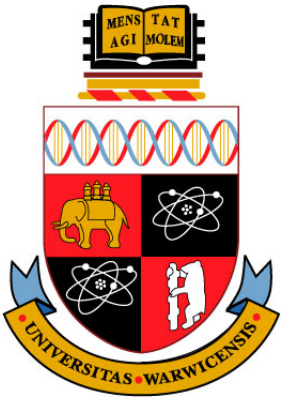


Probabilistic Weather Forecasting with Recursive Algorithms and Vine Copulas

Kuan-Hung Chen
(Warwick ID: 5612597)

Supervisor: Dr Ritabrata Dutta
Department of Statistics
University of Warwick
15 September 2025

A dissertation submitted in partial fulfillment of the requirements for the degree of MSc in Statistics at the University of Warwick

Abstract

Probabilistic weather forecasting has long been a topic of academic interest. Mainstream methods, such as numerical ensemble simulations and Generative Neural Networks, often require a long time to run and demand a huge amount of data to train. This dissertation explores the possibility of applying an efficient nonparametric recursive marginal density estimation method to dependent data and utilising vine copulas to estimate complex nonlinear temporal dependence structures in time series. A simple and efficient autoregression method that takes the two above estimates as input is introduced to produce predictive distributions for future values. We also incorporate parameter optimisation via scoring rule minimisation into the training process to further enhance performance. This prediction framework is not only computationally efficient but also allows for partial real-time updates of estimates as new information is received. We test the prediction framework on several synthetic and real-life weather datasets to demonstrate its capability and potential for further study and improvement.

Keywords: Vine Copulas, Probabilistic Forecasting, Recursive Estimation, Scoring Rules

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr Ritabrata Dutta for his constant support and invaluable guidance throughout this research project. On a personal note, I extend my heartfelt thanks to my family for their care, support, and encouragement over the past year. Writing this dissertation would not have been such a rewarding and enjoyable journey without all of them.

Contents

Chapter 1

Introduction

When providing predictions for future events, it is equally important that the forecaster quantify and report their uncertainty about the predictions. This is the spirit of probabilistic forecasting (Gneiting and Katzfuss, 2014). When making probabilistic forecasts, instead of quoting a single value, called a point forecast, the forecaster has to provide a probability distribution that includes all possible outcomes and their corresponding probabilities. A good probabilistic forecaster aims to maximise the sharpness (how concentrated the distribution is) of the distribution, while at the same time ensuring calibration (the discrepancy between the predictive distribution and the realisations).

Probabilistic forecasting is prevalent in the domain of weather forecasting. A common approach to probabilistic weather forecasting is the ensemble forecasting framework within Numerical Weather Prediction (NWP) (Palmer, 2012), where physics-based mathematical models encoding atmospheric and oceanic data are used to simulate multiple different possible future outcomes with different initial conditions or parametrisations. Another emerging field of study is the use of machine learning models, specifically Generative Neural Networks (Pacchiardi et al., 2024), which produce forecasts by transforming random noise to plausible future outcomes conditioning on past events. However, both methods require considerable computing power and time (Zhang et al., 2015). Moreover, how one makes use of newly acquired data to make fine adjustments to working models and update parameters, a process called data assimilation, is also of interest in forecasting research. Although the latest observations can be incorporated into prediction models as conditioning variables to help make more accurate predictions, it remains complex to use them to fine-tune model parameters in real time, as this often involves

re-training the entire model from the beginning. Finally, many current methods are data-demanding. Due to the large number of parameters to optimise, modern models often require enormous training data to achieve stable predictive power.

Therefore, this dissertation aspires to propose a data-efficient probabilistic forecasting framework, called Recursive Vine Predictive Estimation, that allows partial online model updates. Inspired by the recursive density estimation algorithm proposed by Hahn et al. (2018) and vine copulas' strong capability to capture linear and nonlinear dependency structures between variables, this dissertation combines the two mathematical tools to produce accurate probabilistic forecasts. This innovation differs from many modern solutions in several ways:

- It is data-efficient. There are fewer parameters and hyperparameters to optimise for this framework, so it does not require as much data as many Neural Networks to achieve stable performance.
- It supports data assimilation. Thanks to the partially recursive nature of this framework, the model can update its predictive distribution as new data comes, without the need to re-run the whole training process from the beginning.
- It takes a short time to run. The recursion algorithm used and the simplifying assumption of vine copulas mean that the training and fitting processes are simple, and it does not take as long as many Neural Networks.

The dissertation will be presented in the following structure: Chapter 2 will be dedicated to providing comprehensive foundations upon which the prediction framework is based, including vine copulas and scoring rules. Chapter 3 will offer mathematical explanations on some recursive density estimation algorithms that will be relevant to the framework proposed. We will also explore the possibility of extending the algorithms to settings where data may be dependent. The methodology and the implementation will be presented in Chapter 4, in which a toy example will be demonstrated to showcase the training and testing processes. In Chapter 5, we extend the application to more challenging settings of weather forecasting. A couple of chaotic models and a real-world dataset will be used to test the capability of the prediction framework, and its performance will be compared with other predictors. In Chapter 6, limitations and potential improvements to the framework, as

well as directions for future studies, will be discussed. A conclusion chapter follows in Chapter 7.

Chapter 2

Backgrounds

2.1 Vine Copulas

Vine copulas are a useful tool when one wishes to model dependencies between multiple variables. It offers a simple and convenient way to construct a valid multivariate copula without the need to work in the multi-dimensional space.

2.1.1 Introduction to Copulas

A copula is a d-dimensional probability distribution supported on a unit hypercube $[0,1]^d$ that encodes the dependency structure of d random variables.

Definition 1 (Copulas) *A copula function $C : [0,1]^d \to [0,1]$ is a cumulative distribution function (cdf) of d (possibly) dependent uniform variables $U_1, \ldots, U_d$. The copula function is defined by*

$$C(u_1, \ldots, u_d) = \mathbb{P}(U_1 \leq u_1, \ldots, U_d \leq u_d)$$

The power of copulas lies in their ability to separate a joint distribution function into individual marginal distributions and a copula function that describes the dependence between variables. Note that for a random variable X with a continuous cdf F_X, $F_X(X) \sim U[0,1]$. Therefore, by plugging a random variable back into its own cdf, one obtains a uniform distribution supported on $[0,1]$. This property enables us to link the copula function

with the joint distribution of multiple random variables using Sklar's theorem (Sklar, 1959).

Theorem 1 (Sklar's Theorem) *Let $X_1, \ldots, X_d$ be random variables with marginal cdfs $F_{X_1}(x_1), \ldots, F_{X_d}(x_d)$. The joint cdf $F_{X_1,\ldots,X_d}(x_1, \ldots, x_d)$ is linked to the marginal cdfs through a copula C. Thus, the joint cdf can be written in the form:*

$$F_{X_1,\ldots,X_d}(x_1, \ldots, x_d) = C(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)).$$

Moreover, if the marginal cdfs are continuous, then the copula is unique. Assuming all densities exist, the joint pdf can be written in the form:

$$f_{X_1,\ldots,X_d}(x_1, \ldots, x_d) = c(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)) \prod_{k=1}^{d} f_{X_k}(x_k) \tag{2.1}$$

where c is the copula density function.

As can be seen in (2.1), the copula isolates the dependency structure between the random variables, allowing one to write the joint pdf as the product of marginal pdfs multiplied by the copula density. This opens a way for a divide-and-conquer estimation technique, since the marginal densities and the copula density can be estimated individually in different ways.

2.1.2 Common Copula Families

Various copula families, both parametric and nonparametric, have been introduced to model different dependency structures between random variables. One crucial relationship between random variables that can be represented by copulas is the tail dependence, i.e., whether the probability of an extreme value observed for a random variable is associated with the probability of an extreme value observed for another variable. Tail dependence can be further classified into upper tail dependence (higher values in one variable are associated with higher values in another variable), lower tail dependence (lower values in one variable are associated with lower values in another variable), and dependence in both tails. Here we give details on some common copula families.

Independence copula The independence copula (often denoted as Π), also known as the product copula, models complete independence between random variables. It has the form:

$$\Pi(u_1, \ldots, u_d) = \prod_i^d u_i$$

and the copula density $c(u_1, \ldots u_d) = 1$ for all $u_1, \ldots, u_d \in [0, 1]^d$. For random variables whose dependency structure is represented by an independence copula, the joint density is simply the product of all marginal densities. This can be proven by applying (2.1) directly:

$$f_{X_1,\ldots,X_d}(x_1, \ldots, x_d) = \Pi(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)) \prod_{k=1}^d f_{X_k}(x_k) = \prod_{k=1}^d f_{X_k}(x_k).$$

Gaussian copula The Gaussian copula, inspired by the multivariate Gaussian distribution, is defined via the correlation matrix Σ and can be written as

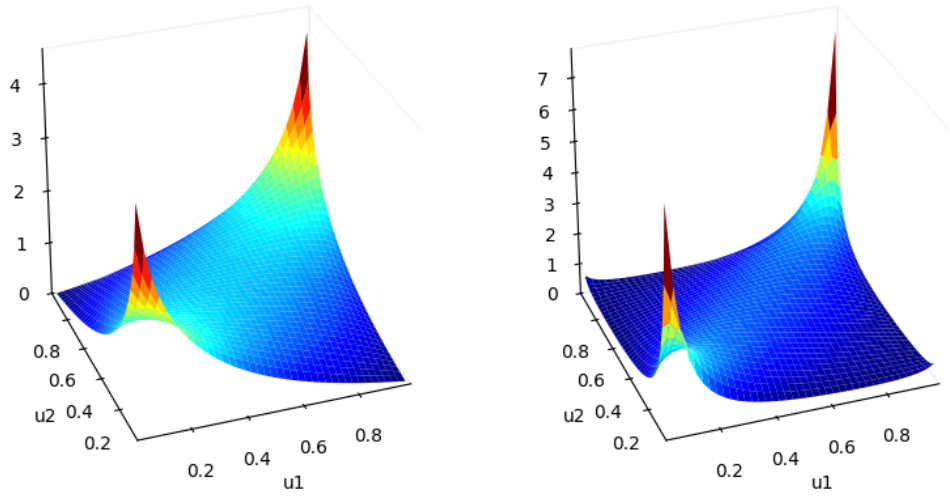
$$C_\Sigma(u_1, \ldots, u_d) = \Phi_\Sigma(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d))$$

where Φ_Σ is the multivariate standard normal cdf with correlation matrix Σ and Φ^{-1} is the inverse function of the univariate standard normal cdf. The Gaussian copula is suitable in representing general dependency structures with moderate dependence in both tails.

Student's t copula The Student's t copula, inspired by the Student's t distribution, has two parameters, the degree of freedom ν and the correlation matrix Σ. Just as the univariate Student's t distribution has heavier tails than the Gaussian distribution, the Student's t copula is effective in modelling relationships with stronger tail dependence than the Gaussian copula. The Student's t copula has the form:

$$C_{\nu,\Sigma}(u_1, \ldots u_d) = T_{\nu,\Sigma}(T_\nu^{-1}(u_1), \ldots, T_\nu^{-1}(u_d))$$

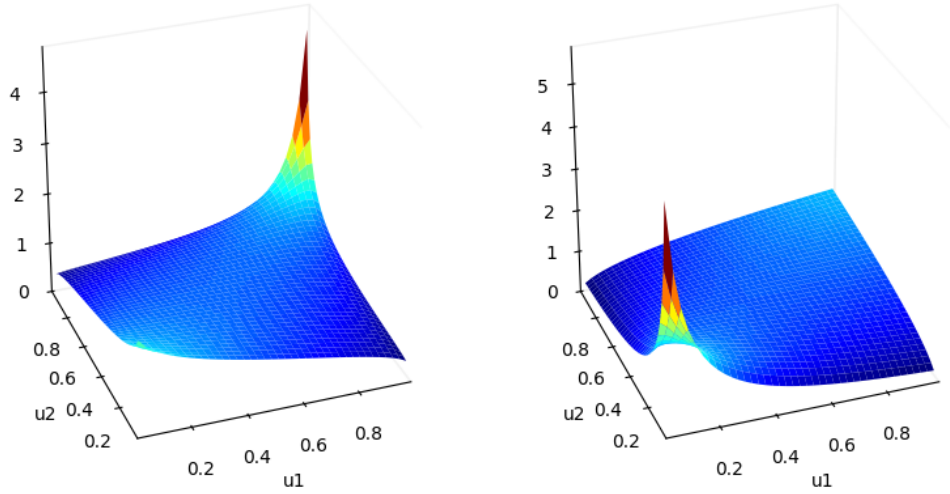
where $T_{\nu,\Sigma}$ is the multivariate Student's t cdf function with ν degrees of freedom and the correlation matrix Σ, and T_ν^{-1} is the inverse of the cdf of a univariate Student's t distribution with ν degrees of freedom.



(a) Gaussian copula, $\rho = 0.5$ **(b)** Student's t copula, $\rho = 0.5$, $\nu = 2$

Figure 2.1: Density plots for a bivariate Gaussian copula and a bivariate Student's t copula. Plots generated by the `plot()` function from the `pyvinecopulib` package.

Archimedean copulas Some other common parametric copula families are the Archimedean copulas. The Archimedean copulas are single-parameter copula families that are useful when modelling different kinds of tail dependence. The parameter θ controls the general dependence between variables. The Archimedean copulas include the Gumbel copula (modelling upper tail dependence), the Clayton copula (modelling lower tail dependence), etc., which are commonly used in modelling the dependence between financial assets (Patton, 2012).



(a) Gumbel copula, $\theta = 1.2$ **(b)** Clayton copula, $\theta = 0.5$

Figure 2.2: Density plots for a bivariate Gumbel copulas and a bivariate Clayton copula. Plots generated by the `plot()` function from the `pyvinecopulib` package.

Nonparametric copulas Aside from the parametric copula families mentioned above, there are also some nonparametric copula estimation methods. Here, we focus on nonparametric bivariate copulas. One common approach involves first transforming data supported on $[0, 1]^2$ to the unbounded $\mathbb{R}^2$ space and applying kernel density estimation (KDE). The estimated density is then transformed back to the unit square to produce a valid copula density (Nagler, 2018). The transformation method precludes the boundary bias problem (estimation bias tends to be high near the boundaries) the kernel estimator suffers from. Another method, called the Transformation Local-Likelihood (TLL) copula estimator, follows the same procedure, but instead of using KDE in the transformed space, it uses a local likelihood estimator, which is claimed to create smoother estimates (Geenens et al., 2017; Loader, 1996).

All the copula families introduced are helpful when modelling dependence between two variables. However, as more random variables are involved and a multivariate copula is needed, the parametric copula families may not be able to fully capture the dependence between variables, and nonparametric methods can become unstable as the dimensionality grows. Several approaches have been proposed to tackle this problem, one of which is the vine copula.

2.1.3 Vine Copulas

Vine copulas (Joe, 1996; Aas et al., 2009) are a highly flexible approach to represent dependency structures between multiple random variables. It is constructed by decomposing a d-dimensional copula into $d(d-1)/2$ unconditional and conditional bivariate copulas, called pair copulas. By doing so, one can model the dependency structure between any two variables using different copulas, rather than being restricted to a single copula for all variables.

We here give an example of vine copula decomposition for $d = 3$. Let X_1, X_2, X_3 be three continuous random variables, and let F_i and f_i denote the marginal cdf and pdf of variable X_i. The joint density function $f(x_1, x_2, x_3)$ has one factorisation according to the chain rule:

$$f(x_1, x_2, x_3) = f_{3|12}(x_3|x_1, x_2) f_{2|1}(x_2|x_1) f_1(x_1)$$

where $f_{i|D}$ is the conditional density function of X_i given X_j for $j \in D$. By Sklar's theorem, the second term on the right-hand side can be written as

$$\begin{aligned} f_{2|1}(x_2|x_1) &= \frac{f_{1,2}(x_1, x_2)}{f_1(x_1)} \\ &= \frac{f_1(x_1) f_2(x_2) c_{12}(F_1(x_1), F_2(x_2))}{f_1(x_1)} \\ &= c_{12}(F_1(x_1), F_2(x_2)) f_2(x_2) \end{aligned}$$

and the first term can be written as

$$\begin{aligned} f_{3|12}(x_3|x_1, x_2) &= \frac{f_{13|2}(x_1, x_3|x_2)}{f_{1|2}(x_1|x_2)} \\ &= \frac{f_{3|2}(x_3|x_2) f_{1|2}(x_1|x_2) c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2)}{f_{1|2}(x_1, x_2)} \\ &= f_{3|2}(x_3|x_2) c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2) \\ &= f_3(x_3) c_{32}(F_3(x_3), F_2(x_2)) c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2). \end{aligned}$$

Combining the terms, we have

$$\begin{aligned} f(x_1, x_2, x_3) = {} & c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2) \\ & \times c_{12}(F_1(x_1), F_2(x_2)) c_{32}(F_3(x_3), F_2(x_2)) \\ & \times f_3(x_3) f_2(x_2) f_1(x_1). \end{aligned} \tag{2.2}$$

As can be seen from the decomposition in (2.2), the joint density function can be expressed as the multiplication of three components: the product of marginals, the unconditional pair copulas, and the conditional copula. Also, as expected, what would originally be written as a 3-dimensional copula $c(F_1(x_1), F_2(x_2), F_3(x_3))$ is now expressed in terms of $3 \times (3-1)/2 = 3$ bivariate copulas. Each of the copulas can be modelled by different parametric or nonparametric copulas.

However, the decomposition is not unique; indeed, one can arbitrarily choose any factorisation of $f(x_1, x_2, x_3)$ using the chain rule and end up with a different decomposition. For $d = 3$, there are two other ways to decompose the joint density function:

$$\begin{aligned} f(x_1, x_2, x_3) = {} & c_{12;3}(F_{1|3}(x_1|x_3), F_{2|3}(x_2|x_3); x_3) \\ & \times c_{13}(F_1(x_1), F_3(x_3)) c_{23}(F_2(x_2), F_3(x_3)) \\ & \times f_3(x_3) f_2(x_2) f_1(x_1) \end{aligned}$$

and

$$\begin{aligned} f(x_1, x_2, x_3) = {} & c_{23;1}(F_{2|1}(x_2|x_1), F_{3|1}(x_3|x_1); x_1) \\ & \times c_{21}(F_2(x_2), F_1(x_1)) c_{31}(F_3(x_3), F_1(x_1)) \\ & \times f_3(x_3) f_2(x_2) f_1(x_1). \end{aligned}$$

When $d = 4$, one will have two main ways to form a vine construction:

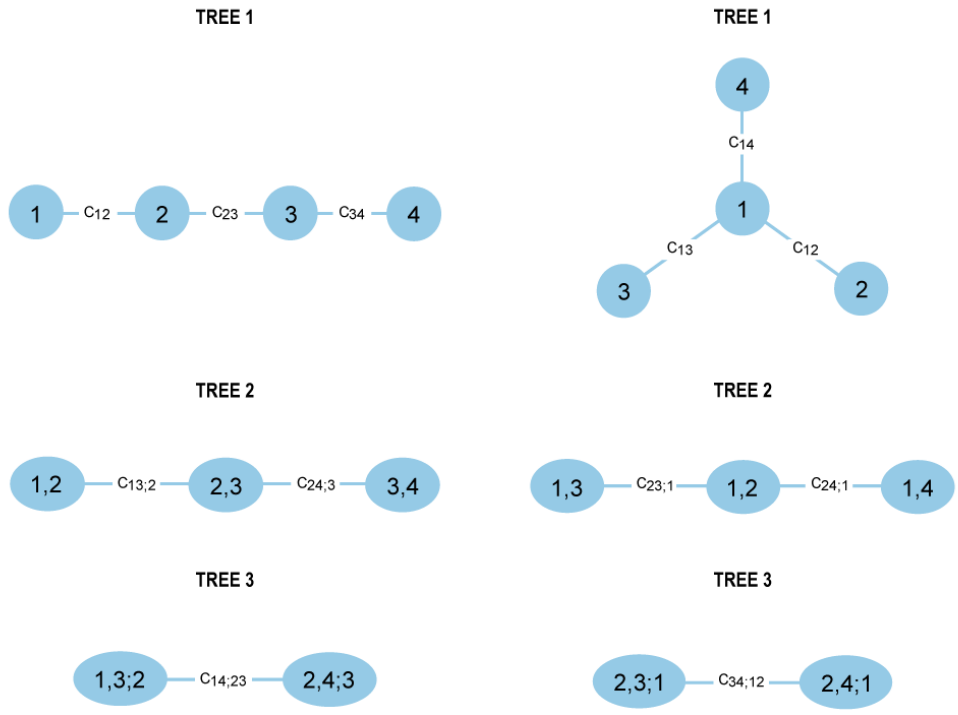
$$\begin{aligned} f(x_1, x_2, x_3, x_4) = {} & c_{14;23}(F_{1|23}(x_1|x_2, x_3), F_{4|23}(x_4|x_2, x_3); x_2, x_3) \\ & \times c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2) \\ & \times c_{24;3}(F_{2|3}(x_2|x_3), F_{4|3}(x_4|x_3); x_3) \\ & \times c_{34}(F_3(x_3), F_4(x_4)) c_{23}(F_2(x_2), F_3(x_3)) c_{12}(F_1(x_1), F_2(x_2)) \\ & \times f_4(x_4) f_3(x_3) f_2(x_2) f_1(x_1) \end{aligned} \tag{2.3}$$

and

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = {} & c_{34;12}(F_{3|12}(x_3|x_1, x_2), F_{4|12}(x_4|x_1, x_2); x_1, x_2) \\ & \times c_{23;1}(F_{2|1}(x_2|x_1), F_{3|1}(x_3|x_1); x_1) \\ & \times c_{24;1}(F_{2|1}(x_2|x_1), F_{4|1}(x_4|x_1); x_1) \\ & \times c_{14}(F_1(x_1), F_4(x_4)) c_{13}(F_1(x_1), F_3(x_3)) c_{12}(F_1(x_1), F_2(x_2)) \\ & \times f_4(x_4) f_3(x_3) f_2(x_2) f_1(x_1) \end{aligned} \tag{2.4}$$

and with each construction comes 12 different permutations of variables. One thus ends up with 24 different decompositions of the joint density. The two

main ways of construction can be more easily understood through graphical representations. If one treats copulas as 'links' connecting variables and sees copulas with different conditioning variables as in different layers, one can draw graphs demonstrated in Figure 2.3.



(a) D-vine **(b)** C-vine

Figure 2.3: Examples of the D-vine and C-vine structure when $d = 4$.

For construction expressed as (2.3) and represented by Figure 2.3a, the nodes are linked in the form of a long chain (or path), and is often referred to as the drawable vine (D-vine). For construction expressed as (2.4) and represented by Figure 2.3b, the nodes are linked in a star-like shape, with one node in the middle, connected by all the other nodes. This is often referred to as the canonical vine (C-vine). A C-vine is generally useful when modelling a system where there is naturally a variable in the centre interacting with all the other variables in the system. A good example of a C-vine is when modelling spatial relationships on a grid. On the other hand, a D-vine is particularly suitable for situations where the variables naturally form a path and adjacent

variables have the greatest impact on each other. When modelling time series, especially those exhibiting strong temporal dependencies, a D-vine is appropriate.

Useful as the vine decomposition, as d increases, the number of possible constructions grows superexponentially (Morales-Napoles, 2016) and possible shapes are no longer restricted to the D-vine and C-vine constructions. Also, it becomes increasingly difficult to determine whether a vine construction is valid or not. For example, the number of possible configurations of a vine copula when $d = 10$ can exceed 4.8×10^{14}. Therefore, Bedford and Cooke (2001, 2002) proposed a way to organise vine structures with a graphical tool called the regular vine (R-vine). An R-vine associated with d variables consists of multiple pair copulas whose structure is governed by a set of linked trees $\mathcal{V} = (T_1, \ldots, T_{d-1})$ satisfying

1. $T_1 = (V_1, E_1)$ is a tree with a node set $V_1 = \{1, \ldots d\}$ and and edge set E_1.
2. For $m = 2, \ldots, d-1$, the tree $T_m = (V_m, E_m)$ consists of nodes defined by the edges from the previous tree $V_m = E_{m-1}$ and edges E_m.
3. (Proximity condition) For $m = 2, \ldots, d-1$, two nodes in T_m can only be connected by an edge if the corresponding edges of T_{m-1} have a common node.

Although multiple R-vine decompositions and shapes exist for $d \geq 5$, the C-vine and D-vine remain two valid structures, and with the help of the R-vine, the two structures can be constructed fairly easily even when d is large. In the C-vine construction, every tree in $\mathcal{V}$ has a star-like shape, with a node, called the root, connecting all the other nodes. In the D-vine construction, every tree in $\mathcal{V}$ is a path, with all but two nodes having exactly 2 neighbouring nodes. Each of the two nodes at both ends of a D-vine, called the leaf, has only one neighbouring node. Thanks to the proximity condition, both C-vines and D-vines can be specified relatively easily: For a C-vine, one can construct it by specifying a sequence of roots, one for each tree, and making the other nodes in the same tree connected to it; for a D-vine, a set of linked trees can be easily defined by specifying the first tree. In higher-level trees, two nodes are connected if and only if the corresponding edges in the previous tree are attached to the same node. Therefore, once the first tree is determined, all other higher-level trees are also automatically determined.

2.1.4 Simplified Vine Copulas

Although possible vine structures have been regulated by R-vines, it is still intractable to fit every conditional pair copula when d becomes large, since conditional dependencies can be hard to model and estimate. In practice, one often makes the simplifying assumption that the conditioned variables in a conditional pair copula do not depend on the exact values of the conditioning variable set. This is a weaker assumption than fully conditional independence. The simplifying assumption allows for conditional dependence; it only requires that the dependence does not change for different values of the conditioning variables. A vine copula fitted under the simplifying assumption is often referred to as a simplified vine copula.

For example, under the simplifying assumption, the 3-dimensional vine construction in (2.2) can now be written as (Czado and Nagler, 2022):

$$\begin{aligned} f(x_1, x_2, x_3) = {} & c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2)) \\ & \times c_{12}(F_1(x_1), F_2(x_2)) c_{32}(F_3(x_3), F_2(x_2)) \\ & \times f_1(x_2) f_2(x_2) f_3(x_3), \end{aligned}$$

that is, the conditional copula density $c_{13;2}$ now depends on x_2 only via the conditional cdfs but not directly so. The shape of the copula and the strength of dependence thus do not change with x_2. Of course, this might not be a valid representation of the true dependency structure where there is really conditional dependence. Hobæk Haff et al. (2010) however showed numerically that parametric vine copulas under the simplifying assumption still serve as a good approximation even if the true model is not simplified.

Several Python packages for vine copula fitting and estimation are available online. The `pyvinecopulib` package (Nagler and Vatter, 2025) is used throughout this dissertation. It is a Python interface to the header-only C++ library `vinecopulib`. When fitting a vine copula with this package, as with most packages online, the simplifying assumption is used to make fitting feasible and efficient.

2.1.5 Selecting the Optimal Vine Structure

Kendall's tau One concept closely related to copulas is the strength of dependence. Multiple measures of strength of dependence between two random variables exist, such as the Pearson Correlation Coefficient, Spearman's ρ, and Kendall's τ. Among them, Kendall's τ is often the choice in copula literature due to several reasons. Similar to Spearman's ρ, Kendall's τ

does not take absolute distance between data points into consideration. It measures the degree to which observations from two variables have a similar rank. Kendall's τ between two random variables X, Y is defined as

$$\tau_{XY} = \mathbb{P}\left((X_1 - Y_1)(X_2 - Y_2) > 0\right) - \mathbb{P}\left((X_1 - Y_1)(X_2 - Y_2) < 0\right)$$

where X_1, X_2, Y_1, Y_2 are iid copies of X and Y, respectively. As seen from the expression above, Kendall's τ is a measure of ordinal association and thus is invariant to monotone transformation of the marginals and robust against outliers. It can be shown that Kendall's τ can be expressed as a function of the copula associated with X and Y:

$$\tau_{X,Y} = 4 \iint C(F_X(x), F_Y(y)) dC(F_X(x), F_Y(y)) - 1.$$

Empirically, Kendall's τ is evaluated by counting the concordant and discordant pairs of observations in a sample. Specifically,

$$\hat{\tau}_{X,Y} = \frac{2}{m(m-1)} \sum_{j=2}^{m} \sum_{i=1}^{j-1} \operatorname{sign}(x_i - x_j) \operatorname{sign}(y_i - y_j)$$

where m is the sample size and x_i, y_i are the i-th observations of X and Y, respectively.

Another reason Kendall's τ is a popular choice in copula literature is that there exist one-to-one relationships between Kendall's τ and the parameter θ for different Archimedean copulas (Hofert et al., 2012). Therefore, an Archimedean copula can be estimated via estimation of Kendall's τ.

Since Kendall's τ can be used to evaluate the strength of dependence, selecting the optimal vine copula can be performed by finding the vine structure with the strongest dependence in terms of Kendall's τ.

Dißman's algorithm Now that vine structures are under the simplifying assumption and regulated by R-vines, one remaining problem is to efficiently select the optimal R-vine structure among numerous choices when d is large. One pervasive algorithm, first proposed by Dißmann et al. (2013), is a greedy approach that aims to maximise the strength of dependence at lower tree levels. Specifically, Dißman's algorithm starts with the first tree T_1 and finds the maximum spanning tree, i.e., an undirected acyclic tree that contains all the nodes and has the largest possible sum of weights of its edges. Here the weight of an edge is defined by the absolute value of empirical Kendall's

τ between the two nodes connected by the edge. Therefore, finding the maximum spanning tree in this context involves identifying the tree with the largest sum of absolute empirical Kendall's τ between nodes. After the structure of T_1 is determined, optimal copula families and their parameters are selected with maximum likelihood estimation. For any subsequent tree, all tree structures satisfying the proximity condition are considered, and the process of finding the maximum spanning tree and fitting the optimal copula families and parameters is repeated for each tree. The rationale behind this algorithm is to attempt to capture as much dependence as possible in lower-level trees, as this algorithm is commonly used in conjunction with truncation. In practice, it is common to truncate the vine copula after some number of trees and set all copulas in subsequent trees as independence copulas or Gaussian copulas (Brechmann et al., 2012; Brechmann and Joe, 2015). By concentrating most dependence in lower trees, this can help reduce dimensionality without sacrificing too much information.

Since this structure selection technique is designed for general R-vines, it naturally works well with the simpler C-vine and D-vine structures. When only the C-vine structure is considered, it chooses the tree with the largest sum of absolute empirical Kendall's τ from a set of trees with different possible roots in each level. Therefore, for a d-dimensional C-vine, this only requires evaluating the sum of absolute empirical Kendall's τ for $d(d-1)/2$ times in total, and even fewer with truncation. When only the D-vine structure is considered, it searches for the path that has the largest sum of absolute empirical Kendall's τ in T_1, and all subsequent trees are automatically defined. This only requires comparisons between $d!$ trees in the first tree level.

It is worth noting that Dißman's algorithm is a greedy algorithm, which means that it only strives to maximise the sum of weights at the current tree level and does not take subsequent trees into consideration. Therefore, this might not give the 'optimal' tree set in the sense that the sum of all empirical absolute Kendall's τ in all tree levels is maximised. This, however, does not compromise our goal, as we aim to capture most dependence in the lower-level trees, and higher-level trees may be truncated.

2.2 Scoring Rules

A scoring rule (SR) $S(\hat{P}, y)$ is a metric for a probability distribution $\hat{P}$ against the realisation y. It is a function of a distribution and an observation, assigning a penalty value to the distribution based on how much the distribution

deviates from the observed value. SRs are useful as an alternative to the conventional likelihood function, especially when the likelihood is difficult, if not impossible, to compute for certain complex systems. Also, since SRs are penalty terms imposed upon a distribution rather than a single predicted value (such as the RMSE or the coefficient of determination), they are suitable for quantifying the performance of a probabilistic forecasting model. It enables the forecaster to predict a future value along with the uncertainty of the prediction, which is crucial for many real-world problems such as weather forecasting.

Borrowing the analogy from Dawid and Musio (2014), one can think of SRs as a 'score' for the game between a decision-maker and the Nature: Consider a random variable Y that can take values in $\mathcal{Y}$, and $\mathcal{P}$ is a family of distributions over $\mathcal{Y}$. The decision-maker has to quote a probability distribution $\hat{P}$ for Y. When the true value y of Y is observed, $S(\hat{P}, y)$ serves as the penalty term for the quoted distribution.

If the random variable Y is generated by some true distribution P^*, then the decision-maker can compute the expected penalty with the formula:

$$S(\hat{P}, P^\star) \coloneqq \mathbb{E}_{Y \sim P^\star} S(\hat{P}, Y).$$

This is called the expected SR. Assuming $\hat{P}, P^* \in \mathcal{P}$, i.e. the model is well-specified, a SR is said to be *proper* relative to a family of distributions $\mathcal{P}$ if the expected SR is minimised when $\hat{P} = P^\star$, and *strictly proper* if the minimum is unique.

For a strictly proper scoring rule, the minimised value $S(P^*, P^*)$ is called the *entropy*, and the quantity $D(P^\star || \hat{P}) \coloneqq S(\hat{P}, P^\star) - S(P^\star, P^\star)$ defines a statistical divergence, since for a strictly proper SR, the defined divergence satisfies non-negativity and attains zero if and only if $\hat{P} = P^\star$.

2.2.1 Common Scoring Rules

One common example of SRs is the negative log-likelihood (or the log score) $S_{NLL}(\hat{P}, y) = -\log \hat{p}(y)$, where $\hat{p}$ is the density of $\hat{P}$. It can be shown that the negative log-likelihood is strictly proper, and the corresponding statistical divergence is the Kullback-Leibler divergence:

$$D_{KL}(\hat{P} || P^\star) \coloneqq \int p^\star(x) \log \frac{p^\star(x)}{\hat{p}(x)} \, dx.$$

Another SR that is also commonly used and versatile is the Kernel Score, defined as

$$S_k(\hat{P}, y) \coloneqq \mathbb{E}\big[k(X, X')\big] - 2\mathbb{E}\big[k(X, y)\big], \quad X, X' \sim \hat{P}$$

where k is a positive-definite kernel function. If the kernel is the negative Euclidean norm $k(x, y) = -||x - y||_2^\beta$ for $\beta \in (0, 2)$, then one derives the Energy Score:

$$S_{\mathrm{E}}^{(\beta)}(\hat{P}, y) \coloneqq 2 \cdot \mathbb{E}\left[\|X - y\|_2^\beta\right] - \mathbb{E}\left[\|X - X'\|_2^\beta\right], \quad X, X' \sim \hat{P}.$$

Also, if $X, Y \in \mathbb{R}$ and $k(x, y) = -|x - y|$ and the original Kernel Score is divided by 2, then one derives the Continuous Ranked Probability Score (CRPS). The CRPS can thus be written in the following form (Gneiting and Raftery, 2007; Székely and Rizzo, 2005):

$$CRPS(\hat{P}, y) \coloneqq \mathbb{E}\Big[\big|X - y\big|\Big] - \frac{1}{2}\mathbb{E}\Big[\big|X - X'\big|\Big], \quad X, X' \sim \hat{P}.$$

When the cdf of the estimated distribution $\hat{P}$ can be evaluated easily, another equivalent form of the CRPS is more commonly used:

$$CRPS(\hat{P}, y) \coloneqq \int_{-\infty}^{\infty} \Big(\hat{P}(x) - H(x - y)\Big)^2 \, dx \tag{2.5}$$

where $H(t)$, called the Heaviside step function, attains 1 when $t \geq 0$ and 0 when $t < 0$. It can be shown that the CRPS is also a strictly proper SR.

There is an intuitive explanation behind the integral form of the CRPS: a good estimator should be able to produce a distribution whose density is concentrated around the true value as tightly as possible. This phenomenon is reflected in a cdf where the cumulative density rises sharply around the true value. A perfect estimator should put all the mass of its estimated distribution at the value, which is exactly what the Heaviside step function represents. Any deviation from the perfect estimate should be punished. Figure 2.4 demonstrates how the CRPS is computed.

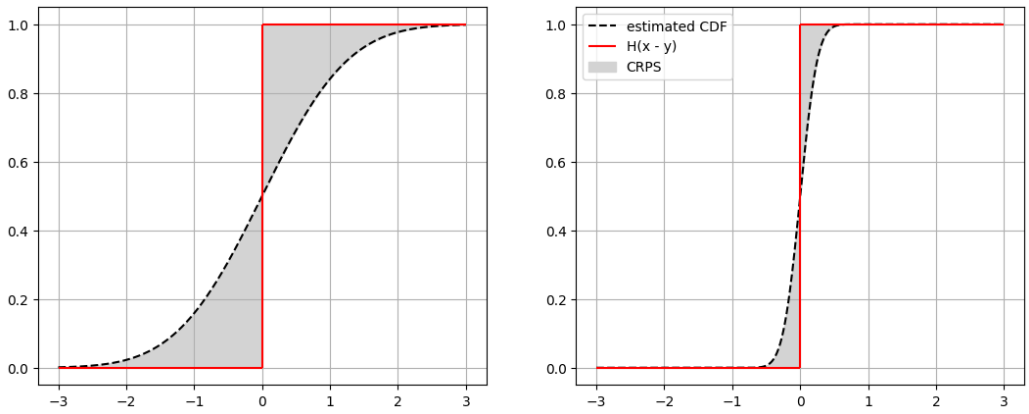


Figure 2.4: Demonstration of the CRPS when the true observation $y = 0$. The area of the shaded region is the CRPS. The left panel shows a poorer estimate, characterised by a larger CRPS, and the right panel shows a better estimate, characterised by a lower CRPS. Even though both estimators correctly put the mode at the real value, the estimator on the left is punished for its uncertainty.

2.2.2 Prequential Scoring Rules

When making probabilistic forecasts in discrete time settings and when the data are not independent, one might consider evaluating the loss of some estimate of a future value Y_{t+l} when the observations up to the current time step $\mathbf{y}_{1:t}$ have been collected. A variation of conventional SRs, termed prequential (predictive-sequential) SRs (Dawid, 1984), can be a suitable candidate. To compute the prequential SR, one evaluates the performance of the estimated probability distribution $\hat{P}_t$ up to time step t on the observed value y_{t+l} via the SR $S(\hat{P}_t, y_{t+l})$ for every $t \in 1, \ldots, T$. The cumulative loss is the prequential SR:

$$S_T(\hat{P}, y) = \sum_{t=1}^{T-l} S(\hat{P}_t, y_{t+l}).$$

It can be shown that if the SR in the summand is (strictly) proper, then the prequential SR is also (strictly) proper (Dawid and Musio, 2015; Pacchiardi et al., 2024).

Chapter 3

Recursive Density Estimation Algorithms

This chapter will be dedicated to the introduction to some nonparametric recursive algorithms for mixing and mixture density estimation. Inspired by the Bayesian updating scheme, these recursive algorithms aim to estimate distributions in an online manner, updating and refining their estimates as new data comes. Also, the algorithms introduced here are nonparametric: Instead of assuming a distribution and updating its parameters, the algorithms utilise a highly flexible prior called the Dirichlet process, and with every new data point, they update the entire distribution function.

This chapter also explores the possibility of extending the recursive algorithms to settings where data are dependent, as the consistency of these algorithms are proven based on the assumption that data are independent. This chapter aims to show that under mild dependence, the algorithms still work well empirically.

3.1 Dirichlet Process

A Dirichlet process (DP), to put it briefly, is a distribution over distributions. It is a nonparametric approach to put a prior on an unknown distribution of interest. A DP can be specified with a base distribution H and a positive constant α called the concentration parameter. If a distribution G follows a DP specified as above, it must satisfy the following:

Definition 2 (Dirichlet Process) *A probability distribution G follows a Dirichlet process with a base distribution H and a concentration parameter α, denoted by $G \sim DP(\alpha, H)$, if for every finite measurable partition $(A_1, \ldots, A_n)$ of the sample space,*

$$(G(A_1), \ldots, G(A_n)) \sim \mathrm{Dir}(\alpha H(A_1), \ldots, \alpha H(A_n)).$$

The Blackwell-MacQueen urn scheme To understand the DP, several constructive definitions have been introduced to provide an intuitive understanding of its properties. One perspective relevant to the algorithms to be introduced later is the Blackwell-MacQueen urn scheme (Blackwell and MacQueen, 1973). We start with a metaphorical empty urn, and specify a positive constant α and a base distribution H. For step 1, we draw a ball of colour $\theta_1 \sim H$ from the sample space Θ and put it into the urn. Then, for every subsequent step $n+1$ for $n = 1, 2, \ldots$, we either, with probability $\frac{\alpha}{n+\alpha}$, draw another ball of colour θ_{n+1} from the same base distribution H and put it into the urn, or, with probability $\frac{n}{n+\alpha}$, randomly draw a ball from the urn, note the colour, replace it, and put another ball of the same colour into the urn. Therefore at step $n+1$, there will have been n balls in the urn, and the probability of drawing a ball of colour $\theta_{n+1} \in A$ will become:

$$\mathbb{P}(\theta_{n+1} \in A \mid \theta_1, \ldots, \theta_n) = \frac{\alpha}{n+\alpha} H(A) + \frac{n}{n+\alpha} \left(\frac{1}{n} \sum_{i=1}^{n} \delta_{\theta_i}(A) \right) \tag{3.1}$$

where δ_x is the Dirac measure, which concentrates the entire probability mass of 1 at x and 0 everywhere else, so the measure $\delta_x(A)$ of any set A is 1 if $x \in A$ and 0 otherwise. Since the draws θ_n are random, the empirical distribution G generated by $\theta_1, \theta_2, \ldots$ is therefore random. The process characterised as such defines a Bayesian prior over G, which is precisely the Dirichlet process $DP(\alpha, H)$. The urn scheme illustrates the characteristic 'rich-get-richer' phenomenon captured by the DP. The concentration parameter α can also be seen as expected pseudo-draws or the degree of confidence in the base distribution H. The larger the α is, the less impact new draws have on the posterior distribution.

The stick-breaking construction Yet another version of the constructive view which will prove helpful later in this chapter is the stick-breaking construction (Sethuraman, 1994). We start by imagining a stick of unit length. For $i = 1, 2, \ldots$, we draw a proportion $\beta_i \sim \mathrm{Beta}(1, \alpha)$, then assign a weight according to $\pi_1 = \beta_1$ and $\pi_i = \beta_i \prod_{k=1}^{i-1}(1 - \beta_k)$ for $i > 1$. The

weights are essentially the length of the stick fragments broken off from the remaining stick from the previous round. For each i we also draw a label θ_i from some distribution H and associate it with the weight π_i at step i. The resulting probability distribution G constructed in this process will then have the form:

$$G = \sum_{i=1}^{\infty} \pi_i \delta_{\theta_i}. \tag{3.2}$$

According to Sethuraman (1994), the distribution G will also follow a DP with a base distribution H and a concentration parameter α, that is,

$$G \sim DP(\alpha, H).$$

It is worth noting that although the base distribution H of a DP can either be continuous or discrete, any distribution generated by the DP will always be discrete with probability 1, as shown clearly in (3.2).

3.2 Recursive Mixing Density Estimation

Consider a mixture distribution with probability density function $f(x) = \int f(x|u)p(u)du$ whose conditional distribution $f(x|u)$ is known and u follows some unknown mixing distribution P with density p. If one who wants to estimate the mixing density with the Bayesian approach specifies their prior uncertainty with a Dirichlet process $DP(\alpha, P_0)$, after observing the first outcome, they can update their posterior belief according to the following formula (Newton et al., 1998):

$$\begin{aligned} P_1(A) &= \mathbb{P}(u \in A \mid X_1) \\ &= \mathbb{E}\left[\mathbb{P}(u \in A \mid u_1) \mid X_1\right] \\ &= \mathbb{E}\left[\frac{\alpha}{1+\alpha} P_0(A) + \frac{1}{1+\alpha}\delta_{u_1}(A) \mid X_1\right] \\ &= \frac{\alpha}{1+\alpha} P_0(A) + \frac{1}{1+\alpha}\mathbb{P}(u_1 \in A \mid X_1) \end{aligned}$$

where the third line is obtained by considering the predictive distribution of the Blackwell-MacQueen urn scheme (3.1) after the first draw. Therefore, after receiving the first observation, the posterior predictive distribution can be written as a linear combination of the base distribution of the DP and the

posterior distribution. Moreover, assuming P_0 is continuous and has density p_0, and setting $w_1 = \frac{1}{1+\alpha}$, we can write down the density form of the formula:

$$p_1(u) = (1 - w_1)p_0(u) + w_1 \frac{f(x_1|u)p_0(u)}{\int f(x_1)p_0(x_1|u)\,du}. \tag{3.3}$$

In other words, $p_1(u)$ is the density of the *exact* Bayesian posterior predictive distribution after having the first observation if one starts with a prior summarised in a Dirichlet Process. To compute the posterior after observing 2 values $P_2(A) = \mathbb{P}(u \in A \mid X_1, X_2)$, however, is far more complicated and does not enjoy the same simple form as when computing $P_1(A)$. As more observations arrive, the calculation of the posterior becomes more intractable. That being said, Newton et al. (1998) proposed generalising (3.3) and making the updating scheme recursive, naming it the Predictive Recursion (PR) algorithm. Specifically, after observing the n-th value x_n, one estimates the predictive density with the formula:

$$p_n(u) = (1 - w_n)p_{n-1}(u) + w_n \frac{f(x_n|u)p_{n-1}(u)}{\int f(x_n)p_{n-1}(x_n|u)\,du}. \tag{3.4}$$

The weight sequence (w_n) must satisfy $\sum_{n=1}^{\infty} w_n = \infty$ and $\lim_{n\to\infty} w_n = 0$. In general, the estimate p_n is not the exact Bayesian posterior predictive density unless $n = 1$ but only an approximation thereof, but this allows one to do online estimation as new information arrives in a recursive manner, and it has been proven that under some mild conditions, the estimator is consistent, i.e. $p_n \to p$ as $n \to \infty$ (Newton, 2002; Tokdar et al., 2009). Figure 3.1 shows a simple demonstration of the Predictive Recursion estimation of mixing density on a 2-component Gaussian Mixture model.

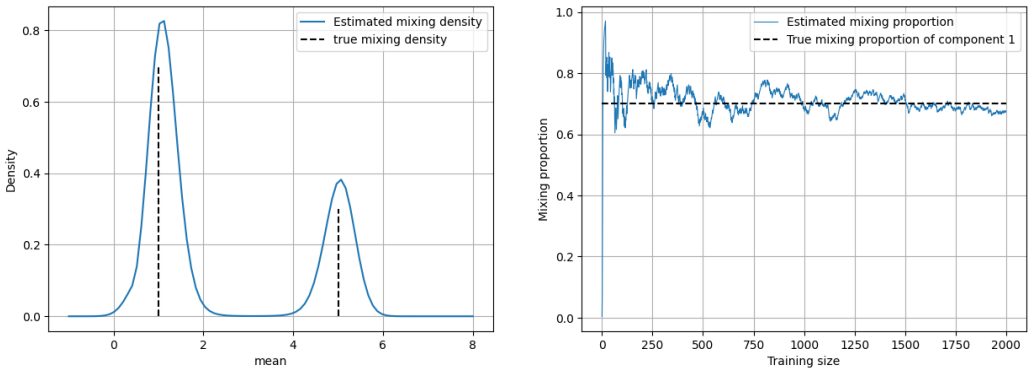


Figure 3.1: Predictive Recursion estimate of a 2-component Gaussian Mixture model with a mean vector $(1, 5)$, a variance vector $(1, 1)$, and a weight vector $(0.7, 0.3)$. The initial guess is a $U[-1, 8]$ distribution. The left panel shows the estimation result with 2,000 data points. The right panel demonstrates the convergence of the estimates to the true mixing proportion of component 1 (which is 0.7).

3.3 Recursive Bayesian Predictive Update

Sometimes, the mixture distribution $f(x)$, rather than the mixing distribution $p(u)$, is more often than not the quantity of interest. However, estimation of the mixture distribution with the mixing distribution with Predictive Recursion often involves the integration $f_n(x) = \int f(x \mid u)p_n(u)du$ and evaluation of the normalising constant, which is usually difficult, if not intractable. Modern solutions include Markov Chain Monte Carlo (MCMC) methods, but these are generally computationally expensive.

Hahn et al. (2018) thus introduced a Recursive Bayesian Predictive (R-BP) method that shares the same spirit as Predictive Recursion, but bypasses the need to estimate the mixing distribution. They consider a bivariate function $k(x, x_n)$ that allows for a recursive update of the estimated mixture density:

$$f_n(x) = f_{n-1}(x)k(x, x_n).$$

It follows that $k(x, x_n)$ must satisfy:

$$k(x, x_n) = \frac{f_n(x)}{f_{n-1}(x)} = \frac{\int f(x|u)f\,(x_n|u)\,p_{n-1}(\mathrm{d}u)}{\int f(x|u)p_{n-1}(\mathrm{d}u)\int f\,(x_n|u)\,p_{n-1}(\mathrm{d}u)}.$$

The expression on the right-hand side can be seen as a bivariate copula

density function that depends on the sample size n. Therefore, by writing $k(x, x_n) = c_n(F_{n-1}(x), F_{n-1}(x_n))$, the recursive updating formula can be written as:

$$f_n(x) = c_n(F_{n-1}(x), F_{n-1}(x_n)) \cdot f_{n-1}(x).$$

For the recursive estimates to converge, it has to be ensured that $c_n \to 1$ as $n \to \infty$, that is, the copula sequence converges to the independence copula. However, this requires impractical specification of the copula sequence (c_n). They therefore show in the paper that if the prior guess f_0 (with cdf F_0) is a standard Gaussian mixture (so that $f(x \mid u) = N(x; u, 1)$) with the mixing distribution G summarised by a DP with a base distribution $G_0 = N(0, \tau^{-1})$ and a concentration parameter c, it can be written as the following by the stick-breaking construction in Section 3.1 :

$$f_0(x) = \int f(x \mid u)\, dG_0(u) = \sum_{j=1}^{\infty} w_j f(x \mid u_j)$$

where $u_j \overset{iid}{\sim} G_0$ and w_j follows the weights in the construction with Beta$(1, c)$. The Bayesian update f_1 after receiving the first observation can be written in the form:

$$\begin{aligned} f_1(x) &= (1 - \alpha_1) f_0(x) + \alpha_1 f_0(x) c_{\rho_0}(F_0(x), F_0(x_1)) \\ &= f_0(x)\big[(1 - \alpha_1) + \alpha_1 c_{\rho_0}(F_0(x), F_0(x_1))\big] \end{aligned} \tag{3.5}$$

where $\alpha_1 = \sum_{j=1}^{\infty} \mathbb{E}[w_j^2]$ and c_{ρ_0} is a Gaussian copula with correlation $\rho_0 = (1 + \tau)^{-1}$. They claim that as soon as the expression is formed, we no longer require the prior f_0 to be a Gaussian mixture; it can be any distribution, since the assumption of Gaussian-ness of the conditional distribution is captured by the Gaussian copula instead. This thus becomes a truly nonparametric method without any restriction on the prior.

As mentioned in the paper, the expression (3.5) can be seen as a mixture of an independence copula and a Gaussian copula, with their respective weights being $1 - \alpha_1$ and α_1. Similar to the Predictive Recursion algorithm, this update does not directly apply to $f_{n-1} \to f_n$ generally, but with a carefully specified weight sequence (α_n), the estimated distributions converge to the true distribution in the Kullback-Leibler sense with probability 1 (Hahn et al., 2018). The general update step after receiving the n-th observation x_n can thus be written in the following form:

$$f_n(x) = f_{n-1}(x)\big[(1 - \alpha_n) + \alpha_n c_{\rho}(F_{n-1}(x), F_{n-1}(x_n))\big] \tag{3.6}$$

where c_ρ is the density of the Gaussian copula with correlation ρ and a weight sequence (α_n) satisfies

$$\sum_{n=1}^{\infty} \alpha_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2 < \infty.$$

This cleverly spares one the need to specify the copula sequence (c_n), in exchange for the specification of the weight sequence (α_n), which is easier to do in practice. The weight sequence proposed in (Hahn et al., 2018) is $\alpha_n = (1+n)^{-1}$, another weight sequence proposed in (Huk et al., 2024) is $\alpha_n = (2 - \frac{1}{n})(1+n)^{-1}$, both of which satisfy the requirements. Now, there remains one parameter ρ to be provided for the copula. The copula with a fixed ρ is then used throughout the entire recursion.

One can also directly estimate the cdf with the following expression:

$$F_n(x) = (1-\alpha_n)F_{n-1}(x) + \alpha_n H_\rho\left(F_{n-1}(x), F_{n-1}(x_n)\right)$$

with

$$H_\rho(u,v) = \Phi\left(\frac{\Phi^{-1}(u) - \rho\Phi^{-1}(v)}{\sqrt{1-\rho^2}}\right)$$

where Φ is the cdf of a standard univariate Gaussian distribution. Here $\sqrt{1-\rho^2}$ serves similar purposes to the bandwidth for KDE (Hahn et al., 2018). Therefore, although $\rho = 0.95$ is suggested in the original paper and is claimed to be a suitable choice for most cases, it makes sense to choose an optimal ρ for different datasets, similar to what people commonly do when performing KDE. Here we provide simple demonstrations of the R-BP estimates of a 2-component Gaussian Mixture model and a 2-component Poisson Mixture model in Figure 3.2 and Figure 3.3.

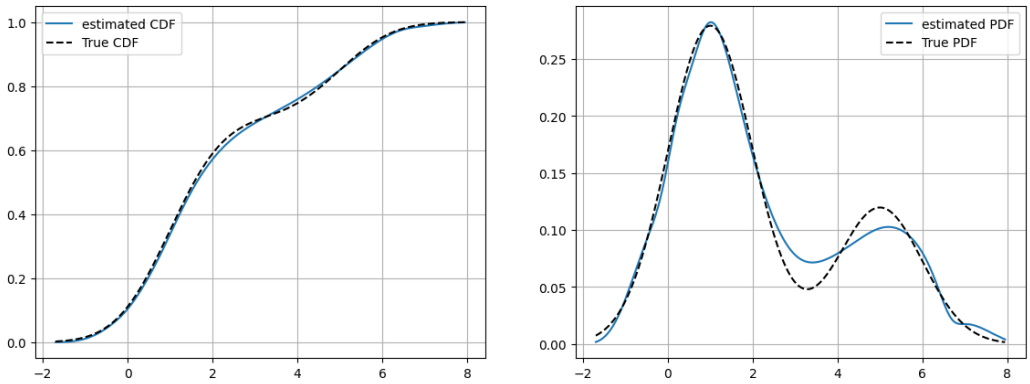


Figure 3.2: The R-BP estimated cdf and pdf (blue solid lines) of a 2-component Gaussian Mixture model (dashed lines) with a mean vector $(1, 5)$, a variance vector $(1, 1)$, and a weight vector $(0.7, 0.3)$, with training data of size $n = 1000$. The initial guess is a $N(3, 2)$ distribution and $\rho = 0.85$.

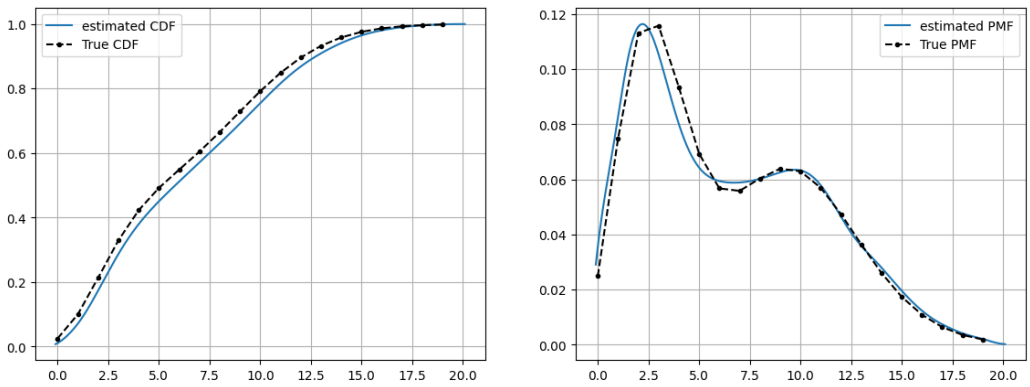


Figure 3.3: The R-BP estimated cdf and pmf (blue solid lines) of a 2-component Poisson Mixture model (dashed lines) with a rate vector $(3, 10)$ and a weight vector $(0.5, 0.5)$, with training data of size $n = 1000$. The initial guess is a $N(5, 5)$ distribution and $\rho = 0.85$.

One common problem for this updating scheme is the choice of the initial guess f_0, as for the update to be feasible, the support of the prior should contain the support of the true distribution. Additionally, even if the condition is satisfied, the update would become very unstable or computationally infeasible if the tails of the prior are too light compared to the true distribution due to numerical overflow. Some common workarounds include choosing a distribution with heavier tails, such as a Cauchy distribution, as has been

experimented with in (Hahn et al., 2018; Huk et al., 2024) and standardisation of data (Huk et al., 2024). However, a good strategy to choose a proper prior guess for general cases has yet to be formally proposed.

Huk et al. (2024) further extended the algorithm to multivariate settings (with the assumption that each variable is iid) by incorporating vine copulas. They decompose the multivariate update into two parts: the update of marginal distributions and the estimation of the multivariate copula, which involves the use of vine copulas.

In the paper, they show that the recursive update of a d-dimensional multivariate distribution f can be expressed in the following form:

$$\frac{f_n(x_1,\ldots,x_d)}{f_{n-1}(x_1,\ldots,x_d)} = \prod_{i=1}^{d}\left\{\frac{f_n^{(i)}(x_i)}{f_{n-1}^{(i)}(x_i)}\right\}\cdot\frac{c_n\left(F_n^{(1)}(x_1),\ldots,F_n^{(d)}(x_d)\right)}{c_{n-1}\left(F_{n-1}^{(1)}(x_1),\ldots,F_{n-1}^{(d)}(x_d)\right)}$$

where $f^{(i)}$ and $F^{(i)}$ are the marginal pdf and cdf of variable X_i for $i = 1,\ldots,d$.

The first term on the right-hand side involves updating each marginal distribution, which can be done by using the univariate R-BP algorithm (3.6). The second term involves a recursive update of the multivariate copula. Practically, one can leave the copula recursion as an implicit process and focus solely on updating the marginals. When the final estimates $f_n^{(i)}$ are obtained, one can simply estimate the multivariate copula density by using the R-vine decomposition and estimating all pair copulas, parametrically or nonparametrically, with all the data all at once. The estimated marginals and the multivariate copula density are then multiplied to produce the final estimate.

The paper also mentions the possibility of adapting the method to regression tasks. It is done by noticing that the conditional density of some random variable Y given a vector of explanatory variables $\mathbf{x} = (x_1,\ldots,x_d)$ can be expressed as:

$$f(y \mid \mathbf{x}) = \frac{f(y,\mathbf{x})}{f(\mathbf{x})} = \frac{f_y(y)\cdot\prod_{i=1}^{d}\{f_i(x_i)\}\cdot c(y,\mathbf{x})}{\prod_{i=1}^{d}\{f_i(x_i)\}\cdot c(\mathbf{x})} = \frac{c(y,\mathbf{x})\cdot f_y(y)}{c(\mathbf{x})}. \quad (3.7)$$

However, this method is not the main focus of discussion in the paper and is therefore not explored further, except as a proposed application of the algorithm.

3.4 R-BP Algorithm on Dependent Data

All research results introduced thus far assume the random variables are iid. Even dependencies between multiple variables are considered in (Huk et al., 2024), dependencies, especially temporal dependencies, within a single data sequence are not considered. Moreover, one common challenge of the Predictive Recursion algorithm, as pointed out in (Newton et al., 1998; Newton, 2002), is that the estimates are order-dependent, that is, the order in which the data are input and processed by the estimator will have an impact on the estimate. In the original paper, estimating the distribution with different data permutations and taking the average is proposed as one of the solutions. However, when the data are a time series and are received in real time, permutation becomes impossible to perform. Therefore, whether the recursive estimator can correctly converge when the variables are no longer iid remains an interesting topic to explore.

For estimating mixing distributions, Guha and Roy (2022) showed that for weakly stationary time series (time series with constant mean and finite constant variance) with exponentially decaying memory and under some technical conditions, the estimator is still consistent, i.e., the estimates are still able to converge to the true mixing proportion (the mixing distribution). They also show mathematically that the rate of convergence drops as the autocorrelation of the time series increases. No research has been done, however, on the estimation of mixture distributions with the R-BP algorithm. Therefore, before proposing the main prediction framework in this dissertation, we will empirically demonstrate the extent to which the R-BP algorithm remains a good estimator when data are temporally dependent.

We test the R-BP algorithm on two time series with different rates of memory decay. The first one is an autoregressive (AR) process of order 1, whose memory decays exponentially. The value of a general AR(p) process at any time t has the following form:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t$$

where c is the intercept, ε_t is white noise (or innovation) at time t with zero mean and constant variance, and ϕ_i is the autoregressive coefficient of the i-th lag. An AR(1) process with no intercept term can be written as

$$X_t = \phi_1 X_{t-1} + \varepsilon_t.$$

The other time series is an autoregressive fractionally integrated moving average (ARFIMA) process (Brockwell and Davis, 2016). The ARFIMA process has the same form as the autoregressive integrated moving average (ARIMA) process:

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right)(1-B)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i B^i\right)\varepsilon_t.$$

where B is the backshift operator such that $BX_t = X_{t-1}$ and $B^k X_t = X_{t-k}$ for any $k > 0$, and $(1-B)^k$ is defined via the power series expansion,

$$(1-B)^k = 1 + \sum_{j=1}^{\infty} \frac{k \cdot (k-1) \cdot \cdots \cdot (k-j+1)}{j!}(-B)^j.$$

The only difference between ARFIMA and ARIMA lies in the fact that the differencing parameter d in an ARFIMA process is allowed to take non-integer values, hence the modifier 'fractionally' in its name. This makes ARFIMA processes exhibit a much slower rate of memory decay than AR, MA, or ARMA processes, whose memory decays exponentially. When $d \in (0, 0.5)$, the ARFIMA process is weakly stationary.

The ARFIMA(0, d, 0) process, which will be tested below, is the simplest ARFIMA process. Without any AR or MA components, it can be simply expressed in the form:

$$(1-B)^d X_t = \varepsilon_t.$$

Rearranging the terms and expanding gives us the value of X_t:

$$X_t = (1-B)^{-d}\varepsilon_t = \sum_{i=0}^{\infty} \psi_i \varepsilon_{t-i}$$

where ψ_i is the coefficient of x^i in the power series expansion $(1-x)^{-d}, |x| < 1$. Precisely, $\psi_i = \binom{i+d-1}{i}$. Therefore, past shocks in the process do not die out as quickly as exponentially but only polynomially. Due to its pure form and slower rate of memory decay, an ARFIMA(0, d, 0) process is a good candidate for testing how the R-BP algorithm performs when the iid or weak-dependence assumptions are violated.

Multiple datasets are used to test the algorithm. Each dataset is a mixture of two AR(1) or ARFIMA(0, d, 0) time series, one with mean 0 and the other 5. All other parameters are fixed for both mixture components. Before estimation, the data are standardised to make sure we can use a $N(0,1)$ distribution as initial guess. Innovation variances of the components are

adjusted accordingly to ensure all processes have variance 1 in stationarity to rule out the influence of differences in variance on performance. For the AR(1) process, data with different autoregressive parameters ϕ_1 are used; for the ARFIMA(0, d, 0) process, data with different difference parameters d are used. The results are presented in Figure 3.4 and Figure 3.5.

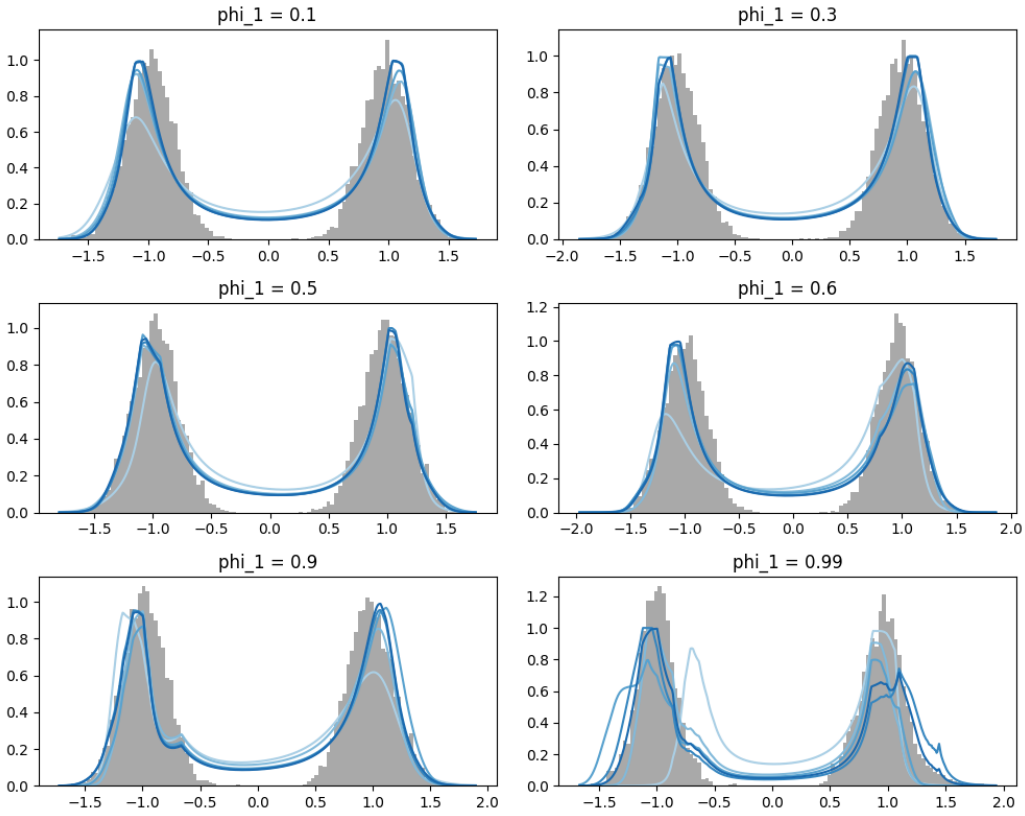


Figure 3.4: Density estimates of the AR(1) process for different ϕ_1. Five estimated curves, from the lightest to the darkest, represent the different training sizes $n = 100, 500, 1000, 5000, 10000$. Histograms of the data (in grey) are included for comparison. The correlation parameter ρ for the estimator is fixed at 0.85 and not tuned for each dataset.

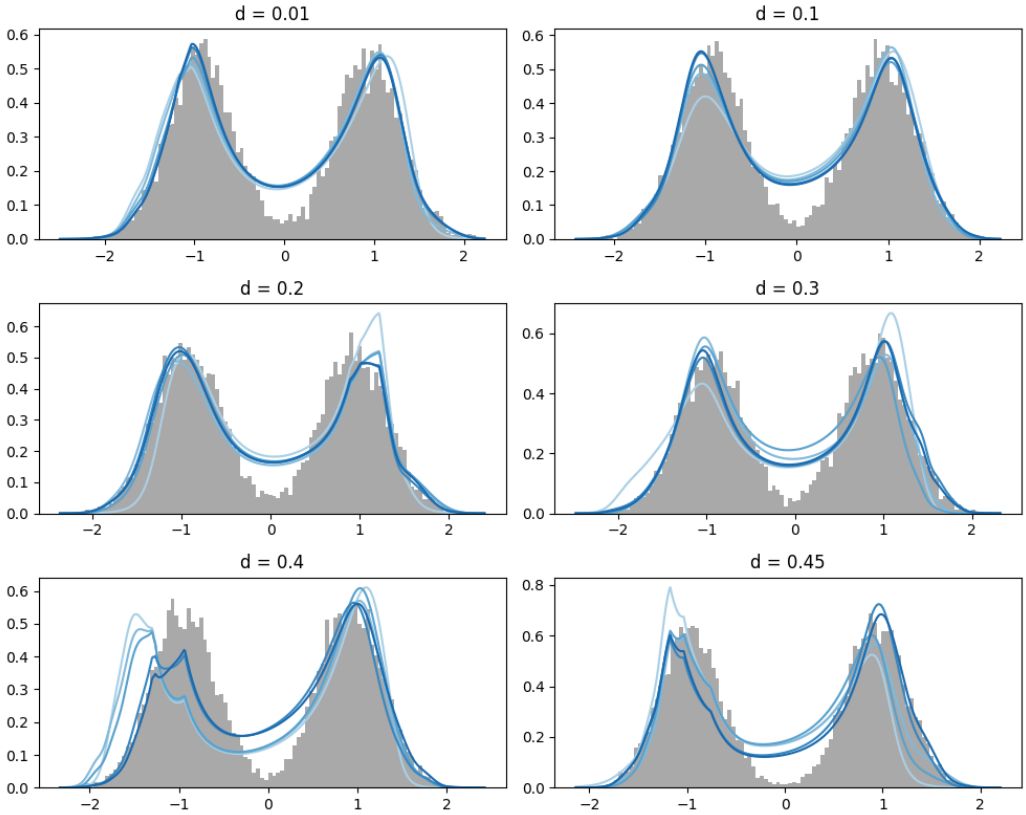


Figure 3.5: Density estimates of the ARFIMA(0, d, 0) process for different d. Five estimated curves, from the lightest to the darkest, represent the different training sizes $n = 100, 500, 1000, 5000, 10000$. Histograms of the data (in grey) are included for comparison. The correlation parameter ρ for the estimator is fixed at 0.85 and not tuned for each dataset.

As can be seen in Figure 3.4, as the autoregressive parameter ϕ_1 grows, it takes more data to enable the estimator to converge to the true distribution. When $\phi_1 = 0.99$, the estimate remains very unstable even after using up to 10,000 training data. On the other hand, from Figure 3.5, it can be seen that the estimator is still able to capture the overall shapes of the mixture distributions when d is moderate, even though the processes have long memory. Estimates become more unstable as d approaches 0.5. The results indicate that although convergence of the R-BP algorithm cannot be guaranteed mathematically when a time series is dependent or even has long memory, it has been shown empirically that the estimate remains a good approximation when the dependence is moderate.

It should be noted that this section and the discussion here only serve the purpose of justifying using the R-BP algorithm for density estimation on temporally dependent data, and to demonstrate that the estimate remains a good approximation even if the time series exhibits slow memory decay to some extent. Formal proofs of the results shown here have not been studied and are beyond the scope of this dissertation.

Chapter 4

Recursive Vine Predictive Estimation

In this chapter we will propose the main prediction framework. We term the framework Recursive Vine Predictive Estimation, as it combines the recursive R-BP algorithm and vine copulas to produce predictive distributions for future values, conditioning on some past values.

4.1 Method

This framework applies the R-BP algorithm to dependent time series to estimate the marginal distribution and utilises vine copulas to summarise information from a given number of past values, in order to make probabilistic forecasts about future values. The framework is motivated by noticing that one can modify (3.7) to take the past values $\mathbf{x}_{1:n}$ for some time series as conditioning variables and the future, unrealised value x_{n+t} as response variable to estimate the density:

$$f(x_{n+t} \mid \mathbf{x}_{1:n}) = \frac{c(x_{n+t}, \mathbf{x}_{1:n}) \cdot f_{x_{n+t}}(x_{n+t})}{c(\mathbf{x}_{1:n})}. \tag{4.1}$$

This allows one to make probabilistic forecasting about a future event, given the information collected up to the current time period. Each component in (4.1) can be estimated efficiently using different methods: The marginal density of x_{n+t} can be estimated with the R-BP algorithm (3.6) with a tuneable parameter ρ; the multivariate copula $c(x_{n+t}, \mathbf{x}_{1:n})$ in the numerator can be

estimated with a simplified vine copula. The multivariate copula $c(\mathbf{x}_{1:n})$ in the denominator is simply the normalising constant and can be practically evaluated with numerical integration in many cases.

To evaluate the marginal density $f_{x_{n+t}}(x_{n+t})$, one can incorporate all the data gathered until time n and run the R-BP algorithm to obtain the estimate $f_n(x_{n+t})$. As new data comes, one simply needs to update the estimate according to (3.6). This estimate serves as a 'base' probability with which a certain realisation of x_{n+t} will show up when the entire process is run for a long enough time. Although the value of ρ is fixed throughout the entire estimation process, we can still select an optimal value before the recursion starts. Unlike how people commonly use some rule-of-thumb bandwidths for KDE, there has not been a rule-of-thumb correlation ρ estimator for the R-BP algorithm. However, optimisation is still viable with gradient descent. Any loss function designed for probability distributions can be used. In this dissertation, the prequential CRPS is used to optimise ρ.

In practice, it is more common to make predictions based on a given number of past observations instead of the entire history. If the last k observations are used to make a prediction about a value t steps away in the future, this general formula can be used:

$$f(x_{n+t} \mid \mathbf{x}_{n-k+1:n}) = \frac{c(x_{n+t}, \mathbf{x}_{n-k+1:n}) \cdot f_{x_{n+t}}(x_{n+t})}{c(\mathbf{x}_{n-k+1:n})}. \tag{4.2}$$

The optimal size of observation window can vary from dataset to dataset, depending on several factors such as the strength of memory, the rate of decay of memory, the degree of chaotic-ness in a chaotic system, or even the truncation level specified for the vine copula (i.e. the complexity of the model), so it makes sense to allow the model to automatically select the optimal observation window size based on some criterion. Generally, as the dimensionality of a vine copula increases, it becomes harder to accurately evaluate its density due to the curse of dimensionality and the loss of information incurred by the simplifying assumption. Therefore, if a simpler model with a smaller observation window can perform as well as a more complex model, the simpler one should be preferred. Based on this rationale, a simple observation window size selection scheme can be conducted by starting with including only one past value in the vine copula (that is, using a simple bivariate copula $c(x_{n+t}, x_n)$) and gradually increasing the window size to the point where some metric of interest does not improve any more. Since this is a probabilistic forecasting task, a reasonable metric candidate is the CRPS, which will be used for this dissertation.

This method has three major advantages: 1. It is computationally efficient to run compared to many modern RNN-based time series prediction models, e.g., Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) due to its simplicity, 2. It is also more interpretable compared to RNN-based models, whose weights and parameters often contain obscure meanings and may not carry interpretive importance for their users, and 3. The update of marginal density can be done online, that is, the update can be made as new information arrives thanks to its recursive nature.

4.2 Simulation Study: AR Process

We implement the prediction framework on Python [1] and explore its capability empirically. Here we list the details on how the model is designed, what the training process involves, and how the performance is evaluated and compared. We will test the prediction model on a toy example of an AR(3) process first in this section. Application on more complex datasets and real-world situations will be investigated in the next chapter.

Setup We generate 5,000 data from a AR(3) process, which has the form:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \phi_3 X_{t-3} + \varepsilon_t.$$

Here we choose $\phi_1 = 0.1, \phi_2 = 0.3, \phi_3 = 0.5, c = 0$ and $\sigma_\varepsilon = 1$. The data are then standardised to ensure they have a mean of 0 and a standard deviation of 1. The standardisation process is critical since it ensures we can use a $N(0, 1)$ or Cauchy$(0, 1)$ distribution as a prior guess when estimating the density with R-BP for any dataset. The data are split into a training set, a validation set, and a test set, according to the proportions 30%-20%-50%. One of the advantages of the prediction framework is its low requirement for training data sizes, so this train-test split proportion is different from those for most ML-based models, which typically use most of the data for training. The model will select an optimal observation window size k, and for every time step n in the study, it can access k historical values $\mathbf{x}_{n-k+1:n}$ and will give a probabilistic forecast as well as a point forecast of the t-step-ahead future value X_{n+t}. A summary plot for the AR(3) dataset generated is shown in Figure 4.1.

[1] Code for reproducing the main results in this dissertation is available at: `https://github.com/kckhchen/Recursive-VineCop`.

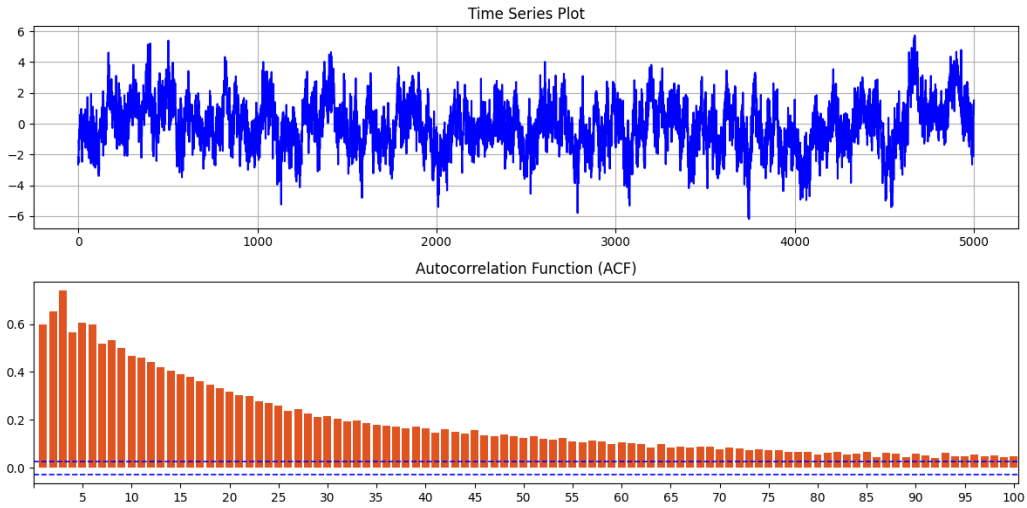


Figure 4.1: Summary plots for the AR(3) system with $\phi_1 = 0.1, \phi_2 = 0.3, \phi_3 = 0.5, \sigma_\varepsilon = 1$ and no intercept. The upper plot shows the trajectory of the time series, while the lower plot shows the estimates of autocorrelation between different lags. The blue dashed line represents the credible intervals for no autocorrelation.

Parameters and hyperparameters The model has several parameters and hyperparameters. Some of them are of importance to this dissertation and are thus optimised and their impacts are studied, while the others, although being no less important, are set to be fixed in the interest of time and will be left for future studies. First, the correlation ρ of the bivariate Gaussian copula for the R-BP algorithm is optimised. As stated above, ρ serves a similar role to the bandwidth parameter in a KDE. If ρ is set too high, the bandwidth will be small, and the estimated marginal density will appear to be 'spiky' and less smooth. On the other hand, if ρ is too low, the estimated density will look too smooth and fail to capture the true shape of the true distribution. Another parameter tuned in this study is the observation window size k. It is worth noting that $k + 1$ can be alternatively interpreted as the dimensionality of the vine copula. This governs the complexity of the model, as more observations available for the predictor can generally be beneficial, but an excessively high-dimensional vine copula could render the predictions unstable due to the curse of dimensionality. Other hyperparameters, e.g., the truncation level of the vine copula, and the learning rate, patience, and tolerance in the training process, are set to be fixed. The weight sequence used for the R-BP algorithm is set to be $\alpha_n = (2 - \frac{1}{n})(1 + n)^{-1}$, as suggested in (Huk et al., 2024).

Training process The training process consists of two training stages: optimisation of the ρ for marginal density estimation and optimisation of the observation window size for copula density estimation. We set the initial distribution to $N(0, 1)$ to run the R-BP algorithm, evaluate the prequential CRPS, and update ρ with gradient descent according to the CRPS evaluated on the training set. An early-stopping mechanism is implemented to prevent overfitting. The early-stopping mechanism is triggered when the CRPS on the validation set stops dropping for a certain period of time. Once ρ has been optimised, it is used to perform another predictive recursion algorithm, this time using all training and validation data to produce the final estimate of marginal density. Afterward, the same training and validation data are used to optimise the observation window size for the vine copula. Although there are no hard rules regulating which vine structure to use for time-series data, a D-vine structure is imposed. The D-vine provides a path structure, which is natural for data that are sequentially arranged and correlated. Other vine structures have also been experimented with, but the D-vine structure yields the best predictive results, so it is set as the default in the study. The data will first undergo an inverse transformation (with the marginal cdf estimated by R-BP) and become uniformly distributed within the interval $[0, 1]$ to allow for copula fitting. We start by fitting a vine copula using the training data, with an observation window of size one, which essentially is a bivariate copula that encodes the dependence between two time points t steps away. The estimated copula density is then multiplied by the estimated marginal density to obtain the final probabilistic forecasts, and the CRPS against the validation set will be evaluated. This step will be repeated, each time with the observation window size increased by 1, until the validation CRPS no longer drops. For copula fitting, the model utilises Dißman's algorithm to find the optimal D-vine structure for the given data. The candidate copula families include the Gaussian copula, the Student's t copula, the independence copula, and the nonparametric Transformation Local-Likelihood (TLL) copula. Other common Archimedean copulas, e.g., the Clayton copula or the Frank copula, are not included for simplicity.

Models for comparison In the simulation study, four prediction models are considered and their performance on the dataset is compared. The main focus of the dissertation is to show that a vine copula that sees further back into the past, even though restricted by the simplifying assumption and truncation, is still more powerful than a simple bivariate copula that only has access to one past value. Therefore, two prediction models using the same framework introduced here will be compared, the only difference between the

two being that one is equipped with a flexible vine copula with an adjustable observation window size (Recursive-VineCop, the studied model), while the other is only equipped with a bivariate copula (Recursive-BiCop). Two other models serve as the benchmarks for the study: a 'naïve' model that predicts every step only with the estimated marginal density from the R-BP algorithm (in other words, not considering temporal dependencies whatsoever), and a 'persistence' model that purely predicts the next value to be the current value, the motivation behind which is that when a temporal dependent system exhibits strong positive dependence and long memory, the future value should not be too far off from the current one. The persistence predictor serves as a good benchmark for comparison, as a good predictor should be able to capture the underlying relationships or higher-order dependencies, and thus make use of its knowledge about dependencies to 'nudge' the current value in the right direction by the right amount, outperforming the persistence model. However, the persistence model, unlike the other models here, intrinsically provides point forecasts, instead of a full probability distribution. Therefore, all probabilistic forecasting models will also provide the median values as point forecasts, and two performance metrics will be provided, one for probabilistic forecasts and the other for point forecasts. They will be discussed in the next paragraph.

Performance metrics Two different metrics are provided for performance comparison in the study, each serving a different purpose. For probabilistic forecasting, the CRPS (2.5) is evaluated. Models for comparison in terms of the CRPS include Recursive-VineCop, Recursive-BiCop, and the naïve model. On the other hand, for median point forecasts, the root mean squared error (RMSE) is used. The RMSE is given by the formula:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{\left(\hat{x_i} - x_i\right)^2}{n}}.$$

Here the three models above are again compared. In addition, the persistence model is included for comparison.

Results The result for marginal density estimation on the AR(3) dataset produced by the R-BP algorithm, along with optimal ρ, is shown in Figure 4.2. The resulting optimal ρ for this dataset is much lower than the suggested 0.95 in the original R-BP paper, but still leads to a very good estimate of the marginal. This may be due to the fact that the stationary

distribution of an AR(p) system is already a Normal distribution and has been standardised, so the $N(0, 1)$ prior naturally fits perfectly. However, the same phenomenon also happens on the other datasets, which will be discussed later in Chapter 5. Generally, a ρ lower than 0.95 is enough for making good estimation. On the contrary, if ρ was set to be 0.95, we would end up with an overly spiky estimate (see Appendix A). This justifies the ρ-tuning process proposed for this framework.

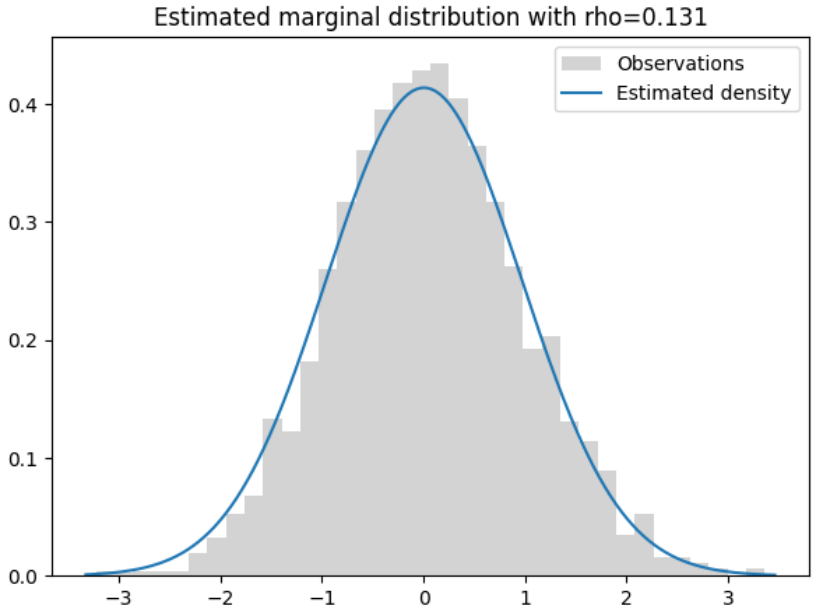


Figure 4.2: Estimate of the marginal distribution of the AR(3) dataset. The estimated density function (blue solid line) is plotted with the histogram of observations (grey) for comparison.

For this task, the predictors aim to make a prediction for the next time step x_{n+1} for any given time n. The observation window size selection process is conducted, and the Recursive-VineCop model ends up with an observation window size $k = 4$, i.e., a 5-dimensional copula. A comparison plot including point forecasts and credible intervals from Recursive-VineCop and Recursive-BiCop is given in Figure 4.3. The naïve marginal predictor will always predict the same values and credible intervals irrespective of time steps and therefore is not included here for readability purposes.

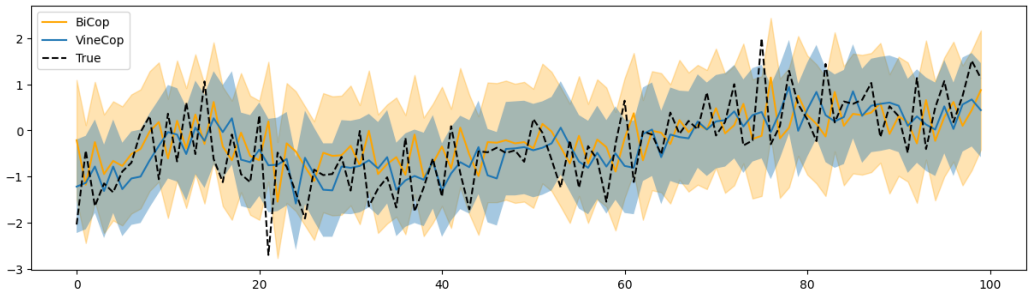


Figure 4.3: Median point forecasts (solid lines) and the 90% credible intervals (shaded areas) for the AR(3) dataset. 100 observations are shown here for demonstration. The true values (dashed line) fall inside the CIs most of the time.

It seems from the plot that the Recursive-BiCop model behaves like a persistence prediction model and assumes the future value will remain the same as the current value. It can be seen that the median forecasts from Recursive-BiCop (orange solid line) resemble the true value, except that they are one lag behind. On the other hand, the Recursive-VineCop model, which looks further into the past, captures the true trajectory more accurately and produces smaller credible intervals. Boxplots of CRPS for the three probabilistic forecasting models are presented in Figure 4.4 and comparisons of performance metrics are provided in Table 4.1. Although the difference in CRPS between the Recursive-VineCop and the Recursive-BiCop models is not statistically significant, there is still a visible improvement in performance in terms of the average, the median, and the spread of the CRPS, as well as the RMSE.

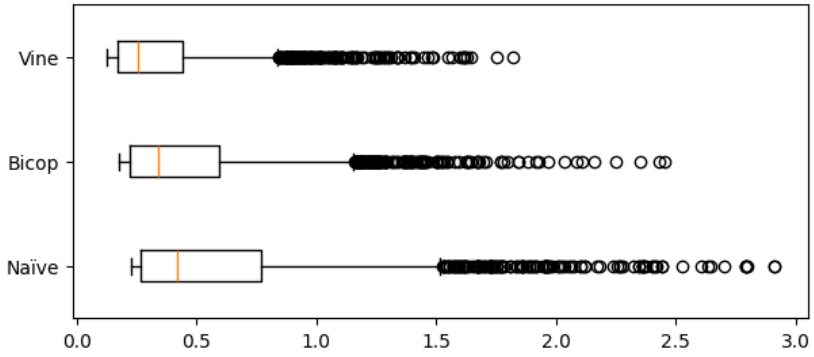


Figure 4.4: Boxplots of CRPS for probabilistic forecasting models on the AR(3) system. The Recursive-VineCop model has a lower median and spread of CRPS.

	Mean CRPS	Std. CRPS	RMSE
VineCop	**0.35193**	**0.25268**	**0.70735**
BiCop	0.46697	0.33830	0.81320
Naive	0.59822	0.45392	0.91645
Persistence	-	-	0.85453

Table 4.1: Performance metrics for probabilistic forecasting models on the AR(3) system. For each metric, lower is better. The Recursive-VineCop model has the lowest average CRPS, standard deviation of CRPS, and the RMSE.

Chapter 5

Case Studies: Weather Forecasting

We have thus shown that the framework works well for a simple autoregressive process dataset. In this chapter, we extend the application of the framework to some more complex and challenging situations. We mainly focus on probabilistic weather forecasting tasks. Here the models will be tested on three complex datasets, including two deterministic yet chaotic systems which aim to emulate the dynamics and movements of the atmosphere in nature, and finally a real-world meteorological dataset.

The same setup as in Chapter 4 is used: For each dataset, 5,000 data points are generated or collected, before they are standardised and split into 30% of training data, 20% of validation data, and 50% of test data. The training and evaluation process remains the same as well. We predict the future value t steps away with the four aforementioned models and compare their performance.

5.1 Deterministic Chaotic Systems

In this section we consider two deterministic chaotic systems, the Lorenz63 system and Lorenz96 system, introduced by meteorologist Edward Lorenz in 1963 and 1996 respectively (Lorenz, 1963, 1996), to emulate the chaotic nature of the atmosphere.

5.1.1 Lorenz63 System

The Lorenz63 system is a set of three ordinary differential equations designed to model the dynamics of atmospheric convection:

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \sigma(y - x) \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= x(\rho - z) - y \\ \frac{\mathrm{d}z}{\mathrm{d}t} &= xy - \beta z\end{aligned}$$

where σ, ρ, and β are constants that govern the behaviour of the system and t is the time index. The three variables x, y, z are related to the rate of convection, the horizontal temperature variation, and the vertical temperature variation in the atmosphere, respectively (Anderson et al., 2004). Following the original 1963 paper we set $\sigma = 10, \rho = 28$, and $\beta = \frac{8}{3}$.

In this study, after generating the data, we only keep the y-axis as observations. This is therefore a univariate time series. The summary plots of the y-axis of the Lorenz63 system are presented in Figure 5.1.

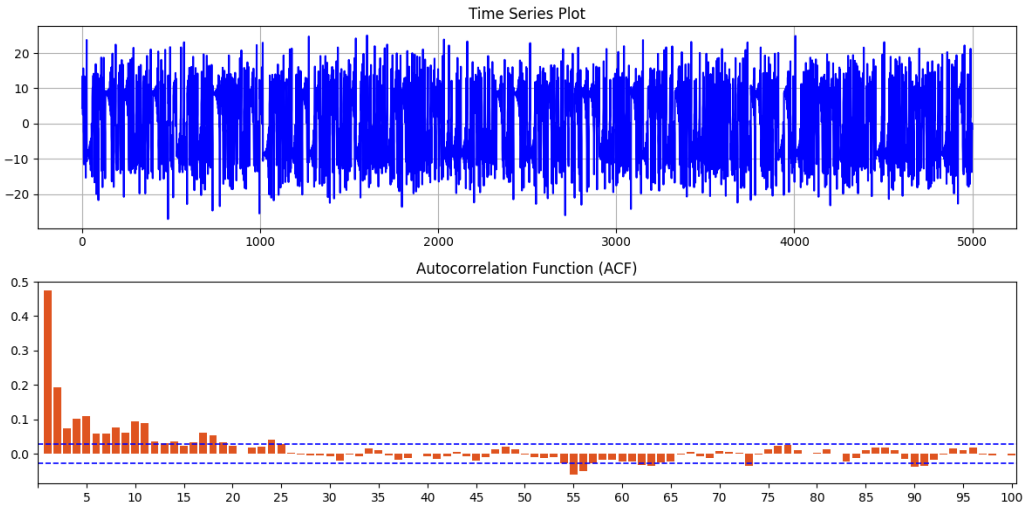


Figure 5.1: Summary plots for the y-axis of the Lorenz63 system. The upper plot shows the trajectory of the time series, while the lower plot shows the estimates of autocorrelation between different lags. The blue dashed line represents the credible intervals for no autocorrelation.

For this task, the predictor will attempt to predict the next immediate value ($t = 1$). The estimate of the marginal distribution is shown in Figure 5.2.

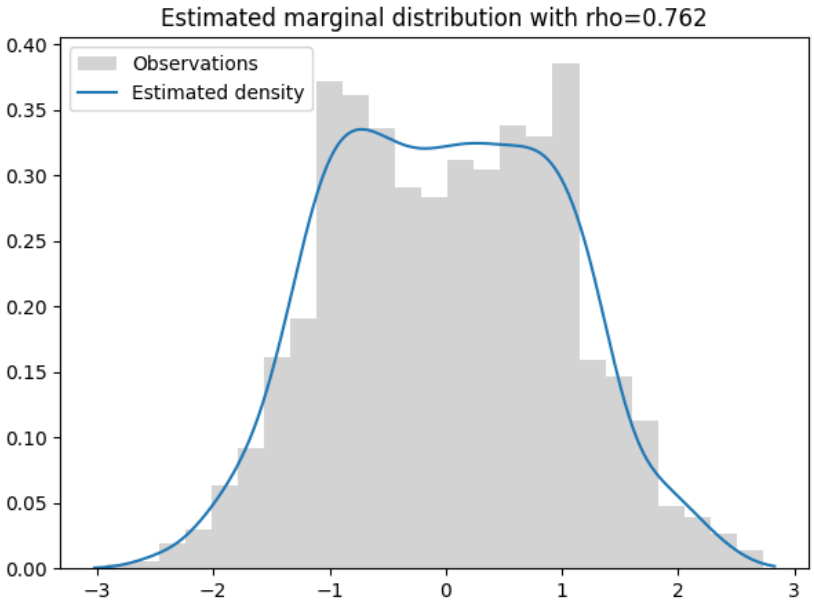


Figure 5.2: Estimate of the marginal distribution of the Lorenz63 system. The estimated density function (blue solid lines) is plotted with the histogram of observations (grey) for comparison.

The estimate seems to be too smooth in the centre of the distribution and does not fully capture the bimodal shape. However, it successfully captures the slopes around the tails. For the vine copula training, the model selects an observation window size $k = 6$. The forecasting results are shown in Figure 5.3.

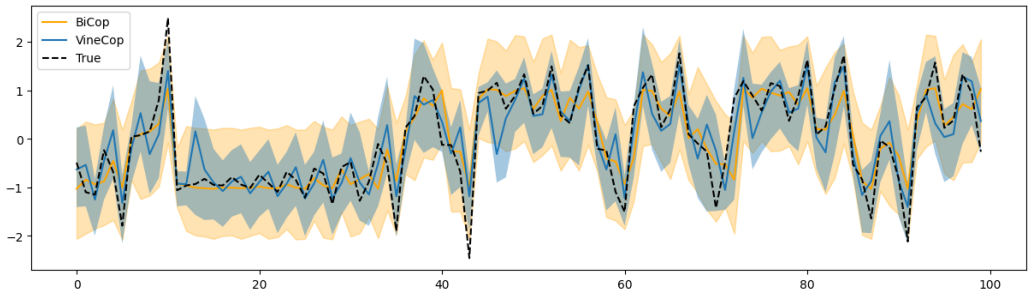


Figure 5.3: Median point forecasts (solid lines) and the 90% credible intervals (shaded areas) for the Lorenz63 system. 100 observations are shown here for demonstration. Forecasts produced by the Recursive-VineCop model have narrower CIs.

It can be seen from the forecast plot that the Recursive-VineCop model greatly reduces the width of the credible intervals of its predictions in comparison to those by the Recursive-BiCop model, and for most of the time the true values lie within its credible intervals. Also, the Recursive-VineCop model is generally able to follow the fluctuation of the true values, while the Recursive-BiCop model constantly underestimates the magnitude of fluctuation. Boxplots of the CRPS for each probabilistic forecasting models and comparisons of metrics are shown in Figure 5.4 and Table 5.1, respectively.

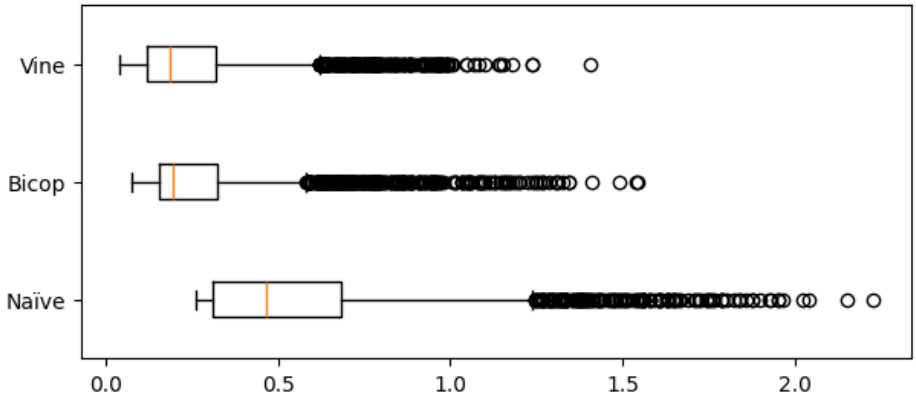


Figure 5.4: Boxplots of CRPS for probabilistic forecasting models on the Lorenz63 system. The Recursive-VineCop model has a lower median and spread of CRPS on both systems.

	Mean CRPS	Std. CRPS	RMSE
VineCop	**0.25716**	**0.20529**	**0.60751**
BiCop	0.29032	0.22456	0.61468
Naïve	0.57152	0.34217	0.91573
Persistence	-	-	0.86572

Table 5.1: Performance metrics for probabilistic forecasting models on the Lorenz63 system. For each metric, lower is better. The Recursive-VineCop model has the lowest average CRPS, standard deviation of CRPS, and the RMSE.

Although the RMSE suggests that the Recursive-VineCop and the Recursive-BiCop models are similar in terms of predictability, the former does yield a lower CRPS, indicating that it can predict the right value with more confidence.

5.1.2 Lorenz96 System

The Lorenz96 model is a multivariate time series comprising two sets of differential equations linking two sets of variables X_k and $Y_{j,k}$, to mimic the dynamics of the extratropical atmosphere (Balwada et al., 2023):

$$\frac{d}{dt}X_k = -X_{k-1}\left(X_{k-2} - X_{k+1}\right) - X_k + F - \left(\frac{hc}{b}\right)\sum_{j=0}^{J-1} Y_{j,k}$$

$$\frac{d}{dt}Y_{j,k} = -cbY_{j+1,k}\left(Y_{j+2,k} - Y_{j-1,k}\right) - cY_{j,k} + \frac{hc}{b}X_k$$

where $X_k, k = 1, \ldots, K$ represents the k-th slow-moving variable and $Y_{j,k}, j = 1, \ldots, J$ denotes the j-th fast-moving, transient variable interacting with X_k. There are therefore $K + J \times K$ variables in total in a system. The numbers of slow-moving variables and fast-moving variables influencing each slow-moving variable can be specified by the user. Other parameters b, c, h, F are treated as constants. Among them, the forcing parameter F governs the magnitude of the chaotic behaviour of the system. In this study, 8 slow-moving variables, each interacting with 32 fast-moving variables, are included in the data-generating process (therefore $K = 8, J = 32$), and we set the constants $b = 10, c = 10, h = 1$, and $F = 20$, following the configuration set by Arnold et al. (2013). Every step in the system corresponds roughly to 5 atmospheric days with respect to predictability under this parametrisation (Arnold et al., 2013). Here we adjust the data generation and collection frequency to mimic data observed and collected every 6 hours, so a 1-step-ahead prediction task corresponds roughly to a 6h prediction in real life. In the system, only the first slow-moving component of the multivariate time series (X_1) is kept as observations, and all the other variables are treated as unobserved. X_1 is therefore a univariate time series. The summary plots of the first component of the Lorenz96 system are presented in Figure 5.5.

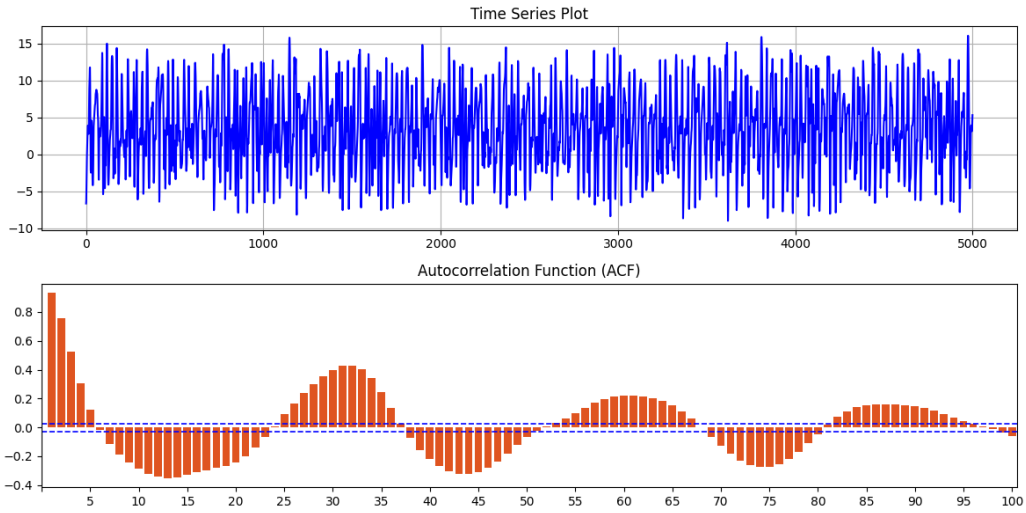


Figure 5.5: Summary plots for the first component of the Lorenz96 system. The upper plot shows the trajectory of the time series, while the lower plot shows the estimates of autocorrelation between different lags. The blue dashed line represents the credible intervals for no autocorrelation.

It can be seen from the summary plots that the Lorenz96 dataset exhibits a seasonal trend and has long memory. For this study the predictors will make 1-step-ahead (6h) predictions. The estimate of the marginal distribution of this system is shown in Figure 5.6. The histogram suggests there is a point of high density in the centre of the distribution, which the estimator fails to capture. However, it successfully capture the overall shape of the distribution and therefore is thought of as a good approximation.

The forecasting results of the two probabilistic forecasting models on the system are shown in Figure 5.7. For the Lorenz96 system, the vine copula selects an observation window size $k = 5$.

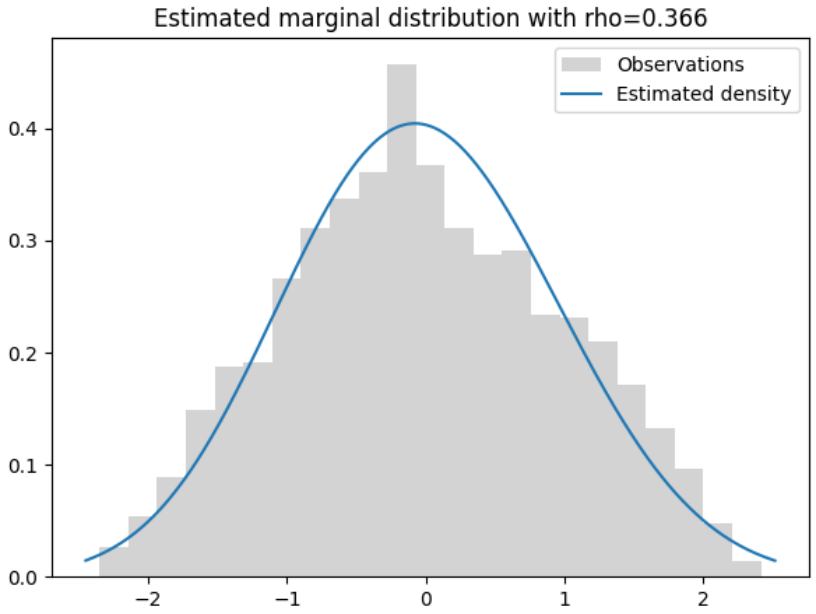


Figure 5.6: Estimate of the marginal distribution of the Lorenz96 system. The estimated density function (blue solid lines) is plotted with the histogram of observations (grey) for comparison.

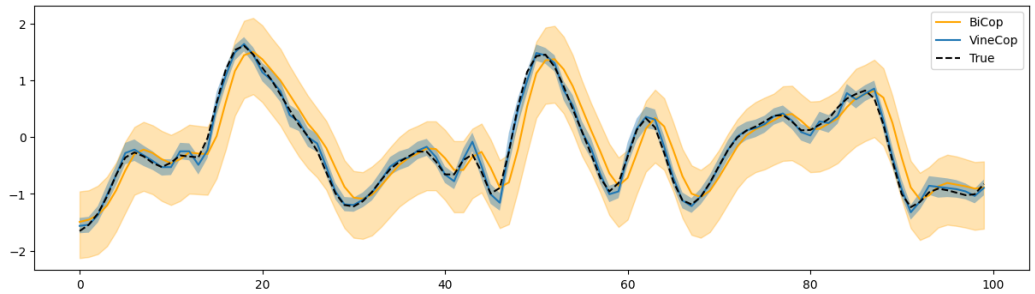


Figure 5.7: Median point forecasts (solid lines) and the 90% credible intervals (shaded areas) for the first component of the Lorenz96 system. 100 observations are shown here for demonstration. Forecasts produced by the Recursive-VineCop model have narrower CIs.

For this dataset, the Recursive-VineCop model can even predict with supreme accuracy and confidence compared to its opponent. Also, it is most obvious in this dataset that the Recursive-BiCop model employs a persistence strategy, with its predicted values mimicking the true values one step ago. Forecasts produced by the Recursive-VineCop model, on the other hand, al-

most overlaps the true values, and the credible intervals also follow the true values tightly. Comparisons of performance metrics presented in Table 5.2 also reveal the same discoveries: The RMSE of the Recursive-BiCop model is similar to that of the persistence model on the Lorenz96 dataset, showing that the bivariate copula model fails to outperform a simplistic persistence model. On the other hand, the Recursive-VineCop model outperforms all rivals by a large margin. The improvement in performance can also be seen in the boxplots in Figure 5.8.

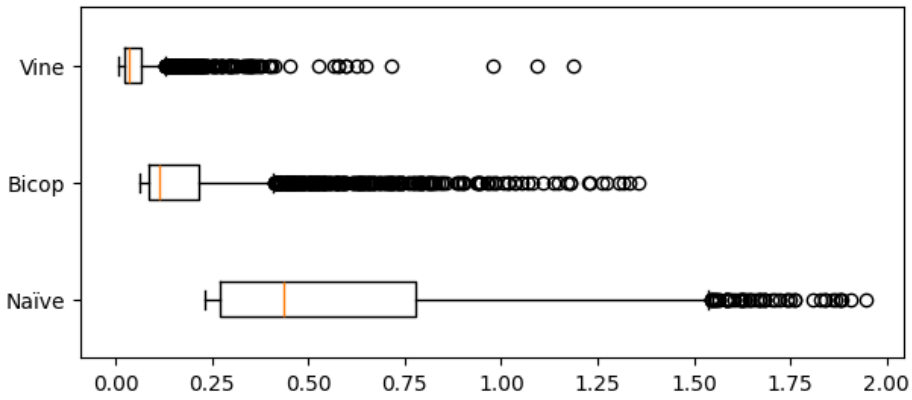


Figure 5.8: Boxplots of CRPS for probabilistic forecasting models on the Lorenz96 systems. The Recursive-VineCop model has a lower median and spread of CRPS on both systems, despite a few outliers.

	Mean CRPS	Std. CRPS	RMSE
VineCop	**0.05983**	**0.07290**	**0.28583**
BiCop	0.19454	0.18626	0.50805
Naïve	0.57972	0.37777	0.90931
Persistence	-	-	0.51633

Table 5.2: Performance metrics for probabilistic forecasting models on the Lorenz96 system. For each metric, lower is better. The Recursive-VineCop model has the lowest average CRPS, standard deviation of CRPS, and the RMSE.

5.2 Real-World Meteorological Dataset

Finally, a real-world dataset is considered for this prediction framework. WeatherBench2 (Rasp et al., 2024) is a project providing various kinds of datasets for researchers to test weather forecasting models. Here we make use of the 500 hPa geopotential height (Z500) data with the finest resolution available (0.25° over both longitude and latitude) at the coordinates of the University of Warwick (52.5°N, 1.5°W). The 500 hPa geopotential height is the altitude needed for the atmospheric pressure to drop to roughly 500 hPa at a given time. The Z500 is useful for making short- to medium-term forecasts about atmospheric states (Weyn et al., 2019; Dueben and Bauer, 2018). For this study, data from 2017-07-31 00:00:00 UTC to 2020-12-31 00:00:00 UTC are collected, with a frequency of 6 hours, which totals up to 5,000 data points. The predictors aim to provide a prediction with a lead of 6 hours (the next immediate value). The summary plots of the data are shown in Figure 5.9.

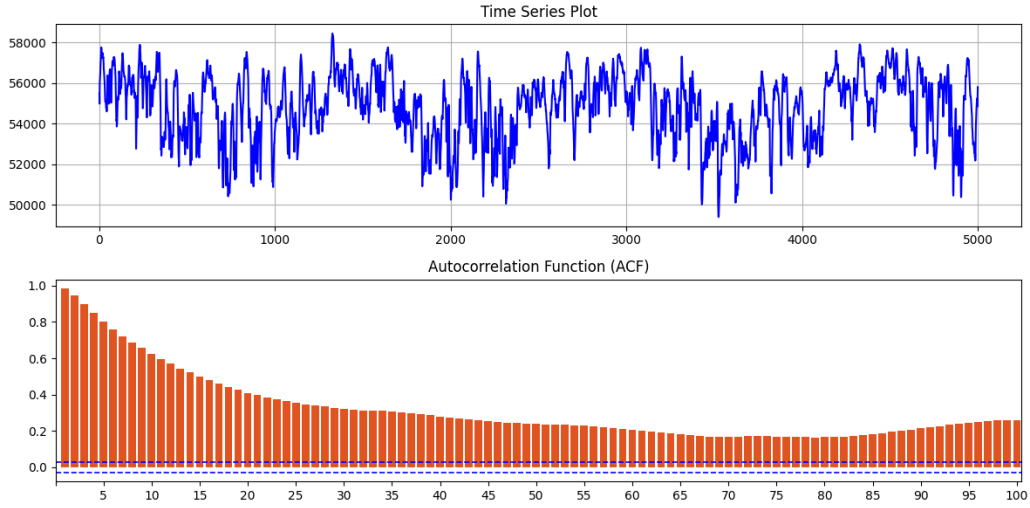


Figure 5.9: Summary plots for the Z500 data at the coordinates of the University of Warwick. The upper plot shows the trajectory of the time series, while the lower plot shows the estimates of autocorrelation between different lags. The blue dashed line represents the credible intervals for no autocorrelation.

As can be seen from the summary plots, the Z500 dataset exhibits a long-memory pattern and a near-one correlation between two adjacent values. The marginal density estimation result is shown in Figure 5.10. Forecasting

results are shown in Figure 5.11. For this dataset, an optimal observation window size $k = 4$ is selected for the vine copula.

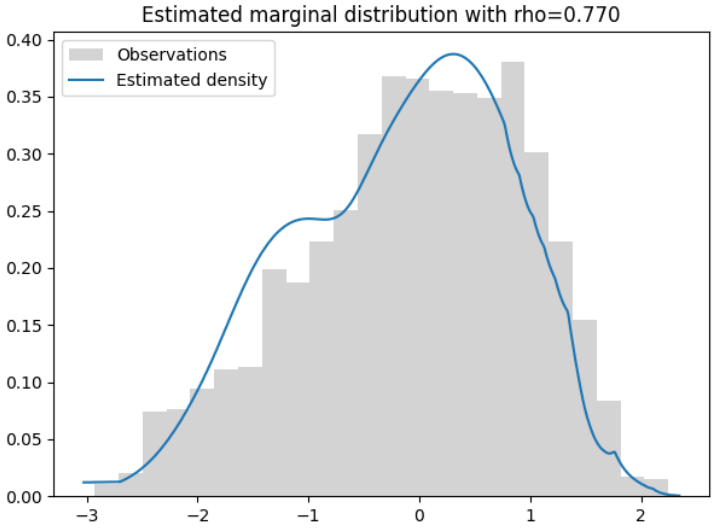


Figure 5.10: Estimate of the marginal distribution of the Z500 data at the coordinates of the University of Warwick. The estimated density function (blue solid line) is plotted with the histogram of observations (grey) for comparison.

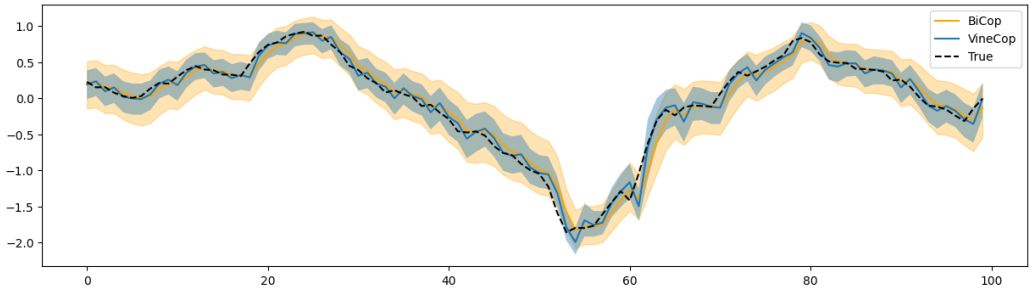


Figure 5.11: Median point forecasts (solid lines) and the 90% credible intervals (shaded areas) for the Z500 data at the coordinates of the University of Warwick. 100 observations are shown here for demonstration. Forecasts produced by the Recursive-VineCop model have narrower CIs.

Comparisons of performance metrics and boxplots are presented in Table 5.3 and Figure 5.12, respectively. Again, the Recursive-BiCop model has similar performance to the persistence predictor, possibly due to the strong corre-

lation between two consecutive values in this dataset, while the Recursive-VineCop model, having the advantage of access to more historical values, performs better in terms of probabilistic forecasting and point forecasting tasks.

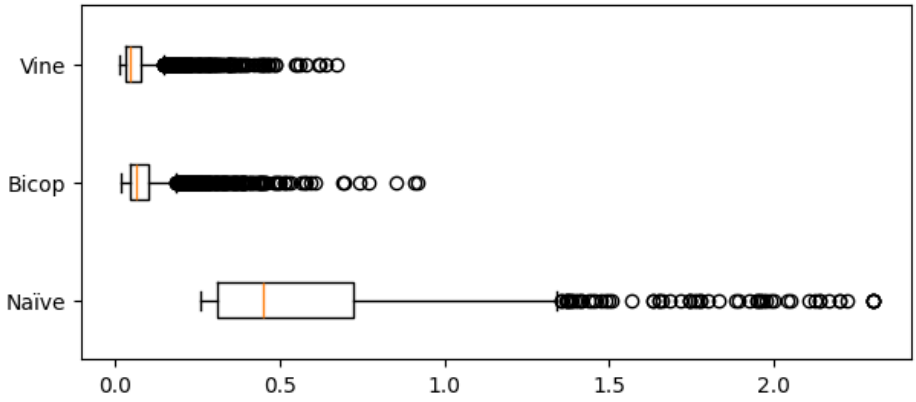


Figure 5.12: Boxplots of CRPS for probabilistic forecasting models on the Z500 data at the coordinates of the University of Warwick. The Recursive-VineCop model has a lower median and spread of CRPS.

	Mean CRPS	Std. CRPS	RMSE
VineCop	**0.06935**	**0.07164**	**0.30843**
BiCop	0.09109	0.08712	0.35249
Naïve	0.56644	0.33881	0.90413
Persistence	-	-	0.35282

Table 5.3: Performance metrics for probabilistic forecasting models on the Z500 data at the coordinates of the University of Warwick. For each metric, lower is better. The Recursive-VineCop model has the lowest average CRPS, standard deviation of CRPS, and the RMSE.

The total computational costs for the Recursive-VineCop model on different datasets (for a single run) are reported in Table 5.4. This includes optimising ρ, optimising the observation window size, fitting the final model, and testing. The model is run on the CPU of a MacBook Air with 16GB RAM and an M1 chip. For each dataset, most of the runtime is spent on finding the optimal ρ and observation window size, as for gradient descent, it takes longer for the ρ to reach the optimal value if the initial value is far away. Also, the optimisation of observation window can also take up a lot of time,

since the vine copula is fitted for every window size. As the window size increases, it takes longer for a vine copula to fit. As can be seen from the table, generally more time is spent on a dataset when the resulting ρ and the optimal observation window size is larger. However, it is still considered a very efficient method given its performance, which takes no more than a few minutes to run.

	AR(3)	Lorenz63	Lorenz96	Z500
Runtime (s)	25.6	160.5	82.4	146.5
Optimal ρ	0.131	0.762	0.366	0.770
Window Size	4	6	5	4

Table 5.4: Total computational costs (seconds) for the Recursive-VineCop model on different datasets for a single run. Optimal ρ and observation window sizes are included for reference.

Chapter 6

Discussion

In this chapter we study the impact of one specific hyperparameter and propose two potential improvements that can be made on the current prediction framework. The potential improvements have been lightly explored in the making of this research project but were not done and thus not reported in the main body of the dissertation. We nonetheless document them here for future research.

Impact of Truncation Levels One hyperparameter not tuned in the studies is the truncation level of the vine copula. Nonetheless, we find that truncation levels can have a major impact on the optimal window size, especially when the imposed vine structure is a D-vine. Figure 6.1 shows an example in which as the specified truncation level increases, the model tends to include more past values (hence a larger window size), and potentially make better predictions, thus yielding a lower CRPS on the validation set. However, the best validation CRPS a model can yield flattens out as the truncation level grows.

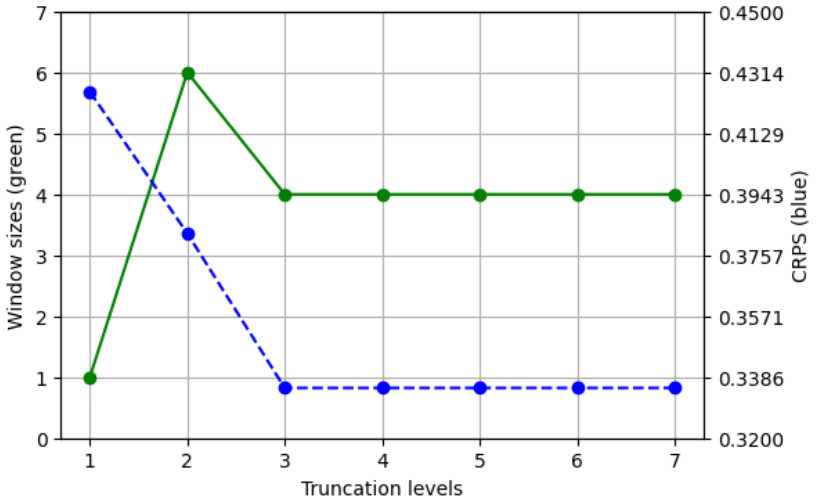


Figure 6.1: Influence of truncation levels on optimal window sizes (green) and the corresponding validation CRPS (blue) for the AR(3) dataset.

When a D-vine is used to model dependencies, truncation levels have a great influence on whether the impact of a value far away in the past on the current state can be recorded and contribute meaningfully to the prediction. For example, when the trees are truncated at the first level, since only adjacent nodes are connected and have their dependencies evaluated by the copulas, incorporating more past values will have little, if any, influence on the dependence structure associated with the leaf. On the other hand, as the truncation level increases, the influence of past values can reach the current state not only through the first tree, but also through higher-level trees where their influence can be recorded with them being conditioning or conditioned variables. Truncation levels are thus a direct measure of complexity of the model, and by setting a truncation level that is too low, higher-order dependencies will not be successfully captured by the model and this will lead to underfitting. When the truncation level is set too high, however, the model is forced to estimate copulas in a very high-dimensional space with not enough data. This can lead to overfitting, rendering the estimates unstable with very high variance. From Figure 6.1, it can be seen that the optimal window size stops growing even with higher truncation levels, and the CRPS also stops dropping. This suggests that even when allowed to estimate in a higher-dimensional space, the model hardly improves its performance by doing so.

The simplifying assumption of the vine copula might also be a reason for

such a phenomenon. When a vine copula is estimated under the simplifying assumption, all dependencies between nodes in every tree are considered to be unconditional on the previous tree. This will make estimating higher-order dependencies harder and less accurate. Plots for all datasets are provided in Appendix B.

We set the truncation level to 5 in the studies, as we found that it generally works well in the tasks discussed in this dissertation. It allows the model to capture more complex dependence structures without risking overfitting. Although truncation levels are not tuned in the study in the interest of time and complexity of the task, they remain an interesting topic to explore and a potential way to further enhance the performance of the predictor.

Incorporating Spatial Dependencies This dissertation puts its focus solely on univariate time series. However, this prediction framework has great potential to be extended to multivariate time series. Although this approach could be easily adapted to multivariate settings by treating each dimension as a univariate time series, it is also sensible to consider the dependencies between different variables. In particular, it will be especially useful to take spatial dependencies between neighbouring points into consideration when making weather forecasts for a larger area, instead of treating them as individual isolated points. Therefore, one possible improvement is to use another vine copula (potentially a C-vine, when there is a natural central point in the dataset) or even a spatial vine copula (a vine copula that incorporates the concept of distance between data points) to capture more information relevant or potentially informative to the prediction task. The use of spatial copulas to monitor environmental hazards such as volcanic activities has been considered in (Puricelli, 2023), and a general multivariate case application has also been explored in (Huk et al., 2024), but it would be beneficial if temporal as well as spatial dependencies could be considered simultaneously. But this is left for future works.

Online Updates of Copulas Although the proposed framework allows for online updates of marginal estimates, in order to utilise information from new data maximally, the ability to update copula estimates in real time is also desired. High-dimensional copulas can be data-hungry, so incoming data can be of great help to stabilise and enhance estimates and reduce variance. However, a recursive copula estimator has yet to be proposed. Hahn et al. (2018) attempted to extend the R-BP algorithm to apply to bivariate data. They show that the recursive update form can be written as (following the

notation in Section 3.3):

$$F_n(y,x) = (1-\alpha_n)F_{n-1}(y,x) + \alpha_n H_\rho\left(F_{n-1}(y,x), F_{n-1}\left(y_n,x\right)\right) H_\rho\left(F_{n-1}(y,x), F_{n-1}\left(y,x_n\right)\right).$$

This can potentially be applied to recursive updates of bivariate copulas, or even multivariate copulas, through updating pair copulas under the vine decomposition. However, how this can actually be done still requires study. The framework can achieve complete online updating if vine copula estimates can also be updated in a similar recursive manner, but this is beyond the scope of this dissertation.

Chapter 7

Conclusions

In this dissertation, we propose a prediction framework that combines an efficient recursive marginal density estimator and a multivariate copula estimator called the vine copula to make probabilistic forecasts. This framework allows for quick online update of marginal density estimates and takes a short time to run and train due to its simple structure and small number of parameters.

We have shown that although the B-RP algorithm assumes data to be iid, it still provides good approximations even when data are temporally dependent. Then, we have shown that when the predictor has the capability to look longer into the past, even if there are some simplifications and restrictions on the vine copula, such as the simplifying assumption and truncation, it can still greatly improve its predictive power. We test the prediction framework on several synthetic and real-world datasets to showcase empirically the power of the predictor. Finally, we acknowledge both the limitations and the potential of the predictor and provided directions for future studies.

Data and Code Availability

The data and code that support the main findings and produce the results in this dissertation are openly available at `https://github.com/kckhchen/Recursive-VineCop`.

Bibliography

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

Anderson, J. L., Collins, N., Gharamti, M., Hendricks, J., Hoar, T., Johnson, B., Kershaw, H., Raczka, B., Raeder, K., and Smith, M. (2004). Data Assimilation Research Testbed. UCAR/NCAR - Computational and Information Systems Laboratory (CISL).

Arnold, H. M., Moroz, I. M., and Palmer, T. N. (2013). Stochastic parametrizations and model uncertainty in the Lorenz '96 system. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1991):20110479.

Balwada, D., Abernathey, R., Acharya, S., Adcroft, A., Brener, J., Balaji, V., Bhouri, M. A., Bruna, J., Bushuk, M., Chapman, W., Connolly, A., Deshayes, J., Fernandez-Granda, C., Gentine, P., Gorbunova, A., Gregory, W., Guillaumin, A., Gupta, S., Holland, M., Johnsson, E., Sommer, J. L., Li, Z., Loose, N., Lu, F., O'gorman, P., Perezhogin, P., Reichl, B., Ross, A., Sane, A., Shamekh, S., Verma, T., Yuval, J., Zampieri, L., Zhang, C., and Zanna, L. (2023). Learning Machine Learning with Lorenz-96.

Bedford, T. and Cooke, R. M. (2001). Probability Density Decomposition for Conditionally Dependent Random Variables Modeled by Vines. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):245–268.

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

Blackwell, D. and MacQueen, J. B. (1973). Ferguson Distributions Via Polya Urn Schemes. *The Annals of Statistics*, 1(2):353–355.

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

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

Brockwell, P. J. and Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing, Cham.

Czado, C. and Nagler, T. (2022). Vine Copula Based Modeling. *Annual Review of Statistics and Its Application*, 9(1):453–477.

Dawid, A. P. (1984). Present Position and Potential Developments: Some Personal Views: Statistical Theory: The Prequential Approach. *Journal of the Royal Statistical Society. Series A (General)*, 147(2):278.

Dawid, A. P. and Musio, M. (2014). Theory and Applications of Proper Scoring Rules. *METRON*, 72(2):169–183.

Dawid, A. P. and Musio, M. (2015). Bayesian Model Selection Based on Proper Scoring Rules. *Bayesian Analysis*, 10(2):479–499.

Dißmann, J., Brechmann, E., Czado, C., and Kurowicka, D. (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59:52–69.

Dueben, P. D. and Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10):3999–4009.

Geenens, G., Charpentier, A., and Paindaveine, D. (2017). Probit transformation for nonparametric kernel estimation of the copula density. *Bernoulli*, 23(3):1848–1873.

Gneiting, T. and Katzfuss, M. (2014). Probabilistic Forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151.

Gneiting, T. and Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Guha, N. and Roy, A. (2022). Stochastic Approximation Algorithm for Estimating Mixing Distribution for Dependent Observations.

Hahn, P. R., Martin, R., and Walker, S. G. (2018). On recursive Bayesian predictive distributions. *Journal of the American Statistical Association*, 113(523):1085–1093.

Hobæk Haff, I., Aas, K., and Frigessi, A. (2010). On the simplified pair-copula construction — Simply useful or too simplistic? *Journal of Multivariate Analysis*, 101(5):1296–1310.

Hofert, M., Mächler, M., and McNeil, A. J. (2012). Likelihood inference for Archimedean copulas in high dimensions under known margins. *Journal of Multivariate Analysis*, 110:133–150.

Huk, D., Zhang, Y., Steel, M., and Dutta, R. (2024). Quasi-Bayes meets Vines.

Joe, H. (1996). Families of m-Variate Distributions with Given Margins and m(m-1)/2 Bivariate Dependence Parameters. *Lecture Notes-Monograph Series*, 28:120–141.

Loader, C. R. (1996). Local likelihood density estimation. *The Annals of Statistics*, 24(4):1602–1618.

Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141.

Lorenz, E. N. (1996). Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, volume 1.

Morales-Napoles, O. (2016). About the number of vines and regular vines on n nodes.

Nagler, T. (2018). **Kdecopula** : An *R* Package for the Kernel Estimation of Bivariate Copula Densities. *Journal of Statistical Software*, 84(7):1–22.

Nagler, T. and Vatter, T. (2025). Pyvinecopulib. Zenodo.

Newton, M. A. (2002). On a Nonparametric Recursive Estimator of the Mixing Distribution. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 64(2):306–322.

Newton, M. A., Quintana, F. A., and Zhang, Y. (1998). Nonparametric Bayes methods using predictive updating. In Bickel, P., Diggle, P., Fienberg, S., Krickeberg, K., Olkin, I., Wermuth, N., Zeger, S., Dey, D., Müller, P., and Sinha, D., editors, *Practical Nonparametric and Semiparametric Bayesian Statistics*, volume 133, pages 45–61. Springer New York, New York, NY.

Pacchiardi, L., Adewoyin, R. A., Dueben, P., and Dutta, R. (2024). Probabilistic Forecasting with Generative Networks via Scoring Rule Minimization. *Journal of Machine Learning Research*, 25(45):1–64.

Palmer, T. N. (2012). Towards the probabilistic Earth-system simulator: A vision for the future of climate and weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 138(665):841–861.

Patton, A. J. (2012). A review of copula models for economic time series. *Journal of Multivariate Analysis*, 110:4–18.

Puricelli, A. (2023). Spatial Vine Copulas: An Application to Satellite Data for Environmental Hazard Detection. Master's thesis, Il Politecnico di Milano.

Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russell, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., Chantry, M., Ben Bouallegue, Z., Dueben, P., Bromberg, C., Sisk, J., Barrington, L., Bell, A., and Sha, F. (2024). WeatherBench 2: A Benchmark for the Next Generation of Data-Driven Global Weather Models. *Journal of Advances in Modeling Earth Systems*, 16(6).

Sethuraman, J. (1994). A Constructive Definition of Dirichlet Priors. *Statistica Sinica*, 4(2):639–50.

Sklar, A. (1959). Fonctions de Répartition à n Dimensions et Leurs Marges. *Publications de l'Institut Statistique de l'Université de Paris*, 8:229–231.

Székely, G. J. and Rizzo, M. L. (2005). A new test for multivariate normality. *Journal of Multivariate Analysis*, 93(1):58–80.

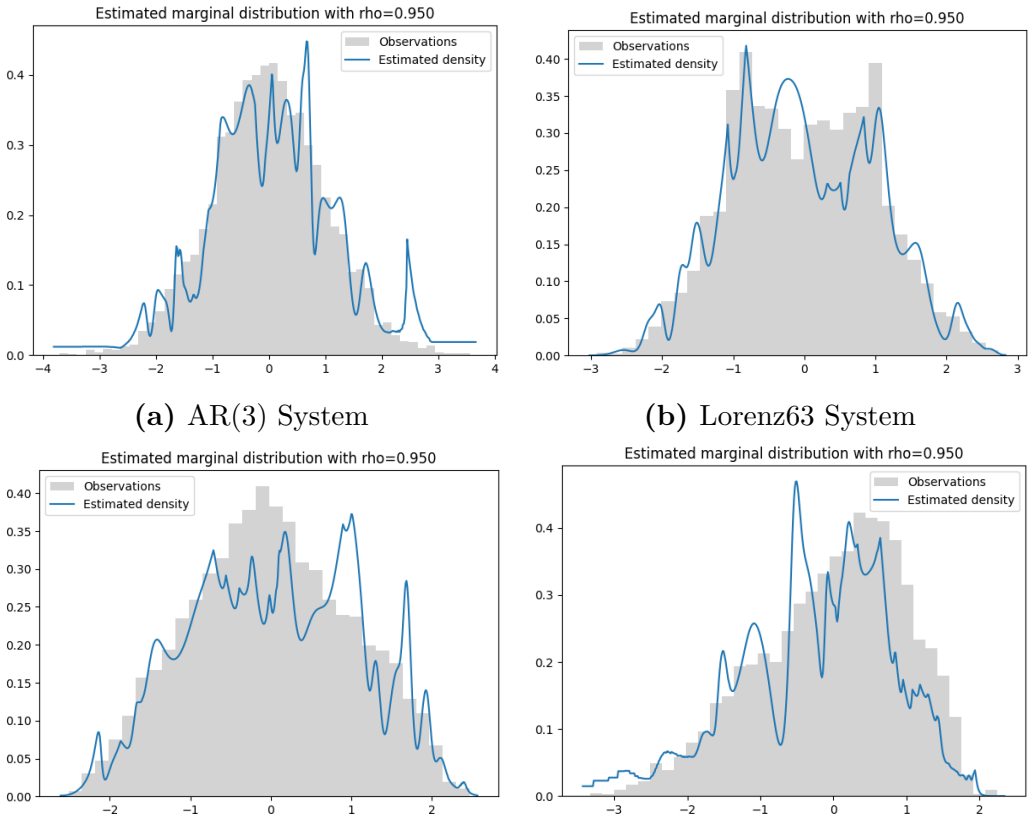
Tokdar, S. T., Martin, R., and Ghosh, J. K. (2009). Consistency of a recursive estimate of mixing distributions. *The Annals of Statistics*, 37(5A):2502–2522.

Weyn, J. A., Durran, D. R., and Caruana, R. (2019). Can Machines Learn to Predict Weather? Using Deep Learning to Predict Gridded 500-hPa Geopotential Height From Historical Weather Data. *Journal of Advances in Modeling Earth Systems*, 11(8):2680–2693.

Zhang, J., Draxl, C., Hopson, T., Monache, L. D., Vanvyve, E., and Hodge, B.-M. (2015). Comparison of numerical weather prediction based deterministic and probabilistic wind resource assessment methods. *Applied Energy*, 156:528–541.

Appendices

A Marginal Estimates with rho = 0.95



(a) AR(3) System

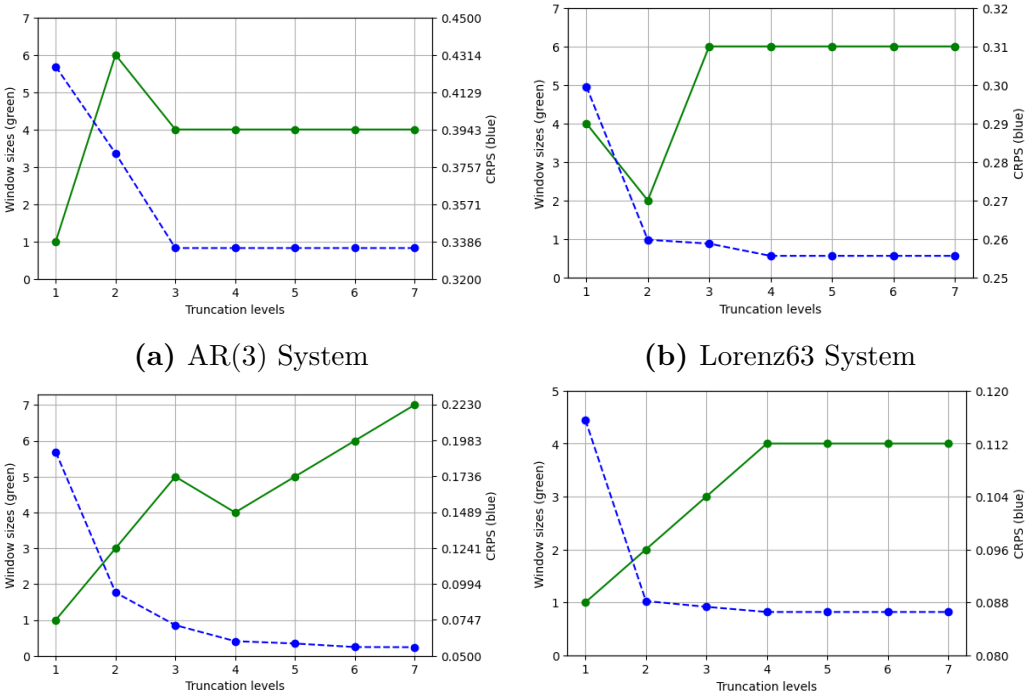
(b) Lorenz63 System

(c) Lorenz96 System

(d) Z500 - University of Warwick

Figure 7.1: Marginal estimates by the R-BP algorithm on the datasets with $\rho = 0.95$. The training data size is 2,500.

B Truncation and Optimal Window Sizes



(a) AR(3) System

(b) Lorenz63 System

(c) Lorenz96 System

(d) Z500 - University of Warwick

Figure 7.2: Impact of truncation levels on different datasets. Green solid lines represent the optimal observation window size for a given truncation level. Blue dashed lines represent the corresponding CRPS.