Journal of Uncertain Systems
Vol.6, No.2, pp.118-121, 2012
Online at: www.jus.org.uk

I-complexity and Discrete Derivative of Logarithms: A Symmetry-based Explanation

Vladik Kreinovich*, Jaime Nava
Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA

Received 1 June 2011; Revised 14 July 2011

Abstract

In many practical applications, it is useful to consider Kolmogorov complexity $K(s)$ of a given string s, i.e., the shortest length of a program that generates this string. Since Kolmogorov complexity is, in general, not computable, it is necessary to use computable approximations $\widetilde{K}(s)$ to $K(s)$. Usually, to describe such an approximations, we take a compression algorithm and use the length of the compressed string as $\widetilde{K}(s)$. This approximation, however, is not perfect: e.g., for most compression algorithms, adding a single bit to the string s can drastically change the value $\widetilde{K}(s)$ – while the actual Kolmgorov complexity only changes slightly. To avoid this problem, V. Becher and P. A. Heiber proposed a new approximation called I-complexity. The formulas for this approximation depend on selecting an appropriate function $F(x)$. Empirically, the function $F(x) = \log(x)$ works the best. In this paper, we show that this empirical fact can be explained if we take in account the corresponding symmetries.

Keywords: Kolmogorov complexity, I-complexity, symmetries

1 Formulation of the Problem

Kolmogorov complexity. Kolmogorov complexity $K(s)$ of a string s is defined as the shortest length of a program that computes s; see, e.g. [3]. This notion is useful in many applications. For example, a sequence is random if and only if its Kolmogorov complexity is close to its length.

Another example is that we can check how close are two DNA sequences s and s' by comparing $K(ss')$ with $K(s) + K(s')$:

- if s and s' are unrelated, then the only way to generate ss' is to generate s and then generate s', so $K(ss') \approx K(s) + K(s')$; but
- if s and s' are related, then we have $K(ss') \ll K(s) + K(s')$.

Need for computable approximations to Kolmogorov complexity. The big problem is that the Kolmogorov complexity is, in general, not algorithmically computable [3]. Thus, it is desirable to come up with computable approximations to $K(s)$.

Usual approaches to approximating Kolmogorov complexity: description and limitations. At present, most algorithms for approximating $K(s)$ use some loss-less compression technique to compress s, and take the length $\widetilde{K}(s)$ of the compression as the desired approximation.

This approximation has limitations. For example, in contrast to $K(s)$, where a small (one-bit) change in x cannot change $K(s)$ much, a small change in s can lead to a drastic change in $\widetilde{K}(s)$.

The general notion of I-complexity. To overcome this limitation, Becher and Heiber proposed the following new notion of *I-complexity* [1, 2]. For each position i of the string $s = (s_1 s_2 \ldots s_n)$, we first find the largest $B_s[i]$ of the lengths ℓ of all strings $s_{i-\ell+1} \ldots s_i$ which are substrings of the sequence $s_1 \ldots s_{i-1}$.

Then, we define $I(s) \stackrel{\text{def}}{=} \sum\limits_{i=1}^{n} f(B_s[i])$, for an appropriate decreasing function $f(x)$.

*Corresponding author. Email: vladik@utep.edu (V. Kreinovich).

Example. For example, for *aaaab*, the corresponding values of $B_s(i)$ are 01230. Indeed:

- For $i = 1$, the sequence $s_1 \dots s_{i-1}$ is empty, so $B_s(1) = 0$.
- For $i = 2$, with $s_1 s_2 = aa$, a string $s_2 = a$ is a substring of length 1 of the sequence $s_1 \dots s_{i-1} = s_1 = a$. So, here, $B_s(2) = 1$.
- For $i = 3$, with $s_1 s_2 s_3 = aaa$, a string $s_2 s_3 = aa$ is a substring of length 2 of the sequence $s_1 \dots s_{i-1} = s_1 s_2 = aa$. So, here, $B_s(3) = 2$.
- For $i = 4$, with $s_1 s_2 s_3 s_4 = aaaa$, a string $s_2 s_3 s_4 = aaa$ is a substring of length 3 of the sequence $s_1 \dots s_{i-1} = s_1 s_2 s_3 = aaa$. So, here, $B_s(4) = 3$.
- For $i = 5$, none of the strings $s_{i-\ell+1} \dots s_i$ ending with $s_i = s_4 = b$ is a substring of the sequence $s_1 \dots s_{i-1} = s_1 s_2 s_3 s_4 = aaaa$. So, here, $B_s(5) = 0$.

Good properties of I-complexity. Thus defined I-complexity has many properties which are similar to the properties of the original Kolmogorov complexity $K(s)$:

- If a string s starts with a substring s', then $I(s) \le I(s')$.
- We have $I(0s) \approx I(s)$ and $I(1s) \approx I(s)$.
- We have $I(ss') \le I(s) + I(s')$.
- Most strings have high I-complexity.

On the other hand, in contrast to non-computable Kolmogorov complexity $K(s)$, I-complexity can be computed feasibly: namely, it can be computed in linear time.

Empirical fact. Which function $f(x)$ should we choose? It turns out that the following *discrete derivative of the logarithm* works the best: $f(x) = \mathrm{dlog}(x+1)$, where $\mathrm{dlog}(x) \stackrel{\text{def}}{=} \log(x+1) - \log(x)$.

Natural question. How can we explain this empirical fact?

2 Towards Precise Formulation of the Problem

Discrete derivatives. Each function $f(n)$ can be represented as the *discrete derivative* $F(n+1) - F(n)$ for an appropriate function $F(n)$: e.g., for $F(n) = \sum\limits_{i=1}^{n-1} f(i)$. In terms of the function $F(n)$, the above question takes the following form: what is the best choice of the function $F(n)$?

From a discrete problem to a continuous problem. The function $F(x)$ is only defined for integer values x – if we use bits to measure the length of the longest repeated substring. If we use bytes, then x can take rational values, e.g., 1 bit corresponds to 1/8 of a byte, etc. If we use Kilobytes to describe the length, we can use even smaller fractions. In view of this possibility to use different units for measuring length, let us consider the values $F(x)$ for arbitrary real lengths x.

Continuous quantities: general observation. In the continuous case, the numerical value of each quantity depends:

- on the choice of the measuring unit and
- on the choice of the starting point.

By changing them, we get a new value $x' = a \cdot x + b$.

Continuous dependencies: case of length x. In our case, x is the length of the input. For length x, the starting point 0 is fixed, so we only have re-scaling $x \to \overline{x} = a \cdot x$.

Natural requirement: the dependence should not change if we simply change the measuring unit. When we re-scale x to $\overline{x} = a \cdot x$, the value $y = F(x)$ changes, to $\overline{y} = F(a \cdot x)$. It is reasonable to require that the value $\overline{y}$ represent the same quantity, i.e., that it differs from y by a similar re-scaling: $\overline{y} = F(a \cdot x) = A(a) \cdot F(x) + B(a)$ for appropriate values $A(a)$ and $B(a)$.

Resulting precise formulation of the problem. Find all monotonic functions $F(x)$ for which there exist auxiliary functions $A(a)$ and $B(a)$ for which

$$F(a \cdot x) = A(a) \cdot F(x) + B(a)$$

for all x and a.

3 Main Result

Observation. One can easily check that if a function $F(x)$ satisfies the desired property, then, for every two real numbers $c_1 > 0$ and c_0, the function $\overline{F}(x) \stackrel{\text{def}}{=} c_1 \cdot F(x) + c_0$ also satisfies this property. We will thus say that the function $\overline{F}(x) = c_1 \cdot F(x) + c_0$ is *equivalent* to the original function $F(x)$.

Main result. *Every monotonic solution of the above functional equation is equivalent to* $\log(x)$ *or to* x^α.

Conclusion. So, symmetries do explain the selection of the function $F(x)$ for I-complexity.

Proof.

1°. Let us first prove that the desired function $F(x)$ is differentiable.

Indeed, it is known that every monotonic function is almost everywhere differentiable. Let $x_0 > 0$ be a point where the function $F(x)$ is differentiable. Then, for every x, by taking $a = x/x_0$, we conclude that $F(x)$ is differentiable at this point x as well.

2°. Let us now prove that the auxiliary functions $A(a)$ and $B(a)$ are also differentiable.

Indeed, let us pick any two real numbers $x_1 \neq x_2$. Then, for every a, we have $F(a \cdot x_1) = A(a) \cdot F(x_1) + B(a)$ and $F(a \cdot x_2) = A(a) \cdot F(x_2) + B(a)$. Thus, we get a system of two linear equations with two unknowns $A(a)$ and $B(a)$.

$$F(a \cdot x_1) = A(a) \cdot F(x_1) + B(a).$$

$$F(a \cdot x_2) = A(a) \cdot F(x_2) + B(a).$$

Based on the known formula (Cramer's rule) for solving such systems, we conclude that both $A(a)$ and $B(a)$ are linear combinations of differentiable functions $F(a \cdot x_1)$ and $F(a \cdot x_2)$. Hence, both functions $A(a)$ and $B(a)$ are differentiable.

3°. Now, we are ready to complete the proof.

Indeed, based on Parts 1 and 2 of this proof, we conclude that

$$F(a \cdot x) = A(a) \cdot F(x) + B(a)$$

for differentiable functions $F(x)$, $A(a)$, and $B(a)$. Differentiating both sides by a, we get

$$x \cdot F'(a \cdot x) = A'(a) \cdot F(x) + B'(a).$$

In particular, for $a = 1$, we get $x \cdot \dfrac{dF}{dx} = A \cdot F + B$, where $A \stackrel{\text{def}}{=} A'(1)$ and $B \stackrel{\text{def}}{=} B'(1)$. So, $\dfrac{dF}{A \cdot F + b} = \dfrac{dx}{x}$; now, we can integrate both sides.

Let us consider two possible cases: $A = 0$ and $A \neq 0$.

3.1°. When $A = 0$, we get $\dfrac{F(x)}{b} = \ln(x) + C$, so

$$F(x) = b \cdot \ln(x) + b \cdot C.$$

3.2°. When $A \neq 0$, for $\widetilde{F} \stackrel{\text{def}}{=} F + \dfrac{b}{A}$, we get $\dfrac{d\widetilde{F}}{A \cdot \widetilde{F}} = \dfrac{dx}{x}$, so $\dfrac{1}{A} \cdot \ln(\widetilde{F}(x)) = \ln(x) + C$, and $\ln(\widetilde{F}(x)) = A \cdot \ln(x) + A \cdot C$. Thus, $\widetilde{F}(x) = C_1 \cdot x^A$, where $C_1 \stackrel{\text{def}}{=} \exp(A \cdot C)$. Hence, $F(x) = \widetilde{F}(x) - \dfrac{b}{A} = C_1 \cdot x^A - \dfrac{b}{A}$.

The statement is proven.

Acknowledgments

This work was supported in part:

- by the National Science Foundation grants HRD-0734825 and DUE-0926721, and
- by Grant 1 T36 GM078000-01 from the National Institutes of Health.

References

[1] Becher, V., and P.A. Heiber, A better complexity of finite sequences, *Abstracts of the 8th International Conference on Computability and Complexity in Analysis CCA'2011 and 6th International Conference on Computability, Complexity, and Randomness CCR'2011*, Cape Town, South Africa, p.7, 2011.

[2] Becher, V., and P.A. Heiber, A linearly computable measure of string complexity, *Theoretical Computer Science*, to appear.

[3] Li, M., and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, Berlin, Heidelberg, New York, 2008.