5-2018

Algorithmic Need for Subcopulas

Thach N. Nguyen
Banking University of Ho Chi Minh City, ajeb@buh.edu.vn

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Nguyen Hoang Phuong
Thang Long University, nhphuong2008@gmail.com

Comments:
Technical Report: UTEP-CS-18-49
Nguyen T.N., Kosheleva O., Kreinovich V., Nguyen H.P. (2019) Algorithmic Need for Subcopulas. In: Kreinovich V., Sriboonchitta S. (eds) Structural Changes and their Econometric Modeling. TES 2019. Studies in Computational Intelligence, vol 808. Springer, Cham
https://doi.org/10.1007/978-3-030-04263-9_13

Algorithmic Need for Subcopulas

Thach Ngoc Nguyen, Olga Kosheleva, Vladik Kreinovich, and
Hoang Phuong Nguyen

Abstract One of the efficient ways to describe the dependence between random variables is by describing the corresponding copula. For continuous distributions, the copula is uniquely determined by the corresponding distribution. However, when the distributions are not continuous, the copula is no longer unique, what is unique is a *subcopula*, a function $C(u,v)$ that has values only for some pairs (u,v). From the purely mathematical viewpoint, it may seem like subcopulas are not needed, since every subcopula can be extended to a copula. In this paper, we prove, however, that from the algorithmic viewpoint, it is, in general, not possible to always generate a copula. Thus, from the algorithmic viewpoint, subcopulas are needed.

1 Formulation of the Problem

Copulas: a brief reminder. There are many ways to describe a probability distribution of a random variable:

- we can use its probability density function (pdf),
- we can use its moments,
- its cumulative distribution function (cdf), etc.

Most of these types of descriptions are not always applicable:

Thach Ngoc Nguyen
Banking University of Ho Chi Minh City, 56 Hoang Dieu 2, Quan Thu Duc, Thu Duc
Ho Chi Minh City, Vietnam, e-mail: Thachnn@buh.edu.vn

Olga Kosheleva and Vladik Kreinovich
University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA
e-mail: olgak@utep.edu, vladik@utep.edu

Hoang Phuong Nguyen
Division Informatics, Math-Informatics Faculty, Thang Long University, Nghiem Xuan Yem Road
Hoang Mai District, Hanoi, Vietnam, e-mail: nhphuong2008@gmail.com

- for a discrete distribution, pdf is not defined,
- for a distribution with heavy tails, moments are sometimes infinite, etc.

Out of the known representations, the representation as a cdf is the most universal, it does not seem to have limitations. In view of this, to take into account that in econometrics, one can encounter discrete distributions (for which no pdf is known), heavy-tailed distributions (for which moments are infinite), etc., it is reasonable to use a cdf

$$F_X(x) = \text{Prob}(X \le x)$$

to describe a random variable X.

Similarly, to describe a joint distribution of two random variables (X,Y), it is reasonable to use a joint cdf

$$F_{XY}(x,y) = \text{Prob}(X \le x \,\&\, Y \le y).$$

When random variables X and Y are independent, we have $F_{XY}(x,y) = F_X(x) \cdot F_Y(y)$. In general, the dependence may be more complicated. It is reasonable to describe this dependence by a function $C(u,v)$ for which

$$F_{XY}(x,y) = C(F_X(x), F_Y(y)). \tag{1}$$

A function with this property is known as a *copula*; see, e.g., [3, 6, 7, 8]. Copulas have been successful used in many application areas, in particular, in econometrics.

Existence and uniqueness of copulas. It has been proven that such a copula always exists, and that the copula function $C(u,v)$ is itself a 2-D cdf on the square

$$[0,1] \times [0,1].$$

In situations when the distributions of X and Y are continuous – e.g., when there exists pdf's – the copula is uniquely determined. Indeed, in this case, the value $F_X(x)$ continuously depends on x and thus, attains all possible values between 0 and 1. So, to find $C(u,v)$, it is sufficient to find the values x and y for which $F_X(x) = u$ and $F_Y(y) = v$, then $F_{XY}(x,y)$ will give is the desired value of $C(u,v)$.

However, if the distribution of one of the variables – e.g., X – is discrete (for example, there are some values which have positive probabilities), then the value $F_X(x)$ jumps, and for thus, for some intermediate values u, we do not have values x for which $F_X(x) = u$. In such situations, the copula is *not* uniquely determined, since we can have different values $C(u,v)$ for this jumped-over u.

Subcopulas: reminder. While the copula is not always unique, there is a variant of this notion which is always unique; this variant is known as a *subcopula*. In precise terms, a subcopula is also defined by the formula (1), the only difference is that:

- while a copula has to be defined for all possible values $u \in [0,1]$ and $v \in [0,1]$,
- a subcopula $C(u,v)$ is only defined for the values u and v which have the form $u = F_X(x)$ and $v = F_Y(y)$ for some x and y.

Subcopulas have also been successfully used in econometrics; see, e.g., [9, 11, 13, 14, 17].

Main question: do we need subcopulas? From the purely mathematical viewpoint, it may seem that do not need subcopulas, since every subcopula can be, in principle, extended to a copula.

However, the fact that many researchers use subcopulas seem to indicate that, from the algorithmic viewpoint, subcopulas may not be easy to extend to copulas. An indirect argument in support of this not-easiness is that known extension proofs use non-constructive arguments such as Zorn's Lemma (which is equivalent to a non-constructive Axiom of Choice); see, e.g., [2].

What we do in this paper. In this paper, we prove that indeed, in situations of non-uniqueness, it is not algorithmically possible to always construct a copula. In other words, we prove that, from the algorithmic viewpoint, subcopulas are indeed needed.

2 What Is Computable: A Brief Reminder

What is computable: main definitions. In order to analyze when a copula is computable and when it is not, let us recall the main definitions of computability; for details, see, e.g., [1, 4, 15] (for random variables, see also [5]).

A real number x is *computable* if we can compute it with any given accuracy. In other words, a number is computable if there exists an algorithm that, given an integer n (describing the accuracy), returns a rational number r_n for which

$$|x - r_n| \le 2^{-n}.$$

Intuitively, a function $f(x)$ is computable if there is an algorithm that, given x, returns the value $f(x)$. In precise terms, this means that for any desired accuracy n, we can compute a rational number r_n for which $|f(x) - r_n| \le 2^{-n}$; in this computation, the program can pick some integer m and ask for an 2^{-m}-approximation to the input.

Similarly, a function $f(x,y)$ of two variables is called computable if, given x and y, it can compute the value $f(x,y)$ with any given accuracy. Again, in the process of computations, this program can pick some m and ask for a 2^{-m}-approximation to x and to y.

Comment. These definitions describe the usual understanding of computability; so, not surprisingly, all usual computable functions – e.g., all elementary functions, all continuous functions – are computable in this sense as well.

What is not computable. What is not computable in this sense are discontinuous functions such as $\mathrm{sign}(x)$ which is equal:

- to -1 when $x < 0$,

- to 0 when $x = 0$, and
- to 1 when $x > 0$.

Indeed, if this function was computable, then we would be able to check whether a computable real number is equal to 0 or not, and such checking is not algorithmically possible; see, e.g., [4] and references therein.

Indeed, the possibility of such checking contradicts to the known result that it is not possible,

- given a program,
- to check whether this program halts or not.

Indeed, based on each program, we can form a sequence r_n each element of which is:

- equal to 2^{-n} if the program did not yet halt by time n and
- equal to 2^{-t} if it halted at time $t \le n$.

Then:

- If the program does not halt, this sequence describes the computable real number

$$x = 0.$$

- If the program halts at time t, this sequence describes the computable real number

$$x = 2^{-t} > 0.$$

Thus, if we could check whether a real number is equal to 0 or not, we would be able to check whether a program halts or not – and we know that this is not algorithmically possible.

What does it mean for the cdf to be computable. In real life, when we say that we have a random variable, it means that we have a potentially infinite sequence of observations which follow the corresponding distribution. Based on these observations, for each computable real number x, we would like to compute the value $F(x)$.

The value $F(x)$ is the probability that the value of a random variable X is $\le x$. A natural practical way to estimate a probability based on a finite sample is to estimate the frequency of the corresponding event. Thus, to estimate $F(x)$, a natural idea is to take n observations $X_1, \ldots, X_n$, find out how many of them are $\le x$, and then compute the desired frequency by dividing the result of the counting by n.

Even in the ideal case, when all the values X_i are measured exactly, the frequency is, in general, different from the probability. It is known (see, e.g., [12]) that for large n, the difference between the frequency f and the probability p is approximately normally distributed, with 0 means and standard deviation

$$\sigma = \sqrt{\frac{p \cdot (1-p)}{n}} \le \frac{0.5}{\sqrt{n}}.$$

From the practical viewpoint, any deviation larger than 6 sigma has a probability of less than 10^{-8} and is, thus, usually considered practically impossible. (If you do not view 6 sigma as impossible, take 20 sigma; one can always come up with a probability so small that it is practically impossible.) Thus, if for a given $\varepsilon > 0$, we select n so large that $6\sigma \le 6 \cdot \dfrac{0.5}{\sqrt{n}} \le \varepsilon$ Then, the resulting frequency f is guarantee to be ε-close to the desired probability $F(x)$: $|f - F(x)| \le \varepsilon$, i.e., equivalently,

$$F(x) - \varepsilon \le f \le F(x) + \varepsilon.$$

In practice, we also need to take into account that the values X_i can only be measured with a certain accuracy δ; see, e.g., [10]. Thus, what we compare with the given number x are not the actual values X_i but the results $\widetilde{X}_i$ of their measurement which are δ-close to X_i:

- If $\widetilde{X}_i \le x$, we cannot conclude that $X_i \le x$, we can only conclude that $X_i \le x + \delta$.
- Similarly, if $\widetilde{X}_i > x$, we cannot conclude that $X_i > x$, we can only conclude that

$$X_i > x - \delta.$$

Thus, the only think that we can guarantee for the observed frequency f is that

$$F(x - \delta) - \varepsilon \le f \le F(x + \delta) + \varepsilon. \tag{2}$$

This is how a computable cdf is defined: that, given every a computable number x and rational numbers $\varepsilon > 0$ and $\delta > 0$, we can efficiently find a rational number f that satisfies the inequality (2).

A similar inequality

$$F(x - \delta, y - \delta) - \varepsilon \le f \le F(x + \delta, y + \delta) + \varepsilon \tag{3}$$

defines a computable 2-D cdf.

Comment. Note that a cdf can be discontinuous – e.g., if we have a random variable that is equal to 0 with probability 1, then:

- $F(x) = 0$ for $x < 0$ and
- $F(x) = 1$ for $x \ge 0$.

We already know such a function cannot be computable, so a computable cdf is not necessarily a computable function.

However, as we will see, when the computable cdf is continuous, it *is* a computable function.

Proposition 1. *When the distributions are continuous, a computable cdf is a computable function.*

Proof. Indeed, the inequalities (2) can be rewritten as follows:

$$f_{\delta,\varepsilon}(x - \delta) - \varepsilon \le F(x) \le f_{\delta,\varepsilon}(x + \delta), \tag{3}$$

where $f_{\delta,\varepsilon}(x)$ means a frequency estimated by comparing the measured values $\widetilde{X}_i$ (measured with accuracy δ) with the value x, based on a sample large enough to guarantee the accuracy ε.

Also, due to (2), we have

$$F(x-2\delta)-\varepsilon \le f_{\delta,\varepsilon}(x-\delta) \text{ and } f_{\delta,\varepsilon}(x+\delta) \le F(x+2\delta)+\varepsilon,$$

thus

$$F(x-2\delta)-2\varepsilon \le f_{\delta,\varepsilon}(x-\delta)-\varepsilon \le F(x) \le f_{\delta,\varepsilon}(x+\delta)+\varepsilon \le F(x+2\delta)+2\varepsilon. \quad (4)$$

When the cdf $F(x)$ is a continuous function, then, for each x, the difference

$$F(x+2\delta)-F(x-2\delta)$$

tends to 0 as δ decreases. Thus, the difference between the values $F(x+2\delta)+2\varepsilon$ and $F(x-2\delta)-2\varepsilon$ also tends to 0 as $\delta \to 0$ and $\varepsilon \to 0$. Thus, if we take $\delta = \varepsilon = 2^{-k}$ for $k = 1,2,\ldots$, we will eventually encounter an integer k for which this difference is smaller than a given number 2^{-n}. In this case, due to (4), the difference between the inner bounds $f_{\delta,\varepsilon}(x+\delta)+\varepsilon$ and $f_{\delta,\varepsilon}(x-\delta)-\varepsilon$ is also $\le 2^{-n}$. In this case, each of these bounds can be used as the desired 2^{-n}-approximation to $F(x)$.

Thus, to compute $F(x)$ with accuracy 2^{-n}, it is sufficient to compute, for $k = 1,2,\ldots$,

- values $\varepsilon = \delta = 2^{-k}$ and then
- values $f_{\delta,\varepsilon}(x+\delta)+\varepsilon$ and $f_{\delta,\varepsilon}(x-\delta)-\varepsilon$.

We continue these computations for larger and larger k until the difference between $f_{\delta,\varepsilon}(x+\delta)+\varepsilon$ and $f_{\delta,\varepsilon}(x-\delta)-\varepsilon$ becomes smaller than or equal to 2^{-n}.

Once this condition is satisfied, we return $f_{\delta,\varepsilon}(x+\delta)+\varepsilon$ as the desired 2^{-n}-approximation to $F(x)$.

The proposition is proven.

Comment. In the 2-D case, we can use a similar proof.

3 Main Results and Their Proofs

Proposition 2. *There exists an algorithm that, given a continuous computable cdf $F_{XY}(x,y)$, generates the corresponding copula – i.e., generates a computable cdf $C(u,v)$ that satisfies the formula (1).*

Proposition 3. *No general algorithm is possible that, given a computable cdf $F_{XY}(x,y)$, would generate the corresponding copula – i.e., that would generates a computable cdf $C(u,v)$ that satisfies the formula (1).*

Comment. This result proves that from the algorithmic viewpoint, it is, in general, not possible to always generate a copula. Thus, from the algorithmic viewpoint, subcopulas are indeed needed.

Proof of Proposition 2. This proof is reasonably straightforward, it follows the above idea of finding the copula for a continuous cdf.

Indeed:

- suppose that we are given two computable numbers $u, v \in [0,1]$, and
- we want to find the desired approximation to the value $C(u,v)$.

To do that, we first find x for which $F_X(x)$ is δ-close to u.

This value can be found as follows. First, we pick any x_0 and compute $F_X(x_0)$ with accuracy δ; we can do it, since, according to Proposition 1, for continuous distributions, the cdf is a computable function. If we get a value which is δ-close to u, we are done.

If the approximate value $F_X(x_0)$ is larger than u, we take $x_0 - 1$, $x_0 - 2$, etc., until we find a new value x_- for which $F_X(x_-) < u$.

Similarly, if the approximate value $F_X(x_0)$ is smaller than u, we take $x_0 + 1$, $x_0 + 2$, etc., until we find a new value x_+ for which $F_X(x_+) > u$.

In both cases, we have an interval $[x_-, x_+]$ for which $F(x_-) < u < F(x_+)$. Now, we can use bisection to find the desired x: namely, we take a midpoint x_m of the interval. Then:

- If $|F_X(m) - u| \le \delta$, we are done.
- If this ideal inequality is not satisfied, then we have:
  - either $F_X(x_m) < u$
  - or $F_X(x_m) > u$.
- In the first case, we know that the desired value x is in the half-size interval

$$[x_m, x_+].$$

- In the second case, we know that the desired value x is in the half-size interval

$$[x_-, x_m].$$

- In both cases, we get a new half-size interval.

To the new interval, we apply the same procedure until we get the desired x.

Similarly, we can compute y for which $F_Y(y) \approx v$. Now, we can take the approximation to $F_{XY}(x,y)$ as the desired approximation to $C(u,v)$.

The proposition is proven.

Proof of Proposition 3. For each real number a for which $|a| \le 0.5$, we can form the following probability distribution $F_a(x,y)$ on the square $[0,1] \times [0,1]$: it is uniformly distributed on a straight line segment $y = 0.5 + \mathrm{sign}(a) \cdot (x - 0.5)$ corresponding to $x \in [0.5 - |a|, x + |a|]$. Thus:

- when $a > 0$, we take $y = x$; and

- when $a < 0$, we take $y = 1 - x$.

One can easily check that $F_a(x,y)$ is indeed a computable cdf – although it is *not* always a computable function, since, e.g., for $a = 0$ the whole probability distribution is concentrated at the point $(0.5, 0.5)$.

For each a, the marginal distribution $F_X(x)$ is uniformly distributed on the interval $[0.5 - |a|, 0.5 + |a|]$ of length $2|a|$. Thus, for the values x from this interval, we have

$$F_X(x) = \frac{x - (0.5 - |a|)}{2|a|}.$$

So for any $u \in [0,1]$, to get $F_X(x) = u$, we must take

$$x = 0.5 - |a| + 2u \cdot |a| = 0.5 - (1 - 2u) \cdot |a|.$$

Similarly, to get the value y for which $F_Y(y) = v$, we should take

$$y = 0.5 - (1 - 2v) \cdot |a|.$$

For $u, v \le 0.5$, we get $x \le 0.5$ and $y \le 0.5$.

In particular, for $u = v = 0.25$, we should take $x = y = 0.5 - 0.5|a|$, and for $u = v = 0.5$, we should take $x = y = 0.5$.

Since the distribution is symmetric, when $u = v$, we have the same values $x = y$ for which $F_X(x) = u$ and $F_Y(y) = u$.

- When $a > 0$, we have $X = Y$. Thus for every $u = v$, we have

  $$C(u,u) = F_{XY}(x,x) = \text{Prob}(X \le x \,\&\, Y \le x) = \text{Prob}(X \le x) = u,$$

  i.e., $C(u,u) = u$. In particular, we have

  $$C(0.25, 0.25) = 0.25 \text{ and } C(0.5, 0.5) = 0.5.$$

- When $a < 0$, then $Y = 1 - X$. Thus, when $X \le 0.5$, we have $Y \ge 0.5$, so when $u, v \le 0.5$, we cannot have both $X \le u$ and $Y \le v$, and thus, we get

  $$C(u,u) = F_{XY}(x,x) = \text{Prob}(X \le x \,\&\, Y \le x) = 0.$$

  In particular, we have $C(0.25, 0.25) = C(0.5, 0.5) = 0$.

If it was possible, given a computable real number a, to compute a computable cdf $C(u,v)$, then, by definition of a computable cdf, we would be able to compute:

- given a,
- the value $f_{\delta,\varepsilon}(x)$ corresponding to $x = 0.375$, $\delta = 0.125$ and $\varepsilon = 0.1$,

i.e., the value $f \stackrel{\text{def}}{=} f_{0.125,0.1}(0.375)$ for which

$$C(x - \delta, x - \delta) - \varepsilon \le f \le C(x + \delta, x + \delta) + \varepsilon,$$

i.e., for which

$$C(0.25,0.25)-0.1 \le f \le C(0.5,0.5)+0.1.$$

Here:

- when $a<0$, we have $C(0.5,0.5)=0$, hence $f \le 0.1$ and therefore $f<0.125$;
- when $a>0$, then $C(0.25,0.25)=0.25$, hence $f \ge 0.15$ and therefore $f>0.125$.

So, by comparing the resulting value f with 0.125, we will able to check whether $a>0$ or $a<0$ – and this is known to be algorithmically impossible; see, e.g., [1, 4, 15]. This contradiction shows that it is indeed not possible to have an algorithm that always computes the copula.

The proposition is proven.

Acknowledgments

This work was supported in part by the US National Science Foundation via grant HRD-1242122 (Cyber-ShARE Center of Excellence).

The authors are thankful to Professor Hung T. Nguyen for valuable discussions.

References

1. E. Bishop and D. Bridges, *Constructive Analysis*, Springer Verlag, Heidelberg, 1985.
2. J. Fernández-Sánchez and M. Úbeda-Flores, "Proving Sklar's Theorem via Zorn's Lemma", *International Journal of Uncertainty, Fuzziness, and Knowlegde-Based Systems*, 2018, Vol. 26, No. 1, pp. 81–85.
3. P. Jaworski, F. Durante, W. K. Härdle, and T. Rychlik (eds.), *Copula Theory and Its Applications*, Springer Verlag, Berlin, Heidelberg, New York, 2010.
4. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1997.
5. V. Kreinovich, A. Pownuk, and O. Kosheleva, "Combining Interval and Probabilistic Uncertainty: What Is Computable?", in: P. Pardalos, A. Zhigljavsky, and J. Zilinskas (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Verlag, Cham, Switzerland, 2016, pp. 13–32.
6. J.-F. Mai and M. Scherer, *Simulating Copulas: Stochastic Models, Sampling Algorithms, and Applications*, World Scientific, Singapore, 2017.
7. A. J. McNeil, R. Frey, and P. Embrechts, *Quantitative Risk Management: Concepts, Techniques, and Tools*, Princeton University Press, Princeton, New Jersey, 2015.
8. R. B. Nelsen, *An Introduction to Copulas*, Springer Verlag, Berlin, Heidelberg, New York, 2007.
9. O. Okhrin, Y. Okhrin, and W. Schmidt, "On the structure and estimation of hierarchical Archimedean copulas", *Journal of Econometrics*, 2013, Vol. 173, pp. 189–204.
10. S. G. Rabinovich, *Measurement Errors and Uncertainty: Theory and Practice*, Springer Verlag, Berlin, 2005.
11. M. Ruppert, *Contributions to Static and Time-varying Copula-based Modeling of Multivariate Association*, EUL Verlag, Koeln, Germany, 2012.

12. D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.
13. Z. Wei, T. Wang, and P. A. Nguyen, "Multivariate dependence concepts through copulas", *International Journal of Approximate Reasoning*, 2015, Vol. 65, No. 1, pp. 24–33.
14. Z. Wei, T. Wang, and W. Panichkitkosolkul, "Dependence and Association Concepts through Copulas", In: V.-N. Huynh, V. Kreinovich, and S. Sriboonchitta (eds), *Modeling Dependence in Econometrics*, Springer Verlag, Cham, Switzerland, 2014, pp. 113–126.
15. K. Weihrauch, *Computable Analysis*, Springer Verlag, Berlin, 2000.
16. Y. Zhang, M. Beer, and S. T. Quek, "Long-term performance assessment and design of offshore structures", *Computers and Structures*, 2015, Vol. 154, pp. 101–115.
17. X. Zhu, T. Wang, S. T. B. Choy, and K. Autchariyapanitkul, "Measures of Mutually Complete Dependence for Discrete Random Vectors", In: V. Kreinovich, S. Sriboonchitta, and N. Chakpitak (eds), *Predictive Econometrics and Big Data*, Springer Verlag, Cham, Switzerland, 2018, pp. 303–317.