Math 197: Senior Thesis

Kolmogorov Complexity and Graphs

John Hearn

Complexity

When attempting to characterize the complexity of an object, a useful question to ask is: How much information does it contain? In other words, what is the shortest description we can give the object such that no information about that object is lost, that is, it can be accurately reproduced? For our purposes, a description is a program which outputs the object. Kolmogorov complexity is a measure of the information contained in such a description. Specifically, the Kolmogorov complexity of an object is the length (literally the number of 1s and 0s) of the shortest binary string that is sufficient to replicate it. Hence, we have only countably many describable objects.

The first objection one would likely raise at this point is that program length is dependent on language. For instance, some objects are more simply described using C++ than say FORTRAN. It turns out the difference in description length for an object programmed in different languages is bounded by an additive constant. We will use the language of Turing machines.

Definitions and Notation

Definition 1. By associating inputs and outputs, a Turing machine defines a partial function from n-tuples of integers onto the integers, with $n \geq 1$. We call such a function *partial recursive* or *computable*. If the Turing machine halts for all inputs, then the function computed is defined for all arguments and is called *total recursive*, or simply *recursive*.

For some countable set of objects, S, we can assume some standard enumeration where $x \in S$ is associated with a natural number $n(x)$. We want to know if there exists another specification for x more space efficient than n. That is, a method f is a partial function over naturals where $n(x) = f(p)$. It is convenient to think of p as a program and f as the programming language, compiler, and computer. We denote length by $l(p)$.

We say:

$$C_f(x) = \min\{l(p) : f(p) = n(x)\}$$

where p is the shortest program that generates x (with no input) with respect to some partial function f. We call $C_f(x)$ the *unconditional Kolmogorov complexity* with respect f. If no such p exists, we say $C_f(x) = \infty$.

If for all x in S, $C_f(x) \leq C_g(x) + c$, where c is a constant, we say method f *minorizes* method g, and f and g are *equivalent* if they minorize each other. Each $x \in S$ might rely on any of the distinct methods $f_1, f_2, \ldots, f_r$ for a minimal Kolmogorov complexity. By reserving the first $\log r$ bits of p to indicate which f_i is used for producing x from p we have a method f minorized by all f_i where $c \approx \log r$.

Definition 2. Let $\mathbf{C}$ be a subclass of the partial functions over $\mathbb{N}^+$. A function f is *universal* (or additively optimal) for $\mathbf{C}$ if it belongs to $\mathbf{C}$ and if for every function $g \in \mathbf{C}$ there is a constant $c_{f,g}$ s.t. $C_f(x) \leq C_g(x) + c_{f,g}$ for all x. Here $c_{f,g}$ depends on f and g but not x.

The definition for a class of two-variable functions is had by replacing occurrences of x above with $\langle x, y\rangle$.

We say additively optimal methods f, g of specifying objects in S are equivalent in the following way:

$$\left|C_f(x) - C_g(x)\right| \leq c_{f,g}$$

for all x, where $c_{f,g}$ is a constant depending only on f and g.

There is no universal partial function f for all programs p. However, there does exist a universal element in the class of partial *recursive* functions. This is a modest and rather natural restriction of our descriptions, as there would be little use in attempting to define the information content of the non-existent output of programs which do not halt. We thus consider the class of description methods $\{\phi : \phi$ is a partial recursive function$\}$. We use ϕ_0 to denote the universal description method, which gives us the following definition (2, 95-97).

Definition 3. Let x, y, p be natural numbers. Any partial recursive function ϕ, together with program p and input y, such that $\phi(\langle y, p\rangle) = x$, is a *description* of x. The *complexity* C_ϕ of x *conditional* to y is defined by

$$C_\phi(x|y) = \min\{l(p) : \phi(\langle y, p\rangle) = x\}$$

and $C_\phi(x|y) = \infty$ if there is no such p. We call p a program to compute x by ϕ, given y.

By selecting a fixed ϕ_0 as our reference function for C, we can drop the subscript to denote the *conditional Kolmogorov complexity* where $C(x|y) = C_{\phi_0}(x|y)$. Note the unconditional Kolmogorov complexity $C(x) = C(x|\epsilon)$.

Two Theorems

The Invariance Theorem and the Incompressibility Theorem form the basis for the whole study of Kolmogorov Complexity, and are sufficient for many important proofs.

Lemma 1. There is a universal partial recursive function.

This result from computability theory generalizes to the Invariance Theorem, which considers the complexity of an object x facilitated by an already specified object y. Recall that Kolmogorov complexity for arbitrarily many conditionals can be defined by recursive use of the bijective pairing function.

The Invariance Theorem 1. There is a universal partial recursive function ϕ_0 for the class of partial recursive functions to compute x given y. Formally this says that $C_{\phi_0}(x|y) \leq C_\phi(x|y) + c_\phi$ for all partial recursive functions ϕ and all x and y, where c_ϕ is a constant depending on ϕ but not x or y.

Notice that the universal description method may not give the shortest description for all x, but no other method gives a shorter description for more than finitely many cases. We also note a trivial upper bound given by the following lemmas (but omit the proofs for brevity).

Lemma 2. There is a constant c such that for all x and y

$$C(x) \leq l(x) + c \text{ and } C(x|y) \leq C(x) + c.$$

In the case of objects conditionally belonging to finite sets, we can offer an improved upper bound with the following lemma.

Lemma 3. Let $A \subset \mathbb{N} \times \mathbb{N}$ be recursively enumerable, and $y \in \mathbb{N}$. Suppose $Y = \{x : (x, y) \in A\}$ is finite. Then, for some constant c depending only on A, for all $x \in Y$, we have $C(x|y) \leq l(|Y|) + c$.

Now that we have established that a method exists for describing all but a finite number of x in S with maximal efficiency, what can we infer about those descriptions? Well, for each n there are 2^n binary strings of length n, but only $2^n - 1$ descriptions shorter than n. Thus there exists at least one binary string x of length n with $C(x) \geq n$. We then say x is *incompressible*.

Definition 4. For each constant c we say a string x is *c-incompressible* if $C(x) \geq l(x) - c$.

The Incompressibility Theorem 2. Let $c \in \mathbb{N}^+$. For each fixed y, every finite set A of cardinality m has at least $m(1 - 2^{-c}) + 1$ elements x with $C(x|y) \geq \log m - c$.

What we see by this theorem is the fairly surprising result that of all binary strings of length n, at least half of them can only be compressed by no more than one digit. Another quarter or more of the strings can only be compressed by at most 2 digits, and so on. This itself has some rather counter intuitive results.

For instance, if x is an incompressible string, are all substrings in x also incompressible? Intuitively, the ability to compress a substring would seem to give us a means to compress x. We can place a lower bound on substring v given by $C(v) \leq l(v) - O(\log n)$ but cannot prove $C(v) \leq l(v) - O(1)$. If the latter were true, x could contain no long regular subsequences since, for example, a sequence of k zeroes has complexity $O(\log k)$. But for strings of length n, only a small subset have no regular substrings, which gives us an easy way to describe them. Thus, for x to be incompressible, it *must* have compressible substrings (2, 112).

Graphs

Canonically, a graph $G = (V, E)$ with n vertices labeled $V = \{1, 2, \ldots, n\}$ is encoded as a $n(n-1)/2$ length string $E(G)$ where each bit corresponds lexicographically to a vertex pair. Thus $E(G) = e_{1,2}e_{1,3}\ldots e_{1,n}e_{2,3}e_{2,4}\ldots e_{n-1,n}$ where $e_{u,v} = 1$ if $(u, v) \in E$ and $e_{u,v} = 0$ otherwise. Thus by vertex relabeling we have $n!$ distinct strings, each encoding some member of an equivalence class of isomorphic graphs. However, the equivalence class has fewer than $n!$ members if there are automorphisms: multiple labelings that produce the same string.

Often we encounter problems on unlabeled graphs that depend only on the graph's structure. It would be gratifying to have our string encoding reflect the graph's complexity in some intuitive way. Kolmogorov Complexity is an attractive metric, as it would seemingly give us a measure based on the graph's compressibility. Unfortunately, the compressibility of the string $E(G)$ is clearly very dependent on our label permutation. Consider labeled graphs G_1 and G_2 (Fig. 1).

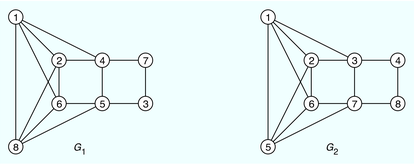

Figure 1: *Isomorphic graphs.*

Our labelings are $E(G_1) = 10101010101010101010101010$ and $E(G_2) = 11011001011001001000011101101$. While $E(G_2)$ may or may not be compressible, $E(G_1)$ is clearly *very* compressible and in all likelihood the most compressible string in the equivalence class.

We also find that graph regularity does not translate into string regularity. Consider the graph in Fig. 2, which contains two isomorphic subgraphs. While G_1 has some regularities and the same number of vertices as G_3, it has 4 more edges, is non-planar and is not visually broken easily into a set of isomorphic subgraphs. As far as our intuition is concerned, G_1 is more 'complex' than G_3, yet (presumably) G_1 has the lower Kolmogorov complexity. Apparently a graph with 'high' complexity can be encoded by a string with 'low' complexity.

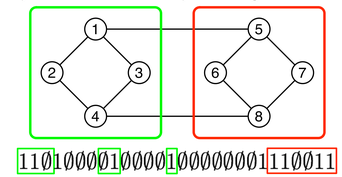

Figure 2: *Graph regularity.*

My research was motivated by a desire for an elegant resolution to this problem.

References

[1] Béla Bollobás, *Modern Graph Theory*, Graduate Texts in Mathematics, vol. 184, Springer-Verlag, New York, 1998.

[2] Ming Li and Paul Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, second ed., Graduate Texts in Computer Science, Springer-Verlag, New York, 1997.

Acknowledgments

I would like to thank my advisor, Prof. Ran Libeskind-Hadas for his support, assistance, and direction throughout my research. I would also like to thank Prof. Michael Orrison, Prof. Art Benjamin, and Prof. Henry Krieger for their input and assistance at various sticking points during the year, and Prof. Ming Li at the University of Waterloo for allowing me to take his course on Kolmogorov Complexity.

Advisor: Ran Libeskind-Hadas