Learning Vine Copula Models for Synthetic Data Generation

Yi Sun,[1] **Alfredo Cuesta-Infante,**[2] **Kalyan Veeramachaneni**[1]
[1]MIT, [2]Universidad Rey Juan Carlos
yis@mit.edu, alfredo.cuesta@urjc.es, kalyan@csail.mit.edu

Abstract

A vine copula model is a flexible high-dimensional dependence model which uses only bivariate building blocks. However, the number of possible configurations of a vine copula grows exponentially as the number of variables increases, making model selection a major challenge in development. In this work, we formulate a vine structure learning problem with both vector and reinforcement learning representation. We use neural network to find the embeddings for the best possible vine model and generate a structure. Throughout experiments on synthetic and real-world datasets, we show that our proposed approach fits the data better in terms of log-likelihood. Moreover, we demonstrate that the model is able to generate high-quality samples in a variety of applications, making it a good candidate for synthetic data generation.

1 Introduction

The machine learning (ML) community is increasingly interested generative modeling. Broadly, generative modeling consists of modeling either both the joint distribution of data and classes for supervised learning or of modeling only the joint distribution of data for unsupervised learning. In tasks involving classification, a generative model is useful to augment smaller, labeled datasets, which are especially problematic when developing deep learning applications for novel fields (Wang and Perez 2017).

In unsupervised learning (clustering), a generative model supports development of model-driven algorithms (where each cluster is represented by a model). These algorithms scale better than data-driven algorithms, and are typically faster and more reliable. Additionally, these algorithms are crucial tools in tasks that don't easily fit in the supervised/unsupervised paradigm, including, for example, survival analysis - e.g. predicting when a chronically ill person will return to the hospital, or how long will a project last in a Kickstarter (Vinzamuri, Li, and Reddy 2014).

Synthetic data generation - i.e. sampling new instances from joint distribution - can also be carried out by a generative model. Synthetic data has found multiple uses within machine learning. On one hand, it is useful for testing the scalability and the robustness of new algorithms; on the other, it is safe to be shared openly, preserving the privacy and confidentiality of the actual data (Li et al. 2014).

The central underlying problem for generative modeling is to construct a joint probability distribution function, usually high-dimensional and comprising both continuous and discrete random variables. This is often accomplished by using probabilistic graphical models (PGM) such as Bayesian networks (BN) and conditional random fields (CRF), to mention only two of many possible approaches (Jordan 1999; Koller and Friedman 2009). In PGM, the joint probability distribution obtained is simplified due to assumptions on the dependence between variables, which is represented in form of a graph. Thus the major task for PGM is the process of learning such a graph; this problem is often understood as a structure learning task that can be solved in a constructive way, adding one node at a time while attempting to maximize the likelihood or some information criterion. In PGM, continuous variables are quantized before a structure is learned or parameters identified, and, due to this quantization, this solution looses information and scales poorly.

Copula functions are joint probability distributions in which any univariate continuous probability distribution can be plugged in as a marginal. Thus, the copula captures the joint behaviour of the variables and models the dependence structure, whereas each marginal models the individual behaviour of its corresponding variable. In other words, the choice of the copula and the marginals results in the construction of the joint probability distribution directly (Nelsen 2006). However, in practice, there are many bivariate copula families but only a few multivariate ones, with the Gaussian copula and the T-copula being the most prominent. For this reason, these two families have been used extensively, leading to models that most of the time outperform the multivariate normal (MVN). However, these models still assume a dependence structure that may only loosely capture the interaction between a subset of variables. The strongest and most well-known case against the abuse of Gaussian copula can be observed in (MacKenzie and Spears 2014). The key problem pinpointed in that work is that financial quantities are seldom jointly linear, and even if so, the measure of such association, i.e. the correlation, is not stable across time. Therefore, the Gaussian bivariate copula, whose parameter is the correlation between its covariates, is a bad choice. Other copula families depend on non-linear degrees

of association such as Kendall's τ, but it is equally unwise to model the joint behavior of more than two covariates with any of them on the hope that a single scalar will be able to capture all the pairwise dependencies.

Vine copulas provide a powerful alternative in modeling dependence of high-dimensional distributions (Kurowicka and Joe 2011). To explicate, consider the same case presented in (MacKenzie and Spears 2014) under three distinct generative models. Let the multivariate normal distribution (MVN) be the first one, the Gaussian copula next, and finally a vine model. For the sake of simplicity, let us also assume that there are three covariates involved, x_i, for $i = \{1, 2, 3\}$ with probability density functions (PDF) $f_i(x_i)$. Hence, the MVN correlation matrix $\mathbf{R} \in [-1, 1]^{3\times3}$. Since the marginals of MVN are also normal whereas the actual marginals f_i might be very different, samples from the MVN are very likely to represent the actual data poorly. In this case, one would then proceed by finding $\hat{f}_i(x_i)$, the best match with the actual marginals $f_i(x_i)$. Together with the Gaussian copula, a more accurate model is thus constructed and sampled. If covariates are not jointly linear, samples will differ from actual data; for example, when x_1 and x_2 occur jointly in similar rank positions, but not necessarily in linear correlation, Kendall's τ rank correlation is a much better parameter, and this pair of covariates would be better modeled by a copula parameterized by τ, such as the Clayton, the Frank or the Gumbel family, to mention only the most popular ones. This copula would be the first block of the vine model. Following with the example, next we have to plug the third covariate to either the left end or the right end of the tree, the decision is due to some metric such as likelihood, information criteria, goodness-of-fit, etc, with another copula following the same procedure. Thus, the first tree for three covariate would be ended. For the second tree, the copulas (edges) in the previous trees are now nodes that are to be linked. Since we only have two, they can only form one possible bond, and the construction ends for we cannot iterate once more.

The dependence structure in Vine Copula is constructed using bivariate copulas as building blocks; thus, two or more variables can have a completely different behavior than the rest. A vine is represented as a sequence of trees, organized in levels, so they can be considered as a PGM. Another distinctive feature of vine copulas is that the joint probability density factorization resulting from the entire graph can be derived by the chain rule. In other words, theoretically, there are no assumptions about the independence between pairs of variables, and in fact the building block in such a case is the independence copula. However, in practice usually only the top levels of a vine are constructed. Therefore, learning a vine has the cost of learning the graph, tree-by-tree, plus the cost of selecting the function that bonds each pair of nodes. The problem is more challenging when the vine is pruned from one level downwards to the last. Yet this effort is rewarded by a superior and more realistic model.

This paper presents the following contributions. Firstly, we formulate the model selection problem for regular vine as a reinforcement learning (RL) problem and relax the tree-by-tree modeling assumption, where each level of tree is selected sequentially. Moreover, we use long-short term memory (LSTM) networks in order to learn from vine configurations tested long and short ago. Second, as far as we are aware, this work is the first to use regular vine copula models for generative modeling and synthetic data generation. Finally, a novel and functional technique for evaluating model accuracy is a side result of synthetic data generation. We propose that a model can be admitted if it generates data that produces a performance similar to the actual data on a number of ML techniques; e.g. decision trees, SVM, Neural Networks, etc.

The rest of the paper is organized as follows. In section 2, we present the literature related to the topics of this paper. In section 3, we introduce the definition and construction of a regular vine copula model. Section 4 describes our proposed learning approach for constructing regular vine model. In section 5, we apply the algorithm to several synthetic and real datasets and evaluate the model in terms of fitness and the quality of the samples generated.

2 Motivation and Related work

The rise of deep learning (DL) in the current decade has brought forth new machine learning techniques such as convolutional neural networks (CNN), long-short term memory networks (LSTM) or generative adversarial networks (GAN) (Goodfellow, Bengio, and Courville 2016). These techniques outrank the state-of-the-art in problems from many fields, but require large datasets for training, which can be a significant problem given that often collecting data is expensive or time consuming. Even when data is already collected, often it cannot be released due to privacy or confidentiality issues. Synthetic data generation, currently a well researched topic in machine learning, provides a promising solutions for these problems (Alzantot, Chakraborty, and Srivastava 2017; Libes, Lechevalier, and Jain 2017; Soltana, Sabetzadeh, and Briand 2017; Sagduyu, Grushin, and Shi 2018). Generative models - that is, high-dimensional multivariate probability distribution functions - are a natural way to generate data. More recently, the rise of GAN and its variations provides a way of generating realistic synthetic images (Goodfellow, Bengio, and Courville 2016; Ratner et al. 2017).

Copula functions, and PGM involving copulas and vines, have gained momentum in ML since the early proposal of the copula based regression models (Kolev and Paiva 2009) and copula Bayesian networks (Elidan 2010). Gaussian process vine copulas were introduced in (Lopez-Paz, Hernandez-Lobato, and Ghahramani 2013). The Copula Discriminant Analysis (CODA) is a high-dimensional classification method based on the Gaussian Copula proposed in (Han, Zhao, and Liu 2013). The multi-task copula was introduced in (Zhou and Tao 2014) for multi-task learning. A copula approach has been applied to jointly modeling of longitudinal measurements and survival times in AIDS studies (Ganjali and Baghfalaki 2015). A vine copula classifier has performed competitively compared to the four best classification methods presented at the Mind Reading Challenge Competition 2011 (Carrera, Santana, and Lozano 2016).

In this paper we obtain them by means of a copula function based PGM known as Vine. Vines were first introduced in (Bedford and Cooke 2001) as a probabilistic construction of multivariate distributions based on the simple building blocks of bi-variate copulas. These constructions are organized graphically as a sequence of nested undirected trees. Compared to black-box deep learning models, vine copula has better interpretability since it uses a graph like structure to represent correlations between variables. However, learning a vine model is generally a hard problem. In general, there exists $\frac{d!}{2}2^{\binom{d-2}{2}}$ different d-dimensional regular vines with d variables, and $|B|^{\binom{d}{2}}$ different combinations of bivariate copula families where $|B|$ is the size of the candidate bivariate families (Morales-Napoles 2010).

To reduce the complexity of model selection, (Dissmann et al. 2013) proposed a tree-by-tree approach that selects each tree $T_1, ..., T_{d-1}$ sequentially, with a greedy maximum-spanning tree algorithm where edge weights are chosen to reflect large dependencies. Although this method works well in lower-dimensional problems, the greedy approach does not ensure optimal solutions for high-dimensional data. Gruber and Czado (2015, 2018) proposed a Bayesian approach to estimate regular vine structure along with the pair copula families from an arbitrary set of candidate families. However, the sequential, tree-by-tree, Bayesian approach is computationally intensive and cannot be used for more than 20 dimensions. A novel approach to high-dimensional copulas has been recently proposed by Müller and Czado (2018).

In this paper, we reformulate the model selection as a sequential decision-making process, and cast it as an RL problem, which we solve with policy learning (Sutton and Barto 1998). Additionaly, when constructing the Vine, a decision made in the first tree can limit the choices in construction of subsequent trees. Therefore, we cannot assure the markovian property, i.e. that the next state depends only of the current state and current decisions. Such a non-markovian property suggests the use of Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997). LSTM in conjunction with model-free RL was presented in (Bakker 2001) as a solution to non-Markovian RL tasks with long-term dependencies.

3 Regular Vine Copula

In this section we summarize the essential facts about the meaning of the vine graphical model and how it is transformed into a factorization of the copula density function that models the dependence structure. A deeper explanation about vines can be found in (Aas et al. 2009).

According to Sklar's theorem, the copula density function is what it needs to complete the joint probability density of d continuous covariates in case they are not independent. In other words, if the individual behaviour of covariate x_i is given by the marginal probability density function $f_i(x_i)$ for $i = 1, \ldots, d$, then the copula c brings the dependence structure into the joint probability distribution. Such a structure is also independent of every covariate distribution, so it is usually represented as a function of a different set of variables $\{u_i\}$, that are referred to as *transformed covariates*.

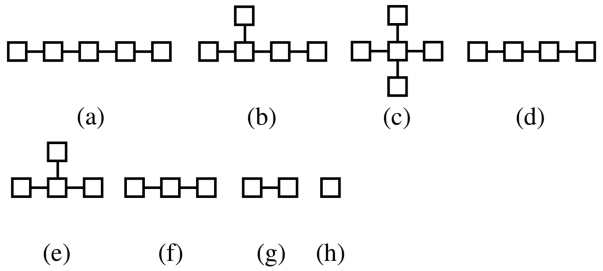

Figure 1: (a-h) All the different layouts of every possible tree in a 5-dim vine. (a-c) correspond to the 1st level, (d-e) correspond to the 2nd level, (f,g,h) are the unique possible layouts for trees in 3rd, 4th and 5th level respectively.

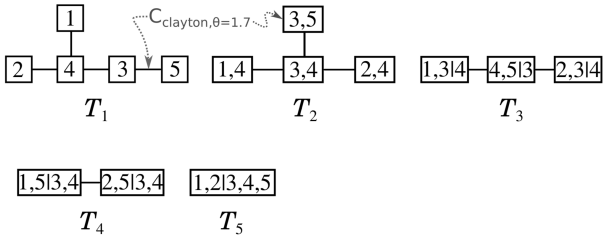

Figure 2: An example of 5-dim vine constructed with layouts {b,e,f,g,h} in Figure 2, and a valid choice of edges in it. For the sake of clarity, there is only one edge with the detail of its copula family and parameter.

Formally expressed, we have:

$$f(x_1, x_2, \ldots, x_d) = c(u_1, u_2, \ldots, u_d) \prod_{i=1}^{d} f_i(x_i),$$

where $u_i = F_i(x_i)$.

There are many families of parametric bivariate copula functions but only a few parametric d-variate ones. Among the latter, Gaussian and T copulas are quite flexible because depend on the correlation matrix of the transformed covariates. A better approach is to use bivariate copulas as building blocks for multivariate copulas. This solution was first presented in (Joe 1996). Bedford and Cooke (2001) later developed a more general factorization of multivariate densities and introduced regular vines.

Definition 3.1 *(R-Vine on d variables) A regular vine on d variables consists of $d-1$ connected trees $T_1, ..., T_{d-1}$, that satisfy the following:*

1. *T_1 consists of the node set $N_1 = \{1, ..., d\}$, where each variable is represented by only one of the nodes; and the edge set E_1, with each edge representing a copula that links two variables.*
2. *For $i = 2, \ldots, d-1$, the tree T_i consists of the node set $N_i = E_{i-1}$ and the edge set E_i*
3. *Each Tree T_k has exactly $d-k$ edges, for $k = 1, \ldots, d-1$. Two nodes in Tree T_1 can always form an edge in T_1. Two nodes in Tree T_i, with $i \geq 2$, can form an edge in*

T_i only if their corresponding edges in Tree T_{i-1} share a common node.

The regular vine copula $(\mathcal{V}, \mathcal{B}_\mathcal{V}, \theta_\mathcal{V})$ has density function defined as:

$$c(u; \mathcal{V}, \mathcal{B}_\mathcal{V}, \theta_\mathcal{V}) = \prod_{T_k \in \mathcal{V}} \prod_{e \in E_k} c_{\mathcal{B}_e}(u_{i(e)|D(e)}, u_{j(e)|D(e)}; \theta_e)$$

where $\mathcal{B}_\mathcal{V}$ is the set of bivariate copula families selected for the Vine $\mathcal{V}$ and $\theta_\mathcal{V}$ the corresponding parameters.

For the sake of clarity, let us consider 5 variables $x_1, \ldots, x_5$, and that they have been already transformed via their corresponding marginals into $u_1, \ldots, u_5$, so $u_k = f_k(x_k)$. According to definition 3.1, a Tree T_1 will have 5 nodes and 4 edges. Therefore, its layout will necessarily be one out of those displayed in Figure 1(a–c). Each one of these layouts leads to many possible and different trees T_1, depending on how variables are arranged in the layout. For this example, let us assume that such an arrangement happens to be the one shown in the tree T_1 of Figure 2, in which it has been explicitly shown that the dependence (edge) between u_3 and u_5 is modeled with a Clayton copula, with parameter $\theta = 1.7$. The layout of the next tree, T_2 may, or may not be one out of Figure 1(d–e). It depends on whether it satisfies the third requirement. In this example, the T_1 layout in Figure 1(a) imposes the layout in T_2 to be exclusively the one in Figure 1(d), resulting in the so called D-Vine. On the other hand, the T_1 layout in Figure 1(b) allows both Figures 1(d) and 1(e) to be layouts for T_2. When arranging the variables in the layout, it is a good practice to write the actual variables, and not the edges of the precedent tree. Then, all variables shared in both nodes of an edge are turned into conditioning variables, and the remaining are conditioned. Eventually, the factorization of the copula density is the product of all the nodes from T_2 on. Thus, copula density of the vine shown in Figure 2 is

$$c(u_1, \ldots, u_5) = c_{14} c_{24} c_{34} c_{35} c_{13|4} c_{45|3} c_{23|4} c_{15|34} c_{25|34} c_{12|345}$$

where the $c_{ij|k..}$ denotes the bi-variate copula density $c\big(F_i(x_i|x_k, ..), (F_j(x_j|x_k, ..)\big)$.

4 Methodology

Model selection for vine is essentially a combinatorial problem over a huge search space, which is hard to solve with heuristic approaches. Dissmann proposed a tree-by-tree approach (Dissmann et al. 2013) , which selects maximum spanning trees at each level according to pairwise dependence. The problem with this locally greedy approach is that there is no guarantee that it will select a structure that is close to the optimal structure.

In this section, we describe in details our learning approach for regular vine copula construction. In order to feed a vine configuration using a neural network based meta learning approach, we need to represent it in a compact way. Complexity arises since the construction of each level of tree depends on previous level's tree. The generated vine also needs to satisfy the following desirable properties:

- **Tree Property** Each level in the vine should be a tree with no cycles.
- **Dependence Structure** The layout of each level tree depends on its previous level's tree.
- **Sparsity** In general, a sparse model where most of edges are independent copulas are preferred.

Next we compare two different representations for a regular vine model: Vector representation and RL representation.

Vector Representation

An intuitive way to embed a vine is to flatten all edges in the vine into a vector. Since the number of edges in the vine is fixed, the network can output a vector of edge indices. The representation proposed is depicted in Figure 3.

Let the set of edges in each tree be T_k, the likelihood of the generated configuration can be computed as:

$$\mathcal{L} = \sum_{T_k \in \mathcal{V}} \sum_{e \in E_k} \log c_{\mathcal{B}_e}(u_{i(e)|D(e)}, u_{j(e)|D(e)}; \theta_e))$$

If the generated configuration contains a cycle at level k, a penalty will be subtracted from the objective function. The penalty will decrease with the level, indicating that since later trees depend on early trees, violation in tree property in early levels will incur a larger penalty. Let the number of cycles at level $k \in \{1 \ldots K\}$ be C_k, thus the penalty due to violation of tree property can be computed as:

$$J_1 = \frac{1}{k} \sum_{k=1}^{K} C_k$$

In practice, most variables are not correlated and the vines tend to be sparse. To avoid over-fitting, we favor smaller edges in the vine graph by adding a penalty term J_2, defined as:

$$J_2 = \frac{1}{\sum_{k=1}^{K} |E_k|}$$

At each training iteration, we are maximizing the following objective function:

$$J_\phi = \mathcal{L} - \lambda J_1 + \mu J_2$$

where λ and μ are hyper-parameters that can be tuned.

Reinforcement Learning Representation

One problem with the vector representation is that the vine configuration generated are not guaranteed to satisfy the tree property at each training step. On the other hand, the construction of the vine model can also be seen as a sequential decision making process: At each step, the set of nodes $\mathcal{N}$ are partitioned into two sets, $\mathcal{N}^L$ and $\mathcal{N}^R$, where $\mathcal{N}^L$ denotes set of nodes that have been already added to the vine, and $\mathcal{N}^R$ denotes set of nodes that are not in the vine. When building tree T_k, in each step a node in $\mathcal{N}^R$ is selected and linked to a node in $\mathcal{N}^L$, which is equivalent to adding a new edge to the tree T_k. Since pair of nodes already in the vine will never be selected at the same time, there won't be cycles forming, and therefore the tree property is maintained throughout the construction.

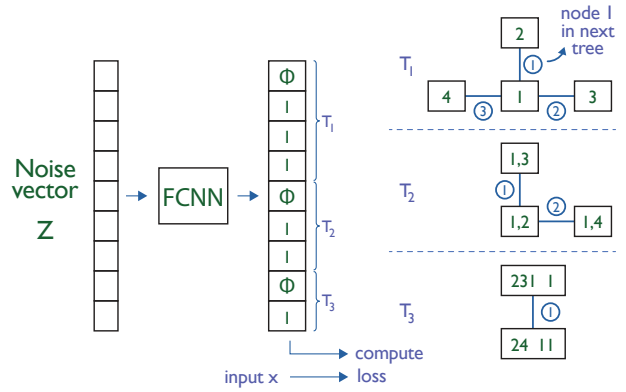

Figure 3: A example of the vector representation of vine. The model takes in a random initial configuration vector and search for the best vector that maximizes the objective function using a fully connected neural network (FCNN). The first element in the output vector ("Φ") means that first node in T_1 is not connected, the second element ("1") means that second node in T_1 is connected to 1, etc. The layout on the right shows how to take the output vector and assembles it into a regular vine.

After we obtain the set of edges E_k for tree k, we repeat the process for the next level of tree. The decision process can be defined as a fixed length sequence of actions of choosing $E_1, E_2, E_3, ...$ where $|E_k| = d - k$. For an untruncated vine with d variables, the total number of edges adds up to $\frac{d(d-1)}{2}$. Motivated by recent developments in RL, we reformulate the problem in its syntax.

States Let e_t be the t-th edge added to the vine model, which consists of a pair of indexes (l_t, r_t). At step t, the states can be represented by the current set of edges in the vine E_t, and the partitioned vertices set $\mathcal{N}_t^L$ and $\mathcal{N}_t^R$. Formally $s_t = (E_t, \mathcal{N}_t^L, \mathcal{N}_t^R)$ is the state representation for the current vine at step t.

Actions The set of possible actions consists of all pairs of nodes that can be added to the current tree. Formally $A_s = \{(l_t, r_t) : i_L \in \mathcal{N}^L, i_R \in \mathcal{N}^R\}$.

Rewards The log likelihood of fitting a data point $\boldsymbol{x}$ to the vine s_t at step t can be decomposed into a set of trees and a set of independent nodes in $\mathcal{N}_t^R$ that have not been added to the tree:

$$\mathcal{L}(x, s_t) = \sum_{e \in E_t, e=(l,r)} \log(c_{\mathcal{B}_e}(u_{l|D(e)}, u_{r|D(e)}; \theta_e) + \sum_{i \in \mathcal{N}_t^R} \log(u_i)$$

where $\mathcal{L}(x, s_0) = \sum_{v \in \mathcal{N}} log(u_v)$.

Then the incremental reward at state S_t can be defined as:

$$\begin{aligned} R_{likelihood}(S_t) &= \mathcal{L}(x, s_t) - \mathcal{L}(x, s_{t-1}) \\ &= c_{B_{e_t}}(u_{l_t|D(e_t)}, u_{r_t|D(e_t)}; \theta_{e_t}) \\ &\quad - \log(u_{r_t}) \end{aligned}$$

where e_t is the newly added edge to the vine.

As before, we add a penalty term to avoid over-fitting. Hence, the total reward is defined as:

$$R = R_{likelihood} + \lambda R_{penalty}$$

Policy Network Let $\mathbb{P}_r$ be the true underlying distribution, and $\mathbb{P}_c$ be the distribution of the network. Our goal is to learn a sequence of edges (pair of indices) E of length $N-1$ such that the following objective function is maximized:

$$J_\phi = \mathbb{E}_{\tau \sim \mathbb{P}_c} \mathbb{E}_{x \sim P_r} \left[\mathcal{L}(x, s_0) + \sum_{t=1}^{N-1} R(s_t) \right]$$

Let π_ϕ be the stochastic policy defined implicitly by network $C_\phi$with parameters ϕ. The policy gradient of the above objective function J_ϕ can be computed as:

$$\nabla_\phi J(\phi) = \mathbb{E}_{\tau \sim \mathbb{P}_c} \mathbb{E}_{x \sim P_r} \left[\sum_{t=1}^{N-1} R(s_t) \nabla_\phi log \pi_\phi(\tau_t | \hat{s}_{t-1}) \right]$$

At each step, the policy network outputs a distribution over all possible actions, from which an action (edge) is sampled. Following standard practice, the expectation is approximated by sampling n data points and m action sequences per data point. Additionally, we use discounted reward and subtract a baseline term to reduce variance of gradients (Greensmith, Bartlett, and Baxter 2001).

Algorithm 1 Vine Learning

procedure TRAINING
  for number of training iterations **do**
    Sample m examples $\{x^{(1)}, ..., x^{(m)}\} \in \mathbf{R}^d$ from real distribution
    Transform examples into $\{F_1(x_1^{(1)}), ..., F_d(x_d^{(1)})\}$
    $E = \emptyset, \mathcal{N}^L = \emptyset, \mathcal{N}^R = \mathcal{N}$
    while $|E|$ <total num of edges **do**
      sample action $i, j \sim \pi(a|s_t)$
      Find $\mathcal{B}^*, \theta^*$ for edge $\{i, j\}$
      $\mathcal{N}^L = \mathcal{N}^L \bigcup \{i\}, \mathcal{N}^R = \mathcal{N}^R \setminus \{j\}$
      $E_t = E_{t-1} \bigcup \{i, j\}$
      Calculate step reward R_t
    update the model by descending its stochastic gradient $\nabla_\phi J(\phi)$

Figure 4: Algorithm for learning vine structure

When constructing the Vine, a decision made in the first tree can affect to the nodes in deeper trees. Therefore, we cannot assure the markovian property, i.e. that the next state depends only of the current state and current decisions. A natural choice will be to adopt LSTM as a solution to this non-Markovian reinforcement learning tasks with long-term dependencies. Once the configuration is determined, we can find copula family and parameter for each edge following the method described in the following section.

Pair Copula Selection

For each edge, we estimate the log-likelihood of the empirical density, and both the fit to the left tail and to the right tail.

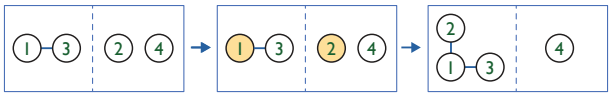

Figure 5: A example of a training step for a 4-dim vine with RL formulation. The set of nodes is partitioned into $\mathcal{V}^L = \{1, 3\}$ and $\mathcal{V}^R = \{2, 4\}$. Node 1 in $\mathcal{V}^L$ and node 2 in $\mathcal{V}^R$ is sampled by the policy network and edge $\{1, 2\}$ is added to the vine.

Then we combine these three measurements in a hard-voting fashion to select a bivariate copula family according to the three-way check methods (Veeramachaneni, Cuesta-Infante, and O'Reilly 2015). The parameter is estimated according to its max likelihood fit after fitting the bivariate copula family. This provides us an approximated reward for adding an edge that guides us through the search space.

Sampling From A Vine

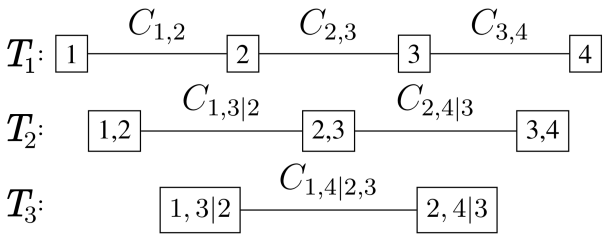

Figure 6: D-Vine of 4 variables.

After learning a vine model from the data, we can sample synthetic data from it. Kurowicka and Cooke first proposed an algorithm to sample an arbitrary element from a regular vine, which they call the Edging Up Sampling Algorithm (Kurowicka and Cooke 2007). The sampling procedure requires a complete vine model of n nodes $X_1, X_2, ..., X_n$, and their corresponding marginal distributions $F_1, F_2, ..., F_n$. Considering the case where we have a D-Vine model with 4 nodes, shown in Figure 6. We start by sampling univariate distributions u_1, u_2, u_3, u_4 from Uniform Distribution over [0,1]. We randomly pick a node to start with, say X_2.

Then the first variable $x_2 \sim X_2$ can be sampled as:

$$x_2 = F_2^{-1}(u_2) \tag{1}$$

After we have x_2, we randomly pick a node connected to X_2. Suppose we pick X_3, recall that the conditional density $f_{3|2}$ can be written as:

$$f_{3|2}(x_3|x_2) = f_3(x_3)c_{2,3}(F_2(x_2), F_3(x_3)) \tag{2}$$
$$= f_3(x_3)c_{2,3}(u_2, u_3) \tag{3}$$
$$= f_3(x_3)c_{3|2}(u_3) \tag{4}$$

Thus, $x_3 \sim X_3|X_2$ can be sampled by:

$$x_3 = F_3^{-1}(C_{3|2}^{-1}(u_3)) \tag{5}$$

where $C_{3|2}$ can be obtained from $C_{2,3}$ in T_1 by plugging in sampled values of u_2.

Similarly, we pick a node that shares an edge with X_3, say X_4. Then $x_4 \sim X_4|X_2, X_3$ can be sampled as:

$$x_4 = F_4^{-1} \circ C_{4|3}^{-1} \circ C_{4|23}^{-1}(u_4) \tag{6}$$

Finally $x1 \sim X_1|X_2, X_3, X_4$ can be sampled as:

$$x_1 = F_1^{-1} \circ C_{1|4}^{-1} \circ C_{1|34}^{-1} \circ C_{1|234}^{-1}(u_1) \tag{7}$$

For a more general vine graph, we can use a modified Breadth First Search to traverse the tree and keep track of nodes that have already been sampled. The general procedure is described in Algorithm 4.

Algorithm 2 Sampling from a regular vine

procedure SAMPLING
- *explore* ← Queue()
- *visited* ← list()
- *start* ← randomly chosen start node
- *explore*.enqueue(start)
- **while** explore not empty **do**
  - *cur* ← *explore*.dequeue()
  - *i* ← len(visited)
  - $x_{cur} = F_{cur}^{-1} \circ C_{cur|visited[i-1]}^{-1} \cdots \circ C_{cur|visited[1,..,i-1]}^{-1}(u_{cur})$
  - **for** s ∈ neighbor(cur) **do**
    - **if** s ∈ visited **then**
      - Continue
    - **else**
      - *explore*.enqueue(s)
  - *visited*.left_append(*cur*)

Figure 7: Algorithm for sampling from the learned vine

5 Experiments

Baselines and Experiment Setup

We compare our models with the tree-by-tree greedy approach (Dissmann et al. 2013), which selects maximum spanning trees at each level according to pairwise dependence, as well as the bayesian approach (Gruber and Czado 2015). To test our approach, we used constructed examples. In this we construct vines manually, sample data from them and then use our learning approach to recreate the vine. We use two of the constructed examples defined in (Min and Czado 2010) and compare the results with both the greedy approach and the Bayesian approach. Second, we used three real data sets and compared the two approaches. The neural network used for creating vines for the vector representation is set-up as fully connected feed forward neural networks with two hidden layers. Each layer uses ReLU as activation function and the output layer is normalized by a softmax layer. The network for the reinforcement learning representation is set up as LSTM. The algorithm is trained over 50 epochs in the experiments.

Constructed Examples

To illustrate a scenario where the greedy approach might fail, we independently generate data sets of size 500 from a

pre-specified regular vine. The fitting abilities of each model are measured by the ratio of log-likelihood of the estimated model and log-likelihood of the true underlying model.

- **Dense Vine** Dense Vine is a 6-dim vine where all bivariate copulas are non-independent. The bivariate families and thetas are listed in Table 6.
- **Sparse Vine** A 6-dim vine where all trees consisted of independent bivariate copulas except the first tree. In other words, the vine is truncated after the first level.

Table 1: Comparison of relative log-likelihood on constructed examples. All results reported here are based on 100 independent trails.

	Dense	Sparse	Dense T1 correct
Dissmann relative loglik(%)	76.6	**101.3**	No
Gruber relative loglik(%)	81.0	100.6	No
Vector relative loglik(%)	80.3	99.8	No
RL relative loglik(%)	**84.2**	100.2	**Yes**

In the dense vine example, the RL vine is the only model able to recover the correct first tree and also achieves the highest relative log-likelihood. In the sparse vine example, all four models achieve similar results, among which Dissmann obtains the highest likelihood. As argued in (Min and Czado 2010), the higher likelihood in Dissmann is achieved at the expense of over-fitting. The two examples demonstrate the fitness of the model is improved by our model under different scenarios. Moreover, for a 6-dimensional data set of size 500, the RL algorithm finishes in approximately 15 minutes with a single GPU.

Real Data

In this section, we apply vine model to real data sets for the task of synthetic data generation. The three data sets that are picked are a binary classification problem, a multi-class classification problem and a regression problem. The batch size used is 64 for breast cancer dataset and 128 for the other two datasets. The three data sets used in experiments are:

- **Wisconsin Breast Cancer** Describes 30 variables computed from a digitized image of a fine needle aspirate (FNA) of a breast mass and a binary variable indicating if the mass is benign or malignant. This dataset includes 569 instances.
- **Wine Quality** This dataset includes 11 physiochemical variables and a quality score between 0 and 10 for red (1599 instances) and white (4898 instances) variants of the Portuguese "Vinho Verde" wine.
- **Crime** The communities and crime dataset includes 100 variables related to crimes ranging from socio-economic data to law enforcement data and an attribute to be predicted (Per Capita Violent Crime) . This dataset has 1994 instances.

To evaluate the quality of the synthetic data, we first evaluate its log likelihood. As shown in 2, synthetic data generated from RL Vine achieves the highest log likelihood per instance in all three datasets.

Table 2: log-likelihood per instance truncated after third tree

	Breast Cancer	Wine	Crime
Dissmann	0.79	0.033	0.26
Vector Rep	0.84	0.037	0.27
RL Vine	**0.91**	**0.045**	**0.31**

Besides log-likelihood, the quality of the generated synthetic data is also evaluated from a more practical perspective. High quality synthetic datasets enable people to draw conclusions and make inferences as if they are working with a real data set. We first use both Dissmann's algorithm and our proposed algorithms to learn copula vine models from a real data set. Later, we generate synthetic data from each model and train models for target variables on the synthetic training set as well as real training set, and use the real testing set to compute the corresponding evaluation metric (F1 score for classification and MSE for regression). For ease of computation, the learned vines are truncated after the third level, which means all pair copulas are assumed to be independent beyond the third level. All results reported are based on 10-fold cross-validation over different splits of training and testing set.

As shown in table 2 and table 3, synthetic data from RL Vine achieves highest F1 score and the results are comparable to real data. For regression data, table 4 demonstrates that synthetic data obtains lowest Mean Squared Error (MSE) among the models. The results shown demonstrate that our proposed model improve the overall model selection and is able to generate reliable synthetic data.

Table 3: F1 score of different end classifiers on breastcancer dataset

	Decision Tree	SVM	5 layer MLP
Real Data	0.92 ± 0.01	0.90 ± 0.02	0.93 ± 0.02
Dissmann	0.77 ± 0.12	0.55 ± 0.05	0.76 ± 0.06
Vector Rep	0.76 ± 0.09	0.62 ± 0.07	0.71 ± 0.05
RL Vine	$\mathbf{0.81} \pm 0.06$	$\mathbf{0.72} \pm 0.03$	$\mathbf{0.79} \pm 0.03$

Table 4: F1 score of different end classifiers on wine quality datset (averaged over 11 classes)

	Decision Tree	SVM	5 layer MLP
Real Data	0.36 ± 0.016	0.11 ± 0.015	0.16 ± 0.016
Dissmann	0.13 ± 0.008	0.09 ± 0.006	0.09 ± 0.006
Vector Rep	0.15 ± 0.011	0.07 ± 0.004	0.08 ± 0.005
RL Vine	$\mathbf{0.21} \pm 0.006$	$\mathbf{0.09} \pm 0.006$	$\mathbf{0.11} \pm 0.008$

Table 5: MSE error of different end classifiers on crime dataset

	Decision Tree	SVM	5 layer MLP
Real Data	0.045 ± 0.004	0.021 ± 0.001	0.023 ± 0.001
Dissmann	0.137 ± 0.016	0.081 ± 0.009	0.124 ± 0.053
Vector Rep	0.112 ± 0.014	0.075 ± 0.006	0.116 ± 0.021
RL Vine	$\mathbf{0.096} \pm 0.011$	$\mathbf{0.072} \pm 0.007$	$\mathbf{0.109} \pm 0.032$

6 Conclusion

In this paper we presented a meta learning approach to create vine models for modeling high-dimensional data. Vine models allow for the creation of flexible structures using bivariate building blocks. However, to learn the best possible model, one has to identify the best possible structure, which necessitates identifying the connections between the variables and selecting between the multiple bivariate copulas for each pair in the structure. We formulated the problem as a sequential decision making problem similar to reinforcement learning, used long-short-term memory networks to simultaneously learn the structure and select the bivariate building blocks. We compared our results to the state of the art approaches and found that we achieve significantly better performance across multiple data sets. We also show that our approach can generate higher quality synthetic data that could be directly used to learn a machine learning model replacing the real data.

Acknowledgment

Dr. Cuesta-Infante is funded by the Spanish Government Research Project TIN-2015-69542-C2-1-R (MINECO/FEDER) and the Banco de Santander grant for the Computer Vision and Image Processing Excellence Research Group (CVIP). Dr. Kalyan Veeramachaneni and Yi Sun acknowledge the generous funding provided by Accenture and National Science Foundation under the grant "*CIF21 DIBBs: Building a Scalable Infrastructure for Data-Driven Discovery and Innovation in Education*", Award # 1443068.

7 Appendix

Experiment

Dense	Copula	Sparse	Copula
$C_{1,2}$	N(0.59)	$C_{1,2}$	N(0.41)
$C_{2,3}$	C(0.71)	$C_{2,3}$	C(0.50)
$C_{3,4}$	C180(0.80)	$C_{3,4}$	C180(0.50)
$C_{3,5}$	N(-0.71)	$C_{3,5}$	N(-0.33)
$C_{3,6}$	T(0.65,3)	$C_{3,6}$	T(0.49,5)
$C_{1,3\|2}$	G(0.75)		
$C_{2,4\|3}$	N(0.41)		
$C_{2,5\|3}$	C270(-0.60)		
$C_{2,6\|3}$	N(-0.37)		
$C_{1,4\|2,3}$	T(0.26,5)		
$C_{1,5\|2,3}$	N(-0.26)		
$C_{1,6\|2,3}$	C90(-0.56)		
$C_{4,6\|1,2,3}$	N(0.13)		
$C_{5,6\|1,2,3}$	C(0.20)		
$C_{5,6\|1,2,3,4}$	G180(0.52)		

Table 6: synthetic vine example. The pair copulas families used are: Gaussian(N) Clayton(C), Gumbel(G), Student's T(T) and their corresponding rotations

References

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

Alzantot, M.; Chakraborty, S.; and Srivastava, M. 2017. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops*, 188–193.

Bakker, B. 2001. Reinforcement Learning with Long Short-term Memory. In *Proc. of the 14th Int. Conf. on Neural Information Processing Systems*, NIPS'01.

Bedford, T., and Cooke, R. 2001. Probability density decomposition for conditionally dependent random variables modeled by vines. *Annals of Mathematics and Artificial Intelligence* 32:245–268.

Carrera, D.; Santana, R.; and Lozano, J. A. 2016. Vine copula classifiers for the mind reading problem. *Progress in Artificial Intelligence* 5(4):289–305.

Dissmann, J.; Brechmann, E.; Czado, C.; and Kurowicka, D. 2013. Selecting and estimating regular vine copula and application to financial returns. *Computational Statistics and Data Analysis* 59:52–69.

Elidan, G. 2010. Copula bayesian networks. In *Advances in Neural Information Processing Systems 23*, 559–567.

Ganjali, M., and Baghfalaki, T. 2015. A Copula Approach to Joint Modeling of Longitudinal Measurements and Survival Times Using Monte Carlo Expectation-Maximization with Application to AIDS Studies. *Journal of Biopharmaceutical Statistics* 25(5):1077–1099.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. The MIT Press.

Greensmith, E.; Bartlett, P. L.; and Baxter, J. 2001. Variance reduction techniques for gradient estimates in reinforcement learning. In *Journal of Machine Learning Research*, volume 5, 1471–1530.

Gruber, L., and Czado, C. 2015. Sequential Bayesian Model Selection of Regular Vine Copulas. *Bayesian Analysis* 10(4).

Gruber, L., and Czado, C. 2018. Bayesian model selection of regular vine copulas. *Bayesian Analysis* 13(4).

Han, F.; Zhao, T.; and Liu, H. 2013. Coda: high dimensional copula discriminant analysis. *J. Mach. Learn. Res.* 14(1):629–671.

Hochreiter, S., and Schmidhuber, J. 1997. LONG SHORT-TERM MEMORY. In *Neural Computation*, 1735–1780.

Joe, H. 1996. Families of m-variate distributions with given margins and m(m-1)/2 bivariate dependence parameters. *Distributions with Fixed Marginals and Related Topics, IMS Lecture Notes - Monograpgh Series* 28:120–141.

Jordan, M. I. 1999. *Learning in Graphical Models*. MIT Press.

Kolev, N., and Paiva, D. 2009. Copula-based regression models: A survey. *Journal of statistical planning and inference* 139(11):3847–3856.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

Kurowicka, D., and Cooke, R. 2007. Sampling algorithms for generating joint uniform distributions using the vine-copula method. *Computational Statistics & Data Analysis* 51(6):2889–2906.

Kurowicka, D., and Joe, H. 2011. *Dependence Modeling: Vine Copula Handbook*. World Scientific Publishing Company.

Li, H.; Xiong, L.; Zhang, L.; and Jiang, X. 2014. Dpsynthesizer: differentially private data synthesizer for privacy preserving data sharing. *Proceedings of the VLDB Endowment* 7(13):1677–1680.

Libes, D.; Lechevalier, D.; and Jain, S. 2017. Issues in synthetic data generation for advanced manufacturing. In *2017 IEEE International Conference on Big Data (Big Data)*, 1746–1754.

Lopez-Paz, D.; Hernandez-Lobato, J.; and Ghahramani, Z. 2013. Gaussian process vine copulas for multivariate dependence. In *JMLR 28(2): Proceedings of The 30th International Conference on Machine Learning*, 10–18. JMLR.

MacKenzie, D., and Spears, T. 2014. 'The formula that killed Wall Street': The Gaussian copula and modelling practices in investment banking. *Social Studies of Science* 44(3):393–417.

Min, A., and Czado, C. 2010. Bayesian inference for multivariate copulas using pair-copula constructions. *Journal of Financial Econometrics* 8(4):511–546.

Morales-Napoles, O. 2010. *Dependence Modeling Vine Copula Handbook*. World scientific. chapter Counting Vines, 189–218.

Muller, D., and Czado, C. 2018. Selection of sparse vine copulas in high dimensions with the lasso. *Statistics and Computing* 29:269–287.

Nelsen, R. B. 2006. *An introduction to copulas*. Springer Series in Statistics, 2nd. edition.

Ratner, A. J.; Ehrenberg, H. R.; Hussain, Z.; Dunnmon, J.; and Christopher, R. 2017. Learning to Compose Domain-Specific Transformations for Data Augmentation. In *Proc. of Neural Information Processing Systems (NIPS) 2017*, volume 30, 3239–3249.

Sagduyu, Y.; Grushin, A.; and Shi, Y. 2018. Synthetic social media data generation. *IEEE Transactions on Computational Social Systems* 1–16.

Soltana, G.; Sabetzadeh, M.; and Briand, L. 2017. Synthetic data generation for statistical testing. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 872–882.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning - an introduction*. MIT Press.

Veeramachaneni, K.; Cuesta-Infante, A.; and OReilly, U.-M. 2015. Copula Graphical Models for Wind Resource Estimation. In *Proc. of the 24th Int. Joint Conference on Artificial Intelligence, IJCAI'15*, 2646–2654.

Vinzamuri, B.; Li, Y.; and Reddy, C. K. 2014. Active learning based survival regression for censored data. In *Proc. of the 23rd ACM Int. Conference on Information and Knowledge Management, CIKM'14*, 241–250.

Wang, J., and Perez, L. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. In *ArXiv*.

Zhou, T., and Tao, D. 2014. Multi-task copula by sparse graph regression. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 771–780.