Generative Modeling With Vine Models

Anonymous Author(s)
Affiliation
Address
email

1 Formulation

The goal is to optimize the construction of the vine copula model in order to minimize the Wasserstein distance between the true underlying distribution and the distribution of model. The construction of the first tree can be seen as a sequential decision making process: in each step we select a variable and link it with a variable in the existing tree. Therefore it is natural to model the process in the syntax of Reinforcement Learning.

Let $\mathbb{P}_r$ be the true underlying distribution and $\mathbb{P}_g$ be the distribution of the vine copula model used to generated synthetic data. The Wasserstein-1 distance is defined as:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]$$

where the supremum is over all the 1-Lipschitz functions. There are two choices that can be made:

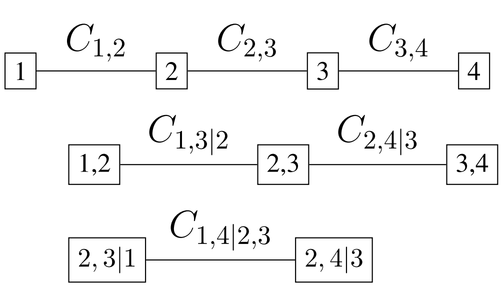
1. The choices of bi-variate copula functions (Is that possible to locally search for the best bi-variate copula function subject to some constraints.)
2. The sequence of variables to join when construct the vine

The construction of the first tree can be seen as a sequential decision making process: in each step we select a variable and link it to a variable in the existing tree.

For a fix generator, the discrimator solves the following optimizatin problem:

2 Related Work

3 Sampling



Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

The sampling procedure requires a complete vine model of n nodes $X_1, X_2, ..., X_n$, and their corresponding marginal distribution $F_1, F_2, ..., F_n$. Considering the case where we have a D-Vine model with 4 nodes. We start by sampling uni-variate distribution 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:

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

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 below:

Algorithm 1 Generative Modeling with Vine Models

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*)

4 Experiment

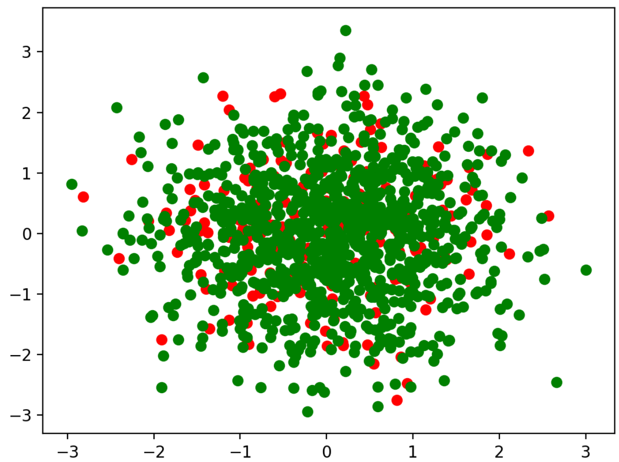


Figure 1: Independent Gaussian

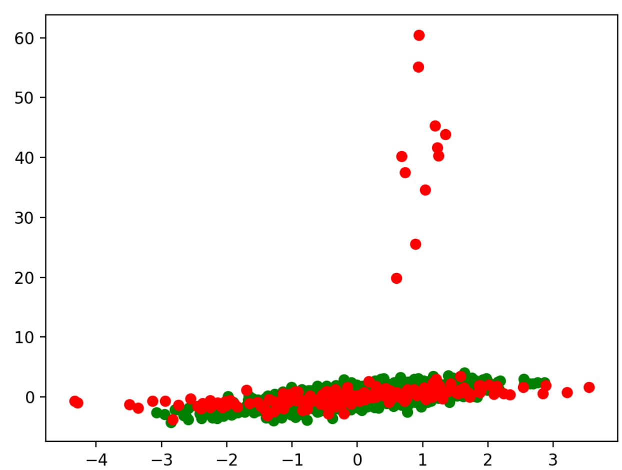


Figure 2: Correlated Gaussian