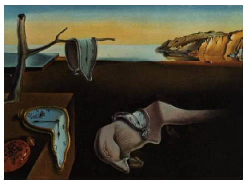

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Code Snippet II. *doi: 10.18637/jss.v000.i00*

Faster estimation of High-Dimensional Vine Copulas with Automatic Differentiation

Hongbo Dong and **Wei Li** and **Haijun Li**

Abstract

Vine copula is an important tool in modeling dependence structures of continuous-valued random variables. The maximum likelihood estimation (MLE) for vine copulas has long been considered computationally difficult in higher dimensions, even in 10 or 20 dimensions. Current computational practice, including the implementation in the state-of-the-art R package **VineCopula**, suffers from the bottleneck of slow calculations of gradients for the log-likelihood function. We show that by using techniques and tools developed in *automatic differentiation*, gradients of the log-likelihood function can be accurately calculated in orders of magnitude faster than the current practice. This change immediately accelerates the MLE for vine copulas by 1-2 orders of magnitudes in moderately high dimension. Furthermore, implementation with automatic differentiation is much simpler, e.g., only the objective function evaluation needs to be implemented, and higher order derivatives are readily available. We implement the case of D-Vines with Clayton copulas, and demonstrate such advantages over the current implementation in **VineCopula**. The purpose of this paper is to further advocate the usage of automatic differentiation tools, which appear to be less known or used in the statistical community.

Keywords: vine copula, maximum likelihood estimation, automatic differentiation, C++.

1. Introduction

Copula is a powerful tool to model the dependence structure of multivariate continuous random variables, especially when the usual multivariate normality is in question. A copula is a multivariate distribution with the *cumulative distribution function* (CDF) $C(x_1, ..., x_d)$, whose one-dimensional marginals are all uniform distributions on $[0, 1]$. The Sklar's theorem Sklar (1959) states that every multivariate distribution F with marginals $F_1, ..., F_d$ can be written as

$$F(x_1, ..., x_d) = C(F_1(x_1), ..., F_d(x_d)),$$

for some d-dimensional copula C. Copula models enable the separated analysis of marginals and dependence structure, and are widely used in applications of risk analysis and management, such as in finance and actuarial science, e.g., McNeil, Frey, and Embrechts (2014).

Many parametric classes of bivariate copulas have been proposed and analyzed, including the Gaussian, the Student, the Clayton, and the Gumbel copula. See Joe (1997) for a review. In higher dimensions, few options are available. The *vine copula* proposed and analyzed by Joe (1996); Bedford and Cooke (2001b, 2002); Kurowicka and Cooke (2006); Aas, Czado, Frigessi, and Bakken (2009) is a flexible approach to construct higher-dimensional copulas according to a sequence of trees, by using bivariate copulas as building blocks. We describe the main construction here.

A "regular vine" $\mathcal{V}$ in d variables is the set of $d-1$ number of *trees* $\{\mathcal{T}_1, ..., \mathcal{T}_{d-1}\}$, where each $\mathcal{T}_i$ is characterized by a node set N_i and an edge set E_i, satisfying the following conditions,

1. $N_1 = \{\{1\}, ..., \{d\}\}$; for each $i = 1, ..., d-2$, $N_{i+1} = \{a \cup b \mid \forall (a, b) \in E_i\}$;
2. for $i = 1, ..., d-2$, $(a, b) \in E_{i+1}$ only when $|a \Delta b| = 2$, where Δ denotes the symmetric difference of two sets a and b, and $|\cdot|$ denotes the cardinality of a set.

A simple special case is called the D-vine, where the first tree $\mathcal{T}_1$ is a line graph with edge set $E_1 = \big\{(\{1\},\{2\}), (\{2\},\{3\}), ..., (\{d-1\},\{d\})\big\}$ up to permutations of $\{1, ..., d\}$. All subsequent trees are then completely determined. For example the second tree $\mathcal{T}_2$ has edge set $E_2 = \{(\{1,2\},\{2,3\}), (\{2,3\},\{3,4\}), ..., (\{d-2,d-1\},\{d-1,d\})\}$.

For each $e \in E_i, 1 \le i \le d-1$, the *conditioning set* of $e \in E_i, 1 \le i \le d-1$, denoted by D_e, is the intersection $j \cap k \subseteq \{1, ..., d\}$, where j, k are the two nodes in N_i associated with e. The *conditioned set* for each of the node j associated to $e \in E(\mathcal{V})$ (where $E(\mathcal{V}) := \cup_{1 \le i \le d-1} E_i$) is $K_{j,e} := j \setminus D_e$. Note that by construction, $K_{j,e}$ for any $j \in e, e \in E(\mathcal{V})$ is a singleton. In the previous D-vine example, if $e = (\{1,2,3\},\{2,3,4\}) \in E_3$, let $j = \{1,2,3\}$ and $k = \{2,3,4\}$, then $D_e = \{2,3\}$, $K_{j,e} = \{1\}$ and $K_{k,e} = \{4\}$.

Given a fixed regular vine $\mathcal{V}$, along with conditional copulas density functions $c_{e_1,e_2|D_e}(\cdot,\cdot)$ for each $e \in E(\mathcal{V})$ (where e_1 and e_2 are the two conditioned sets of e), a multivariate distribution with *probability density function* (PDF) $f = f(x_1, ..., x_d)$ may be written as

$$f = \prod_{i=1}^{d} f_i(x_i) \prod_{e \in E(\mathcal{V})} c_{e_1,e_2|D_e}(F_{e_1|D_e}, F_{e_2|D_e}).$$

Each f_i is the i-th univariate PDF of f and $F_{e_i|D_e}$ is the conditional marginal CDF for $e \in E(\mathcal{V}), i = 1, 2$. To facilitate sampling and inference, it is usually assumed that the conditional copula $c_{e_1,e_2|D_e}$ itself does not depend on variables in D_e (although $F_{e_i|D_e}$ does). Therefore we use $c_{e_1,e_2}(\cdot,\cdot)$ instead of $c_{e_1,e_2|D_e}(\cdot,\cdot)$ from now on. Further assuming all of the one-dimensional margins are uniform $U[0,1]$, and $c_{e_1,e_2|D_e}(u,v) = c_{e_1,e_2}(u,v;\theta_{e_1,e_2})$ with parameter θ_{e_1,e_2}, the log-likelihood function can be evaluated recursively, i.e.,

$$\log f = \sum_{i=1}^{d-1} \sum_{e \in E_i} \log c_{e_1,e_2}(F_{e_1|D_e}, F_{e_2|D_e}; \theta_{e_1,e_2}).$$

For any $i > 1$, Joe (1996) showed that

$$F_{e_j|D_e} = \frac{\partial C_{e_j,k|D_e\setminus\{k\}}(F_{e_j|D_e\setminus\{k\}}, F_{k|D_e\setminus\{k\}}; \theta_{e_j,k})}{\partial F_{k|D_e\setminus\{k\}}}, \quad \textit{for any } k \in D_e,$$

where $C_{\cdot|\cdot}(\cdot|\cdot)$ is the corresponding CDF of correponding copulas. By the tree construction there is always a proper choice of $k \in D_e$ such that $D_e \setminus \{k\}$ is the conditioning set of some edge $e' \in E_{i-1}$. Therefore for any given data point $(x_1, ..., x_d)$ the log-likelihood function can be evaluated by recursively applying the corresponding PDF and partial derivative of CDF for bivariate copulas.

For efficient optimization the gradient information of the log-likelihood function is highly desired, especially when the number of parameters to be estimated is large, e.g. ≥ 10. In principle the gradient of the log-likelihood function can be evaluated recursively by applying chain rules to the recursive formulation above, with manually derived function forms of the derivatives of the PDFs and partial derivatives of CDFs. In fact this has been the main strategy used in the literature and implemented softwares Stöber and Schepsmeier (2013); Schepsmeier and Stöber (2014). However for higher-dimensional vine copulas it is highly non-trivial to avoid unnecessary computation in this process. For example, computed values for lower-level conditional distributions $F_{\cdot|\cdot}$ can be re-used in the evaluation of multiple entries of the gradient. We show that *automatic differentiation* provides an easily implementable solution for gradient evaluation, and accelerates the overall MLE for vine copula models significantly. Computational routines for higher-order derivatives are also available, if needed.

Automatic differentiation is different from *numerical differentiation* and *symbolic differentiation*, it uses a "computational representation" of a function (e.g., a computational routine to evaluate the function value) to either produce code for calculating derivatives, or keep a record of the computation steps of function evaluations, and use it when calculating derivatives later. It is based on the observation that any function evaluation can be represented (by adding intermediate variables) with a directed graph, whose nodes are (input, output and intermediate) variables and arcs representing elementary arithmetic operations. Pointers to numerous survey papers, publications on methods and applications of automatic differentiations, as well as software tools can be found at the website `http://www.autodiff.org/`. A high-level description of algorithms for automatic differentiation, including the *forward mode sweep* and *backward mode sweep*, can be found in Chapter 8 in Nocedal and Wright (1999).

Many different automatic differentiation tools are available, with differences in supported programming languages and implementation strategies. A recent survey can be found at Baydin, Pearlmutter, Radul, and Siskind (2015). In the next section we will describe our implemented R package using C++ (through package **Rcpp**) with the automatic differentiation library **ADOL-C**.

2. Maximum Liklihood Estimation for Vine Copulas with ADOL-C

We implement the special case of D-Vine with Clayton copula in our attached package **DVineAD**. While extensions to general R-vines as well as other bivariate copula families are straightforward, it is not our intention to initiate another "full-fledged" vine copula package. We wish to demonstrate that automatic differentiation techniques can be easily exploited in the computation of vine copulas. Our core code consists of around 200 lines C++ code, the majority of which are for memory handling and data structures.

We employ the automatic differentiation package **ADOL-C** in our implementation. Source code and documentation for **ADOL-C** are accessible through the COIN-OR project (`https://projects.coin-or.org/ADOL-C`). This C++ package utilize *operator overloading*, therefore it is easy to apply to existing C or C++ code with minimum adaptation (mainly expressing input and output variables of the function to be differentiated with data type `adouble`, a class implemented in **ADOL-C**). In the program execution, information of steps of the function evaluation is recorded in a "tape", which is in the form of a file or a data set in the memory buffer. Such tapes will be used in function calls for gradient and higher order derivatives.

In order to successfully install our implemented package the user needs to have the **ADOL-C** software installed. Furthermore, path information of such installation must be put into the file `DVineAD/src/Makevars`. The included `Makevars` file contains the following lines.

```
## User defined libs
PKG_LIBS= -L/home/hdong/lib -ladolc -Wl,-rpath=/home/hdong/lib
## Pre-processor flags
PKG_CPPFLAGS= -I/home/hdong/include
```

The path `/home/hdong/lib` is to be changed to a folder containing the **ADOL-C** library file, e.g., `libadolc.so`, and the path `/home/hdong/include` should be a path containing a folder `adolc` with **ADOL-C** header files, e.g., `adolc.h`.

A short example to simulate samples with **VineCopula**, and perform estimation using our implemented `DVineMLE` function is given below. This file is also included in the folder `DVineAD/tests/` of our **DVineAD** package.

```
library("DVineAD")
library("VineCopula")

d <- 4
TruncLevel <- 2
dd <- (2*d-TruncLevel-1)*TruncLevel/2
par <- c(1, 2, 3, 0.9, 0.8)
startpar <- runif(dd)

# Simulate data with VineCopula (use independence copula in level 3)
family <- c(rep(3,dd), rep(0, d*(d-1)/2 - dd))
parFill <- c(par, rep(0, d*(d-1)/2 - dd))
RVM <- D2RVine(1:d, family, parFill)
simdata <- RVineSim(400, RVM)

# Establish Tape for function and gradient evaluation in MLE
DVineAD_EstablishTape(simdata, par, TruncLevel)
res <- DVineMLE(startpar, trace=1)
print(res)
```

```
iter   10 value -567.047218
final  value -567.047218
converged
$par
[1] 1.107393 2.071233 3.267004 1.039231 0.745096
```

3. Computational Results

We compare our implementation with current state-of-the-art R package **VineCopula** downloaded at `https://github.com/tnagler/VineCopula`[1] on simulated data sets in various dimension d. Our computational results are generated with the R script `tests/CompareTime.R` contained in our package. We summarize our computational results in the following two tables. Columns labeled with T(loglik), T(grad) and T(MLE) report the total elapsed time (in seconds) needed to evaluate the log-likelihood function, the gradient, and to perform the full MLE with optimization routine `optim` (with algorithm choice L-BFGS-B). For **DVineAD**, T(tape) records the time needed to established the initial tape, which is done only once for each data set. Table 1 contains comparison with 200 samples and truncation level (TL) 3, meaning that all bivariate copulas associated with edges in E_i (i>3) are independence copulas. Our implementation is much faster for larger d, and can be more than an order of magnitudes faster when $d = 60$. The speedup is even more significant when the vine copula model is dense, i.e., there is no truncation or $TL = d - 1$. As reported in Table 2, a 40 times speedup is observed for $d = 60$. A clear message is that the ratio T(grad)/T(loglik) does not increase with d, which matches the folklore in the automatic differentiation community.

Finally, we remark that the tape size may not be small. For the case $d = 60$ without truncation we observed the total tape size is about 1 gigabyte (apparently tape size depends linearly on the sample size). In the total elapsed time, around 10-20% (e.g., judging by the system time reported by the R system) is spent on reading/writing the tape. Therefore on a computing system with slow file I/O speed, caution may be needed in implementation. Alternatively, other automatic differentiation tools that generate source code for evaluating gradients, or the “tapeless” mode of **ADOL-C**, may be considered in such cases. In a “big data” scenario, i.e., very large n, optimization algorithms that only use a small portion of data in function/gradient evaluations, such as the stochastic gradient methods Bottou, Curtis, and Nocedal (2016), may be the only viable solution.

	VineCopula			**DVineAD**			
d	T(loglik)	T(grad)	T(MLE)	T(tape)	T(loglik)	T(grad)	T(MLE)
10	1.7E-1	5.6E-2	2.0E+0	9.8E-2	2.6E-2	9.1E-2	2.2E+0
20	5.7E-2	2.7E-1	1.2E+1	1.1E-1	5.5E-2	1.4E-1	5.7E+0
30	1.2E-1	8.9E-1	4.2E+1	1.7E-1	8.5E-2	2.0E-1	9.1E+0
40	2.1E-1	2.6E+0	1.2E+2	2.3E-1	1.2E-1	2.8E-1	1.5E+1
50	3.4E-1	4.5E+0	2.6E+2	2.9E-1	1.4E-1	3.4E-1	2.3E+1
60	4.8E-1	7.9E+0	4.0E+2	3.5E-1	1.8E-1	4.2E-1	2.6E+1

Table 1: Time comparison, $n = 200$, $TL = 3$

References

Aas K, Czado C, Frigessi A, Bakken H (2009). “Pair-copula Constructions of Multiple Dependence.” *Insurance: Mathematics and Economics*, **44**(2), 182–198.

[1]Accessed on Jan. 15, 2017

	VineCopula		**DVineAD**		
d	T(loglik)	T(grad)	T(tape)	T(loglik)	T(grad)
10	2.1E-2	1.5E-1	1.5E-1	5.2E-2	1.6E-1
20	8.9E-2	2.4E+0	4.5E-1	2.3E-1	5.5E-1
30	2.0E-1	1.3E+1	1.1e+0	5.3E-1	1.3E+0
40	3.7E-1	4.5E+1	2.0e+0	9.6E-1	2.3E+0
50	8.8E-1	1.1E+2	3.1e+0	1.5E+0	3.6E+0
60	1.0E+0	2.3E+2	4.5E+0	2.3E+0	5.6E+0

Table 2: Time comparison, $n = 200$, no truncation

Baydin AG, Pearlmutter BA, Radul AA, Siskind JM (2015). "Automatic Differentiation in Machine Learning: A Survey." arXiv:1502.05767[cs.SC].

Bedford T, Cooke RM (2001b). "Probability Density Decomposition for Conditionally Dependent Random Variables Modeled by Vines." *Annals of Mathematics and Artificial Intelligence*, **32**, 245–268.

Bedford T, Cooke RM (2002). "Vines - a New Graphical Model for Dependent Random Variables." *Annals of Statistics*, **30**, 1031–1068.

Bottou L, Curtis FE, Nocedal J (2016). "Optimization Methods for Large-Scale Machine Learning." https://arxiv.org/abs/1606.04838.

Joe H (1996). "Families of m-Variate Distributions with Given Margins and $m(m-1)/2$ Bivariate Dependence Parameters." In L Rüschendorf, B Schweizer, MD Taylor (eds.), *Distributions with Fixed Marginals and Related Topics*.

Joe H (1997). *Multivariate Models and Multivariate Dependence Concepts*. Chapman and Hall/CRC.

Kurowicka D, Cooke RM (2006). *Uncertainty Analysis with High Dimensional Dependence Modelling*. New York: Wiley.

McNeil AJ, Frey R, Embrechts P (2014). *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press.

Nocedal J, Wright S (1999). *Numerical Optimization*. Springer-Verlag, New York.

Schepsmeier U, Stöber J (2014). "Derivatives and Fisher Information of Bivariate Copulas." *Statistical Papers*, **55**(2), 525–542.

Sklar A (1959). "Fonctions de Répartition à n Dimensions et Leurs Marges." *Publ. Inst. Stat. Univ. Paris*, **8**, 229–231.

Stöber J, Schepsmeier U (2013). "Estimating Standard Errors in Regular Vine Copula Models." *Computational Statistics*, **28**(6), 2679–2707.

Affiliation:

Hongbo Dong,
Department of Mathematics and Statistics
Washington State University
Pullman, WA, 99164, USA
E-mail: hongbo.dong@wsu.edu
URL: http://www.math.wsu.edu/math/faculty/hdong/welcome.php

Wei Li,
Department of Mathematics and Statistics
Washington State University
Pullman, WA, 99164, USA
E-mail: wei.li@wsu.edu

Haijun Li,
Department of Mathematics and Statistics
Washington State University
Everett, WA, 98201, USA
E-mail: lih@math.wsu.edu
URL: http://www.math.wsu.edu/faculty/lih/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Code Snippet II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd