Generative Archimedean Copulas

Yuting Ng[1] **Ali Hasan**[2] **Khalil Elkhalil**[1] **Vahid Tarokh**[1]

[1]Department of Electrical and Computer Engineering, Duke University
[2]Department of Biomedical Engineering, Duke University

Abstract

We propose a new generative modeling technique for learning multidimensional cumulative distribution functions (CDFs) in the form of copulas. Specifically, we consider certain classes of copulas known as Archimedean and hierarchical Archimedean copulas, popular for their parsimonious representation and ability to model different tail dependencies. We consider their representation as mixture models with Laplace transforms of latent random variables from generative neural networks. This alternative representation allows for computational efficiencies and easy sampling, especially in high dimensions. We describe multiple methods for optimizing the network parameters. Finally, we present empirical results that demonstrate the efficacy of our proposed method in learning multidimensional CDFs and its computational efficiency compared to existing methods.

1 INTRODUCTION

Copulas are a special class of cumulative distribution functions (CDFs) that model the dependencies between multiple random variables in isolation from their marginals [Nelsen, 2010, Joe, 2014]. Copulas have found applications in many areas, including hydrology [Genest and Favre, 2007] and finance [Cherubini et al., 2004]. In finance, for example, more expressive modeling using copulas of the joint distribution of two stocks can result in more pairs trading opportunities [Stander et al., 2013, Liew and Wu, 2013].

In machine learning, copulas have been used to create new distributions, increasing the flexibility of modeling multivariate dependencies [Wilson and Ghahramani, 2010, Elidan, 2010, Huang and Frey, 2011, Tagasovska et al., 2019, Wiese et al., 2019, Kamthe et al., 2021, Chilinski and Silva, 2020]. 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 and estimation are still unsolved.

A particularly useful class of copulas are known as Archimedean copulas, which endow a specific structure for representing the dependence function in terms of a one-dimensional *generator* function. Most work involving Archimedean copulas consider different parameterizations for this generator function. Parameterizations of the generator function have generally been limited to simple forms, since complicated generator functions lead to difficulties in computing the copula density, a necessary component for maximum likelihood estimation. Ling et al. [2020] proposed parameterizing the generator function as a neural network, but ran into computational difficulties for dimensions greater than 5. We consider an alternative construction based on a mixture representation with latent random variables, first proposed by Marshall and Olkin [1988], wherein we parameterize a latent distribution, whose Laplace transform acts as the generator function, with a generative neural network. Depending on the application, this latent variable is sometimes known as a *resilience* or *frailty* parameter. Using this construction, we can scale computations to higher dimensions and bypass numerical issues involved with automatic differentiation.

Employing the Laplace transform to a learned latent model also provides important benefits beyond computational efficiency and numerical stability. When sampling from the copula using established approaches [Marshall and Olkin, 1988, McNeil, 2008, Hering et al., 2010], knowledge of the latent distribution is necessary. Parameterizing the latent distribution with a generative neural network allows for efficient sampling after training.

Archimedean copulas can also be extended to the so-called hierarchical (or nested) Archimedean copulas, where multiple generators are used in conjunction to increase expressiveness of the model [Joe, 1997]. This architecture mitigates a central deficiency of vanilla Archimedean copulas — the assumed symmetry in the dependence structure. We use a construction based on Lévy subordinators, i.e. non-decreasing Lévy processes such as the compound Poisson process, first proposed by Hering et al. [2010], and parameterize the Lévy subordinators using generative neural networks. We also use Laplace transforms as in the vanilla Archimedean copula to obtain the generator functions and subsequently recover a richer class of copulas.

RELATED WORK

Our part of work on Archimedean copulas is related to [Ling et al., 2020], where a neural network is proposed to represent the generator function of an Archimedean copula. We propose instead a generative neural network to represent the latent random variable, whose Laplace transform gives the generator function of the Archimedean copula. We then approximate the Laplace transform with the empirical Laplace transform using samples from the generative neural network. We note that there exist prior work that replace the Laplace transform with the empirical Laplace transform, such as those on the estimation of compound Poisson processes and distribution goodness-of-fit tests. These can be found in [Csörgő and Teugels, 1990, Henze et al., 2012], but do not consider/employ neural networks.

Existing semiparametric methods for Archimedean copulas are mainly concentrated on two dimensional cases, and their efficacy in higher dimensions remains unclear [Hernández-Lobato and Suárez, 2011, Hoyos-Argüelles and Nieto-Barajas, 2020]. Other work on the mixture representation with a latent random variable is 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 [McNeil, 2008, Hofert, 2008].

Our part of work on hierachical Archimedean copulas is inspired by [Hering et al., 2010] who recognized that sufficient nesting conditions of hierarchical Archimedean copulas may be satisfied using Lévy subordinators. We then let the increments associated with the Lévy measure of the Lévy subordinator be the output of a generative neural network, and compute its integral in the Laplace exponent as an expectation with samples from the generative neural network. Related work parameterizing the Lévy measure with a neural network can be found in [Xu and Darve, 2020], but the integral is approximated as a Riemann sum, and it does not relate to hierarchical Archimedean copulas.

Other related works 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 [McNeil, 2008]. Thus, related works on heterogeneous Archimedean copulas have resulted in limited combinations of different families [McNeil, 2008, Hofert, 2008, Savu and Trede, 2010, Okhrin et al., 2013, Górecki et al., 2017].

MAIN CONTRIBUTIONS

First, we propose to use a generative neural network to represent the latent random variable, whose Laplace transform provides the generator function of an Archimedean copula. This allows approximation of the Laplace transform with its empirical version through samples from the generative neural network. Computing higher-order derivatives using the properties of the empirical Laplace transform additionally allows scalability to higher-dimensional data. Second, we extend this concept to modeling hierarchical Archimedean copulas with Lévy subordinators. We represent the Lévy measure of a Lévy subordinator with a generative neural network and compute its Laplace exponent using samples from the generative neural network. We then propose three methods for training: 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. Finally, we adapt existing Marshall-Olkin type efficient sampling algorithms to our parameterization with generative neural networks. The source code for this paper may be found at `https://github.com/yutingng/gen-AC`.

OUTLINE

Section 2 provides the mathematical background on copulas, Archimedean copulas and hierarchical Archimedean copulas. Section 3 discusses modeling, sampling and training generative Archimedean copulas. Section 4 extends the construction to hierarchical Archimedean copulas. Section 5 shows our experiment results on learning known Archimedean and hierarchical Archimedean copulas that have different tail dependencies. We also compare its flexibility in fitting real-world data to commonly-used one-parameter families. In addition, we show its computational efficiency and sampling in higher-dimensions. Finally, we conclude the paper in Section 6.

2 BACKGROUND

We begin by describing the necessary background on copulas. A *copula* is a multivariate cumulative distribution function (CDF) where all univariate margins are uniform, i.e. it is the CDF of a vector of dependent uniform random variables. Multidimensional dependence modeling with copulas is based on a theorem due to Sklar [1959] which gives a general representation of a multivariate CDF as a composition of its univariate margins and a copula.

Theorem 1 (Sklar's theorem)**.** *For a $d-$variate cumulative distribution function F, with jth univariate margin F_j, and j^{th} quantile function F_j^{-1}, the copula associated with F is a cumulative distribution function $C : [0,1]^d \to [0,1]$ with $U(0,1)$ margins satisfying:*

$$F(\mathbf{x}) = C(F_1(x_1), \cdots, F_d(x_d)),\ \mathbf{x} \in \mathbb{R}^d, \tag{1}$$

$$C(\mathbf{u}) = F(F_1^{-1}(u_1), \cdots, F_d^{-1}(u_d)),\ \mathbf{u} \in [0,1]^d. \tag{2}$$

In addition, if F is continuous, then C is unique.

Moreover, due to Sklar's theorem, every CDF endows such a decomposition. Thus, copulas allow characterization of the multivariate dependence between the random variables $X_1, \cdots, X_d$ separately from their univariate margins $F_1, \cdots, F_d$ [Nelsen, 2010, Joe, 2014].

2.1 ARCHIMEDEAN COPULAS

An important class of copulas are the Archimedean copulas, due to their ease of construction and ability to represent different tail dependencies. An Archimedean copula is defined as:

$$C(\mathbf{u}) = \varphi\left(\varphi^{-1}(u_1) + \cdots + \varphi^{-1}(u_d)\right), \tag{3}$$

with density:

$$\begin{aligned} c(\mathbf{u}) &= \frac{\partial^d C(u_1, \cdots, u_d)}{\partial u_1 \cdots \partial u_d} \\ &= \frac{\varphi^{(d)}(\varphi^{-1}(u_1) + \cdots + \varphi^{-1}(u_d))}{\prod_{i=1}^d \varphi'(\varphi^{-1}(u_i))}. \end{aligned} \tag{4}$$

For the above expression to be a valid copula for all d, the one-dimensional function $\varphi : [0, \infty) \to [0,1]$, known as the *generator* of the Archimedean copula must satisfy:

- $\varphi(0) = 1,\ \varphi(\infty) = 0,$
- φ is *completely monotone*,
  i.e. $(-1)^k \varphi^{(k)} \geq 0$ for all $k \in \{0, 1, 2, \cdots\}$.

The criteria that φ is completely monotone, i.e. its derivatives change signs, guarantees positiveness of the copula density [Kimberling, 1974]. The class of completely monotone φ coincides with the class of Laplace-Stieltjes transforms (henceforth simply Laplace transforms) of a positive random variable [Bernstein, 1929, Widder, 1941].

Theorem 2 (Bernstein [1929] and Widder [1941])**.** *φ is completely monotone and $\varphi(0) = 1$ if and only if φ is the Laplace transform of a positive random variable,*

$$\varphi(x) = \int_0^\infty e^{-xs} dF_M(s), \tag{5}$$

where $M > 0$ is a positive random variable with Laplace transform φ.

Conversely, a probabilistic construction of the Archimedean copula as a mixture model, with the variables being conditionally independent given a positive latent random variable, leads to the Laplace transform representation for φ. For a given d, φ may come from a broader class of functions than Laplace transforms [McNeil and Nešlehová, 2009]. However, if φ is not a Laplace transform, the simple mixture representation fails [Marshall and Olkin, 1988]. In the mixture representation, the latent variable, depending on its application, is known as a *resilience* or *frailty* parameter [Marshall and Olkin, 1988, Joe, 1997]. Common Archimedean copulas such as the Ali-Mikhail-Haq, Clayton, Frank, Gumbel and Joe copulas can be respectively derived from geometric, gamma, logarithmic, stable, and Sibuya latent distributions. The mixture representation also leads to efficient sampling algorithms [Marshall and Olkin, 1988, McNeil, 2008].

We restate the probabilistic construction and sampling algorithm in the supplementary material.

2.2 HIERARCHICAL ARCHIMEDEAN COPULAS

While Archimedean copulas have been widely employed, the functional symmetry of the Archimedean copula implies exchangeability of the underlying dependence structure, which is sometimes not realistic. Hierarchical (or nested) Archimedean copulas are popular for overcoming this drawback [Joe, 1997].

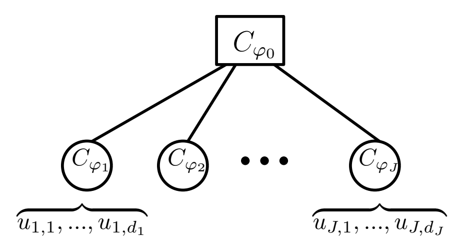


Figure 1: Hierarchical Archimedean copula with J nested, possibly hierarchical, Archimedean copulas.

In this case, the copula can be written as:

$$C(\mathbf{u}) = C_{\varphi_0}(C_{\varphi_1}(\mathbf{u}_1), \cdots, C_{\varphi_J}(\mathbf{u}_J)), \tag{6}$$

where C_{φ_j} are nested, possibly hierarchical, Archimedean copulas with generators φ_j, inputs $\mathbf{u}_j = [u_{j,1}, \cdots, u_{j,d_j}]$, $j \in \{1, \cdots, J\}$, and $\mathbf{u} = [\mathbf{u}_1, \cdots, \mathbf{u}_J]$, $d_1 + \cdots + d_J = d$.

For the above expression to be a valid copula, additional sufficient nesting conditions, derived from the nested mixture representation, first given in [Joe, 1997] and restated in [McNeil, 2008] for nesting to arbitrary depth are:

- φ_j for all $j \in \{0, 1, \cdots, J\}$ are completely monotone,
- $(\varphi_0^{-1} \circ \varphi_j)'$ for $j \in \{1, \cdots, J\}$ are completely monotone.

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 produce a completely monotone Laplace transform *nested generator* of the form $e^{-M\varphi_0^{-1}\circ\varphi_j}$, where M is distributed with Laplace transform φ_0 [Joe, 1997, McNeil, 2008]. This criteria is addressed in [Hering et al., 2010] 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' $t \geq 0$ have the form $e^{-t\psi_j}$, where the Laplace exponent ψ_j has completely monotone derivative. Conversely, a probabilistic construction by combining Lévy subordinators evaluated at common 'time' $t = M$, leads to a well-defined hierarchical Archimedean copula with an efficient sampling algorithm [Hering et al., 2010]. We restate the probabilistic construction from [Hering et al., 2010] in the supplementary material.

Thus for a given *outer generator* φ_0, a *compatible inner generator* φ_j can be modeled as a composition of the outer generator and the Laplace exponent ψ_j of a Lévy subordinator:

$$\varphi_j(x) = (\varphi_0 \circ \psi_j)(x), \tag{7}$$

where the Laplace exponent $\psi_j : [0,\infty) \to [0,\infty)$ of a Lévy subordinator has a convenient representation with drift $\mu_j \geq 0$ and Lévy measure ν_j on $(0,\infty)$ due to the Lévy-Khintchine theorem [Sato, 1999]:

$$\psi_j(x) = \mu_j x + \int_0^\infty (1 - e^{-xs})\nu_j(ds). \tag{8}$$

A popular Lévy subordinator is the compound Poisson process with drift $\mu_j \geq 0$, jump intensity $\beta_j > 0$ and jump size distribution determined by its Laplace transform φ_{M_j}. In this case, the Laplace exponent has the following expression:

$$\begin{aligned}\psi_j(x) &= \mu_j x + \beta_j(1 - \varphi_{M_j}(x)) \qquad (9)\\ &= \mu_j x + \beta_j\left(1 - \int_0^\infty e^{-xs} dF_{M_j}(s)\right),\end{aligned}$$

where $M_j > 0$ is a positive random variable with Laplace transform φ_{M_j} characterizing the jump sizes of the compound Poisson process.

In addition, we choose $\mu_j > 0$ to satisfy the condition $\varphi_j(\infty) = (\varphi_0 \circ \psi_j)(\infty) = 0$ such that φ_j is a valid generator of an Archimedean copula.

3 GENERATIVE ARCHIMEDEAN COPULAS

Motivated by the probabilistic construction of the Archimedean copula, we propose to learn the distribution of the positive latent variable by approximating its Laplace transform using samples from a generative neural network.

3.1 MODELING THE LATENT VARIABLE WITH A GENERATIVE NEURAL NETWORK

We let M be the output of a generative neural network such that samples $M \sim F_M$ are computed as $M = G(\epsilon;\theta)$, where $G(.;\theta)$ represents the generative neural network with parameters θ and ϵ is a source of randomness. Unlike the modeling of monotone functions with neural networks [Chilinski and Silva, 2020], there is no restriction on the weights and intermediate activations of $G(.;\theta)$. In this preliminary work, the network architecture is a multilayer perceptron. To guarantee that M is a positive random variable, we use $\exp(.)$ as the output activation.

We then approximate the Laplace transform with its empirical version using L samples of M from $G(.;\theta)$ as:

$$\varphi(x) = \int_0^\infty e^{-xs} dF_M(s) = \mathbb{E}_M[e^{-Mx}] \approx \frac{1}{L}\sum_{l=1}^{L} e^{-M_l x}. \tag{10}$$

Derivatives of the Laplace transform are similarly approximated with their empirical version as:

$$\varphi^{(k)}(x) = \mathbb{E}_M[(-M)^k e^{-Mx}] \approx \frac{1}{L}\sum_{l=1}^{L}(-M_l)^k e^{-M_l x}. \tag{11}$$

Subsequently, we replace instances of φ and $\varphi^{(k)}$ in the copula distribution, density and sampling algorithm with their sample approximations computed as in (10) and (11).

3.2 GENERATING SAMPLES FROM THE ARCHIMEDEAN COPULA

We modify existing Marshall-Olkin type sampling algorithms [Marshall and Olkin, 1988, McNeil, 2008] to our parameterization with generative neural networks, as detailed in Algorithm 1 and Figure 2, on the next page.

This sampling method is efficient as it only requires sampling unit exponential random variables $E_j \sim \text{Exp}(1),\ j \in \{1, \cdots, d\}$ and a latent random variable $M = G(\epsilon;\theta)$. In addition, unlike the conditional sampling method, this sampling method does not require differentiation of the copula distribution to get the conditional distribution and does not require inversion of the conditional distribution.

3.3 TRAINING METHODS

An important consideration when modeling CDFs is the optimization procedure for fitting the model to data. We describe multiple methods for fitting the model to data with various performance and efficiency trade-offs.

Algorithm 1 Sampling Generative Archimedean Copulas

Input: $G(.;\theta)$,

1: Sample M as $M = G(\epsilon;\theta)$.
2: Sample i.i.d. $E_j \sim \text{Exp}(1), j \in \{1, \cdots, d\}$.
3: Approximate φ with samples $\{M_l\}_{l=1}^L$, where $M_l = G(\epsilon_l;\theta)$, as in (10).
4: Compute $\mathbf{U}$ where $U_j = \varphi(E_j/M), j \in \{1, \cdots, d\}$.

Output: $\mathbf{U}$.

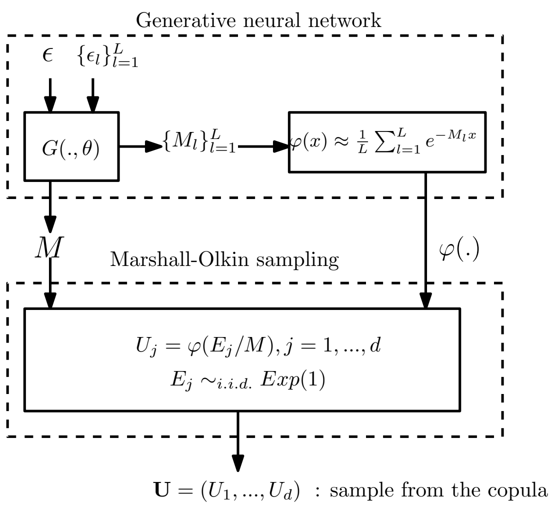


Figure 2: Sampling Archimedean copulas using generative neural networks and Marshall-Olkin type sampling.

3.3.1 Training with Maximum Likelihood

We consider training through maximum likelihood by minimizing the negative log likelihood with backpropagation gradient descent on the model parameters, similar to the proposal in [Ling et al., 2020]. However, since the copula models the CDF, differentiation is required to obtain the copula density. Unlike [Ling et al., 2020] that computes the copula density from the copula distribution using automatic differentiation, we compute the copula density from its analytical expression in (4) using the properties of the Laplace transform for computing higher-order derivatives in (11). For increasing dimensions, computing higher-order derivatives using the Laplace transform representation instead of automatic differentiation leads to a significant speed up in computation.

For the computation of φ^{-1} and its derivative with respect to model parameters, we borrow the method in [Ling et al., 2020]. The inverse is computed using Newton's root-finding method. The derivatives are computed from the derivatives of φ then supplemented to backpropagation.

3.3.2 Training with Goodness-of-Fit

To circumvent computing the copula density, the model may also be fitted to data via minimum distance criterions used in goodness-of-fit tests [Genest et al., 2009]. Though not statistically efficient compared to maximum likelihood estimation, minimum distance estimation is significantly less computationally intensive.

We consider the Cramér-von Mises statistic [Cramér, 1928] to measure a discrepancy between the model copula C_θ and the empirical copula C_N:

$$S_N = \frac{1}{N}\sum_{i=1}^{N}(C_\theta(\mathbf{u}_i) - C_N(\mathbf{u}_i))^2, \tag{12}$$

where $\mathbf{u}_i$ is an observation of the margins, N is the number of observations and C_N is the empirical copula given by:

$$C_N(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\{u_{i,1} \leq u_1, \cdots, u_{i,d} \leq u_d\}. \tag{13}$$

3.3.3 Adversarial Training with Samples

An alternative way to train the model is by minimizing a divergence between true samples from data and fake samples from the model copula, similar to generative adversarial networks (GANs) [Goodfellow et al., 2014]. In this case, we solve the minimax problem in GANs where the generating network must satisfy an Archimedean copula. This is another method that allows training without computing the copula density.

We create a discriminative neural network $D(.;\phi)$ with parameters ϕ and sigmoid$(.)$ output activation to distinguish between true samples from data and fake samples from the copula. We then minimize the Jensen-Shannon loss between true and fake samples as in [Goodfellow et al., 2014]:

$$\min_\theta \max_\phi \mathbf{E}_{U\sim\text{data}}[\log(D(U;\phi))] + \mathbf{E}_{\tilde{U}\sim C}[\log(1 - D(\tilde{U};\phi))], \tag{14}$$

where $\tilde{U} \sim C$ is generated via the sampling method described in Algorithm 1 using the latent random variable represented as the output of the generative neural network $G(.;\theta)$ with parameters θ as discussed in Section 3.1.

4 GENERATIVE HIERARCHICAL ARCHIMEDEAN COPULAS

In the following, we extend the application of generative neural networks to hierarchical Archimedean copulas. We present our results for two levels of hierarchy, but our construction extends to nesting with more levels.

4.1 MODELING THE LAPLACE EXPONENT WITH A GENERATIVE NEURAL NETWORK

For a given outer generator φ_0, the inner generator $\varphi_j, j \in \{1, \cdots, J\}$ is obtained as the composition $\varphi_j = \varphi_0 \circ \psi_j$, as in (7), where ψ_j is the Laplace exponent of a compound Poisson process with Lévy-Khintchine representation, as in (9). We let the drift $\mu_j > 0$ and the jump intensity $\beta_j > 0$ be trainable parameters with $\exp(.)$ output activation. We let the jump size $M_j > 0$ be the output of a generative neural network $G(.;\theta_j)$ with parameters θ_j and $\exp(.)$ output activation. We then compute the Laplace transform φ_{M_j} and its derivatives $\varphi_{M_j}^{(k)}$ using samples from $G(.;\theta_j)$ as in (10) and (11).

4.2 GENERATING SAMPLES FROM THE HIERARCHICAL ARCHIMEDEAN COPULA

We modify the Marshall-Olkin type algorithm given in [Hering et al., 2010] to work with our parameterization using generative neural networks. We first describe sampling of a compound Poisson process in Algorithm 2. We then describe sampling of a generative hierarchical Archimedean copula in Algorithm 3. A sample from the hierarchical Archimedean copula is obtained by combining compound Poisson processes evaluated at a common 'time' $t = M$, where M is the random variable with distribution given by the Laplace transform outer generator φ_0.

Algorithm 2 Sampling compound Poisson process with jump sizes parameterized by generative neural network

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

1: Sample $N_j(t) \sim$ 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 $G(.;\theta_j)$.
3: Compute $\Lambda_j(t) = \mu_j t + \sum_{i=1}^{N_j(t)} M_{j,i}$.

Output: $\Lambda_j(t)$.

4.3 TRAINING WITH GOODNESS-OF-FIT AND MAXIMUM LIKELIHOOD

We first fit the outer generator φ_0, fix it, then fit the inner generators $\varphi_j = \varphi_0 \circ \psi_j$. Fixing the outer generator then optimizing the inner generator provides additional numerical stability during training. In our experiments, the outer generator was trained using minimium distance estimation with the empirical copula based Cramér-von Mises statistic in (12) and empirical copulas on C_{φ_j}. An alternative method may be to train the outer generator using a composite likelihood with bivariate margins since bivariate margins are Archimedean with generator given by the outer generator. The inner generators were trained using maximum likelihood estimation with copula densities c_{φ_j} in (4).

Algorithm 3 Sampling Generative Hierarchical Archimedean Copulas

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

1: Sample $t = M$ from $G(.;\theta_0)$.
2: Approximate φ_0 with samples from $G(.;\theta_0)$, as in (10).

3: **for** $j \in \{1, \cdots, J\}$ **do**
4: Sample $\Lambda_j(M)$, the compound Poisson process at $t = M$, following Algorithm 2.
5: Approximate ψ_j with samples from $G(.;\theta_j)$, as in (10).
6: **end for**

7: Sample i.i.d. $E_{j,i} \sim$ Exp(1), $j \in \{1, \cdots, J\}$, $i \in \{1, \cdots, d_j\}$.
8: Compute $\mathbf{U} = (U_{1,1}, \cdots, U_{J,d_J})$ as $U_{j,i} = (\varphi_0 \circ \psi_j)(E_{j,i}/\Lambda_j(M)), j \in \{1, \cdots, J\}, i \in \{1, \cdots, d_J\}$.

Output: $\mathbf{U}$.

5 EXPERIMENTS

5.1 GENERATIVE ARCHIMEDEAN COPULA

5.1.1 Learning Bivariate Copulas with Different Tail Dependencies and Fitting Real-World Data

Following the experiment setup in [Ling et al., 2020], we consider the Clayton, Frank, and Joe copulas, chosen for their different tail dependencies, and the following real-world data sets: Boston housing, Intel-Microsoft (INTC-MSFT) stocks and Google-Facebook (GOOG-FB) stocks. We applied the three training methods discussed earlier: maximum likelihood, goodness-of-fit and adversarial training. All training methods were implemented in PyTorch and converged within 10k epochs. Experiment details are given in the supplementary material.

The negative log-likelihoods from learning known copulas are reported in Table 1. We use the following shorthands 'GT', 'ACNet', 'MLE', 'CvM', 'GAN' to respectively denote ground truth, ACNet [Ling et al., 2020], and generative Archimedean copulas trained with maximum likelihood, goodness-of-fit and adversarial training. The negative log-likelihoods from fitting real-world data are reported in Table 2, where the log-likelihood of the best-fit single parameter copula (chosen from Clayton, Frank, Joe and Gumbel, as in [Ling et al., 2020]), with shorthand 'BF' is reported in place of the ground truth. The proposed generative Archimedean copulas achieved comparable performance to ACNet in terms of log-likelihood scores. 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 proposed alternative losses.

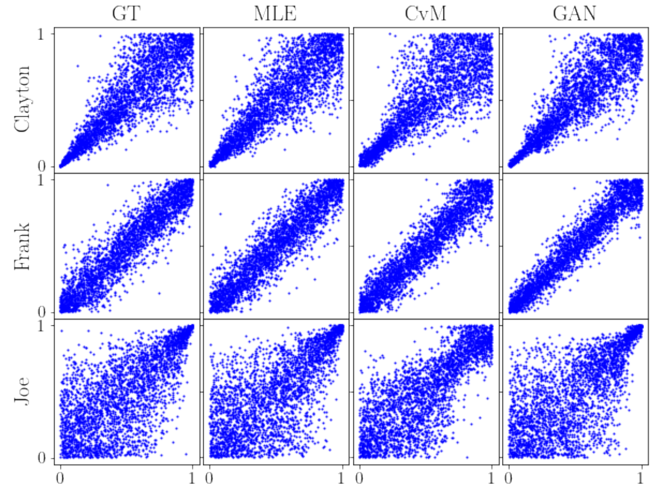
Table 1: Negative log-likelihoods of learning known copulas

	Benchmark		Generative AC		
Dataset	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

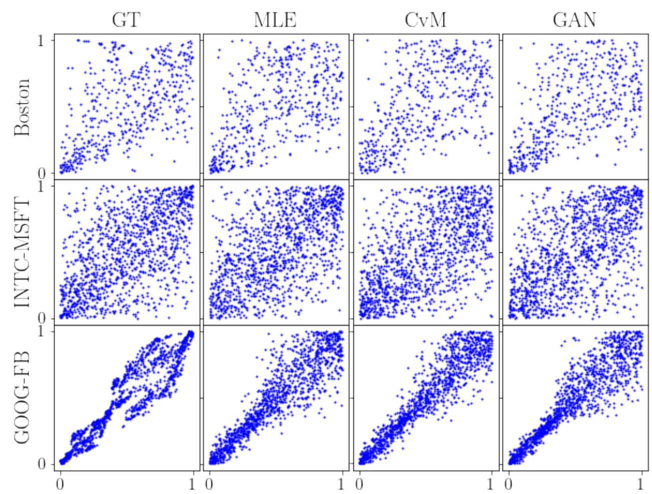
Table 2: Negative log-likelihoods of fitting real-world data

	Benchmark		Generative 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

Samples from the learned copulas are compared to the ground truth in Figure 3. We additionally note the differences in sampling time between our method and the conditional sampling method used in ACNet [Ling et al., 2020]. The time to generate 3000 samples using our method was on average 3.8×10^{-2} seconds. In comparison, the conditional sampling method via automatic differentiation of the copula distribution followed by inversion of the conditional distribution, takes on average $1.98 \times 10^{+2}$ seconds, the difference on the order of 3 magnitudes.



(a)



(b)

Figure 3: Samples from ground truth and learned copulas fitted with maximum likelihood, goodness-of-fit and adversarial training. In (a), the copulas are Clayton, Frank and Joe. In (b), the datasets are Boston housing, Intel-Microsoft stocks and Google-Facebook stocks.

5.1.2 Learning Latent Distributions

The generative neural network was able to learn 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 4.

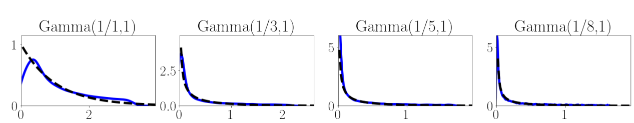


Figure 4: Gamma latent distributions of Clayton copulas with parameters 1, 3, 5, 8, learned in solid blue; ground truth in dashed black.

5.1.3 Learning Higher-Dimensional Copulas

While ACNet faces numerical issues for dimensions $d \geq 5$ due to repeated automatic differentiation when computing the copula density [Ling et al., 2020], the Laplace transform representation allows efficient computation of higher-order derivatives without automatic differentiation.

In addition to the bivariate copulas in Section 5.1.1, we fitted Clayton, Frank and Joe copulas for 10 and 20 dimensions. The negative log-likelihoods are given in Table 3. When compared to the ground truth negative log-likelihoods for 10-dimensional and 20-dimensional datasets, the learned negative log-likelihoods were off by 2%. During our experiments, we could not obtain a reasonably trained ACNet for high dimensions due to the computational complexity.

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 5.

Table 3: Negative log-likelihoods of learning higher-dimensional copulas

Dataset	Ground Truth 10-dim	Ground Truth 20-dim	Generative AC 10-dim	Generative AC 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

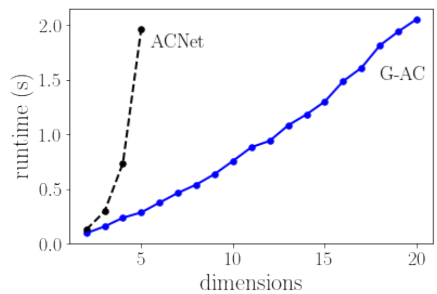
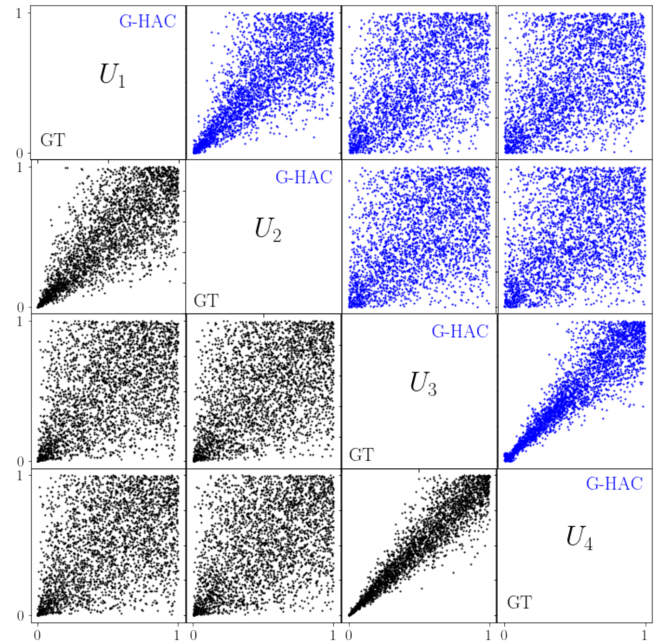


Figure 5: CPU runtimes for computing the likelihoods of 3000 samples from generative Archimedean copula in solid blue; ACNet [Ling et al., 2020] in dashed black.

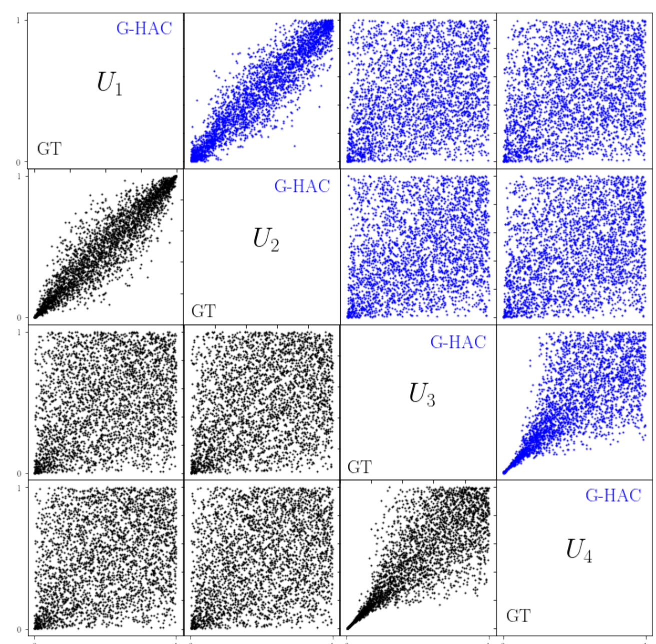
5.2 HIERARCHICAL ARCHIMEDEAN COPULA

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 $C_{\varphi_0}(C_{\varphi_1}(u_1, u_2), C_{\varphi_2}(u_3, u_4))$. The ground truth was generated using the state-of-the-art HA-Copula Toolbox [Górecki et al., 2017]. Samples from the learned copulas are compared to the ground truth in Figure 6. In (a), $C_{\varphi_0}, C_{\varphi_1}, C_{\varphi_2}$ are Clayton copulas with parameters 1, 3, and 8. We let the outer generator be a generative Archimedean copula. In (b), $C_{\varphi_0}, C_{\varphi_1}, C_{\varphi_2}$ are Clayton, '12' and '19' with parameters 0.5, 3, and 1. Since our model is compatible with outer generators of other forms, we let the outer generator be a one-parameter Clayton copula instead of a generative Archimedean copula.



(a)



(b)

Figure 6: Samples, displayed as mirrors on the diagonal, from generative hierarchical Archimedean copulas above in blue and from ground truth below in black. Each plot is a bivariate margin (U_i, U_j). In (a), a homogeneous nested Clayton copula. In (b), a heterogeneous hierarchical Archimedean copula with a Clayton outer generator combined with inner generators '12' and '19', numbering following [Nelsen, 2010, Górecki et al., 2017]

.

6 CONCLUSIONS

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 gave efficient sampling algorithms for sampling from the generative Archimedean and hierarchical Archimedean copulas. We also described three methods for fitting the model to data: maximum likelihood with the copula density, goodness-of-fit with the empirical copula-based Cramér von-Mises statistic and adversarial training by minimizing a divergence between true samples from data and fake samples from the copula. Empirically, the generative Archimedean copula was able to learn known copulas with different tail dependencies and fit real-world data. We also showed an extension to higher-dimensional data using hierarchical Archimedean copulas. Future work includes an end-to-end application such as pairs trading and architecture selection for the generative neural network.

Acknowledgements

This work was supported in part by the Air Force Office of Scientific Research under award number FA9550-20-1-0397.

References

Serge Bernstein. Sur les fonctions absolument monotones. *Acta Mathematica*, 52:1 – 66, 1929. doi: 10.1007/BF02592679. URL https://doi.org/10.1007/BF02592679.

Umberto Cherubini, Elisa Luciano, and Walter Vecchiato. *Copula Methods in Finance*. The Wiley Finance Series. Wiley, 2004. ISBN 978-0-470-86344-2.

Pawel Chilinski and Ricardo Silva. Neural likelihoods via cumulative distribution functions. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 420–429. PMLR, 03–06 Aug 2020. URL http://proceedings.mlr.press/v124/chilinski20a.html.

Harald Cramér. On the composition of elementary errors. *Scandinavian Actuarial Journal*, 1928(1):13–74, 1928. doi: 10.1080/03461238.1928.10416862. URL https://doi.org/10.1080/03461238.1928.10416862.

Sándor Csörgő and Jef L. Teugels. Empirical laplace transform and approximation of compound distributions. *Journal of Applied Probability*, 27(1):88–101, 1990. ISSN 00219002. URL http://www.jstor.org/stable/3214597.

Gal Elidan. Copula bayesian networks. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23, pages 559–567. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/2a79ea27c279e471f4d180b08d62b00a-Paper.pdf.

Christian Genest and Anne-Catherine Favre. Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of Hydrologic Engineering*, 12(4):347–368, 2007. doi: 10.1061/(ASCE)1084-0699(2007)12:4(347). URL https://ascelibrary.org/doi/abs/10.1061/%28ASCE%291084-0699%282007%2912%3A4%28347%29.

Christian Genest, Bruno Remillard, and David Beaudoin. Goodness-of-fit tests for copulas: A review and a power study. *Insurance: Mathematics and Economics*, 44(2):199–213, 2009. URL https://EconPapers.repec.org/RePEc:eee:insuma:v:44:y:2009:i:2:p:199-213.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Jan Górecki, Marius Hofert, and Martin Holeňa. On structure, family and parameter estimation of hierarchical archimedean copulas. *Journal of Statistical Computation and Simulation*, 87(17):3261–3324, 2017. doi: 10.1080/00949655.2017.1365148. URL https://doi.org/10.1080/00949655.2017.1365148.

Norbert Henze, Simos G. Meintanis, and Bruno Ebner. Goodness-of-fit tests for the gamma distribution based on the empirical laplace transform. *Communications in Statistics - Theory and Methods*, 41(9):1543–1556, 2012. doi: 10.1080/03610926.2010.542851. URL https://doi.org/10.1080/03610926.2010.542851.

Christian Hering, Marius Hofert, Jan-Frederik Mai, and Matthias Scherer. Constructing hierarchical archimedean copulas with lévy subordinators. *Journal of Multivariate Analysis*, 101(6):1428–1433, 2010. ISSN 0047-259X. doi: https://doi.org/10.1016/j.jmva.2009.10.005. URL https://www.sciencedirect.com/science/article/pii/S0047259X09001961.

José Miguel Hernández-Lobato and Alberto Suárez. Semiparametric bivariate archimedean copulas. *Computational Statistics & Data Analysis*, 55(6):2038–2058, 2011.

Marius Hofert. Sampling archimedean copulas. *Computational Statistics & Data Analysis*, 52(12):5163–5174, 2008. ISSN 0167-9473.

doi: https://doi.org/10.1016/j.csda.2008.05.019. URL https://www.sciencedirect.com/science/article/pii/S0167947308002910.

Ricardo Hoyos-Argüelles and Luis Nieto-Barajas. A bayesian semiparametric archimedean copula. *Journal of Statistical Planning and Inference*, 206:298–311, 2020. ISSN 0378-3758. doi: https://doi.org/10.1016/j.jspi.2019.09.015. URL https://www.sciencedirect.com/science/article/pii/S0378375819301004.

Jim C. Huang and Brendan J. Frey. Cumulative distribution networks and the derivative-sum-product algorithm: Models and inference for cumulative distribution functions on graphs. *Journal of Machine Learning Research*, 12(10): 301–348, 2011.

Harry Joe. *Multivariate models and dependence concepts*. Chapman and Hall/CRC, May 1997. ISBN 9780367803896. doi: 10.1201/9780367803896. URL https://www.taylorfrancis.com/books/9781466581432.

Harry Joe. *Dependence modeling with copulas*. Chapman and Hall/CRC, June 2014. ISBN 9781466583238. doi: 10.1201/b17116. URL https://www.taylorfrancis.com/books/9781466583238.

Sanket Kamthe, Samuel Assefa, and Marc Deisenroth. Copula flows for synthetic data generation, 2021. URL https://arxiv.org/abs/2101.00598.

Clark H. Kimberling. A probabilistic interpretation of complete monotonicity. *aequationes mathematicae*, 10 (2):152–164, 1974. doi: 10.1007/BF01832852. URL https://doi.org/10.1007/BF01832852.

Rong Qi Liew and Yuan Wu. Pairs trading: A copula approach. *Journal of Derivatives & Hedge Funds*, 19(1): 12–30, 2013. doi: 10.1057/jdhf.2013.1. URL https://doi.org/10.1057/jdhf.2013.1.

Chun Kai Ling, Fei Fang, and J. Zico Kolter. Deep archimedean copulas. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1535–1545. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/10eb6500bd1e4a3704818012a1593cc3-Paper.pdf.

Albert W. Marshall and Ingram Olkin. Families of multivariate distributions. *Journal of the American Statistical Association*, 83(403):834–841, 1988. doi: 10.1080/01621459.1988.10478671. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1988.10478671.

Alexander J. McNeil. Sampling nested archimedean copulas. *Journal of Statistical Computation and Simulation*, 78(6):567–581, 2008. doi: 10.1080/00949650701255834. URL https://doi.org/10.1080/00949650701255834.

Alexander J. McNeil and Johanna Nešlehová. Multivariate Archimedean copulas, d-monotone functions and $l1$-norm symmetric distributions. *The Annals of Statistics*, 37(5B): 3059 – 3097, 2009. doi: 10.1214/07-AOS556. URL https://doi.org/10.1214/07-AOS556.

Roger B. Nelsen. *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 edition, 2010. ISBN 9780387286785 9781441921093. OCLC: 700190717.

Ostap Okhrin, Yarema Okhrin, and Wolfgang Schmid. On the structure and estimation of hierarchical archimedean copulas. *Journal of Econometrics*, 173(2):189–204, 2013. URL https://EconPapers.repec.org/RePEc:eee:econom:v:173:y:2013:i:2:p:189-204.

Ken-iti Sato. *Lévy processes and infinitely divisible distributions*. Number 68 in Cambridge studies in advanced mathematics. Cambridge University Press, 1999. ISBN 9780521553025.

Cornelia Savu and Mark Trede. Hierarchies of archimedean copulas. *Quantitative Finance*, 10(3):295–304, 2010. doi: 10.1080/14697680902821733. URL https://doi.org/10.1080/14697680902821733.

Abe Sklar. Fonctions de répartition à n dimensions et leurs marges. 8:229–231, 1959.

Yolanda Stander, Daniel Marais, and Ilse Botha. Trading strategies with copulas. *Journal of Economic and Financial Sciences*, 6(1):83–107, 2013. doi: 10.10520/EJC135921.

Natasa Tagasovska, Damien Ackerer, and Thibault Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/15e122e839dfdaa7ce969536f94aecf6-Paper.pdf.

David Vernon Widder. *Laplace Transform (PMS-6)*. Princeton University Press, 1941. doi: doi:10.1515/9781400876457. URL https://doi.org/10.1515/9781400876457.

Magnus Wiese, Robert Knobloch, and Ralf Korn. Copula & marginal flows: Disentangling the marginal from its joint, 2019. URL https://arxiv.org/abs/1907.03361.

Andrew G Wilson and Zoubin Ghahramani. Copula processes. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23, pages 2460–2468. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/fc8001f834f6a5f0561080d134d53d29-Paper.pdf.

Kailai Xu and Eric Darve. Calibrating multivariate Lévy processes with neural networks. In Jianfeng Lu and Rachel Ward, editors, *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*, pages 207–220, Princeton University, Princeton, NJ, USA, 20–24 Jul 2020. PMLR. URL http://proceedings.mlr.press/v107/xu20a.html.