Modeling Archimedean, Extreme-Value and Archimax Copulas with Neural Networks

by

Yuting Ng

Department of Electrical and Computer Engineering
Duke University

Date: October 11 2023

Approved:

Vahid Tarokh, Supervisor

Cynthia D. Rudin

Galen Reeves

Jose H. Blanchet

Youssef M. Marzouk

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University
2023

Abstract

Modeling Archimedean, Extreme-Value and Archimax Copulas with Neural Networks

by

Yuting Ng

Department of Electrical and Computer Engineering
Duke University

Date: October 11 2023

Approved:

Vahid Tarokh, Supervisor

Cynthia D. Rudin

Galen Reeves

Jose H. Blanchet

Youssef M. Marzouk

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University
2023

Copyright © 2023 by Yuting Ng
All rights reserved except the rights granted by the
Creative Commons Attribution-Noncommercial Licence

Abstract

Copulas are popular in high-dimensional statistical applications as they allow for dependence modeling with arbitrary margins. They are also used in rare event analysis where tail behaviours are important. Current successful applications of copulas however, generally focus on simplified assumptions, a risk in areas such as healthcare, safety and finance where model misspecification may lead to potential catastrophes. Furthermore, for rare event simulation and prediction, likelihood-based estimation may lead to dependencies in the tail being overlooked by dependencies in the bulk of the data. Moreover, there are still unsolved challenges related to the parameterization, estimation and sampling of copulas.

We propose neural representations to better fit data, and solve challenges related to parameterization, estimation and sampling. We focus on the classes of Archimedean, hierarchical Archimedean, extreme-value and Archimax copulas. Notably, these copulas have stochastic representations as mixture models of latent random variables. In particular, the functions that characterize these copulas may be described by expectations of the latent variables. The first neural representation is motivated by the ability of neural networks to represent expectations and function compositions. The second neural representation is motivated by the impressive ability of generative networks at sampling and the convergence of empirical expectations. To avoid repeated differentiation in computing the likelihood of observations during training, we make use of data transforms to collapse d-dimensional observations into

multiple one-dimensional transformed observations. To avoid repeated differentiation in computing the conditional distributions in conditional sampling, we use the sampling frameworks that come naturally with the stochastic representations.

We consider the probabilistic construction of Archimedean copulas as mixture models with the completely monotone Archimedean generator given by the Laplace transform of a latent random variable. We model the latent variable with a generative network and consider the empirical Laplace transform given samples of the latent variable. We compute higher-order derivatives using the properties of the Laplace transform. We modify existing Marshall-Olkin type sampling to our parameterization with generative networks. We extend our training and sampling method to an existing network representation. We also extend our method to hierarchical Archimedean copulas, subsequently recovering a richer class of copulas.

We consider the spectral decomposition of the stable tail dependence function in extreme-value copulas. We propose both network and generative representations for the latent spectral variable. Motivated by the data transform in the Pickands estimator, we transform d-dimensional observations into one-dimensional exponentially distributed transformed observations, then perform maximum likelihood estimation using the transformed observations. We sample using the connection to the spectral representation of max-stable processes.

We build on the stochastic representation of Archimax copulas with latent radial and simplex components. The challenges are estimating a flexible Archimedean generator and sampling the simplex component. We consider the more general d-monotone Archimedean generator given as the Williamson d-transform of the radial component, with connection to the decomposition of the Kendall function. We also consider the correspondence between the spectral component, simplex component and generalized Pareto copulas. We empirically validate our algorithms on high dimensional data, extrapolation to extreme and out-of-distribution detection.

Contents

Abstract **iv**

List of Tables **x**

List of Figures **xi**

Acknowledgements **xiii**

1 Introduction **1**

1.1 Modeling Copulas with Neural Networks 1

1.2 Related Work 3

1.3 Contributions of This Thesis 5

1.3.1 Modeling Archimedean Copulas with Neural Networks 5

1.3.2 Modeling Extreme-Value Copulas with Neural Networks . . . 5

1.3.3 Modeling Archimax Copulas with Neural Networks 6

1.4 Thesis Outline 7

2 Background **8**

2.1 Copulas 8

2.2 Archimedean Copulas 10

2.2.1 Completely Monotone Archimedean Generators 10

2.2.2 Nested Archimedean Generators 13

2.2.3 d-Monotone Archimedean Generators 15

2.3 Extreme-Value Copulas 16

2.3.1 Stable Tail Dependence Function 19
2.3.2 Pickands Estimator 20
2.3.3 Max-Stable Processes 22
2.4 Archimax Copulas 23
2.4.1 Stochastic Representations of Archimax Copulas 24
2.4.2 Modified Pickands Estimator 25
2.4.3 Kendall Distribution Function 26
2.4.4 Generalized Pareto Copulas 27
2.5 Performance Metrics 29
2.5.1 Log-Likelihood 29
2.5.2 Cramér-von Mises Statistic 29
2.5.3 Lambda Map for Archimedean Generators 30
2.5.4 Integrated Squared Error for STDFs 31
2.6 Additional Background 31
2.6.1 Additional Background on Copulas 31
2.6.2 Additional Background on Copulas in Machine Learning . . . 33
3 Generative Archimedean Copulas 35
3.1 Introduction 35
3.2 Modeling the Archimedean Generator 36
3.3 Training the Archimedean Copula 38
3.3.1 Training with Maximum Likelihood 38
3.3.2 Training with Goodness-of-Fit 39
3.3.3 Training with Minimum Discrepancy between Samples 39
3.4 Sampling the Archimedean Copula 40
3.5 Modeling the Nested Archimedean Generator 42

3.5.1 Training the Hierarchical Archimedean Copula 42
3.5.2 Sampling the Hierarchical Archimedean Copula 43
3.6 Experiments 44
3.6.1 Learning Different Tail Dependencies 44
3.6.2 Learning the Latent Distribution of the Archimedean Copula . 48
3.6.3 Learning High-Dimensional Archimedean Copulas 49
3.6.4 Learning Hierarchical Archimedean Copulas 49
3.7 Summary 53
4 Neural Extreme-Value Copulas 54
4.1 Introduction 54
4.2 Modeling the Stable Tail Dependence Function 55
4.3 Training with Connection to Pickands Estimator 58
4.4 Sampling with Connection to Max-Stable Processes 59
4.5 Experiments 60
4.5.1 Learning and Sampling Asymmetric Logistic Copula 60
4.6 Summary 63
5 Generative Archimax Copulas 64
5.1 Introduction 64
5.2 Training the Stable Tail Dependence Function 66
5.3 Sampling the Simplex Component 67
5.4 Training the Archimedean Generator 69
5.5 Sampling the Radial Component 75
5.6 Training and Sampling the Archimax Copula 75
5.7 Experiments 77
5.7.1 Learning the Archimedean Generator and STDF 78

5.7.2 Application to Modeling Nutrient Intake 84
5.7.3 Application to Extrapolating to Extreme Rainfall 99
5.7.4 Application to Out-of-Distribution Detection 101
5.7.5 Learning High-Dimensional Archimax Copulas 102
5.8 Summary 105
6 Open Problems **106**
7 Conclusions **108**
Bibliography **110**

List of Tables

2.1 Examples of common copulas 10

2.2 Examples of completely monotone Archimedean generators 12

3.1 Negative log-likelihoods of fitting known Archimedean copulas 46

3.2 Negative log-likelihoods of fitting real-world data 46

3.3 Negative log-likelihoods of fitting higher-dimensional data 49

4.1 IRAE of fitting stdf of asymmetric logistic copula 61

5.1 MSE of K, λ, and time taken, with $n_r = 100$ and $n_z \in \{20, \cdots, 100\}$. 79

5.2 Inference of φ given true ℓ and inference of ℓ given true φ 83

5.3 Goodness-of-fit to $17d$ nutrient intake data and $100d$ C-NSD copula . 86

5.4 Goodness-of-fit to dependence in the extremes 100

5.5 Accuracy of out-of-distribution detection 102

List of Figures

2.1 Examples of different radial envelopes 12

2.2 Hierarchical Archimedean copula 13

2.3 Examples of different asymptotic dependencies 19

2.4 Examples of different radial envelopes and asymptotic dependencies . 23

3.1 Sampling from generative Archimedean copulas 41

3.2 Samples from fitting known Archimedean copulas 47

3.3 Samples from fitting real-world data 47

3.4 Learnt latent Gamma distributions of Clayton copulas 48

3.5 Samples from learnt Archimedean copula compared to GAN 48

3.6 CPU runtimes for computing the likelihood of 3000 samples 50

3.7 Samples from fitting a hierarchical Clayton copula 51

3.8 Samples from fitting a hierarchical Archimedean copula 52

4.1 Stdfs from fitting ASL with 3000 training samples 61

4.2 Stdfs from fitting ASL with 240 training samples 61

4.3 Samples from fitting ASL with 3000 training samples 62

4.4 Samples from fitting ASL with 240 training samples 62

5.1 Flow chart of iterative stdf and Archimedean generator updates . . . 66

5.2 Estimates of λ with $n_r = 100$ and $n_z \in \{20, \cdots, 100\}$ 79

5.3 MSE of K, λ, and time taken, with $n_r = 100$ and $n_z \in \{20, \cdots, 100\}$. 80

5.4 Estimates of λ given 200 samples of $\mathbf{S}$ from true stdf 81

5.5 Estimates of λ given 1000 samples of $\mathbf{S}$ from true stdf 82
5.6 IRAE of fitting ℓ given true φ and time taken 84
5.7 Selection of block size using test for max-stable property 86
5.8 Samples from generative Archimax copula 87
5.9 Samples from Gaussian copula 88
5.10 Samples from R-vine copula 89
5.11 Samples from C-vine copula 90
5.12 Samples from D-vine copula 91
5.13 Samples from Archimedean copula 92
5.14 Samples from hierarchical Archimedean copula 93
5.15 Samples from extreme-value copula 94
5.16 Samples from Clayton-Archimax copula 95
5.17 Samples from Wasserstein GAN with gradient penalty 96
5.18 Samples from masked autoregressive flow 97
5.19 Samples from variational autoencoder 98
5.20 Results from extrapolating to dependence in the extremes 100
5.21 Samples from likelihood-based out-of-distribution detection 102
5.22 Samples from fitting $100d$ Clayton-NSD copula 103
5.23 Samples from fitting $100d$ Clayton-NSD copula (continued) 104

Acknowledgements

This thesis would not have been possible without my adviser, Professor Vahid Tarokh, for his patient guidance, especially the encouragement and support to establish strong fundamentals. In addition, I am really fortunate to be part of the team with Vahid and Ali, where work is always exciting and there is an air of trust and appreciation. I also enjoy the company of my labmates and classmates, including Marko, Khalil, Suya, Yanping, Cat, Haoming and many others, who made learning and research fun. I would like to thank my thesis committee, Dr. Cynthia Rudin, Dr. Galen Reeves, Dr. Jose Blanchet and Dr. Youssef Marzouk for taking the time to read my thesis, attend my defense and for providing me with suggestions. I would also like to acknowledge support from Duke University and DSO National Laboratories Singapore. Finally, I would like to thank my family for love and understanding throughout my many years of graduate study.

1

Introduction

1.1 Modeling Copulas with Neural Networks

Copulas are a special class of cumulative distribution functions (CDFs) that model the dependencies between multiple random variables in isolation from their marginals [68, 98]. Modeling dependencies between random variables is a central task in statistics, and understanding the dependence between covariates throughout the distribution is important in characterizing a distribution outside of its areas of highest densities. Moreover, since copulas can be used to represent CDFs, they are particularly useful in situations where tail events are important, such as ensuring algorithms are robust to observations that are infrequent but have devastating effects.

Copulas have found applications in many areas, including hydrology [39], healthcare [95] and finance [20]. In machine learning, copulas have been used to create new distributions, increasing the flexibility of modeling multivariate dependencies [21, 30, 32, 60, 70, 79, 102, 117, 128]. The utility of copulas can be attributed to their powerful representation capabilities, ease of use and intuitive decomposition into marginals and a dependence function.

However, many challenges related to parameterization, estimation and sampling are still unsolved. As CDFs of dependent uniform random variables, copulas need to satisfy properties which lead to constraints in parameterization. In addition, repeated differentiation to get densities and conditional distributions for likelihood-based estimation and conditional sampling may run into computational difficulties. Current successful applications of copulas have largely been limited to a few basic parametric families or low dimensions, preventing their widespread adoption in settings where the dependence is complicated or the dimension is large.

We propose new techniques for modeling, training and sampling copulas with neural networks. Specifically, we consider the classes of Archimedean, hierarchical Archimedean, extreme-value and Archimax copulas.

The Archimedean copula is popular for its simplicity and ability to represent different tail behaviours. The hierarchical Archimedean copula relaxes the exchangeability assumption of the Archimedean copula. The extreme-value copula is the limit of copulas of component-wise maximas, and is thus a tool motivated by extreme-value theory for modeling rare events where observations are limited. The Archimax copula models and allows extrapolation of bulk to extremes.

Notably, these copulas have convenient stochastic representations as mixture models of latent random variables. In particular, the functions that describe these copulas may be characterized by expectations of the latent variables. In addition, sampling is straightforward given samples of the latent distributions.

We consider both network and generative representations. Neural networks are expressive, can represent expectations of latent variables and compositions of functions. Their flexible parameterization also aids in satisfying functional constraints necessary to uphold the properties of the copula. Generative networks are impressive at sampling, and considering the empirical expectations of latent variables modeled by generative networks bypasses many functional constraints. In addition, samples

of the latent distribution is necessary for effective sampling of the full distribution. The last piece of the puzzle is training the models in high dimensions.

1.2 Related Work

Generative modeling is a major task in machine learning, with techniques such as generative adversarial networks (GANs) [24], normalizing flows (NFs) [104], and variational autoencoders (VAEs) [74] being the major developments for representing complex distributions. However, these techniques are largely used for modeling the bulk of a distribution and may not extrapolate well to out-of-distribution samples or samples within the tails [80]. In addition, the above methods are mainly used for sample generation, and mainly do not provide a way of quantifying the dependencies between multiple random variables in terms of CDFs and conditional CDFs.

Copulas have been applied in machine learning wherein techniques from machine learning have been used in conjunction with traditional copula theory to model more general classes of densities, examples include copula processes [128], copula Bayesian networks that use copulas to model conditional densities [32], cumulative distribution networks that use products of copulas [60], and neural likelihoods via CDFs that use copulas to join margins parameterized by neural networks [21] and many others [21, 30, 70, 79, 90, 102, 117]. Most of the above works used the Gaussian copula or an autoregressive dependence. With regards to the construction of new Archimedean, hierarchical Archimedean, extreme-value and Archimax copulas in machine learning, there is only one related work [83]. Ling et al. [83] parameterized the Archimedean generator as a neural network, but ran into computational difficulties and thus demonstrated results only for dimensions lower than or equal to three [83].

Our work builds on related work in classical copula modeling. Our part of work on Archimedean copulas relies on the Laplace transform and mixture representation [92]. Our part of work on hierachical Archimedean copulas relies on satisfying sufficient

nesting conditions with Lévy subordinators [51]. Our part of work on extreme-value copulas borrows from the Pickands estimator that transforms a d-dimensional observation into a one-dimensional exponentially distributed transformed observation [105], the spectral decomposition and the spectral representation of max-stable processes [27, 62]. Our part of work on Archimax copulas is rooted in the stochastic representation of the Archimax copulas [18], generalized Pareto copulas [35], the more general Williamson d-transform [96] and the modeling of latent variables with discrete supports [38, 42]. In addition, we reference the state-of-the-art method in estimating Archimax copulas [19].

However, there remain challenges to be solved. Related work on the mixture representation of Archimedean copulas are limited to cases of known distributions that can be sampled and for which the Laplace transform can be calculated, since it is often challenging to find and sample from a distribution corresponding to arbitrary Laplace transforms [54, 94]. Related work on the hierarchical Archimedean copula combine one-parameter families of Archimedean copulas, usually in a homogeneous manner, where all components are from the same family. It is challenging to combine Archimedean copulas from different families due to the nesting conditions. For example, the Clayton and Gumbel copulas are not compatible for nesting [94]. Thus, related work on heterogeneous Archimedean copulas have resulted in limited combinations of different families [48, 54, 94, 101, 112]. Classical methods of estimating extreme-value copulas do not necessarily satisfy functional properties. In addition, sampling from the estimated extreme-value copula is an unsolved challenge. With regard to Archimax copulas, the method in Chatelain et al. [19] assumes known or simplified radial dependence. In addition, past applications of Archimax copulas were of dimensions lower than or equal to three [4, 19, 95].

1.3 Contributions of This Thesis

1.3.1 Modeling Archimedean Copulas with Neural Networks

Current successful applications of Archimedean copulas are either single-parameter to use the method of inverse Kendall tau or low dimension because of difficulties in computing the density. We consider the completely monotone Archimedean generator given by the empirical Laplace transform of latent variables modeled with generative networks.[1] We use the properties of the Laplace transforms to compute higher-order derivatives in likelihood-based training. We sample using Marshall-Olkin type sampling given samples of the latent variable. We extend our method to an existing network representation. This estimation and sampling method allows for computational efficiencies, especially in high dimensions.

Hierarchical Archimedean copulas partially address the drawback that Archimedean copulas are exchangeable. Satisfying sufficient nesting conditions of nested Archimedean generators however is a challenge. We use a construction based on Lévy subordinators [51]. We extend our method of modeling the Laplace transform in Archimedean generators to modeling the Laplace transform in the Laplace exponent of nested Archimedean generators.

1.3.2 Modeling Extreme-Value Copulas with Neural Networks

Current non-parametric estimators of extreme-value copulas do not necessarily satisfy functional constraints of the stable tail dependence function (stdf). In addition, sampling from the learnt copula is an unsolved challenge. We model the stdf using its spectral decomposition, where the distribution of the latent spectral variable is encoded either as weights in a neural network or as outputs of a generative net-

[1] The part of work on Archimedean copulas was co-authored with Ali Hasan, Khalil Elkhalil and Vahid Tarokh [99].

work.[2] To avoid high-order derivatives in computing the likelihood, we make use of the Pickands transform to collapse d-dimensional observations into multiple one-dimensional exponentially distributed transformed observations with rates given by the stdf. We then train the networks using the likelihood of transformed observations. We sample using the connection to the spectral representation of max-stable processes.

1.3.3 Modeling Archimax Copulas with Neural Networks

Archimax copulas effectively balance the representation of data within the bulk of the distribution while extrapolating to the tails by combining the tractability of Archimedean copulas with the tail properties of extreme-value copulas. Rather than separating the two as is typically done in practice, incorporating additional information from the bulk may improve inference of the tails, where observations are limited. In addition, Archimax copulas remove the simplified symmetry assumption among covariates that is present in Archimedean copulas and the max-stable property of extreme-value copulas. The use of Archimax copulas has resulted in better fit to data in applications such as healthcare [95] and hydrology [4, 19]. However, existing computational methods do not allow feasible inference and sampling for Archimax copulas, preventing their widespread use.

Building on the works on Archimedean and extreme-value copulas, we propose neural representations for the Archimedean generator, stdf, and latent variables.[3] In particular, we consider the more general d-monotone Archimedean generator, associated Williamson d-transform, and decomposition of the Kendall distribution function, a probability integral transform from d-dimensional to one-dimensional transformed observations. We also consider the simplex distribution and associated

[2] The part of work on extreme-value copulas was co-authored with Ali Hasan, Khalil Elkhalil, João M. Pereira, Sina Farsiu, Jose Blanchet and Vahid Tarokh [50].

[3] The part of work on Archimax copulas was co-authored with Ali Hasan, and Vahid Tarokh [100].

generalized Pareto copulas. We put these methods together for a viable estimation and sampling algorithm. We empirically verify the method in high dimensions, show improved tail extrapolation and out-of-distribution detection over existing generative modeling techniques.

1.4 Thesis Outline

Chapter 2 covers necessary background. It provides formal definitions and stochastic representations for copulas, Archimedean, hierarchical Archimedean, extreme-value and Archimax copulas. Chapter 3 focuses on neural models for the completely monotone Archimedean generator, related to Laplace transforms and mixture models. It then generalizes to nested Archimedean generators. Chapter 4 focuses on neural models for the stable tail dependence function, related to spectral decompositions and max-stable processes. In addition, it borrows from the Pickands estimator to train on the likelihood of transformed observations. Chapter 5 builds on the stochastic representation of Archimax copulas with radial and simplex components. It considers the more general d-monotone Archimedean generator and generalized Pareto copulas for inference and sampling. In addition, it experimentally validates modeling in high dimensions, tail extrapolation and out-of-distribution detection. In Chapter 6, we discuss open problems. Chapter 7 concludes this thesis.

2

Background

In this chapter, we review the background material.[1]

2.1 Copulas

Copulas are given by separating the joint dependence from the marginal distributions of a multivariate random variable. Specifically, consider

$$C(\mathbf{u}) = F(F_1^{-1}(u_1), \cdots, F_d^{-1}(u_d)), \tag{2.1}$$

$$F(\mathbf{x}) = C(F_1(x_1), \cdots, F_d(x_d)), \tag{2.2}$$

where F is a d-variate cumulative distribution function (CDF) of the random variable $\mathbf{X} = (X_1, \cdots, X_d) \in \mathbb{R}^d$, with jth univariate margin F_j and jth quantile function F_j^{-1}, and the copula C is a d-variate CDF describing the dependence between the uniform random variables $\mathbf{U} = (U_1, \cdots, U_d) = (F_1(X_1), \cdots, F_d(X_d)) \in [0,1]^d$. Copulas may be constructed from given CDFs with (2.1); a CDF may be described by a copula and univariate margins with (2.2), where in addition, if the marginals F_j are continuous, then the copula C is unique [115].

[1] Notation: random variables in uppercase, observations in lowercase, vectors in boldface letters.

It is sometimes more convenient to work with multivariate survival distribution functions (SDFs). Specifically, consider

$$\bar{C}(\mathbf{u}) = \bar{F}(\bar{F}_1^{-1}(u_1), \cdots, \bar{F}_d^{-1}(u_d)), \tag{2.3}$$

$$\bar{F}(\mathbf{x}) = \bar{C}(\bar{F}_1(x_1), \cdots, \bar{F}_d(x_d)), \tag{2.4}$$

where $\bar{F}$ is a d-variate survival function, with jth univariate survival function $\bar{F}_j$ and jth inverse $\bar{F}_j^{-1}$, and the copula $\bar{C}$ associated with $\bar{F}$ is a d-variate CDF describing the dependence between uniform random variables $\mathbf{U} = (U_1, \cdots, U_d) = (\bar{F}_1(X_1), \cdots, \bar{F}_d(X_d)) \in [0,1]^d$. In this case, $\bar{C}$ is also known as the survival copula. Analagously, copulas may be constructed from given SDFs with (2.3); an SDF may be described by a copula and univariate SDFs with (2.4), and if the marginal survival functions $\bar{F}_j$ are continuous, then the survival copula $\bar{C}$ is unique [115].

While we may construct valid copulas by satisfying functional properties of CDFs with uniform margins, i.e. for $C(\cdot, \cdots, \cdot) : [0,1]^d \to [0,1]$,

$$C(1, \cdots, 1, u_i, 1, \cdots, 1) = u_i, \tag{2.5}$$

$$C(\mathbf{u}) \leqslant C(\mathbf{v}) \text{ for } \mathbf{u} \leqslant \mathbf{v}, \tag{2.6}$$

and for $\mathbf{u} = (u_1, \cdots, u_d)$, $\mathbf{v} = (v_1, \cdots, v_d)$, $u_i < v_i$,

$$\sum_{(w_1, \cdots, w_d) \in \prod_{i=1}^d \{u_i, v_i\}} (-1)^{|i : w_i = u_i|} C(w_1, \cdots, w_d) \geqslant 0, \tag{2.7}$$

the decompositions in (2.1)-(2.4) due to Sklar [115] allow us to construct copulas using stochastic representations.

We list some examples of common copulas in Table 2.1. We appended more details on copulas and copulas in machine learning at the back of this chapter. In addition, the following textbooks and collection of works are great resources [39, 65, 66, 68, 98].

Table 2.1: Examples of common copulas

Gaussian	$\Phi_{\mathbf{R}}(\Phi^{-1}(u_1), \cdots, \Phi^{-1}(u_d))$
Vine	$\prod_{e \in \mathcal{E}(\mathcal{V})} c_{U_{e_1}, U_{e_2} \mid \{U_{e_d}\}}(F_{U_{e_1} \mid \{U_{e_d}\}}(u_{e_1}), F_{U_{e_2} \mid \{U_{e_d}\}}(u_{e_2}))$
Archimedean	$\varphi(\varphi^{-1}(u_1) + \cdots + \varphi^{-1}(u_d))$
Hierarchical Archimedean	$C_0(C_1(\cdots), \cdots, C_J(\cdots))$, $C_i \in$ Archimedean
Extreme-Value	$\exp(-\ell(-\log(u_1), \cdots, -\log(u_d)))$

2.2 Archimedean Copulas

An important class of copulas are the Archimedean copulas, popular for their simplicity and ability to model different tail behaviours. They are defined as:

Definition 1 (Archimedean copula)**.**

$$C(\mathbf{u}) = \varphi(\varphi^{-1}(u_1) + \cdots + \varphi^{-1}(u_d)), \tag{2.8}$$

where $\varphi : [0, \infty) \to [0, 1]$, $\varphi(0) = 1$, $\varphi(\infty) = 0$ *is known as the Archimedean generator and it satisfies certain monotonity constraints.*

Definition 2 (monotonicity)**.** *An Archimedean generator* φ *is called* d*-monotone, where* $d \geqslant 2$*, if it is differentiable up to order* $d-2$*, the derivatives satisfy*

$$(-1)^k \varphi^{(k)} \geqslant 0 \text{ for } k \in \{0, 1, 2, \cdots, d-2\}, \tag{2.9}$$

and $(-1)^{d-2}\varphi^{(d-2)}$ *is nonincreasing and convex. For* $d = 1$*,* φ *is called* 1*-monotone if it is nonnegative and nonincreasing. If* φ *has derivatives of all orders and* $(-1)^k f^{(k)} \geqslant 0$ *for all* k*, then* φ *is called completely monotone.*

For more background on Archimedean copulas, we refer to [40, 42, 51, 92, 96].

2.2.1 Completely Monotone Archimedean Generators

The Archimedean generators may be grouped into two categories: completely monotone and d-monotone, each with their own stochastic representations. We first discuss the completely monotone Archimedean generators.

The criteria that φ is *completely monotone*, i.e. its derivatives change signs, guarantees positiveness of the copula density [71]. The class of completely monotone φ coincides with the class of Laplace transforms of a positive random variable [8, 125].

Theorem 1 (Bernstein-Widder)**.** *φ is completely monotone iff φ is a Laplace transform of a random variable $M > 0$, i.e.*

$$\varphi(x) = \int_0^\infty e^{-sx} \mathrm{dF_M(s)} = \mathrm{E_M[e^{-Mx}]}. \tag{2.10}$$

In this case, the Archimedean copula is the survival copula of

$$\mathbf{X} = (X_1, \cdots, X_d) = (E_1/M, \cdots, E_d/M), \tag{2.11}$$

where E_j are independent and identically distributed (i.i.d.) unit exponentials, M is a latent variable with Laplace transform φ, E_j and M are independent, and the marginal SDFs are given by φ.

We see the correspondence between the Archimedean copula C, the Archimedean generator φ and the stochastic representation $\mathbf{X}$ by writing the SDF and marginal SDFs of the random variable $\mathbf{X}$ and applying Sklar's theorem (2.3).

$$\begin{aligned}
P(X_1 > x_1, \cdots, X_d > x_d) &= P(E_1/M > x_1, \cdots, E_d/M > x_d) \\
&= \int_0^\infty P(E_1 > Mx_1, \cdots, E_d > Mx_d | M = m) P(M = m) dm \\
&= \int_0^\infty P(E_1 > mx_1) \cdots P(E_d > mx_d) p(m) dm \\
&= \int_0^\infty \exp(-mx_1) \cdots \exp(-mx_d) p(m) dm \\
&= \int_0^\infty \exp(-m(x_1 + \cdots + x_d)) p(m) dm \\
&= \mathrm{E_M[\exp(-M(x_1 + \cdots + x_d))]} \\
&= \varphi(x_1 + \cdots + x_d). \\
P(X_j > x_j) &= \varphi(x_j).
\end{aligned}$$

The probabilistic construction of the Archimedean copula as a mixture model has the interpretation where $X_1, \cdots, X_d$ are conditionally independent 'survival times' given M a 'frailty' parameter. The mixture representation also leads to an efficient sampling algorithm when given φ and samples of M [92].

We list some examples of completely monotone Archimedean generators, their parameter settings for different levels of dependence and latent distributions in Table 2.2 [92, 94].

Table 2.2: Examples of completely monotone Archimedean generators

	$\varphi_\theta(x)$	$\theta_{\tau=0.2}$	$\theta_{\tau=0.5}$	latent distribution
Clayton	$(1+x)^{-1/\theta}$	0.5	2	Gamma
Frank	$-\log(1-(1-\exp(-\theta))\exp(-x))/\theta$	1.86	5.74	Logarithmic
Joe	$1-(1-\exp(-t))^{1/\theta}$	1.44	2.86	Sibuya
Gumbel	$\exp(-t^{1/\theta})$	1.25	2	Stable

The Clayton generator results in the Clayton copula that is lower tail dependent, upper tail independent; the Frank copula is symmetric in both lower and upper tails; the Joe and Gumbel copulas are lower tail independent and upper tail dependent. We give bivariate plots of different radial envelopes corresponding to the Clayton, Frank and Joe copulas in Figure 2.1.

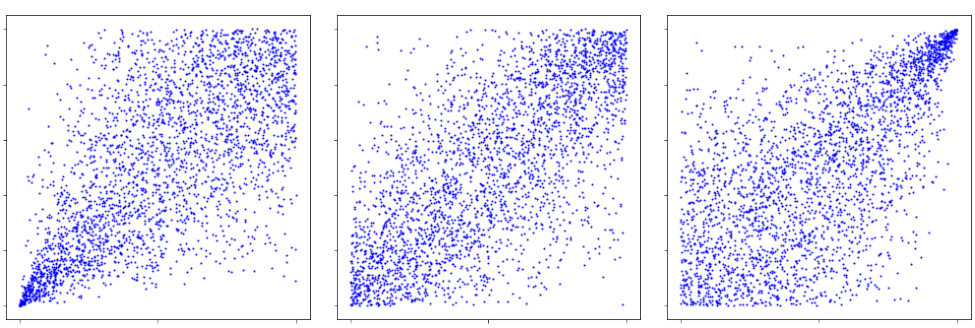

FIGURE 2.1: Examples of different radial envelopes, plots of 3000 samples on $[0,1]^2$.

2.2.2 Nested Archimedean Generators

While Archimedean copulas have been widely employed, the functional symmetry of the Archimedean copula implies exchangeability of the underlying dependence. Hierarchical (or nested) Archimedean copulas are popular for partially overcoming this drawback [67]. The name hierarchical (or nested) Archimedean copula comes from the hierarchy (or nesting) of Archimedean copulas, as shown in Figure 2.2.

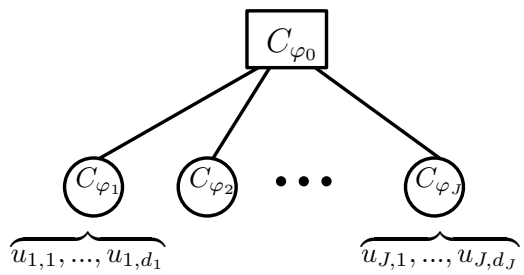

FIGURE 2.2: Hierarchical Archimedean copula with J nested Archimedean copulas.

In this case, the copula can be written as:

Definition 3 (Hierarchical Archimedean copula)**.**

$$C(\mathbf{u}) = C_{\varphi_0}(C_{\varphi_1}(u_{1,1}, \cdots, u_{1,d_1}), \cdots, C_{\varphi_J}(u_{J,1}, \cdots, u_{J,d_J})), \tag{2.12}$$

where C_{φ_j} are (possibly nested) Archimedean copulas with generators that satisfy sufficient nesting conditions: φ_0, φ_j and $(\varphi_0^{-1} \circ \varphi_j)'$ are completely monotone [67, 94].

The criteria that $(\varphi_0^{-1} \circ \varphi_j)'$ are completely monotone come from the composition of an *outer generator* φ_0 and an *inner generator* φ_j to get a completely monotone Laplace transform *nested generator* of the form $e^{-M\varphi_0^{-1} \circ \varphi_j}$, where $M > 0$ is distributed with Laplace transform φ_0 [67, 94]. This criteria was addressed in [51] using Lévy subordinators, i.e. non-decreasing Lévy processes, such as the compound Poisson process, by recognizing that the Laplace transform of Lévy subordinators at a given 'time' $t \geqslant 0$ have the form $e^{-t\psi_j}$, where the Laplace exponent ψ_j has a completely monotone derivative.

Theorem 2 (Hering-Hofert-Mai-Schrer)**.** *Given an outer generator φ_0, represent an inner generator φ_j as $\varphi_j = \varphi_0 \circ \psi_j$, where ψ_j is the Laplace exponent of a Lévy subordinator, since $(\varphi_0^{-1} \circ \varphi_j)'$ is completely monotone iff $\varphi_0^{-1} \circ \varphi_j = \psi_j$.*

Furthermore, the Laplace exponent of a Lévy subordinator has a convenient representation due to the Lévy-Khintchine theorem [111]:

Theorem 3 (Lévy-Khintchine)**.** *The Laplace exponent $\psi_j : [0, \infty) \to [0, \infty)$ of a Lévy subordinator is given as:*

$$\psi_j(x) = \mu_j x + \int_0^\infty (1 - e^{-sx})\nu_j(ds), \tag{2.13}$$

for drift $\mu_j \geqslant 0$ and Lévy measure ν_j on $(0, \infty)$.

A popular Lévy subordinator is the compound Poisson process:

$$\Lambda_j(t) = \mu_j t + \sum_{i=1}^{N_j(t)} M_{j,i}, \tag{2.14}$$

where $N_j(t)$ is distributed as a Poisson random variable with rate $\beta_j t$ and $M_{j,i}$ are i.i.d. with Laplace transform φ_{M_j} characterizing the jump sizes of the compound Poisson process. For each 'time' $t \geqslant 0$, The Laplace transform of $\Lambda_j(t)$ is given as:

$$\mathrm{E}_{\Lambda_j(t)}[\mathrm{e}^{-x\Lambda_j(t)}] = \mathrm{e}^{-t\psi_j(x)}, \tag{2.15}$$

where the Laplace exponent $\psi_j(x)$ has the following expression:

$$\psi_j(x) = \mu_j x + \beta_j(1 - \varphi_{M_j}(x)), \tag{2.16}$$

$$= \mu_j x + \beta_j(1 - \int_0^\infty e^{-sx} \mathrm{dF}_{\mathrm{M_j}}(\mathrm{s})), \tag{2.17}$$

$$= \mu_j x + \beta_j(1 - \mathrm{E}_{\mathrm{M_j}}[\mathrm{e}^{-\mathrm{M_j x}}]), \tag{2.18}$$

with drift $\mu_j \geqslant 0$, jump intensity $\beta_j > 0$ and jump size distribution determined by the Laplace transform φ_{M_j}. In addition, we may set $\mu_j > 0$ to satisfy the condition $\varphi_j(\infty) = (\varphi_0 \circ \psi_j)(\infty) = 0$ such that φ_j is a valid Archimedean generator.

Putting it together, an inner generator φ_j would be of the following form:

$$\begin{aligned}\varphi_j(x) &= \varphi_0 \circ \psi_j, \\ &= \varphi_0(\mu_j x + \beta_j(1 - \mathrm{E}_{\mathrm{M_j}}[\mathrm{e}^{-\mathrm{M_j x}}])).\end{aligned} \tag{2.19}$$

In this case, the hierarchical Archimedean copula is the survival copula of

$$\mathbf{X} = (\frac{E_{1,1}}{\Lambda_1(M)}, \cdots, \frac{E_{1,d_1}}{\Lambda_1(M)}, \cdots, \cdots, \cdots, \frac{E_{J,1}}{\Lambda_J(M)}, \cdots, \frac{E_{J,d_J}}{\Lambda_J(M)}), \tag{2.20}$$

where $E_{j,i}$ are i.i.d. unit exponentials, Λ_j are Lévy subordinators with Laplace exponents ψ_j that have jump size distributions described by Laplace transforms φ_{M_j}; $E_{j,i}$ and Λ_j are independent. The Lévy subordinators Λ_j are evaluated at a common 'time' $t = M$, where M has a distribution described by the Laplace transform φ_0. Furthermore, the marginal SDFs are given by φ_j. The probabilistic construction by combining Lévy subordinators evaluated at common 'time', leads to a well-defined hierarchical Archimedean copula with the interpretation of 'nested frailties' and an efficient sampling algorithm when given $\varphi_0, \varphi_1, \cdots, \varphi_j$ and samples of M, $\Lambda_1(M), \cdots, \Lambda_J(M)$ [51, 56].

2.2.3 d-Monotone Archimedean Generators

For a given d, φ may come from a broader class of functions than Laplace transforms [96]. In this case, the class of *d-monotone* φ coincides with the class of Williamson d-transforms of a positive random variable [96, 127].

Theorem 4 (Williamson, McNeil and Nešlehová)**.** *φ is d-monotone iff φ is a Williamson d-transform of a random variable $R > 0$, i.e.*

$$\varphi(x) = \int_x^\infty \left(1 - \frac{x}{r}\right)^{d-1} \mathrm{dF_R(r)} = \mathrm{E_R}\left[\left(1 - \frac{\mathrm{x}}{\mathrm{R}}\right)_+^{\mathrm{d}-1}\right], \tag{2.21}$$

where $(y)_+ := \max(0, y)$ *for* $y \in \mathbb{R}$.

In this case, the Archimedean copula is the survival copula of

$$\mathbf{X} = (X_1, \cdots, X_d) = (RS_1, \cdots, RS_d), \tag{2.22}$$

where R and $\mathbf{S} = (S_1, \cdots, S_d)$ are independent and known as the radial and simplex components [96]. R has Williamson d-transform φ, marginal SDFs are φ, and R dictates the radial envelope across all covariates. $\mathbf{S}$ is uniformly distributed on the unit simplex $\Delta_{d-1} = \{\mathbf{s} = (s_1, \cdots, s_d) \in [0,1]^d : \|\mathbf{s}\|_1 = s_1 + \cdots + s_d = 1\}$ and may be given as $\mathbf{S} = (E_1/\|\mathbf{E}\|_1, \cdots, E_d/\|\mathbf{E}\|_1)$ where E_j are i.i.d. unit exponentials.[2] This stochastic representation has the interpretation of a resource sharing model, where R is a resource to be distributed across $\mathbf{S}$.

2.3 Extreme-Value Copulas

The class of extreme-value copulas, as limits of copulas of component-wise maximas, is a useful tool motivated by extreme-value theory for modeling rare events where observations are limited. We begin by discussing univariate generalized extreme-value distributions to provide more context for multivariate extreme-value copulas.

Let $X_1, X_2, \cdots$ be i.i.d. random variables with distribution F. In addition, let

$$S_n = X_1 + \cdots + X_n, \tag{2.23}$$

$$M_n = \max\{X_1, \cdots, X_n\} = X_1 \vee \cdots \vee X_n. \tag{2.24}$$

If there is a sequence $a_n \in \mathbb{R}$ and $b_n > 0$, such that $(S_n - a_n)/b_n$ converges to a non-degenerate distribution, then the central limit theorem (CLT) states that the limit is a stable distribution. In addition, if X has finite variance, then the limit is a normal distribution. For example, $a_n = n\mu$, $b_n = \sqrt{n}\sigma$, $(S_n - a_n)/b_n$ converges to $\mathcal{N}(0,1)$, where μ and σ are the mean and standard deviation of X.

[2] Restricting to the subset of completely monotone φ, we have the relationship $R \overset{d}{=} \|\mathbf{E}\|_1/M$ where R has Williamson d-transform φ, M has Laplace transform φ, and $\|\mathbf{E}\|_1$ is Erlang(d) [96].

In a similar spirit, if $(M_n - a_n)/b_n$ converges to a non-degenerate distribution, where we note that M_n is increasing stochastically with n, the Fisher–Tippett–Gnedenko theorem from extreme-value theory (EVT) states that the limit is a generalized extreme-value distribution (GEV) defined, for a standardized variable $s = (x - \mu)/\sigma$ where $\mu \in \mathbb{R}$, $\sigma > 0$ are the location and scale parameters, as

$$F(s;\xi) = \begin{cases} \exp(-\exp(-s)) & \text{for } \xi = 0, \\ \exp(-(1+\xi s)^{-1/\xi}) & \text{for } \xi \neq 0, \quad \xi s > -1, \\ 0 & \text{for } \xi > 0, \quad \xi s \leqslant -1, \\ 1 & \text{for } \xi < 0, \quad \xi s \leqslant -1, \end{cases} \tag{2.25}$$

where a larger shape parameter ξ gives a heavier tail. For example, let X have the exponential distribution $F(x) = 1 - e^{-x}$, then $\mathrm{P}(\frac{\mathrm{M_n - a_n}}{\mathrm{b_n}} \leqslant \mathrm{x}) = \mathrm{P}(\mathrm{M_n} \leqslant \mathrm{a_n} + \mathrm{b_n x}) \to \exp(\mathrm{e^{-x}})$, a Gumbel distribution with $a_n = \log(n)$ and $b_n = 1$.[3]

In addition, the GEV has the location-scale property, i.e. if $\mathrm{P}(\frac{\mathrm{M_n - a_n}}{\mathrm{b_n}} \leqslant \mathrm{y}) \to \mathrm{F(y)}$, then $\mathrm{P}(\frac{\mathrm{M_n - a_n}}{\mathrm{b_n/b^*}} + \mathrm{a^*} \leqslant \mathrm{y}) = \mathrm{P}(\frac{\mathrm{M_n - a_n}}{\mathrm{b_n}} \leqslant \frac{\mathrm{y - a^*}}{\mathrm{b^*}}) \to \mathrm{F}(\frac{\mathrm{y - a^*}}{\mathrm{b*}})$. The shape of F does not change, only the point of evaluation in F changes.

While the CLT allows the application of methods that work for normal distributions even if the original variables are not normally distributed, the EVT allows the characterization of distributions in the tails even if the distribution of the original variables is not completely known due to limited observations.

In the multivariate case, let $\mathbf{X}_1, \mathbf{X}_2, \cdots$, where $\mathbf{X}_i = (X_{i1}, \cdots, X_{id})$, be i.i.d. random vectors with distribution F. In addition, let $\mathbf{M}_n = (M_{n1}, \cdots, M_{nd})$, where

$$M_{nj} = \max\{X_{1j}, \cdots, X_{nj}\}, \tag{2.26}$$

be a vector of component-wise maximas. If there are sequences $(a_{n1}, \cdots, a_{nd}) \in \mathbb{R}^d$ and $(b_{n1}, \cdots, b_{nd}) \in \mathbb{R}^d_+$ such that the limit of shifted and scaled component-

[3] For another example, let Y have the Pareto$(\alpha, 1)$ distribution, then $\mathrm{P(M_n \leqslant s) = (1-(1+s)^{-\alpha})^n}$, $\mathrm{P((M_n - a_n)/b_n \leqslant z) \to \exp(-z^{-\alpha})}$ a Fréchet$(\alpha, 1)$ distribution with $a_n = -1$ and $b_n = n^{1/\alpha}$.

wise maximas $(\frac{M_{n1}-a_{n1}}{b_{n1}}, \cdots, \frac{M_{nd}-a_{nd}}{b_{nd}})$ converges, then the limit has extreme-value dependence described by an extreme-value copula, defined as [46, 105]:

Definition 4 (Extreme-value copula)**.**

$$C(\mathbf{u}) = \exp(-\ell(-\log(u_1), \cdots, -\log(u_d))), \tag{2.27}$$

where $\ell : [0, \infty)^d \to [0, \infty)$ is known as the stable tail dependence function and it satisfies certain properties, most notably homogeneity of order 1 and d-max-decreasing.

In addition, the extreme-value copula exhibits the max-stable property,

Definition 5 (max-stable)**.**

$$C^n(x_1^{1/n}, \cdots, x_d^{1/n}) = C \textit{ for all } n \geqslant 1, \tag{2.28}$$

and a copula is an extreme value copula if and only if it is max-stable.

We see the max-stable property of the extreme-value copula by writing the CDF and marginal CDFs of the maximas and applying Sklar's theorem (2.1).

$$\begin{aligned}
P(M_{n1} \leqslant x_1, \cdots, M_{nd} \leqslant x_d) &= \exp(-n\ell(1/x_1, \cdots, 1/x_d)), \\
P(M_{nj} \leqslant x_j) &= \exp(-n/x_j), \\
C_n(\mathbf{u}) &= \exp(-n\ell(-\frac{1}{n}\log(u_1), \cdots, -\frac{1}{n}\log(u_d)), \\
&= \exp(-n\frac{1}{n}\ell(-\log(u_1), \cdots, -\log(u_d)), \\
&= \exp(-\ell(-\log(u_1), \cdots, -\log(u_d)), \\
C^n(u_1^{\frac{1}{n}}, \cdots, u_n^{\frac{1}{n}}) &= C(u_1, \cdots, u_d) \text{ (max-stable)}.
\end{aligned}$$

For more background on extreme-value copulas, we refer to [22, 46].

2.3.1 Stable Tail Dependence Function

The stable tail dependence function (stdf) dictates the asymptotic dependence between covariates, with examples in Figure 2.3.

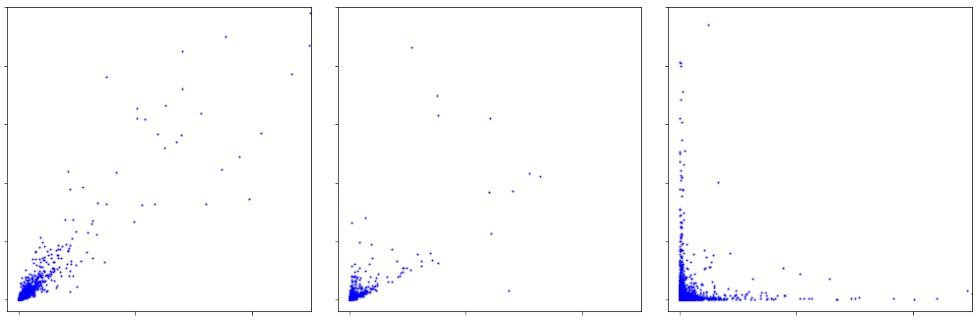

FIGURE 2.3: Examples of different asymptotic dependencies.

We describe the normalized version of the stdf, also known as the Pickands dependence function, in terms of a spectral decomposition:

Theorem 5 (Spectral Decomposition of STDF)**.** *$\ell : \Delta_{d-1} \to [1/d, 1]$ is a stable tail dependence function iff ℓ has the spectral decomposition with a spectral variable $\mathbf{W} \in \mathbb{R}_+^d$ such that*

$$\ell(\mathbf{x}) = \int_{\mathbb{R}_+^d} \max_{j \in \{1,\cdots,d\}} \{s_j x_j\} \mathrm{dF}_{\mathbf{W}}(\mathrm{s}) = \mathrm{E}_{\mathbf{W}} \left[\max_{\mathrm{j} \in \{1,\cdots,\mathrm{d}\}} \{\mathrm{W_j x_j}\} \right], \tag{2.29}$$

with moment constraints $\mathrm{E}_{\mathbf{W}}[\mathrm{W_j}] = 1$ for $j \in \{1, \cdots, d\}$ [27, 62, 108].[4]

Notably, the definition completely relies on an expectation with respect to a spectral variable $\mathbf{W} \in \mathbb{R}_+^d$ that is positive and a maxima across coordinates. This is related to the d-max-decreasing property of ℓ, a stronger form of convexity needed to ensure subsets of coordinates give valid stdfs, where a function $\ell(\mathbf{w}) : \mathbb{R}^d \to \mathbb{R}$ is d-

[4] The moment constraints are scale conveniences and not necessities, they give uniform margins for the extreme-value copula and unit Fréchet margins for the related max-stable process [38].

max-decreasing [57] if and only if for any $\mathbf{x} \leqslant \mathbf{y} \leqslant 0$ and any subset $E \subsetneq \{1, \ldots, d\}$[5]:

$$\sum_{\substack{\mathbf{m}=\{0,1\}^d, \\ m_j=1 \text{ if } j\in E}} ((-1)^{d+1-\sum_{j\leqslant d} m_j} \left(\sum_{j\leqslant d} -y_j^{m_j} x_j^{1-m_j} \right) \ell \left(\frac{y_1^{m_1} x_1^{1-m_1}}{\sum_{j\leqslant d} y_j^{m_j} x_j^{1-m_j}}, \cdots, \frac{y_d^{m_d} x_d^{1-m_d}}{\sum_{j\leqslant d} y_j^{m_j} x_j^{1-m_j}} \right)) \geqslant 0. \tag{2.30}$$

and $\ell(\mathbf{e}_j) = 1$ where $\mathbf{e}_j$ is the jth canonical basis function. Moreover, the following three characterizations are equivalent [57, Theorem 3.1.1]: (1) $F(\mathbf{x}) = \exp(-\ell(1/\mathbf{x}))$ defines a multivariate extreme-value distribution; (2) there exists a spectral measure $F_{\mathbf{W}}$ such that ℓ can be given as the expectation in (5.4); (3) $\ell(\mathbf{w})$ is d-max-decreasing. The definition is also related to the expression of ℓ as a d-norm, where a function $\ell(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ is a d-norm generated by $\mathbf{W}$ if it can be written as the expectation in (5.4) where the $\max(\cdot, \cdots, \cdot)$ operator is equivalently written as the $\|\cdot\|_\infty$ norm.[6]

Moreover, compositions of stdfs and hierarchies of spectral measures define valid stdfs [56, Section 2.2], i.e.

$$\ell(\mathbf{x}) = \mathrm{E}\left[\mathrm{s}_1^{(2,1)} \mathrm{E}\left[\max_{\mathrm{j}=1,\cdots,\mathrm{d}} \mathrm{s}_\mathrm{j}^{(1,1)} \mathrm{x}_\mathrm{j}\right], \cdots, \mathrm{s}_\mathrm{d}^{(2,1)} \mathrm{E}\left[\max_{\mathrm{j}=1,\cdots,\mathrm{d}} \mathrm{s}_\mathrm{j}^{(1,\mathrm{d})} \mathrm{x}_\mathrm{j}\right]\right]. \tag{2.31}$$

From the expectation in (5.4), we also see that ℓ is homogeneous of order 1, i.e. $\ell(cx_1, \cdots, cx_d) = c\,\ell(x_1, \cdots, x_d)$ for $c > 0$. In addition, $\max_{j\in\{1,\cdots,d\}} x_j \leqslant \ell(\mathbf{x}) \leqslant 1$ for $\mathbf{x} \in \Delta_{d-1}$ and, as mentioned, $\ell(\mathbf{e}_j) = 1$ where $\mathbf{e}_j$ is the jth canonical basis vector.

2.3.2 Pickands Estimator

The definition of the extreme-value copula hints at a method to transform the distribution of observations into an exponential with rate $\ell(\mathbf{x})$ given by the stdf ℓ evaluated

[5] For more background on the d-max-decreasing property, refer to Hofmann [57, Theorem 5.2.2] and Ressel [108, Theorem 6].

[6] For more background on d-norms, refer to Falk et al. [35].

at a test point $\mathbf{x}$. An estimate for $\ell(\mathbf{x})$ may then be computed as the reciprocal of the mean of the transformed observations, also known as the Pickands estimator [105]. We give more information on the Pickands estimator here and use the transformation from the Pickands estimator in our training of neural networks.

The Pickands transform of an observation $\mathbf{u}$ for a particular test point $\mathbf{x} \in \Delta_{d-1}$ results in the transformed observation,

$$\xi(\mathbf{u}, \mathbf{x}) = \min\{-\log(u_1)/x_1, \cdots, -\log(u_d)/x_d\}, \tag{2.32}$$

with survival distribution

$$\begin{aligned}
&P(\xi(\mathbf{U}, \mathbf{x}) > x) \\
&\quad = P(\min\{-\log(U_1)/x_1, \cdots, -\log(U_d)/x_d\} > x), \\
&\quad = P(-\log(U_1) > xx_1, \cdots, -\log(U_d) > xx_d), \\
&\quad = P(U_1 < \exp(-xx_1), \cdots, U_d < \exp(-xx_1)), && \exp(-\cdot) \text{ is decreasing}, \\
&\quad = \exp(-\ell(xx_1, \cdots, xx_d)), && C(\mathbf{u}) := \exp(-\ell(-\log(\mathbf{u}))), \\
&\quad = \exp(-x\ell(x_1, \cdots, x_d)), && \ell(c\mathbf{x}) = c\ell(\mathbf{x}), c > 0, \\
&\quad = \exp(-x\ell(\mathbf{x})),
\end{aligned}$$

such that $\xi(\mathbf{U}, \mathbf{x})$ is an exponential with rate $\ell(\mathbf{x})$.

In the case of the Pickands estimator, the stdf at a particular test point $\ell(\mathbf{x})$ is then estimated from the mean of the transformed observations, with endpoint corrections, as [105]:

$$\hat{\ell}(x) = \left(\sum_{i=1}^{n} -\log\left(\frac{i}{n+1}\right)\right) / \left(\sum_{i=1}^{n} \xi(\mathbf{u}_i, \mathbf{x})\right). \tag{2.33}$$

A modification to the Pickands estimator, using the transformation $\log(\xi(\mathbf{U}, \mathbf{x}))$, leads to the more robust CFG estimator [12]:

$$\log(\hat{\ell}(\mathbf{x})) = \frac{1}{n}\sum_{i=1}^{n} \log\left(-\log\left(\frac{i}{n+1}\right)\right) - \frac{1}{n}\sum_{i=1}^{n} \log(\xi(\mathbf{u}_i, \mathbf{x})). \tag{2.34}$$

Given as a mean of transformed observations, there are established results on the convergence of the Pickands and CFG estimators [19].

2.3.3 Max-Stable Processes

Related to the spectral decomposition of the stdf is the spectral representation of max-stable processes.

Let $W_1(\mathbf{s}), W_2(\mathbf{s}), \cdots$ be i.i.d. observations of random process $W(\mathbf{s})$, $\mathbf{s} \in \mathcal{S} \subset \mathbb{R}^q$ such that $W(\mathbf{s}) \geqslant 0$ and $\mathrm{E_W[W(s)]} = 1$ for $\mathbf{s} \in \mathcal{S}$. Furthermore, let $P_1, P_2, \cdots$ be i.i.d. observations of a Poisson point process on $[0, \infty)$ with intensity $x^{-2}\mathrm{dx}$. Then

$$Z(\mathbf{s}) = \sup_{i \geqslant 0}(P_i W_i(\mathbf{s})) \tag{2.35}$$

is a max-stable process with unit Fréchet margins.

For a subset of points $\mathbf{s}_1, \cdots, \mathbf{s}_d$, the random vector $\mathbf{X} = (X_1, \cdots, X_d) \stackrel{d}{=} (Z(\mathbf{s}_1), \cdots, Z(\mathbf{s}_d))$ has CDF $\exp(-\ell(1/x_1, \cdots, 1/x_d))$ with unit Fréchet margins $\exp(-1/x_j)$ and dependence described by an extreme-value copula with stdf given by

$$\ell(x_1, \cdots, x_d) = \mathrm{E_W}[\max_{\mathrm{j}=1,\cdots,\mathrm{d}} \mathrm{x_j W(s_j)}]. \tag{2.36}$$

Based on the spectral representation of max-stable processes [27, 113], approximate and exact simulation schemes have been proposed [29, 113]. We briefly describe approximate sampling by truncating the supremum over $i = 1, 2, \cdots$ to a finite number and choosing a finite spatial domain, replacing $W(\mathbf{s}_1, \cdots, \mathbf{s}_d)$ with $\mathbf{W} = (W_1, \cdots, W_d)$ [56, 113]. First, sample N Poisson points $p_1, \cdots, p_N$ in decreasing order as $p_i = 1/\sum_{k=1}^{i} e_k$ where e_k are i.i.d. unit exponentials. Then sample N i.i.d. spectral points $\mathbf{w}_1, \cdots, \mathbf{w}_N$. Finally, set $\mathbf{X} = \max_{1 \leqslant i \leqslant N}(p_i \mathbf{w}_i)$.

2.4 Archimax Copulas

Archimax copulas effectively balance representation of data within the bulk of the distribution and extrapolation to the tails by combining the tractability of Archimedean copulas with the tail properties of extreme-value copulas. They are defined as:

Definition 6 (Archimax copula)**.**

$$C(\mathbf{u}) = \varphi(\ell(\varphi^{-1}(u_1), \cdots, \varphi^{-1}(u_d))), \tag{2.37}$$

where $\varphi : [0, \infty) \to [0, 1]$ *is an Archimedean generator and* $\ell : [0, \infty)^d \to [0, \infty)$ *is a stable tail dependence function (stdf) [12, 18, 97].*

Archimax copulas generalize Archimedean and extreme-value copulas. When $\ell(\mathbf{x}) = \|\mathbf{x}\|_1 = (x_1 + \cdots + x_d)$, Archimax copulas reduce to Archimedean copulas. When $\varphi(x) = \exp(-x)$, Archimax copulas reduce to extreme-value copulas.

Intuitively, the Archimedean generator dictates the shape of the radial envelope applied across all covariates, while the stdf dictates the asymptotic dependence between covariates, with examples given in Figure 2.4.

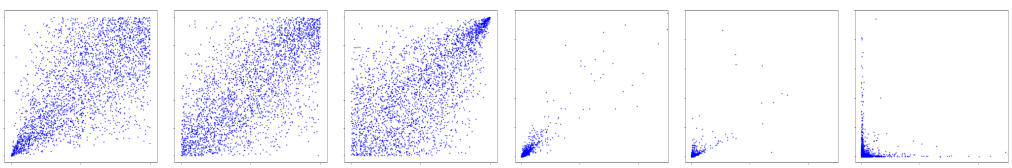

FIGURE 2.4: Examples of different radial envelopes and asymptotic dependencies

Archimax copulas were initially developed as a tool to study the behaviour of methods used to estimate the joint distribution of extreme events [12]. They allow extrapolation of dependence in the bulk to the extremes, where the limit of shifted and scaled component-wise maximas have extreme-value dependence with the following stdf:

$$\ell_*(x_1, \cdots, x_d) = \ell^m(x_1^{1/m}, \cdots, x_d^{1/m}), \tag{2.38}$$

for $\varphi^{-1}(1-1/t)$ regularly varying with degree $-1/m$ or equivalently $1-\varphi(1/t)$ regularly varying with degree $-m$, i.e. $\lim_{t\to\infty}\frac{1-\varphi(1/(tx))}{1-\varphi(1/t)} = x^m$, for some $m \geqslant 1$ [13, Proposition 4.1 $d=2$], [18, Proposition 6.1 $d \geqslant 2$].

In modeling applications, Archimax copulas allow insights into the distribution of extreme events (e.g. very strong and rare earthquakes) from a mix of moderately less extreme data (e.g. strong earthquakes) and extreme data. This in turn can be used to generate samples for further studies and simulations.

Archimax copulas were applied to areas such as nutrient intake [95], river flow rates [4] and rainfall [19], where the dependence is asymmetric and sub-asymptotic. In these applications, the authors noted a better fit when using Archimax copulas over Archimedean and extreme-value copulas.

For more background on Archimax copulas, we refer to [13, 18, 56].

2.4.1 Stochastic Representations of Archimax Copulas

The stochastic representations of Archimax copulas due to Charpentier et al. [18] are reminiscent of the Archimedean copulas. For the class of completely monotone φ, where φ is a Laplace transform, the Archimax copula is the survival copula of

$$\mathbf{X} = (X_1, \cdots, X_d) = (-\log(U_1)/M, \cdots, -\log(U_d)/M), \tag{2.39}$$

where $\mathbf{U} = (U_1, \cdots, U_d)$ has CDF given by an extreme-value copula with stdf ℓ, M is a latent variable with Laplace transform φ, $\mathbf{U}$ and M are independent, and the marginal SDFs are also given by φ [18, Proposition 3.1]. In this case, the Archimax copula may be interpreted as scale mixture of extremes or dependent frailties.

For the more general class of d-monotone φ, where φ is a Williamson d-transform, the Archimax copula is the survival copula of

$$\mathbf{X} = (X_1, \cdots, X_d) = (RS_1, \cdots, RS_d), \tag{2.40}$$

where the simplex component $\mathbf{S} = (S_1, \cdots, S_d)$ has SDF given by

$$P(\mathbf{S} > \mathbf{s}) = \max(0, 1 - \ell(s_1, \cdots, s_d))^{d-1}, \tag{2.41}$$

the radial component R has Williamson d-transform φ, R and $\mathbf{S}$ are independent, and the marginal SDFs are also given by φ [18, Theorem 3.3]. We see the correspondence between C, φ, ℓ, R and $\mathbf{S}$ by writing the SDF and marginal SDFs of $\mathbf{X}$ and applying Sklar's theorem (2.3).

$$\begin{aligned}
P(X_1 > x_1, \cdots, X_d > x_d) &= P(RS_1 > x_1, \cdots, RS_d > x_d) \\
&= \int_0^\infty P(S_1 > x_1/R, \cdots, S_d > x_d/R | R = r) P(R = r) dr \\
&= \int_0^\infty (1 - \ell(x_1/r, \cdots, x_d/r))_+^{d-1} p(r) dr \\
&= \int_0^\infty (1 - \ell(x_1, \cdots, x_d)/r)_+^{d-1} p(r) dr \\
&= \mathrm{E_R}[(1 - \ell(\mathbf{x})/\mathrm{R})_+^{\mathrm{d}-1}] \\
&= \varphi(\ell(\mathbf{x})). \\
P(X_j > x_j) &= \varphi(x_j).
\end{aligned}$$

In this case, the Archimax copula may be interpreted as resource sharing where R is a resource to be distributed randomly among d agents in a way specified by $\mathbf{S}$, where both R and $\mathbf{S}$ are themselves results of independent random processes. For example, R may be profits, and $\mathbf{S}$ may be the way profits is to be divided between stakeholders.

2.4.2 Modified Pickands Estimator

Given the Archimedean generator, we may modify the Pickands estimator from extreme-value copulas to estimate the stdf of the Archimax copula. The modified Pickands transformation of an observation $\mathbf{u}$ for a particular test point $\mathbf{x} \in \Delta_{d-1}$

results in the transformed observation,

$$\xi(\mathbf{u}, \mathbf{x}) = \min\{\varphi^{-1}(u_1)/x_1, \cdots, \varphi^{-1}(u_d)/x_d\}, \tag{2.42}$$

with survival distribution

$$\begin{aligned}
&P(\xi(\mathbf{U}, \mathbf{x}) > x) \\
&\quad = P(\min\{\varphi^{-1}(U_1)/x_1, \cdots, \varphi^{-1}(U_d)/x_d\} > x), \\
&\quad = P(\varphi^{-1}(U_1) > xx_1, \cdots, \varphi^{-1}(U_d) > xx_d), \\
&\quad = P(U_1 < \varphi(xx_1), \cdots, U_d < \varphi(xx_1)), && \varphi \text{ is decreasing}, \\
&\quad = \varphi(\ell(xx_1, \cdots, xx_d)), && C(\mathbf{u}) := \varphi(\ell(\varphi^{-1}(\mathbf{u}))), \\
&\quad = \varphi(x\ell(x_1, \cdots, x_d)), && \ell(c\mathbf{x}) = c\ell(\mathbf{x}), c > 0, \\
&\quad = \varphi(x\ell(\mathbf{x})).
\end{aligned}$$

In the case of extreme-value copulas, $\varphi(x) := \exp(-x)$, $\varphi^{-1}(u) := -\log(u)$ and $C(\mathbf{u}) := \exp(-\ell(-\log(u_1), \cdots, -\log(u_d)))$, such that the d-dimensionsal observations distributed according to the extreme-value copula are transformed into 1-dimensional observations distributed according to an exponential with rate $\ell(\mathbf{x})$. The stdf at a particular test point $\ell(\mathbf{x})$ is then estimated from the mean of the transformed observations [105]. A modification to the CFG estimator was made by Chatelain et al. [19] for the case of Archimax copulas:

$$\log(\hat{l}(\mathbf{x})) = \frac{1}{n}\sum_{i=1}^{n} \log\left(\varphi^{-1}\left(\frac{i}{n+1}\right)\right) - \frac{1}{n}\sum_{i=1}^{n} \log(\xi(\mathbf{u}_i, \mathbf{x})). \tag{2.43}$$

2.4.3 Kendall Distribution Function

Estimating an arbitrary Archimedean generator for the Archimax copula is an unsolved challenge. In our work, we build on the decomposition of the Kendall distribution function used in the estimation of Archimedean copulas [40, 42]. The Kendall distribution is formally defined as:

Definition 7 (Kendall distribution)**.** *Let C be a copula, and let the CDF of the random variable $\mathbf{U} \in [0,1]^d$ be the copula C. Define the random variable $W := C(\mathbf{U})$. The Kendall distribution of C is the multivariate probability integral transform given by the CDF of W:*

$$K(w) = P(C(\mathbf{U}) \leqslant w), \quad w \in [0,1]. \tag{2.44}$$

In addition, the empirical Kendall distribution K_N for i.i.d. observations $(u_{11}, \cdots, u_{1d})$, $\cdots, (u_{N1}, \cdots, u_{Nd})$ is defined as:

Definition 8 (Empirical Kendall distribution)**.**

$$K_N(w) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{w_i \leqslant w\}, \tag{2.45}$$

where

$$w_i = \frac{1}{N+1} \sum_{k=1}^{N} \mathbb{1}\{u_{k1} < u_{i1}, \cdots, u_{kd} < u_{id}\}. \tag{2.46}$$

In the case of Archimedean copulas, the Archimedean generator can be obtained as a function of the Kendall distribution [40], the support for the radial component can be solved from the support of the transformed random variables [42], and the convergence of Archimedean copulas is related to the convergence of the empirical Kendall process [42]. We detail the approach for Archimax copulas in Chapter 5.

2.4.4 Generalized Pareto Copulas

Sampling the simplex component of the Archimax copula is another unsolved challenge. In our work, we make use of the correspondence between the simplex component $\mathbf{S}$ and the spectral component $\mathbf{W}$ through generalized Pareto copulas defined in terms of the stdf ℓ as:

Definition 9 (generalized Pareto copula)**.** *A generalized Pareto copula is defined as the copula of a generalized Pareto distribution with support on* $(-\infty, 0]^d$ *and CDF specified by* ℓ *as:*

$$P(X_1 < x_1, \cdots, X_d < x_d) = \max(0, 1 - \ell(-x_1, \cdots, -x_d)). \tag{2.47}$$

Moreover, it has stochastic representation as:

$$\mathbf{X} \stackrel{d}{=} -\left(\frac{U}{W_1}, \cdots, \frac{U}{W_d}\right), \tag{2.48}$$

such that

$$\begin{aligned}
&P(X_1 < x_1, \cdots, X_d < x_d), \\
&\quad = P(-U/W_1 < x_1, \cdots, -U/W_1 < x_d), \\
&\quad = \int_{\Delta_{d-1}} P(U > -W_1 x_1, \cdots, U > -W_d x_d | \mathbf{W} = \mathbf{w}) P(\mathbf{W} = \mathbf{w}) d\mathbf{w}, \\
&\quad = \int_{\Delta_{d-1}} P(U > \max_j -W_j x_j) P(\mathbf{W} = \mathbf{w}) d\mathbf{w}, \\
&\quad = \max(0, \mathrm{E}_{\mathbf{W}}[1 - \max_{\mathrm{j}}(-\mathrm{W_j x_j})]), \\
&\quad = \max(0, 1 - \ell(-x_1, \cdots, -x_d)),
\end{aligned}$$

where U *is uniformly distributed on* $[0, 1]$*, independent of spectral component* $\mathbf{W} = (W_1, \cdots, W_d)$ *from the spectral decomposition of* ℓ *in* (5.4) *[11, 35].*

Thus, given samples $(x_{11}, \cdots, x_{1d}), \cdots, (x_{(d-1)1}, \cdots, x_{(d-1)d})$ from a generalized Pareto distribution with CDF in (5.12), we may obtain a sample of $\mathbf{s}$, where coordinates s_j for $j = \{1, \cdots, d\}$ are computed as [18]:

$$s_j = -\max(x_{1j}, \cdots, x_{(d-1)j}). \tag{2.49}$$

The coordinate-wise maxima is taken over $d-1$ samples to get the $d-1$ exponent in the SDF of $\mathbf{S}$, corresponding to the $d-1$ exponent in the expression of φ.

The challenge of sampling the simplex component is thus transformed into the challenge of sampling the spectral component. We detail the approach in Chapter 5.

2.5 Performance Metrics

Model misspecification may lead to misspecification of risks, leading to potentially catastrophic outcomes in areas such as healthcare, safety and finance. Risks may be mitigated by confirming a reasonable fit between the model and the data. We now discuss some evaluation metrics to assess the performance of a model.

2.5.1 Log-Likelihood

In cases where the copula density exists, we may use the log-likelihood as a performance metric. Since copulas are given as CDFs however, obtaining the log-likelihood may be challenging due to the need to differentiate the copula d times.

2.5.2 Cramér-von Mises Statistic

Another metric is the Cramér-von Mises (CvM) statistic [23], commonly used for goodness-of-fit tests [107]. The CvM statistic measures a discrepancy between the model copula $C(\cdot, \cdots, \cdot; \theta)$ and the empirical copula $C_N(\cdot, \cdots, \cdot)$, where there are established results on the weak convergence of the empirical copula process [36] and we may reject models based on hypothesis testing and p-values [107].

The CvM statistic is defined as:

Definition 10 (Cramér-von Mises statistic)**.**

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (C(\mathbf{u}_i; \theta) - C_N(\mathbf{u}_i))^2, \tag{2.50}$$

where $\mathbf{u}_1, \cdots, \mathbf{u}_N$, with $\mathbf{u}_i = (u_{i1}, \cdots, u_{i1d})$, are i.i.d. observations of the margins. In addition, the empirical copula C_N is defined as:

Definition 11 (Empirical copula)**.**

$$C_N(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{u_{i1} \leqslant u_1, \cdots, u_{id} \leqslant u_d\}. \tag{2.51}$$

In the two-sample version of the CvM statistic, the model copula is given as the empirical copula of samples from the model copula [107].

2.5.3 Lambda Map for Archimedean Generators

The map $\lambda : (0, 1] \to (-\infty, 0]$ for a given Archimedean generator φ is defined as:

Definition 12 (Lambda map for Archimedean generator)**.**

$$\lambda(w) = \varphi^{-1}(w)/(\varphi^{-1}(w))' = \{\varphi' \circ \varphi^{-1}(w)\}\varphi^{-1}(w), \tag{2.52}$$

such that an estimate of φ can be recovered from λ as

$$\varphi^{-1}(w) = \exp\left\{\int_{w_0}^{w} 1/\lambda(t)dt\right\}. \tag{2.53}$$

In dimension $d = 2$, λ is directly related to the Kendall distribution function as $\lambda(w) = w - K(w)$, such that the empirical lambda map is computed as $\lambda_N(w) = w - K_N(w)$ [40], where there are established results on the convergence of the empirical Kendall process [5]. The asymptotic variance for λ_N may be computed from the asymptotic variance for K_N, where the asymptotic variance of the independence copula may be easily computed as:

$$\sigma^2_{\lambda_N}(x) = \frac{x(x - \log(x) - 1)}{N}. \tag{2.54}$$

Although the relationship between λ and K is more complicated in $d > 2$, and the copula is often not the independence copula, the above tools are often used for drawing approximate confidence bands around λ_N to reject models whose λ fall

outside the band [40, 42]. It is also convenient to present results in terms of λ since it is scale invariant, unlike φ, where $\varphi(cx)$ for any $c > 0$ lead to the same Archimedean copula. Results may also be given in terms of mean-squared error (MSE) to the empirical Kendall distribution.

2.5.4 Integrated Squared Error for STDFs

The integrated squared error (ISE) and integrated relative absolute error (IRAE) between the modeled stdf $\ell(\cdot, \cdots, \cdot; \theta)$ and the true stdf $\ell(\cdot, \cdots, \cdot)$ is defined as [19]:

Definition 13 (Integrated squared errror)**.**

$$\mathrm{ISE} = \frac{1}{|\Delta_{\mathrm{d}-1}|} \int_{\Delta_{\mathrm{d}-1}} (\ell(\mathbf{x}; \theta) - \ell(\mathbf{x}))^2 \, \mathrm{d}\mathbf{x}. \tag{2.55}$$

Definition 14 (Integrated relative absolute error)**.**

$$\mathrm{IRAE} = \frac{1}{|\Delta_{\mathrm{d}-1}|} \int_{\Delta_{\mathrm{d}-1}} \frac{|\ell(\mathbf{x}; \theta) - \ell(\mathbf{x})|}{\ell(\mathbf{x})} \, \mathrm{d}\mathbf{x}. \tag{2.56}$$

Both ISE and IRAE are often computed using Monte Carlo integration with samples $\mathbf{x}_i$ drawn uniformly at random from the simplex Δ_{d-1}.

We may also compare the model stdf to the empirical stdf obtained as the mean of transformed observations (2.33), (2.34) and (2.43), where there are established results on the convergence of the empirical stdf [19].

2.6 Additional Background

Having reviewed the necessary background material, we now provide additional background on copulas and copulas in machine learning.

2.6.1 Additional Background on Copulas

Copulas summarize the dependence described by an arbitrary joint CDF after the marginals have been normalized to be uniform. They provide easy marginaliza-

tion and calculation of tails. In addition, when used in a graphical model, some conditional independence that cannot be easily represented with Markov random fields or Bayesian networks, can be easily represented with cumulative distribution networks [60]. They are also particularly convenient in some applications, such as ranking, where the likelihood is a CDF [58].

We noted a list of resources earlier and repeat them here [39, 65, 66, 68, 98].

We also briefly discuss the common copulas given in Table 2.1 earlier. The Gaussian copula has a tractable expression for both the CDF and the density. However, it is independent in the tails, a significant reason why Gaussian copulas are not suitable for modeling financial risks. Vine copulas, such as R-vines, C-vines and D-vines, are computationally intensive and hard to interpret due to repeated conditioning with pair copulas. Archimedean copulas are symmetric in all coordinates, which is an assumption that is usually not held in practice. Hierarchical Archimedean copulas partially address this symmetry but are difficult to construct due to nesting conditions that are hard to satisfy. Extreme-value copulas are max-stable copulas which result in lower tail independence, an assumption that is sometimes not held in practice. On the other hand, many copulas are not flexible and independent in the extremes. In the case of Archimedean copulas, only Gumbel copulas satisfy the tail dependence. In addition, being Archimedean copulas, Gumbel copulas only describe symmetric dependence. In high dimensions, none of the existing copulas typically fit data well. Model misspecification is often accepted in return for tractability, and some dependence is better than independence [55].

Inferring the parameters of a copula is usually done via maximum likelihood estimation if the copula density exists. For many copulas, there may be singularities and thus the copula density may not exist at all points. In addition, since copulas are given as CDFs, computing the density is usually not tractable in high dimensions,

due to repeated differentiation,

$$c(\mathbf{u}) = \frac{\partial^{(d)}}{\partial u_1 u_2 \cdots u_d} C(u_1, u_2, \cdots, u_d), \tag{2.57}$$

where c is the density if it exists. In cases where a density cannot be computed, the copula is usually fitted using minimum distance estimators and goodness-of-fit tests [17]. In both cases, expectations are usually replaced by their empirical versions.

Sampling from a copula using the conditional sampling method with Rosenblatt transform is usually not tractable in high dimensions, again due to repeated differentiation to compute conditional CDFs and inverses $C^{-1}(u_1)$, $C^{-1}(u_2|u_1)$, $\cdots$, $C^{-1}(u_d|u_1, \cdots, u_{d-1})$, where

$$\begin{aligned} C(u_i|u_1, \cdots, u_{i-1}) = \frac{\partial^{(i-1)}}{\partial u_1 \cdots u_{i-1}} C(u_1, \cdots, u_{i-1}, u_i) / \\ \frac{\partial^{(i-1)}}{\partial u_1 \cdots u_{i-1}} C(u_1, u_2, \cdots, u_{i-1}, 1). \end{aligned} \tag{2.58}$$

In our experiments, the conditional sampling method, using automatic differentiation in `PyTorch` breaks down at dimension $d = 4$. In general, only models with stochastic representations may be easy to sample [89].

2.6.2 Additional Background on Copulas in Machine Learning

Copulas is a rising topic in machine learning, as evident from numerous publications, including but not limited to:

- Cumulative distribution networks, modeled as a product of copulas [59, 60, 61]. They can represent some conditional independencies not represented by Markov random fields and Bayesian networks, allow loops [69] and mixed graphs [114], with application to ranking [58] and heavy-tailed distributions [69].

- Copula Bayesian networks, modeled as a product of conditional copulas [32], with application to missing data [33], classification [34], time-series [31] and fast structure learning [118, 119].
- Copula processes [128] and application of copulas to time-series [52, 86, 109, 124].
- Copula-based dependence measures and distances [16, 77, 87, 93, 106].
- Copula variational inference, allowing dependencies between latent variables [49, 53, 120].
- Generative modeling [1, 9, 21, 63, 79, 117]. Recent work on deep network based copulas, including Archimedean [83, 99], extreme-value [50], Archimax [100], autoregressive [70, 102], optimal transport [14, 90] and transformer-attentional copulas [30].
- Applications of copula in areas including graph neural networks [88], multi-label learning [84], multi-agent interactions [123], bundle pricing [81], missing value [78, 122, 130], sparse representation [126], outlier detection [82], causal discovery [25], domain adaptation and transfer learning [85, 110] and structure learning [15].

3

Generative Archimedean Copulas

3.1 Introduction

Archimedean copulas

$$C(\mathbf{u}) = \varphi(\varphi^{-1}(u_1) + \cdots + \varphi^{-1}(u_d)) \tag{3.1}$$

are popular for their simplicity and ability to model different tail behaviours.[1] We consider their probabilistic construction as mixture models with the completely monotone Archimedean generators given by Laplace transforms of latent random variables.[2] We model the latent variables with generative neural networks and consider the empirical Laplace transforms using samples of the latent variables. This alternative representation allows for computational efficiencies in computing the likelihood and sampling, especially in high dimensions. The properties of the empirical Laplace transform is used to compute higher-order derivatives. Samples of the latent distribution is necessary in effective sampling of the learnt copula using Marshall-

[1] Part of the material presented in this chapter was co-authored with Ali Hasan, Khalil Elkhalil and Vahid Tarokh [99].

[2] We will subsequently consider the more general d-monotone Archimedean generators given by Williamson d-transforms in Chapter 5 on Archimax copulas.

Olkin type algorithms. We also describe an application of this method to scale up an existing neural parameterization of the Archimedean generator. Archimedean copulas can be extended to hierarchical (or nested) Archimedean copulas, where multiple Archimedean generators are used in conjunction to increase the expressiveness of the model. We generalize our method to nested Archimedean generators and subsequently recover a richer class of copulas.

3.2 Modeling the Archimedean Generator

From Section 2.2.1, the Archimedean generator φ is given as the Laplace transform of a latent variable $M > 0$:

$$\varphi(x) = \mathrm{E_M}[\mathrm{e}^{-\mathrm{Mx}}]. \tag{3.2}$$

We let M be the output of a generative network such that a sample m is computed as $m = g(\epsilon; \theta)$ where $g : \mathbb{R} \to \mathbb{R}^+$ is a generative network with parameters θ and ϵ is a sample from a source of randomness. Unlike the modeling of monotone functions with neural networks [21, 83], there are no restrictions on the weights and intermediate activations of $g(\cdot; \theta)$. We let $g(\cdot; \theta)$ be a multilayer perceptron. To ensure a positive output, we use $\exp(\cdot)$ as the output activation.

We then approximate the Laplace transform with its empirical version using samples $m_1, \cdots, m_L$ from $g(\cdot; \theta)$ as:

$$\varphi(x) = \mathrm{E_M}[\mathrm{e}^{-\mathrm{Mx}}] \approx \frac{1}{\mathrm{L}} \sum_{\mathrm{l}=1}^{\mathrm{L}} \mathrm{e}^{-\mathrm{m_l x}}. \tag{3.3}$$

To avoid scale ambiguities, we add a penalty term such that $\mathrm{E_M}[\mathrm{M}] = 1$.

We next present an existing neural parameterization [83] in the form of a Laplace transform in order to subsequently use the properties of the Laplace transform and the stochastic representation. In [83], the Archimedean generator is given as a com-

position of negative exponentials, specifically:

$$\varphi_{0,1}(t) = 1, \tag{3.4}$$

$$\varphi_{l,i}(t) = \exp(-B_{l,i}t) \sum_{j=1}^{H_{l-1}} A_{l,i,j}\varphi_{l-1,j}(t), \tag{3.5}$$

$$\varphi(t) = \varphi_{L+1,1}(t) = \sum_{j=1}^{H_L} A_{L+1,1,j}\varphi_{L,j}(t), \tag{3.6}$$

where the input and output layers have widths $H_0 = H_{L+1} = 1$; there are L hidden layers where the width of the lth layer is H_l. Each A_l is a positive matrix of size $H_l \times H_{l-1}$, where each row is normalized to be a H_{l-1}-dimensional probability vector, i.e. $\sum_{j=1}^{H_{l-1}} A_{l,i,j} = 1$. Each B_l is a positive vector of size H_l. To fulfill the constraints on A and B, the softmax$(\cdot)$ and $\exp(\cdot)$ weight activations are applied.

Following the derivation in [83][Appendix 8.3 and 8.4], each layer can be written as a convex combination of negative exponentials such that φ has the mixture representation:

$$\varphi(t) = \sum_{k=1}^{K} \alpha_k \exp(-\beta_k t), \tag{3.7}$$

where $K = \prod_{l=1}^{L} H_l$ denotes the number of mixture components and

$$\alpha_k = \prod_{l=1}^{L+1} A_{l,z_l^k,z_{l-1}^k}, \qquad \beta_k = \sum_{l=1}^{L} B_{l,z_l^k}, \tag{3.8}$$

where $(z_0^k, z_1^k, \cdots, z_{L+1}^k)$ for $k \in \{1, \cdots, K\}$ denotes a forward path through the layers of the network where the network is seen as a Markov reward process. In this case, φ is the Laplace transform of a discrete random variable M taking values at β_k with probability α_k for $k \in \{1, \cdots, K\}$. Analogous to our representation with generative networks, to avoid scale ambiguities, we add a penalty term such that $\mathrm{E_M[M]} = 1$. While a single layer is also a valid Archimedean generator, the network presents an

efficient way of encoding an exponentially larger support of M. Using the properties of the Laplace transform and the stochastic representation, it is possible to scale the method to high dimensions. In [83], the Laplace transform representation was given in the appendix but its properties were not used, thus the results were limited to dimensions lower than and equal to three.

It is helpful to have both generative and network representations for φ. While both representations are equivalent, each is more suited for a particular task. The generative representation is more suitable for sample generation while the deterministic representation is more suitable for computing statistical quantities.

3.3 Training the Archimedean Copula

3.3.1 Training with Maximum Likelihood

The density of the Archimedean copula is obtained by differentiating the copula d times:

$$c(\mathbf{u}) = \frac{\partial^d C(u_1, \cdots, u_d)}{\partial u_1 \cdots \partial u_d} = \frac{\varphi^{(d)}(\varphi^{-1}(u_1) + \cdots + \varphi^{-1}(u_d))}{\prod_{i=1}^{d} \varphi'(\varphi^{-1}(u_i))}. \tag{3.9}$$

Since φ is a Laplace transform, its derivatives can be computed, and in the generative modeling case, approximated with samples $m_1, \cdots, m_L$ from $g(\cdot;\theta)$ as:

$$\varphi'(x) = \mathrm{E_M}[-\mathrm{M}\mathrm{e}^{-\mathrm{M}\mathrm{x}}] \approx \frac{1}{\mathrm{L}} \sum_{\mathrm{l}=1}^{\mathrm{L}} -\mathrm{m_l}\mathrm{e}^{-\mathrm{m_l}\mathrm{x}}, \tag{3.10}$$

$$\varphi^{(d)}(x) = \mathrm{E_M}[(-\mathrm{M})^{\mathrm{d}}\mathrm{e}^{-\mathrm{M}\mathrm{x}}] \approx \frac{1}{\mathrm{L}} \sum_{\mathrm{l}=1}^{\mathrm{L}} (-\mathrm{m_l})^{\mathrm{d}}\mathrm{e}^{-\mathrm{m_l}\mathrm{x}}.$$

This avoid numerical issues with differentiating the copula d times, allowing the method to scale to high dimensions.

The inverse φ^{-1} is computed numerically as in [83], using algorithms such as Newton's method or the bisection method. Since φ is an expectation, we can make

use of Jensen's inequality to reduce the search space.

$$\log(\mathrm{E_M}[\mathrm{e}^{-\mathrm{Mx}}]) \geqslant \mathrm{E_M}[\log(\mathrm{e}^{-\mathrm{Mx}})] = \mathrm{E_M}[-\mathrm{Mx}] = -\mathrm{E_M}[\mathrm{M}]\,\mathrm{x} = -\mathrm{x} \tag{3.11}$$

where we let $\mathrm{E_M}[\mathrm{M}] = 1$ and thus $x \geqslant -\log(y)$ where $x = \varphi^{-1}(y)$.

We then train the model using gradient descent on the negative log-likelihood with respect to the model parameters using automatic differentiation tools such as `PyTorch`. The derivatives of the inverse are computed from the derivatives of φ using the implicit function theorem then supplemented to backpropagation, as in [83]:

$$\frac{\partial \varphi^{-1}(u;\theta)}{\partial u} = 1 / \frac{\partial \varphi(x;\theta)}{\partial x}, \tag{3.12}$$

$$\frac{\partial \varphi^{-1}(u;\theta)}{\partial \theta} = -\frac{\partial \varphi(x;\theta)}{\partial \theta} / \frac{\partial \varphi(x;\theta)}{\partial x}. \tag{3.13}$$

3.3.2 Training with Goodness-of-Fit

The model may also be fitted to data via minimum distance criterions used in goodness-of-fit tests [41]. Though not statistically efficient compared to maximum likelihood estimation, minimum distance estimation is significantly less computationally intensive. We consider the Cramér-von Mises (CvM) statistic [23] to measure a discrepancy between the model copula $C(\cdot, \cdots, \cdot; \theta)$ and the empirical copula $C_N(\cdot, \cdots, \cdot)$ as described in Section 2.5.2.

3.3.3 Training with Minimum Discrepancy between Samples

An advantage of the generative representation is sample generation, which allows training the model by minimizing a discrepancy between true samples from data and fake samples from the model, as in generative adversarial networks (GANs) [44]. A variety of discrepancies may be used, including Jensen-Shannon divergence [44], Wasserstein distance [3], Maximum Mean Discrepancy [45], and two-sample CvM test [23]. We use the discriminator from vanilla GAN. In contrast to GANs, the distribution of samples from our model satisfies an Archimedean copula.

3.4 Sampling the Archimedean Copula

From Section 2.2.3, the Archimedean copula is the survival copula of

$$\mathbf{X} = (X_1, \cdots, X_d) = (E_1/M, \cdots, E_d/M), \tag{3.14}$$

where E_j are i.i.d. unit exponentials, M is the latent variable with Laplace transform φ and the marginal SDFs are also given by φ. We modify existing Marshall-Olkin type sampling algorithms [92, 94] to our parameterization with generative networks. We first sample d i.i.d. unit exponentials $e_1, \cdots, e_d$ and $L+1$ i.i.d. latents $m_1, \cdots, m_L, m$ where $m = g(\epsilon; \theta)$. We then compute $\mathbf{u} = (u_1, \cdots, u_d)$ as $u_j = \varphi(e_j/m)$ for $j \in \{1, \cdots, d\}$ where φ is approximated with samples $m_1, \cdots, m_L$ as in (3.3). We summarize the sampling method in Figure 3.1 and Algorithm 1.

Unlike the conditional sampling method with inverse Rosenblatt transforms, this sampling method does not require repeated differentiation of the copula to get conditional CDFs and does not require inversion of the conditional CDFs.

For the existing neural parameterization [83], M may been sampled as a Markov reward process [83][Appendix 8.4], then used in the above Marshall-Olkin type sampling algorithm. In [83], the stochastic representation was given in the appendix but its properties were not used, thus the results were limited to dimensions lower than and equal to three.

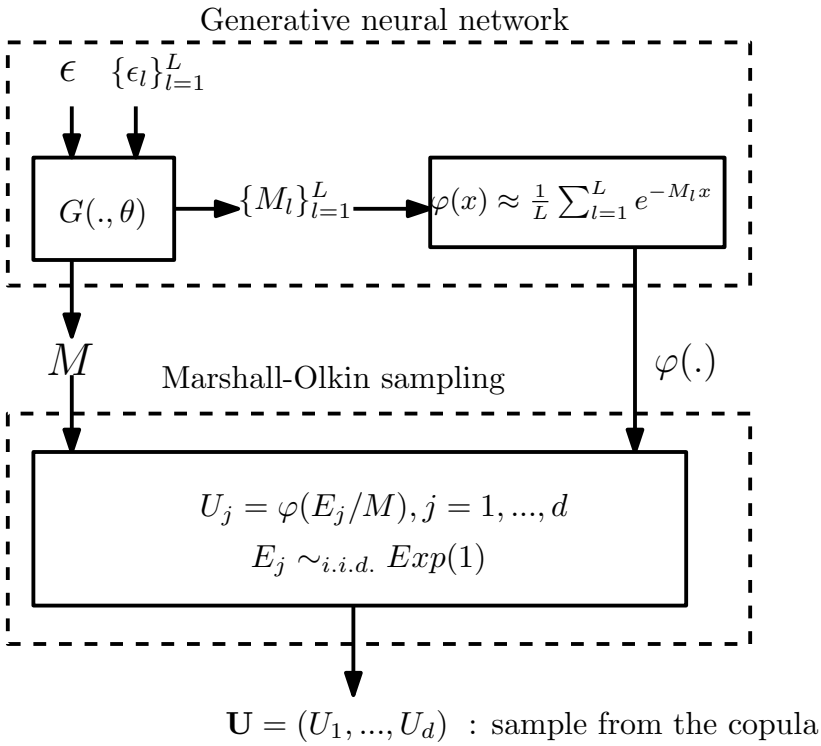

FIGURE 3.1: Sampling from generative Archimedean copulas

Algorithm 1 Sampling generative Archimedean copulas

Input: $g(\cdot;\theta)$,

1: Sample i.i.d. $e_1, \cdots, e_d$ unit exponentials.
2: Sample i.i.d. $m_1, \cdots, m_L, m$ latents as $m = g(\epsilon;\theta)$ where ϵ is a noise source.
3: Approximate φ with samples $\{m_l\}_{l=1}^L$ as in (3.3).
4: Compute $\mathbf{u} = (u_1, \cdots, u_d)$ where $u_j = \varphi(e_j/m)$ for $j \in \{1, \cdots, d\}$.

Output: $\mathbf{u}$.

3.5 Modeling the Nested Archimedean Generator

From Section 2.2.2, the hierarchical Archimedean copula is defined as:

$$C(\mathbf{u}) = C_{\varphi_0}(C_{\varphi_1}(u_{1,1}, \cdots, u_{1,d_1}), \cdots, C_{\varphi_J}(u_{J,1}, \cdots, u_{J,d_J})), \tag{3.15}$$

where $C_{\varphi_0}, C_{\varphi_1}, \cdots, C_{\varphi_J}$ are (possibly nested) Archimedean copulas with generators $\varphi_0, \varphi_1, \cdots, \varphi_J$ and an inner generator φ_j as the composition of an outer generator φ_0 and the Laplace exponent ψ_j of a compound Poisson process Λ_j such that:

$$\psi_j(x) = \varphi_0^{-1} \circ \varphi_j(x) = \mu_j x + \beta_j(1 - \mathrm{E}_{\mathrm{M_j}}[\mathrm{e}^{-\mathrm{M_j x}}]), \tag{3.16}$$

$$\varphi_j(x) = \varphi_0 \circ \psi_j(x) = \varphi_0(\mu_j x + \beta_j(1 - \mathrm{E}_{\mathrm{M_j}}[\mathrm{e}^{-\mathrm{M_j x}}])). \tag{3.17}$$

In an analogous manner, we approximate the Laplace transform φ_{M_j} in the Laplace exponent ψ_j with its empirical version using samples $m_{j1}, \cdots, m_{jL}$ from $g_j(\cdot; \theta_j)$ as:

$$\varphi_{M_j}(x) = \mathrm{E}_{\mathrm{M_j}}[\mathrm{e}^{-\mathrm{M_j x}}] \approx \frac{1}{\mathrm{L}} \sum_{\mathrm{l}=1}^{\mathrm{L}} \mathrm{e}^{-\mathrm{m_{jl} x}}. \tag{3.18}$$

where $g_j : \mathbb{R} \to \mathbb{R}^+$ is a generative network with parameters θ_j, modeled as a multilayer perceptron with $\exp(\cdot)$ output activation. We let the drift $\mu_j > 0$ and the jump intensity $\beta_j > 0$ be trainable parameters with $\exp(.)$ weight activation. In addition, to avoid scale ambiguities, we add a penalty term such that $\mathrm{E}_{\mathrm{M_j}}[\mathrm{M_j}] = 1$.

We note that $\varphi_0, \varphi_{M_1}, \cdots, \varphi_{M_J}$ may have arbitrary parameterization, such as the single-parameter families, the network and generative representations.

3.5.1 Training the Hierarchical Archimedean Copula

There are multiple competing methods for estimating hierarchical Archimedean copulas [48, 101, 112]. We present two methods in this section.

One method is to train the entire copula jointly with maximum likelihood. The density of the hierarchical Archimedean copula may be obtained by differentiating

the copula d times. To avoid numerical difficulties, replace instances of $\varphi_0^{-1} \circ \varphi_j$ with ψ_j, compute derivatives of φ_0, φ_{M_j} using the properties of the Laplace transform, compute the inverse of φ_j numerically, the derivatives of the inverse φ_j via the implicit function theorem then supplement them to backpropagation.

Another method is to train the copula sequentially from outer to inner generators, since an inner generator is a composition of an outer generator and a Laplace exponent. The outer generator may be trained using a composite likelihood of bivariate margins that correspond to different inner generators, as they are Archimedean with generator given by the outer generator. The outer generator may also be trained using goodness-of-fit with CvM and empirical copulas on C_{φ_j}. After the outer generator is fixed, the inner generators may be separately trained using methods for training Archimedean copulas, such as maximum likelihood estimation with copula densities c_{φ_j} as in (3.9).

3.5.2 Sampling the Hierarchical Archimedean Copula

From Section 2.2.2, the hierarchical Archimedean copula is the survival copula of

$$\mathbf{X} = (\frac{E_{1,1}}{\Lambda_1(M)}, \cdots, \frac{E_{1,d_1}}{\Lambda_1(M)}, \cdots, \cdots, \cdots, \frac{E_{J,1}}{\Lambda_J(M)}, \cdots, \frac{E_{J,d_J}}{\Lambda_J(M)}), \tag{3.19}$$

where $E_{j,i}$ are i.i.d. unit exponentials, Λ_j are Lévy subordinators with Laplace exponents ψ_j that have jump size distributions described by Laplace transforms φ_{M_j}. The Lévy subordinators Λ_j are evaluated at a common 'time' $t = M$, where M has a distribution described by the Laplace transform φ_0.

We modify the Marshall-Olkin type algorithm given in [51] to work with our parameterization using generative neural networks. We first sample i.i.d. exponentials $e_{1,1}, \cdots, e_{J,d_J}$, the latent m, then the compound Poisson processes $\lambda_1(m), \cdots, \lambda_J(m)$. We then sample i.i.d. copies of M, $M_1, \cdots, M_J$ to approximate $\varphi_1, \cdots, \varphi_J$. We compute $\mathbf{u} = (u_{1,1}, \cdots, u_{J,d_J})$ as $u_{j,k} = \varphi_j(e_{j,k}/m)$ for $j \in \{1, \cdots, J\}$ and $k \in \{1, \cdots, d_j)$.

We summarize the sampling algorithms for the compound Poisson processes and the hierarchical Archimedean copula here:

Algorithm 2 Sampling compound Poisson processes

Input: $t, \mu_j, \beta_j, g_j(.;\theta_j)$,

1: Sample $n_j(t)$ where $N_j(t) \sim \text{Pois}(\beta_j t)$,
   i.e. the number of jumps by time t of a Poisson random variable with rate β_j.
2: Sample $n_j(t)$ samples of M_j from $g_j(.;\theta_j)$.
3: Compute $\lambda_j(t) = \mu_j t + \sum_{i=1}^{n_j(t)} m_{j,i}$.

Output: $\lambda_j(t)$.

Algorithm 3 Sampling generative hierarchical Archimedean copulas

Input: $g_0(.;\theta_0), \{\mu_j, \beta_j, g_j(.;\theta_j)\}_{j=1}^{J}$,

1: Sample $t = m$ from $g_0(.;\theta_0)$.
2: Approximate φ_0 with samples from $g_0(.;\theta_0)$, as in (3.3).
3: **for** $j \in \{1, \cdots, J\}$ **do**
4: Sample $\lambda_j(m)$,
   i.e. the compound Poisson process at $t = m$, following Algorithm 2.
5: Approximate ψ_j with samples from $g_j(.;\theta_j)$, as in (3.3).
6: **end for**
7: Sample i.i.d. unit exponentials $e_{j,i}$ for $j \in \{1, \cdots, J\}$ and $i \in \{1, \cdots, d_j\}$.
8: Compute $\mathbf{u} = (u_{1,1}, \cdots, u_{J,d_J})$ as $u_{j,i} = (\varphi_0 \circ \psi_j)(e_{j,i}/\lambda_j(m))$ for $j \in \{1, \cdots, J\}$ and $i \in \{1, \cdots, d_j\}$.

Output: $\mathbf{u}$.

3.6 Experiments

3.6.1 Learning Different Tail Dependencies

Following the experiment setup in [83], we consider the Clayton, Frank and Joe copulas, introduced in Section 2.2.1, for their different tail behaviours, with parameters chosen to be 5, 15, and 3 respectively. Each dataset had 2000 train and 1000 test points. Following [83], we also consider the real-world data sets: Boston housing, Intel-Microsoft (INTC-MSFT) stocks and Google-Facebook (GOOG-FB) stocks. Each dataset was divided into train and test points in a 3:1 ratio, then rank-normalized to get approximately uniform margins. We then applied the three train-

ing methods discussed earlier: maximum likelihood, goodness-of-fit and adversarial training. Experiments were conducted using `PyTorch`, on a 2.7 GHz Intel Core i7 with 16 GB of RAM, with code available at `https://github.com/yutingng/gen-AC`.

The generative network was a multilayer perceptron of comparable size to [83], 2 hidden layers, each of width 10. We used $\text{Unif}(0, 1)$ as the input source of randomness, default weight initialization, LeakyReLU intermediate activations and $\exp(\cdot)$ output activation. To reduce computation complexity during training, we used a smaller number of samples $L = 100$ from the generative network to approximate the Laplace transforms. To increase inference accuracy for evaluation, we used a larger number of samples $L = 1000$ from the generative network to approximate the Laplace transforms. The tolerance for Newton's root-finding method was 1e-10.

For training with maximum likelihood, we used the same optimization parameters: stochastic gradient descent (SGD) with learning rate 1e-5 and momentum 0.9 on sum of log-likelihoods. For training with goodness-of-fit, we used SGD with learning rate 1e-3 and momentum 0.9. For adversarial training, we used Adam [72] with learning rate 1e-4, momentum 0.9 and betas (0.5, 0.999). The discriminative neural network had a single hidden layer of width 20, default weight initialization, LeakyReLU intermediate activations and $\text{sigmoid}(\cdot)$ output activation. All training methods used the same batch size of 200 and converged within 10k epochs. We reported the results at 10k epoch.

We evaluate the performance of the methods using log-likelihood and report the negative log-likelihoods (NLLs) from learning known Archimedean copulas in Table 3.1. We use the shorthands 'GT', 'ACNet', 'MLE', 'CvM', 'GAN' to denote ground truth, ACNet [83], and generative Archimedean copulas (Gen-AC) trained with maximum likelihood, goodness-of-fit and adversarial training. The NLLs from fitting real-world data are reported in Table 3.2, where the NLL of the best-fit single parameter copula (chosen from Clayton, Frank, Joe and Gumbel, as in [83]), with

the shorthand 'BF' is reported in place of the ground truth.

Table 3.1: Negative log-likelihoods of fitting known Archimedean copulas

	Benchmark		**Gen-AC**		
Known AC	GT	ACNet	MLE	CvM	GAN
Clayton	-0.94	-0.92	-0.89	-0.86	-0.89
Frank	-0.90	-0.88	-0.89	-0.86	-0.89
Joe	-0.51	-0.49	-0.48	-0.35	-0.47

Table 3.2: Negative log-likelihoods of fitting real-world data

	Benchmark		**Gen-AC**		
Dataset	BF	ACNet	MLE	CvM	GAN
Boston	-0.30	-0.27	-0.29	-0.30	-0.28
INTC-MSFT	-0.19	-0.20	-0.16	-0.15	-0.17
GOOG-FB	-0.93	-0.96	-0.95	-0.92	-0.94

Gen-AC achieved comparable NLLs to ACNet. In addition, out of the three methods, training with maximum likelihood achieved the best results; however, its increased computation cost, due to computing derivatives and inverses, motivates the use of the alternative losses discussed earlier.

Samples from the learned copulas are compared to the ground truth in Figures 3.2 and 3.3. We additionally note the differences in sampling time between our method and the conditional sampling method used in [83]. The time to generate 3000 samples using our method was on average 3.8×10^{-2} seconds. In comparison, the conditional sampling method that used automatic differentiation of the copula followed by inversion of the conditional distribution, takes on average $1.98 \times 10^{+2}$ seconds, the difference on the order of 3 magnitudes.

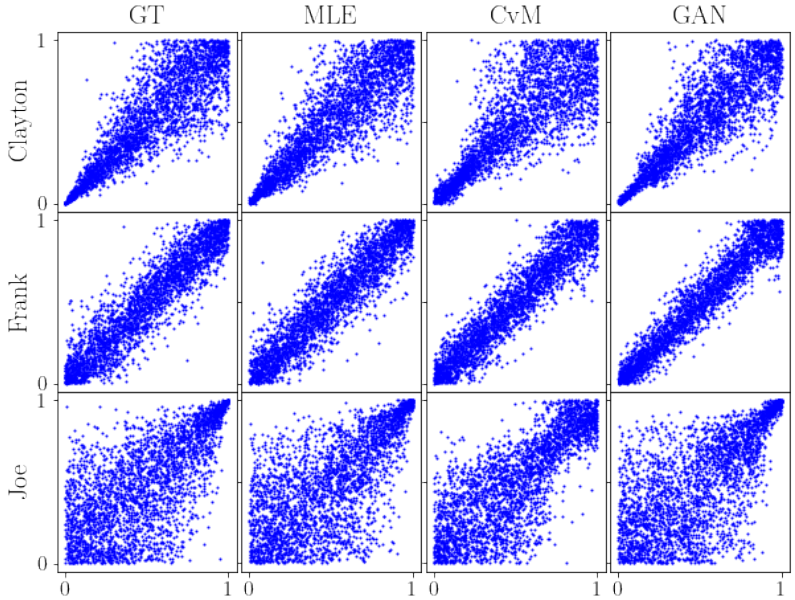

FIGURE 3.2: Samples from fitting known Archimedean copulas

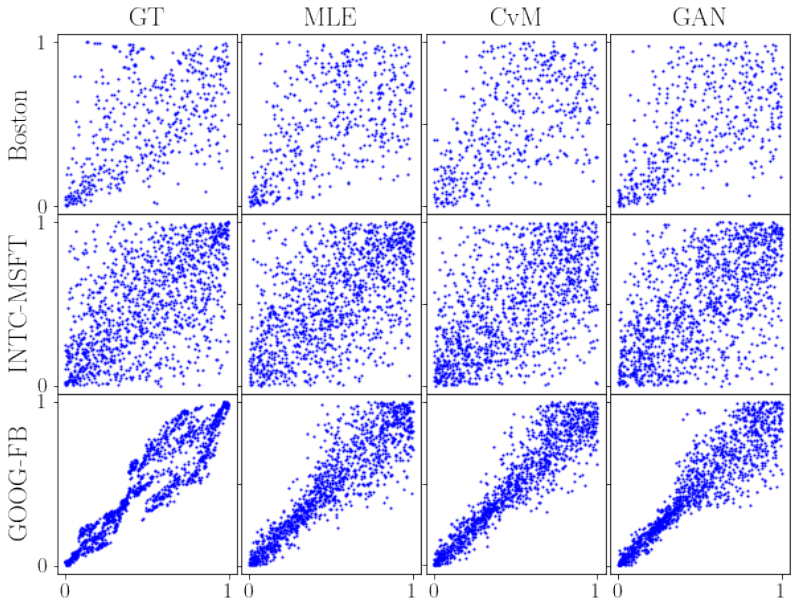

FIGURE 3.3: Samples from fitting real-world data

3.6.2 Learning the Latent Distribution of the Archimedean Copula

Gen-AC learned the latent Gamma distributions whose Laplace transforms give the generator functions of Clayton copulas. We show the learned latent distributions for Clayton copulas with parameters 1, 3, 5, 8 in Figure 3.4.

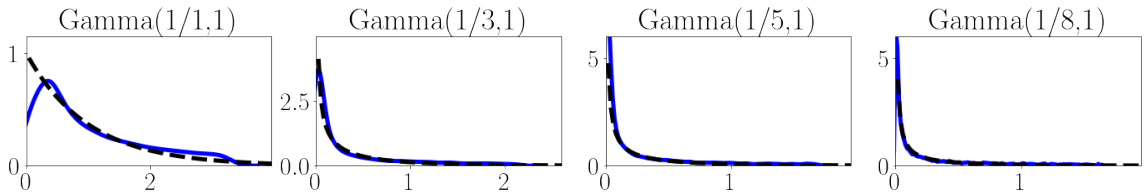

FIGURE 3.4: Learnt latent Gamma distributions of Clayton copulas with parameters 1, 3, 5, 8. The learnt distribution is given in solid blue; the ground truth is given in dashed black.

In addition, we note that compared to vanilla GAN [44], our generating network must satisfy an Archimedean copula. We show this via the training progression for learning a Clayton copula in Figure 3.5.

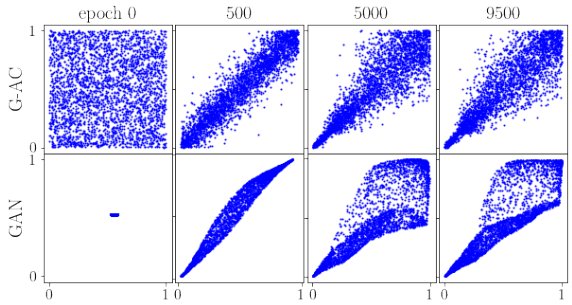

FIGURE 3.5: Samples from learnt Archimedean copula compared to GAN, training progression at epochs 0, 500, 5000 and 10000, for a Clayton copula. Samples from our copula, shown on top, must satisfy an Archimedean copula, while that from a vanilla GAN, shown below, may not.

3.6.3 Learning High-Dimensional Archimedean Copulas

The Laplace transform representation allows efficient computation of higher-order derivatives without automatic differentiation. We fitted Clayton, Frank and Joe copulas for 10 and 20 dimensions. The NLLs are given in Table 3.3. When compared to the ground truth NLLs for 10- and 20-dimensional datasets, the learned NLLs were off by 2%. During our experiments, we could not obtain a reasonably trained ACNet for high dimensions due to the computational complexity.

Table 3.3: Negative log-likelihoods of fitting higher-dimensional data

	Ground Truth		**Gen-AC**	
Known AC	10-dim	20-dim	10-dim	20-dim
Clayton	-10.6	-23.2	-10.4	-22.8
Frank	-10.4	-23.1	-10.4	-23.1
Joe	-5.4	-12.2	-5.3	-12.0

Moreover, while the CPU runtimes of ACNet for computing the copula density increases exponentially with dimensions, the CPU runtimes of computing the copula density using the Laplace transform representation increases linearly with dimensions, as shown in Figure 3.6.

3.6.4 Learning Hierarchical Archimedean Copulas

We demonstrate that our model can represent more complex dependence structures, beyond the exchangeability implied by the functional symmetry of Archimedean copulas, and learn hierarchical Archimedean copulas.

We experiment with fitting a four-variate hierarchical Archimedean copula given by $C_{\varphi_0}(C_{\varphi_1}(u_1, u_2), C_{\varphi_2}(u_3, u_4))$. The ground truth was generated using the state-of-the-art HACopula Toolbox [48]. Samples from the learned copulas are compared to the ground truth in Figure 3.7. In Figure 3.7, $C_{\varphi_0}, C_{\varphi_1}, C_{\varphi_2}$ are Clayton copulas with

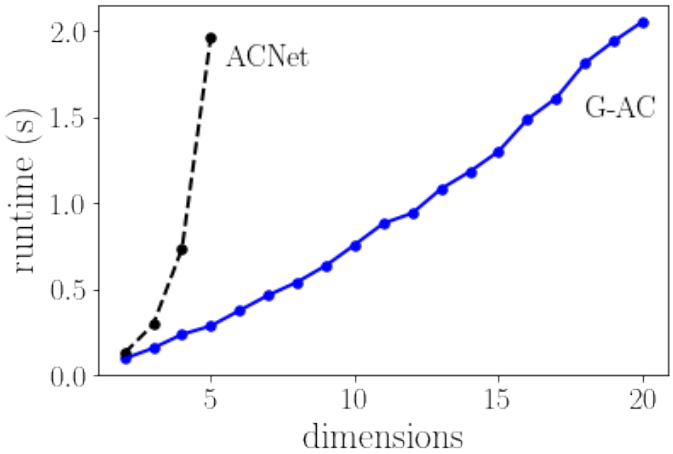

FIGURE 3.6: CPU runtimes for computing the likelihood of 3000 samples. The runtimes for Gen-AC is given in solid blue; ACNet [83] in dashed black.

parameters 1, 3, and 8. We let the outer generator be a generative Archimedean generator. In Figure 3.8, $C_{\varphi_0}, C_{\varphi_1}, C_{\varphi_2}$ are Clayton, '12' and '19' with parameters 0.5, 3, and 1, the numbering of the generators follow [48, 98]. Since our model is compatible with outer generators of other forms, we let the outer generator be a Clayton generator instead of a generative Archimedean generator.

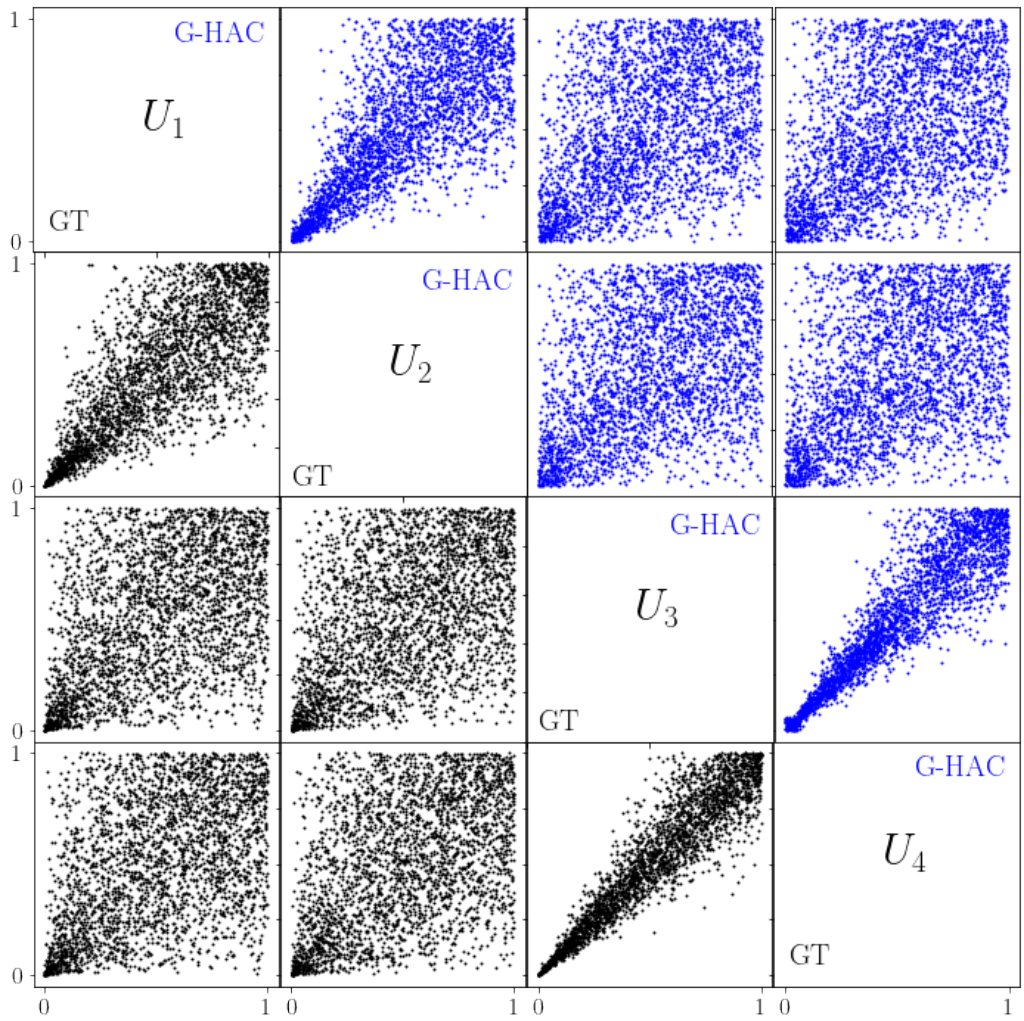

FIGURE 3.7: Samples from fitting a hierarchical Clayton copula displayed as mirrors on the diagonal, Gen-HAC above in blue and ground truth below in black. Each plot is a bivariate margin (U_i, U_j).

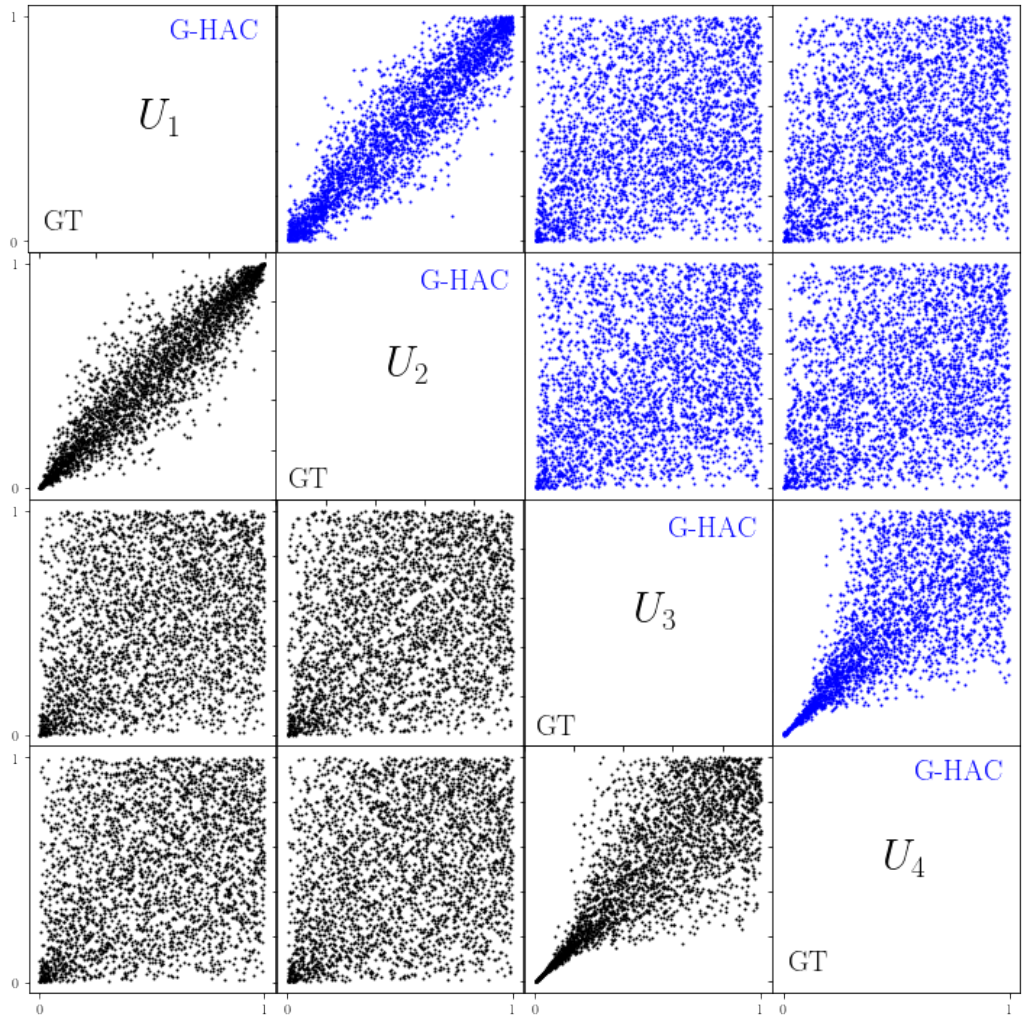

FIGURE 3.8: Samples from fitting a hierarchical Archimedean copula displayed as mirrors on the diagonal, Gen-HAC above in blue and ground truth below in black. Each plot is a bivariate margin (U_i, U_j).

3.7 Summary

We modeled Archimedean and hierarchical Archimedean copulas with generative neural networks based on their probabilistic constructions as mixture and nested mixture models with latent random variables. We described three methods for fitting the model to data: maximum likelihood with the copula density, goodness-of-fit with the Cramér von-Mises statistic and adversarial training by minimizing a divergence between true samples from data and fake samples from the copula. We modified efficient Marshall-Olkin type sampling algorithms to our parameterization of latent variables with generative neural networks. In addition, we described the application of the above estimation and sampling techniques to an existing network representation. Empirically, the generative Archimedean copula was able to learn known Archimedean copulas with different tail behaviours and fit real-world data. We showed results in higher dimensions and an extension of the above techniques to hierarchical Archimedean copulas.

4

Neural Extreme-Value Copulas

4.1 Introduction

Extreme-value copulas

$$C(\mathbf{u}) = \exp(-\ell(-\log(u_1), \cdots, -\log(u_d)), \tag{4.1}$$

as limits of copulas of component-wise maximas, are useful tools motivated by extreme-value theory for modeling rare events where observations are limited.[1] We contribute by developing flexible models that respect the properties of the extreme-value copula, allow training using the likelihood of transformed observations, and allow sampling from the learnt copula.[2] We give two neural representations, both rooted in the spectral decomposition of the stable tail dependence function (stdf). In the generative representation, we model the latent spectral variables with generative neural networks and consider the empirical expectation using samples of the spectral variables. In the network representation, we model the support of the spec-

[1] Part of the material presented in this chapter was co-authored with Ali Hasan, Khalil Elkhalil, João M. Pereira, Sina Farsiu, Jose Blanchet and Vahid Tarokh [50].

[2] In contrast, classical techniques do not automatically uphold the properties of the extreme-value copula and do not automatically enable sampling from the learnt copula.

tral variables with weights of a network designed to satisfy the properties of the stdf. Both representations are equivalent, the generative representation is more suitable for sample generation while the network representation is more suitable for computing statistical quantities. We train the copula using maximum likelihood on exponentially distributed transformed observations with rate given by the stdf, motivated by the Pickands estimator. We sample the learnt copula using techniques based on the spectral representation of max-stable processes.

4.2 Modeling the Stable Tail Dependence Function

From Section 2.3.1 the stdf ℓ is given by a spectral decomposition with spectral variable $\mathbf{W} > 0$ such that:

$$\ell(\mathbf{x}) = \mathrm{E}_{\mathbf{W}}\left[\max_{\mathrm{j}\in\{1,\cdots,\mathrm{d}\}}\{\mathrm{x_j W_j}\}\right], \tag{4.2}$$

with moment constraints $\mathrm{E}_{\mathbf{W}}[\mathrm{W_j}] = 1$ for $j \in \{1, \cdots, d\}$ [27, 62, 108].

For the generative representation, we first scale the expectation by d and the spectral variable by $1/d$. We then let $\mathbf{W} \in \Delta_{d-1}$ be the output of a generative network such that a sample $\mathbf{w} = (w_1, \cdots, w_d)$ is computed as $\mathbf{w} = g(\epsilon; \theta)$ where $g : \mathbb{R}^d \to \Delta_{d-1}$ is a generative network with parameters θ and ϵ is a sample from a source of randomness. We then approximate the expectation with its empirical version using samples $(w_{11}, \cdots, w_{1d}), \cdots, (w_{L1}, \cdots, w_{Ld})$ from $g(\cdot, \cdots, \cdot; \theta)$ as:

$$\ell(\mathbf{x}) = d\,\mathrm{E}_{\mathbf{W}}\left[\max_{\mathrm{j}\in\{1,\cdots,\mathrm{d}\}}\{\mathrm{x_j W_j}\}\right] \approx \frac{\mathrm{d}}{\mathrm{L}}\sum_{\mathrm{l}=1}^{\mathrm{L}}\left[\max_{\mathrm{j}\in\{1,\cdots,\mathrm{d}\}}\mathrm{x_j w_{kj}}\right]. \tag{4.3}$$

We let $g(\cdot; \theta)$ be a multilayer perceptron (MLP). Unlike the modeling of convex functions with neural networks [2], there are no restrictions on the weights and intermediate activations of $g(\cdot; \theta)$. Empirically, we notice that batch normalization of the intermediate layers help in training. To ensure an output in the unit simplex,

we use softmax$(\cdot)$ as the output activation. To maintain the moment constraints, we add a penalty term such that $\mathrm{E}_{\mathbf{W}}[\mathrm{W_j}] = 1/\mathrm{d}$ for $j \in \{1, \cdots, d\}$. The moment constraints are scale conveniences and not necessities.[3]

We next present a network parameterization inspired by Maxout Networks [43], Input Convex Neural Networks [2], hierarchical stable tail dependence functions [56] and the results of Chapter 3. The stdf is given as a composition stdfs [4], specifically:

$$\ell_{0,i}(\mathbf{x}) = \max_{j=1,\cdots,d} A_{0,i,j} x_j, \tag{4.4}$$

$$\ell_{l,i}(\mathbf{x}) = \max_{j=1,\cdots,H_{l-1}} A_{l,i,j} \ell_{l-1,j}(\mathbf{x}), \tag{4.5}$$

$$\ell(\mathbf{x}) = \ell_{L+1,1}(\mathbf{x}) = \frac{1}{H_L} \sum_{i=1}^{H_L} (\ell_L \circ \ell_{L-1} \circ \cdots \circ \ell_0(\mathbf{x}))_i, \tag{4.6}$$

where there are L hidden layers with widths H_l and each A_l is a positive matrix of size $H_l \times H_{l-1}$ such that elements $A_{l,i,j} > 0$, e.g. with weight activation [83, 99] or clamping [2, 3, 50]. To satisfy the moment constraints on the spectral decomposition of ℓ such that $\ell(\mathbf{e}_j) = 1$ and $\max_j x_j \leqslant \ell(\mathbf{x}) \leqslant 1$, where $\ell(\mathbf{x}) = \max_j x_j$ and $\ell(\mathbf{x}) = \|\mathbf{x}\|_1 \equiv 1$ describe completely dependent tails and independent tails, the canonical basis vectors are used to normalize the input such that $\mathbf{x} = (\tilde{x}_1/\ell(\mathbf{e}_1), \cdots, \tilde{x}_d/\ell(\mathbf{e}_d))$ where $\tilde{\mathbf{x}}$ is the input prior to normalization. In addition, to maintain the moment constraints, we add a penalty term such that $\ell(\mathbf{e}_j) = 1$.

There are two proofs that the network represents valid stdfs, both rooted in the spectral decomposition. The first proof relies on the composition of d-norms [35, 56], where the definition of a d-norm is equivalent to the definition of the stdf, i.e. $\ell(\mathbf{x}) = \|\mathbf{x}\|_d = \mathrm{E}_{\mathbf{W}}(\|\mathbf{x}\mathbf{W}\|_\infty)$.[5] In addition, compositions of d-norms are d-norms and thus

[3] They give uniform margins for the extreme-value copula and unit Fréchet margins for the max-stable process. They are not necessary in maintaining the d-max decreasing property of the stdf [38].

[4] The architecture presented in [50] is slightly different and called d-max neural networks (dMNNs).

[5] For more details on d-norms, refer to [35].

compositions of stdfs are stdfs.[6] The functions $\ell_{0,i}(\mathbf{x})$ and $\ell_{l,i}(\mathbf{x})$ represent the $(0,i)$th and (l,i)th stdfs with the supports of $\mathbf{W}_{0,i}$ and $\mathbf{W}_{l,i}$ as single delta functions on $A_{0,i,\cdot}$ and $A_{l,i,\cdot}$. The key to the d-max decreasing property is the positiveness of the support, which is achieved with $A_{l,i,j} > 0$. The second proof computes the finite support of $\mathbf{W}$, where the kth term is expressed as:

$$\mathbf{w}_k = (w_{k1}, \cdots, w_{kd}) = ((\ell_L \circ \ell_{L-1} \circ \cdots \circ \ell_0(\mathbf{e}_1))_k, \cdots, (\ell_L \circ \ell_{L-1} \circ \cdots \circ \ell_0(\mathbf{e}_d))_k), \quad (4.7)$$

such that

$$w_{kj} = \max_{j_{L-1}}(A_{L,k,j_{L-1}} \cdots \max_{j_1}(A_{2,\cdot,j_1} \max_{j_0}(A_{1,\cdot,j_0} A_{0,\cdot,j}))), \quad (4.8)$$

a product of $A_{l,i,j}$ with the terms in the product selected by the max operations in a forward path through the layers of the network. We then follow with normalization such that $\mathrm{E}_{\mathbf{W}}[\mathrm{W_j}] = 1$. The size of the support is determined by the width of the last layer, H_L. This can also be seen from the first proof, where the support of each intermediate stdf is a single delta function. While the network representation of the stdf does not encode an exponentially larger support like the network parameterization of the Archimedean generator in Chapter 3, this representation is still useful in improving the loss landscape during training. While a single layer is also a valid stdf, the intermediate layers of the network allow parameter sharing across coordinates, and perhaps more training opportunities for gradient descent.

It is helpful to have both generative and network representations for ℓ. Analogous to the double representations for φ, while both representations are equivalent, each is more suited for a particular task. The generative representation is more suitable for sample generation while the deterministic representation is more suitable for estimating statistical quantities.

[6] For more details on compositions of d-norms, refer to [56].

4.3 Training with Connection to Pickands Estimator

Since the extreme-value copula is given as a CDF, computing the density for likelihood-based training may be challenging due to the need to differentiate the copula d times. In addition, our parameterization of the stdf will lead to singularities where the copula density does not exist. Instead, motivated by the Pickands estimator [12, 105], we transform d-dimensional observations into one-dimensional exponentially distributed observations with rate given by the stdf. We then train the stdf using the likelihood of transformed observations [50].

From Section 2.3.2, the Pickands transform $\xi(\mathbf{u}, \mathbf{x})$ for an observation $\mathbf{u}$ and a pseudo-observation $\mathbf{x} = (x_1, \cdots, x_d) \in \Delta_{d-1}$ is given as

$$\xi(\mathbf{u}, \mathbf{x}) = \min\{-\log(u_1)/x_1, \cdots, -\log(u_d)/x_d\}, \tag{4.9}$$

such that $\xi(\mathbf{U}, \mathbf{x})$ is an exponential with rate $\ell(\mathbf{x})$. In classical Pickands estimators [12, 105], the values of $\ell(\mathbf{x})$ are then estimated as the reciprocal of the mean of the transformed observations, where convergence is guaranteed by the central limit theorem. Instead, we sample $\mathbf{x}$ uniformly on the unit simplex Δ_{d-1} and train $\ell(\cdot, \cdots, \cdot; \theta)$ to approximate ℓ by minimizing the negative log-likelihood of transformed observations with the log-likelihood of the transformed observation x derived as:

$$\log \mathcal{L}(x; \ell) = -x\ell(x) + \log(\ell(\mathbf{x})). \tag{4.10}$$

In addition, we may train for a correspondence between the generative and network representations with the integrated relative absolute error (IRAE), defined in Section 2.5.4.

4.4 Sampling with Connection to Max-Stable Processes

From Section 2.3.3, the extreme-value copula is the copula of a max-stable process evaluated at a finite subset of points

$$\mathbf{X} = (X_1, \cdots, X_d) = \sup_{i=1,2,\cdots} (P_i W_{i1}, \cdots, P_i W_{id}), \tag{4.11}$$

where $P_1, P_2, \cdots$ are i.i.d. observations of a Poisson point process on $[0, \infty)$ with intensity $x^{-2}\mathrm{dx}$, and $(W_{11}, \cdots, W_{1d}), (W_{21}, \cdots, W_{2d}), \cdots$, are i.i.d. observations of the spectral variable in the spectral decomposition of the stdf ℓ. In addition, the marginal CDFs are unit Fréchet. We modify existing approximate sampling of max-stable processes to our representations of the spectral variable [50, 56, 113]. We first sample N Poisson points $p_1, \cdots, p_N$ in decreasing order as $p_i = 1/\sum_{k=1}^{i} e_k$ where e_k are i.i.d. unit exponentials. We then sample N i.i.d. spectral points $\mathbf{w}_1, \cdots, \mathbf{w}_N$. For the generative representation, we compute $\mathbf{w}_i = dg(\epsilon_i, \theta)$, where we scale by d to undo the scaling by $1/d$. For the network representation, we first propagate through the network to obtain the finite support then sample uniformly from the finite support. We then compute $\mathbf{x} = \max_{1 \leqslant i \leqslant N}(p_i \mathbf{w}_i)$ from the max-stable process and normalized $\mathbf{u} = (\exp(-1/x_1), \cdots, \exp(-1/x_d))$ from the extreme-value copula. We summarize the sampling method in Algorithm 4.

Algorithm 4 Sampling extreme-value copulas

Input: $g(\cdot; \theta)$,

1: Sample i.i.d. $p_1, \cdots, p_N$ Poisson points
in decreasing order as $p_i = 1/\sum_{k=1}^{i} e_k$ where e_k are i.i.d. unit exponentials.

2: Sample i.i.d. $\mathbf{w}_1, \cdots, \mathbf{w}_N$ as $\mathbf{w}_i = d\, g(\epsilon_i, \theta)$ where ϵ_i is a noise source.

3: Compute $\mathbf{z} = \max_{1 \leqslant i \leqslant N}(p_i \mathbf{w}_i)$ from the truncated max-stable process
and normalized $\mathbf{u} = (\exp(-1/z_1), \cdots, \exp(-1/z_d))$.

Output: $\mathbf{u}$.

4.5 Experiments

4.5.1 Learning and Sampling Asymmetric Logistic Copula

We consider the asymmetric logistic copula (ASL) for an extreme-value copula that is asymmetric with a known sampling method. We use the parameters $\alpha = (1/2, 1/3)$, $\theta = ((1.e^{-15}, 0.2, 0.8), (1, 0.8, 0.2))$ and stdf defined as $\ell(\mathbf{x}) = \sum_j (\sum_i (\theta_{ij} x_j)^{1/\alpha_j})^{\alpha_j}$.

The IRAE, described in Section 2.5.4, of the learnt stdf is reported in Table 4.1 for 3000 and 240 training samples. The large training sample size was to demonstrate convergence of the method, the small training sample size was to simulate the performance of the method when applied to rare events where observations are limited. The IRAE of the well-established endpoint corrected CFG estimator [12] is given as benchmark. Both our generative and network-based models achieved comparable IRAE to the CFG estimator. We also demonstrated the success of CFG on samples from our models, as another motivation is to learn and sample extreme events for subsequent statistical tasks.

For a qualitative comparison, we also plot the stdfs in Figures 4.1 and 4.2. Samples from the learned copulas are compared to the ground truth in Figures 4.3 and 4.4.

Experiments were conducted using `PyTorch`, on a 2.7 GHz Intel Core i7 with 16 GB of RAM.

Table 4.1: IRAE of fitting stdf of asymmetric logistic copula

	Our Methods			**Benchmark**
sample size	Network	Gen	CFG on our samples	CFG
3000	0.01	0.01	0.02	0.02
240	0.03	0.04	0.03	0.04

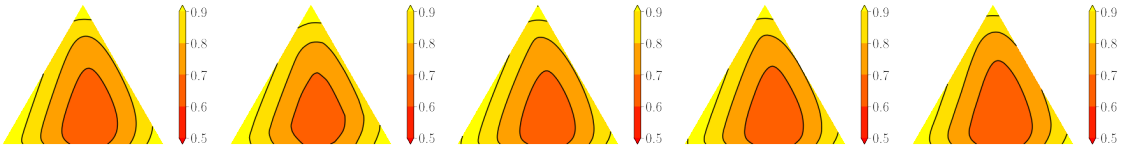

FIGURE 4.1: Stdfs from fitting ASL with 3000 training samples. From left to right, the ground truth, network model, generative model, CFG on 3000 samples from our generative model, CFG as benchmark.

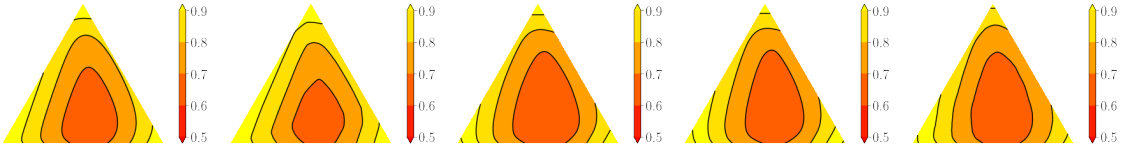

FIGURE 4.2: Stdfs from fitting ASL with 240 training samples. From left to right, the ground truth, network model, generative model, CFG on 3000 samples from our generative model, CFG as benchmark.

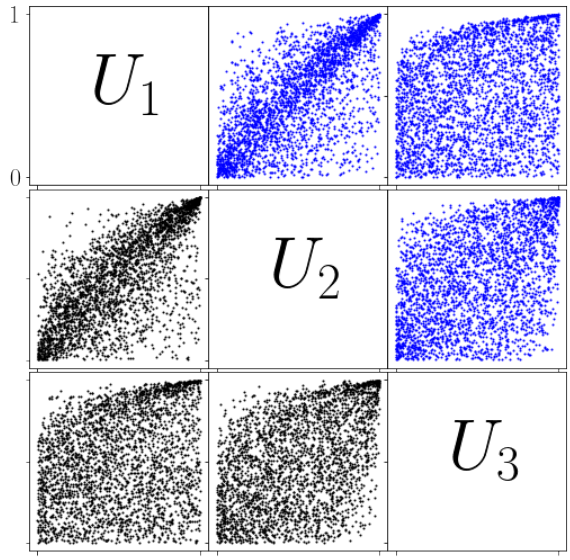

FIGURE 4.3: Samples from fitting ASL with 3000 training samples. Samples are displayed as mirrors on the diagonal, generative extreme-value copulas above in blue and ground truth below in black. Each plot is a bivariate margin (U_i, U_j).

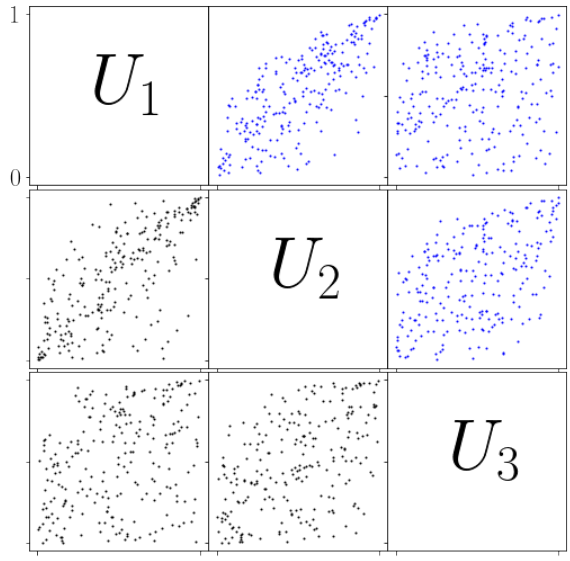

FIGURE 4.4: Samples from fitting ASL with 240 training samples. Samples are displayed as mirrors on the diagonal, generative extreme-value copulas above in blue and ground truth below in black. Each plot is a bivariate margin (U_i, U_j).

4.6 Summary

Building on the spectral decomposition of the stdf, we gave both generative and network representations for the stdf. Analogous to the double representations for Archimedean generators, while both representations are equivalent, the stochastic representation is more suitable for sample generation while the deterministic representation is more suitable for estimating statistical quantities. We trained these models using the log-likelihood of transformed observations, where the transform from d-dimensional observations into multiple one-dimensional exponentially distributed observations with rate given by the stdf is borrowed from the Pickands estimator. We sampled these models using samples of the spectral variables and approximate sampling methods from max-stable processes. We empirically verify training and sampling from the asymmetric logistic copula.

5

Generative Archimax Copulas

5.1 Introduction

Archimax copulas

$$C(\mathbf{u}) = \varphi(\ell(\varphi^{-1}(u_1), \cdots, \varphi^{-1}(u_d))) \tag{5.1}$$

effectively balance representation of data within the bulk and extrapolation to the tails by combining the tractability of *Archimedean* copulas with the tail properties of *extreme-value* copulas. The use of Archimax copulas has resulted in better fit to data in areas such as healthcare [95] and hydrology [4, 19]. However, these applications were of dimensions lower than or equal to three. We contribute by developing flexible representations and scalable methods for inference and sampling.

We build on the stochastic representation of Archimax copulas from Charpentier et al. [18] as the survival copulas of

$$\mathbf{X} = R \times (S_1, \cdots, S_d), \tag{5.2}$$

where R and $\mathbf{S}$ are independent, known as the *radial* and *simplex* components, and

Part of the material presented in this chapter was co-authored with Ali Hasan and Vahid Tarokh [100].

$\mathbf{S}$ has a one-to-one correspondence with the *spectral* component $\mathbf{W}$. Intuitively, R relates to the radial envelope while $\mathbf{W}$ relates to the asymptotic dependence in the tails. In addition, the *Archimedean generator* φ and the *stable tail dependence function* (stdf) ℓ may be represented as expectations of R[1] and $\mathbf{W}$:

$$\varphi(x) = \mathrm{E_R}\left[\left(1 - \frac{\mathrm{x}}{\mathrm{R}}\right)_+^{\mathrm{d}-1}\right], \tag{5.3}$$

$$\ell(\mathbf{x}) = d\,\mathrm{E}_{\mathbf{W}}\left[\max_{\mathrm{j}\in\{1,\cdots,\mathrm{d}\}}\{\mathrm{x_j W_j}\}\right]. \tag{5.4}$$

We first model these expectations where R and $\mathbf{W}$ are discrete random variables with finite support [38, 42]. Building on the results of Chapters 3 and 4, we let them be outputs of generative networks $g_r(\cdot;\theta_r)$, $g_{\mathbf{w}}(\cdot;\theta_{\mathbf{w}})$ in the limit of infinite support:

$$\varphi_\theta(x) = \frac{1}{n_r}\sum_{j=1}^{n_r}\left(1 - \frac{x}{r_j}\right)_+^{d-1}, \tag{5.5}$$

$$\ell_\theta(\mathbf{x}) = \frac{d}{n_w}\sum_{k=1}^{n_w}\left[\max_{j\in\{1,\cdots,d\}} x_j w_{kj}\right]. \tag{5.6}$$

The rest of this chapter then focuses on learning and sampling the Archimax copula using the generative representations. We first describe the training update for ℓ_θ followed by the training update for φ_θ, which uses samples of $\mathbf{S}$, derived from samples of $\mathbf{W}$. Parameter estimation for ℓ_θ and φ_θ is iterative, described in the flowchart given in Figure 5.1. Sampling the Archimax copula is straightforward given samples of R and $\mathbf{S}$.

---

[1] In this chapter, we work with the more general class of *d-monotone* φ which coincides with the class of Williamson d-transforms of a positive random variable $R > 0$. In Chapter 3, we worked with the more restricted class of *completely monotone* φ which coincided with the class of Laplace transforms of a positive random variable $M > 0$. The technique for approximating φ with samples from generative networks, computing derivatives by properties of the expectation and computing inverse is analogous. The d-monotone representation allows a further inference technique for φ which we will discuss in this chapter.

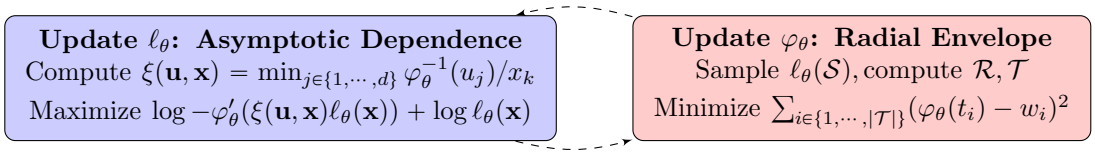

FIGURE 5.1: Flow chart of iterative stdf and Archimedean generator updates

5.2 Training the Stable Tail Dependence Function

For this section, we suppose that the Archimedean generator φ is known and the goal is to update the modeled stdf ℓ_θ. Building on the modified Pickands estimator [19] and the results of Chapter 4, we define the likelihood of transformed data observations and update the parameters of ℓ_θ to maximize this likelihood.

Given the Archimedean generator φ, we can define the transformation $\xi(\mathbf{u},\mathbf{x})$ for an observation $\mathbf{u}$ and a pseudo-observation $\mathbf{x} = (x_1, \cdots, x_d) \in \Delta_{d-1}$ as

$$\xi(\mathbf{u},\mathbf{x}) = \min\{\varphi^{-1}(u_1)/x_1, \cdots, \varphi^{-1}(u_d)/x_d\}, \tag{5.7}$$

a generalization of the transformation used in the Pickands estimator for estimating extreme-value copulas [12, 19, 105]. By the definition of the Archimax copula, the decreasing property of φ and the homogeneity property of ℓ, the CDF of the random variable $\xi(\mathbf{U},\mathbf{x})$ is expressed as:

$$P(\xi(\mathbf{U},\mathbf{x}) \leqslant x) = 1 - P(\xi(\mathbf{U},\mathbf{x}) > x) = 1 - C(\varphi(x\mathbf{x})) = 1 - \varphi(x\ell(\mathbf{x})). \tag{5.8}$$

We differentiate the CDF of $\xi(\mathbf{U},\mathbf{x})$ with respect to x and obtain the log-likelihood of the transformed observation x as:

$$\log \mathcal{L}(x;\varphi,\ell) = \log\left(-\varphi'(x\,\ell(\mathbf{x}))\right) + \log(\ell(\mathbf{x})). \tag{5.9}$$

We may then train the generative network $g_{\mathbf{w}}(\cdot;\theta_{\mathbf{w}})$ such that ℓ_θ approximates ℓ by minimizing the negative log-likelihood (NLL) of transformed observations $x = \xi(\mathbf{u},\mathbf{x})$, with $\mathbf{x}$ uniformly sampled on the unit simplex Δ_{d-1}.

The inverse φ^{-1} in the computation of $\xi(\mathbf{u}, \mathbf{x})$ in (5.7) can be computed numerically by Newton's method. The derivative φ' in the computation of $\log \mathcal{L}(x; \varphi, \ell)$ in (5.9) can be calculated explicitly from 5.3 as:

$$\varphi'(x) = \mathrm{E_R}\left[(\mathrm{d}-1)\left(-\frac{1}{\mathrm{R}}\right)\left(1-\frac{\mathrm{x}}{\mathrm{R}}\right)_+^{\mathrm{d}-2}\right]. \tag{5.10}$$

Putting these computations together results in the likelihood of ℓ_θ given φ. We summarize the method for training ℓ_θ given φ in Algorithm 5 here:

Algorithm 5 Learn stable tail dependence function (stdf)

input observations $\{\mathbf{u}_i : i = 1, ..., m\}$.
input Archimedean generator φ.
initialize $g_{\mathbf{w}}$.
do while loss = NLL + reg **not converged**:
  sample $\{\mathbf{x}_j : j = 1, ..., n\}$ from Unif(Δ_{d-1}).
  compute $\{\xi(\mathbf{u}_i, \mathbf{x}_j) : i = 1, ..., m,\ j = 1, ..., n\}$ from (5.7).
  sample $\{\mathbf{w}_k : k = 1, ..., l\}$ from $g_{\mathbf{w}}$.
  compute $\{\ell_\theta(\mathbf{x}_j) : j = 1, ..., n\}$ from (5.6).
  compute NLL $= \frac{1}{mn}\sum_{i=1,j=1}^{i=m,j=n} -\log \mathcal{L}(\xi(\mathbf{u}_i, \mathbf{x}_j); \varphi, \ell_\theta)$ from (5.9).
  compute reg $= \sum_{j=1}^{d}(\sum_{k=1}^{l} w_{kj}/l - 1/d)^2$.
  descent $\mathrm{argmin}_{g_{\mathbf{w}}}$ NLL + reg.
end while
return learned $g_{\mathbf{w}}$.

5.3 Sampling the Simplex Component

We now consider sampling the simplex component $\mathbf{S}$ through sampling the spectral component $\mathbf{W}$. From the result of Charpentier et al. [18], $\mathbf{S}$ has survival distribution

$$P(\mathbf{S} > \mathbf{s}) = \max(0, 1 - \ell(s_1, \cdots, s_d))^{d-1}, \tag{5.11}$$

which relates to the so-called generalized Pareto distribution

$$P(X_1 < x_1, \cdots, X_d < x_d) = \max(0, 1 - \ell(-x_1, \cdots, -x_d)). \tag{5.12}$$

with stochastic representation

$$\mathbf{X} \stackrel{d}{=} -\left(\frac{U}{W_1}, \cdots, \frac{U}{W_d}\right), \tag{5.13}$$

where U is uniformly distributed on $[0, 1]$, independent of spectral component $\mathbf{W} = (W_1, \cdots, W_d)$ from the spectral decomposition of ℓ in (5.4) [11, 35].

Thus, given samples $(x_{11}, \cdots, x_{1d}), \cdots, (x_{(d-1)1}, \cdots, x_{(d-1)d})$ from a generalized Pareto distribution with CDF in (5.12), we may obtain a sample of $\mathbf{s}$, where coordinates s_j for $j = \{1, \cdots, d\}$ are computed as [18]:

$$s_j = -\max(x_{1j}, \cdots, x_{(d-1)j}). \tag{5.14}$$

The coordinate-wise maxima is taken over $d-1$ samples to get the $d-1$ exponent in the SDF of $\mathbf{S}$, corresponding to the $d-1$ exponent in the expression of φ.

Putting these computations together with samples of $\mathbf{W}$ from $g_{\mathbf{w}}(\cdot; \theta_{\mathbf{w}})$ gives us samples of $\mathbf{S}$. We summarize the method for sampling $\mathbf{S}$ given samples of $\mathbf{W}$ in Algorithm 6 here:

Algorithm 6 Sample simplex component

input learned $g_{\mathbf{w}}$.
do while $\ell_\theta(\mathbf{s}) > 1$:
  sample $\{u_i : i = 1, ..., d-1\}$ from Unif$(0, 1)$.
  sample $\{(w_{i1}, ..., w_{id}) : i = 1, ..., d-1\}$ from $g_{\mathbf{w}}$.
  compute $\{(x_{i1}, ..., x_{id}) : i = 1, ..., d-1\}$ from (5.13).
  compute $\mathbf{s} = (s_1, ..., s_d)$ from (5.14).
  compute $\ell_\theta(\mathbf{s})$ from (5.6).
end while
return a sample $\mathbf{s} = (s_1, ..., s_d)$.

As a side note, from Definition 5.11, the marginals of the simplex component are distributed as Beta$(1, d-1)$, where d is the dimension. To enforce the marginals, we compute the samples of the empirical copula $\mathbf{u}$ from unnormalized samples $\tilde{\mathbf{s}}$, then

apply the quantile function, i.e. inverse cumulative distribution function (CDF), to obtain the normalized samples $\mathbf{s}$ such that:

$$s_j = 1 - (1 - u_j)^{\frac{1}{d-1}} \text{ for } j = \{1, \cdots, d\}. \tag{5.15}$$

5.4 Training the Archimedean Generator

We now assume that the stdf ℓ is known, we have samples of $\mathbf{S}$ from $\mathbf{W}$, and the goal is to infer the Archimedean generator φ. We consider the general representation of φ as a d-monotone function specified by the Williamson d-transform of the radial component R in (5.3) [18, 96]. We first model the expectation where R is a discrete random variable with finite support [42], then let it be the output of a generative network in the limit of infinite support.

To provide an outline for the overall approach: we first recall the Kendall distribution which describes an integral transform of the copula given in Section 2.4.3. We then use the fact that the empirical Kendall distribution converges to the true Kendall distribution as the empirical copula converges to the true copula, providing a means of estimation.

By the definition of the Archimax copula, the stochastic representation and the homogeneity property of ℓ, the Kendall distribution is expressed as:

$$K(w) = P(\varphi(R\,\ell(S_1, \cdots, S_d)) \leqslant w). \tag{5.16}$$

We then consider the empirical Kendall distribution K_n for n given observations $(u_{11}, \cdots, u_{1d}), \cdots, (u_{n1}, \cdots, u_{nd})$, defined as:

$$K_n(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{w_i \leqslant w\}, \tag{5.17}$$

where

$$w_i = \frac{1}{n+1} \sum_{k=1}^{n} \mathbb{1}\{u_{k1} < u_{i1}, \cdots, u_{kd} < u_{id}\}. \tag{5.18}$$

In the case of non-parametric inference of Archimedean copulas [42], where dependence is symmetric, $\mathbf{S}$ is uniformly distributed on the unit simplex and $\ell(S_1, \cdots, S_d) = S_1 + \cdots + S_d \equiv 1$. Then, R is discrete with the cardinality of the support the same as W and K_n is the Kendall distribution of a unique Archimedean copula [42].

In the non-parametric inference of Archimax copulas, we define the random variables:

$$Z := \ell(S_1, \cdots, S_d) \quad \text{and} \quad T := RZ \tag{5.19}$$

where T is a discrete random variable with support the same size as W and K_n is the Kendall distribution of an Archimax copula. Note that R and Z are independent as R and $\mathbf{S}$ are independent. Using K_n, we reconstruct the support of R, given ℓ and the support of $\mathbf{S}$, which in turn provides the support of Z. With the final objective of letting R and $\mathbf{S}$ be i.i.d. outputs of generative networks, we first (linearly) interpolate K_n to be equispaced, such that each w_i has the same probability $k_i = 1/(n_r n_z)$, where n_r and n_z are the chosen sizes of supports for R and Z.[2,3]

We suppose that the supports of the distributions of R, Z, W, T are finite and respectively denoted by:

$$\mathcal{W} = \{w_1, w_2, \cdots, w_{n_r n_z}\}, \tag{5.20}$$

$$\mathcal{R} = \{r_1, r_2, \cdots, r_{n_r}\}, \tag{5.21}$$

$$\mathcal{Z} = \{z_1, z_2, \cdots, z_{n_z}\}, \tag{5.22}$$

$$\mathcal{T} = \{r_j z_l : r_j \in \mathcal{R}, z_l \in \mathcal{Z}\} = \{t_1, t_2, \cdots, t_{n_r n_z}\}, \tag{5.23}$$

where the $n_r n_z$ elements of $\mathcal{W}$ are sorted in decreasing (non-increasing) order, and the $n_r n_z$ elements of $\mathcal{T}$ are sorted in increasing (non-decreasing) order. This reverse ordering is due to φ being a decreasing function.

[2] We describe the reconstruction procedure for the general case with non-i.i.d. random variables including cases where $r_j z_l = r_{j'} z_{l'}$ following the i.i.d. case.

[3] The sizes of supports n_r and n_z are chosen empirically with examples in the experiment section.

We minimize the mean sum of square residuals motivated by the uniform convergence of the empirical Kendall process $\sqrt{n}(K_n - K)$ [5, 40, 42]:

$$\frac{1}{n_r n_z} \sum_{i=1}^{n_r n_z} (w_i - \varphi_\theta(t_i))^2 \tag{5.24}$$

where, in (5.5),

$$\varphi_\theta(t_i) = \frac{1}{n_r} \sum_{j=1}^{n_r} \left(1 - \frac{t_i}{r_j}\right)_+^{d-1}. \tag{5.25}$$

The finite support assumption of R and Z is not necessary, and we consider a modification where the supports $\mathcal{R}, \mathcal{Z}$ are specified by samples from generative networks. The main objective is to learn the parameters of the generative network $g_r(\cdot; \theta_r)$ given the empirical Kendall distribution K_n and samples of Z. Since scaling the support $\mathcal{R}$ by a constant $c > 0$ does not change the copula, we add a regularization term for $\mathbb{E}_R[R] = 1$. The algorithm can be understood as an alternating minimization algorithm, where the map between $\mathcal{R}$ and $\mathcal{Z}$ to $\mathcal{W}$ via $\mathcal{T}$, and the support $\mathcal{R}$ are updated in an alternating fashion. We summarize the method for training the φ_θ given ℓ and samples of $\mathbf{S}$ in Algorithm 7 here:

Algorithm 7 Learn Archimedean generator

input Kendall observations $\mathcal{W} = \{w_i : i = 1, ..., n_r n_z\}$.
input learned g_z such that $Z := \ell_\theta(\mathbf{S})$,
  $\mathbf{S}$ sampled from learned $g_{\mathbf{w}}$ with Algorithm 6.
initialize g_r.
sort $\mathcal{W}$ in decreasing (non-increasing) order.
do while MSE $= \sum_{i=1}^{n_r n_z} (w_i - \varphi_\theta(t_i))^2 / (n_r n_z) > \epsilon$:
  sample $\mathcal{R} = \{r_1, ..., r_{n_r}\}$ from g_r.
  sample $\mathcal{Z} = \{z_1, ..., z_{n_z}\}$ from g_z.
  compute $\mathcal{T} = \{r_j z_l,\ r_j \in \mathcal{R}, z_l \in \mathcal{Z}\}$.
  sort $\mathcal{T}$ in increasing (non-decreasing) order.
  compute $(\varphi_\theta(t_1), ..., \varphi_\theta(t_{n_r n_z}))$ from (5.25).
  descent $\text{argmin}_{g_r} \sum_{i=1}^{n_r n_z} (w_i - \varphi_\theta(t_i))^2 / (n_r n_z) + (\sum_{j=1}^{n_r} r_j / n_r - 1)^2$.
end while
return learned g_r.

Reconstruction with Non-IID Random Variables and Repetition of Elements We first summarize the steps in obtaining the expression of the Kendall function for Archimax copulas here:

$$
\begin{aligned}
K(w) &= P(C(U_1, \cdots, U_d) \leqslant w), \\
&= P(\varphi(\ell(\varphi^{-1}(U_1), \cdots, \varphi^{-1}(U_d))) \leqslant w), && C(\mathbf{u}) := \varphi(\ell(\varphi^{-1}(\mathbf{u})), \\
&= P(\varphi(\ell(RS_1, \cdots, RS_d)) \leqslant w), && \mathbf{U} \stackrel{d}{=} \varphi(R\mathbf{S}), \\
&= P(\varphi(R\,\ell(S_1, \cdots, S_d)) \leqslant w), && \ell(c\,\mathbf{s}) = c\,\ell(\mathbf{s}), c > 0, \\
&= P(\varphi(RZ) \leqslant w), && Z := \ell(\mathbf{S}), \\
&= P(\varphi(T) \leqslant w), && T := RZ.
\end{aligned}
$$

To provide an outline for the overall approach, given the current estimate of $\mathcal{R}$, we compute the mapping between $\mathcal{R}, \mathcal{Z}$ and $\mathcal{T}$, solve for the probabilities $P_{\mathcal{R}}$, update the support $\mathcal{R}$, and iterate as needed. The algorithm can be understood as an alternating minimization algorithm, where the map between $\mathcal{R}, \mathcal{Z}$ to $\mathcal{W}$ via $\mathcal{T}$, and the support $\mathcal{R}$ are updated in an alternating fashion.

We initialize by computing $\mathcal{W} = \{w_1, \cdots, w_m\}$ and $P_{\mathcal{W}} = \{p(w_1), \cdots, p(w_m)\}$ of the empirical Kendall distribution function in (5.17) and (5.18), where $m \leqslant n_r n_z$. We do not perform the additional (linear) interpolation step of the main text. We then sort $\mathcal{W}$ in decreasing order. We also initialize $\mathcal{R} = \{r_{n_r} = 1,\, r_j = r_{j+1}\alpha_j$ for $j = \{1, ..., n_r - 1\}\}$, where we select $\alpha = (0.9, ..., 0.9)$. The support $\mathcal{Z}$ and probabilities $P_{\mathcal{Z}}$ are assumed to be given.

Given $\mathcal{Z}$ and the current estimate of $\mathcal{R}$, we compute

$$\mathcal{T} = \{r_j z_l : r_j \in \mathcal{R}, z_l \in \mathcal{Z}\}. \tag{5.26}$$

We then sort $\mathcal{T}$ in increasing order and compute the ordering

$$(\sigma_r, \sigma_z)(i) : \{1, \cdots, m\} \to \{1, \cdots, n_r\} \times \{1, \cdots, n_z\} \tag{5.27}$$

defined as a surjective function such that

$$t_i = r_{\sigma_r(i)} z_{\sigma_r(i)}. \tag{5.28}$$

We solve for the probabilities $P_{\mathcal{R}} = \{p(r_1), \cdots, p(r_{n_r})\}$ by minimizing the residuals

$$\sum_{i=1}^{m} \left(\left(\sum_{(\sigma_r^{-1}(j), \sigma_z^{-1}(l))=i} p(r_j)p(z_l) \right) - p(w_i) \right)^2. \tag{5.29}$$

We solve for the support $\mathcal{R} = \{r_1, \cdots, r_{n_r}\}$, by minimizing the residuals

$$\sum_{i=1}^{m} (\varphi_\theta(t_i) - w_i)^2 \tag{5.30}$$

where, following 2.21,

$$\varphi_\theta(t_i) = \sum_{j=1}^{n_r} p(r_j) \left(1 - \frac{t_i}{r_j}\right)_+^{d-1}. \tag{5.31}$$

We give the details of the algorithm with a simple numerical example.

Consider the supports $\mathcal{R} = (r_1, r_2, r_3) = (1, 2, 3)$, $\mathcal{Z} = (z_1, z_2) = (0.5, 0.75)$ with probabilities $P_{\mathcal{R}} = (0.4, 0.4, 0.2)$, $P_{\mathcal{Z}} = (0.25, 0.75)$ and noisy observations $\mathcal{W} = (0.07, 0.19, 0.34, 0.49, 0.67)$, $P_{\mathcal{W}} = (0.07, 0.33, 0.14, 0.31, 0.15)$.

In this case, $(t_1, t_2, t_3, t_4, t_5) = (0.5, 0.75, 1.0, 1.5, 2.25)$ corresponds to $(r_1z_1, r_1z_2, r_2z_1, r_2z_2 \cup r_3z_1, r_3z_2)$ with probabilities $(0.1, 0.3, 0.1, 0.3 + 0.05, 0.15)$.

For $P_{\mathcal{R}}$, we solve the overdetermined system of linear equations:

$$\begin{pmatrix} 0.25 & & \\ 0.75 & & \\ & 0.25 & \\ & 0.75 & 0.25 \\ & & 0.75 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} 0.07 \\ 0.33 \\ 0.14 \\ 0.31 \\ 0.15 \end{pmatrix} \tag{5.32}$$

and obtain the solution as $(0.43, 0.37, 0.20)$, where the solution has been normalized to sum to 1.

For $\mathcal{R}$, we minimize the residuals in (5.30). Since scaling such that $\mathcal{R} = \{cr_1, \cdots, cr_{n_r}, c > 0\}$ does not change the copula, we solve for $\mathcal{R}$ in terms of ratios $(\alpha_1, \cdots, \alpha_{n_r-1})$, recursively defined such that $r_{n_r} = 1$ and $r_j = r_{j+1}\alpha_j$ for $j = \{1, \cdots, n_r - 1\}$.

We summarize the reconstruction with non-iid random variables and repetition of elements in Algorithm 8 here:

Algorithm 8 Estimate radial component

input $\mathcal{W} = \{w_i : i = 1, ..., m\},\ P_{\mathcal{W}} = \{p(w_i) : i = 1, ..., m\}$.
input $\mathcal{Z} = \{z_l : l = 1, ..., n_z\},\ P_{\mathcal{Z}} = \{p(z_l) : l = 1, ..., n_z\}$.
initialize $(\alpha_1, ..., \alpha_{n_r-1})$ for instance $(0.9, ..., 0.9)$.
sort $\mathcal{W}$ in decreasing order.
do while MSE $= \frac{1}{m}\sum_{i=1}^{m}(\varphi_\theta(t_i) - w_i)^2 > \epsilon$:
  compute $\mathcal{R} = \{r_{n_r} = 1,\ r_j = r_{j+1}\alpha_j$ for $j = 1, ..., n_r - 1\}$.
  compute $\mathcal{T} = \{r_j z_l,\ r_j \in \mathcal{R}, z_l \in \mathcal{Z}\}$.
  sort $\mathcal{T}$ in increasing order.
  compute $\{(\sigma_r, \sigma_z)(i) : i = 1, ..., m\}$ from (5.28).
  solve $P_{\mathcal{R}}$ from (5.29).
  compute $(\varphi_\theta(t_1), ..., \varphi_\theta(t_m))$ from (5.31).
  solve $\text{argmin}_\alpha \sum_{i=1}^{m}(w_i - \varphi_\theta(t_i))^2$ such that $\alpha \in (\alpha_l, \alpha_u)$ for instance (0.01,1).
end while
return Estimated support $\mathcal{R}$ and probabilities $P_{\mathcal{R}}$.

The ratios α may be solved iteratively for a unique solution. In our case, motivated by the uniform convergence of the empirical process $\sqrt{n}(K_n - K)$ as $n \to \infty$ [5], we optimize for a least-squares solution with bounds $\alpha \in (0.01, 1)$ using `scipy.optimize.least_squares` [10].

The disadvantage of the above general approach that directly solves for the radial component compared to the approach where the radial component is represented by a generative network is the direct relationship between the computation cost and n_r, n_z, the sizes of supports for R, Z.

5.5 Sampling the Radial Component

The learned g_r provides a source of samples for the radial component R, which we subsequently use to generate full samples from the Archimax copula.

5.6 Training and Sampling the Archimax Copula

The combination of training the stdf, sampling the simplex component, training the Archimedean generator, and sampling the radial component culminates into an iterative technique for training and sampling Archimax copulas with successive updates.

We begin by describing the training process. We initialize with $\varphi(x) = \exp(-x)$ and infer ℓ to learn $g_{\mathbf{W}}(\cdot; \theta_{\mathbf{W}})$, following Section 5.2. This special combination of φ and ℓ corresponds to extreme-value dependence with the max-stable property. To aid inference for this initialization step, we pre-process our data to have extreme-value dependence. We do so by computing the *block maxima*, a technique from extreme-value copulas where we group observations and take the coordinate-wise maximas within each group, as described in Section 2.3. Specifically, given observations $(x_{11}, \cdots, x_{1d}), \cdots, (x_{n1}, \cdots, x_{nd})$, the block-maximas $(m_{11}, \cdots, m_{1d}), \cdots, (m_{k1}, \cdots, m_{kd})$ for k blocks of size n/k are computed as:

$$m_{ij} = \max(x_{(n/k)(i-1)+1\,j}, \cdots, x_{(n/k)(i)\,j}), \tag{5.33}$$

for $i = \{1, \cdots, k\}$ and $j = \{1, \cdots, d\}$. To determine the block size, with larger block sizes n/k better approximating extreme-value dependence and more blocks k for more observations, we test the block maximas for extreme-value dependence via the max-stable property, as described in Section 2.3, using the test by Kojadinovic et al. [75], and select the first k where the null hypothesis of extreme-value dependence is not rejected, in decreasing order starting from $k = n$. Specifically, we compute the Cramér–von Mises (CvM) statistic, as described in Section 2.5.2, for a correspondence

between $C_k(u_1, \cdots, u_d)$ and $C_k^r(u_1^{1/r}, \cdots, u_1^{1/r})$ for $r \in \{1, 2, \cdots\}$ where C_k is the empirical copula of the block maximas and randomizing the order of observations may help to create different block-maximas in each test.

In addition, we considered initialization with different single-parameter families of Archimedean generators, with choice of generator based on the highest log-likelihood of transformed observation ξ, from equations (5.7) and (5.9). The parameter for each family may be computed from an average of inversion of pairwise Kendall tau, as per the following equation [13], for each pair:

$$\tau_{\varphi,\ell} = \tau_\ell + (1 - \tau_\ell)\tau_\varphi, \tag{5.34}$$

where $\tau_{\varphi,\ell}$ is the Kendall's tau of the Archimax bivariate marginal, τ_ℓ is the Kendall's tau of the extreme-value component and τ_φ is the Kendall's tau of the Archimedean component. An average of inversion of pairwise Kendall tau was employed in [19], with emphasis on the Clayton generator.

Initialization with specific families of Archimedean generators might bias initialization, and thus we suggest pre-processing the initial data to have extreme-value dependence using block-maximas and initialization using the Archimedean generator first set to $\varphi(x) = \exp\{-x\}$ representing extreme-value copulas. This is also motivated by the experiment on extrapolating to extremes.

Given an estimate of ℓ and a learned $g_{\mathbf{W}}(\cdot; \theta_{\mathbf{W}})$, we are able to generate many samples of $\mathbf{S}$, thereby providing a way to compute $Z = \ell(\mathbf{S})$. We then infer φ, learning $g_r(\cdot; \theta_r)$, following Section 5.4. At this point, we repeat the estimation for $g_{\mathbf{W}}(\cdot; \theta_{\mathbf{W}})$ with the updated φ to improve our estimate for ℓ.

It follows from the results of Chatelain et al. [19] that given φ, ℓ is unique; and given ℓ, φ is unique. In addition, power transformations of φ and ℓ, and the scale ambiguity of φ are the only sources of non-identifiability [19]. The power transformation of φ and ℓ can be illustrated through the following example: Consider

the pairs of generators and stdf ($\varphi_a(x) = \exp(-x^{1/\theta})$ and $\ell_a(\mathbf{x}) = \|\mathbf{x}\|_1$) and ($\varphi_b(x) = \exp(-x)$ and $\ell_b(\mathbf{x}) = \|\mathbf{x}\|_\theta = (x_1^\theta + \cdots + x^\theta)^{1/\theta}$). Both (φ_a, ℓ_a) and (φ_b, ℓ_b) lead to the same Archimax copula. The scale ambiguity of φ comes from the fact that $\varphi(cx), c > 0$ leads to the same Archimax copula. We note that these sources of non-identifiability are non-issues in our methods. For the power transformation, there is no ambiguity of φ and ℓ since the class of $1 - \varphi(1/\cdot)$ where $\varphi(\cdot)$ is calculated as the Williamson d-transform of R with a finite support is regularly varying with index -1 [7, 19]. In addition, we include a regularization term such that $\mathbb{E}_R[R] = 1$. The algorithms may be iterated as needed with suggested convergence criteria such as CvM statistic between successively estimated copulas.

Given samples of the radial and simplex components from the learnt generative networks, the stochastic representation of Archimax copulas (5.2) gives a straightforward method for sampling Archimax copulas [18]. Specifically, we sample $\mathbf{S}$ and R, then multiply and normalize by φ; we summarize the sampling method given learned generative networks g_r and $g_{\mathbf{w}}$ describing φ_θ and ℓ_θ in Algorithm 9 here:

Algorithm 9 Sample Archimax copulas

input learned g_r, $g_{\mathbf{w}}$
sample $r, r_1, \cdots, r_{n_r}$ from g_r.
sample $\mathbf{s} = (s_1, \cdots, s_d)$ from $g_{\mathbf{w}}$ with Algorithm 6.
compute $\mathbf{u} = (\varphi_\theta(rs_1), \cdots, \varphi_\theta(rs_d))$ from (5.25).
return a sample $\mathbf{u} = (u_1, \cdots, u_d)$.

5.7 Experiments

We conduct a number of empirical studies to understand the modeling and sampling capabilities of the proposed algorithms. We compare the proposed method to existing copula based and deep learning based methods for density estimation. Related to the main focus of understanding the dependencies between the variables in the bulk and the tail, we also perform experiments where we wish to extrapolate to the tails. In

addition, we give an experiment on outlier detection. Experiments were conducted using `PyTorch`, on a 2.7 GHz Intel Core i7 with 16 GB of RAM, with code available at `https://github.com/yutingng/gen-AX`.

5.7.1 Learning the Archimedean Generator and STDF

Our first set of experiments involve estimating the Archimedean generator φ_θ given the true stdf ℓ. The generative network g_r is a multilayer perceptron with layers: (Linear$(1, d_h)$, BatchNorm(d_h), ReLU(), Linear(d_h, d_h), Batch Norm(d_h), ReLU(), Linear$(d_h, 1)$, Exp()), where d_h is the number of nodes in the hidden layers. The number of layers and number of hidden nodes may be modified as needed. Unlike in $g_{\mathbf{w}}$, the use of batch normalization is not essential in g_r.

We empirically select the support size of $\mathcal{Z}$, where $Z = \ell(\mathbf{S})$, for estimating the Archimedean generator. For $n_r = 100$ and $n_z \in \{20, 30, \cdots, 100\}$, we plot the estimates of λ in Figure 5.2, where λ is a map used for evaluating the goodness-of-fit of the Archimedean generator, described in Section 2.5.3. The ground truth is an Archimax copula composed of a Clayton generator and a negative scaled extremal Dirichlet (NSD) stdf. The Clayton generator is popular for use with Archimax copulas as its lower tail dependence breaks the lower tail independence of max-stable extreme-value copulas [19]. It is given as $\varphi(x) = (1 + x)^{-1/\theta}$ and we used the parameter setting $\theta_{\tau=0.2} = 0.5$. The NSD stdf is a flexible class of stdfs encompassing many parametric stdfs, including the logistic, asymmetric logistic, negative logistic, and extremal Dirichlet models [7]. The NSD stdf is specified by $\ell(\mathbf{x}) = \frac{\Gamma(\alpha_1+\cdots+\alpha_d-\rho)}{\Gamma(\alpha_1+\cdots+\alpha_d)}\mathbb{E}_{\mathbf{D}}\left[\max_{j=1,\cdots,d}\left(\frac{x_j D_j^{-\rho}\Gamma(\alpha_j)}{\Gamma(\alpha_j-\rho)}\right)\right]$ where $\mathbf{D} = (D_1, ..., D_d)$ is distributed as a Dirichlet$(\alpha_1, ..., \alpha_d)$ with $\alpha_1, ..., \alpha_d > 0$, $\rho \in (0, \min(\alpha_1, \cdots, \alpha_d))$, and we used the parameter settings $\alpha = (1, 1, 1, 1, 2, 2, 2, 3, 3, 4)$, $\rho = 0.69$. We consider data with dimension $d = 10$ and sample size $n = 1000$. Results from different runs are

plotted in blue. The ground truth is plotted in black and approximate confidence bands around the ground truth is plotted in dotted black. In addition, the mean squared error (MSE) in fitting K and λ and the time taken are given in Table 5.1 where the standard deviation is given in parenthesis. For a visual comparison, we plotted the tabular results in Figure 5.3.

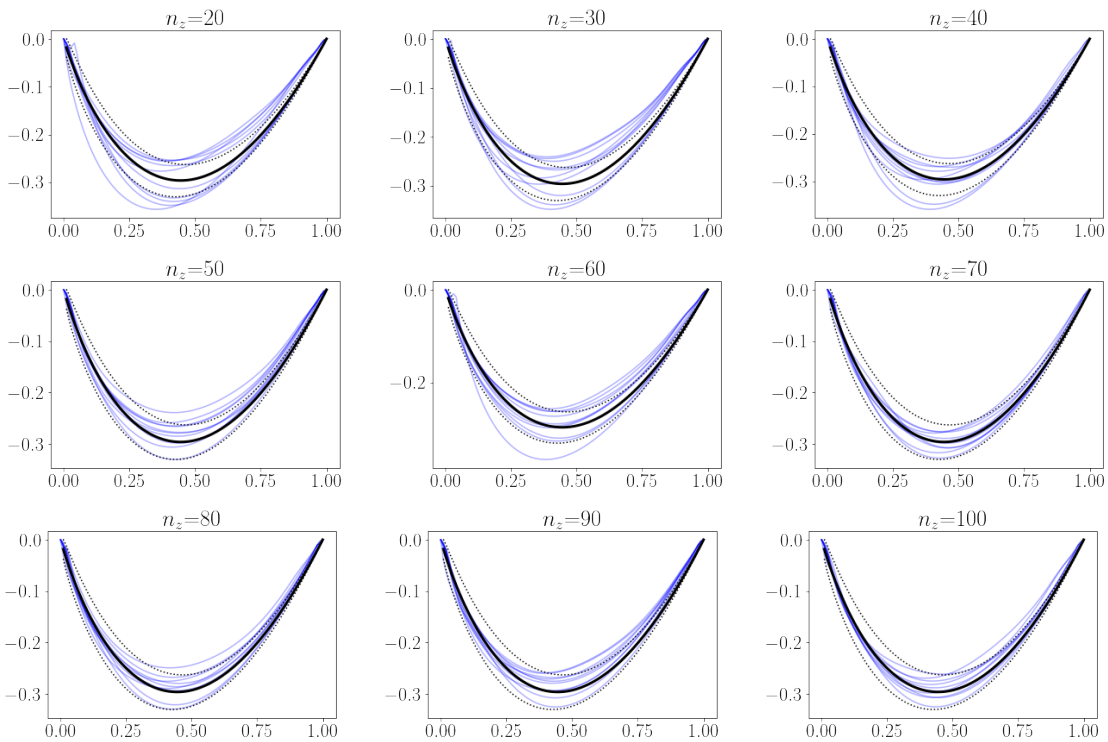

FIGURE 5.2: Estimates of λ with $n_r = 100$ and $n_z \in \{20, \cdots, 100\}$. Ground truth in black, approximate confidence bands in dotted black. Estimates from different runs in blue.

Table 5.1: MSE of K, λ, and time taken, with $n_r = 100$ and $n_z \in \{20, \cdots, 100\}$

$n_r = 100, n_z$	10	20	30	40	50	60	70	80	90	100
MSE K $\times 10^{-3}$	5.99(11.08)	1.84(3.60)	0.55(0.91)	0.34(0.56)	0.21(0.57)	0.46(1.02)	0.60(1.06)	0.23(0.39)	**0.12(0.31)**	0.16(0.31)
MSE λ $\times 10^{-3}$	1.24(1.27)	1.10(0.75)	1.04(0.56)	0.57(0.62)	0.49(0.49)	0.74(0.72)	**0.18(0.13)**	0.32(0.32)	0.39(0.27)	0.23(0.21)
TIME (SEC)	8.59(1.78)	21.30(7.10)	40.06(11.11)	43.46(8.23)	62.46(18.45)	86.08(21.91)	100.62(29.75)	217.68(71.45)	190.10(56.29)	239.04(96.97)

From the results, for $n_r = 100$, it would be appropriate to use $n_z \in \{70, \cdots, 100\}$, with a tradeoff between accuracy and computation time. In addition, to speed up

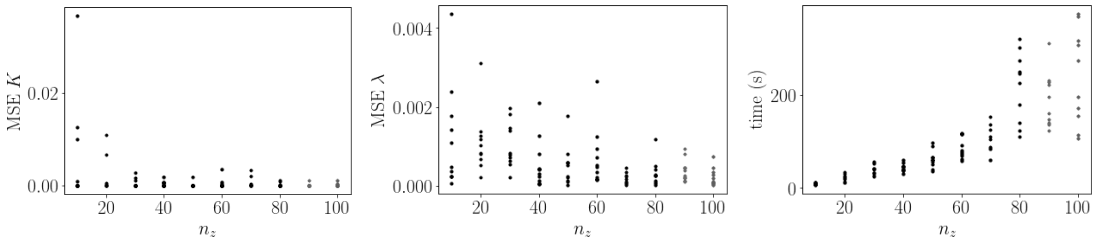

FIGURE 5.3: MSE of K, λ, and time taken, with $n_r = 100$ and $n_z \in \{20, \cdots, 100\}$.

training, we initially resample $\mathbf{Z}$ only once every k mini-batch iterations, decreasing k until $k = 1$ as we approach convergence.

We then evaluated our proposed method on the Clayton (C), Frank (F), Joe (J) and Gumbel (G) generators, representing different radial envelopes, for Kendall's tau of $\tau = \{0.2, 0.5\}$, representing different associations, as introduced in Table 2.2. We used the same NSD stdf and consider data with the same dimension $d = 10$ and sample size $n = 1000$. To the best of our knowledge, our proposed method is the first method for non-parametric inference of flexible Archimedean generators in Archimax copulas. As such, there are no baselines for comparison. Instead, we again compute the results in terms of the map λ, as described in Section 2.5.3, due to its scale invariant property and known asymptotic variance, which was described as a useful metric for evaluating the fit of the Archimedean generator [40, 42]. The next best method is comparing to a Clayton generator estimated by pairwise Kendall's tau [19]. We plot estimates of λ in Figures 5.4 and 5.5 where all estimates were within the asymptotic variance of $\lambda(w)$, $w \in (0, 1)$. We also give the results in terms of MSE to λ in Table 5.2.

While our method infers an arbitrary Archimedean generator and takes the joint dependence across covariates into account, the method in [19] infers a Clayton generator from pairwise Kendall taus. Thus the performance gap between our method and the method in [19] is expected to increase as the generator differs from the Clayton generator and as the observations become less symmetric.

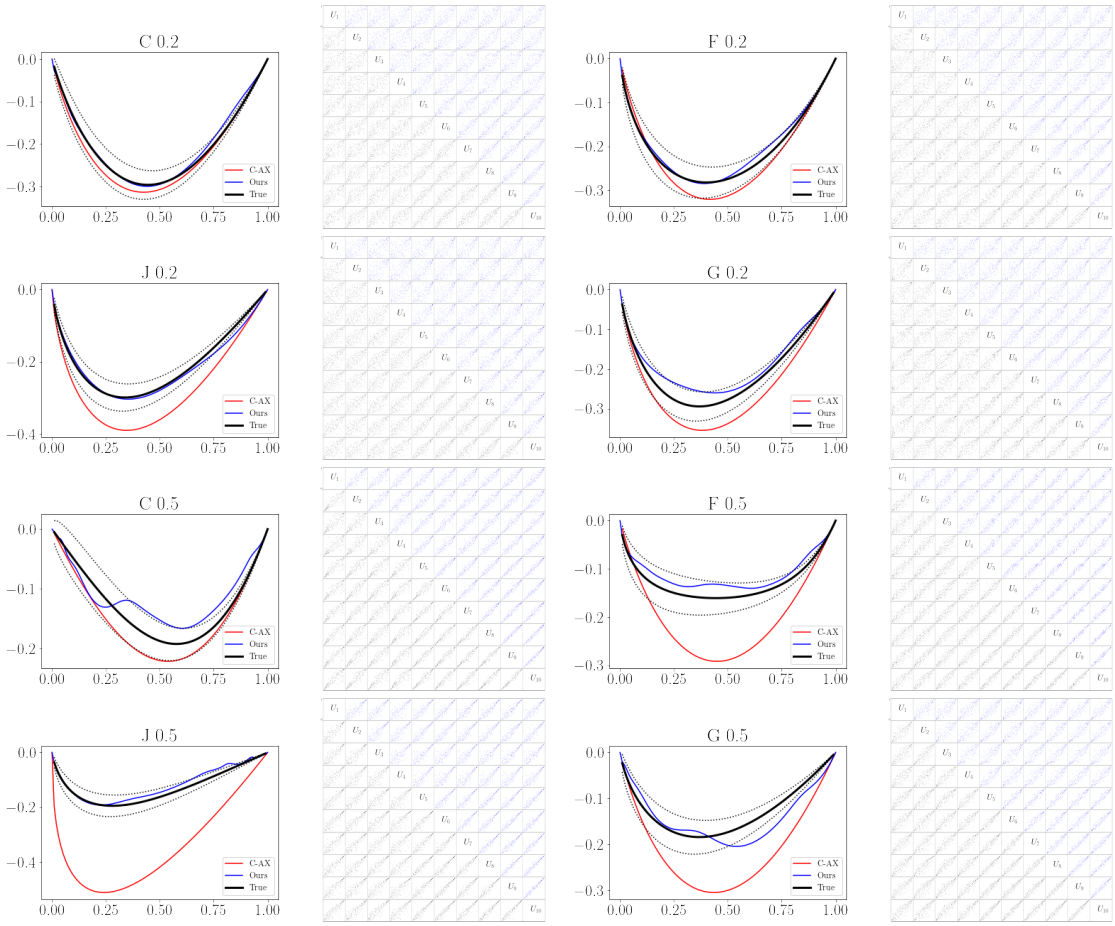

FIGURE 5.4: Estimates of λ given 200 samples of $\mathbf{S}$ from true stdf. Ground truth in black, approximate confidence bands in dotted black, the method from [19] in red, our method in blue. On the right of each plot of λ is a plot of samples from the copula displayed as mirrors on the diagonal, our method above in blue and ground truth below in black.

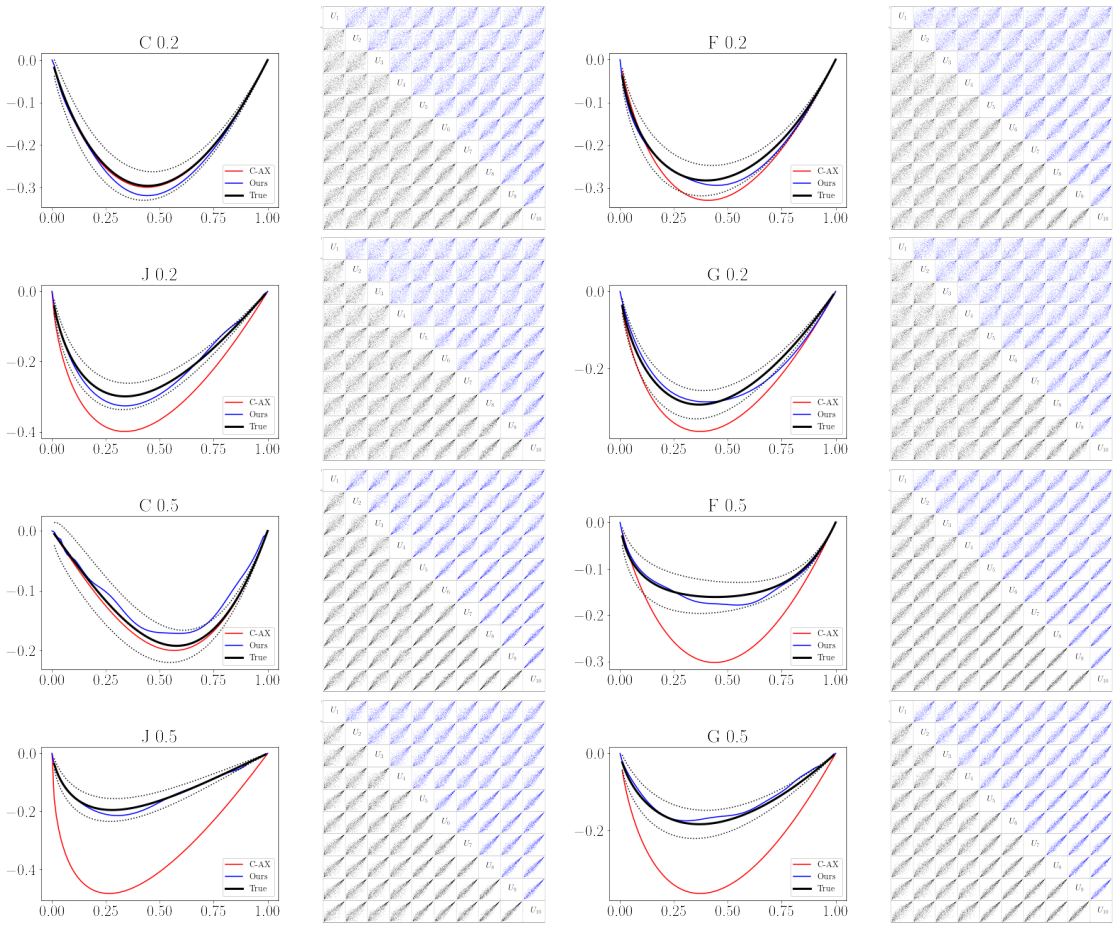

FIGURE 5.5: Estimates of λ given 1000 samples of $\mathbf{S}$ from true stdf. Ground truth in black, approximate confidence bands in dotted black, the method from [19] in red, our method in blue. On the right of each plot of λ is a plot of samples from the copula displayed as mirrors on the diagonal, our method above in blue and ground truth below in black.

Increasing the number of observations improved accuracy, hinting at consistency.

Table 5.2: Inference of φ given true ℓ and inference of ℓ given true φ

	C 0.2	C 0.5	F 0.2	F 0.5	J 0.2	J 0.5	G 0.2	G 0.5
MSE $\times 10^{-3}$ [19]	**0.01**	**0.04**	0.9	9	1	9	0.7	8
MSE $\times 10^{-3}$ (Ours)	0.2	0.2	**0.1**	**0.1**	**0.3**	**0.1**	**0.2**	**0.1**
IRAE ± 0.01 P [19]	0.16	1.00	0.05	0.06	0.06	0.07	0.08	0.17
IRAE ± 0.01 CFG [19]	0.05	0.11	0.04	0.04	0.05	1.00	0.06	0.15
IRAE ± 0.01 (Ours)	0.06	0.12	0.04	0.05	0.06	0.07	0.08	0.15

We then consider the reverse scenario where we wish to estimate the stdf ℓ_θ given the true Archimedean generator φ. For the same experiment settings as above, we give the results in terms of integrated relative absolute error (IRAE), as described in Section 2.5.4, in Table 5.2. The IRAE is computed using Monte Carlo integration with 10,000 samples $\mathbf{x}$ drawn uniformly at random from the simplex Δ_{d-1}.

The generative network $g_{\mathbf{w}}$ is a multilayer perceptron with layers: (Linear(d, d_h), BatchNorm(d_h), ReLU(), Linear(d_h, d_h), Batch Norm(d_h), ReLU(), Linear(d_h, d), Softmax()), where d is the dimension of the observations and d_h is the number of nodes in the hidden layers. The number of layers and number of hidden nodes may be modified as needed. Batch normalization, i.e. BatchNorm$(\cdot)$, greatly helps in preventing samples from being static during training. In addition, to reduce computational complexity, we use mini-batch gradient descent and a smaller number of samples in the empirical expectations of φ_θ and ℓ_θ during training.

Though our estimator is based on the Pickands estimator, it performs better than the Pickands estimator. This may be due to our estimator representing the class of valid stdfs, a phenomenon noted in [37] where projecting to the class of valid stdfs reduced estimation error. Furthermore, since our estimator is already a valid stdf, we do not need an additional endpoint correction step. On the other hand, the Pickands and CFG estimators may not be valid stdfs [50]. Moreover, our direct method of training with the generative network generating the class of valid stdfs may perform

better than the alternative method that first estimates $\hat{\ell}$ then train the generative network to match $\hat{\ell}$. A future direction may be to improve the robustness of our estimator with the CFG modification, where in [19], the modified CFG estimator performs better than the modified Pickands estimator for Archimax copulas.

We also provide the IRAE and time taken versus number of minibatch iterations in Figure 5.6. All timings are with a 2.7 GHz Intel Core i7, 16GB 2133MHz LPDDR3. The plots suggest that our algorithm converges and is not computationally intensive.

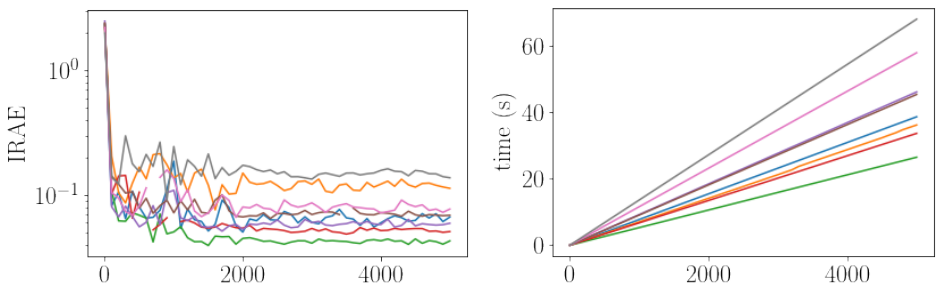

FIGURE 5.6: IRAE of fitting ℓ given true φ and time taken versus number of minibatch iterations. Each line is for a different copula setting in Table 5.2.

5.7.2 Application to Modeling Nutrient Intake

The USDA studied the nutrient intake of women [121]. There are $n = 1459$ observations of dimension $d = 17$, corresponding to the variables: Energy, Protein, Vitamin A (IU), Vitamin A (RE), Vitamin E, Vitamin C, Thiamin, Riboflavin, Niacin, Vitamin B6, Folate, Vitamin B12, Calcium, Phosphorus, Magnesium, Iron, Zinc. We can model and understand the dependencies between the intake of different nutrients with an Archimax copula. This dataset was previously studied in [95] where using a Clayton-NSD Archimax copula improved fit over a Clayton Archimedean copula. However, in [95], the experiment was limited to only $n = 737$ and $d = 3$, corresponding to Calcium, Iron and Protein.

We compared our method to copula based models and deep network based models

using the CvM statistic, defined in Section 2.5.2, where the integral is computed using Monte Carlo integration with 10,000 samples of $\mathbf{u}$ drawn uniformly at random from $[0,1]^{d-1}$.

For the copula models, we compare to the Gaussian (GC), R-vine (RV), C-vine (CV), D-vine (DV), Archimedean (AC*), hierarchical Archimedean (HAC) [48], extreme-value (EV†) and the state-of-the-art Clayton-Archimax (C-AX†) [19] copulas, where for copulas marked with $*$, we use φ described in Section 5.4 and for copulas marked with $\dagger$ we use the sampling methods for $\mathbf{S}$ described in Sections 5.2 and 5.3. The estimation and sampling of the other copula based models were done using the `Copulas` library [26] in Python and the `HACopula` toolbox [47] in MATLAB.

For the deep generative models, we compare to the Wasserstein GAN with gradient penalty (WGAN) [3], masked autoregressive flow (MAF) [103], and variational autoencoders (VAE) [73]. The architectures of the deep networks were:

- WGAN: 3 layers for generator, 3 layers for discriminator, hidden size 128.
- MAF: 2 flows, hidden size 128 for each flow.
- VAE: 3 layers for encoder, 2 layers for decoder, hidden size 128, latent size 16.

We denote our model with the abbreviation (Gen-AX*†) and used the architecture: 3 layers with hidden size of 30 for $g_{\mathbf{w}}$ and 3 layers with hidden size of 10 for g_r. For all deep models, the implementation was with `PyTorch` and the Adam optimizer was used with learning rate 1e-3.

With regard to our model, we first initialize ℓ_θ with Algorithm 5 on block-maximas. Using the test for extreme-value dependence [75], we chose the block size $n/k = 5$. The plots for block sizes $n/k \in \{1, 2, 5, 10\}$ and exponents $r \in \{2, 3, \cdots, 10\}$ are given in Figure 5.7. We compared our initial estimate ℓ_θ to the state-of-the-art CFG estimator applied on the block maximas $\hat{\ell}$ [12]. The plot of IRAE against mini-

batch iterations show convergence in 2000 mini-batch iterations, with a duration of 20s, and an IRAE of 0.08. We then learn φ_θ with Algorithm 7 on the full dataset, with samples of $\mathbf{S}$ from Algorithm 6. The plot of MSE in fitting the empirical Kendall distribution show convergence in 2000 mini-batch iterations, with a duration of 25s, and an MSE of 0.0008. We note that the initialization scheme seems to be performing well since the learnt φ_θ focused on modifying only the lower tail. We then update ℓ_θ with Algorithm 5 on the full dataset, given φ_θ. The NLL of transformed observations ξ went from 1.67 in the initialization to -1.12 with the use of φ_θ. The IRAE to the state-of-the-art modified CFG estimator on the full dataset was 0.078.

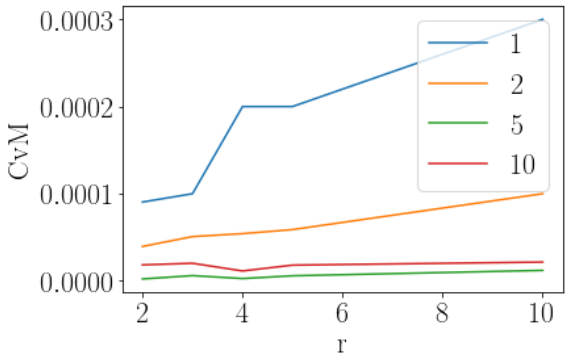

FIGURE 5.7: Selection of block size using test for max-stable property, $n/k = 5$ [75].

The results comparing our model to the copula based models and deep network based models are presented in Table 5.3.

Table 5.3: Goodness-of-fit to $17d$ nutrient intake data and $100d$ C-NSD copula

	COPULAS								DEEP NETS			OURS
	GC	RV	CV	DV	AC *	HAC	EV †	C-AX †	GAN	MAF	VAE	Gen-AX *†
Nutrient CvM $\times 10^{-3}$	0.081	0.3	0.2	0.6	0.030	0.2	0.4	0.6	0.033	0.036	0.053	**0.026**
C-NSD CvM $\times 10^{-5}$	-	-	-	-	-	-	-	-	16	**3**	5	**3**

The proposed Gen-AX*† has the lowest CvM statistic among the competing methods, suggesting that the data may have Archimax dependence. The C-AX† did not

perform well possibly due to the difficulty in scaling the single parameter Clayton generator to a higher dimension. The AC* possibly benefited from the use of our proposed φ. We additionally provide plots of generated samples versus true samples in Figure 5.9 (a-l). The generated samples are plotted in blue above the diagonal while the true samples are plotted in black below the diagonal.

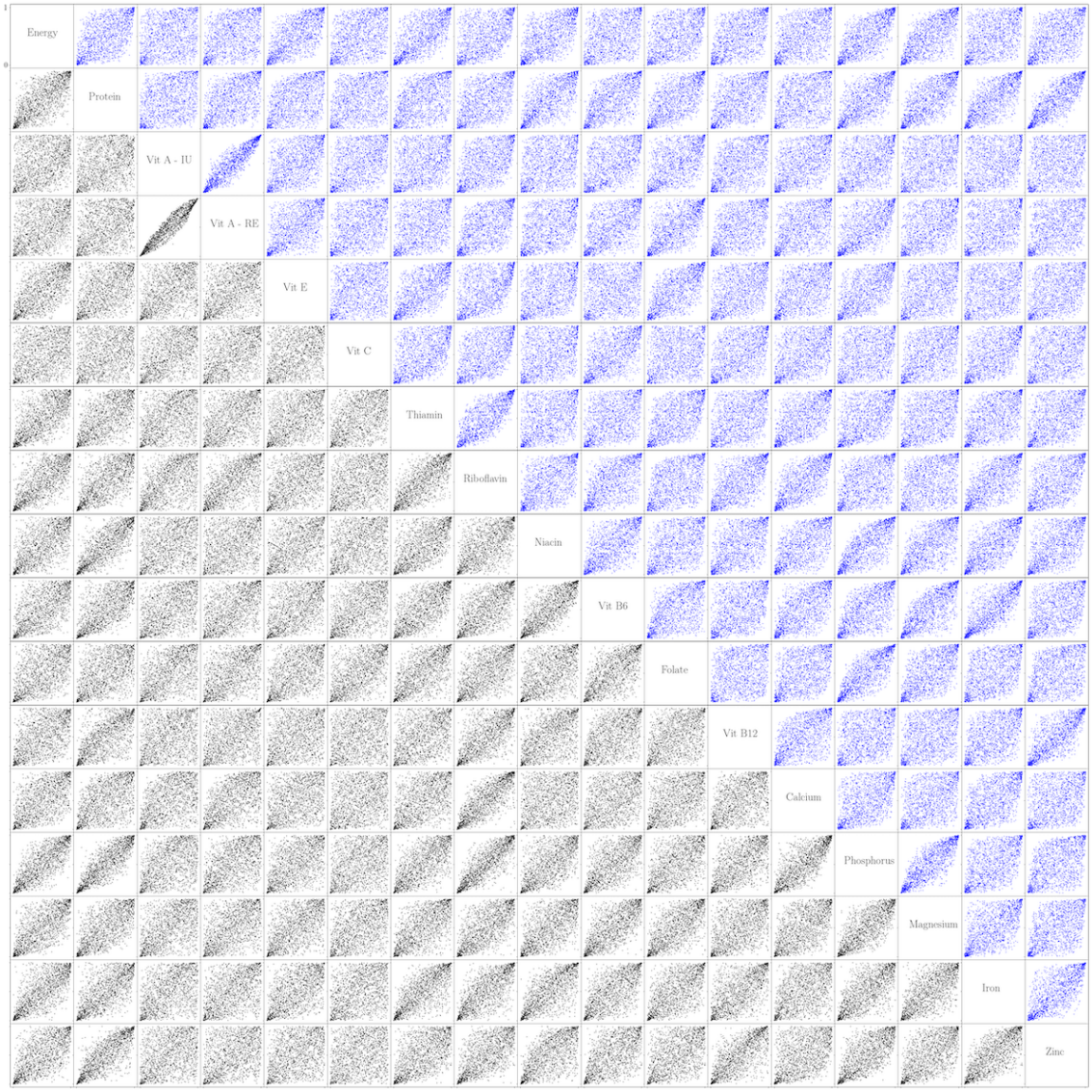

FIGURE 5.8: Samples from generative Archimax copula in blue inferred and sampled with our methods.

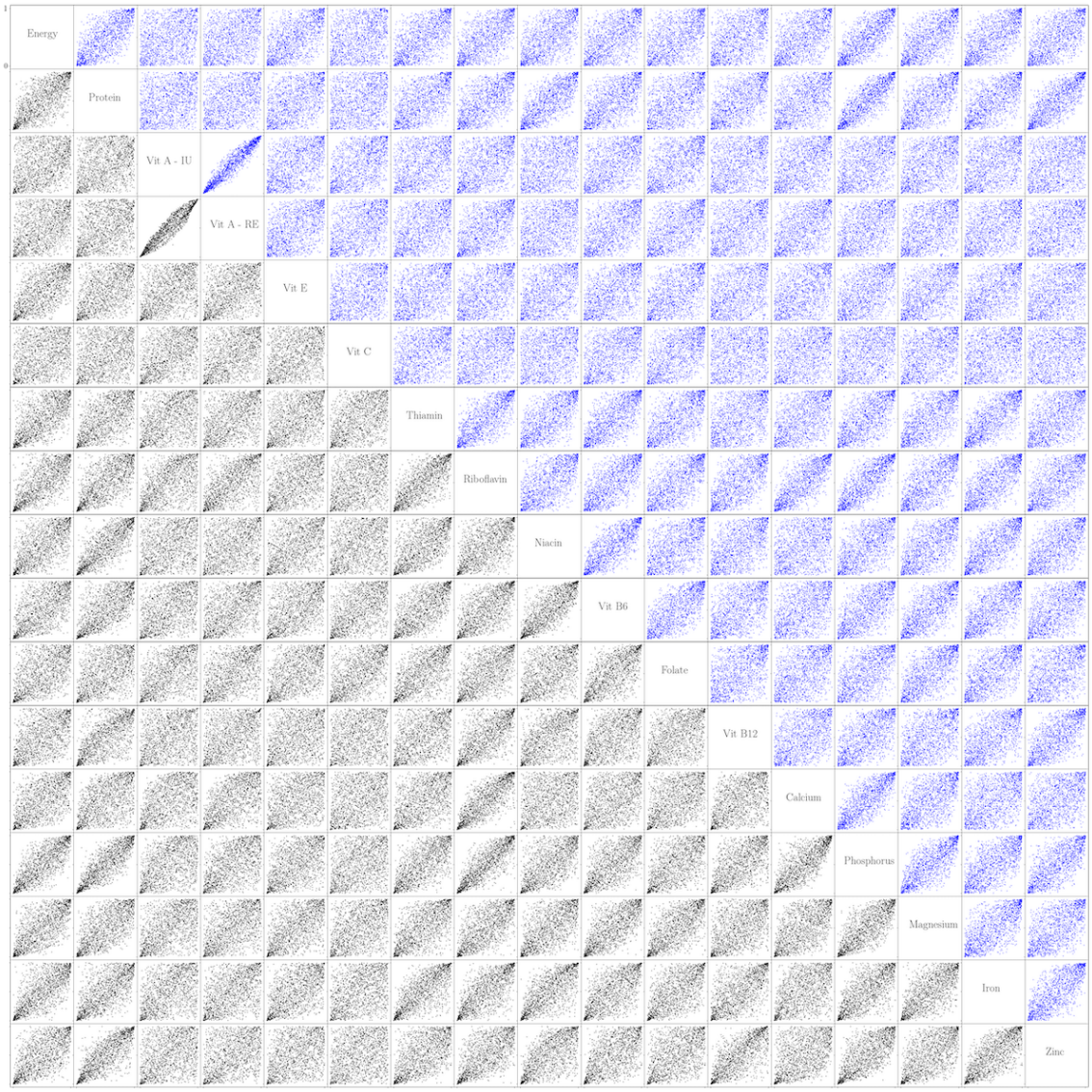

FIGURE 5.9: Samples from Gaussian copula in blue.

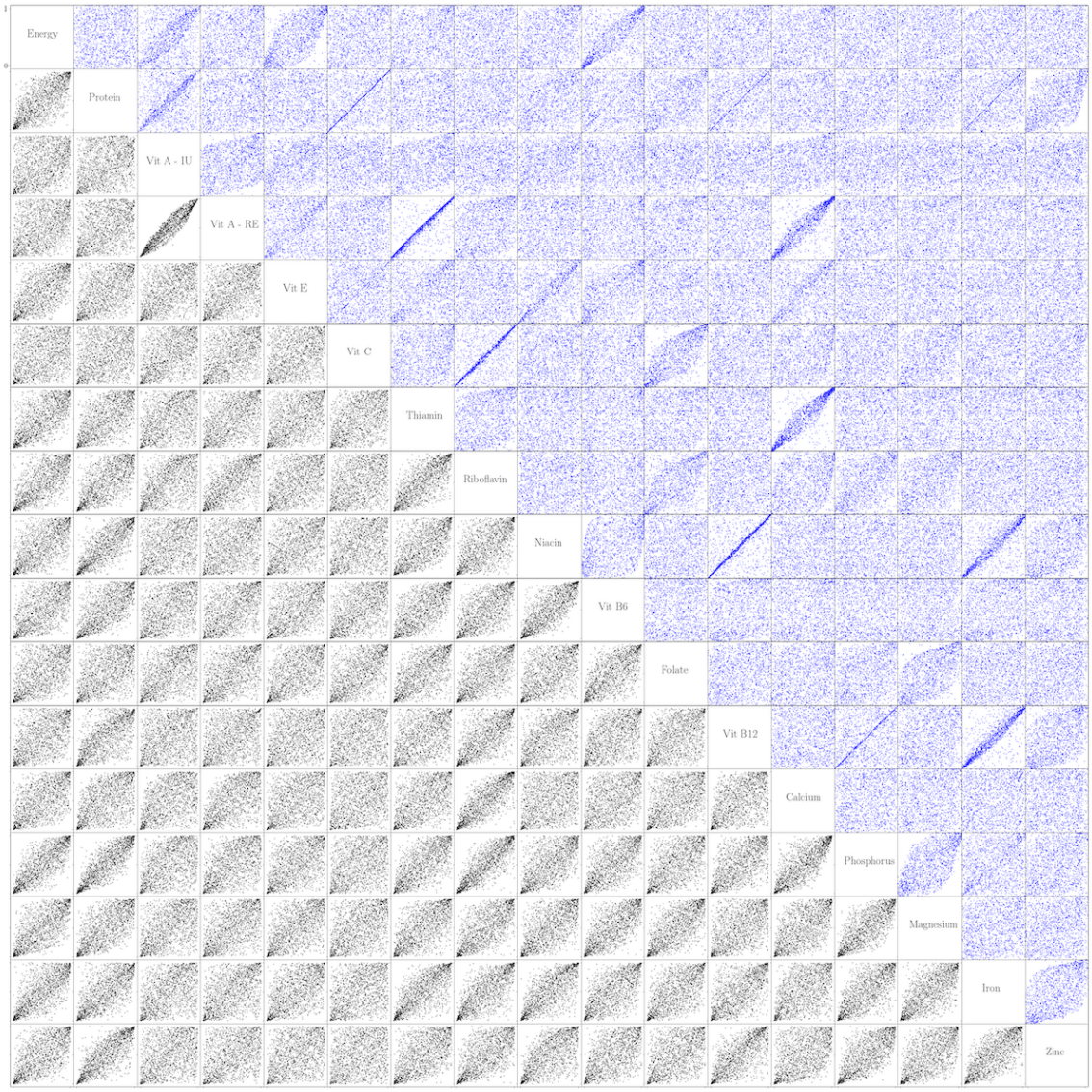

FIGURE 5.10: Samples from R-vine copula in blue.

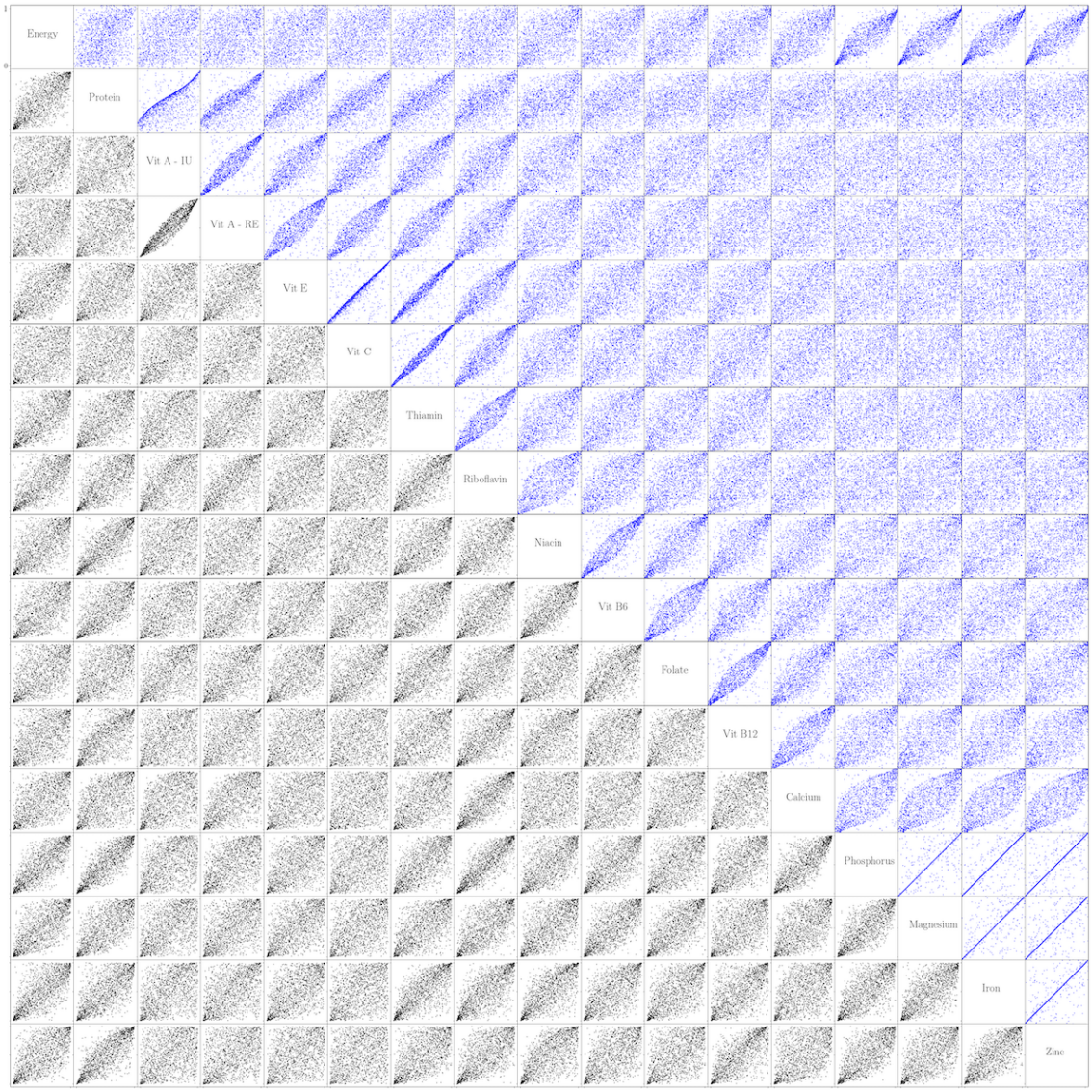

FIGURE 5.11: Samples from C-vine copula in blue.

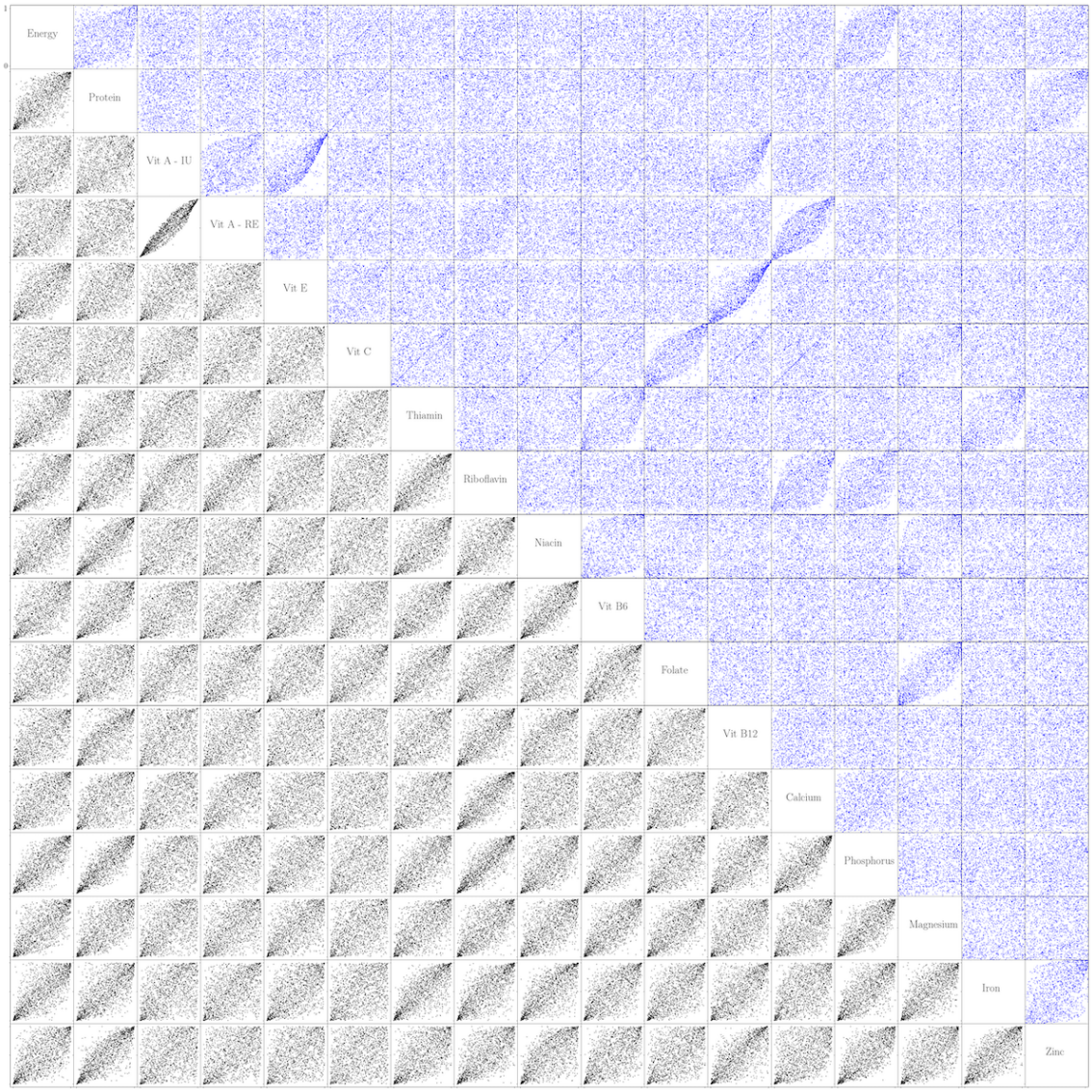

FIGURE 5.12: Samples from D-vine copula in blue.

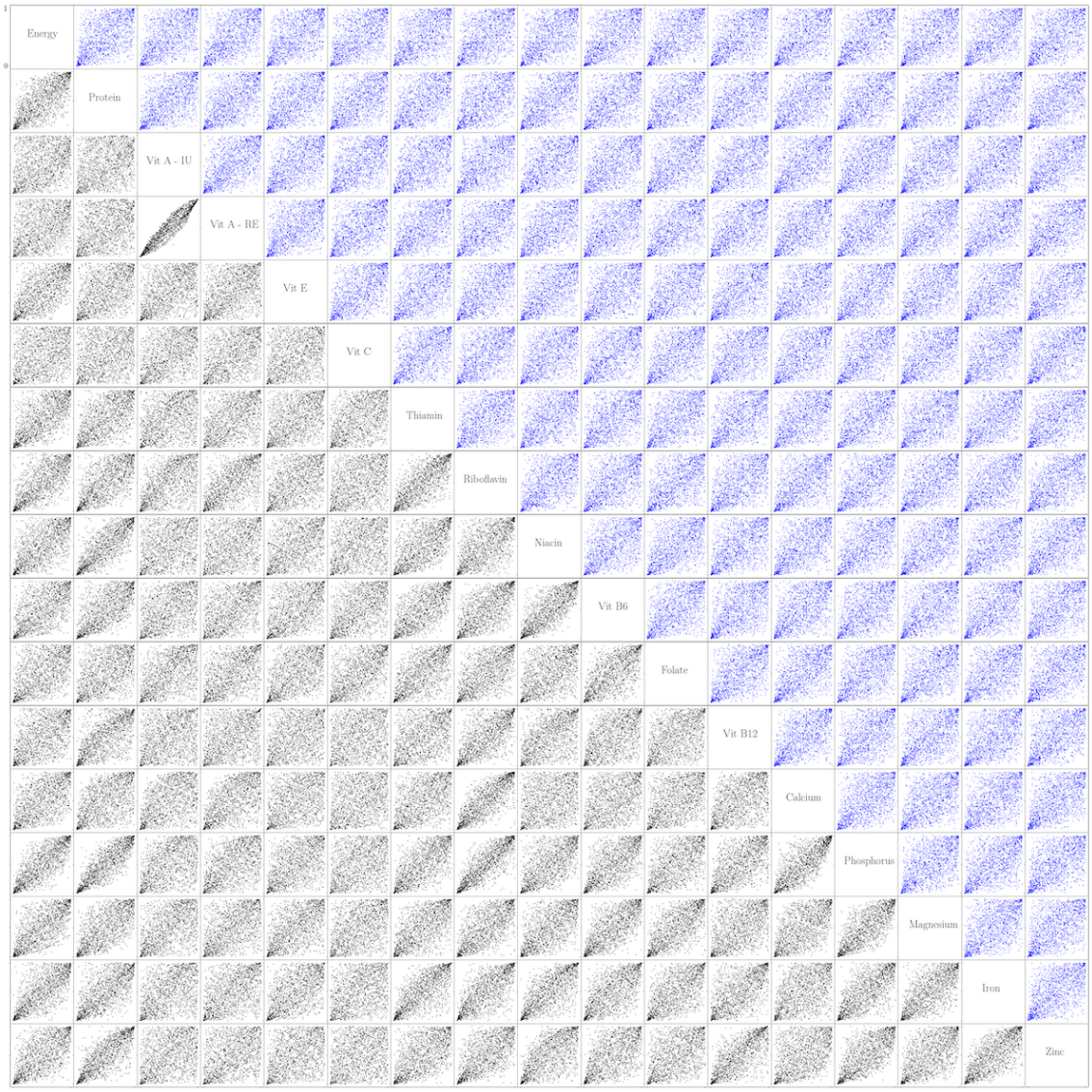

FIGURE 5.13: Samples from Archimedean copula in blue inferred with Algorithm 7.

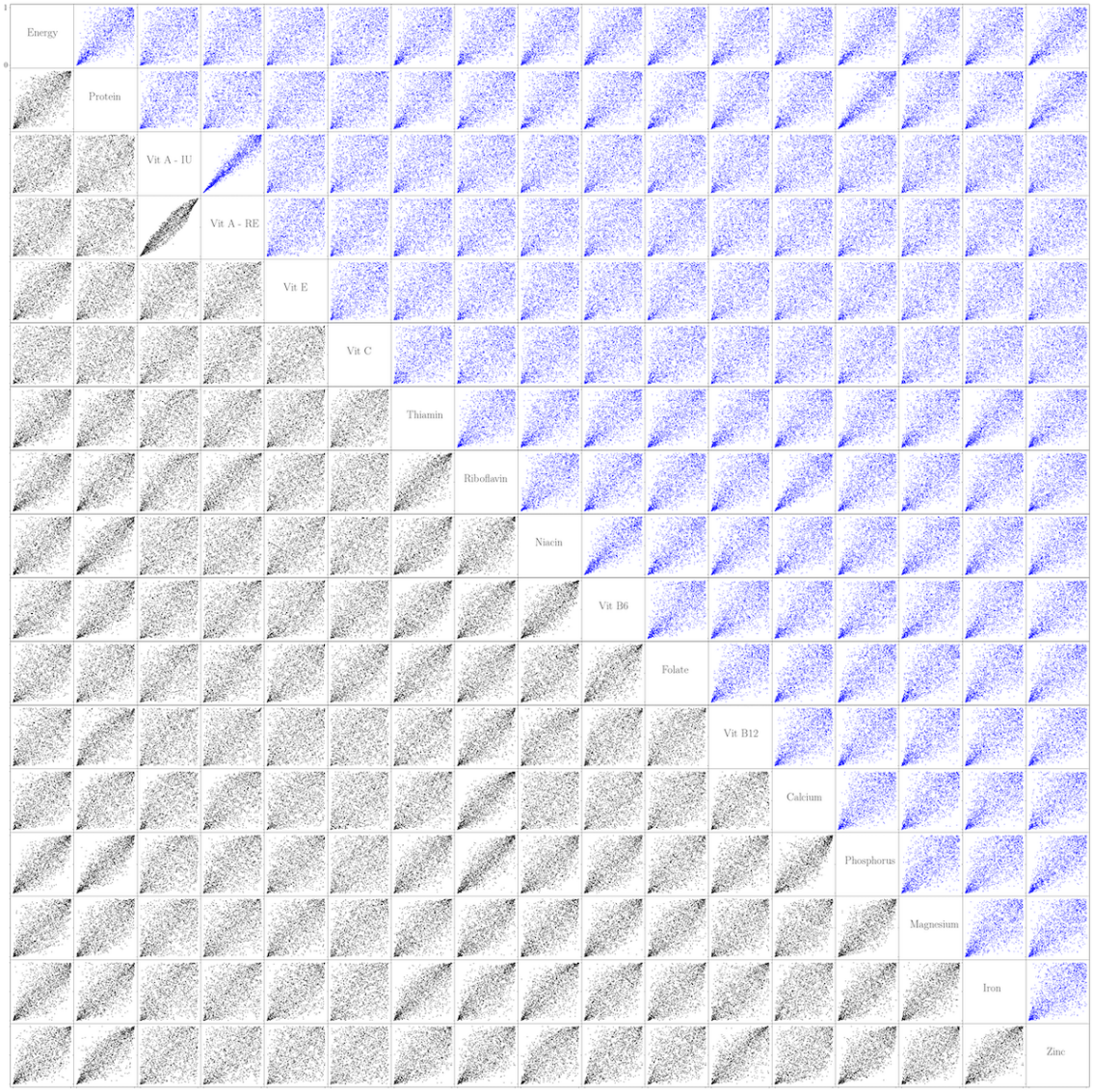

FIGURE 5.14: Samples from hierarchical Archimedean copula in blue [48].

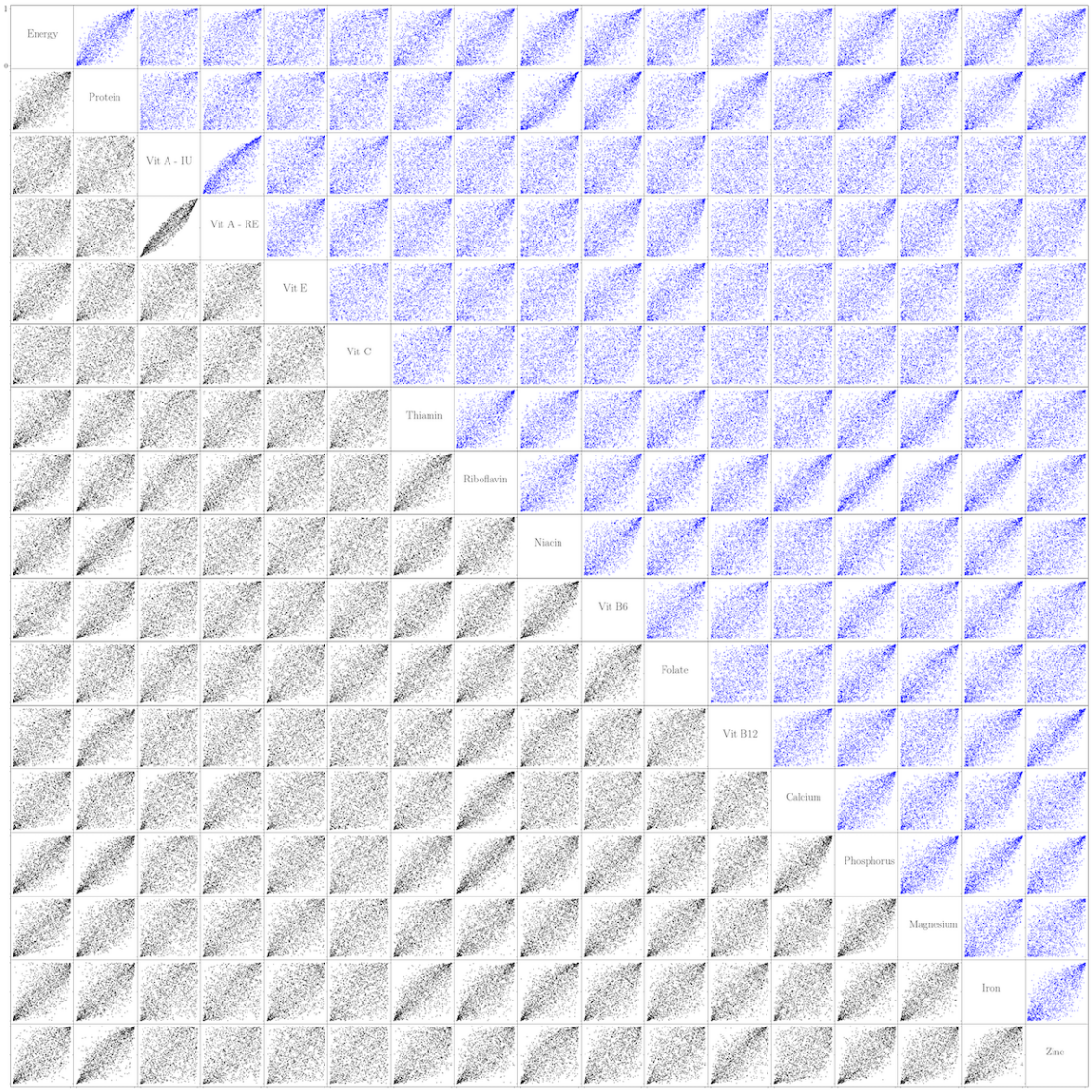

FIGURE 5.15: Samples from extreme-value copula in blue sampled with Algorithms 6 and 9.

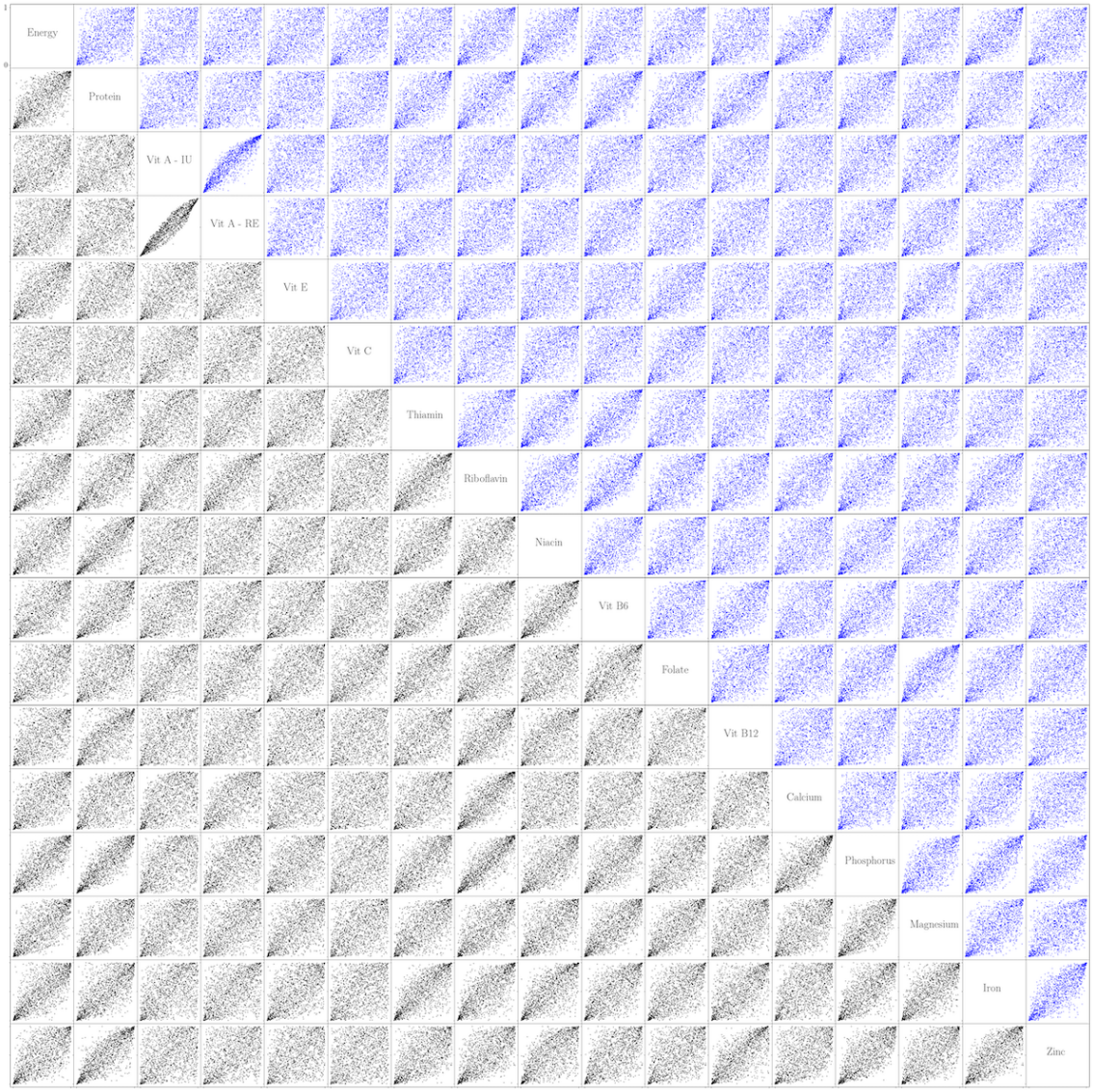

FIGURE 5.16: Samples from Clayton-Archimax copula in blue inferred with the state of the art [19], sampled with Algorithms 6 and 9.

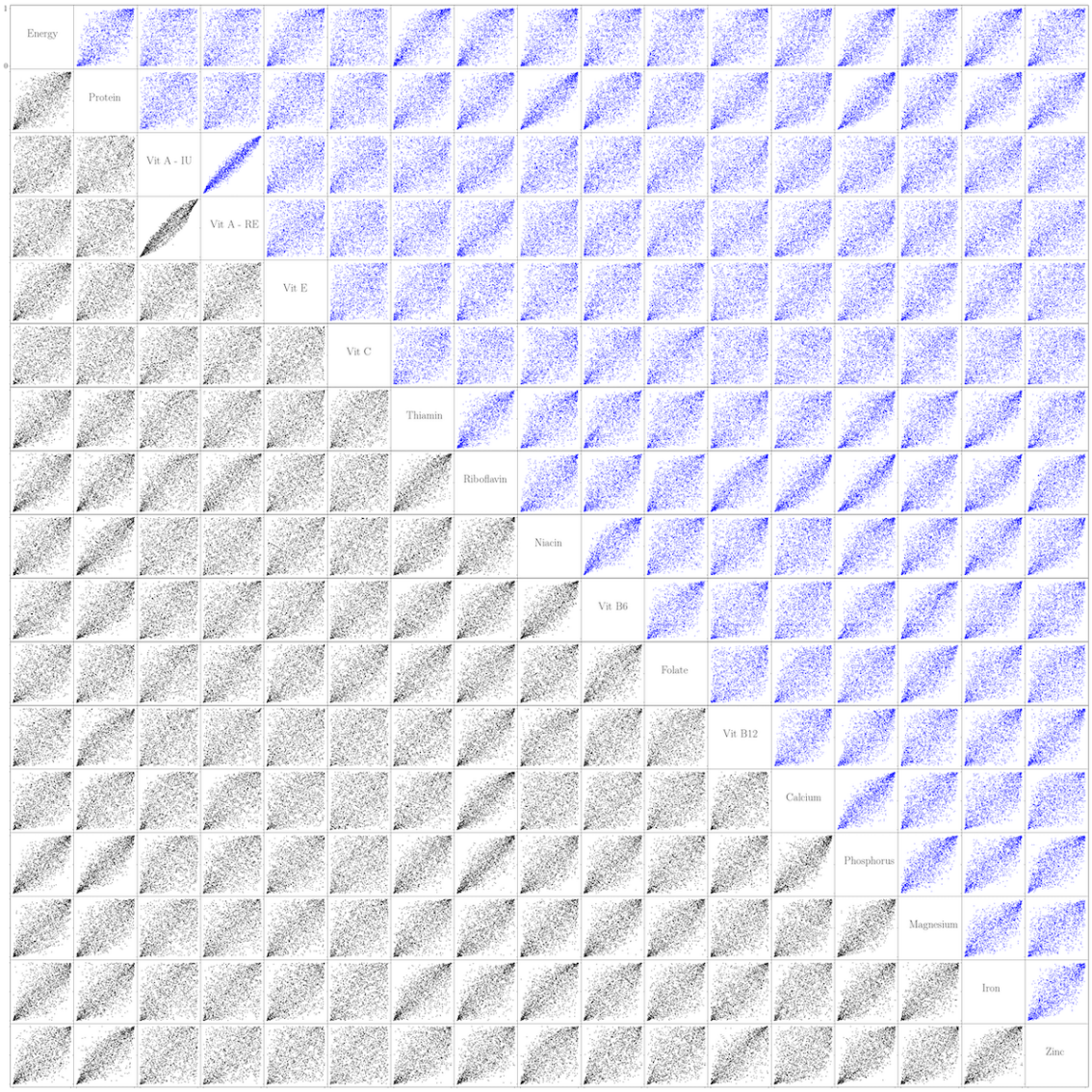

FIGURE 5.17: Samples from Wasserstein GAN with gradient penalty in blue.

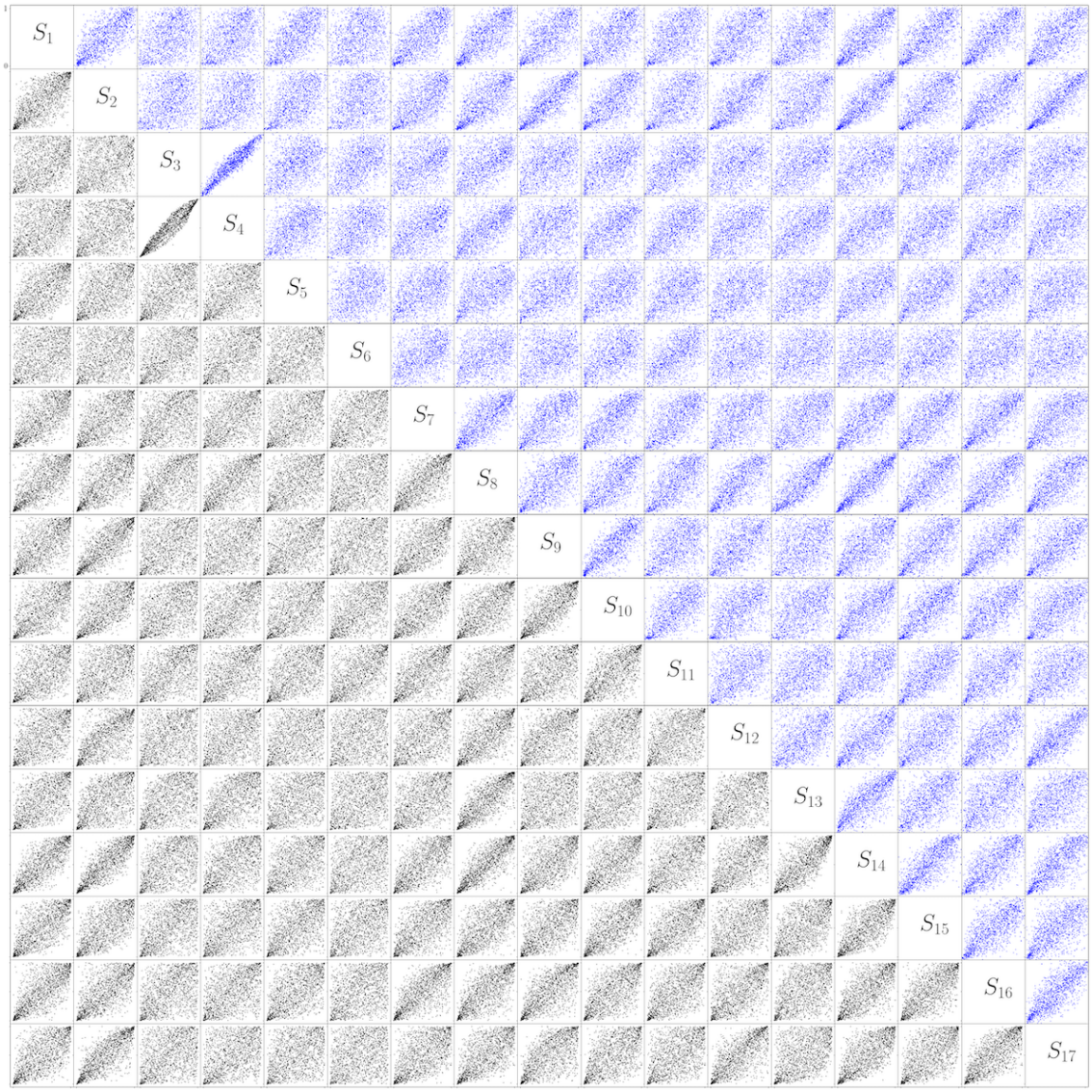

FIGURE 5.18: Samples from masked autoregressive flow in blue.

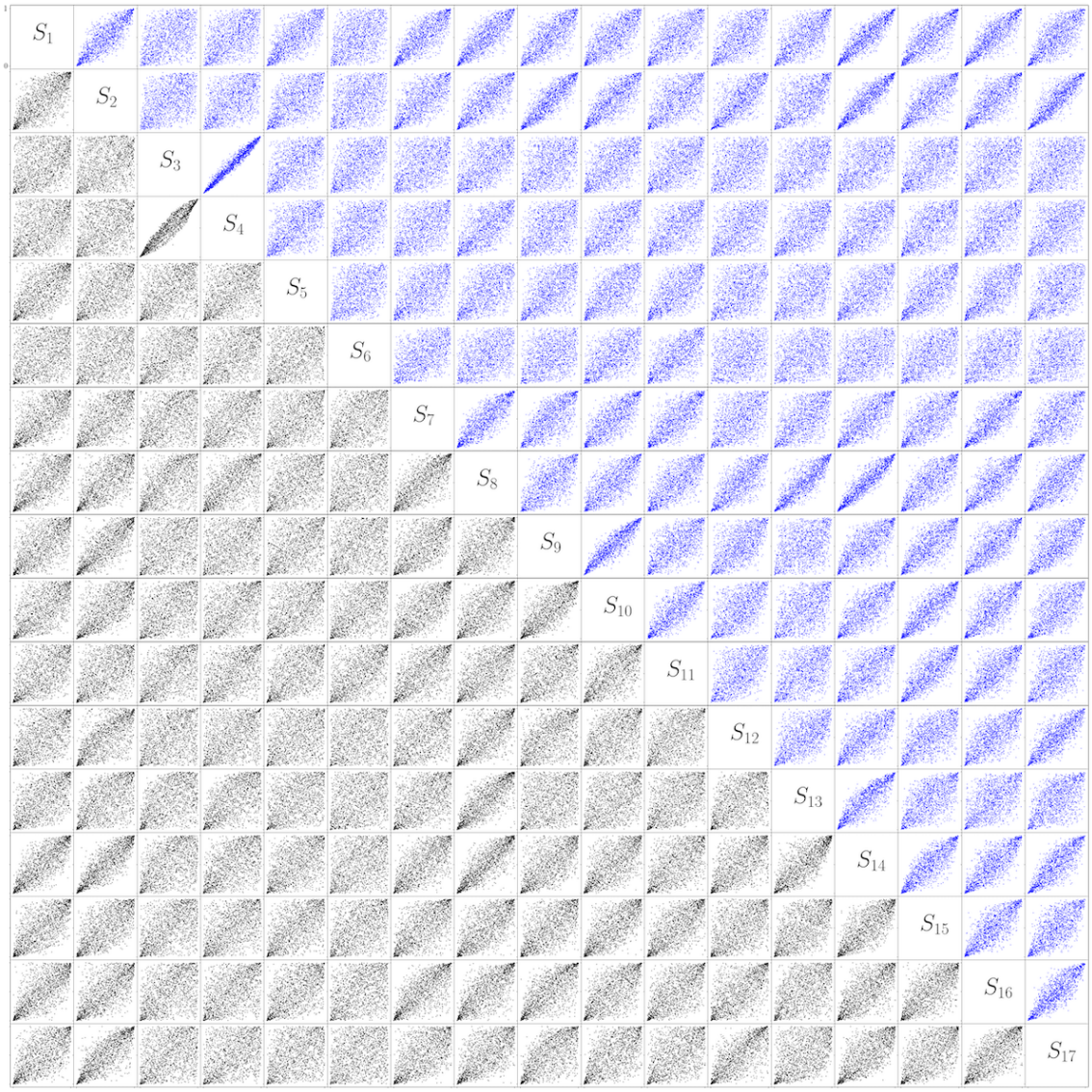

FIGURE 5.19: Samples from variational autoencoder in blue.

5.7.3 Application to Extrapolating to Extreme Rainfall

Archimax copulas were introduced as a tool to study the behaviour of methods that estimate the joint distribution of extreme events [13]. The extreme-value copula that arises in the limit can be understood from the stdf ℓ and the index of regular variation of the Archimedean generator φ. Unlike extreme-value copulas which emerge from the limiting distribution of extreme events, the motivation for the use of Archimax copulas is to model extreme data, where observations are rare, from a mix of moderately less extreme data, where observations are relatively more abundant.

In this experiment, we consider a realistic dataset that models the monthly rainfall in French Britanny as studied in [19]. This data simulates the monthly rainfall for 3 locations (Belle-Ile, Groix, and Lorient) in French Brittany, where the dependence is asymmetric and non-extreme. We follow the Archimax model in [19] and extend the experiment with different Archimedean generators of the same Kendall tau τ. We are interested in testing how well the proposed method can extrapolate to the extremes from non-extreme data. Specifically, we analyze the monthly rainfall data, which did not pass the test of extreme-value dependence [19, 75].

The methods used for comparison are Wasserstein GAN with gradient penalty (WGAN), masked autoregressive flow (MAF), variational autoencoder (VAE) and the state-of-the-art Clayton-Archimax copula (C-AX). All methods were first trained on all observations $n = 240$ with dimension $d = 3$. Many samples were then generated from the trained model to estimate $\hat{\ell}$ from block maximas using the state-of-the-art CFG estimator for extreme-value copulas [12]. The results are presented in Table 5.4, where we also compare to the stdf estimated by [19]. For the generative models, plots of the generated samples and generated samples in the extremes are also given in Figure 5.20.

Table 5.4: Goodness-of-fit to dependence in the extremes

IRAE ± 0.01	GAN	MAF	VAE	C-AX	Ours
C-NSD	0.52	0.42	0.16	0.12	**0.03**
F-NSD	0.10	0.15	**0.04**	**0.04**	**0.04**
J-NSD	**0.03**	0.48	**0.03**	0.08	**0.03**
G-NSD	0.07	0.38	0.08	0.16	**0.04**

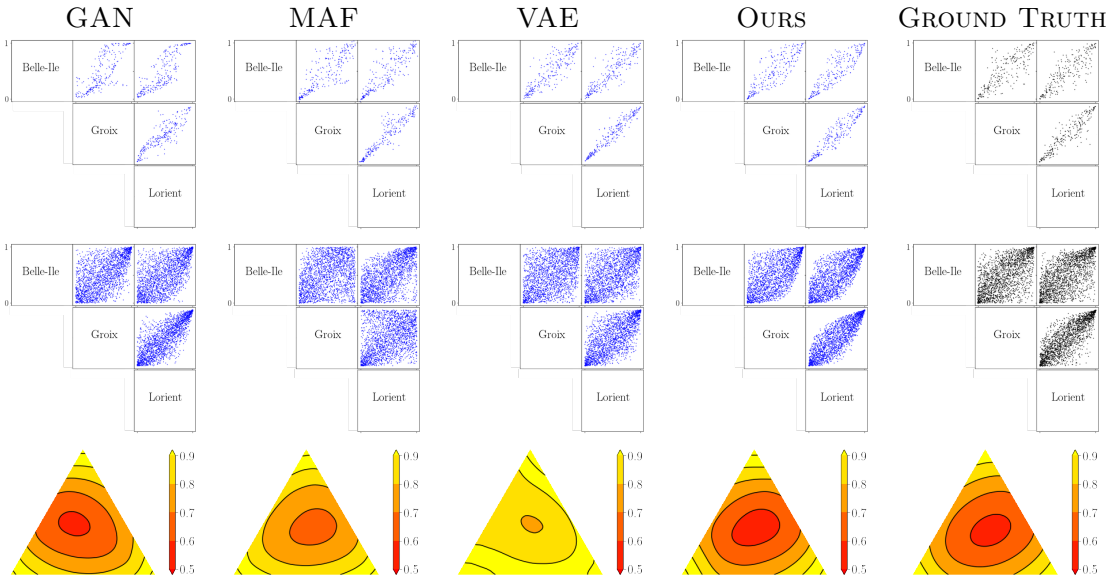

Figure 5.20: Results from extrapolating to dependence in the extremes. Plots of generated samples (top), generated samples in the extremes (middle) and stdf $\ell(\mathbf{w}), \mathbf{w} \in \Delta_{3-1}$ (bottom).

Table 5.4 show our method performs consistently across the different Archimedean generators; always the best and in the case of ties, always among the best.

For extrapolation to extremes, we did not compare to copula based models since classical copulas are generally Gumbel or independent in the extremes and thus not suitable for this application. On the other hand, for the purpose of modeling monthly rainfall without extrapolating to extremes, we compared to a variety of copula based models, including skew-t copulas which is a class of flexible asymmetric copulas [28, 76, 116, 129]. For the Clayton-NSD Archimax copula experiment setting, the CvM statistics for the Archimax, Gaussian, extreme-value, t and skew-t copulas are: **0.0003** (lower is better), 0.0005, 0.0006, 0.0008, 0.0027.

For this scenario, the Archimax and Gaussian copula performed better than the extreme-value, t and skew-t copulas. This may be because the Clayton-NSD copula does not exhibit extreme-value dependence, i.e. it fails the test of extreme-value dependence [75]. The skew-t copula might have performed not as well as the t copula due to over-parameterization. In addition, maximum likelihood estimation for the skew-t copula was extremely time consuming even for three dimensions and thus intractable for higher dimensions.

5.7.4 Application to Out-of-Distribution Detection

Using the same dataset as above, we added outliers uniformly at random on the unit cube. The number of inlier observations was $n_{in} = 225$, the number of outlier observations was $n_{out} = 25$, corresponding to 10% data contamination.

The masked autoregressive flow (MAF) provided an explicit likelihood. The likelihood of the variational autoencoder (VAE) was approximated from the reconstruction error. Including the KL divergence to the latent prior made results worse. The likelihood of the Archimax copula was approximated from the inclusion-exclusion scheme, checked to converge using various interval sizes. The AUC and F1 scores for outlier detection based on likelihoods are given in Table 5.5. A visual comparison of the results is given in Figure 5.21.

Table 5.5: Accuracy of out-of-distribution detection

	MAF	VAE	OURS
AUC	0.82	0.37	**0.92**
F1	0.48	0.04	**0.72**

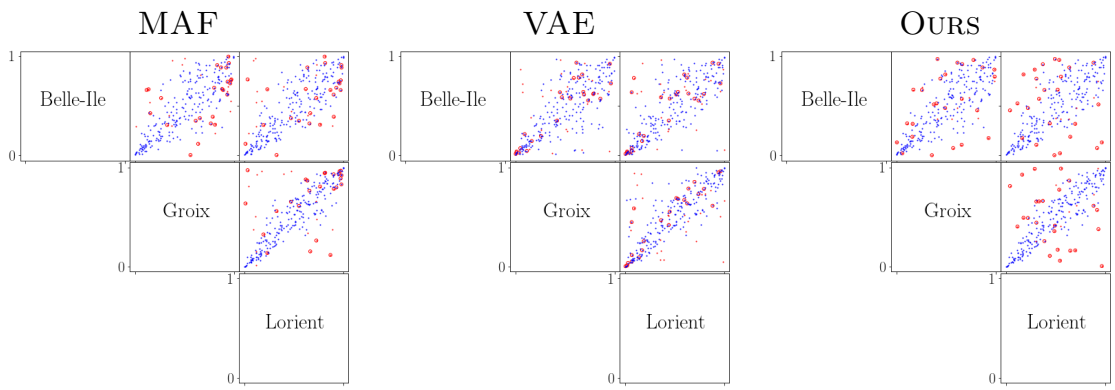

FIGURE 5.21: Samples from likelihood-based out-of-distribution detection. Inliers (blue), outliers (red), and detected points (circled in red).

5.7.5 Learning High-Dimensional Archimax Copulas

We finally consider an experiment where we infer and sample data from a 100-dimensional Clayton-NSD Archimax copula, with parameters $\theta = 2$, $\alpha = (\alpha_0, \cdots, \alpha_0)$, $\alpha_0 = (1, 1, 1, 1, 2, 2, 2, 3, 3, 4)$ and $\rho = 0.69$. Scaling to high dimensions is an important motivation for the proposed method since many existing copula methods fail to scale. We only compare with deep generative models since the existing copula models resulted in numerical errors during optimization. We reported the CvM statistic in the second line of Table 5.3. We plotted the samples, 20 coordinates at a time, with coordinates $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29)$ in Figure 5.22 and coordinates $(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 9, 19, 29, 39, 49, 59, 69, 79, 89, 99)$ in Figure 5.23.

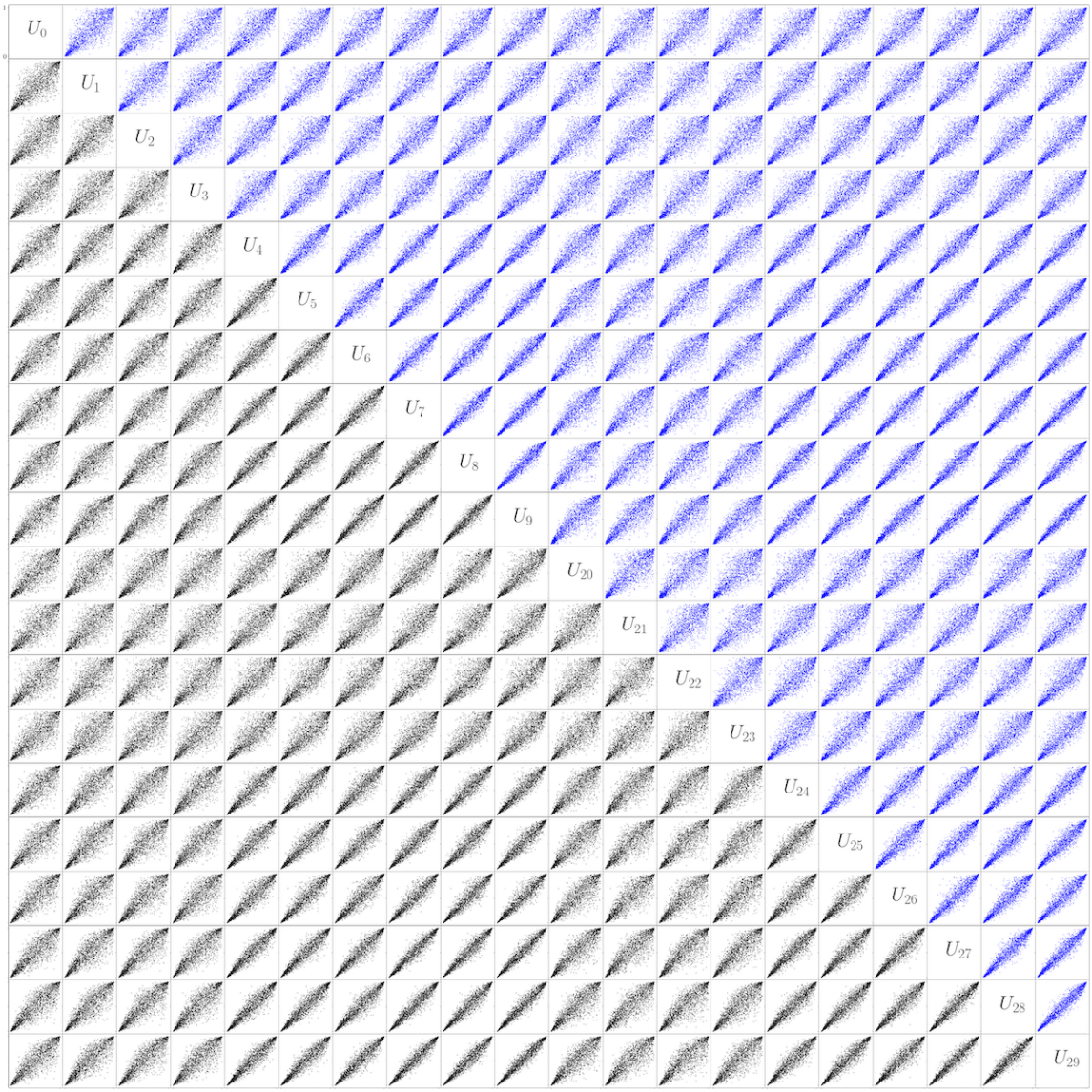

FIGURE 5.22: Samples from fitting $100d$ Clayton-NSD copula, showing coordinates $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29)$, displayed as mirrors on the diagonal, generative Archimax copulas above in blue and ground truth below in black. Each plot is a bivariate margin (U_i, U_j).

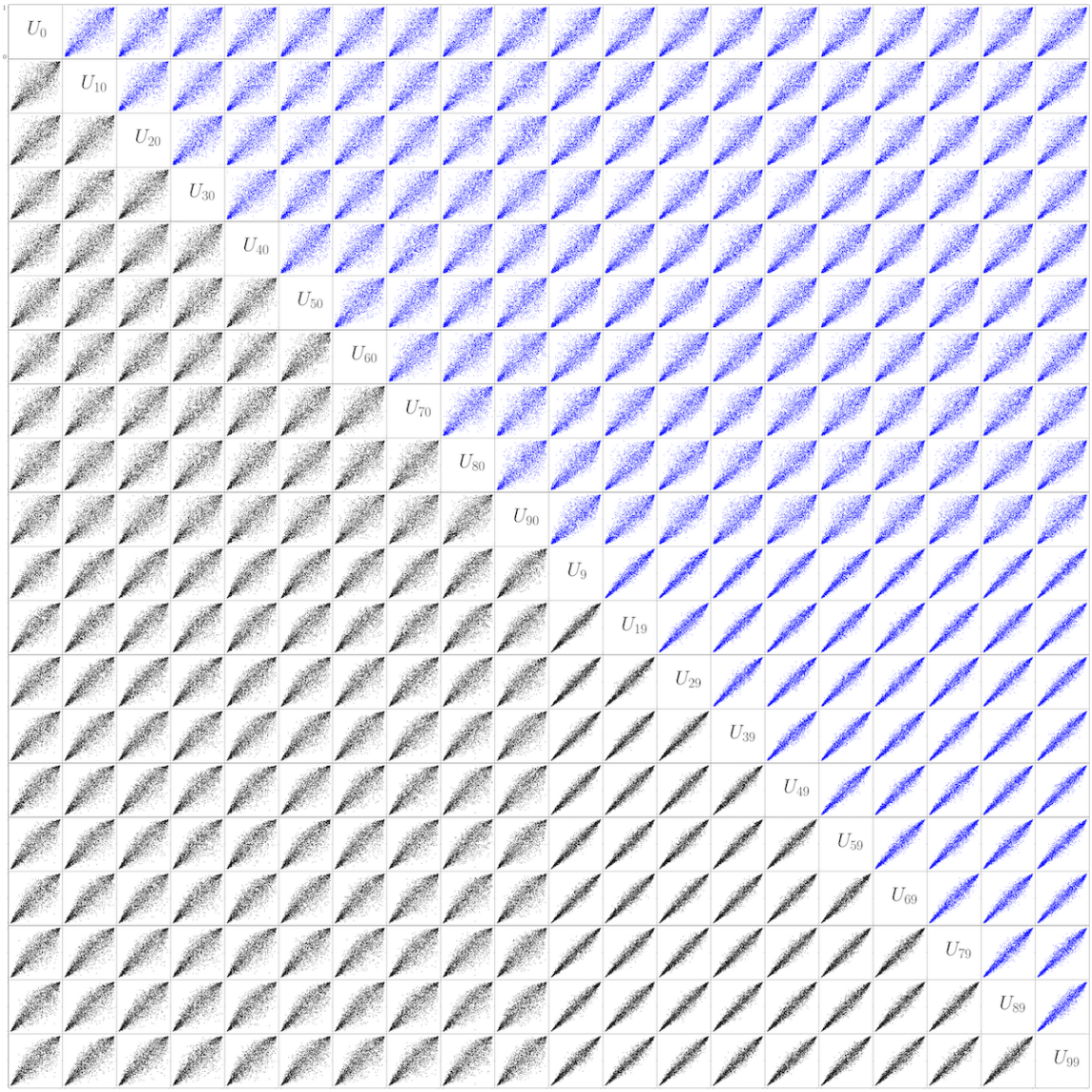

FIGURE 5.23: Samples from fitting $100d$ Clayton-NSD copula (continued), showing coordinates $(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 9, 19, 29, 39, 49, 59, 69, 79, 89, 99)$, displayed as mirrors on the diagonal, generative Archimax copulas above in blue and ground truth below in black. Each plot is a bivariate margin (U_i, U_j).

5.8 Summary

We developed flexible representations and scalable methods for inference and sampling, facilitating the use of Archimax copulas in practical settings. We focus on the stochastic representation of Archimax copulas and generative networks to represent the latent radial and spectral components. Key takeaways are sampling the simplex component through the spectral component and inferring the radial component through sampling the simplex component. We experimentally compare to state-of-the-art density modeling techniques, and the results suggest that the generative Archimax copula effectively models data in the bulk, extrapolates to tails, while scaling to high dimensions.

6

Open Problems

In this thesis, we only discussed Archimedean, extreme-value and Archimax copulas. In addition, the current parameterization of the Archimedean generator does not lend itself to a straightforward calculation of the inverse. It may be interesting to borrow ideas from invertible neural networks to design an Archimedean generator with a straightforward calculation of the inverse. Another line of work is to explore a network parameterization for a hierarchy of d-monotone nested Archimedean generators. Yet another line of work is to consider random variables whose distributions are conveniently described by Laplace transforms, such as the first hitting times of Brownian motions. In addition, it will be helpful to characterize the properties and performances of the given parameterizations, estimation and sampling methods. For example, building on the Lipschitz continuity of copulas, and the weak convergence of the empirical copula processes [36], perhaps we can write some guarantees on the generalization error of deep network based copulas. Furthermore, for the copulas to be interpretable, we need deeper insights into the probabilistic constructions and natural modelling situations.

Other open problems include flexible models not covered by the above classes,

with effective estimation and sampling methods in high dimensions. An interesting line of work may be to consider the copula as the gradient of input convex neural networks and training with the optimal transport loss [14, 91]. Another interesting line of work may be to consider the application of copulas in robust optimization, such as the use of copulas to provide additional constraints on the adversary, and in reverse, the application of robust optimization in the estimation of copulas, such as considering the set of all copulas compatible with the dataset.

Another open problem may be to analyze the dependencies given by some neural structures in terms of copulas. For example, if an exchangeable structure is present, it might be useful to think in terms of Archimedean copulas.

An existing problem is the intersection of copulas and stochastic processes. Existing works have mainly been divided into studying either temporal dependence or spatial dependence. Interesting applications using copula for temporal dependence include unemployment hysteresis in which unemployment appears to become "stuck" at relatively high levels using copula with upper tail dependence, and non-reversible temporal behaviour such as relatively long periods of fairly steady growth, interspersed with shorter periods of sharp decline in financial time series using asymmetric copulas [6]. Interesting works on using copula for time-varying spatial dependence include copula-Itô processes [64]. It may be interesting to study the copulas of McKean-Vlasov processes, a generalization of Itô processes by including distributional dependence in the parameterization of the process.

A word of caution, borrowing from [65], copulas can help us understand some problems about stochastic dependence, but they are not the solution to all problems of stochastic dependence.

7

Conclusions

In summary, we developed flexible neural representations, scalable training and sampling methods for neural Archimedean, extreme-value and Archimax copulas.

For Archimedean copulas with the completely monotone Archimedean generator, we considered the empirical Laplace transform using samples of the latent variable modeled with generative networks. We used the properties of the Laplace transform to compute higher-order derivatives for likelihood-based training. We modified existing Marshall-Olkin type sampling to our parameterization with generative networks. We described an application of our training and sampling techniques to scale an existing neural representation to higher dimensions. We extend our method with latent representations of Laplace transforms to hierarchical Archimedean copulas.

For extreme-value copulas, we considered the spectral decomposition of the stable tail dependence function (stdf). We modeled the support of the spectral variable with weights of a neural network; we also considered the empirical expectation using samples of the spectral variable modeled with generative networks. We borrow from the Pickands estimator and perform likelihood-based training on exponentially distributed transformed observations. We modified existing approximate sampling

of max-stable processes to our parameterization with neural networks.

For Archimax copulas, we considered the stochastic representation with radial and simplex components and the more general d-monotone Archimedean generator given as the Williamson d-transform of the radial component. We train the stdf with a modification to the Pickands transform on the Archimedean generator. We sample the simplex component through the stochastic representation of generalized Pareto copulas and samples of the spectral component from the learnt stdf. We train the Archimedean generator using a decomposition of the Kendall distribution, another data transformation, and samples of the radial and simplex components. We put these training and sampling techniques together to learn and sample from the generative Archimax copula. We empirically demonstrate the results on high dimensional data, tail extrapolation and out-of-distribution detection.

The stochastic representations of the above copulas give their characterizing functions as expectations of latent variables. In the network representations, the support of the latent variables are encoded in the weights of the network. In the generative representations, we consider empirical expectations using samples of the latent variables modeled by generative networks. While both representations are equivalent, the deterministic representation is more suitable for computing statistical quantities while the generative representation is more suitable for sample generation. We used the properties of the expectation and data transformations to avoid repeated differentiation in likelihood-based training. We modified the sampling frameworks that came naturally with the stochastic representations to our parameterization with generative networks. We empirically validated the expressiveness of the models, accuracy of the estimation and sampling algorithms, extensions to hierarchical Archimedean copulas, high dimensional Archimax copulas with complex dependencies, tail extrapolation and out-of-distribution detection.

Bibliography

[1] Allouche, M., Girard, S., and Gobet, E. (2022), "EV-GAN: Simulation of extreme events with ReLU neural networks," *Journal of Machine Learning Research*, 23, 1–39.

[2] Amos, B., Xu, L., and Kolter, J. Z. (2017), "Input Convex Neural Networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 146–155, PMLR.

[3] Arjovsky, M., Chintala, S., and Bottou, L. (2017), "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR.

[4] Bacigál, T., Jágr, V., and Mesiar, R. (2011), "Non-exchangeable random variables, Archimax copulas and their fitting to real data," *Kybernetika*, 47, 519–531.

[5] Barbe, P., Genest, C., Ghoudi, K., and Rémillard, B. (1996), "On Kendall's Process," *Journal of Multivariate Analysis*, 58, 197–229.

[6] Beare, B. K. (2010), "Copulas and Temporal Dependence," *Econometrica*, 78, 395–410.

[7] Belzile, L. R. and Nešlehová, J. G. (2017), "Extremal attractors of Liouville copulas," *Journal of Multivariate Analysis*, 160, 68–92.

[8] Bernstein, S. (1929), "Sur les fonctions absolument monotones," *Acta Mathematica*, 52, 1 – 66.

[9] Boulaguiem, Y., Zscheischler, J., Vignotto, E., van der Wiel, K., and Engelke, S. (2022), "Modeling and simulating spatial extremes by combining extreme value theory with generative adversarial networks," *Environmental Data Science*, 1, e5.

[10] Branch, M. A., Coleman, T. F., and Li, Y. (1999), "A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems," *SIAM Journal on Scientific Computing*, 21, 1–23.

[11] Buishand, T. A., de Haan, L., and Zhou, C. (2008), "On Spatial Extremes: With Application to a Rainfall Problem," *The Annals of Applied Statistics*, 2, 624–642.

[12] Capéraà, P., Fougères, A.-L., and Genest, C. (1997), "A nonparametric estimation procedure for bivariate extreme value copulas," *Biometrika*, 84, 567–577.

[13] Capéraà, P., Fougères, A.-L., and Genest, C. (2000), "Bivariate Distributions with Given Extreme Value Attractor," *Journal of Multivariate Analysis*, 72, 30–49.

[14] Carlier, G., Chernozhukov, V., and Galichon, A. (2016), "Vector quantile regression: An optimal transport approach," *The Annals of Statistics*, 44, 1165 – 1192.

[15] Chang, B., Pan, S., and Joe, H. (2019), "Vine copula structure learning via Monte Carlo tree search," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, eds. K. Chaudhuri and M. Sugiyama, vol. 89 of *Proceedings of Machine Learning Research*, pp. 353–361, PMLR.

[16] Chang, Y., Li, Y., Ding, A., and Dy, J. (2016), "A Robust-Equitable Copula Dependence Measure for Feature Selection," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, eds. A. Gretton and C. C. Robert, vol. 51 of *Proceedings of Machine Learning Research*, pp. 84–92, Cadiz, Spain, PMLR.

[17] Charpentier, A., Fermanian, J.-D., and Scaillet, O. (2007), "The estimation of copulas: Theory and practice," *Copulas: From theory to application in finance*, pp. 35–64.

[18] Charpentier, A., Fougères, A.-L., Genest, C., and Nešlehová, J. (2014), "Multivariate Archimax copulas," *Journal of Multivariate Analysis*, 126, 118–136.

[19] Chatelain, S., Fougères, A.-L., and Nešlehová, J. G. (2020), "Inference for Archimax copulas," *The Annals of Statistics*, 48, 1025 – 1051.

[20] Cherubini, U., Luciano, E., and Vecchiato, W. (2004), *Copula Methods in Finance*, The Wiley Finance Series, Wiley.

[21] Chilinski, P. and Silva, R. (2020), "Neural Likelihoods via Cumulative Distribution Functions," in *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, eds. J. Peters and D. Sontag, vol. 124 of *Proceedings of Machine Learning Research*, pp. 420–429, PMLR.

[22] Coles, S. (2001), *An Introduction to Statistical Modeling of Extreme Values*, Springer Series in Statistics, Springer London, London.

[23] Cramér, H. (1928), "On the composition of elementary errors," *Scandinavian Actuarial Journal*, 1928, 13–74.

[24] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018), "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, 35, 53–65.

[25] Cui, R., Groot, P., Schauer, M., and Heskes, T. (2018), "Learning the causal structure of copula models with latent variables," in *UAI*, Corvallis: AUAI Press.

[26] Data to AI Lab at MIT (2022), "Copulas v0.7.0," .

[27] De Haan, L. (1984), "A Spectral Representation for Max-stable Processes," *The Annals of Probability*, 12, 1194 – 1204.

[28] Demarta, S. and McNeil, A. J. (2005), "The t Copula and Related Copulas," *International Statistical Review / Revue Internationale de Statistique*, 73, 111–129.

[29] Dombry, C., Engelke, S., and Oesting, M. (2016), "Exact simulation of max-stable processes," *Biometrika*, 103, 303–317.

[30] Drouin, A., Marcotte, E., and Chapados, N. (2022), "TACTiS: Transformer-Attentional Copulas for Time Series," in *Proceedings of the 39th International Conference on Machine Learning*, eds. K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, vol. 162 of *Proceedings of Machine Learning Research*, pp. 5447–5493, PMLR.

[31] Eban, E., Rothschild, G., Mizrahi, A., Nelken, I., and Elidan, G. (2013), "Dynamic copula networks for modeling real-valued time series," in *Artificial Intelligence and Statistics*, pp. 247–255, PMLR.

[32] Elidan, G. (2010a), "Copula Bayesian Networks," in *Advances in Neural Information Processing Systems*, eds. J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, vol. 23, pp. 559–567, Curran Associates, Inc.

[33] Elidan, G. (2010b), "Inference-less density estimation using copula bayesian networks," in *UAI*.

[34] Elidan, G. (2012), "Copula network classifiers (cncs)," in *Artificial intelligence and statistics*, pp. 346–354, PMLR.

[35] Falk, M., Padoan, S. A., and Wisheckel, F. (2019), "Generalized Pareto copulas: A key to multivariate extremes," *Journal of Multivariate Analysis*, 174, 104538.

[36] Fermanian, J.-D., Radulovic, D., and Wegkamp, M. (2004), "Weak convergence of empirical copula processes," *Bernoulli*, 10, 847 – 860.

[37] Fils-Villetard, A., Guillou, A., and Segers, J. (2008), "Projection estimators of Pickands dependence functions," *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, 36, 369–382.

[38] Fougères, A.-L., Mercadier, C., and Nolan, J. (2013), "Dense classes of multivariate extreme value distributions," *Journal of Multivariate Analysis*, 116, 109–129.

[39] Genest, C. and Favre, A.-C. (2007), "Everything You Always Wanted to Know about Copula Modeling but Were Afraid to Ask," *Journal of Hydrologic Engineering*, 12, 347–368.

[40] Genest, C. and Rivest, L.-P. (1993), "Statistical Inference Procedures for Bivariate Archimedean Copulas," *Journal of the American Statistical Association*, 88, 1034–1043.

[41] Genest, C., Remillard, B., and Beaudoin, D. (2009), "Goodness-of-fit tests for copulas: A review and a power study," *Insurance: Mathematics and Economics*, 44, 199–213.

[42] Genest, C., Nešlehová, J., and Ziegel, J. (2011), "Inference in multivariate Archimedean copula models," *TEST: An Official Journal of the Spanish Society of Statistics and Operations Research*, 20, 223.

[43] Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013), "Maxout Networks," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28 of *Proceedings of Machine Learning Research*, pp. 1319–1327, Atlanta, Georgia, USA, PMLR.

[44] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014), "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, eds. Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, vol. 27, pp. 2672–2680, Curran Associates, Inc.

[45] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012), "A Kernel Two-Sample Test," *Journal of Machine Learning Research*, 13, 723–773.

[46] Gudendorf, G. and Segers, J. (2010), "Extreme-Value Copulas," in *Copula Theory and Its Applications*, pp. 127–145, Springer Berlin Heidelberg.

[47] Górecki, J. and Kurz, M. S. (2020), "The HACopula Toolbox," .

[48] Górecki, J., Hofert, M., and Holeňa, M. (2017), "On structure, family and parameter estimation of hierarchical Archimedean copulas," *Journal of Statistical Computation and Simulation*, 87, 3261–3324.

[49] Han, S., Liao, X., Dunson, D., and Carin, L. (2016), "Variational Gaussian Copula Inference," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, eds. A. Gretton and C. C. Robert, vol. 51 of *Proceedings of Machine Learning Research*, pp. 829–838, Cadiz, Spain, PMLR.

[50] Hasan, A., Elkhalil, K., Ng, Y., Pereira, J. M., Farsiu, S., Blanchet, J., and Tarokh, V. (2022), "Modeling Extremes with d-max-decreasing Neural Networks," in *The 38th Conference on Uncertainty in Artificial Intelligence*.

[51] Hering, C., Hofert, M., Mai, J.-F., and Scherer, M. (2010), "Constructing hierarchical Archimedean copulas with Lévy subordinators," *Journal of Multivariate Analysis*, 101, 1428–1433.

[52] Hernández-Lobato, J. M., Lloyd, J. R., and Hernández-Lobato, D. (2013), "Gaussian Process Conditional Copulas with Applications to Financial Time Series," in *Advances in Neural Information Processing Systems*, eds. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, vol. 26, Curran Associates, Inc.

[53] Hirt, M., Dellaportas, P., and Durmus, A. (2019), "Copula-like Variational Inference," in *Advances in Neural Information Processing Systems*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, vol. 32, Curran Associates, Inc.

[54] Hofert, M. (2008), "Sampling Archimedean Copulas," *Computational Statistics & Data Analysis*, 52, 5163–5174.

[55] Hofert, M., Mächler, M., and McNeil, A. J. (2013), "Archimedean Copulas in High Dimensions: Estimators and Numerical Challenges Motivated by Financial Applications," *Journal de la société française de statistique*, 154, 25–63.

[56] Hofert, M., Huser, R., and Prasad, A. (2018), "Hierarchical Archimax copulas," *Journal of Multivariate Analysis*, 167, 195–211.

[57] Hofmann, D. (2009), "Characterization of the D-norm corresponding to a multivariate extreme value distribution," Ph.D. thesis, Universität Würzburg.

[58] Huang, J. and Frey, B. J. (2008a), "Structured ranking learning using cumulative distribution networks," in *Advances in Neural Information Processing Systems*, eds. D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, vol. 21, Curran Associates, Inc.

[59] Huang, J. and Jojic, N. (2010), "Maximum-likelihood learning of cumulative distribution functions on graphs," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, eds. Y. W. Teh and M. Titterington, vol. 9 of *Proceedings of Machine Learning Research*, pp. 342–349, Chia Laguna Resort, Sardinia, Italy, PMLR.

[60] Huang, J. C. and Frey, B. J. (2008b), "Cumulative Distribution Networks and the Derivative-Sum-Product Algorithm," in *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, p. 290–297, Arlington, Virginia, USA, AUAI Press.

[61] Huang, J. C. and Frey, B. J. (2011), "Cumulative Distribution Networks and the Derivative-sum-product Algorithm: Models and Inference for Cumulative Distribution Functions on Graphs," *Journal of Machine Learning Research*, 12, 301–348.

[62] Huang, X. (1992), "Statistics of bivariate extreme values," PhD, Tinbergen Institute.

[63] Janke, T., Ghanmi, M., and Steinke, F. (2021), "Implicit Generative Copulas," in *Advances in Neural Information Processing Systems*, eds. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, vol. 34, pp. 26028–26039, Curran Associates, Inc.

[64] Jaworski, P. (2019), "On Copula-Itô processes," *Dependence Modeling*, 7, 322–347.

[65] Jaworski, P., Durante, F., Hardle, W. K., and Rychlik, T. (eds.) (2010), *Copula theory and its applications*, Lecture notes in statistics, Springer, Berlin, Germany, 2010 edn.

[66] Jaworski, P., Durante, F., and Hardle, W. K. (eds.) (2013), *Copulae in mathematical and quantitative finance*, Lecture notes in statistics, Springer, Berlin, Germany, 2013 edn.

[67] Joe, H. (1997), *Multivariate models and dependence concepts*, Chapman and Hall/CRC.

[68] Joe, H. (2014), *Dependence modeling with copulas*, Chapman and Hall/CRC.

[69] Jojic, N., Meek, C., and Huang, J. (2010), "Exact inference and learning for cumulative distribution functions on loopy graphs," in *Advances in Neural Information Processing Systems*, eds. J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, vol. 23, pp. 874–882, Curran Associates, Inc.

[70] Kamthe, S., Assefa, S., and Deisenroth, M. (2021), "Copula Flows for Synthetic Data Generation," .

[71] Kimberling, C. H. (1974), "A probabilistic interpretation of complete monotonicity," *aequationes mathematicae*, 10, 152–164.

[72] Kingma, D. P. and Ba, J. (2015), "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[73] Kingma, D. P. and Welling, M. (2014), "Auto-encoding variational Bayes," in *International Conference on Learning Representations*.

[74] Kingma, D. P. and Welling, M. (2019), "An Introduction to Variational Autoencoders," *Foundations and TrendsÂ® in Machine Learning*, 12, 307–392.

[75] Kojadinovic, I., Segers, J., and Yan, J. (2011), "Large-sample tests of extreme-value dependence for multivariate copulas," LIDAM Reprints ISBA 2011025, Université catholique de Louvain, Institute of Statistics, Biostatistics and Actuarial Sciences (ISBA).

[76] Kollo, T. and Pettere, G. (2010), "Parameter Estimation and Application of the Multivariate Skew t-Copula," in *Copula Theory and Its Applications*, eds. P. Jaworski, F. Durante, W. K. Härdle, and T. Rychlik, pp. 289–298, Berlin, Heidelberg, Springer Berlin Heidelberg.

[77] Kom Samo, Y.-L. (2021), "Inductive Mutual Information Estimation: A Convex Maximum-Entropy Copula Approach," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, eds. A. Banerjee and K. Fukumizu, vol. 130 of *Proceedings of Machine Learning Research*, pp. 2242–2250, PMLR.

[78] Landgrebe, E., Udell, M., et al. (2020), "Online mixed missing value imputation using Gaussian copula," in *Workshop on the Art of Learning with Missing Values (Artemiss) hosted by the 37th International Conference on Machine Learning (ICML)*.

[79] Laszkiewicz, M., Lederer, J., and Fischer, A. (2021), "Copula-Based Normalizing Flows," in *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

[80] Le Lan, C. and Dinh, L. (2021), "Perfect density models cannot guarantee anomaly detection," *Entropy*, 23, 1690.

[81] Letham, B., Sun, W., and Sheopuri, A. (2014), "Latent Variable Copula Inference for Bundle Pricing from Retail Transaction Data," in *ICML*, pp. 217–225.

[82] Li, Z., Zhao, Y., Botta, N., Ionescu, C., and Hu, X. (2020), "COPOD: copula-based outlier detection," in *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 1118–1123, IEEE.

[83] Ling, C. K., Fang, F., and Kolter, J. Z. (2020), "Deep Archimedean Copulas," in *Advances in Neural Information Processing Systems*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, vol. 33, pp. 1535–1545, Curran Associates, Inc.

[84] Liu, W. (2019), "Copula Multi-label Learning," in *Advances in Neural Information Processing Systems*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, vol. 32, Curran Associates, Inc.

[85] Lopez-paz, D., Hernández-lobato, J., and Schölkopf, B. (2012), "Semi-Supervised Domain Adaptation with Non-Parametric Copulas," in *Advances in Neural Information Processing Systems*, eds. F. Pereira, C. Burges, L. Bottou, and K. Weinberger, vol. 25, Curran Associates, Inc.

[86] Lopez-Paz, D., Hernández-Lobato, J. M., and Zoubin, G. (2013), "Gaussian process vine copulas for multivariate dependence," in *International Conference on Machine Learning*, pp. 10–18, PMLR.

[87] Ma, J. and Sun, Z. (2011), "Mutual information is copula entropy," *Tsinghua Science and Technology*, 16, 51–54.

[88] Ma, J., Chang, B., Zhang, X., and Mei, Q. (2021), "CopulaGNN: Towards Integrating Representational and Correlational Roles of Graphs in Graph Neural Networks," in *International Conference on Learning Representations*.

[89] Mai, J.-F. and Scherer, M. (2017), *Simulating Copulas*, WORLD SCIENTIFIC, 2nd edn.

[90] Makkuva, A., Taghvaei, A., Oh, S., and Lee, J. (2020a), "Optimal transport mapping via input convex neural networks," in *Proceedings of the 37th International Conference on Machine Learning*, eds. H. D. III and A. Singh, vol. 119 of *Proceedings of Machine Learning Research*, pp. 6672–6681, PMLR.

[91] Makkuva, A., Taghvaei, A., Oh, S., and Lee, J. (2020b), "Optimal transport mapping via input convex neural networks," in *Proceedings of the 37th International Conference on Machine Learning*, eds. H. D. III and A. Singh, vol. 119 of *Proceedings of Machine Learning Research*, pp. 6672–6681, PMLR.

[92] Marshall, A. W. and Olkin, I. (1988), "Families of Multivariate Distributions," *Journal of the American Statistical Association*, 83, 834–841.

[93] Mazaheri, B., Jain, S., and Bruck, J. (2020), "Robust Correction of Sampling Bias using Cumulative Distribution Functions," in *Advances in Neural Information Processing Systems*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, vol. 33, pp. 3546–3556, Curran Associates, Inc.

[94] McNeil, A. J. (2008), "Sampling nested Archimedean copulas," *Journal of Statistical Computation and Simulation*, 78, 567–581.

[95] McNeil, A. J. and Nešlehová, J. (2010), "From Archimedean to Liouville copulas," *Journal of Multivariate Analysis*, 101, 1772–1790.

[96] McNeil, A. J. and Nešlehová, J. (2009), "Multivariate Archimedean copulas, d-monotone functions and $l1$-norm symmetric distributions," *The Annals of Statistics*, 37, 3059 – 3097.

[97] Mesiar, R. and Jágr, V. (2013), "d-Dimensional dependence functions and Archimax copulas," *Fuzzy Sets and Systems*, 228, 78–87.

[98] Nelsen, R. B. (2010), *An introduction to Copulas*, Springer series in statistics, Springer New York, New York, NY, 2. ed. 2006. corr. 2. pr. softcover version of original hardcover edition 2006 edn., OCLC: 700190717.

[99] Ng, Y., Hasan, A., Elkhalil, K., and Tarokh, V. (2021), "Generative Archimedean copulas," in *37th Conference on Uncertainty in Artificial Intelligence (UAI)*.

[100] Ng, Y., Hasan, A., and Tarokh, V. (2022), "Inference and Sampling for Archimax Copulas," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 17099–17116.

[101] Okhrin, O., Okhrin, Y., and Schmid, W. (2013), "On the structure and estimation of hierarchical Archimedean copulas," *Journal of Econometrics*, 173, 189–204.

[102] Ostrovski, G., Dabney, W., and Munos, R. (2018), "Autoregressive Quantile Networks for Generative Modeling," in *ICML*, pp. 3933–3942.

[103] Papamakarios, G., Pavlakou, T., and Murray, I. (2017), "Masked autoregressive flow for density estimation," *Advances in neural information processing systems*, 30.

[104] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021), "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research*, 22, 1–64.

[105] Pickands, J. (1981), "Multivariate extreme value distribution," in *43rd Session of the International Statistical Institution*, vol. 2, pp. 859–878,894–902, Buenos Aires.

[106] Póczos, B., Ghahramani, Z., and Schneider, J. (2012), "Copula-Based Kernel Dependency Measures," in *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML'12, p. 1635–1642, Madison, WI, USA, Omnipress.

[107] Remillard, B. and Scaillet, O. (2007), "Testing for Equality between Two Copulas," 8, 229–231.

[108] Ressel, P. (2013), "Homogeneous distributions—And a spectral representation of classical mean values and stable tail dependence functions," *Journal of Multivariate Analysis*, 117, 246–256.

[109] Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. (2019), "High-dimensional multivariate forecasting with low-rank Gaussian Copula Processes," in *Advances in Neural Information Processing Systems*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, vol. 32, Curran Associates, Inc.

[110] Salinas, D., Shen, H., and Perrone, V. (2020), "A Quantile-based Approach for Hyperparameter Transfer Learning," in *Proceedings of the 37th International Conference on Machine Learning*, eds. H. D. III and A. Singh, vol. 119 of *Proceedings of Machine Learning Research*, pp. 8438–8448, PMLR.

[111] Sato, K.-i. (1999), *Lévy processes and infinitely divisible distributions*, no. 68 in Cambridge studies in advanced mathematics, Cambridge University Press.

[112] Savu, C. and Trede, M. (2010), "Hierarchies of Archimedean copulas," *Quantitative Finance*, 10, 295–304.

[113] Schlather, M. (2002), "Models for Stationary Max-Stable Random Fields," *Extremes*, 5, 33–44.

[114] Silva, R., Blundell, C., and Teh, Y. W. (2011), "Mixed Cumulative Distribution Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, eds. G. Gordon, D. Dunson, and M. Dudík, vol. 15 of *Proceedings of Machine Learning Research*, pp. 670–678, Fort Lauderdale, FL, USA, PMLR.

[115] Sklar, A. (1959), "Fonctions de Répartition à n Dimensions et Leurs Marges," 8, 229–231.

[116] Smith, M. S., Gan, Q., and Kohn, R. J. (2012), "Modelling dependence using skew t copulas: Bayesian inference and applications," *Journal of Applied Econometrics*, 27, 500–522.

[117] Tagasovska, N., Ackerer, D., and Vatter, T. (2019), "Copulas as High-Dimensional Generative Models: Vine Copula Autoencoders," in *Advances in Neural Information Processing Systems*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, vol. 32, Curran Associates, Inc.

[118] Tenzer, Y. and Elidan, G. (2013), "Speedy model selection (SMS) for copula models," in *UAI*.

[119] Tenzer, Y. and Elidan, G. (2014), "HELM: Highly Efficient Learning of Mixed copula networks." in *UAI*, pp. 790–799.

[120] Tran, D., Blei, D., and Airoldi, E. M. (2015), "Copula variational inference," in *Advances in Neural Information Processing Systems*, eds. C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, vol. 28, Curran Associates, Inc.

[121] USDA (1985), "DATASET: CSFII 1985, CONTINUING SURVEY OF FOOD INTAKES BY INDIVIDUALS, Women 19-50 Years of Age and Their Children 1-5 Years of Age, 6 Waves, 1985," Dataset, U.S. Department of Agriculture, Agriculture Research Service, `https://www.ars.usda.gov/ARSUserFiles/80400530/pdf/8586/csfii85_6waves_doc.pdf`, `https://www.ars.usda.gov/northeast-area/beltsville-md-bhnrc/beltsville-human-nutrition-research-center/food-surveys-research-group/docs/csfii-1985-1986/`.

[122] Wang, H., Fazayeli, F., Chatterjee, S., and Banerjee, A. (2014), "Gaussian Copula Precision Estimation with Missing Values," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, eds. S. Kaski and J. Corander, vol. 33 of *Proceedings of Machine Learning Research*, pp. 978–986, Reykjavik, Iceland, PMLR.

[123] Wang, H., Yu, L., Cao, Z., and Ermon, S. (2021), "Multi-agent imitation learning with copulas," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 139–156, Springer.

[124] Wen, R. and Torkkola, K. (2019), "Deep generative quantile-copula models for probabilistic forecasting," in *ICML*.

[125] Widder, D. V. (1941), *Laplace Transform (PMS-6)*, Princeton University Press.

[126] Wieser, M., Wieczorek, A., Murezzan, D., and Roth, V. (2018), "Learning Sparse Latent Representations with the Deep Copula Information Bottleneck," in *International Conference on Learning Representations*.

[127] Williamson, R. E. (1956), "Multiply monotone functions and their Laplace transforms," *Duke Mathematical Journal*, 23, 189 – 207.

[128] Wilson, A. G. and Ghahramani, Z. (2010), "Copula Processes," in *Advances in Neural Information Processing Systems*, eds. J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, vol. 23, pp. 2460–2468, Curran Associates, Inc.

[129] Yoshiba, T. (2018), "Maximum likelihood estimation of skew-t copulas with its applications to stock returns," *Journal of Statistical Computation and Simulation*, 88, 2489–2506.

[130] Zhao, Y. and Udell, M. (2020), "Matrix Completion with Quantified Uncertainty through Low Rank Gaussian Copula," in *Advances in Neural Information Processing Systems*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, vol. 33, pp. 20977–20988, Curran Associates, Inc.