the marginal flows should project the marginals onto a uniform distribution, leaving only the copula as the joint distribution of the marginals to remain. Figure 5.5 shows that this is not the case: although the marginals appear relatively uniform, the original marginal distribution still reflects itself in the joint distribution of the marginals. Thus, the CM Flow could benefit from a further investigation of the DDSF and alternatives, to find a better fit of the marginals.

To see whether CM Flows can be recommended to use when the underlying distri-



(a) Estimated Clayton Copula (b) Estimated Frank Copula (c) Estimated Gumbel Copula

(d) True Clayton copula (e) True Frank copula (f) True Gumbel copula

Figure 5.4: The first row shows samples from the estimated copula of the trained 1-step CM Flow for the three datasets. The second row shows samples from the true copula of each dataset. The marginal distributions are given at the top and right side of each plot.

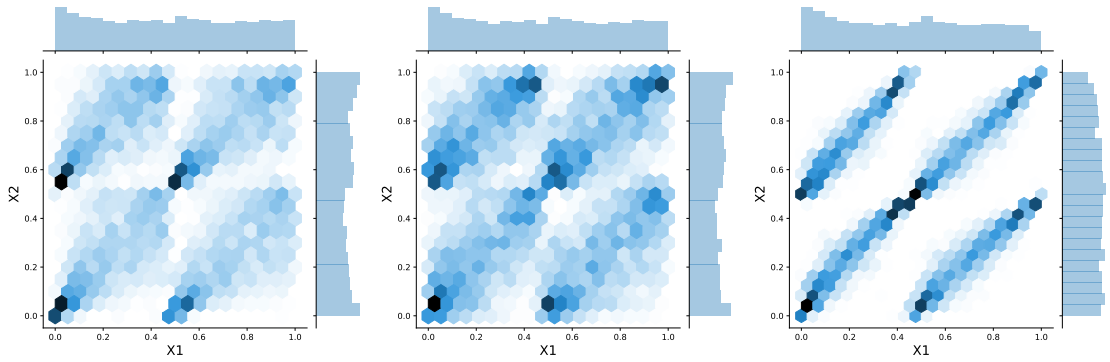
bution is unknown, I compare the CM Flow to a parametric model under false copula family assumptions. A parametric model that, for example, assumes a Gumbel copula when the true copula is a Clayton copula can never find a very close fit because of the inherent differences between Clayton and Gumbel copulas. The CM Flow does not assume a copula family, and should thus be able to find a closer fit. Table 5.3 shows that this is the case for the Clayton and Frank copula. The parametric model only achieves a JS divergence of around 0.13 for both copulas, while the CM Flow achieves 0.1 for the Clayton copula and 0.09 for the Frank copula. For the Gumbel dataset, however, the parametric model outperforms the CM Flow, with a JS divergence of 0.15 compared to 0.22 of the CM Flow. The poor performance on the Gumbel dataset most likely comes from the modes of the bimodal gaussian distribution in the second and fourth quadrant, as seen in figure 5.4. The used Gumbel copula shows a strong positive dependency be-

CM FLOW	JSD$(\widehat{C},C)$	T(1,25)	T(2,25)	M(1,25)	M(2,25)
CLAYTON	**9.96e−02**	1.28e−03	1.02e−03	2.53e−03	1.94e−03
	±4.46e−03	±4.45e−04	±3.49e−04	±8.79e−04	±6.92e−04
FRANK	**8.99e−02**	1.11e−03	1.03e−03	2.50e−03	2.00e−03
	±3.74e−03	±3.63e−04	±3.53e−04	±9.57e−04	±6.71e−04
GUMBEL	2.22e−01	1.26e−03	1.05e−03	2.70e−03	1.88e−03
	±5.59e−03	±4.11e−04	±3.75e−04	±9.59e−04	±6.67e−04
PARAMETRIC	JSD$(\widehat{C},C)$				
CLAYTON	1.29e−01				
	±5.26e−03				
FRANK	1.3e−01				
	6.84 ± e−03				
GUMBEL	**1.55e−01**				
	±8.97e−02				

Table 5.3: Test set evaluation results for the 2-step Copula Flow and the parametric model trained on joint samples. Mean and standard deviation are evaluated on 10 experiments. Clayton copula uses $\theta = 2$, Frank and Gumbel $\theta = 5$. The parametric model assumes a Gumbel copula for the Clayton dataset and a Clayton copula for both the Frank and the Gumbel dataset. JSD$(\hat{C},C)$ describes the JS divergence between the true copula and the estimated samples of the flow. T and M measure the uniformity of the marginals, as described in equations 3.4 and 3.5.

tween the variables, and thus does not have a lot of weight in the lower right and upper left corner of the graph. The placed weight by the CM Flow might cause the high JS divergence. In summary, CM Flows can outperform parametric models under false assumptions on some datasets, but do not seem to separate the marginal distributions fully from the copula.

An exploration of the CM Flows given 1,000 and 100,000 observations can be found in chapter B.2 of the appendix.



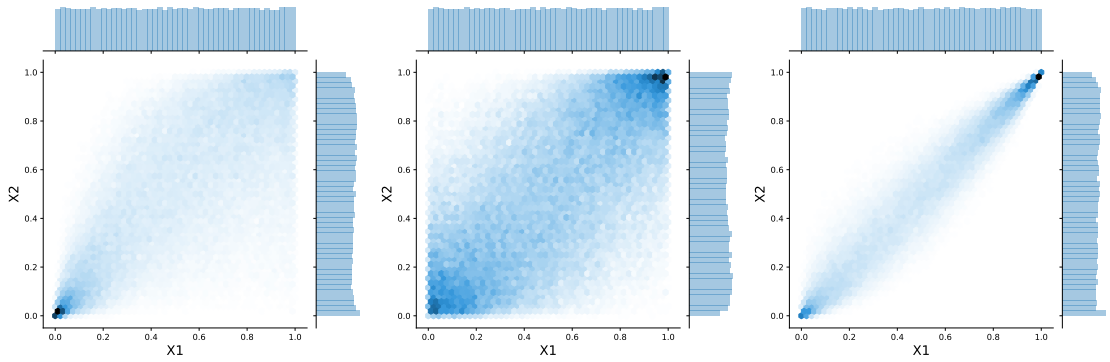
(a) Clayton Copula (b) Frank Copula (c) Gumbel Copula

Figure 5.5: Training set output of the two trained marginal flows for each copula type, concatenated and transformed to uniform marginals. The marginals are depicted on the top and right side of each plot.

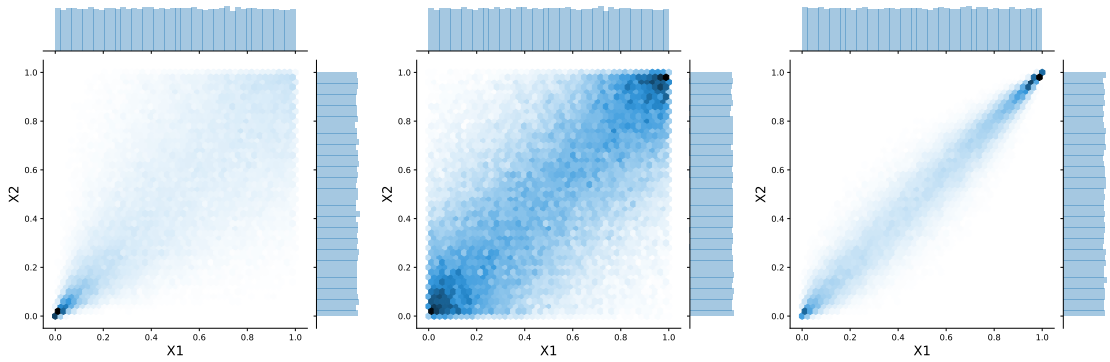
5.2 Multivariate CM Flows

5.2.1 Conditional copula flow

The multivariate CM Flow requires a conditional copula flow that can estimate one marginal of the copula given the other marginal. The conditional copula flow here is also a RealNVP [27] which received one marginal as input and the other marginal as a conditional input. In their paper, Dinh et al. [42] only report results for a class-conditional model. It is thus not obvious, how the conditional copula flow will perform, given that the conditional input is a continuous distribution. The given conditional copula flow is evaluated on samples of the three copulas using one marginal as conditional input and training it on the other. Table 5.1 shows that conditional copula flows achieve a low JS divergence, being lower than 0.05 for all tested copula types. $T(1,25)$ and $M(1,25)$ are not evaluated, since they are directly drawn from a uniform distribution. $T(2,25)$ is slightly higher than in the unconditional copula flow, being at 6e$-$04 instead of 5e$-$05. $M(2,25)$ is similar to the unconditional copula flow, being between 1e$-$04 and 0.001. Figure 5.6 confirms this: the estimated copula shows almost no noticeable difference to the true copula, while the estimated marginals are slightly less uniform than the ground truth in the lower panel. Overall, conditional copula flows seem to adequately estimate the conditional copula and we can move on to test the performance of the R-Vines.



(a) Estimated Conditional Clayton (b) Estimated Conditional Frank (c) Estimated Conditional Gumbel



(d) True Clayton copula (e) True Frank copula (f) True Gumbel copula

Figure 5.6: The first row shows the estimated conditional copula samples for Clayton, Frank and Gumbel copula. For comparison, he second row shows the true copula for each dataset. X_1 is given as conditional input, while X_2 is the target distribution. The distribution of X_1 is shown on the top side of the plot, the estimated distribution X_2 is shown on the right side.

5.2.2 R-Vines copula estimation

The performance of multivariate CM Flows is evaluated using the JS divergence of the true und predicted multivariate copula. Given that the multivariate dataset also only contains 10,000 observations, the JS divergence is expected to be higher than for the bivariate CM Flows. Multivariate CM Flows can also be expected to perform worse on the Gumbel copula, since it is build upon bivariate CM Flows, which have shown to not fit the Gumbel copula well. Table 5.4 shows that this is the case: the JS divergence is between 0.31 and 0.66 depending on the dataset. The JS divergence is lowest for the Frank copula dataset, 0.31, higher for the Clayton copula dataset with 0.39, then the mixed dataset with 0.53 and finally the Gumbel copula dataset with 0.66.

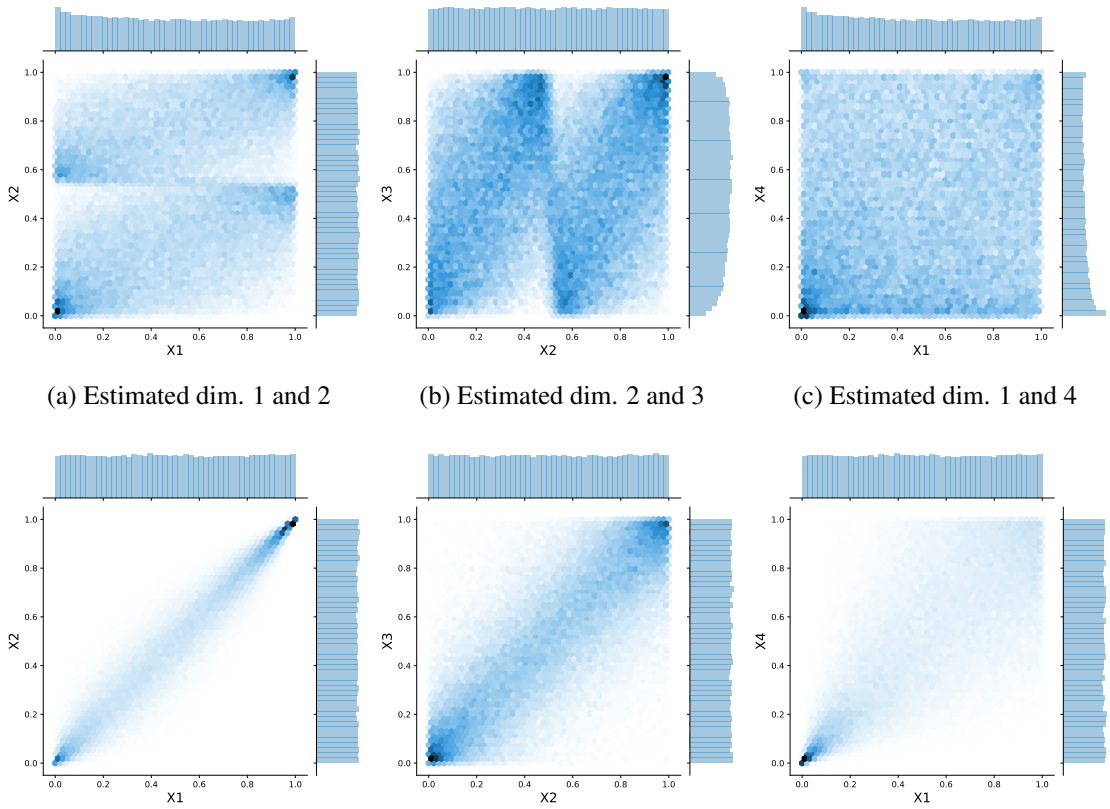
Overall, the JS divergence is quite high, indicating a suboptimal fit of the multivariate copulas. Because of this, a comparison with a parametric model under false copula family assumptions is omitted.

To further investigate why the estimated multivariate copula differs from the true copula, I take a look at some of the estimated dimensions of the multivariate copula. Figure 5.7 shows that similar to the bivariate CM Flows results, most of weight of the copula is in the right place, but that unexpected patterns appear. Here, the patterns represent just one of the bimodal gaussians, since the conditional copula flow receives one marginal as an input. Overall, this suggests that the suboptimal performance of the multivariate CM Flow rather comes from the issues the bivariate CM Flow has already shown than from the R-vine selection algorithm.

GAUSSIAN MIXTURE MARGINAL	JSD C	
CLAYTON	3.87e−01	±2.05e−02
FRANK	3.10e−01	±3.35e−02
GUMBEL	6.61e−01	±4.64e−03
MIXED	5.31e−01	±1.97e−02
UNIFORM MARGINAL	JSD C	
CLAYTON	1.24e−01	
FRANK	7.18e−02	
GUMBEL	1.11e−01	
MIXED	1.87e−01	

Table 5.4: R-Vine test set results for a 4 dimensional dataset. For the Clayton, Frank and Gumbel dataset only the Clayton, Frank and Gumbel copula respectively were used to simulate the R-Vine data. For the mixed dataset a model with a mixture of the three copulas was used. JS divergence C describes the JS divergence between the true copula and the predicted samples of the multivariate CM Flow. For Gaussian mixture marginals, mean and standard deviation are reported for 10 experiments. The uniform marginal experiment is only performed once, for comparison.

To further verify, that the suboptimal performance of the R-Vines can be attributed to the insufficient projection of the marginals flows and the difficulty of estimating the Gumbel copula of the copula flows, I test the copula estimation using only uniform marginals for the four datasets, and disable the marginal transformation. This copula



(a) Estimated dim. 1 and 2 (b) Estimated dim. 2 and 3 (c) Estimated dim. 1 and 4

(d) True dim. 1 and 2 (e) True dim. 2 and 3 (f) True dim. 1 and 4

Figure 5.7: The first row shows the estimated conditional copula of the multivariate CM Flow for the mixed dataset in three exemplary dimension pairs. The second row shows the true multivariate copula for these dimensions. The marginal distributions are given at the top and right side of each plot.

estimation should result in better results for at least the Clayton, Frank and Gumbel copula datasets, since they were estimated well in the copula flow. Table 5.4 confirms this expectation, showing a low JS divergence for all datasets. This suggests that the R-vine estimation technique can work, if the marginal flows projection is improved.

Chapter 6

Discussion

This chapter discusses the results obtained in chapter 5. First, the direct density estimation results are discussed, namely the experiments for the marginal flow and conditional and unconditional copula flows. Then, the results for the bivariate CM Flow are discussed, followed by a discussion of the multivariate CM Flow experiments.

6.1 Direct density estimation

The results on marginal flows and copula flows indicate that they work well on direct density estimation, confirming the results stated by Wiese et al. [2]. They both show a low divergence to the given true distribution, indicating a good fit.

Slight aberrations in the estimated density of the marginals flows appear for some distributions such as the Gaussian distribution and lognormal distribution, which it seems to underfit, and the uniform distribution which is seems to overfit. This and the diverse results of the hyperparameter search on each distribution show that the flows work best when they are adapted to each given dataset, but that a single hyperparameter setting can work for a multitude of distributions. Whether the flows should be tuned to each dataset at hand is a design choice that trades performance against ease of usage.

The copula flows indicate a good fit on the tested copulas, showing a JS divergence below 0.06 for each dataset. On the other side, Wiese et al. [2] show that neural networks do not model the tail sufficiently. Thus, we cannot conclude how good or bad copula flows are at estimating the tails. In practice, since observations in the tails are rare and underestimating the tails is a serious risk, tail behavior could be assumed as suggested in Wiese et al. [2].

Surprisingly, the conditional copula flows as evaluated in section 5.2 show a lower

JS divergence than the unconditional copula flows. This is unexpected, since the copula is the dependence structure between the variables, and should not be affected if one uniform marginal is given. The improved performance could derive from the fact that the flow only has to model a one-dimensional distribution, while using the other marginal as a conditional input. They have access to one marginal during sampling, and can generate the other marginal accordingly. This resembles the conditional sampling method for parametric copulas and seems to work better than creating the full copula from random noise. Another positive side effect is that one marginal will be fully uniformly distributed, since it is sampled directly from a uniform distribution. This result warrants further investigation into the trade-off between training the bivariate CM Flows with an unconditional or a conditional CM Flow.

6.2 Bivariate CM Flows

The CM flows show mixed results. On the Clayton and Frank copula datasets, they perform well and beat the parametric baseline which was given false copula family assumptions. On the Gumbel copula, the divergence between the found and true copula is higher than for the parametric model which assumes as Clayton copula. As seen in section 5.1.3, this divergence largely results from the insufficient transformation of the marginal flows. They seem to transform the data to a mostly uniform distribution, but do not remove the bimodal Gaussian completely. Subsequently, the CM Flow learns the modes of the bimodal Gaussian and projects samples to where they were. For the Clayton and Frank copula, this is not a big problem, since they are more spread out. The Gumbel copula, however, is concentrated around the diagonal and thus any projection of data away from the diagonal increases the divergence. To further improve the CM Flows, a more thorough investigation of the marginal flows, including a larger hyperparameter search and a test of alternatives to the DDSF is needed. For example Neural Spline Flows [60] promise higher flexibility in density estimation. For higher efficiency, unconstrained monotonic neural networks [46], cubic-spline flows [61] or block neural autoregressive flows [62] could be tested.

The evaluation of the training architectures of CM Flows shows that the model trained in two steps achieves a better marginal flow fit, while the model trained in one step achieves more uniform copula marginals. The former can be explained by the fact that the 2-step CM Flow trains each marginal flow independently of the copula flow. Thus, the resulting marginal fit should be optimal. The latter result is more paradoxical.

Since the marginal flow fit is worse, one would assume the copula flow to result in less uniform marginals. However, the training of the copula flow within the 1-step CM Flow might be more efficient: when performing the backward step of each marginal flow, the computational graphs of both flows are retained and used for the backward step of the copula flow. The influence of both marginal flows parameters on the copula flow is thus included in the backward step of the copula flow. When performing the optimization step, the copula flow parameters are thus changed with knowledge of the marginal flow parameters, since they construct the inputs for the copula flow. This should not be an issue for the separation of marginal and copula, since the marginal flows are still optimized without knowledge of each other. One danger in the 1-step model is that the copula flow loss might crowd out the marginal flow losses when choosing the best validation loss epoch and thus choose an epoch where the copula flow performs well, but not the marginal flow. Overall, the results are not obvious as to whether one of the architectures always performs better than the other, but rather that there is a tradeoff between different strengths of the models.

6.3 Multivariate CM Flows

The experiments on multivariate CM Flows show that they are not able to capture the multivariate copula density well. The JS divergence between the true and estimated copula is between 0.31 and 0.67 depending on the dataset. As expected, the fit on Clayton and Frank copula datasets is better than on the Gumbel and mixed datasets.

The results in section 5.2 show that the suboptimal fit is most likely a propagation of the problems the bivariate CM Flow has already shown. As seen in the results on bivariate CM Flows (section 5.1.3), the marginal flows do not model the marginal density well enough to fully project it onto a uniform distribution. The results of the marginal flows inform both the estimation of the first tree as well as the estimation of the bivariate unconditional and conditional copulas in the first tree, which then further transforms the data for the second tree, and so on. Thus, a bad fit in the marginal flows propagates through the R-Vine, changing the estimation of the multivariate copula. This is confirmed by the experiments on multivariate distributions with uniform marginals: the R-Vine performs much better, indicating that much of the performance is driven by the fit the marginal flows achieve. Comparing the estimated copula to the true copula in figure 5.1 shows that the estimation does put most of the weight in the correct places, further indicating that is not the estimated R-vine tree that causes the

issue.

Another reason for the lacking performance of R-Vine is the low number of observations they are trained on. The multivariate CM Flow needs to not only transform the marginals and find bivariate copulas using only 10,000 data points, but finds the maximum spanning tree based on the transformed marginals and found copulas of the previous tree. More data might improve the transformation of the marginals and the accuracy of the found copulas, and consequently result in a more meaningful Kendall's tau, which informs the found tree.

Chapter 7

Conclusion

7.1 Summary

This thesis evaluates the ability of bivariate and multivariate CM Flows to estimate copulas given synthetic data. Marginal flows and copulas flows were first implemented and evaluated independently, then in combination as a bivariate CM Flow. Then, the CM Flow was extended on multiple dimensions using R-Vines.

As expected, for marginal flows and copula flows, the results were positive. Both flows capture the respective dataset and show a low divergence between their estimate and the ground truth. CM Flows perform a much harder task, the estimation of the copula from joint distribution samples, and are thus expected to perform worse than copula flows, which are trained directly on the copula. However, they still achieved a JS divergence of under 0.1 for the Clayton and Frank copula dataset, which outperforms a parametric model that assumes a false copula family, which achieved a JS divergence of over 0.129 for the two datasets. Only for the Gumbel copula, the CM Flows performed much worse, with a mean JS divergence of 0.22, compared to 0.15 of the parametric model. This shows that CM Flows can only be a more reliable copula estimation technique than parametric models if the performance on the Gumbel copula can be improved in the future. One starting point for an improvement of the CM Flows could be the marginal flows, which do not seem to transform the marginal distributions to uniform distributions well enough.

To evaluate the performance of multivariate CM Flows, again the divergence between the true multivariate copula and the predicted copula was evaluated. The results show a high divergence between the predicated and true copula, being 0.31 for the Frank copula dataset and 0.61 for the Gumbel copula dataset the predicated and true

copula. Largely, these issues can be attributed to the issues the CM Flows have already shown in the bivariate case. Thus, an improvement of the CM Flows can be expected to improve the multivariate CM Flow as well.

7.2 Future work

The main objective of future work would be to improve the performance of CM Flows, particularly on datasets with a strong dependence between the marginals such as the Gumbel copula. As seen in the results on CM Flows in section 5.1.3, the CM Flows do not fully separate the marginal from the copula. Approaches for improvement should thus focus on improving the marginal flow, by either performing a more extensive hyperparameter search for the DDSF, or by exchanging the DDSF by another normalizing flow, as the ones mentioned in section 6.1.

Another objective would be to enable CM Flows to become a full multivariate simulation tool rather than just a copula estimation tool. For this, the CM Flow needs to be enabled to generate joint samples and to estimate copula and joint densities. To generate joint samples, the marginal flows need to be able to generate samples. One way to enable this would be to write a sampling function for DDSFs. Another way would be to replace DDSFs by a different normalizing flow that can generate samples. The estimation of copula densities does not require a replacement of the copula flow, just an integration of the Gaussian CDF transformation into the change of variable formula. To estimate joint densities again the change of variables formula can be employed.

For the R-vines, there are also several approaches for future work. Once the performance of the CM Flows is improved, different tree selection algorithms could be tested and the overall R-vine selection algorithm could be improved, using Kraus and Czado [63] for example. To avoid the use of regular vines, the copula flow could also be changed to allow multiple dimensions.

Ultimately, CM Flows are a practical tool and could be used, to estimate copulas in empirical data. The CM Flows could for example be directly used to measure dependencies [7] or to estimate mutual information [30], or to measure dependencies [7]. Once extended to allow the simulation of joint samples, CM Flows can be used to model multivariate distributions [9, 10].

Bibliography

[1] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, May 2020. doi: 10.1109/TPAMI.2020.2992934.

[2] Magnus Wiese, Robert Knobloch, and Ralf Korn. Copula & Marginal Flows: Disentangling the Marginal from its Joint. *arXiv*, Jul 2019. URL `https://arxiv.org/abs/1907.03361v1`.

[3] Roger B. Nelsen. *An Introduction to Copulas | Roger B. Nelsen | Springer*. Springer-Verlag New York, 2006. ISBN 978-0-387-28659-4. doi: 10.1007/978-0-387-28678-5.

[4] Gal Elidan. Copulas in Machine Learning. In *Copulae in Mathematical and Quantitative Finance*, pages 39–60. Springer, Berlin, Heidelberg, Apr 2013. ISBN 978-3-642-35406-9. doi: 10.1007/978-3-642-35407-6_3.

[5] M. A. H. Dempster. *Risk Management: Value at Risk and Beyond*. Cambridge University Press, Jan 2002. ISBN 978-052178180-0.

[6] Andrew J. Patton. A review of copula models for economic time series. *J. Multivariate Anal.*, 110:4–18, Sep 2012. ISSN 0047-259X. doi: 10.1016/j.jmva.2012.02.021.

[7] Filippo Domma, Sabrina Giordano, and Pier Francesco Perri. Statistical Modeling of Temporal Dependence in Financial Data via a Copula Function. *Comm. Statist. Simulation Comput.*, 38(4):703–728, Feb 2009. ISSN 0361-0918. doi: 10.1080/03610910802645321.

[8] Ricardo de Melo e. Silva Accioly and Fernando Yassuo Chiyoshi. Modeling dependence with copulas: a useful tool for field development decision process.

Journal of Petroleum Science and Engineering, 44(1):83–91, Oct 2004. ISSN 0920-4105. doi: 10.1016/j.petrol.2004.02.007.

[9] Ur Koumba, Calvin Mudzingiri, and Jules Mba. Does uncertainty predict cryptocurrency returns? A copula-based approach. *Macroeconomics and Finance in Emerging Market Economies*, 13(1):67–88, Jan 2020. ISSN 1752-0843. doi: 10.1080/17520843.2019.1650090.

[10] Shuang Han, Yan-hui Qiao, Jie Yan, Yong-qian Liu, Li Li, and Zheng Wang. Mid-to-long term wind and photovoltaic power generation prediction based on copula function and long short term memory network. *Appl. Energy*, 239:181–191, Apr 2019. ISSN 0306-2619. doi: 10.1016/j.apenergy.2019.01.193.

[11] Song Xi Chen and Tzee-Ming Huang. Nonparametric estimation of copula functions for dependence modelling. *Can. J. Stat.*, 35(2):265–282, Jun 2007. ISSN 0319-5724. doi: 10.1002/cjs.5550350205.

[12] David Oakes. A Model for Association in Bivariate Survival Data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(3):414–422, Jul 1982. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1982.tb01222.x.

[13] David X. Li. On Default Correlation: A Copula Function Approach. *Journal of Fixed Income*, 9(4):43–54, Mar 2000. ISSN 1059-8596. doi: 10.3905/jfi.2000.319253.

[14] Y. Malevergne and D. Sornette. Testing the Gaussian copula hypothesis for financial assets dependences. *Quantitative Finance*, 3(4):231–250, Aug 2003. ISSN 1469-7688. doi: 10.1088/1469-7688/3/4/301.

[15] David M. Zimmer. The role of copulas in the housing crisis. *Rev. Econ. Stat.*, 94(2):607–620, May 2012. ISSN 0034-6535. URL `https://www.jstor.org/stable/23262091`.

[16] Richard A. Davis, Keh-Shin Lii, and Dimitris N. Politis. Remarks on Some Nonparametric Estimates of a Density Function. In *Selected Works of Murray Rosenblatt*, pages 95–100. Springer, New York, NY, Mar 2011. ISBN 978-1-4419-8338-1. doi: 10.1007/978-1-4419-8339-8_13.

[17] Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *Ann. Math. Stat.*, 33(3):1065–1076, Sep 1962. ISSN 0003-4851. URL https://www.jstor.org/stable/2237880.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets, 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets. [Online; accessed 16. Aug. 2020].

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

[20] O. Scaillet and Jean-David Fermanian. Nonparametric Estimation of Copulas for Time Series. Nov 2002. doi: 10.2139/ssrn.372142. [Online; accessed 15. Aug. 2020].

[21] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *arXiv*, Nov 2015. URL https://arxiv.org/abs/1511.06349v4.

[22] Tim Salimans, Diederik Kingma, and Max Welling. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In *International Conference on Machine Learning*, pages 1218–1226, Jun 2015. URL http://proceedings.mlr.press/v37/salimans15.html.

[23] Tim Bedford and Roger M. Cooke. Vines: A New Graphical Model for Dependent Random Variables. *Ann. Stat.*, 30(4):1031–1068, Aug 2002. ISSN 0090-5364. URL https://www.jstor.org/stable/1558694.

[24] Kjersti Aas, Claudia Czado, Arnoldo Frigessi, and Henrik Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2):182–198, Apr 2009. ISSN 0167-6687. doi: 10.1016/j.insmatheco.2007.02.001.

[25] J. Dißmann, E. C. Brechmann, C. Czado, and D. Kurowicka. Selecting and estimating regular vine copulae and application to financial returns. *Comput. Statist.*

Data Anal., 59:52–69, Mar 2013. ISSN 0167-9473. doi: 10.1016/j.csda.2012.08.010.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. http://papers.nips.cc/paper/6581-improved-variational-inference-with-inverse-autoregressive-flow.

[27] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *arXiv*, Oct 2014. URL https://arxiv.org/abs/1410.8516v6.

[28] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural Autoregressive Flows. In *International Conference on Machine Learning*, pages 2078–2087, Jul 2018. URL http://proceedings.mlr.press/v80/huang18d.html.

[29] Yadolah Dodge. *The Oxford Dictionary of Statistical Terms*. OUP Oxford, Jul 2006. ISBN 978-019920613-1.

[30] Houman Safaai, Arno Onken, Christopher D. Harvey, and Stefano Panzeri. Information estimation using nonparametric copulas. *Phys. Rev. E*, 98(5):053302, Nov 2018. ISSN 2470-0053. doi: 10.1103/PhysRevE.98.053302.

[31] Umberto Cherubini, Elisa Luciano, and Walter Vecchiato. *Copula Methods in Finance*. Wiley, May 2004. ISBN 978-0-470-86344-2. URL https://www.wiley.com/en-gb/Copula+Methods+in+Finance-p-9780470863442.

[32] Harry Joe and James Jianmeng Xu. The Estimation Method of Inference Functions for Margins for Multivariate Models. Oct 1996. doi: 10.14288/1.0225985.

[33] C. Genest, K. Ghoudi, and L.-P. Rivest. A semiparametric estimation procedure of dependence parameters in multivariate families of distributions. *Biometrika*, 82(3):543–552, Sep 1995. ISSN 0006-3444. doi: 10.1093/biomet/82.3.543.

[34] Xiaohong Chen and Yanqin Fan. Estimation and model selection of semiparametric copula-based multivariate dynamic models under copula misspecification. *J. Econometrics*, 135(1):125–154, Nov 2006. ISSN 0304-4076. doi: 10.1016/j.jeconom.2005.07.027.

[35] Bruno Rémillard. Goodness-of-Fit Tests for Copulas of Multivariate Time Series. *Econometrics*, 5(1):13, Mar 2017. ISSN 2225-1146. doi: 10.3390/econometrics5010013.

[36] Iène Gijbels and Jan Mielniczuk. Estimating the density of a copula function. *Comm. Statist. Theory Methods*, 19(2):445–464, Jan 1990. ISSN 0361-0926. doi: 10.1080/03610929008830212.

[37] Jean-David Fermanian, Dragan Radulovic, and Marten Wegkamp. Weak convergence of empirical copula processes. *Bernoulli*, 10(5):847–860, Oct 2004. ISSN 1350-7265. doi: 10.3150/bj/1099579158.

[38] Gery Geenens, Arthur Charpentier, and Davy Paindaveine. Probit transformation for nonparametric kernel estimation of the copula density. *Bernoulli*, 23(3):1848–1873, Aug 2017. ISSN 1350-7265. doi: 10.3150/15-BEJ798.

[39] Gery Geenens. Probit Transformation for Kernel Density Estimation on the Unit Interval. *J. Am. Stat. Assoc.*, 109(505):346–358, Jan 2014. ISSN 0162-1459. doi: 10.1080/01621459.2013.842173.

[40] Oren Rippel and Ryan Prescott Adams. High-Dimensional Probability Estimation with Deep Density Models. *arXiv*, Feb 2013. URL `https://arxiv.org/abs/1302.5125v1`.

[41] Peter Laurence, Ricardo J. Pignol, and Esteban G. Tabak. Constrained Density Estimation. In *Quantitative Energy Finance: Modeling, Pricing, and Hedging in Energy and Commodity Markets*, pages 259–284. Springer, New York, NY, Aug 2013. ISBN 978-1-4614-7247-6. doi: 10.1007/978-1-4614-7248-3_10.

[42] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv*, May 2016. URL `https://arxiv.org/abs/1605.08803v3`.

[43] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. pages

2338–2347, 2017. URL http://papers.nips.cc/paper/6828-masked-autoregressive-flow-for-density-estimation. [Online; accessed 15. Aug. 2020].

[44] Durk P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved Variational Inference with Inverse Autoregressive Flow. pages 4743–4751, 2016. [Online; accessed 15. Aug. 2020], http://papers.nips.cc/paper/6581-improved-variational-inference-with-inverse-autoregressive-flow.

[45] Priyank Jaini, Kira A. Selby, and Yaoliang Yu. Sum-of-Squares Polynomial Flow. In *International Conference on Machine Learning*, pages 3009–3018, May 2019. URL http://proceedings.mlr.press/v97/jaini19a.html.

[46] Antoine Wehenkel and Gilles Louppe. Unconstrained Monotonic Neural Networks. pages 1545–1555, 2019. URL http://papers.nips.cc/paper/8433-unconstrained-monotonic-neural-networks. [Online; accessed 15. Aug. 2020].

[47] Claudia Czado. Pair-Copula Constructions of Multivariate Copulas. In *Copula Theory and Its Applications*, pages 93–109. Springer, Berlin, Heidelberg, May 2010. ISBN 978-3-642-12464-8. doi: 10.1007/978-3-642-12465-5_4.

[48] Harry Joe. Families of m-Variate Distributions with Given Margins and m(m-1)/2 Bivariate Dependence Parameters. *Lecture Notes-Monograph Series*, 28: 120–141, 1996. ISSN 0749-2170. URL https://www.jstor.org/stable/4355888.

[49] Dorota Kurowicka and Roger 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.

[50] Claudia Czado. Analyzing Dependent Data with Vine Copulas. 2019. ISSN 0930-0325. doi: 10.1007/978-3-030-13785-4.

[51] Kjersti Aas. Pair-Copula Constructions for Financial Applications: A Review. *Econometrics*, 4(4):43, Oct 2016. ISSN 2225-1146. doi: 10.3390/econometrics4040043.

[52] O. Morales-Nápoles. Counting Vines. In *Dependence Modeling*, pages 189–218. WORLD SCIENTIFIC, Dec 2010. ISBN 978-981-4299-87-9. doi: 10.1142/9789814299886_0009.

[53] Dorota Kurowicka. Optimal Truncation of Vines. In *Dependence Modeling*, pages 233–247. WORLD SCIENTIFIC, Dec 2010. ISBN 978-981-4299-87-9. doi: 10.1142/9789814299886_0011.

[54] Claudia Czado, Stephan Jeske, and Mathias Hofmann. Selection strategies for regular vine copulae. *Journal de la Société Française de Statistique*, 154(1): 174–191, 2013. URL http://journal-sfds.fr/article/view/162. [Online; accessed 15. Aug. 2020].

[55] Lutz Gruber and Claudia Czado. Sequential Bayesian Model Selection of Regular Vine Copulas. *Bayesian Analysis*, 10(4):937–963, Dec 2015. ISSN 1936-0975. doi: 10.1214/14-BA930.

[56] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, Nov 1957. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1957.tb01515.x.

[57] Proceedings of the Python in Science Conference (SciPy): Exploring Network Structure, Dynamics, and Function using NetworkX, Jan 2014. URL http://conference.scipy.org/proceedings/SciPy2008/paper_2. [Online; accessed 19. Aug. 2020].

[58] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv*, Dec 2014. URL https://arxiv.org/abs/1412.6980v9.

[59] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, Feb 2012. ISSN 1532-4435. doi: 10.5555/2188385.2188395.

[60] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7511–7522, 2019.

[61] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Cubic-Spline Flows. *arXiv*, Jun 2019. URL https://arxiv.org/abs/1906.02145v1.

[62] Nicola De Cao, Ivan Titov, and Wilker Aziz. Block Neural Autoregressive Flow. *arXiv*, Apr 2019. URL https://arxiv.org/abs/1904.04676v1.

[63] Daniel Kraus and Claudia Czado. Growing simplified vine copula trees: improving Dißmann's algorithm. *arXiv*, Mar 2017. URL https://arxiv.org/abs/1703.05203v1.

Appendix A

Additional Background

COPULA	$\theta \in$	BIVARIATE COPULA $C_\theta(u,v)$
CLAYTON	$[-1,\infty)$	$[\max\{u^{-\theta}+v^{-\theta}-1;0\}]^{-1/\theta}$
FRANK	$[1,\infty)$	$\frac{1}{\theta}\log\left[1+\frac{(e^{\theta u}-1)(e^{-\theta v}-1)}{e^{-\theta}-1}\right]$
GUMBEL	$(-\infty,\infty)$	$\exp\left[-((-\log(u))^{-\theta}+(-\log(v))^{-\theta})^{(1/\theta)}\right]$

Table A.1: Definitions for Clayton, Frank and Gumbel copula.

Appendix B

Additional Results

B.1 Hyperparameter Search

This appendix includes additional result for the grid and random search of marginal flows and copula flows. Table B.1 shows the grid search results for the marginal flows. It can be seen, that 5 flow layers, 1 hidden layer, 128 hidden units, 16 sigmoid units and 2 sigmoid layers are the best hyperparameters for most datasets. Regarding the random search results in table B.2, no clear best hyperparameters emerge. The grid search results on copula flows in table B.3 show that the Gaussian CDF is the preferable transformation function over the sigmoid function for all datasets, as well as 32 hidden units, the best number of hidden blocks across datasets is 8. The random search in table B.4 finds slightly different results: while also finding the Gaussian CDF, it finds that 4 inverted blocks and 512 hidden units are preferable for two of three datasets.

B.2 Bivariate CM Flows

To see how the CM Flows' performance changes with more or less data, I perform additional experiments using datasets with 1,000 and 100,000 observations, respectively. Table B.5 shows that the CM Flows do not perform significantly worse under just 1,000 observations than under 10,000. With 100,000 observations, however, the copula marginals appear much more uniform, while the JS divergence stays similar. Unfortunately, the JS divergence of the Gumbel copula only seems to rise with more observations.

MARGINAL	FLOW L.	HIDDEN L.	HIDDEN U.	SIGM. U.	SIGM. LAYER	VAL. NLL
GAUSSIAN	5	1	128	16	2	1.29572
GAUSSIAN MIXTURE	10	1	128	8	2	1.02936
UNIFORM	5	2	128	16	2	1.22809
GAMMA	5	1	64	8	2	1.19813
LOGNORMAL	5	1	128	16	2	1.11292

Table B.1: Marginal flow grid search validation set results. Tested hyperparameters are the number of flow layers, number of hidden layers, number of hidden units, number of sigmoid units and number of sigmoid layers. The validation set NLL is reported for the best hyperparameters.

MARGINAL	FLOW L.	HIDDEN L.	HIDDEN U.	SIGM. U.	SIGM. LAYER	VAL. NLL
GAUSSIAN	8	4	1	4	1	1.25505
GAUSSIAN MIXTURE	2	1	2	4	1	1.07675
UNIFORM	1	16	64	128	4	1.24605
GAMMA	8	4	1	4	1	1.25505
LOGNORMAL	2	1	64	2	2	1.07824

Table B.2: Marginal flow random search validation set results. Tested hyperparameters are the number of flow layers, number of hidden layers, number of hidden units, number of sigmoid units and number of sigmoid layers. The validation set NLL is reported for the best hyperparameters. The search space included number of flow layers, number of hidden layers and number of sigmoid layers in $\{2^0, ..., 2^5\}$, and number of hidden units and number of sigmoid units in $\{2^0, ..., 2^{10}\}$.

	TRANSFORM.	INV. BLOCKS	HIDDEN U.	VAL. NLL
CLAYTON(2)	GAUSSIAN CDF	8	32	2.39730
FRANK(5)	GAUSSIAN CDF	16	32	2.51388
GUMBEL(5)	GAUSSIAN CDF	8	32	1.55832

Table B.3: Copula flow grid search results. Tested hyperparameters are the type of transformation function, the number of inverted blocks and the number of hidden units. The validation set NLL is reported for the best hyperparameters.

	TRANSFORM.	INV. BLOCKS	HIDDEN U.	VAL. NLL
CLAYTON(2)	GAUSSIAN CDF	4	512	2.15621
FRANK(5)	GAUSSIAN CDF	4	512	2.30680
GUMBEL(5)	GAUSSIAN CDF	64	256	1.37167

Table B.4: Copula flow random search results. Tested hyperparameters are the type of transformation function, the number of inverted blocks and the number of hidden units. The validation set NLL is reported for the best hyperparameters.The search space included sigmoid and gaussian CDF transformation, number of blocks in $2^0, ..., 2^7$ and number of hidden units in $\{2^0, ..., 2^{10}\}$.

1,000 OBS.	JSD C	T(1,25)	T(2,25)	M(1,25)	M(2,25)
CLAYTON	1.17e−01	2.08e−03	1.91e−03	3.47e−03	3.77e−03
FRANK	9.66e−02	1.99e−03	1.66e−03	3.77e−03	2.8e−03
GUMBEL	1.81e−01	1.55e−03	1.79e−03	2.8e−03	4.05e−03
100,000 OBS.	JSD C	T(1,25)	T(2,25)	M(1,25)	M(2,25)
CLAYTON	9.52e−02	4.51e−06	3.32e−06	8.43e−06	9.89e−06
FRANK	7.5e−02	4.39e−06	4.5e−06	7.53e−06	8.75e−06
GUMBEL	2.14e−01	6.56e−06	5.82e−05	1.29e−06	9.99e−06

Table B.5: Test set evaluation results for the copula flow trained on 1,000 and 100,000 copula samples. Clayton copula uses $\theta = 2$, Frank and Gumbel $\theta = 5$. JS divergence C describes the JS divergence between the true copula and the estimated samples of the flow. T and M measure the uniformity of the marginals, as described in equations 3.4 and 3.5.