Neural Copula: A unified framework for estimating generic high-dimensional Copula functions

Zhi Zeng\textsuperscript{1,2,}\textsuperscript{a}, Ting Wang\textsuperscript{1,\textsuperscript{b}}

\textsuperscript{a}School of Mechano-Electronic Engineering, Xidian University, 2 South Taibai Road, Xian, Shaan Xi, China
\textsuperscript{b}School of Optoelectronic Engineering, Xidian University, 2 South Taibai Road, Xian, Shaan Xi, China

Abstract

The Copula is widely used to describe the relationship between the marginal distribution and joint distribution of random variables. The estimation of high-dimensional Copula is difficult, and most existing solutions rely either on simplified assumptions or on complicating recursive decompositions. Therefore, people still hope to obtain a generic Copula estimation method with both universality and simplicity. To reach this goal, a novel neural network-based method (named Neural Copula) is proposed in this paper. In this method, a hierarchical unsupervised neural network is constructed to estimate the marginal distribution function and the Copula function by solving differential equations. In the training program, constraints are imposed on both the neural network and its derivatives. The Copula estimated by the proposed method is smooth and has an analytic expression. The effectiveness of the proposed method is evaluated on both real-world datasets and complex numerical simulations. Experimental results show that Neural Copula's fitting quality for complex distributions is significantly better than classical methods. The relevant code for the experiments is available on GitHub\textsuperscript{3}. (We encourage the reader to run the program for a better understanding of the proposed method).

Keywords: Copula, Neural network, Probability density function, Cumulative distribution function,

\textsuperscript{1} The authors have contributed equally to the paper. (Email: zhizeng@mail.xidian.edu.cn; tingw2016hg@126.com)
\textsuperscript{2} Corresponding author.
\textsuperscript{3} https://github.com/zengzhi2015/Neural-Copula
Neural Copula

1. Introduction

The Copula function is a general-purpose tool for capturing correlations between random variables, and it constructs a bridge between the joint distribution function and the marginal distribution function [1]. Sklar's theorem shows that the joint distribution of the random variables after transforming by the marginal distribution function can be represented by a Copula function [2]. Since the Copula function can accurately model complex correlations, it is widely used in financial risk management [3], signal processing [4], healthcare [5], healthy diet [6], classification problems [7], image processing [8], etc. [9][10][11].

The estimation of high-dimensional Copula is difficult [12], and most existing solutions rely either on simplified assumptions or on complicating recursive decompositions (such as the vine Copula). When one chooses among parametric [13], semiparametric [14], and nonparametric methods [15][16], a balance must be struck between accuracy and computational complexity. Most Copulas are estimated via maximum likelihood estimation (MLE) [17]. The accuracy relies heavily on the guesses of the Copula family. Once the family selection is wrong it can cause serious estimation errors. For example, in the years before 2008, the collateralized debt obligations (CDO) market relied almost exclusively on Gaussian Copula, a correlation model, for risk management. Gaussian Copula assumes that the correlation is a constant, not a variable. However, the correlation between financial data is known to be unstable. Therefore, the over-reliance on Gaussian Copula leads to the 2008-2009 Global Financial Crisis [18]. Therefore, people have always hoped to obtain a Copula estimation method with both universality and accuracy. In recent years, some scholars have tried to use deep neural networks to approximate the Archimedes Copula function and obtained
valuable results [19]. This convinces us that, using sophisticated neural networks, it is possible to fit more general Copula structures with sufficient accuracy. To the best of our knowledge, there is no neural network-based method to estimate the general high-dimensional Copula function. The essential problem that needs to be solved is how to incorporate the complex constraints as loss functions into the neural network so that convergence can be easily achieved.

To reach this goal, a novel neural network-based method (named Neural Copula) is proposed in this paper. In this method, a hierarchical unsupervised neural network is constructed to estimate the cumulative distribution function (CDF) and the Copula. In the training process, constraints are imposed on both the neural network and its derivatives. The Copula estimated by the proposed method is smooth and has an analytic expression. The effectiveness of the proposed method is evaluated on both real-world datasets and complex numerical simulations.

The rest of the paper is organized as follows. Section II introduces the definition of Copula and Sklar's theorem. Section III introduces the proposed Neural Copula. Numerical experiments are given in Section IV. Section V concludes the paper.

2. The definition of Copula and Sklar's Theorem

This section provides a brief introduction to the definition of Copula Function and Sklar's Theorem.

**Definition 1 (Copula):**

A $d$-dimensional function $C : [0,1]^d \to [0,1]$ is called Copula, when it satisfies the following conditions, and constitutes the CDF whose marginal distribution is uniform [17][20]:

1) If $\exists j \leq d$, makes $u_j = 0$, then there is $C(u) = 0$;

2) If $\exists j \leq d$, makes $\forall i \leq d$ and $i \neq j$, then $u_i = 1$ and $C(u) = u_j$;
3) \( \forall u, v \in [0,1]^d \), if \( \forall i \leq d \) and \( u_i < v_i \), then \( \sum_{w \in S_{u,v}} (-1)^{|w|-1} C(w) \geq 0 \).

According to this definition of Copula, the CDF of any high-dimensional random variable with known marginal distribution can be obtained, which is the famous Sklar's theorem [20].

**Theorem 1 (Sklar's Theorem):**

Suppose there is a \( d \)-dimensional CDF \( F(x) \), and its marginal distributions are denoted as \( \{F_1(x), F_2(x), \ldots, F_d(x)\} \). Then, there exists the Copula function that can describe the relationship between the two.

\[
F(x) = C(u) = C(u_1, u_2, \ldots, u_d) = C(F_1(x_1), F_2(x_2), \ldots, F_d(x_d))
\]

(1)

If both \( F(x) \) and \( C(u) \) are smoothly derivable of infinite order, the probability density function (PDF) can be expressed as

\[
f(x) = c(u) \prod_{i=1}^{d} f_i(x_i)
\]

(2)

where

\[
c(u) = \frac{\partial^d C}{\partial u_1 \cdots \partial u_d}
\]

(3)

### 3. Neural Copula

This section introduces the proposed Neural Copula and shows how to incorporate the complex Copula constraints into the neural network as loss functions that easily lead to convergence. The estimation of Copula is divided into two steps. First, a bunch of fully connected neural networks, named the Marginal model, are used to estimate the CDF. Then, another neural network, named the Copula model, is proposed to approximate the Copula by solving a differential equation.

For the first task, in 2019, Chi-Hua Chen et al. proposed a deep neural model which can be used to estimate the required CDF of the marginal models [21]. Other alternative and improved
techniques can also be found [22][23]. The main structural difference between the proposed method and the above candidates is that we use a more compact neural representation combined with constraints imposed on both the neural network and its derivatives.

For the second task, unlike the case encountered with the Marginal Model, we know from the unique solution conditions of the partial differential equation that, it is not possible to uniquely determine the shape of the CDF if only the distribution of its derivatives and the boundary values of the CDF in higher dimensions. Thus, we first propose an observation added loss to solve this problem.

The reason we did not chose to model the PDF directly, but the CDF is that the neural network can quickly derive the key information for computing the PDF from the CDF model through an automatic derivation mechanism. However, the neural network does not have the ability to integrate out the CDF model quickly and automatically from the PDF model. In other words, if modeling the CDF, one can get the analytic expressions of both the CDF and PDF based on the neural network. However, if only modeling the PDF, one will not be able to get a concise analytic expression for the CDF. Meanwhile, the reason that the method we proposed does not directly sample the CDF before network fitting is that, on the one hand, the sampling of the CDF often involves the calculation of a large amount of statistical information and integration. On the other hand, it was found in the experiments that estimating the PDF by directly fitting the CDF and then deriving the derivative tends to introduce a large amount of noise, which is not conducive to engineering applications.

3.1 The structure of the Marginal Model and Copula Model

First, we use the fully connected network Marginal Model to acquire the CDF value. The model can be expressed as:
Input layer:

\[ \mathbf{h}^0_m = x \in \Omega \subset \mathbb{R} \]  \hspace{1cm} (4)

where \( \Omega \) is the domain of \( x \).

Hidden layer:

\[ \mathbf{h}^{j+1}_m = \tanh \left( \mathbf{w}^j_m \mathbf{h}^j_m + \mathbf{b}^j_m \right) \quad j \in \{0,1,\cdots,L_m-1\} \]  \hspace{1cm} (5)

Output layer:

\[ \hat{F}_m = \text{sigmoid} \left( \mathbf{w}^L_m \mathbf{u}^L_m + \mathbf{b}^L_m \right) \in [0,1] \]  \hspace{1cm} (6)

Taking into account that the neural network has its corresponding parameters (network weights), we denote the output of the Marginal Model as \( \hat{F}_m(x,\theta_m) \), where \( \theta_m \) represents the set of all weights of Marginal Model, that is \( \theta_m = \{\mathbf{w}^j_m\} \cup \{\mathbf{b}^j_m\} \). In accordance with the automatic derivation mechanism of the neural network, the PDF of the marginal distribution can be obtained as follows:

\[ \hat{f}_m(x,\theta_m) = \frac{d\hat{F}_m(x,\theta_m)}{dx} \]  \hspace{1cm} (7)

Secondly, a fully connected neural network, the Copula Model, is proposed to estimate the Copula value. Its framework also consists of three parts:

Input layer:

\[ \mathbf{h}^0_c = \mathbf{u} = [u_1,u_2,\cdots,u_d]^T \in [0,1]^d \]  \hspace{1cm} (8)

Hidden layer:

\[ \mathbf{h}^{j+1}_c = \tanh \left( \mathbf{w}^j_c \mathbf{h}^j_c + \mathbf{b}^j_c \right) \quad j \in \{0,1,\cdots,L_c-1\} \]  \hspace{1cm} (9)

Output layer:

\[ \hat{C} = \text{sigmoid} \left( \mathbf{w}^L_c \mathbf{u}^L_c + \mathbf{b}^L_c \right) \in [0,1] \]  \hspace{1cm} (10)

where \( d \) in the input layer denotes the dimension of \( \mathbf{u} \), where \( u_i \in [0,1] \). The Copula Model has a total of \( L_c-1 \) hidden layers, where \( \mathbf{w}^j_c \) and \( \mathbf{b}^j_c \) represent the weight and bias of the \( j^{th} \)
hidden layer, respectively. The output \( \hat{C}(u) \) is an estimation of the Copula function. Since Copula Model contains parameters, we denote it as \( \hat{C}(u, \theta) \), where \( \theta \) represents the set of all weights of the Copula Model, i.e. \( \theta = \{w_c\} \cup \{b_c\} \). Based on the automatic derivation mechanism, the PDF can be obtained as

\[
\hat{c}(u, \theta) = \frac{\partial^2 \hat{C}(u, \theta)}{\partial u_1 \partial u_2 \ldots \partial u_d}
\]

Hence, the PDF of the sampled data can be expressed as

\[
\hat{f}(x, \{\theta_m\}, \theta) = \hat{c}(\hat{F}_1(x, \theta_m^1), \hat{F}_2(x, \theta_m^2), \ldots, \hat{F}_d(x, \theta_m^d), \theta) \prod_{i=1}^{d} f_i(x, \theta_m^i)
\]

3.2 The loss function for the Marginal Model

The proposed Marginal Model's loss function is composed of four parts:

1) The fitness of \( \hat{f}_m(x, \theta_m) \) to the sampled data can be expressed in terms of log-loss in the maximum likelihood estimation. That is

\[
L_m^1(\theta_m) = \frac{1}{n_m^1} \sum_{i=1}^{n_m^1} \left[ -\log \hat{f}_m(x, \theta_m^i) \right]
\]

The set of all training samples corresponding to this loss is denoted as \( D_m = \{x_i\}_{i=1,2,\ldots,n_m} \). The total number of samples in the training set is \( n_m \).

2) \( \hat{f}_m(x, \theta_m) \) must be non-negative, so the penalty must be given for the negative part of \( \hat{f}_m(x, \theta_m) \), i.e.

\[
L_m^2(\theta_m) = \int_{x_m \in \Omega_m} \text{relu}(-\hat{f}_m(x, \theta_m)) dx_m
\]

where \( \Omega_m \) is the domain of \( x_m \). Practically, the estimation of \( L_m^2(\theta_m) \) can be calculated by

\[
L_m^2(\theta_m) = \frac{1}{n_m^1} \sum_{i=1}^{n_m^1} \text{relu}(-\hat{f}_m(x, \theta_m^i))
\]
The set of all training samples corresponding to this loss is denoted as \( D_m^2 = \{ x_i \}_{i=1,2,\ldots,n_m^2} \). Without loss of generality, the positions of the samples can be uniformly distributed over the domain.

3) The integral of \( \hat{f}_m(x, \theta_m) \) over the universe must be 1, so the corresponding loss function is as follows.

\[
L_m^3(\theta_m) = \left| 1 - \int_{x_m \in \Omega_m} \hat{f}_m(x, \theta_m) dx_m \right|
\]

where \( \Omega_m \) is the domain of \( x_m \).

Practically, the estimation of \( L_m^3(\theta_m) \) can be calculated by

\[
L_m^3(\theta_m) \approx \left| 1 - \frac{1}{n_m^2} \sum_{i=1}^{n_m^2} \hat{f}_m(x_i, \theta_m) \Delta_m \right|
\]

The set of all training samples corresponding to this loss is denoted as \( D_m^3 = \{ x_i \}_{i=1,2,\ldots,n_m^2} \). Without loss of generality, the positions of the samples in this dataset can be uniformly distributed over its domain, and the interval of adjacent data is denoted as \( \Delta_m \).

4) The CDF \( \hat{F}_m(x, \theta_m) \) must satisfy the following conditions

\[
\begin{align*}
\hat{F}_m(0, \theta_m) &= 0 \\
\hat{F}_m(1, \theta_m) &= 1
\end{align*}
\]

The corresponding loss function can be expressed as

\[
L_m^4(\theta_m) = \hat{F}_m(0, \theta_m) + \left| 1 - \hat{F}_m(1, \theta_m) \right|
\]

The set of all training samples corresponding to this loss is \( D_m^4 = \{ 0, 1 \} \).

In conclusion, the loss functions of the Marginal Model can be formulated by a linear combination as

\[
L_m(\theta_m) = \sum_{k=1}^{4} \lambda_k L_k(\theta_m)
\]

3.3 The loss function for the Copula Model
Similar to the Marginal Model, the constraint (loss function) acting on the Copula Model consists of five components:

1) The fitness of $\hat{f}_c \left( x, \{ \theta_m^i \}, \theta_z \right)$ to the data distribution can be represented by the log loss in the maximum likelihood estimation. That is

$$L_c^1 \left( \{ \theta_m^i \}, \theta_z \right) = \frac{1}{n_c^i} \sum_{i=1}^{n_c^i} \left[ \log \hat{f}_c \left( x, \{ \theta_m^i \}, \theta_z \right) \right]$$  \hspace{1cm} (22)

We denote the set of all training samples corresponding to this loss as $D_c^1 = \{ x_i \}_{i=1,2,\ldots,n_c^i}$.

2) $\hat{f}_c \left( x, \{ \theta_m^i \}, \theta_z \right)$ must be non-negative, thus the penalty must be given for the negative part of $\hat{f}_c \left( x, \{ \theta_m^i \}, \theta_z \right)$, i.e

$$L_c^2 \left( \{ \theta_m^i \}, \theta_z \right) = \int \int \cdots \int \text{relu} \left( -\hat{f}_c \left( x, \{ \theta_m^i \}, \theta_z \right) \right) dx_1 dx_2 \cdots dx_d$$  \hspace{1cm} (23)

To simplify the calculation, we estimate the numerical approximation of the above loss function by

$$L_c^2 \left( \{ \theta_m^i \}, \theta_z \right) = \frac{1}{n_c^i} \sum_{i=1}^{n_c^i} \text{relu} \left( -\hat{f}_c \left( x_i, \{ \theta_m^i \}, \theta_z \right) \right)$$  \hspace{1cm} (24)

The set of all training samples corresponding to this loss is denoted as $D_c^2 = \{ x_i \}_{i=1,2,\ldots,n_c^i}$. Without loss of generality, the positions of the samples of this dataset are uniformly distributed over its domain.

3) The integration of $\hat{f}_c \left( x, \{ \theta_m^i \}, \theta_z \right)$ over the universe must be 1, so the corresponding loss function is defined as follows:

$$L_c^3 \left( \{ \theta_m^i \}, \theta_z \right) = \left| 1 - \int \int \cdots \int \hat{f}_c \left( x, \{ \theta_m^i \}, \theta_z \right) dx_1 dx_2 \cdots dx_d \right|$$  \hspace{1cm} (25)

Practically, the estimation of $L_c^3 \left( \{ \theta_m^i \}, \theta_z \right)$ can be calculated by

$$L_c^3 \left( \{ \theta_m^i \}, \theta_z \right) \approx \left| 1 - \frac{1}{n_c^i} \sum_{j=1}^{n_c^i} \hat{f}_c \left( x_j, \{ \theta_m^i \}, \theta_z \right) \right|$$  \hspace{1cm} (26)
The set of all training samples corresponding to this loss is denoted as $D_i^3 = \{x_i\}_{j=1,2,\ldots,n_i^3}$. The positions of the samples of this dataset are uniformly distributed over its domain, and the interval of adjacent data along the $j^{th}$ dimension is denoted as $\Delta_j$.

4) The distribution of $\hat{C}(\mathbf{u}, \theta)$ on the boundary must satisfy Copula's definition, that is

$$
\begin{align*}
\hat{C}(\mathbf{u}, \theta) &= \hat{C}(0,u_2,\ldots,u_d, \theta) = 0 \\
\hat{C}(\mathbf{u}, \theta) &= \hat{C}(u_1,0,\ldots,u_d, \theta) = 0, \quad u_i \in [0,1] \\
\hat{C}(\mathbf{u}, \theta) &= \hat{C}(u_1,u_2,\ldots,0, \theta) = 0 \\
\hat{C}(\mathbf{u}, \theta) &= \hat{C}(u_1,1,\ldots,1, \theta) = u_i \\
\hat{C}(\mathbf{u}, \theta) &= \hat{C}(1,u_2,\ldots,1, \theta) = u_i, \quad u_i \in [0,1] \\
\hat{C}(\mathbf{u}, \theta) &= \hat{C}(1,\ldots,u_d, \theta) = u_d
\end{align*}
$$

The corresponding loss function can be stated as

$$L_i^4(\hat{\mathbf{u}}, \theta) = \sum_{i=1}^{n} \sum_{j=1}^{n} |\hat{F}_i(\mathbf{u}_j, \theta)| = \sum_{i=1}^{n} \sum_{j=1}^{n} |\hat{F}_i(\mathbf{u}_j, \theta)| - u_i$$

We denote the set of all training samples corresponding to this loss as $D_i^4 = \bigcup_{j=1}^{d} \{\mathbf{u}_j\}_{j=1,2,\ldots,n_i^d} \cup \{\mathbf{u}_j\}_{j=1,2,\ldots,n_i^d}$. Without loss of generality, the positions of the samples of this dataset are randomly distributed in the boundary region.

5) Unlike the case encountered with the Marginal Model, we know from the unique solution conditions of the partial differential equation that, it is not possible to uniquely determine the shape of the CDF if only the distribution of its derivatives and the boundary values of the CDF in higher dimensions ($d \geq 3$) are given. For this reason, certain observation values of the CDF must be added to ensure the uniqueness of the estimation. It should be noted that it is usually time-consuming to sample from the CDF, so it is not recommended to calculate an excessive number of observations and then fit it through the network. The values of $\hat{F}_i(x, \theta) \hat{C}(\mathbf{x}, \theta)$ at certain observation points are consistent with the statistical law, that is
\[\hat{F}_c\left(x, \left\{\theta'_n\right\}, \theta_c\right) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} \hat{F}_c\left(y, \left\{\theta'_n\right\}, \theta_c\right) dy_1 dy_2 \cdots dy_d\]  

(30)

Eq. (30) can be approximated by

\[\hat{F}_c\left(x, \left\{\theta'_n\right\}, \theta_c\right) \approx \frac{1}{n_x} \sum_{y \in D} \text{flag}(x, y)\]  

(31)

where

\[\text{flag}(x, y) = \begin{cases} 1 & \forall j \leq d, y_j < x_j \\ 0 & \text{otherwise} \end{cases}\]  

(32)

The loss function corresponding to the above constraints can be formulated as

\[L_c\left(\left\{\theta'_n\right\}, \theta_c\right) = \frac{1}{n_x n_y} \sum_{i=1}^{n_x} \left| \hat{F}_c\left(x, \left\{\theta'_n\right\}, \theta_c\right) - \sum_{y \in D} \text{flag}(x, y) \right|\]  

(33)

We denote the set of all training samples corresponding to this loss as \(D^c = \left\{x_i\right\}_{i=1,2,\ldots,n_c}\). Without loss of generality, the samples of this dataset are randomly distributed in the domain.

Finally, we combine all the above loss functions of the Copula Model using the linear combination. That is

\[L_c\left(\left\{\theta'_n\right\}, \theta_c\right) = \sum_{k=1}^{N} \lambda_k L_k\left(\left\{\theta'_n\right\}, \theta_c\right)\]  

(34)

### 3.4 Summary

This section summarizes Neural Copula's neural network architecture and its training framework in several schematic images.

The framework of the Neural Copula is shown in Fig. 1, which consists of several Marginal Models (the number of which depends on the dimensions of the input data) and a Copula Model. Each channel (dimension) of the input data is fed into its corresponding Marginal Model to estimate the CDF value, and then all CDF values are fed into the Copula Model to obtain the Copula value.
As illustrated in Fig. 2, (A) and (B) are the training schemes for the Marginal Model and Copula Model, respectively. Neural Copula can be trained using an end-to-end way. The Marginal Model is trained by feeding four training sets in parallel and optimizing the parameters via back-propagation. The Copula model is trained by feeding five training sets in parallel and optimizing the parameters via back-propagation. As shown in Fig. 2 (B), when updating the Copula Model, the parameters of the trained Marginal Models are fixed.

**Figure 1.** The framework for Neural Copula and its submodules: the Marginal Model and Copula Model. The elements in the purple boxes indicate the input and output data. ‘split’ means the samples are separated dimension by dimension. ⊗ represents the multiplication operation. The gray arrows indicate the data flow.

As illustrated in Fig. 2, (A) and (B) are the training schemes for the Marginal Model and Copula Model, respectively. Neural Copula can be trained using an end-to-end way. The Marginal Model is trained by feeding four training sets in parallel and optimizing the parameters via back-propagation. The Copula model is trained by feeding five training sets in parallel and optimizing the parameters via back-propagation. As shown in Fig. 2 (B), when updating the Copula Model, the parameters of the trained Marginal Models are fixed.
Figure 2. The training schemes for the Marginal Model (A) and Copula Model (B). The elements in the purple boxes indicate the input and output data. The contents in the light green box are the loss functions. The colored arrows denote that the data (marked on the arrows) should be input in parallel. The gray arrows indicate the data flow. The blue dashed arrows indicate back propagation. The black dashed box of the Marginal Model in (B) indicates that the weights of the Marginal Model are fixed when the Copula Model is training. 'split' means the samples are separated dimension by dimension. × represents the multiplication operation.

Figure 3 illustrates the training results of a Neural Copula for fitting samples generated by a two-dimensional Gaussian function. The estimated marginal distributions by Marginal Models and estimated CDF and PDF values by the Neural Copula are given. The generated samples according to the estimated PDF are also shown, which is to demonstrate more intuitively the accuracy of the proposed method.
4. Experiments

With no loss of generality, this chapter demonstrates the effectiveness of Neural Copula through 2D and 3D examples. Although our proposed method can also be used for higher-dimensional data, it will make the comparison and visualization of algorithm performance more difficult. For this reason, complex 2D and 3D cases are deliberately chosen for this experiment to show the generalizability, robustness, and accuracy of the algorithm.

4.1 Real-world Datasets

To evaluate the effectiveness of Neural Copula in estimating generic Copula functions, the performance of Neural Copula has been compared with three classical parametric Copula models on three real-world datasets.

The Real-world dataset used in this paper is the same as [19]. Three real-world datasets are Boston Housing, INTC-MSFT, and GOOG-FB. The data in these datasets have a limited distribution area, and the data in two dimensions are correlated. This real-world data generally has only a small

![Figure 3. The training results of the Neural Copula for fitting samples generated by a two-dimensional Gaussian function. (A) Scatterplot of the samples. (B) and (E) are the marginal distributions. (C) and (F) are sampled boundary points and observation points, respectively. (D) and (G) are the estimated CDF and PDF by Neural Copula, respectively. (H) is the scatterplot of generated samples from Neural Copula.](image)
number of outliers far from the distribution. Before training the network, the data in each dimension is Min-Max normalized to the range [0, 1]. This ensures that the empirical marginals are approximately uniform [19]. For real-world data, the overfitting is mild or even negligible. Therefore, the data is only divided into training and test sets, which account for 2/3 and 1/3 of the total data, respectively.

In this experiment, the sample numbers of each training set are as follows: the total number of samples in the training set is $n^1_m$, $n^2_m = n^3_m = 1000$. $n^4_c$ is equal to the total number of samples in the training set, $n^5_c = n^6_c = 2500$, $n^7_c = 800$, $n^8_c = 200$. The Marginal Model for the estimation of the CDF values is structured by a 1-5-5-5-5-5-1 fully connected network with an L2 regularization rate of 0.001 for each layer. Other settings include $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\lambda_3 = 2$, $\lambda_4 = 2$. The Adam optimizer is used with a learning rate of 0.001. The Copula Model is structured by using a 2-10-10-10-10-10-10-1 fully connected network with an L2 regularization rate of 0.001 for each layer. Other settings include $\lambda_1 = 0.1$, $\lambda_2 = 1$, $\lambda_3 = 1$, $\lambda_4 = 2$, $\lambda_5 = 5$. The Adam optimizer is used with a learning rate of 0.0001. The following describes each of the three datasets used in this paper.

**Boston Housing:** In the Boston Housing dataset, there is a negative correlation between the percentage of the population with lower status and the median value of owner-occupied homes, but they are not linearly related. This dataset is from Kaggle and has 506 samples.

**INTC-MSFT:** The daily opening prices of Intel Corporation (INTC) and Microsoft (MSFT) have an approximate positive correlation. Data close to the origin of the axes show a strong positive correlation. Data far from the origin of the axes are weakly correlated, and they are more scattered with a distinct bifurcation. The INTC-MSFT dataset is derived from Yahoo Finance and has a total of 1263 samples. The INTC-MSFT dataset is the daily opening prices for INTC and MSFT from
There is a strong positive correlation between the daily opening prices of Alphabet Inc. (GOOG) and Facebook (FB). The GOOG_FB dataset has a dense data distribution and two distinct clustering centers. This dataset is collected by Yahoo Finance and contains 1259 daily closing prices from May 2015 to May 2020.

We compared the performance of Neural Copula with t-Copula as the one used in [24], Frank Copula as the one used in [25], and Gaussian Copula as the one used in [26]. These three models are the most classical methods for estimating Copula and are widely used in many fields such as finance and signal processing. The performance of the four methods is measured through qualitative and quantitative comparisons. The estimation accuracy of the models is compared qualitatively based on how well the marginal and joint distributions fit the shape of the distribution of the samples; The quantitative analysis reports the log-loss of Neural Copula and the three classical Copula models on the testing dataset. The above comparisons reveal that Neural Copula can obtain an estimation accuracy no less than that of three classical methods.

4.1.1 Experiments on the Boston Housing dataset

For the Boston Housing dataset, as shown in the first row of Fig. 4, Neural Copula can obtain marginal distribution comparable to the three classical Copula models. As shown in the second row of Fig. 4, the joint distribution yielded by Neural Copula is also close to that of the three classical Copula models. Visually, the estimations from Neural Copula seem more similar to the data distribution as shown in Fig. 4 (A). It can be seen from Table 1 that the log loss of Neural Copula on the test set is higher than that of the other three classical Copula models. (The larger the log-loss is, the more accurate the estimation is.)
Table 1. Test log-loss of Neural Copula, t-Copula, Frank Copula, and Gaussian Copula on the Boston Housing dataset.

|                  | Neural Copula | t-Copula | Frank Copula | Gaussian Copula |
|------------------|---------------|----------|--------------|----------------|
|                  | 1.269         | 1.242    | 1.251        | 1.229          |

4.1.2 Experiments on the INTC-MSFT dataset

For the INTC-MSFT dataset, as shown in the second row of Fig. 5, the joint distribution of Neural Copula is more similar to Fig. 5 (A) and much better than the three classical models. The marginal distribution of the joint distribution of Neural Copula is also closer to the original distribution than the three classical methods. For the INTC-MSFT dataset, compared with the three classical Coupla models, Neural Copula is able to model not only the data with higher dependence in the bottom left but also the data with lower dependence in the upper right. As can be seen from Table 2, the log loss of Neural Copula on the test set is considerably higher than that of the three classical Coupla models, which quantitatively indicates that Neural Copula has high estimation accuracy.
Table 2. Test log loss of Neural Copula, t-Copula, Frank Copula, and Gaussian Copula on the INTC-MSFT dataset.

|            | Neural Copula | t-Copula | Frank Copula | Gaussian Copula |
|------------|---------------|----------|--------------|-----------------|
| Log Loss   | 1.729         | 0.917    | 0.992        | 0.871           |

4.1.3 Experiments on the GOOG-FB dataset

For the GOOG-FB dataset, as shown in the second row of Fig. 6, the joint distribution of Neural Copula is far better than that of the three classical models, and more similar to Fig. 6 (A). Meanwhile, the marginal distribution of the joint distribution of Neural Copula is closer to the original distribution in Fig. 6 (A) than the three classical methods. As shown in the second row of Fig. 6, Neural Copula has an exceptional ability to learn the characteristics that the original distribution has two clustering centers, while the joint distribution of the three classical Copula models only has one clustering center. In Table 2, the log loss of Neural Copula on the test set is significantly higher than that of the three classical Copula models.
Table 3. Test log-loss of Neural Copula, t-Copula, Frank Copula, and Gaussian Copula on the GOOG-FB dataset.

|                  | Neural Copula | t-Copula | Frank Copula | Gaussian Copula |
|------------------|--------------|----------|--------------|-----------------|
| **1.509**        | 1.151        | 1.056    | 1.144        |

In summary, for the above three real-world datasets, Neural Copula qualitatively and quantitatively outperforms t-Copula, Frank Copula, and Gaussian Copula.

The CPU of the computing platform used in this paper is Intel Core i9 9900X, the GPU is NVIDIA Titan RTX, and the memory is 128 GB. In terms of computational efficiency, Neural Copula takes approximately 290s to train 60,000 epochs, which can be significantly reduced if transfer learning is used, which is suitable for some applications with higher demand for better accuracy but less demanding real-time requirements.

4.2 Complex simulation datasets

For many applications, the distribution of the data is complex and has multiple clustering centers, which is not easy or even impossible to estimate by traditional Copula models. Therefore, to further evaluate the performance of the Neural Copula model, it is tested on two complex
simulation datasets. A comparison of the performance of the Neural Copula with the other three classical parametric Copula models shows that the Neural Copula has a significant advantage in terms of fitting accuracy.

The following 2D and 3D complex simulation datasets are generated by known PDF, and consist of 2601 and 9261 samples, respectively. The distributions of the two datasets contain distinct bar-shaped (or block-shaped) clusters. Furthermore, they are far more complicated than those in previous experiments.

4.2.1 Two-dimensional problem

The 2601 samples in the 2D complex simulation dataset are generated randomly by the following PDF.

$$f(x,y) = \frac{1}{a} \left[ \sin(x+ky)+1 \right] \quad x \in [0,3], \quad y \in [0,2]$$

where $k = 5$, $a = 5.835386372$.

Figure 7. Two-dimensional Copula to be estimated. A. The CDF $F(x,y)$; B. The PDF $f(x,y)$; C. Sampling results according to $f(x,y)$; D. Copula function $C(u,v)$; E. The PDF $c(u,v)$; F. Sampling results according to $c(u,v)$.
The training results of the Marginal Model and Copula Model are shown in Fig. 8 (A) and Fig. 8 (B), respectively.

From the first subplot on the left of Fig. 8 (B), it can be seen that the probability distribution of the Copula function has two obvious bar-shaped clusters, which makes it difficult to fit through three classical models. As can be seen in Fig 8 (B), these three classical models incorrectly capture the pattern of the distribution of the data on the u-v space. Consequently, the fitting error is relatively large. In contrast, Neural Copula is able to accurately approximate the distribution of Copula due to the precise constraints put on its derivatives. As can be seen from the error distribution in Fig 8, the average fitting errors of Neural Copula for the Copula values and the PDF values of the Copula function are no more than 0.012 and 0.113, respectively, which can satisfy most engineering needs.
applications with less demanding requirements for the real-time performance.

4.2.2 Three-dimensional problem

The 9261 samples in the 3D complex simulation dataset are generated randomly by the following PDF.

\[
f(x,y,z) = \frac{4.843}{\sin(x + ky + lz) + 1} \quad x \in [0,3], y \in [0,2]
\]  

(36)

where \( k = 2 \), \( l = 3 \).

Figure 9. Three-dimensional Copula to be estimated. A. The CDF \( F(x,y,z) \); B. The PDF \( f(x,y,z) \); C. Sampling results according to \( f(x,y,z) \); D. Copula \( C(u,v,w) \); E. The PDF \( c(u,v,w) \); F. Sampling results according to \( c(u,v,w) \).
As shown in Fig. 10 (A), the 3D samples of the probability distribution of the Copula function have two distinct clusters and a large space between the two clustering centers. Neural Copula can accurately fit the probability density function and, at the same time, can exactly approximate the Copula function. As can be seen in Fig. 10 (C) and (F), the average fitting error of Neural Copula for the Copula values and the PDF values is considerably small. These two errors are 0.039 and 0.149, respectively, which can satisfy most engineering application scenarios.

5. Conclusion

In this paper, we propose a neural network-based framework (Neural Copula) to estimate generic Copula functions. The method uses the Marginal Model to estimate marginal CDFs and uses the Copula Model to approximate the Copula function. The Copula function obtained by this method has good properties of analytic expression and smooth differentiability. The effectiveness of the
The proposed method is evaluated on both real-world datasets and complex numerical simulations.

**CRediT authorship contribution statement**

**Zhi Zeng:** Conceptualization, Methodology, Formal analysis, Writing; **Ting Wang:** Writing-review & editing, Validation, Experiments.

**Declaration of Competing Interest**

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Acknowledgments**

This research is supported by the financial support from the National Natural Science Foundation of China under No. 61805185.

**Reference**

[1] E. Bouyé, V. Durrleman, A. Nikeghbali, G. Riboulet, T. Roncalli, Copulas for Finance - A Reading Guide and Some Applications, SSRN Electron. J. 1032533 (2000) 1-69.

[2] L. Rüschendorf, On the distributional transform, Sklar’s theorem, and the empirical copula process, J. Stat. Plan. Inference. 139 (2009) 3921–3927.

[3] J.C. Rodriguez, Measuring financial contagion: A Copula approach, J. Empir. Financ. 14 (2007) 401–423.

[4] X. Zeng, J. Ren, Z. Wang, S. Marshall, T. Durrani, Copulas for statistical signal processing (Part I): Extensions and generalization, Signal Processing. 94 (2014) 691–702.

[5] R. Winkelmann, Copula bivariate probit models: With an application to medical expenditures, Heal. Econ. (United Kingdom). 21 (2012) 1444–1455.

[6] F.M.L. Di Lascio, M. Disegna, A copula-based clustering algorithm to analyse EU country diets, Knowledge-Based Syst. 132 (2017) 72–84.

[7] G. Lucca, J.A. Sanz, G.P. Dimuro, B. Bedregal, M.J. Asiain, M. Elkano, H. Bustince, CC-integrals: Choquet-like Copula-based aggregation functions and its application in fuzzy rule-based classification systems, Knowledge-Based Syst. 119 (2017) 32–43.

[8] X. Wang, R. Song, Z. Mu, C. Song, An image NSCT-HMT model based on copula entropy multivariate Gaussian scale mixtures, Knowledge-Based Syst. 193 (2020) 105387.

[9] B. Storvik, G. Storvik, R. Fjørtoft, On the combination of multisensor data using meta-Gaussian distributions, IEEE Trans. Geosci. Remote Sens. 47 (2009) 2372–2379.

[10] G. Mercier, G. Moser, S.B. Serpico, Conditional copulas for change detection in heterogeneous remote sensing images, IEEE Trans. Geosci. Remote Sens. 46 (2008) 1428–1441.

[11] F. Flitti, C. Collet, A. Joannic-Chardin, Unsupervised multiband image segmentation using Hidden Markov quadtree and copulas, in: Proc. - Int. Conf. Image Process. ICIP, 2005: pp. 631–634.

[12] A.K. Nikoloulopoulos, Efficient and feasible inference for high-dimensional normal copula
regression models, Comput. Stat. Data Anal. 1 (2022) 107654.

[13] S.G. Iyengar, P.K. Varshney, T. Damarla, A parametric copula-based framework for hypothesis testing using heterogeneous data, IEEE Trans. Signal Process. 59 (2011) 2308–2319.

[14] X. Chen, Y. Fan, Estimation and model selection of semiparametric copula-based multivariate dynamic models under copula misspecification, J. Econom. 135 (2006) 125–154.

[15] S.X. Chen, T.M. Huang, Nonparametric estimation of copula functions for dependence modelling, Can. J. Stat. 35 (2007) 265–282.

[16] Y. Wang, H. Ni, S. Wang, Nonparametric bivariate copula estimation based on shape-restricted support vector regression, Knowledge-Based Syst. 35 (2012) 235–244.

[17] N. Kolev, D. Paiva, Copula-based regression models: A survey, J. Stat. Plan. Inference. 139 (2009) 3847–3856.

[18] F. Salmon, The formula that killed Wall Street, Significance. 9 (2012) 16–20.

[19] C.K. Ling, F. Fang, J.Z. Kolter, Deep archimedean copulas, in: Adv. Neural Inf. Process. Syst., 2020.

[20] B. Schweizer, A. Sklar, E. Thorp, The metrization of statistical metric spaces, Pacific J. Math. 10 (1960) 673–675.

[21] C.H. Chen, F. Song, F.J. Hwang, L. Wu, A probability density function generator based on neural networks, Phys. A Stat. Mech. Its Appl. 541 (2020) 123344.

[22] D.E.W. Peerlings, J.A. van den Brakel, N. Basturk, M.J.H. Puts, Multivariate Density Estimation by Neural Networks, IEEE Trans. Neural Networks Learn. Syst. (2022) 1–12.

[23] P. Chilinski, R. Silva, Neural likelihoods via cumulative distribution functions, in: Proc. 36th Conf. Uncertain. Artif. Intell. UAI 2020, 2020: pp. 420–429.

[24] Y. Huang, K. Guo, X. Yi, J. Yu, Z. Shen, T. Li, T-copula and Wasserstein distance-based stochastic neighbor embedding, Knowledge-Based Syst. 243 (2022) 108431.

[25] B. De Baets, H. De Meyer, On a conjecture about the Frank copula family, Fuzzy Sets Syst. 228 (2013) 15–28.

[26] H. Liu, F. Han, M. Yuan, J. Lafferty, L. Wasserman, High-dimensional semiparametric Gaussian copula graphical models, Ann. Stat. 40 (2013) 2293-2326.