CS 252, Lecture 4: Kolmogorov Complexity

Nicolas Resch

February 6, 2020

1 Introduction & Motivation

The goal of *Kolmogorov* (or *Kolmogorov-Chaitin* or *descriptive*) complexity is to develop a measure of the complexity or "randomness" of an object. We would furthermore like this measure to be intrinsic to the object in question. For concreteness, the objects we consider will be strings over a fixed alphabet, but this is no real loss of generality. Thus, just as computability/complexity theory aims to quantify the complexity of languages (i.e., sets of strings), Kolmogorov complexity aims to quantify the complexity of individual strings.

Consider the following strings:

$$A = 4444444444$$
$$B = 2718281828$$
$$C = 1756475382$$

Which string is the most random? From the point-of-view of probability, if every 10 digit string has the same likelihood of being sampled, the strings are all "equally" random: each is sampled with probability 10^{-10}.

However, intuitively, string A appears to be very simple, in the sense that it has a short description: it is simply the digit 4 repeated 10 times. Strings B and C, on the other hand, appear to be much more complicated; one might even be tempted to say that they appear "random". However, B actually also has a fairly simple description: it is the first 10 digits in the decimal expansion of Euler's number! C, on the other hand, is indeed the result of randomly sampling a 10 digit number.

This discussion suggests that the Kolmogorov complexity of a string should be somehow linked to the length of its shortest description. Even with this intuition in mind, coming up with a meaningful definition is quite subtle. For example, consider the so-called Richard-Berry paradox: we describe a natural number as "the least natural number that cannot be described in less then twenty words". If this number exists, then we have just described it with thirteen words, a contradiction to its definition. On the other hand, if such a number does not exist, that means that every number can be described in fewer than twenty words, which is equally absurd!

Thus, we need to be very careful about what we consider to be a valid description. Moreover, we would like the definition to be "fair" in the sense that it doesn't artificially favor certain objects with short

descriptions. With these motivations and caveats in mind, let us begin to develop the theory of Kolmogorov complexity.

2 Formal Theory

2.1 Description Methods

Abstractly, a *description method* is just a partial function $f : \Sigma^* \to \Sigma^*$ (with $\Sigma = \{0, 1\}$, say) mapping descriptions to objects. The complexity of string x under description method f is

$$C_f(x) := \min\{|p| : f(p) = x\} .$$

While this is a good start, we promised a measure of a string's complexity that depends only on x. However, $C_f(x)$ clearly depends on f! For this reason, $C_f(x)$ could arbitarily favor certain strings with short descriptions. As an example, define $f(0) = 011101110111000110011$ and $f(1x) = x$ for all other strings x. Under this description method, $C_f(011101110111000110011) = 1$ whereas $C_f(x) = 1 + |x|$ for all other strings x. To remedy this, we will need to settle upon a *universal* description method.

2.2 Universal Description Method

Our goal is to pick a description method that is the "best" in the sense that it gives the minimum description length to all strings. The following definition formalizes this.

Definition 1 (Minorization)**.** A partial function $f : \{0, 1\}^* \to \{0, 1\}^*$ *minorizes* a partial function $g : \{0, 1\}^* \to \{0, 1\}^*$ if there exists a constant c such that for all strings x,

$$C_f(x) \leq C_g(x) + c .$$

That is, there is at most a constant overhead in describing a string x with f versus describing x with g.

We would now like to choose a *minimal* universal description method U which minorizes every other potential description method, and then define a string's complexity to be $C_U(x)$.[1]

A priori, it is not clear that such a U should exist... And in fact, if we desire U to minorize *every* partial function g, then it does not! Here's a quick sketch of the justification: pick an infinite sequence of strings $x_1, x_2, \ldots$ with $p_1, p_2, \ldots$ such that $C_U(x_i) = |p_i|$ for all i and such that $|p_i|$ is increasing. Pick a subsequence $p_{j_1}, p_{j_2}, \ldots$ such that $|p_{j_i}| > 2|p_i|$ for every i, and define g as follows: $g(p_i) = f(p_{j_i})$ for all i, and g agrees with f everywhere else. Now, observe that

$$C_g(x_{j_i}) \leq |p_i| < |p_{j_i}|/2 = C_U(x_{j_i})/2 .$$

Thus, for infinitely many strings z, $C_g(z) < C_U(z)/2$, so U does not minorize g.

[1] Note that such a minimal U need not be unique, as two partial functions can minorize each other. But for the theory of Kolmogorov complexity, any such minimal U will suffice.

To remedy this, we restrict the class of valid description methods so that such a universal description method can emerge. The class of description methods will be all partial recursive functions, which are the set of partial functions for which there exists a Turing machine M which, on input p for which $f(p)$ is defined, halts with $f(p)$ written on the output tape. (Note that if $f(p)$ is not defined, then we make no promises about M's behavior: in fact, it could be that M never terminates!)

Theorem 2. *There exists a partial recursive function U that minorizes every partial recursive function.*

Remark. Before diving into the proof, we make a brief comment about encoding pairs of strings. One can encode the pair $\langle x, y\rangle$ with $2|x| + |y| + 2$ bits: repeat each bit of x twice, add the string 01 as a sentinel, and then write the string y. This is not the most efficient method of encoding pairs of strings, but for our purposes this will suffice.

Proof. The description method U uses the Universal Turing machine. On input $\langle M, w\rangle$ (where we view M as the description of a TM and w as an input), U simulates M on input w and then outputs whatever M outputs (assuming it halts). We have

$$C_U(x) = \min\{|\langle M, w\rangle| : \text{TM } M \text{ halts on input } w \text{ and outputs } x\} .$$

We now demonstrate that U minorizes every other partial recursive function. Given any such function g, let M be a TM that computes it. Let x be any string, and let p be a shortest string such that $M(p) = x$, i.e., M halts on input p with x on its output tape. Clearly, $U(\langle M, p\rangle) = x$, and moreover $|\langle M, p\rangle| = c_M + |p|$, where $c_M = 2|\langle M\rangle| + 2$: importantly, c_M, is independent of p. The proof is therefore concluded by noting

$$C_U(x) \le |\langle M, p\rangle| \le c_M + |p| = c_M + C_g(x) . \qquad \square$$

Using this function U, we can finally define the Kolmogorov complexity.

Definition 3 (Kolmogorov Complexity)**.** The *Kolmogorov complexity* of a string x is

$$K(x) := C_U(x) = \min\{|\langle M, w\rangle| : \text{TM } M \text{ halts on input } w \text{ and outputs } x\} .$$

We can use this function $K(x)$ to formally define what it means for a string to be "random".

Definition 4 (Incompressible strings)**.** A string x is *incompressible* or *Kolmogorov random* if $K(x) \ge |x|$.

Note that there exist incompressible strings of every length: there are 2^n binary strings of length n but only $2^n - 1$ binary strings of length strictly less than n.

2.3 Examples

1. The string 0^n has Kolmogorov complexity $\log n + O(1)$: we just need to specify n and a program which, on input n, outputs 0^n.

2. The first n (binary) digits of e also has Kolmogorov complexity $\log n + O(1)$, really for the same reason as above (use any program for approximating e via, say, its Taylor series).

3. Every string has $K(x) \leq |x| + O(1)$. (Justify!)

4. Do there exist strings x of length n with Kolmogorov complexity, say, $\sqrt{n}$? Yes! Take some $y \in \{0,1\}^{\sqrt{n}}$ with $K(y) \geq \sqrt{n}$. Then, we claim that $x = y0^{n-\sqrt{n}}$ has Kolmogorov complexity $\sqrt{n} \pm \Theta(1)$. If it were too much shorter than $\sqrt{n}$, then we could describe y by describing x and then removing the last $n - \sqrt{n}$ 0's. On the other hand, here's a short way to describe x: describe y and then add an additional $n - \sqrt{n}$ 0's. This argument can be generalized to other (sublinear) complexities.

2.4 Undecidability of determining if a string is incompressible

First, we note that the set of strings that are *not* incompressible is recursively enumerable: on input x, consider all possible pairs $\langle M, w\rangle$ of length less than $|x|$, simulate M on w, and accept x iff some M outputs x on input w. Nonetheless, we have:

Theorem 5. *It is undecidable whether a string is incompressible.*

Note that this implies the set of incompressible strings is *not* recursively enumerable.

Proof. We'll provide two proofs. The first is very slick, while the second provides a reduction from the Halting Problem.

1. Assume for a contradiction we have a program P which can decide if a string is incompressible. Consider the program P' which, on input n, runs P on all strings of length n in lexicographical order and, as soon as P says that a string is incompressible, output that string. This gives a description of the lexicographically first incompressible string of length n of length $|\langle P', \langle n\rangle\rangle| = \log n + O(1)$, contradicting the fact that its Kolmogorov complexity is at least n.

2. Let $H = \{\langle M\rangle : \text{TM } M \text{ halts on input } 0\}$, and as before assume we have a program P which decides whether a given string is incompressible. We can use P to decide whether $\langle M\rangle \in H$ as follows. Let $n = |\langle M\rangle|$, and use P to decide for each y of length $2n$ whether or not it is incompressible. For each y that is compressible, find some program P_y that prints it out for which $|\langle P_y\rangle| < 2n$. Let t_y be the number of steps P_y requires to print out y, and let $t = \max_y t_y$. Then, run M on input 0 for t steps: if it halts, output "$\langle M\rangle \in H$; otherwise, output "$\langle M\rangle \notin H$".

   Why does this work? It is clear that M does not halt on 0 then the above program will correctly say "$\langle M\rangle \notin H$". The only potential problem is that maybe M does halt on 0, but takes more than t steps to do so. So suppose for a contradiction that this is the case. Note that we can specify the running time of M on input 0 using only $n + O(1)$ bits, as we can just provide $\langle M\rangle$ plus a program that counts the number of steps. Thus, we can specify an upper bound t^* on the running time of all the P_y's using only $n + O(1)$ bits. In this way, we can specify the set of all *compressible* strings of length $2n$ using only $n + 2\log(2n) + O(1)$ bits: give a description of t^* and the number $2n$, and then describe a program which simulates all programs of length less than $2n$ for at most t steps, and then outputs whatever the programs output. But then we can use this description of the set of all compressible

strings to describe the lexicographically first incompressible string using only $n + 2\log(2n) + O(1)$ bits! This gives the desired contradiction, as for large enough n, $2n > n + 2\log(2n) + O(1)$.

Thus, we have described a program that decides H, the Halting Problem. This is impossible, implying that there can be no such program P deciding if a string is incompressible.

□

3 Application: Prime Numbers

Kolmogorov complexity can be used to prove the infinitude of primes.

Theorem 6 (Euclid c. 300 BC)**.** *There are infinitely many prime numbers.*

Proof. Suppose not. Let $p_1, p_2, \ldots, p_k$ be a list of all the primes for some integer k. Let $m \in \mathbb{N}$ be a natural number whose encoding in binary is incompressible,[2] and let $n = \lfloor \log_2 m \rfloor + 1$. Write m as a product of primes: for some non-negative integers $e_1, \ldots, e_k$,

$$m = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \ .$$

Note that we can describe m by providing $\langle e_1, e_2, \ldots, e_k \rangle$. We claim that this is a short description of m, contradicting the incompressibility of m. Note that each $e_i \leq \log_2 m \leq n$, so it takes only $\log_2 n$ bits to describe each e_i. Hence, we can descibe $\langle e_1, \ldots, e_k \rangle$ with at most $(2k-1)\log_2 n + 2(k-1) + O(1) = O(\log_2 n)$ bits. (Note that k is constant, independent of n.) But, we assumed that $K(m) \geq n$. Assuming n is large enough (which we may, as there are incompressible strings of every length), this is a contradiction. □

[2] As a minor comment, note that the encoding in binary of such an integer m must begin with a 1. Thus, we need an incompressible string which begins with a 1. However, it cannot be the case that every compressible string of a given length begins with a 0. Do you see why?